In [1]:
```
import numpy as np
import pandas as pd
import matplotlib
from matplotlib import pyplot as plt
import plotly.express as px
import seaborn as sns

from sklearn.feature_selection import SelectKBest, f_classif
from sklearn.decomposition import PCA
from scipy.stats import chi2_contingency

from sklearn.preprocessing import OneHotEncoder
from sklearn.preprocessing import StandardScaler
```

In [6]:
```
df = pd.read_csv('C:\\Users\\thesk\\eclipse-workspace\\crowdsource2018_copy.csv', encoding = "ISO-8859-1
```

In [7]:
```
df.shape
```

Out[7]: (378661, 17)

In [8]:
```
df.head(2)
```

Out[8]:

	ID	name	category	main_category	currency	deadline	goal	launched	days_open	pledged	backers	country
0	1000002330	The Songs of Adelaide & Abullah	Poetry	Publishing	GBP	9/10/2015	1000.0	11/8/2015 12:12	58.49	0.0	0	GB
1	1000003930	Greeting From Earth ZGAC Arts Capsule For ET	Narrative Film	Film & Video	USD	1/11/2017	30000.0	2/9/2017 4:43	59.80	2421.0	15	US

Feature selection - correlation

The Pearson correlation heatmap: features with response

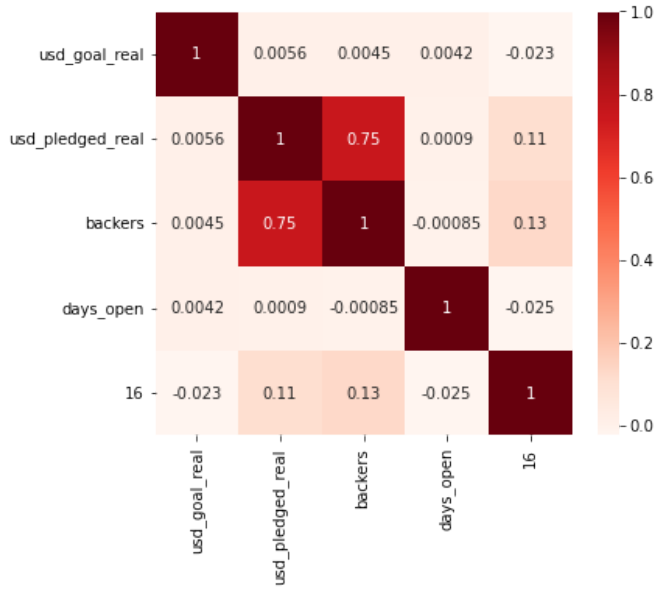
In [15]:
```
# select numeric features for the heatmap (no 'goal' & 'pledged' as numbers in various currencies depend
X_num = df[['usd_goal_real','usd_pledged_real','backers','days_open','stateNY']]

# replace the target Yes/No with 1/0 for calculating correlation
# 17 columns numebered 0 to 16, the last column is 16
X_num.loc[:,16] = X_num['stateNY'].apply(lambda x: 0 if x=='No' else 1)
```

In [16]:
```
X_num.head(2)
```

Out[16]:

	usd_goal_real	usd_pledged_real	backers	days_open	stateNY	16
0	1533.95	0.0	0	58.49	No	0
1	30000.00	2421.0	15	59.80	No	0

In [17]:
```
plt.figure(figsize=(6,5))
cor = X_num.corr()
sns.heatmap(cor, annot=True, cmap=plt.cm.Reds)
plt.show()
```



Correlations between variables and with the target variable are rather low. (At least, we don't have a nightmare for data analysts with many linearly related features present). We do have the target variable as a simple binary (0,1) variable but it is still valid for the Pearson correlation.

'Backers' correlate highly with the 'usd_pledged' which means that we can drop one of those variables so that predictors are not correlated. Keep 'Backers' as slightly more correlated with the target variable.

Feature selection - SelectKBest

SelectKBest uses the f_classif score function (or one-way ANOVA F-test that the means are not equal). It is possible to indicate other functions, e.g. Chi-square.

In [18]:
```
predictors = ['usd_goal_real','usd_pledged_real','backers','days_open']

s = SelectKBest(f_classif, k=2)
X_new = s.fit_transform(X_num[predictors], X_num['stateNY'])

slc_features = s.get_support()
print(slc_features)
s.scores_
```

```
[False  True  True False]
```

Out[18]:
```
array([ 195.5348079 , 4608.73571467, 6295.41468135,  232.52096809])
```

In [19]:
```
X_new.shape
```

Out[19]:
```
(378661, 2)
```

The middle two features, 'usd_pledged_real','backers' were chosen by sk-learn SelectKBest features. Large value means that the feature is non-randomly related to y, and so likely to provide important information. Only k features were retained.

For stats people, the popular nipals algorithm to reduce many features down to just a couple, is not available in sk-learn. It is present in sklearn.cross_decomposition but not for PCA purposes.

PCA

What are the most important features that can sufficiently explain the majority of the variance (squared deviations)?
PCA is a standard technique taught in most multivariate statistics classes and available in most statistical packages. PCA is affected by scale so we'll use StandardScaler to get the features into unit scale (mean = 0, std = 1) which is a requirement for the optimal performance of many machine learning algorithms. The goal is to

- deal with noisy data (garbage in, garbage out)
- speed up the model
- visualize data
- use only the principal components to transform unseen data.

What about categorical features? We can ignore discreteness and treat categorical features as Likert scales (cost: higher numbers have more weigth). Other options: the polychoric correlation matrix. Group means: of a truncated distribution for variable scores. Dummy/one hot encoding but features will have more weight in the PCA as after encoding one categorical feature will have n columns. For their total to have the same weight in the PCA as the original column, divide each new encoded column by the square root of its probability: sqrt(number of ones in the column / n).

SUMMARY:

- Numerical: (PCA)
- categorical (incl. cross-tabs): correspondence analysis (CA)
- only a few categorical variables and the rest numerical: PCA on numerical, and project the group means for the levels of the categorical variables as supplementary (unweighted) points
- more than 2 variables and they are all categorical: multiple correspondence analysis (MCA)
- many categorical or numerical variables: multiple factor analysis (MFA). The method applies PCA on numerical and MCA on categorical variables and combines the results by weighing variable groups.
- both categorical and numerical variables: use factor analysis of mixed data (FAMD)

FAMD: In real life, both categorical and numerical variables is a very common scenario. Two implementations are readily available from third parties:

- from prince import FAMD
- from lightfm import LightFM

In [22]:
```
# PCA
X = X_num.drop(['stateNY'],axis=1).values         # remove target feature 'stateNY'
y = X_num['stateNY'].values                       # mark 'stateNY' as y
```

In [23]:
```
X_scaled = StandardScaler().fit_transform(X)
```

In [24]:
```
pca = PCA(n_components=2)
components = pca.fit_transform(X_scaled)
X_comp = pd.DataFrame(data = components, columns = ['pc 1', 'pc 2'])
```

In [25]:
```
explained_variance = pca.explained_variance_ratio_
print(explained_variance)
# First PC explains 36% of variance in the target variable, second PC explains 20% of variance.
# Total explained by 2 components is ~56%.
```

```
[0.35767431 0.20472315]
```

In [26]:
```
# The components correspond to combinations of the original features
print(components)
```

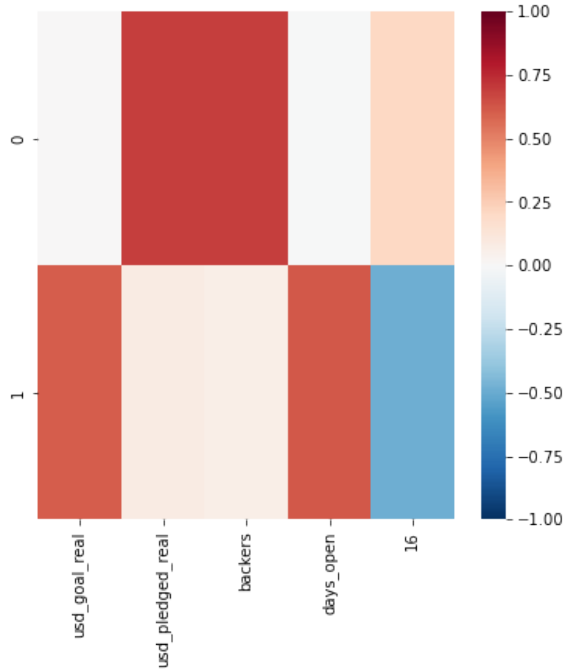
```
[[-0.30870432  0.55638355]
 [-0.27893809  0.5870009 ]
 [-0.3033068   0.45320481]
 ...
 [-0.306444    0.43881057]
 [-0.2997873   0.29928802]
 [-0.28870215  0.26897628]]
```

How do these components correlate with the actual features?

In [27]:
```
X = X_num.drop(['stateNY'],axis=1)
for col in X.columns:
    feature_names = list(X.columns)
```

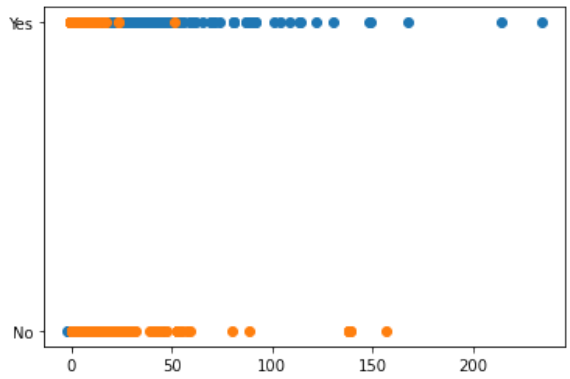
In [28]:
```
map= pd.DataFrame(pca.components_,columns=feature_names)
plt.figure(figsize=(6,6))
sns.heatmap(map,cmap='RdBu_r', vmin=-1, vmax=1)
```

Out[28]: <AxesSubplot:>



The goal and days_open correlate with one component and actual pledged USD and the number of backers correlate with the other component.

In [29]:
```
X_comp = pd.DataFrame(data = components, columns = ['pc 1', 'pc 2'])
X_comp_y = pd.concat([X_comp, df[['stateNY']]], axis = 1)  #can take numc_df here with 0/1; df has Yes/N

fig = plt.figure()
plt.scatter(X_comp_y['pc 1'], X_comp_y['stateNY'])
plt.scatter(X_comp_y['pc 2'], X_comp_y['stateNY'])
plt.show()
```



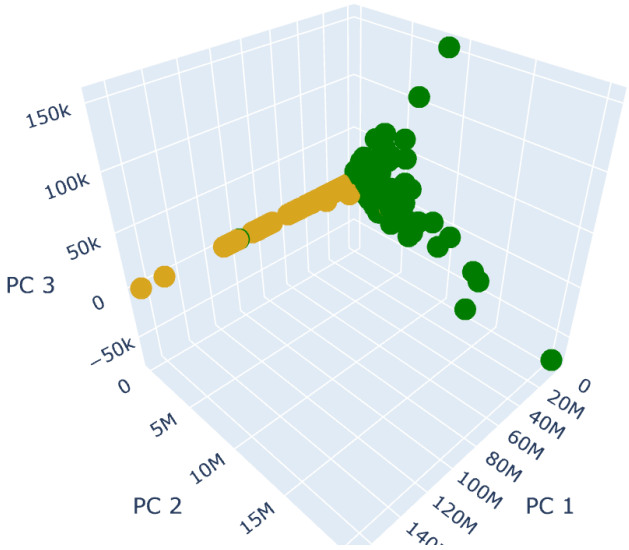
The first principal component (blue) separated funded (Yes) projects and
the second principal component (orange) separated non-funded (No) crowdsourcing projects.

In [33]:
```
# Trying 3 components
pca = PCA(n_components=3)
components = pca.fit_transform(X)

total_var = pca.explained_variance_ratio_.sum() * 100
total_var
```

```
C:\Users\thesk\anaconda3\lib\site-packages\sklearn\utils\validation.py:1675: FutureWarning:

Feature names only support names that are all strings. Got feature names with dtypes: ['int', 'str']. A
n error will be raised in 1.2.
```

Out[33]:
```
99.99999967518076
```

In [34]:
```
fig = px.scatter_3d(
    components, x=0, y=1, z=2, color=df['stateNY'],color_discrete_sequence=["goldenrod","green"],
    labels={'0': 'PC 1', '1': 'PC 2', '2': 'PC 3'})
fig.show()
```

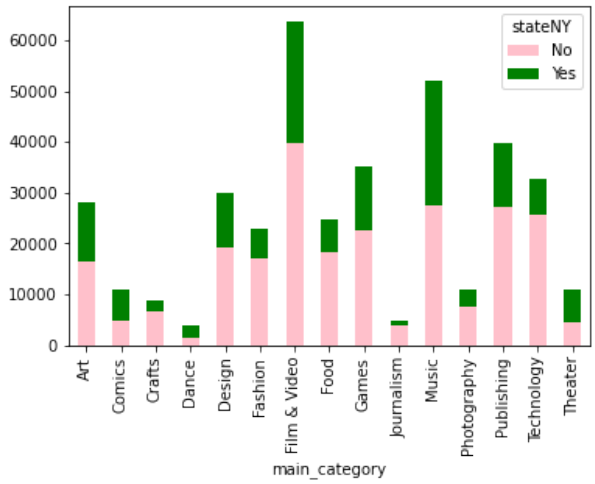


Separated just a bit more cleanly. (View the 3D image in NBViewer if not fully rendered on Github).

Back to categorical features. Let's check whether the main category makes a difference in the project being funded?

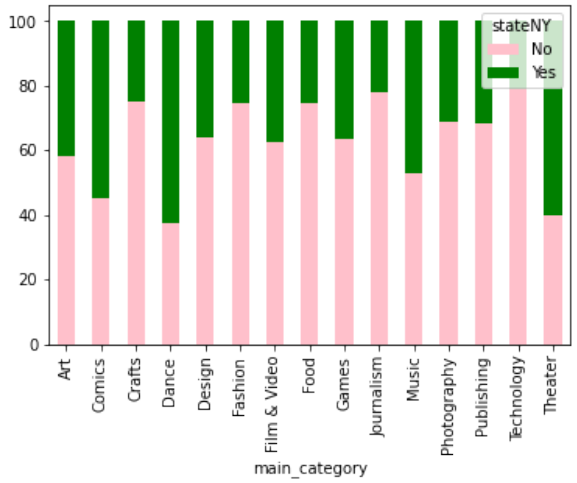
In [35]:
```
df.groupby('main_category').stateNY.value_counts().unstack().plot.bar(stacked = True, color=['pink', 'gr
```

Out[35]: <AxesSubplot:xlabel='main_category'>



In [36]:
```
df.groupby('main_category').stateNY.value_counts(normalize=True).mul(100).unstack().plot.bar(stacked = T
```

Out[36]: <AxesSubplot:xlabel='main_category'>



By normalizing the counts, we see a better visual representation of the proportion of funded projects in each category.
A very small proportion of TECH projects gets funded by crowsourcing and a large proportion of DANCE and THEATER projects.
Are these differences significant? We can use Chi-square test for testing our Null hypothesis that there is no significant difference in whether projects in either category are more likely to be funded. Chi-square test uses a contingency table (a cross-tab with the total counts in each column and row) to compare observed frequencies with those expected if frequencies came from the same population.

In [37]:
```
crostab = pd.crosstab(df.main_category, df.stateNY)
crostab
```

Out[37]:

stateNY	No	Yes
main_category		
Art	16449	11704
Comics	4901	5918
Crafts	6618	2191
Dance	1412	2356
Design	19215	10855
Fashion	16973	5843
Film & Video	39630	23955
Food	18333	6269
Games	22426	12805
Journalism	3712	1043
Music	27440	24478
Photography	7426	3353
Publishing	27275	12599
Technology	25758	6811
Theater	4338	6575

In [38]:
```
chi2, p, dof, expected = chi2_contingency(crostab.to_numpy())   # obs = crostab.to_numpy() as observatio
chi2, p, dof, expected                                          # 15 main categories, therefore dof=14
                                                                # expected: the expected distribution of
```

Out[38]:
```
(15610.551647483944,
 0.0,
 14,
 array([[17985.42659001, 10167.57340999],
        [ 6911.67301095,  3907.32698905],
        [ 5627.59289708,  3181.40710292],
        [ 2407.17107914,  1360.82892086],
        [19210.0940419 , 10859.9059581 ],
        [14575.90640705,  8240.09359295],
        [40621.01196057, 22963.98803943],
        [15716.8850555 ,  8885.1149445 ],
        [22507.17735917, 12723.82264083],
        [ 3037.71191118,  1717.28808882],
        [33167.59768764, 18750.40231236],
        [ 6886.11917784,  3892.88082216],
        [25473.33853764, 14400.66146236],
        [20806.56976557, 11762.43023443],
        [ 6971.72451876,  3941.27548124]]))
```

Chi-square statistic is 15610, significance = 0.0, a significant difference between categories.
It is possible to print post-hoc tests and multiple comparisons correction in Python but a SPSS visually clear output table is added for this case.

main_category

		Art (A)	Comics (B)	Crafts (C)	Dance (D)	Design (E)	Fashion (F)	Film & Video (G)	Food (H)	Games (I)	Journalism (J)	Music (K)	Photography (L)	Publishing (M)	Technology (N)	Theater (O)
stateNY	No	B(.000) D(.000) K(.000) O(.000)	D(.000) O(.000)	A(.000) B(.000) D(.000) E(.000) G(.000) I(.000) K(.000) L(.000) M(.000) O(.000)		A(.000) B(.000) D(.000) G(.000) K(.000) O(.000)	A(.000) B(.000) D(.000) E(.000) G(.000) I(.000) K(.000) L(.000) M(.000) O(.000)	A(.000) B(.000) D(.000) E(.000) G(.000) K(.000) O(.000)	A(.000) B(.000) D(.000) E(.000) G(.000) I(.000) K(.000) L(.000) M(.000) O(.000)	A(.000) B(.000) D(.000) G(.004) K(.000) O(.000)	A(.000) B(.000) C(.013) D(.000) E(.000) F(.000) G(.000) H(.000) I(.000) K(.000) L(.000) M(.000) O(.000)	B(.000) D(.000) O(.000)	A(.000) B(.000) D(.000) E(.000) G(.000) I(.000) K(.000) O(.000)	A(.000) B(.000) D(.000) E(.000) G(.000) I(.000) K(.000) O(.000)	A(.000) B(.000) C(.000) D(.000) E(.000) F(.000) G(.000) H(.000) I(.000) K(.000) L(.000) M(.000) O(.000)	
	Yes	C(.000) E(.000) F(.000) G(.000) H(.000) I(.000) J(.000) L(.000) M(.000) N(.000)	A(.000) C(.000) E(.000) F(.000) G(.000) H(.000) I(.000) J(.000) K(.000) L(.000) M(.000) N(.000)	J(.013) N(.000)	A(.000) B(.000) C(.000) E(.000) F(.000) G(.000) H(.000) I(.000) J(.000) K(.000) L(.000) M(.000) N(.000)	C(.000) F(.000) H(.000) J(.000) L(.000) M(.000) N(.000)	J(.000) N(.000)	C(.000) E(.000) F(.000) H(.000) I(.004) J(.000) L(.000) M(.000) N(.000)	J(.000) N(.000)	C(.000) F(.000) H(.000) J(.000) L(.000) M(.000) N(.000)		A(.000) C(.000) E(.000) F(.000) G(.000) H(.000) I(.000) J(.000) L(.000) M(.000) N(.000)	C(.000) F(.000) H(.000) J(.000) N(.000)	C(.000) F(.000) H(.000) J(.000) N(.000)		A(.000) B(.000) C(.000) E(.000) F(.000) G(.000) H(.000) I(.000) J(.000) K(.000) L(.000) M(.000) N(.000)

Results are based on two-sided tests. For each significant pair, the key of the category with the smaller column proportion appears in the category with the larger column proportion. Significance level for upper case letters (A, B, C): .05. Tests are adjusted for all pairwise comparisons within a row of each innermost subtable using the Bonferroni correction.

We can see that no categories have a smaller proportion of projects being funded than Journalism and Technology! The third difficult-to-get-crowdsourced category is Fashion.

As this entry focuses on dimension reduction, the main_category feature is a significant feature which does show differences in project funding and therefore should not be removed.