Art for Art's Sake?

> 'I love the gallery, the arena of representation. It's a commercial world, and morality is based generally around economics, and that's taking place in the art gallery.' Jeff Koons, American Business Artist
>
> 'I don't buy art in order to leave a mark or to be remembered; clutching at immortality is of zero interest to anyone sane.' Charles Saatchi, British Art Collector
>
> 'Making money is art and working is art and good business is the best art.' Andy Warhol, American Pop Artist

The Museum of Modern Art

Located in mid-town Manhattan, the Museum of Modern Art is an art museum on 53rd St between Fifth and Sixth Avenues. It is one of the largest and most influential art museums in the world, and its collection features an overview of modern and contemporary art. The collection features architecture and design, paintings, drawings, sculpture, photography, prints, illustrated books, artist's books and films and electronic media. It also includes an archive of ephemera relating to artists and groups, and an archive of primary source material relating to modern and contemporary art.

The gallery was developed in 1929 primarily by the wife of John D. Rockefeller Jr and two of her friends, Lillie P. Bliss and Mary Quinn Sullivan, releasing a press release that read 'The belief that New York needs a Museum of Modern Art scarcely requires apology' (https://www.moma.org/momaorg/shared//pdfs/docs/press_archives/1/releases/MOMA_1929-31_0001_1929-08.pdf?2010). Since that time, it has helped to cement the role of Picasso as a giant of modern art, changed the way that native american art is viewed. In 1929, it occupied a 12th floor rental in an office building in Fifth Avenue. Today,

Opinions have differed on the extent to which art and money are linked. In the medieval period and through to the Renaissance, there was no such thing as an 'artist'. Indeed, even the art historian Ernst Gombrich observed that 'there is no art, only artists'. Back then, the patron had control over the art they commissioned, with 'artists' struggling against these shackles.

During the recession in the early 1990s, the London contemporary art scene struggled, and artists began to put on their own shows. Invariably, commercial success of the artists behind would depend on the extent to which patrons would purchase collections. The Sensation exhibition at the Royal Academy in 1997 featured works exclusively by the collector Charles Saatchi.

Today, museums have to stand aside the line between being publicly oriented and catering to the taste of donors. Twenty five years ago, the largest 150 art institutions had a combined annual operating budget of less than USD1 billion. In 2000, the top 5 per cent of US visual art institutions controlled almost 80 per cent of combined revenue, endowments, infrastructure and donations. As of 2013, the MOMA's operating budget totalled USD 125m and its endowment had grown to USD 870 million, a number quite above pre-recession levels. By comparison, when the Whitney Museum announced its plans to build a much bigger institution in downtown Manhattan in 2010, its endowment was only USD 190m.

The fiscal health of a growing art institution is, for the most part, contingent on two sources of revenue: visitor dollars, which only accounts for a small percentage of a museum's revenue, and the larger piece of the pie: private funding. Fiscal success for a museum is tied to visitor numbers insofar as it is a sign to potential donors that the institution is a vital one. In the MOMA's previous large expansion, costing nearly USD 900m, this was primarily bankrolled by trustee donations and other charitable giving, the major source of funding for capital projects. The city contributed USD 65m. In other words, the MOMA's success has relied upon being both public and donor-friendly. But where does that line fall? Does the 'taste' of the collection fall on the public or donor side?

Our project does not seek to make value judgements. Simply to interrogate what factors are most important, based on a hypothesis, for presence in the collection of the MOMA.

Step 1: Identify the Problem

A. Identify and Hypothesise Goals and Criteria for Success

Given the issues at stake in our introduction, we wish to establish the extent to which we can predict the donor of an individual art work within the collection of the Museum of Modern Art.

We believe we can do this because:

* there is an established history of the taste of art patrons shaping the canon of western art
* the Museum of Modern Art is dependent heavily upon endowments for its expansion, and less so on public sources of funding
* there are sufficient features in the dataset to permit us to form a profile of donor tastes

Our prior hypothesis is that the donor has no effect upon the choice of art works within the Museum of Modern Art. Our alternative hypothesis is that the choice of art work does have a positive effect upon the choice of art in the collection of the MOMA.

According to conventions, a Bayes Factor of between 3 and 10 between prior and posterior suggests a significant result. Any Bayes Factor over 10 suggests early conclusion of the experiment may be warranted.

B. Create a Set of Questions for Identifying the Correct Data Set

We want a fine measure of the effect of donor taste on the collection. So the outputs of our project will be twofold:

* a prediction engine, that will use a classifier on training and test data sets of features of art works, to determine the extent to which a donor's influence explains its presence in the collection. If we can predict the presence of art works in the test set with considerable accuracy, we can demonstrate strong correlation between a donor's preferences and the presence of an art work in the collection
* Inferential statistical analysis

In order to do this, we need to obtain a dataset that:

* Provides a catalogue of a large sample (n > 100,000) of works within the collection of the Museum of Modern Art; and

* Provides documentary data for each artwork's medium, scale, derivation, biographical data as far as possible,
* Provides donor data for each artwork
* Covers a substantial universe (> 90 per cent) of the universe of the MOMA to ensure that it is a representative sample

In [2]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
color = sns.color_palette()
%matplotlib inline
import json
from __future__ import unicode_literals

## Load spacy

from spacy.en import English
nlp_toolkit = English()
nlp_toolkit
```

Out[2]:

```
<spacy.en.English at 0x1106637d0>
```

Step 2: Acquire the Data

A. Identify the 'Right' Datasets

As a starting point, the MOMA itself has provided a dataset that is free to download via Kaggle Datasets, with a fairly free rein to interrogate the dataset and obtain insights. Data analysis to-date within kernels has focused on the dimensions of art works, or on numbers of individual artists' works in the collection. In our exploratory data analysis, we will be covering some of these statistics in passing, but they will not be the primary focus of our analysis.

MoMA is committed to helping everyone understand, enjoy, and use its collection. The Museum's website features 72,706 artworks from 20,956 artists. The artworks dataset contains 130,262 records, representing all of the works that have been accessioned into MoMA's collection and cataloged in its database. It includes basic metadata for each work, including title, artist, date, medium, dimensions, and date acquired by the Museum. Some of these records have incomplete information and are noted as “not curator approved.” The artists dataset contains 15,091 records, representing all the artists who have work in MoMA's collection and have been cataloged in the MOMA database. It includes basic metadata for each artist, including name, nationality, gender, birth year, and death year.

The MOMA datasets satisfy our requirements in terms of length of the data and features, containing the following:

Data Dictionary

Feature	Description	Datatype

ArtworkID	Unique identifier for art work	Numeric
Title	Title of work	String
ArtistID	Unique identifier for artist	Numeric
Name	Name of the Artist	String
Date	Date of Artist work	Numeric
Medium	Medium of art work	String
Dimensions	Dimensions of the Art Work	String
Acquisition Date	Date work acquired	DateTime
Credit	Gift/Bequest/Donor information	String
Catalogue	Unknown	String
Department	MOMA Department responsible for the art work	String
Classification	Type of art work	String
Object Number	Unique object identifer (accession number)	Numeric
Diameter	Diameter of object in CM	String
Circumferance	Circumferance of object in CM	String
Height	Height of object in CM	String
Width	Width of object in CM	Numeric
Depth	Depth of object in CM	String
Weight	Weight of object in kg	String
Duration	Duration of object for media art works in seconds	String

The public data itself was released quietly. Fivethirtyeight described it as ['an afterthought'](http://fivethirtyeight.com/features/an-excavation-of-one-of-the-worlds-greatest-art-collections/) to the museum's announcement that it was releasing 375,000 images of its artworks under Creative Commons Zero license — which means that they are available for free and unrestricted use.

In terms of use of the image collection so far:

* [How Bots See Art](https://twitter.com/HowBotsSeeArt) describes pieces from the collection from the perspective of a computer;
* [Public Domain Cut-Up](https://twitter.com/PDCutup) makes collages from MOMA and New York Public Library images;
* [Face-Swap The Met](https://twitter.com/artfaceswaps) rides the pop cultural vogue for such apps

In [3]:

```
artists = pd.read_csv('Data/artists.csv', encoding='utf8')
artworks = pd.read_csv('Data/artworks.csv', encoding='utf8')

types_df = pd.DataFrame(artworks.dtypes)
types_df
```

Out[3]:

	0
Artwork ID	int64
Title	object
Artist ID	object
Name	object
Date	object
Medium	object
Dimensions	object
Acquisition Date	object
Credit	object
Catalogue	object
Department	object
Classification	object
Object Number	object
Diameter (cm)	float64
Circumference (cm)	float64
Height (cm)	float64
Length (cm)	float64
Width (cm)	float64
Depth (cm)	float64
Weight (kg)	float64
Duration (s)	float64

Step 3: Data Preparation

Data Cleansing Checklist

Inspection of the data highlights a numnber of stumbling blocks to exploration of our hypothesis. Set out below are the identified issues and proposed remedies, grouped by type of issue.

A. Errors from Data Entry

- The medium column is highly messy with essentially freeform descriptions of methods deployed. This may be an intractable problem, since cleaning > 100,000 rows by hand would defeat the object. We may be able to use natural language processing to extract value from the series
- That the medium column is like this is understandable. Techniques in art history conservation tend to favour descriptive approaches toward medium description. There is no standardised coding
- However, some fields take the freedom to excess, with `Aquatint and carborundum relief from a portfolio of three aquatints (one with carborundum relief), one carborundum relief, one chromogenic color print, three digital prints, four etchings (two with chine collé, one with embossing), one linoleum cut, one lithograph, three screenprints, two woodcuts, and two polymer gravures (one with woodcut)` a particularly extreme example. This also underscores the fact that some works are grouped together, whereas others are inputted separately. This may introduce some skew into the data, but we hope not perceptibly. This is perhaps something for others to explore

In [4]:

```
def sample_df(df):
    return pd.DataFrame(np.concatenate([df.dtypes.T.values.reshape(1,-1),df.sampl
```

Taking a look at the data using a sample function to take random entries in the artists and artworks dataframes, there appears to be a lot of Nan entries. This is further confirmed by a search for Nan entries. The question is what these Nan entries meant in practice.

In [5]:

```
sample_df(artists)
```

Out[5]:

	dtypes	sample	columns
0	int64	541	Artist ID
1	object	Art Bevacqua	Name
2	object	Nationality unknown	Nationality
3	object	Male	Gender
4	float64	NaN	Birth Year
5	float64	NaN	Death Year

In [6]:

```
sample_df(artworks)
```

Out[6]:

	dtypes	sample	columns
0	int64	81849	Artwork ID
1	object	Hands Holding the Void (Invisible Object)	Title
2	object	2141	Artist ID
3	object	Alberto Giacometti	Name
4	object	1934 (cast c. 1954-55)	Date
5	object	Bronze	Medium
6	object	59 7/8 x 12 7/8 x 10" (152.1 x 32.6 x 25.3 cm)	Dimensions
7	object	1995-12-12	Acquisition Date
8	object	Louise Reinhardt Smith Bequest	Credit
9	object	Y	Catalogue
10	object	Painting & Sculpture	Department
11	object	Sculpture	Classification
12	object	775.1995	Object Number
13	float64	NaN	Diameter (cm)
14	float64	NaN	Circumference (cm)
15	float64	152.1	Height (cm)
16	float64	NaN	Length (cm)
17	float64	32.7	Width (cm)
18	float64	25.4	Depth (cm)
19	float64	NaN	Weight (kg)
20	float64	NaN	Duration (s)

In [7]:

```
artworks.isnull().sum()
```

Out[7]:

```
Artwork ID                0
Title                    52
Artist ID              1460
Name                   1460
Date                   2308
Medium                11919
Dimensions            11463
Acquisition Date       5463
Credit                 3070
Catalogue                 0
Department                0
Classification            0
Object Number             0
Diameter (cm)        128863
Circumference (cm)   130252
Height (cm)           18369
Length (cm)          129526
Width (cm)            19259
Depth (cm)           118819
Weight (kg)          129964
Duration (s)         127178
dtype: int64
```

B. Missing Values

Reflecting further on the identified Nans, the credit, catalogue, department, classfication, object number, artwork ID numbers are all complete, with only 52 items missing titles.

Already we can see that other columns are not as clean as they might be, and that will have to be a first priority in order to extract the real value from the data, alongside the transformations we have already identified.

The duration, diameter and circumference columns all need cleaning. The same perhaps goes for the Height and Length columns. There also appear to have been some cases where height and length have been used, and some where height and width have been used for similar objects. Yet, for the purposes of classification, we should still be able to obtain value from them, it may just make calculation more difficult.

In [8]:

```
# There are even cases where height and length are used when presumably Length/wi

dimension_search = artworks[(artworks['Height (cm)'] >= 1) & (artworks['Length (c
dimension_search
```

Out[8]:

	Artwork ID	Title	Artist ID	Name	Date	Medium	Dimensions	Acquisition Date
1269	1927	Oceana Box	6460	Russel Wright	1931	Wood	3 x 9 1/4" (7.6 x 23.5 cm)	1943-02-18
1316	1993	Frying Pan	9005	Corning Glass Works, Corning, NY	c. 1942	Borosilicate glass and steel	Overall: h. 2 3/4 x l. 12 1/2" (h. 7 x w. 31.8...	1948-03-17
1373	2061	Wall-Hanging	2631	Sheila Hicks	c. 1962	Wool	38 1/2 x 42" (97.8 x 106.7 cm)	NaN
1619	2360	Child's Wheelbarrow	4922	Gerrit Rietveld	1923	Painted wood	12 1/2 x 11 3/8 x 33 1/2" (31.8 x 28.9 x 85.1 cm)	1993-05-04
1825						Steel-		

An important caveat is that in some cases, the Nans may simply be a case that some entries are an artifact from importing the data. Some works do not have a 'depth' characteristic for some reason (which may not literally, empirically be true but for the purposes of artwork conservation, one has not been entered). So we would ascribe these to be missing data more than anything else. The same is true of the height, width columns. Thus the dimensions column may be redundant.

One way to solve this is arguably to use the dimensions column which locates relevant information about the dimensions of the artwork in one place. Yet, it would require signficant effort and cleaning to arrive at usable values.

The dimensions column seems more complete, but would require hefty cleaning to extract value since the values contained within do not follow a familiar pattern, and will require splitting and cleaning. Nevertheless, it's use might be more reliable and swifter than setting calculations on the other constituent dimension columns (e.g. 'Height (cm)' blind). The reason this is important is because we will wish to create additional features (such as area, volume) for artworks, as well as creating size categories that we can use to feed our classifier. It is reasonable to suspect that if donors are important, some may favour investment in larger-scale works, some in books and prints each of which are of a very different scale and size.

In [9]:

```
# Show what we are working with

artworks.head()
```

Out[9]:

	Artwork ID	Title	Artist ID	Name	Date	Medium	Dimensions	Acquisition Date
0	2	Ferdinandsbrücke Project, Vienna, Austria, Ele...	6210	Otto Wagner	1896	Ink and cut-and-pasted painted pages on paper	19 1/8 x 66 1/2" (48.6 x 168.9 cm)	1996-04-09
1	3	City of Music, National Superior Conservatory ...	7470	Christian de Portzamparc	1987	Paint and colored pencil on print	16 x 11 3/4" (40.6 x 29.8 cm)	1995-01-17
2	4	Villa near Vienna Project, Outside Vienna, Aus...	7605	Emil Hoppe	1903	Graphite, pen, color pencil, ink, and gouache ...	13 1/2 x 12 1/2" (34.3 x 31.8 cm)	1997-01-15
3	5	The Manhattan Transcripts Project, New York, N...	7056	Bernard Tschumi	1980	Photographic reproduction with colored synthet...	20 x 20" (50.8 x 50.8 cm)	1995-01-17
4	6	Villa, project, outside Vienna, Austria, Exter...	7605	Emil Hoppe	1903	Graphite, color pencil, ink, and gouache on tr...	15 1/8 x 7 1/2" (38.4 x 19.1 cm)	1997-01-15

5 rows × 21 columns

There are 1460 columns each missing artist name and Artist ID. The number may be coincidental but we should check that out.

In [10]:

```
name_search = artworks[(artworks['Name'] == None) & (artworks['Artist ID'] == Non
name_search
```

Out[10]:

Artwork ID	Title	Artist ID	Name	Date	Medium	Dimensions	Acquisition Date	Credit	Catalogue	...

0 rows × 21 columns

Okay, so it seems that the 1460 figure for each Name and Artist ID column is a coincidence and we don't

Okay, so it seems that the 1460 figure for each Name and Artist ID column is a coincidence and we don't have any columns where both conditions are true.

C. Data Entry Issues

- The title column for art works may contain value for our predictions. However, given it is in the form of freeform string fields, we will need to use natural language processing to transform the data into a usable and interrogatable format
- We have already referenced the similar challenges posed by the freeform data within the medium column, and we propose to treat that the same way as the title column and extract value through natural language processing

Physically Impossible Values

- Some values for works dimensions are particularly large (see later plot in EDA).

D. Proposed Transformations

- We are able to join the fields of the Artworks and Artists files via the use of the 'Artwork ID' field
- The Department column is less helpful than the Classification column for the purposes of categorising artworks, since there are fewer Departments and that grants lower resolution
- We may wish to transform the artist table with more data to more finely delineate artistic periods where possible, to help our predictions along
- The credit column is again freeform String fields. We will categorise the column so that this can be used as a label
- The date column will need parsing since there are a lot of rows with additional text or formatting, and some data ranges, which will not be interpreted well by Pandas or Numpy
- The object number for each item may not be trustworthy - accession numbers tend to concern where an object is brought in and taken out of a collection and is not necessarily a unique identifier per se for the item
- The dimensions for the objects need to be parsed to obtain maximum value from them. For example, the Height column is stored as strings, whereas width is stored as numeric values. We should, with some parsing, be able to create additonal features to help our predictions. Initial thoughts are that:
    - we can parse and clean dimensions fields in order to enable us to carry out calculations using their values, and visualise data
    - we can create dimension categories by looking at the distribution of sizes, in order to provide a further category to help our predictions
- Finally, some columns are not relevant for all works - for example in the case of 'Duration' which is for media art works
- given that we intend to use Support Vector Machines as our classifier, we will need to convert most of our data to numerical data to feed to our model, even where in the intevening period we create categorical data (e.g. in the case of size of art work). We will also have to dummify data across categories, such as would be the case with regard to 'Classification' of art work and 'Department'

It is highly likely that in the course of our data exploration we will establish further lacunae and challenges to negotiate, as well as ways to extract value from the data.

Step 4: Data Exploration

A. What's in the Collection

Let's see how art works split by department.

In [11]:

```
departments_df = pd.DataFrame(artworks['Department'].value_counts())
departments_df.head(10)
```

Out[11]:

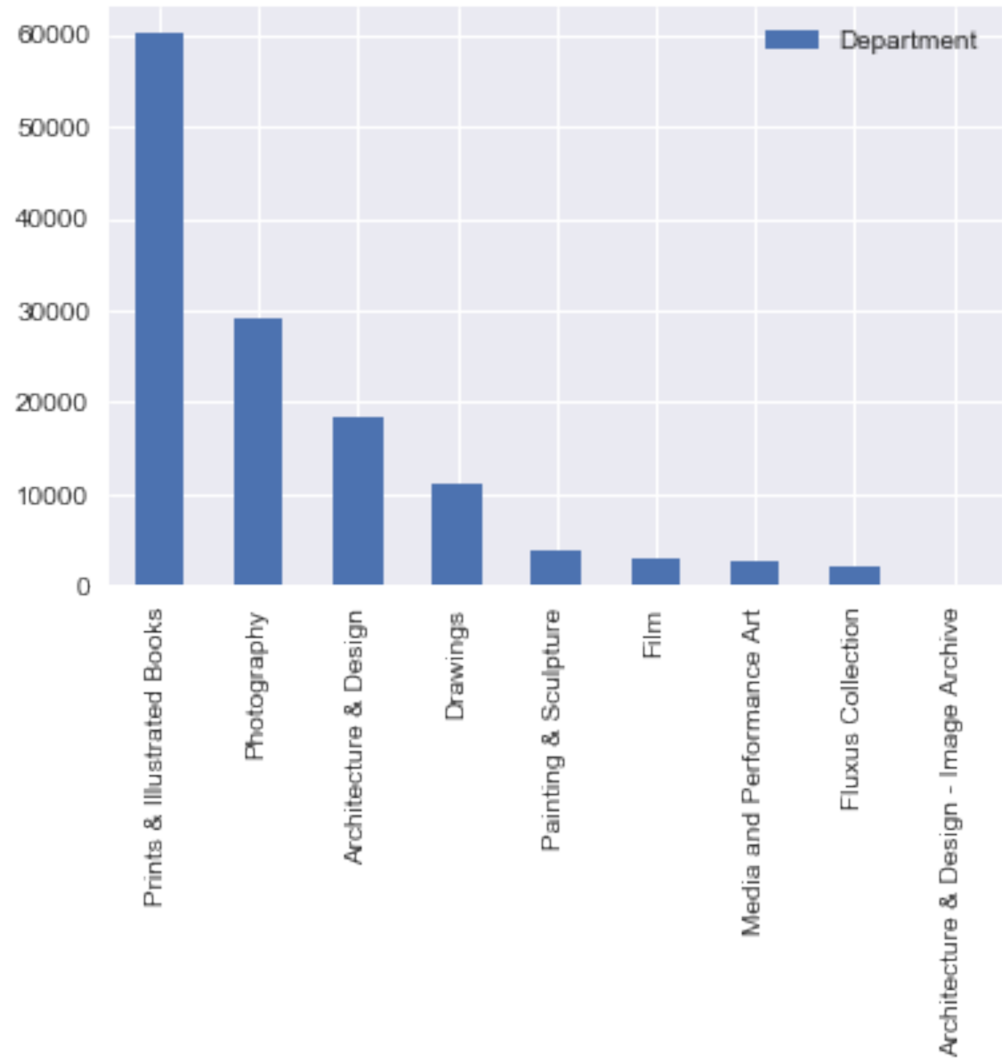
	Department
Prints & Illustrated Books	60128
Photography	29161
Architecture & Design	18269
Drawings	11027
Painting & Sculpture	3806
Film	3088
Media and Performance Art	2627
Fluxus Collection	2135
Architecture & Design - Image Archive	21

In [12]:

```
departments_df.plot(kind='bar')
```

Out[12]:

<matplotlib.axes._subplots.AxesSubplot at 0x112d4de90>



In [13]:

```
classifications_df = pd.DataFrame(artworks['Classification'].value_counts())
classifications_df.head(50)
```

Out[13]:

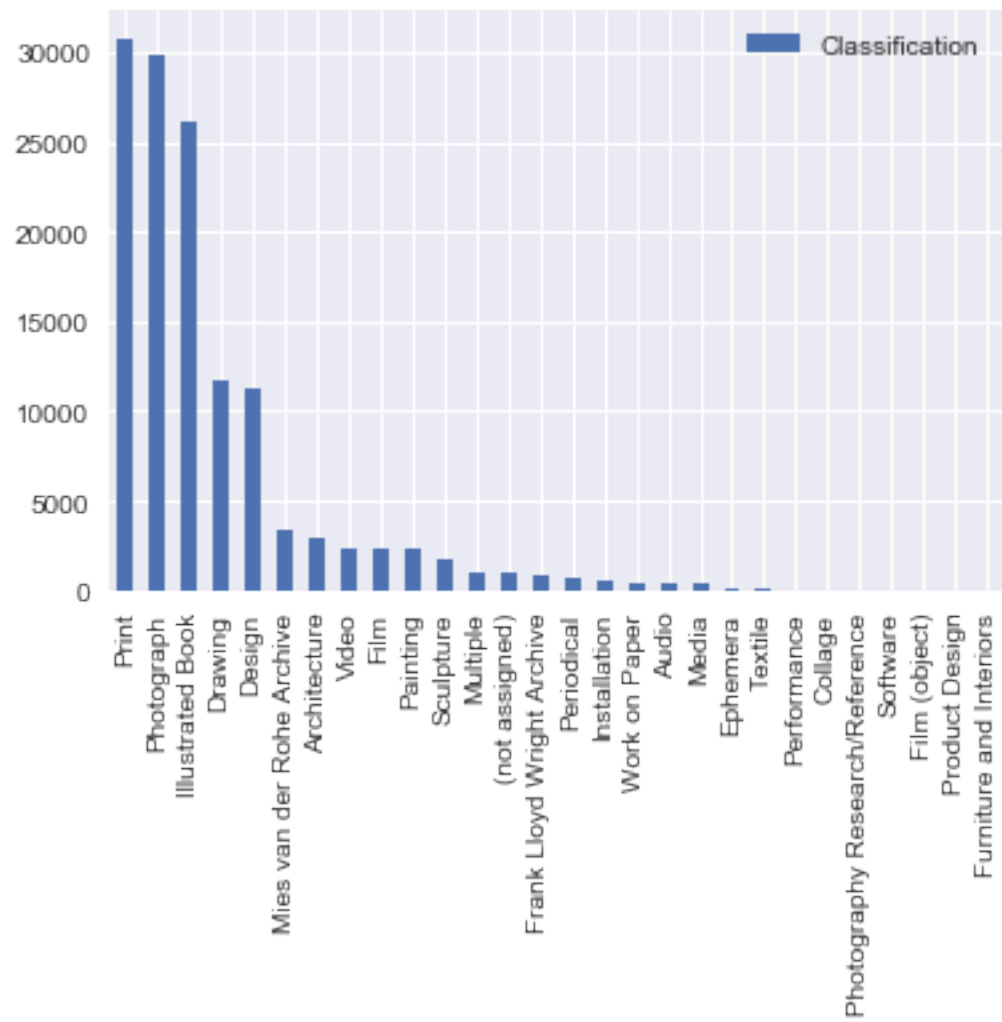
	Classification
Print	30807
Photograph	29909
Illustrated Book	26160
Drawing	11735
Design	11223
Mies van der Rohe Archive	3331
Architecture	2947
Video	2363
Film	2292
Painting	2270
Sculpture	1669
Multiple	1030
(not assigned)	1029
Frank Lloyd Wright Archive	785
Periodical	741
Installation	596
Work on Paper	436
Audio	429
Media	343
Ephemera	89
Textile	33
Performance	24
Collage	9
Photography Research/Reference	4
Software	3
Film (object)	3
Product Design	1
Furniture and Interiors	1

In [14]:

```
classifications_df.plot(kind='bar')
```

Out[14]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1131e1a10>
```



The Mies van der Rohe and Frank Lloyd Wright archives can probably be merged together with the Architecture section. While they belong to a specific collection by each architect, for the purposes of our analysis, they can be grouped with other architecture resources.

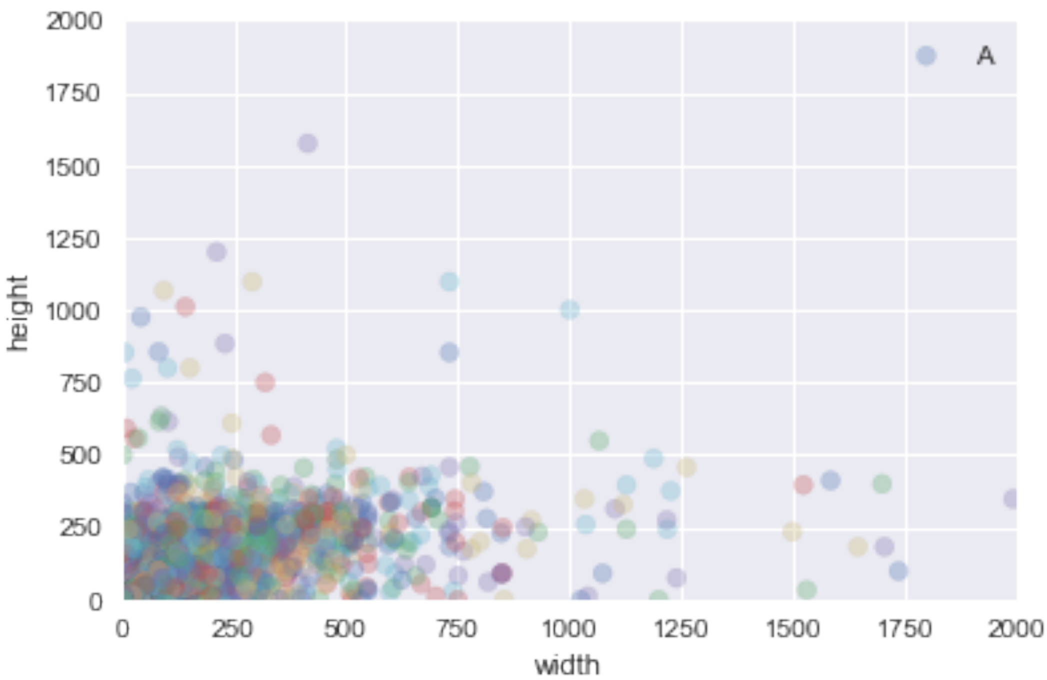
We will likely drop the Software, Film(object), Product Design and Furniture and Interiors classifications since they are not sufficiently granular or scalar to aid our classifier.

In [15]:

```
plt.scatter(artworks['Width (cm)'],artworks['Height (cm)'], alpha=0.3, color=colo
plt.xlim(0,2000)
plt.ylim(0,2000)
plt.xlabel("width")
plt.ylabel("height")
plt.legend("Artwork")
```

Out[15]:

```
<matplotlib.legend.Legend at 0x11477a050>
```



B. Using Datetime to plot acquisitions over time

Let's take a quick detour into time series to explore the growth of the collection over time. We'll read in our artworks csv again and reindex using the acquisition column

In [16]:

```
moma = pd.read_csv('data/artworks.csv', index_col=12)
```

In [17]:

```
moma['Acquisition Date'].dtype
```

Out[17]:

```
dtype('O')
```

In [18]:

```
moma = moma[moma["Acquisition Date"] != '1216-10-18']
```

In [19]:

```
from datetime import datetime

moma['Acquisition Date'] = pd.to_datetime(moma['Acquisition Date'], infer_datetim
```

In [20]:

```
moma.head()
```

Out[20]:

	Artwork ID	Title	Artist ID	Name	Date	Medium	Dimensions	Acc Dat
Object Number								
885.1996	2	Ferdinandsbrücke Project, Vienna, Austria, Ele...	6210	Otto Wagner	1896	Ink and cut-and-pasted painted pages on paper	19 1/8 x 66 1/2" (48.6 x 168.9 cm)	199
1.1995	3	City of Music, National Superior Conservatory ...	7470	Christian de Portzamparc	1987	Paint and colored pencil on print	16 x 11 3/4" (40.6 x 29.8 cm)	199
1.1997	4	Villa near Vienna Project, Outside Vienna, Aus...	7605	Emil Hoppe	1903	Graphite, pen, color pencil, ink, and gouache ...	13 1/2 x 12 1/2" (34.3 x 31.8 cm)	199
2.1995	5	The Manhattan Transcripts Project, New York, N...	7056	Bernard Tschumi	1980	Photographic reproduction with colored synthet...	20 x 20" (50.8 x 50.8 cm)	199
2.1997	6	Villa, project, outside Vienna, Austria, Exter...	7605	Emil Hoppe	1903	Graphite, color pencil, ink, and gouache on tr...	15 1/8 x 7 1/2" (38.4 x 19.1 cm)	199

In [21]:

```
moma = moma.dropna(subset=['Acquisition Date'])
```

In [22]:

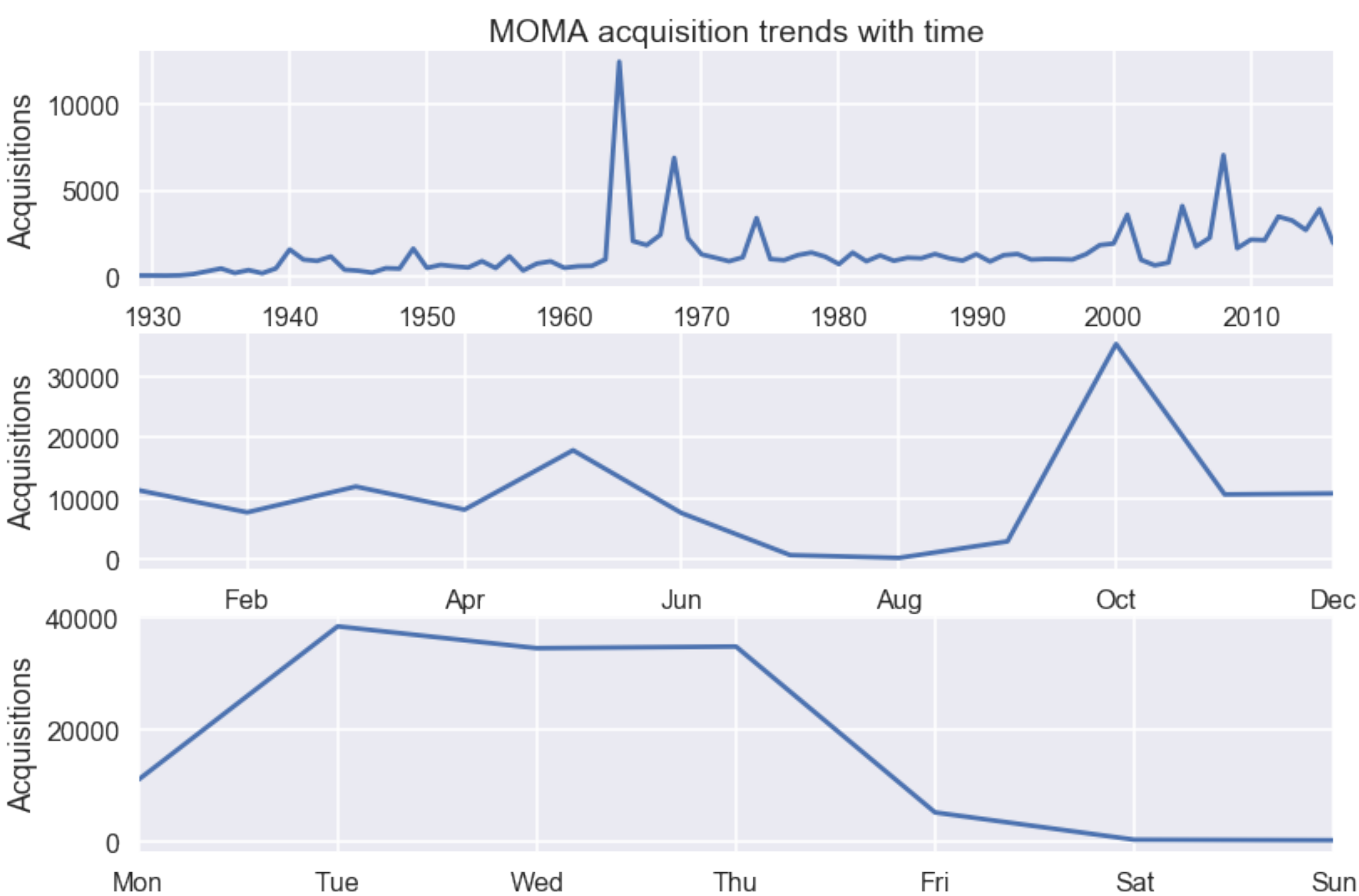
```
moma.iloc[0]
```

Out[22]:

```
Artwork ID
2
Title                      Ferdinandsbrücke Project, Vienna, Austria, Ele
...
Artist ID                                                               6
210
Name                                                             Otto Wag
ner
Date                                                                    1
896
Medium                         Ink and cut-and-pasted painted pages on pa
per
Dimensions                                19 1/8 x 66 1/2" (48.6 x 168.9
cm)
Acquisition Date                                          1996-04-09 00:00
:00
Credit                     Fractional and promised gift of Jo Carole and
...
```

In [23]:

```
sns.set_context('poster')

fig, ax = plt.subplots(3, 1);
ylabel = 'Acquisitions'

(moma.groupby(pd.Grouper(freq='A', key='Acquisition Date'))
 .size()
 .plot(ax=ax[0]))

(moma
 .groupby(moma['Acquisition Date'].dt.month)
 .size()
 .plot(ax=ax[1]))

(moma.
 groupby(moma['Acquisition Date'].dt.weekday)
 .size()
 .plot(ax=ax[2]))

months = {0: '_', 1: 'Jan', 2: 'Feb', 3: 'Mar', 4: 'Apr',
        5: 'May', 6: 'Jun', 7: 'Jul', 8: 'Aug', 9: 'Sep',
        10: 'Oct', 11: 'Nov', 12: 'Dec'}
days = {0: 'Mon', 1: 'Tue', 2: 'Wed', 3: 'Thu', 4: 'Fri', 5: 'Sat', 6: 'Sun'}

ax[0].set_title('MOMA acquisition trends with time')
ax[1].set_xticklabels([months[i] for i in ax[1].get_xticks()]);
ax[2].set_xticklabels([days[i] for i in ax[2].get_xticks()]);

for a in ax:
    a.set_xlabel('');
    a.set_ylabel(ylabel);
```



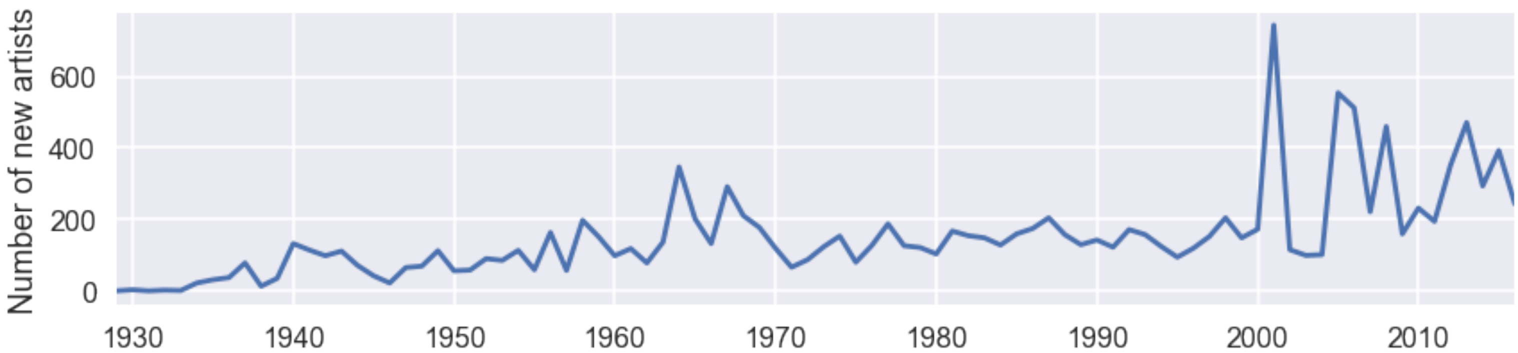
So we have lots of acquisitions in 1964, 1968 and 2008. October is a prime month for acquisitions it seems, and acquisitions seem to peak on a Tuesday each week and tail off toward the weekend.

The peak in 1964 is a strange one. We can't see anything in the Moma archive: that https://www.moma.org/learn/resources/archives/EAD/MoMAExhFiles1960sp.html (https://www.moma.org/learn/resources/archives/EAD/MoMAExhFiles1960sp.html) explains why this should be the case. A new wing did open in May of that year, but Moma moved to its new home in 1939 and other wings opened elsewhere in 1951, 1968, 1981 and these do not see such a pronounced spike in acquisitions. It's a mystery.

Now let's take a look at how the number of new artists has progressed over time.

In [24]:

```
# This is a DataFrame where all items by an artist except their first acquisition
firsts = moma.drop_duplicates('Name')

fig, ax = plt.subplots(figsize=(14, 3))

(firsts.groupby(pd.Grouper(key='Acquisition Date', freq='A'))
 .size()
 .plot())

ax.set_xlabel('');
ax.set_ylabel('Number of new artists');
```



In [ ]:

Next let's take a look at the acquisition of the top artists in the painting and sculpture department. First, we'll need to categorise a few columns. Then we will use the pandas isin() method to construct a boolean series to filter out people who are not in the list.

In [25]:

```
# Let's quickly categorise some columns

categorical_columns = ['Classification', 'Department', 'Catalogue']

for c in categorical_columns:
    moma[c] = moma[c].astype('category')
    print(c, '\n', moma[c].cat.categories)
```

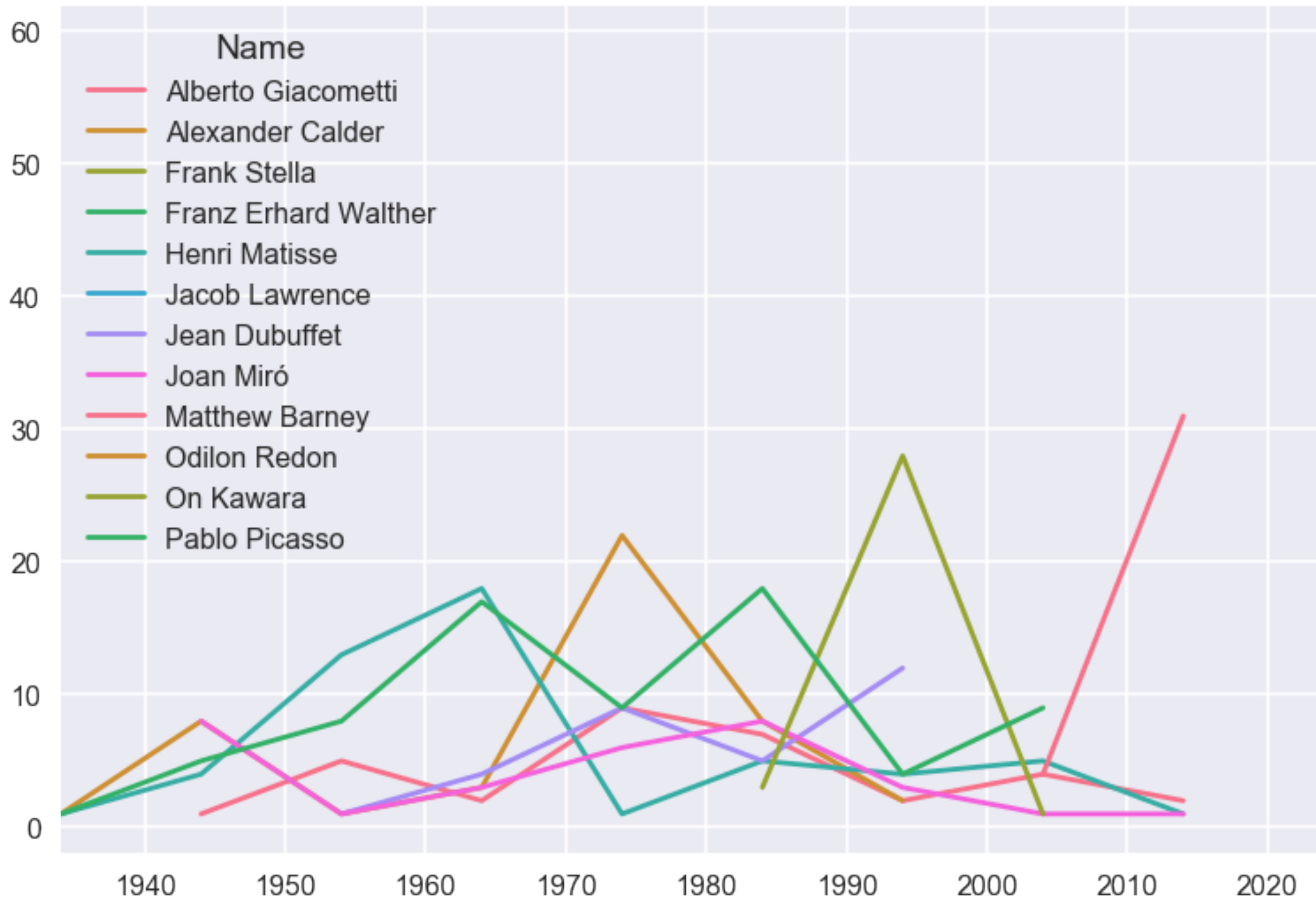
```
(u'Classification', u'\n', Index([u'(not assigned)', u'Architecture'
, u'Audio', u'Collage', u'Design',
       u'Drawing', u'Ephemera', u'Film', u'Film (object)',
       u'Frank Lloyd Wright Archive', u'Illustrated Book', u'Install
ation',
       u'Media', u'Mies van der Rohe Archive', u'Multiple', u'Painti
ng',
       u'Performance', u'Periodical', u'Photograph',
       u'Photography Research/Reference', u'Print', u'Product Design
',
       u'Sculpture', u'Software', u'Textile', u'Video', u'Work on Pa
per'],
      dtype='object'))
(u'Department', u'\n', Index([u'Architecture & Design', u'Drawings',
u'Film', u'Fluxus Collection',
       u'Media and Performance Art', u'Painting & Sculpture', u'Phot
ography',
       u'Prints & Illustrated Books'],
      dtype='object'))
(u'Catalogue', u'\n', Index([u'N', u'Y'], dtype='object'))
```

In [26]:

```
# Next we create a list of people who make paintings and sculptures

top = list(moma[moma['Department'] == 'Painting & Sculpture']
            .groupby('Name')
            .size()
            .sort_values()
            .tail(12) # the top 12 painters and sculptors
            .index)
```

In [27]:

```
with sns.color_palette(palette='husl', n_colors=8):  # more than 6 colors
    fig, ax = plt.subplots()

    (moma[moma['Name'].isin(top) & #if the artist is in our group
          (moma['Department'] == 'Painting & Sculpture')]
     .groupby([pd.Grouper(freq='10A', key='Acquisition Date'), 'Name'])
     .size()
     .unstack()
     .plot(ax=ax))

    ax.set_xlabel('')
```



Artists clearly have their vogueish periods. We can reasonably expect this to be a combination of natural lifespan, fashion and the taste of donors (which are likely to be closely correlated and possibly collinear. However, there are other factors that will influence a donor's taste and therefore this is worth exploring further.

In [1]:

```
import sexmachine.detector as gender

# run first: pip install -i https://pypi.anaconda.org/pypi/simple sexmachine

g = gender.Detector()

def infer_gender(name):
    try:
        return g.get_gender(name.split()[0])
    except:
        return
```

In [28]:

```
firsts.loc[:, 'Gender'] = firsts['Name'].apply(infer_gender)
firsts.groupby('Gender').size()
```

```
/Users/patrickbrown/anaconda/lib/python2.7/site-packages/pandas/core
/indexing.py:337: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/panda
s-docs/stable/indexing.html#indexing-view-versus-copy
(http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-
view-versus-copy)
  self.obj[key] = _infer_fill_value(value)
/Users/patrickbrown/anaconda/lib/python2.7/site-packages/pandas/core
/indexing.py:517: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/panda
s-docs/stable/indexing.html#indexing-view-versus-copy
(http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-
view-versus-copy)
  self.obj[item] = s
```

Out[28]:

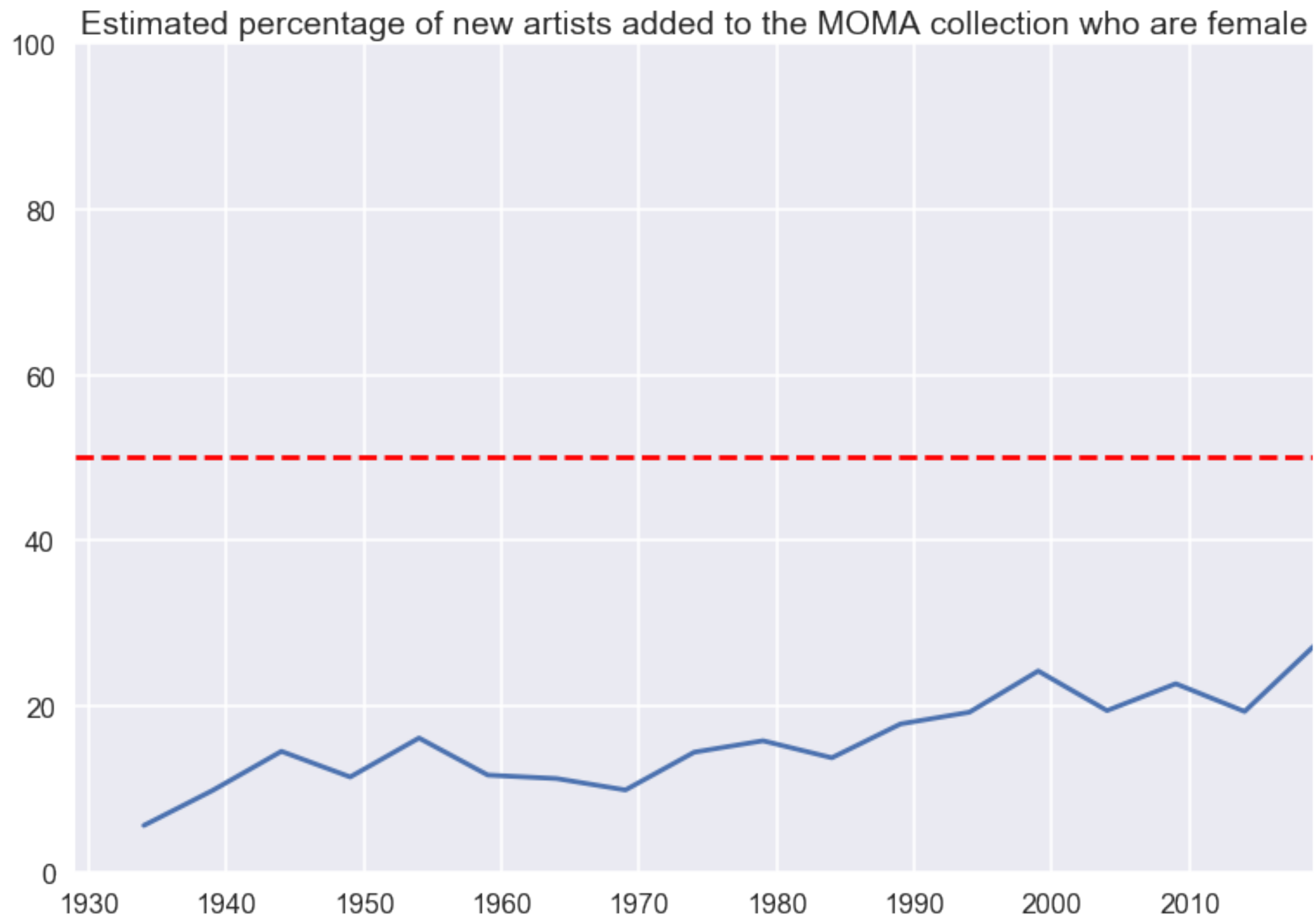
```
Gender
andy             2803
female           1771
male             8372
mostly_female     194
mostly_male       272
dtype: int64
```

In [30]:

```
gender_trends = (firsts
                 .groupby([pd.Grouper(key='Acquisition Date', freq='5A'), 'Gender
                 .size()
                 .unstack())

gender_trends['percent female'] = 100. * gender_trends['female'] / (gender_trends
```

In [31]:

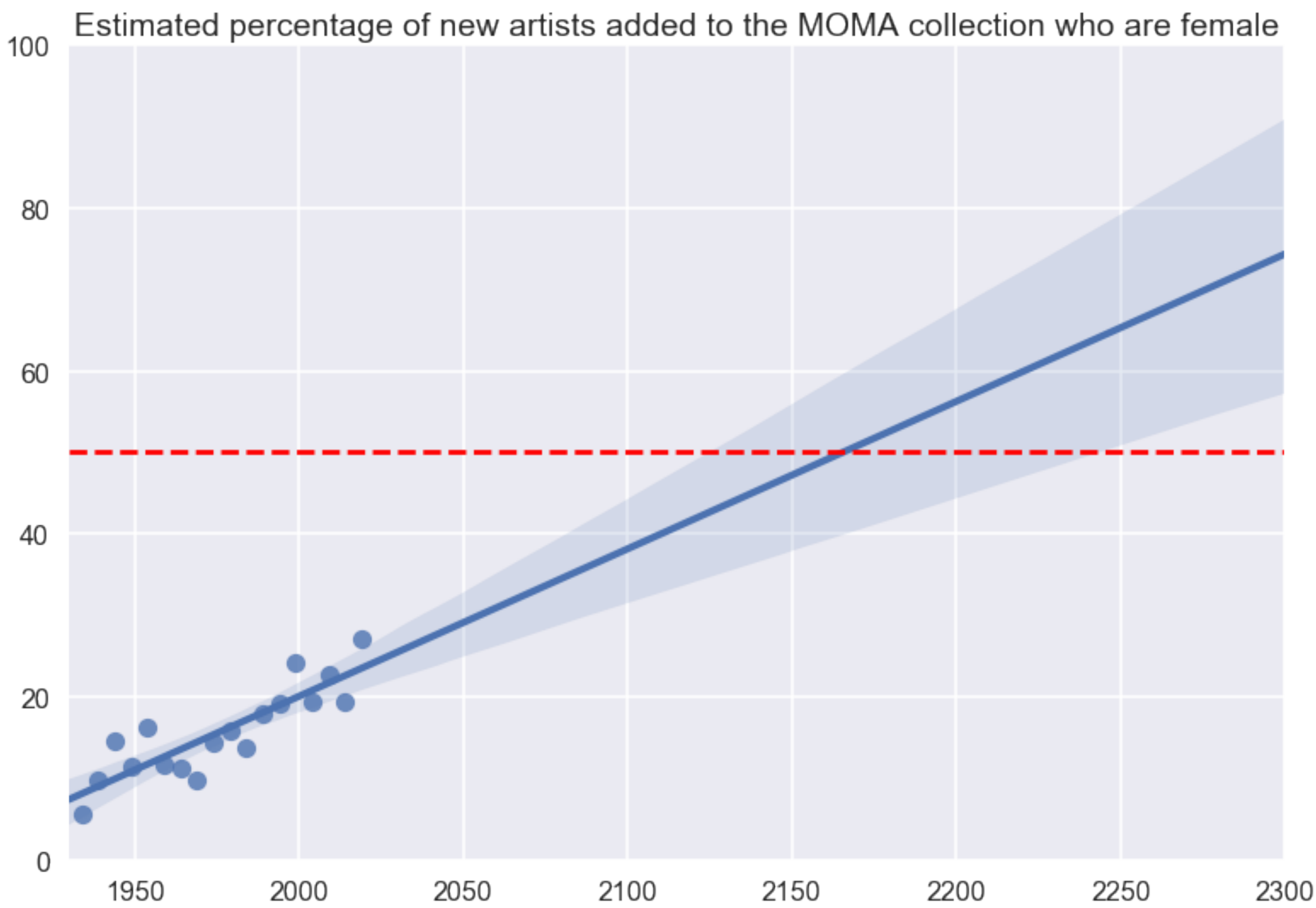
```
ax = gender_trends['percent female'].plot()
ax.set_title('Estimated percentage of new artists added to the MOMA collection wh
ax.set_xlabel('')
ax.plot(ax.get_xlim(), [50, 50], 'r--')
ax.set_ylim(0, 100);
```



The above plot gives us a plot of the percentage of new artists in five year bins who are female. The number is rising year on year, but only about 20% of artists added each year are female. We need to stress that sexmachine, the module we have used, is not perfect since it there are around a quarter of the names in the artworks csv file that are deemed by it to be 'andy' or androgynous and therefore it cannot ascribe those names to male or female. That said, if we assume that the module misattributes a female name to 'andy' as often as it does a male name, then our conclusions are still accurate. So what if this slowly rising trend continues. How long would it take for parity to be achieved between men and women in the Moma collection?

In [32]:

```
fig, ax = plt.subplots()
ax.set_ylim(0, 100)
ax.set_xlim(1930, 2300)
ax = sns.regplot(x=gender_trends.reset_index()['Acquisition Date'].apply(lambda x
            y=gender_trends["percent female"])
ax.set_title('Estimated percentage of new artists added to the MOMA collection wh
ax.set_xlabel('')
ax.set_ylabel('')
ax.plot(ax.get_xlim(), [50, 50], 'r--');
```



At the current rate, sometime around the middle of the next century. The question is really: why should this be the case? Is there some structural reason why men are overrepresented in the MOMA? One for further investigation in another project. We will use the module again further down to add a gender column to our dataframe.

C. What's in the Collection?

Let's move on and concatenate the two csvs together, artists and artworks csvs and take a look.

In [596]:

```
df = pd.concat([artists,artworks],join='inner',keys='Artist ID')
df
```

Out[596]:

	Artist ID	Name

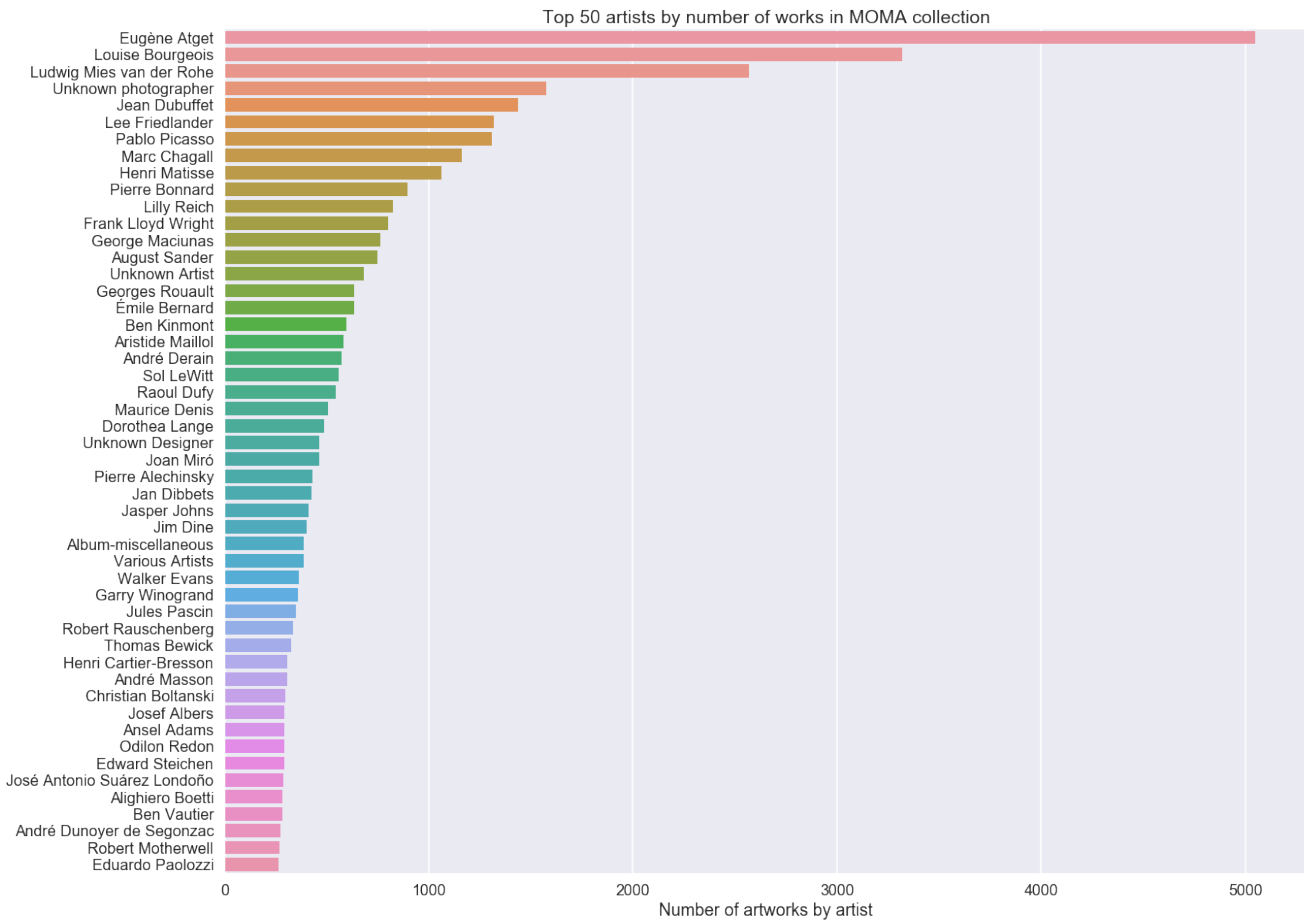
A	**0**	1	Robert Arneson
...	**1**	2	Doroteo Arnaiz
	2	3	Bill Arnold
	3	4	Charles Arnoldi
	4	5	Per Arnoldi
	5	6	Danilo Aroldi
	6	7	Bill Aron
	7	9	David Aronson
	8	10	Irene Aronson
	9	11	Jean (Hans) Arp
	10	12	Jüri Arrak
	11	13	J. Arrelano Fischer
	12	15	Folke Arstrom
	13	16	Cristobal Arteche
	14	18	Artko
	15	19	Richard Artschwager
	16	21	Ruth Asawa
	17	22	Isidora Aschheim
	18	23	Charles Robert Ashbee
	19	24	Donald Ashcraft
	20	25	E. M. Ashe
	21	26	Göran Åslin
	22	27	Erik Gunnar Asplund
	23	28	Geneviève Asse
	24	30	Sergio Asti
	25	31	Dana Atchley
	26	32	Atelier Eggers
	27	33	A.A.P.
	28	34	Alvar Aalto
	29	35	Aino Aalto
...	...	...	...
r	**130232**	2637	Dick Higgins
	130233	2637	Dick Higgins
	130234	2637	Dick Higgins
	130235	2637	Dick Higgins
	130236	44757	Max Neuhaus

130237	756	George Brecht
130238	4469	Nam June Paik
130239	2637, 30644	Dick Higgins, Al Hansen
130240	912	John Cage
130241	2637	Dick Higgins
130242	756	George Brecht
130243	42821, 2928	Earl Brown, Ray Johnson
130244	2637	Dick Higgins
130245	36947	Toshi Ichiyanagi
130246	NaN	NaN
130247	2637	Dick Higgins
130248	67694	Glenn Williams
130249	2637	Dick Higgins
130250	2637	Dick Higgins
130251	2637	Dick Higgins
130252	2637	Dick Higgins
130253	2637	Dick Higgins
130254	2637	Dick Higgins
130255	NaN	NaN
130256	4469	Nam June Paik
130257	4469	Nam June Paik
130258	NaN	NaN
130259	67695	Ely Ramen
130260	NaN	NaN
130261	21398	George Maciunas

145353 rows × 2 columns

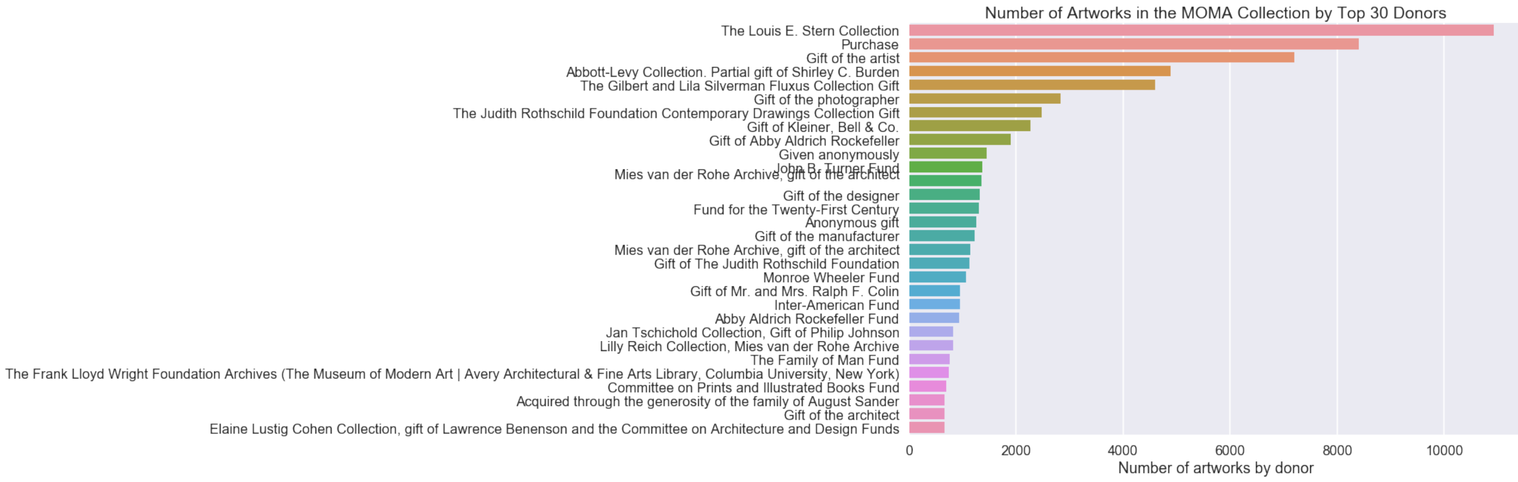
In [597]:

```
def plot_total_counts(data,column='Name',figsize=(20,16),title=None):
    counts = data[column].value_counts()[:50]
    plt.figure(figsize=figsize)
    sns.barplot(counts.values,counts.index)
    plt.xlabel("Number of artworks by artist")
    plt.xticks(rotation=0)
    plt.title(title)
    plt.show()

plot_total_counts(df,title='Top 50 artists by number of works in MOMA collection'
```



In [598]:

```
sns.set_context('poster')
counts = artworks['Credit'].value_counts()[:30]
sns.barplot(counts.values,counts.index)
plt.xlabel("Number of artworks by donor")
plt.xticks(rotation=0)
plt.title('Number of Artworks in the MOMA Collection by Top 30 Donors')
plt.show()
```



When grouping the collection by donor, the top 30 suggest some preliminary directions with regard to donors to include or exclude. The 'Gift of the Artist' category will include a number of different artists' own donations in their own right. The remainder on first inspection can stand as they are.

Trying to run a classifier on all the donors would likely fail. So we will use the top 30 donors as our test set, and use the remainder as training data.

D. Plotting Dimensions

Dimensions may be able to grant insights. Let's see how they cluster, and what proportion is taller than wide, wider than tall etc.

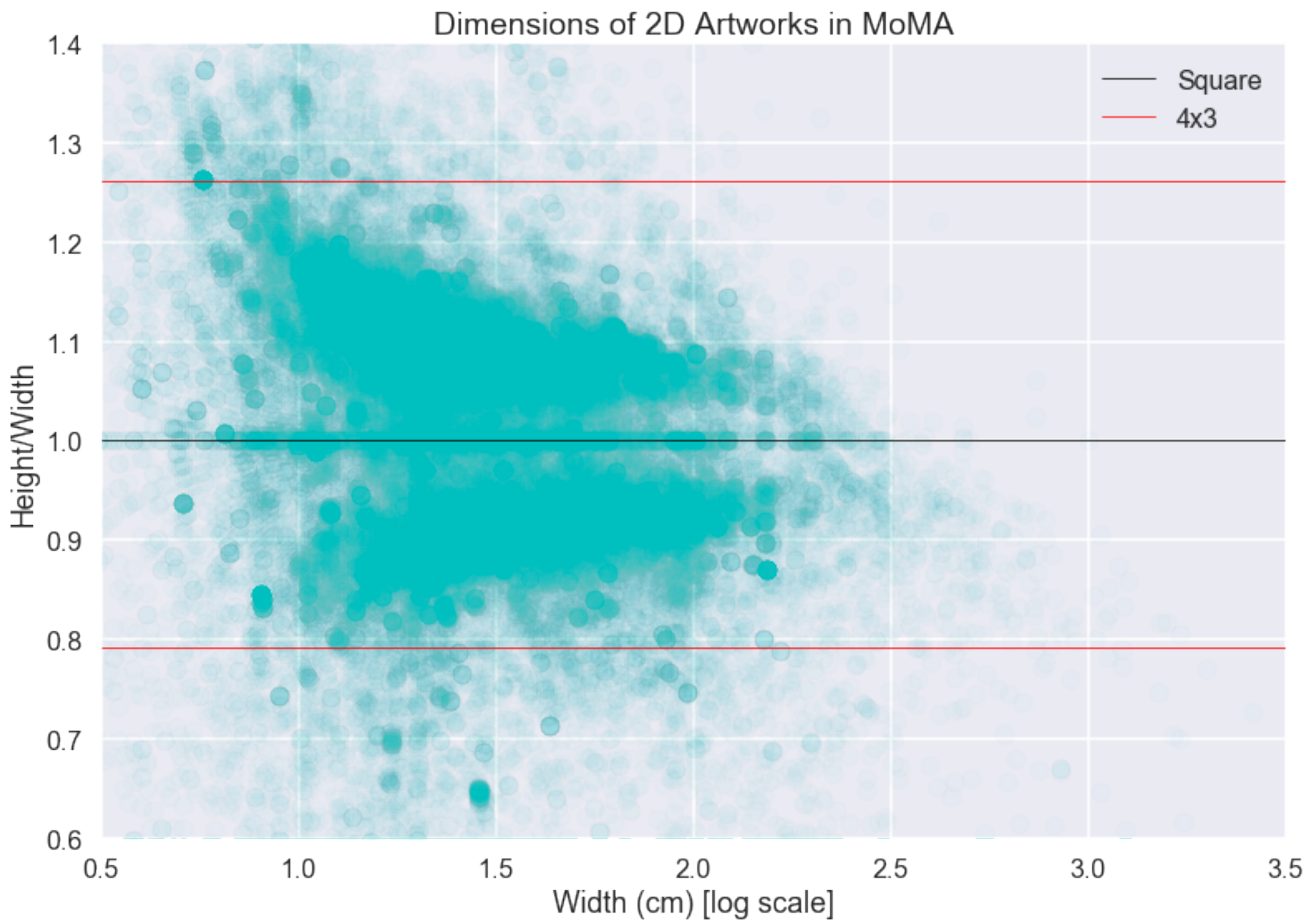
In [599]:

```
# Plot
ratio =  np.log10(artworks['Height (cm)'])/np.log10(artworks['Width (cm)'])
width = np.log10(artworks['Width (cm)'])

# 4/3
four_thirds = np.log10(4)/np.log10(3)
three_fourths = np.log10(3)/np.log10(4)
```

```
/Users/patrickbrown/anaconda/lib/python2.7/site-packages/ipykernel_l
auncher.py:2: RuntimeWarning: divide by zero encountered in log10

/Users/patrickbrown/anaconda/lib/python2.7/site-packages/ipykernel_l
auncher.py:3: RuntimeWarning: divide by zero encountered in log10
  This is separate from the ipykernel package so we can avoid doing
imports until
```

In [600]:

```
h = plt.scatter(width,ratio, alpha=0.02, c='c')
plt.axhline(y=1.0, color='k', linestyle='-',linewidth=0.75,label='Square')
plt.axhline(y=four_thirds, color='r', linestyle='-',linewidth=0.75,label='4x3')
plt.axhline(y=three_fourths, color='r', linestyle='-',linewidth=0.75)
plt.xlim((0.5,3.5))
plt.ylim((0.6,1.4))
plt.title("Dimensions of 2D Artworks in MoMA")
plt.xlabel('Width (cm) [log scale]')
plt.ylabel('Height/Width')
plt.legend()
plt.show()
```



Data Transformation

A. Translating our Classifications to Dummy Variables

We need to be able to classify our artworks. To do that, we'll have to engage in a little feature engineering.

First, as the classification column is free of Nans and looks pretty descriptive of the data, let's dummify the classification column so that we can use it in our classfier.

In [601]:

```
# first, we'll dummify the values

dummy_classifications = pd.get_dummies(artworks['Classification'], prefix='class'
dummy_classifications.head()
```

Out[601]:

	class_(not assigned)	class_Architecture	class_Audio	class_Collage	class_Design	class_Drawing	cl
0	0	1	0	0	0	0	
1	0	1	0	0	0	0	
2	0	1	0	0	0	0	
3	0	1	0	0	0	0	
4	0	1	0	0	0	0	

5 rows × 28 columns

In [602]:

```
# we select the columns we want to retain and join them to our dummies

cols_to_keep = ['Artwork ID','Title','Artist ID','Name','Date','Medium','Dimensic
data = artworks[cols_to_keep].join(dummy_classifications)
```

In [603]:

```
# let's take a look.  Great.

data.head()
```

Out[603]:

	Artwork ID	Title	Artist ID	Name	Date	Medium	Dimensions	Acquisition Date
0	2	Ferdinandsbrücke Project, Vienna, Austria, Ele...	6210	Otto Wagner	1896	Ink and cut-and-pasted painted pages on paper	19 1/8 x 66 1/2" (48.6 x 168.9 cm)	1996-04-09
1	3	City of Music, National Superior Conservatory ...	7470	Christian de Portzamparc	1987	Paint and colored pencil on print	16 x 11 3/4" (40.6 x 29.8 cm)	1995-01-17
2	4	Villa near Vienna Project, Outside Vienna, Aus...	7605	Emil Hoppe	1903	Graphite, pen, color pencil, ink, and gouache ...	13 1/2 x 12 1/2" (34.3 x 31.8 cm)	1997-01-15
3	5	The Manhattan Transcripts Project, New York, N...	7056	Bernard Tschumi	1980	Photographic reproduction with colored synthet...	20 x 20" (50.8 x 50.8 cm)	1995-01-17
4	6	Villa, project, outside Vienna, Austria, Exter...	7605	Emil Hoppe	1903	Graphite, color pencil, ink, and gouache on tr...	15 1/8 x 7 1/2" (38.4 x 19.1 cm)	1997-01-15

5 rows × 48 columns

B. Natural Language Processing on the Title Column

The Title column is only really semi-strucured data. In some cases it's very descriptive of the work itself, and in other cases can be an abstract description. Running natural language processing may be able to extract some value from the Title field.

In this case, we perhaps want to look for patterns of words rather than the relative appearance of individual words. Yet we don't want to weight different words as more or less important necessarily. So we'll use countvectorizer to do the job rather than something like Tfidf to vectorize.

In [604]:

```
from sklearn.feature_extraction.text import CountVectorizer

cv = CountVectorizer(binary=False, stop_words='english', min_df=3, ngram_range=(1
```

In [605]:

```
docs = cv.fit_transform(data.Title.dropna())
```

In [606]:

```
id2word = dict(enumerate(cv.get_feature_names()))
```

In [607]:

```
from gensim.models.ldamodel import LdaModel
from gensim.matutils import Sparse2Corpus
```

In [608]:

```
corpus = Sparse2Corpus(docs,documents_columns = False)
```

In [609]:

```
lda_model = LdaModel(corpus = corpus, id2word=id2word, num_topics=50, random_stat
```

In [610]:

```
num_topics = 50
num_words_per_topic = 10
for ti, topic in enumerate(lda_model.show_topics(num_topics = num_topics, num_wor
    print ("Topic: %d" % (ti))
    print (topic)
    print()
```

```
Topic: 0
(0, u'0.078*"building" + 0.028*"court" + 0.019*"nu" + 0.013*"march"
+ 0.010*"graham" + 0.009*"benjamin" + 0.008*"merz" + 0.008*"door" +
0.008*"dan graham" + 0.007*"cabinet"')
()
Topic: 1
(1, u'0.042*"years" + 0.028*"gallery" + 0.028*"cash" + 0.027*"100 ye
ars" + 0.022*"ca" + 0.017*"blue" + 0.016*"25" + 0.015*"57" + 0.013*"
graphic" + 0.012*"la ciudad"')
()
Topic: 2
(2, u'0.046*"il" + 0.017*"tragic" + 0.013*"version" + 0.013*"towers"
+ 0.012*"1917" + 0.011*"19" + 0.010*"seventies" + 0.010*"master" + 0
.010*"drawings pigozzi" + 0.010*"drawings pigozzi journal"')
()
Topic: 3
(3, u'0.040*"envelope" + 0.029*"water" + 0.023*"arts" + 0.020*"stati
onery envelope" + 0.020*"painting" + 0.018*"glass" + 0.013*"requiem"
+ 0.011*"social" + 0.011*"robert" + 0.010*"sculpture"')
```

Not so great. Let's try it instead with TfIDfVectorizer.

In [611]:

```
titles = data['Title'].fillna('')

from sklearn.feature_extraction.text import TfidfVectorizer

vectorizer = TfidfVectorizer(max_features = 250000, ngram_range=(1,2), stop_words

# Use 'fit' to learn the vocabulary
vectorizer.fit(titles)

# Use 'transform' to generate the sample X word matrix - one column per feature

X = vectorizer.transform(titles).toarray()
X
```

Out[611]:

```
array([[ 0.,  0.,  0., ...,  0.,  0.,  0.],
       [ 0.,  0.,  0., ...,  0.,  0.,  0.],
       [ 0.,  0.,  0., ...,  0.,  0.,  0.],
       ...,
       [ 0.,  0.,  0., ...,  0.,  0.,  0.],
       [ 0.,  0.,  0., ...,  0.,  0.,  0.],
       [ 0.,  0.,  0., ...,  0.,  0.,  0.]])
```

In [612]:

```
idf = vectorizer.idf_
idf
```

Out[612]:

```
array([ 10.21236141,  12.08416358,  12.08416358, ...,  12.08416358,
        12.08416358,  12.08416358])
```

In [613]:

```
feature_weights = dict(zip(vectorizer.get_feature_names(), idf))
# feature_weights_df = pd.DataFrame(feature_weights, columns=['title_term', 'weig
```

In [614]:

```
feature_weights
```

Out[614]:

```
{u'personal effects': 12.084163582136595,
 u'say say': 12.084163582136595,
 u'figure gun': 12.084163582136595,
 u'gai': 12.084163582136595,
 u'chain events': 12.084163582136595,
 u'orange rouge': 12.084163582136595,
 u'sieste ou': 12.084163582136595,
 u'maderista': 12.084163582136595,
 u'fransisco goya': 12.084163582136595,
 u'say saw': 10.831400613641227,
 u'graham robertson': 12.084163582136595,
 u'alm\xe1ssy t\xe9ri': 12.084163582136595,
 u'woods': 8.7699775774640685,
 u'rectangle small': 12.084163582136595,
 u'harvesters near': 12.084163582136595,
 u'woody': 10.985551293468486,
 u'angel took': 12.084163582136595,
 u'l\xe9on blov': 11.678698474028431,
```

C. Applying NLP to the Medium Column

Let's do the same for the medium column. This should perhaps fare better than the Title column analysis since it is more structured and repetitive data than in the case of the Title column. For this reason, we will use countvectorizer rather than Tfidfvectorizer.

In [615]:

```
data['Medium'].value_counts()
```

Out[615]:

```
Gelatin silver print
14103
Lithograph
7034
Albumen silver print
4845
Lithograph, printed in color
1833
Pencil on paper
1725
Etching
1710
Lithograph, printed in black
1523
Chromogenic color print
1488
Letterpress
1420
```

In [616]:

```
cv2 = CountVectorizer(binary=False, stop_words='english',min_df=0, ngram_range=(1
```

In [617]:

```
docs_2 = cv2.fit_transform(data.Medium.dropna())
```

In [618]:

```
id2word_2 = dict(enumerate(cv2.get_feature_names()))
```

In [619]:

```
corpus_2 = Sparse2Corpus(docs_2,documents_columns = False)
```

In [620]:

```
lda_model2 = LdaModel(corpus = corpus_2, id2word=id2word_2, num_topics=50, minimu
```

In [621]:

```
num_topics_2 = 60
num_words_per_topic_2 = 10
for ti, topic in enumerate(lda_model2.show_topics(num_topics = num_topics_2, num_
    print ("Topic: %d" % (ti))
    print (topic)
    print()
```

```
Topic: 0
(0, u'0.062*"coll\xe9" + 0.062*"chine coll\xe9" + 0.060*"chine" + 0.
022*"paper" + 0.013*"ink" + 0.013*"cut" + 0.012*"chine coll\xe9 port
folio" + 0.012*"coll\xe9 portfolio" + 0.011*"etchings chine coll\xe9
" + 0.011*"etchings chine"')
()
Topic: 1
(1, u'0.129*"pencil" + 0.098*"colored" + 0.084*"colored pencil" + 0.
057*"paper" + 0.054*"pencil colored" + 0.053*"pencil colored pencil"
+ 0.029*"pencil colored pencil tracing paper" + 0.029*"pencil colore
d pencil tracing" + 0.026*"pencil paper" + 0.022*"pencil ink"')
()
Topic: 2
(2, u'0.195*"print" + 0.149*"silver" + 0.146*"gelatin" + 0.145*"gela
tin silver" + 0.130*"silver print" + 0.128*"gelatin silver print" +
0.011*"print printed" + 0.009*"relief" + 0.006*"halftone" + 0.006*"r
elief halftone"')
()
Topic: 3
```

D. Binning the Dimensions

The dimensions data is messy. We could extract the cm data from the Dimensions column using Regex, but it would be finickety and involve much time and effort. And perhaps needlessly so, since Pandas is geared to ignore Nans when performing calculations.

As we are planning on performing a classification, we will bin the dimensions individually. We will have a slight problem with the fact that a lot of nans are present because different dimensions are relevant to different artworks. Because of the way pd.cut works, we will bin the data in quantiles, and then afterwards add a zero category that will represent an absence of data.

In [622]:

```
# Let's find our quantiles using data.describe for the relevant columns

data[['Height (cm)', 'Length (cm)', 'Depth (cm)', 'Width (cm)', 'Diameter (cm)',
```

Out[622]:

	Height (cm)	Length (cm)	Depth (cm)	Width (cm)	Diameter (cm)	Circumference (cm)
count	111893.000000	736.000000	11443.000000	111003.000000	1399.000000	10.000000
mean	37.712992	89.117417	18.291359	38.176838	23.248939	44.868020
std	48.151347	329.717487	57.703925	67.250118	45.460079	28.631604
min	0.000000	0.000000	0.000000	0.000000	0.635000	9.900000
25%	18.100000	17.031875	0.000000	17.800000	7.900000	23.500000
50%	27.940056	26.700000	0.700000	25.400100	13.700000	36.000000
75%	44.450100	79.100000	13.335013	44.800000	24.782500	71.125000
max	9140.000000	8321.056600	1808.483617	9144.000000	914.400000	83.800000

In [623]:

```
# Check how many Nans we have in each column

data[['Height (cm)', 'Length (cm)', 'Depth (cm)', 'Width (cm)', 'Diameter (cm)',
```

Out[623]:

```
Height (cm)             18369
Length (cm)            129526
Depth (cm)             118819
Width (cm)              19259
Diameter (cm)          128863
Circumference (cm)     130252
dtype: int64
```

In [624]:

```
# let's hand code our quantiles that we will use based on the percentiles we saw

height_quantiles = [0,18.10, 27.94, 44.45, 1000.00, 9140.00]
length_quantiles = [0,17.03, 26.70, 79.10, 1000.00, 8321.056600]
depth_quantiles = [0,0.70000,13.34, 1000.00, 1808.483617]
width_quantiles = [0,17.80, 25.40, 44.80, 1000.00, 9144.00]
diameter_quantiles = [0.635, 7.90, 13.70, 24.783, 914.40]
circumference_quantiles = [9.90, 23.50, 36.00, 71.13, 83.80]



# create our bins and attribute them to a new _bin column in each case

data['height_bin'] = pd.cut(data['Height (cm)'], bins = height_quantiles, include

data['length_bin'] = pd.cut(data['Length (cm)'], bins = length_quantiles, include

data['depth_bin'] = pd.cut(data['Depth (cm)'], bins = depth_quantiles, include_lc

data['width_bin'] = pd.cut(data['Width (cm)'], bins = width_quantiles, include_lc

data['diameter_bin'] = pd.cut(data['Diameter (cm)'], bins = diameter_quantiles, i

data['circumference_bin'] = pd.cut(data['Circumference (cm)'], bins = circumferen
```

In [625]:

```
# Let's take a look at an example to see how our bin counts look

data.height_bin.value_counts()
```

Out[625]:

```
3    28221
1    28187
4    28020
2    27454
5       11
Name: height_bin, dtype: int64
```

In [626]:

```
# Take a look to check that the number of Nans in the height column and the heigh
# They are, so the Nans haven't been introduced as an artifact from processing th

data[['width_bin', 'Width (cm)']].isnull().sum()
```

Out[626]:

```
width_bin     19259
Width (cm)    19259
dtype: int64
```

Now we need to create the new categories to hold the Nan values. This is done in the cell below by adding a new category by using the pd.cat method.

In [627]:

```
data.height_bin = data.height_bin.cat.add_categories([0])

data.length_bin = data.length_bin.cat.add_categories([0])

data.width_bin = data.width_bin.cat.add_categories([0])

data.depth_bin = data.depth_bin.cat.add_categories([0])

data.circumference_bin = data.circumference_bin.cat.add_categories([0])

data.diameter_bin = data.diameter_bin.cat.add_categories([0])
```

Now we fill in the Nan values with 0 to indicate that they are distinct from the other data.

In [628]:

```
data.diameter_bin = data.diameter_bin.fillna(0)

data.height_bin = data.height_bin.fillna(0)

data.length_bin = data.length_bin.fillna(0)

data.width_bin = data.width_bin.fillna(0)

data.depth_bin = data.depth_bin.fillna(0)

data.circumference_bin = data.circumference_bin.fillna(0)
```

In [629]:

```
data.head()
```

Out[629]:

	Artwork ID	Title	Artist ID	Name	Date	Medium	Dimensions	Acquisition Date
0	2	Ferdinandsbrücke Project, Vienna, Austria, Ele...	6210	Otto Wagner	1896	Ink and cut-and-pasted painted pages on paper	19 1/8 x 66 1/2" (48.6 x 168.9 cm)	1996-04-09
1	3	City of Music, National Superior Conservatory ...	7470	Christian de Portzamparc	1987	Paint and colored pencil on print	16 x 11 3/4" (40.6 x 29.8 cm)	1995-01-17
2	4	Villa near Vienna Project, Outside Vienna, Aus...	7605	Emil Hoppe	1903	Graphite, pen, color pencil, ink, and gouache ...	13 1/2 x 12 1/2" (34.3 x 31.8 cm)	1997-01-15
3	5	The Manhattan Transcripts Project, New York, N...	7056	Bernard Tschumi	1980	Photographic reproduction with colored synthet...	20 x 20" (50.8 x 50.8 cm)	1995-01-17
4	6	Villa, project, outside Vienna, Austria, Exter...	7605	Emil Hoppe	1903	Graphite, color pencil, ink, and gouache on tr...	15 1/8 x 7 1/2" (38.4 x 19.1 cm)	1997-01-15

5 rows × 54 columns

E. Encoding the Catalogue Column

The catalogue column may grant us some value for our classifer. It is determined on the basis of whether or not the

In [630]:

```
data.Catalogue.replace(('Y', 'N'), (1, 0), inplace=True)
```

In [631]:

```
data.head()
```

Out[631]:

	Artwork ID	Title	Artist ID	Name	Date	Medium	Dimensions	Acquisition Date
0	2	Ferdinandsbrücke Project, Vienna, Austria, Ele...	6210	Otto Wagner	1896	Ink and cut-and-pasted painted pages on paper	19 1/8 x 66 1/2" (48.6 x 168.9 cm)	1996-04-09
1	3	City of Music, National Superior Conservatory ...	7470	Christian de Portzamparc	1987	Paint and colored pencil on print	16 x 11 3/4" (40.6 x 29.8 cm)	1995-01-17
2	4	Villa near Vienna Project, Outside Vienna, Aus...	7605	Emil Hoppe	1903	Graphite, pen, color pencil, ink, and gouache ...	13 1/2 x 12 1/2" (34.3 x 31.8 cm)	1997-01-15
3	5	The Manhattan Transcripts Project, New York, N...	7056	Bernard Tschumi	1980	Photographic reproduction with colored synthet...	20 x 20" (50.8 x 50.8 cm)	1995-01-17
4	6	Villa, project, outside Vienna, Austria, Exter...	7605	Emil Hoppe	1903	Graphite, color pencil, ink, and gouache on tr...	15 1/8 x 7 1/2" (38.4 x 19.1 cm)	1997-01-15

5 rows × 54 columns

E. Encoding the Credit column

As the Credit column is our target, we'll take a slightly different approach and encode it to numerical variables. As there are a lot of them, we'll want to retain the Credit column so it can act as a handy key for us. We will also want to go with the top 30 donors so that our classifer can converge.

In [423]:

```
# Ugh I need to select the top 30 donors but it's not working.  I'm doing somethi
# data = data[data['Credit'][0:30]]
```

In [689]:

```
data.head()
```

Out[689]:

	Artwork ID	Title	Artist ID	Name	Date	Medium	Dimensions	Acquisition Date	
1	3	City of Music, National Superior Conservatory ...	7470	Christian de Portzamparc	1987	Paint and colored pencil on print	16 x 11 3/4" (40.6 x 29.8 cm)	1995-01-17	Gi a i Au
2	4	Villa near Vienna Project, Outside Vienna, Aus...	7605	Emil Hoppe	1903	Graphite, pen, color pencil, ink, and gouache ...	13 1/2 x 12 1/2" (34.3 x 31.8 cm)	1997-01-15	G Ri
3	5	The Manhattan Transcripts Project, New York, N...	7056	Bernard Tschumi	1980	Photographic reproduction with colored synthet...	20 x 20" (50.8 x 50.8 cm)	1995-01-17	P pa a
4	6	Villa, project, outside Vienna, Austria, Exter...	7605	Emil Hoppe	1903	Graphite, color pencil, ink, and gouache on tr...	15 1/8 x 7 1/2" (38.4 x 19.1 cm)	1997-01-15	G Ri
5	7	The Manhattan Transcripts Project, New York, N...	7056	Bernard Tschumi	1976	Gelatin silver photograph	14 x 18" (35.6 x 45.7 cm)	1995-01-17	P pa a

5 rows × 55 columns

In [690]:

```
data['Credit'] = data['Credit'].astype('category')
data.dtypes
```

```
/Users/patrickbrown/anaconda/lib/python2.7/site-packages/ipykernel_l
auncher.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/panda
s-docs/stable/indexing.html#indexing-view-versus-copy
(http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-
view-versus-copy)
  """Entry point for launching an IPython kernel.
```

Out[690]:

```
Artwork ID                                        int64
```

```
Title                                                   object
Artist ID                                               object
Name                                                    object
Date                                                    object
Medium                                                  object
Dimensions                                              object
Acquisition Date                                datetime64[ns]
Credit                                                category
Catalogue                                                int64
Department                                              object
Object Number                                           object
Diameter (cm)                                          float64
Circumference (cm)                                     float64
Height (cm)                                            float64
Length (cm)                                            float64
Width (cm)                                             float64
Depth (cm)                                             float64
Weight (kg)                                            float64
Duration (s)                                           float64
class_(not assigned)                                     uint8
class_Architecture                                       uint8
class_Audio                                              uint8
class_Collage                                            uint8
class_Design                                             uint8
class_Drawing                                            uint8
class_Ephemera                                           uint8
class_Film                                               uint8
class_Film (object)                                      uint8
class_Frank Lloyd Wright Archive                         uint8
class_Furniture and Interiors                            uint8
class_Illustrated Book                                   uint8
class_Installation                                       uint8
class_Media                                              uint8
class_Mies van der Rohe Archive                          uint8
class_Multiple                                           uint8
class_Painting                                           uint8
class_Performance                                        uint8
class_Periodical                                         uint8
class_Photograph                                         uint8
class_Photography Research/Reference                     uint8
class_Print                                              uint8
class_Product Design                                     uint8
class_Sculpture                                          uint8
class_Software                                           uint8
class_Textile                                            uint8
class_Video                                              uint8
class_Work on Paper                                      uint8
height_bin                                            category
length_bin                                            category
depth_bin                                             category
width_bin                                             category
diameter_bin                                          category
circumference_bin                                     category
Gender                                                  object
dtype: object
```

In [691]:

```
data['Credit_Code'] = data['Credit'].cat.codes
data.head(20)
```

```
/Users/patrickbrown/anaconda/lib/python2.7/site-packages/ipykernel_l
auncher.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/panda
s-docs/stable/indexing.html#indexing-view-versus-copy
(http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-
view-versus-copy)
  """Entry point for launching an IPython kernel.
```

Out[691]:

	Artwork ID	Title	Artist ID	Name	Date	Medium	Dimensions	Acquisition Date	Credit	Catalog
1	3	City of Music, National Superior	7470	Christian de Portzamparc	1987	Paint and colored pencil on	16 x 11 3/4" (40.6 x 29.8 cm)	1995-01-17	Gift of the architect in honor of Lily	

In [107]:

```
data['Credit_Code'].value_counts()
```

Out[107]:

```
 6821    10927
 6287     8398
 5522     7191
 4        4889
 6778     4603
-1        3070
 5676     2822
 6802     2474
 2872     2266
 1158     1897
 5723     1454
 5865     1371
 6064     1349
 5642     1318
 1064     1307
 627      1254
 5665     1221
 6063     1148
 4733     1133
 6079     1055
 3637      952
 5815      949
 5         929
 5833      825
 5992      821
 6762      756
```

```
6773      748
820       690
493       669
5510      667
         ...
2084        1
1636        1
37          1
6182        1
4135        1
2148        1
101         1
6246        1
4199        1
5734        1
1572        1
6951        1
5222        1
868         1
7015        1
996         1
1060        1
3109        1
1124        1
3173        1
5286        1
3557        1
1252        1
3365        1
1380        1
3429        1
5478        1
1444        1
1508        1
3998        1
Name: Credit_Code, Length: 7031, dtype: int64
```

In [692]:

```
grouped_credit_df = data.groupby('Credit')[['Credit','Credit_Code','Name', 'Title
pd.DataFrame(grouped_credit_df)
```

Out[692]:

	Credit	Credit_Code	Name	Title
1	Gift of the architect in honor of Lily Auchinc...	5490	Christian de Portzamparc	City of Music, National Superior Conservatory ...
2	Gift of Jo Carole and Ronald S. Lauder	2675	Emil Hoppe	Villa near Vienna Project, Outside Vienna, Aus...
3	Purchase and partial gift of the architect in ...	6293	Bernard Tschumi	The Manhattan Transcripts Project, New York, N...
4	Gift of Jo Carole and Ronald S. Lauder	2675	Emil Hoppe	Villa, project, outside Vienna, Austria, Exter...
5	Purchase and partial gift of the architect in ...	6293	Bernard Tschumi	The Manhattan Transcripts Project, New York, N...

F. Changing the Date (of artwork) Column to Datetime

We want to know if there is a preference for art from a certain period. The best way to obtain value from this column is to bin the dates of works.

In [637]:

```
data.Date.unique()
```

Out[637]:

```
array([u'1987', u'1903', u'1980', ..., u'September 8, 1962',
       u'1954\u20131956', u'1955\u20131967'], dtype=object)
```

In [632]:

```
# We cannot infer datetime from years prior to 1900 so let's remove the 1896 work

data = data[data['Date'] != '1896']
```

In [638]:

```
data.Date.dtype
```

Out[638]:

```
dtype('O')
```

In [652]:

```
# Let's look at the patterns of data in the date column

data.Date.isnull().value_counts()
```

Out[652]:

```
False    126917
Name: Date, dtype: int64
```

In the Date column, we have five patterns of data format:

- Pattern 1a: '1976-77' (year ranges)
- Pattern 1b: '1930 - 1931' (year range in a slightly different format)
- Pattern 1c: '1930, published 1931' (another year range or multiple dates in the same field)
- Pattern 2: 'c.1917' (circa a particular year)
- Pattern 3: 'Unknown'
- Pattern 4: 'n.d.'

Our proposed approach will be to remove the year ranges via regex, and removey the 'c.' from the date column to leave the year. We'll also change the 'n.d.' values to 'Unknown'. We could backfill or forward fill years, but clearly the dates in each case would need to relate to the artist's liftime dates rather than, say, the date of the previous work, or the mean or median date of the entire dataset. One way to perhaps do this would be the former, but a little beyond my dark arts at the moment!

In [654]:

```
# remove our 'c.' patterns

data.Date = data.Date.str.replace('c.','')

# change 'n.d's to 'Unknowns'

data.Date = data.Date.str.replace('n.d.','Unknown')

# remove our date ranges leaving the first year in the range as the value

data.Date = data.Date.str.replace('[-][0-9]+','')

# remove other date ranges

data.Date = data.Date.str.replace('[-][0-9][0-9]+','')
```

In [651]:

```
# To get meaning from our data, we will have to drop the unknowns

data = data[data['Date'] != 'Unknown']

# and the Nans

data.Date.dropna(inplace=True)
```

In [656]:

```
data = data[data['Date'].value_counts() > 1]
```

```
/Users/patrickbrown/anaconda/lib/python2.7/site-packages/ipykernel_l
auncher.py:1: UserWarning: Boolean Series key will be reindexed to m
atch DataFrame index.
  """Entry point for launching an IPython kernel.

---------------------------------------------------------------------
-------
IndexingError                             Traceback (most recent cal
l last)
<ipython-input-656-f98d7d332867> in <module>()
----> 1 data = data[data['Date'].value_counts() > 1]

/Users/patrickbrown/anaconda/lib/python2.7/site-packages/pandas/core
/frame.pyc in __getitem__(self, key)
   2054         if isinstance(key, (Series, np.ndarray, Index, list)
):
   2055             # either boolean or fancy integer index
-> 2056             return self._getitem_array(key)
   2057         elif isinstance(key, DataFrame):
   2058             return self._getitem_frame(key)

/Users/patrickbrown/anaconda/lib/python2.7/site-packages/pandas/core
/frame.pyc in _getitem_array(self, key)
   2094             # check_bool_indexer will throw exception if Ser
ies key cannot
   2095             # be reindexed to match DataFrame rows
-> 2096             key = check_bool_indexer(self.index, key)
   2097             indexer = key.nonzero()[0]
   2098             return self.take(indexer, axis=0, convert=False)

/Users/patrickbrown/anaconda/lib/python2.7/site-packages/pandas/core
/indexing.pyc in check_bool_indexer(ax, key)
   1937         mask = isnull(result._values)
   1938         if mask.any():
-> 1939             raise IndexingError('Unalignable boolean Series
provided as '
   1940                                 'indexer (index of the boole
an Series and of '
   1941                                 'the indexed object do not m
atch')

IndexingError: Unalignable boolean Series provided as indexer (index
of the boolean Series and of the indexed object do not match
```

In [657]:

```
data.Date.value_counts()
```

Out[657]:

```
1971                                        1889
1967                                        1839
1966                                        1746
```

1968	1735
1969	1664
1965	1615
1973	1567
1970	1533
1964	1504
1930	1384
1972	1297
1900	1291
1928	1289
1931	1280
1962	1244
1963	1240
2003	1214
1926	1198
2001	1142
1980	1117
1976	1109
1925	1089
1948	1079
1991	1058
1927	1041
2002	1041
1994	1029
1975	1021
1934	999
1983	997
	...
(January 30-February 1, 1963)	1
(January 10) 1969	1
September 2, 1942	1
(February 7) 1963	1
1950. (Print exeted 1944).	1
April 18, 1914	1
1930. (Prints exeted 1920).	1
1889, published later	1
Mar 14, 1977	1
November 21, 1935	1
November 3, 1917	1
(after 1923)	1
(newspaper published August 23, 2004)	1
June 6, 1944	1
1962. (Prints exeted 1959).	1
1996/98	1
(April 9-June 24, 1970)	1
(newspaper published November 13/14, 2004)	1
(February 18, 1963)	1
November 3, 1916	1
1965. (Work exeted 1964).	1
Paris, 1900	1
July 4, 1937	1
May 29, 1980	1
February 3, 1917	1
(Published 1968)	1
(Mar 22) 1963	1
published 1969	1
June 30, 1917	1

```
(February 23, 1973-Mar 20, 1974)                    1
Name: Date, Length: 7300, dtype: int64
```

In [653]:

```
data['Date'] = pd.to_datetime(data['Date'], infer_datetime_format=True)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent cal
l last)
<ipython-input-653-bb9d32dad45d> in <module>()
----> 1 data['Date'] = pd.to_datetime(data['Date'],
infer_datetime_format=True)

/Users/patrickbrown/anaconda/lib/python2.7/site-packages/pandas/core
/tools/datetimes.pyc in to_datetime(arg, errors, dayfirst, yearfirst
, utc, box, format, exact, unit, infer_datetime_format, origin)
    507     elif isinstance(arg, ABCSeries):
    508         from pandas import Series
--> 509         values = _convert_listlike(arg._values, False,
format)
    510         result = Series(values, index=arg.index, name=arg.na
me)
    511     elif isinstance(arg, (ABCDataFrame, MutableMapping)):

/Users/patrickbrown/anaconda/lib/python2.7/site-packages/pandas/core
/tools/datetimes.pyc in _convert_listlike(arg, box, format, name, tz
)
    445                 return DatetimeIndex._simple_new(values,
name=name, tz=tz)
    446             except (ValueError, TypeError):
--> 447                 raise e
    448 
    449     if arg is None:

ValueError: Unknown string format
```

G. Acquisition Dates

Using acquisition dates as a feature would be problematic, as the below groupby query shows. There are a number of gifts that were bulk acquired by a single donor on a single date. This risks collinearity and therefore we will not use in our final dataframe.

In [658]:

```
# remove one clear outlier in dates (Moma certainly wasn't in existence in the 13

data = data[data['Acquisition Date'] != '1216-10-18']
```

In [661]:

```
data['Acquisition Date'] = pd.to_datetime(data['Acquisition Date'], infer_datetim
```

```
/Users/patrickbrown/anaconda/lib/python2.7/site-packages/ipykernel_l
auncher.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/panda
s-docs/stable/indexing.html#indexing-view-versus-copy
(http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-
view-versus-copy)
  """Entry point for launching an IPython kernel.
```

In [662]:

```
data['Acquisition Date'].unique()
```

Out[662]:

```
array(['1995-01-17T00:00:00.000000000', '1997-01-15T00:00:00.0000000
00',
       '1966-01-11T00:00:00.000000000', ...,
       '2016-10-18T00:00:00.000000000', '2016-10-01T00:00:00.0000000
00',
       '2016-05-01T00:00:00.000000000'], dtype='datetime64[ns]')
```

In [663]:

```
data["Acquisition Date"].dtype
```

Out[663]:

```
dtype('<M8[ns]')
```

In [664]:

```
data.head()
```

Out[664]:

	Artwork ID	Title	Artist ID	Name	Date	Medium	Dimensions	Acquisition Date	
1	3	City of Music, National Superior Conservatory ...	7470	Christian de Portzamparc	1987	Paint and colored pencil on print	16 x 11 3/4" (40.6 x 29.8 cm)	1995-01-17	Gi a i Au
2	4	Villa near Vienna Project, Outside Vienna, Aus...	7605	Emil Hoppe	1903	Graphite, pen, color pencil, ink, and gouache ...	13 1/2 x 12 1/2" (34.3 x 31.8 cm)	1997-01-15	G R
3	5	The Manhattan Transcripts Project, New York, N...	7056	Bernard Tschumi	1980	Photographic reproduction with colored synthet...	20 x 20" (50.8 x 50.8 cm)	1995-01-17	P pa a
4	6	Villa, project, outside Vienna, Austria, Exter...	7605	Emil Hoppe	1903	Graphite, color pencil, ink, and gouache on tr...	15 1/8 x 7 1/2" (38.4 x 19.1 cm)	1997-01-15	G R
5	7	The Manhattan Transcripts Project, New York, N...	7056	Bernard Tschumi	1976	Gelatin silver photograph	14 x 18" (35.6 x 45.7 cm)	1995-01-17	P pa a

5 rows × 54 columns

H. Accounting for Gender

It's been a perennial concern in the western canon of art history: 'why are there no great women artists?'. In art created prior to the 20th century, men certainly are overrepresented as a proportion of the canon. However, we wonder if some donors might favour one gender or another.

We don't have a gender column. However, there is a module called sexmachine that can help us here. While it does not use machine learning, as it looks up names from a table, it can help us to infer gender with some fairly okay accuracy:

There is more on gender and data and the merits and demerits of sexmachine here: https://civic.mit.edu/blog/natematias/best-practices-for-ethical-gender-research-at-very-large-scales (https://civic.mit.edu/blog/natematias/best-practices-for-ethical-gender-research-at-very-large-scales)

In [670]:

```
import sexmachine.detector as gender

# run first: pip install -i https://pypi.anaconda.org/pypi/simple sexmachine

g = gender.Detector()

def infer_gender(name):
    try:
        return g.get_gender(name.split()[0])
    except:
        return
```

In [677]:

```
firsts = data.drop_duplicates('Name')
```

In [678]:

```
firsts.loc[:, 'Gender'] = firsts['Name'].apply(infer_gender)
firsts.groupby('Gender').size()
```

Out[678]:

```
Gender
andy              2473
female            1806
male              8649
mostly_female      239
mostly_male        293
dtype: int64
```

So men are overrepresented. Let's rerun this to create a new column with the relevant data. The only caveat is that of the 12921 artists in the collection, nearly a quarter have first names whose gender our module cannot identify. However, we can proceed to use this module on the basis that a) we haven't found anything else that does this, and b) we can make a reasonable assumption that the module categorises a female name as 'andy' (for androgynous) as often as it does a male name.

The key takeaway is that men dominate the collection.

In [679]:

```
data.loc[:, 'Gender'] = data['Name'].apply(infer_gender)
```

In [680]:

```
data.groupby('Gender').size()
```

Out[680]:

```
Gender
andy             14725
female           16538
male             91506
mostly_female     1547
mostly_male       3452
dtype: int64
```

In [682]:

```
data.head()
```

Out[682]:

	Artwork ID	Title	Artist ID	Name	Date	Medium	Dimensions	Acquisition Date	
1	3	City of Music, National Superior Conservatory ...	7470	Christian de Portzamparc	1987	Paint and colored pencil on print	16 x 11 3/4" (40.6 x 29.8 cm)	1995-01-17	Gi a i Au
2	4	Villa near Vienna Project, Outside Vienna, Aus...	7605	Emil Hoppe	1903	Graphite, pen, color pencil, ink, and gouache ...	13 1/2 x 12 1/2" (34.3 x 31.8 cm)	1997-01-15	G R
3	5	The Manhattan Transcripts Project, New York, N...	7056	Bernard Tschumi	1980	Photographic reproduction with colored synthet...	20 x 20" (50.8 x 50.8 cm)	1995-01-17	P pa a
4	6	Villa, project, outside Vienna, Austria, Exter...	7605	Emil Hoppe	1903	Graphite, color pencil, ink, and gouache on tr...	15 1/8 x 7 1/2" (38.4 x 19.1 cm)	1997-01-15	G R
5	7	The Manhattan Transcripts Project, New York, N...	7056	Bernard Tschumi	1976	Gelatin silver photograph	14 x 18" (35.6 x 45.7 cm)	1995-01-17	P pa a

5 rows × 55 columns

5. Data Modelling

In [693]:

```
data['Credit'].value_counts()
```

Out[693]:

```
The Louis E. Stern Collection
10927
Purchase
8344
Gift of the artist
7183
Abbott-Levy Collection. Partial gift of Shirley C. Burden
4847
The Gilbert and Lila Silverman Fluxus Collection Gift
4585
Gift of the photographer
2788
The Judith Rothschild Foundation Contemporary Drawings Collection Gi
ft
2473
Gift of Kleiner, Bell & Co.
2266
Gift of Abby Aldrich Rockefeller
```

Binning the data may help us to extract more value from the data by separating it into discrete 'bins' for analysis. Let's do that for the Acquisition date

Tidy up the Column Names

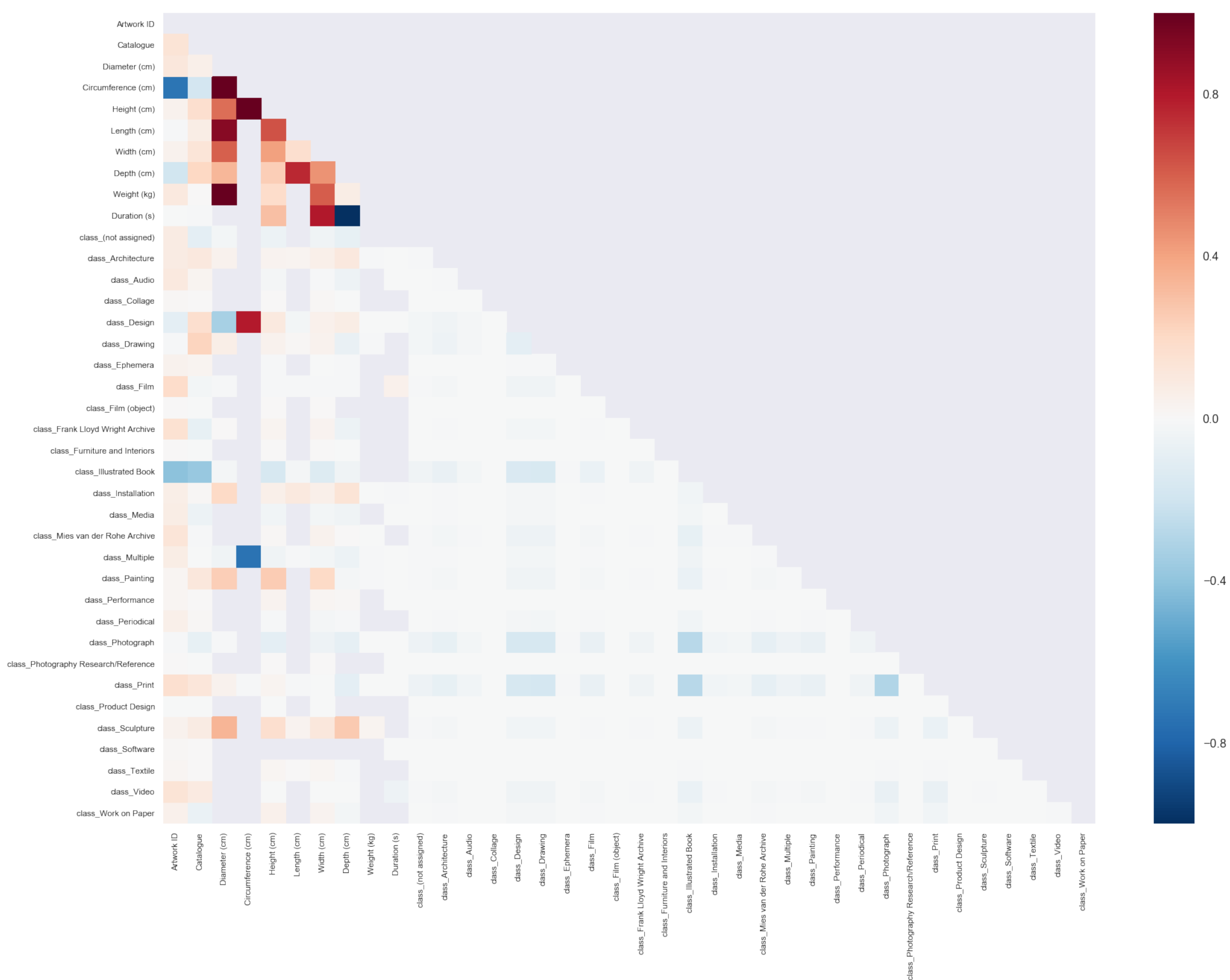
In [ ]:

```
# data.columns = ['artwork_id', 'title', 'artist_id', 'name', 'date', 'medium', '
```

Let's Plot our Correlations

In [683]:

```
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np

mean_corr = data.corr() #Set your correlation matrix.

# Set the default matplotlib figure size:
fig, ax = plt.subplots(figsize=(28,20))

# Generate a mask for the upper triangle (taken from seaborn example gallery)
mask = np.zeros_like(mean_corr, dtype=np.bool)
mask[np.triu_indices_from(mask)] = True

# Plot the heatmap with seaborn.
# Assign the matplotlib axis the function returns. This will let us resize the la
ax = sns.heatmap(mean_corr, mask=mask, ax=ax)

# Resize the labels.
ax.set_xticklabels(ax.xaxis.get_ticklabels(), fontsize=11, rotation = 90)
ax.set_yticklabels(ax.yaxis.get_ticklabels(), fontsize=11, rotation = 0)

# If you put plt.show() at the bottom, it prevents those useless printouts from m
plt.show()
```



Okay, so we have some preliminary correlations between classes and credit_code which is promising.

In [ ]:

```

```

In [687]:

```
data.columns
```

Out[687]:

```
Index([u'Artwork ID', u'Title', u'Artist ID', u'Name', u'Date', u'Me
dium',
       u'Dimensions', u'Acquisition Date', u'Credit', u'Catalogue',
       u'Department', u'Object Number', u'Diameter (cm)',
       u'Circumference (cm)', u'Height (cm)', u'Length (cm)', u'Widt
h (cm)',
       u'Depth (cm)', u'Weight (kg)', u'Duration (s)', u'class_(not
assigned)',
       u'class_Architecture', u'class_Audio', u'class_Collage',
       u'class_Design', u'class_Drawing', u'class_Ephemera', u'class
_Film',
       u'class_Film (object)', u'class_Frank Lloyd Wright Archive',
       u'class_Furniture and Interiors', u'class_Illustrated Book',
       u'class_Installation', u'class_Media',
       u'class_Mies van der Rohe Archive', u'class_Multiple',
       u'class_Painting', u'class_Performance', u'class_Periodical',
       u'class_Photograph', u'class_Photography Research/Reference',
       u'class_Print', u'class_Product Design', u'class_Sculpture',
       u'class_Software', u'class_Textile', u'class_Video',
       u'class_Work on Paper', u'height_bin', u'length_bin', u'depth
_bin',
       u'width_bin', u'diameter_bin', u'circumference_bin', u'Gender
'],
      dtype='object')
```

In [705]:

```
from sklearn.ensemble import RandomForestClassifier

model = RandomForestClassifier(n_estimators = 100, min_samples_leaf = 50, n_jobs=

# Use `fit` to learn the vocabulary of the titles


# Use `tranform` to generate the sample X word matrix - one column per feature (w
Features = data[['Artwork ID',
                 'Catalogue',
                 'class_(not assigned)',
                 'class_Architecture',
                 'class_Audio',
                 'class_Collage',
                 'class_Design',
                 'class_Drawing',
                 'class_Ephemera',
                 'class_Film',
                 'class_Film (object)',
                 'class_Frank Lloyd Wright Archive',
```

```
                 'class_Furniture and Interiors',
                 'class_Illustrated Book',
                 'class_Installation',
                 'class_Media',
                 'class_Mies van der Rohe Archive',
                 'class_Multiple',
                 'class_Painting',
                 'class_Performance',
                 'class_Periodical',
                 'class_Photograph',
                 'class_Photography Research/Reference',
                 'class_Print',
                 'class_Product Design',
                 'class_Sculpture',
                 'class_Software',
                 'class_Textile',
                 'class_Video',
                 'class_Work on Paper',
                 'height_bin',
                 'length_bin',
                 'depth_bin',
                 'width_bin',
                 'diameter_bin',
                 'circumference_bin']]
y = data.Credit_Code

from sklearn.cross_validation import cross_val_score

scores = cross_val_score(model, Features, y, scoring='accuracy')
print('CV AUC {}, Average AUC {}'.format(scores, scores.mean()))
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-705-ba76c01bee6f> in <module>()
     47 from sklearn.cross_validation import cross_val_score
     48 
---> 49 scores = cross_val_score(model, Features, y, scoring='accuracy')
     50 print('CV AUC {}, Average AUC {}'.format(scores, scores.mean()))

/Users/patrickbrown/anaconda/lib/python2.7/site-packages/sklearn/cross_validation.pyc in cross_val_score(estimator, X, y, scoring, cv, n_jobs, verbose, fit_params, pre_dispatch)
   1569                                               train, test, verbose, None,
   1570                                               fit_params)
-> 1571                       for train, test in cv)
```

In [ ]:

```
# problem - some column entries under Artist ID and Name have two artists in ther
```