Portfolio 3 - assignment

Christian Laursen (bfd962)

Mickey Johansson (mvx394)

Mushtaba Osmani (wgq323)

19264 characters - 8,03 normal pages

17/12 - 19

1. SMK's approach and reasoning for using an API

SMK have an API approach to their data and their online-collection. With this, they have started a project called "SMK Open" (SMK, n.d) which aims at making their art available to everybody. The project and its features, has been made possible by utilising artificial intelligence. Every single work in the SMK online collection, has been categorized. And the whole collection is supposed to be tagged, systematized and accompanied by photos. An enormous task, only made possible with the use of AI (SMK, 2019). Additionally, SMK have made their API, with all the underlying data, available and free to use to the public. In order for anybody to "be able to create their own websites and apps based on the museum's data (Ibid.)". These services will be directly connected to the API and will therefore be automatically updated with the newest information whenever SMK updates their data collection.

As it stands now, not everybody has easy access to a physical museum and even if one were to go through and see every single work in SMK, it would still be less than 1% of the entire collection. By digitizing the art, it would be possible to solve this problem and further improve other aspects such as making it possible to study the works in detail and continue development. It would also be possible to use the art in everything from research-and school projects to printing on posters or pillowcases.

2. Description of our URL and its content

The SMK api (https://api.smk.dk/api/v1/docs#/artworks/searchArt) was used to search for the key word "sculpture". This gave a result of 4006 items, which was limited to 200 items in our JSON-file. We chose 200, because the assignment called for at least 100 items and 200 is still limited enough to be easy and comprehensible to work with. The accompanying url was then copied into a JSON beautifier (Codebeautify, n.d) to easier research the content of the file.

Our final url is as follows:
https://api.smk.dk/api/v1/art/search/?keys=sculpture&offset=0&rows=200&lang=en. We used "keys=sculpture" as our word query and limited our row query to 200. Under api.smk.dk we searched for sculptures under "/art/search/". We chose sculptures as our key word as we wanted to statistically showcase the background of the sculptures included in the SMK API, such as each artist's age when creating the sculpture and in what time period the sculptures were created. In continuation one would expect sculptures from many different time periods,

which is also why we chose sculptures under the art search query, as it would indicate how much SMK's API spans in terms of sculpture variety.

a) Different types of metadata in each element:

Every element inherits a large amount of metadata, but this data does not match across every element. For example, some of the elements has "distinguishing_features", "labels" and images of the art.

The metadata that on first glance recurs on most of the elements include:

'Created' - A date resembling when the element was created in the API.

'Credit_line' - The name of the credited people, most often the artist.

'Id' - The ID of the artwork.

'Object_names' - The name of the artwork.

'Production' - Information about the production, including 'creator', 'date' and information about the artist.

Some of the elements do not have a "credit_line", "production" or "production_date", which could be because no such information exists about the artist behind the sculpture or because the SMK open project is still in its beginning phases.

b) Metadata comparison with Dublin Core

We will now attempt to expand on whether our metadata is in compliance with the Dublin Core Metadata Element Set. This element consists of fifteen properties that are used in resource description. The practice of the Dublin Core has been in play since 1998 and aims to standardize the process of creating metadata. The fifteen elements consist of the following: contributor, coverage, creator, date, description, format, identifier, language, publisher, relation, rights, source, subject, title & type (DCIM Usage Board, 2012).

As we examine our JSON file we can argue that the metadata we work with is mostly in compliance with the fifteen elements of the Dublin Core. It has followed the format of including aspects such as creator, date, description, format, rights, title, and type. According to Dublin Core, the creator is the entity responsible for making the resource and in our case that is evidently clear that SMK has followed that process. In terms of date, there is also compliance with Dublin Core as there are start date, end date and period time frame of each sculpture. Our metadata has 'notes' or content description, which in our understanding is in

compliance with the Dublin Core element of description. Most of the sculptures have NaN in place, however, a few have few notes of the sculpture. Format element is in our metadata named dimensions. Language is in our metadata named title_0_language and will is element 'language' include the language of the resource. Rights element in our metadata is following the Dublin Core as it is named 'rights' and includes rights information about various properties that are associated with the resource, which is exactly what the Dublin Core describes 'rights' element as being. In terms of title SMK's metadata is following Dublin Core's description of a 'title' element as well. Lastly, there is the element of 'type' which SMK again includes, however, in an alternate name that is 'materials' under the element of 'techniques'. The 'materials' row in our metadata consists of the materials that are used in the production of each sculpture.

In essence we can conclude that SMK's metadata is in compliance with Dublin Core's fifteen elements, however, it is not a full compliance, as there are aspects that are not followed in identical fashion whether it is excluding some elements or changing their names such as 'title_o_language'.

c) Useful numbers for statistic calculations

There are a few relevant numbers one could choose to look at within the json-file. For example, one could choose to look at the "production_date" field. This reveals from when the art is created and further, from what year/decade most sculptures are created. Together with "creator_date_of_birth" found in "production", it would be possible to find out how old most artists were when they created their art that is represented in the SMK collection. By examining "production_date" and their "start" and "end" further, it would be possible to compare the "credit_line" (the artist) to find out how long it takes each artist to finish their work.

Additionally, with the "dimensions" field, one could inspect the average height and width of each sculpture. Perhaps compare it to the artist to see if there is a size-pattern, if the artist usually creates big or small sculptures.

d) Questions that this dataset could answer

The "credit_line" shows the name of the creator, meaning it is possible, with the value_counts() function, to see what artist has the most sculptures in SMK's database. Furthermore, the "creator_nationality", within the "production" column, enables one to count

what nationality is represented the most. Additionally, in the “techniques”-column, one could look at what techniques and/or materials and/or colors are used most often, when and by whom? Meaning, one could find out what trends were typical for what time period, for example.

3. Cleaning and describing our dataset

To start working with our dataset we first have to import it to a dataframe. We do that using the *requests, pandas* and *json_normalize* modules in python.

```
# import the requests module
import requests
# import pandas
import pandas as pd
# import json normalize module
from pandas.io.json import json_normalize
```

The *requests* module lets us retrieve a response from a URL. If the response is ‘200’, it means that the API requests is successful and we can save the response as a json-file.

```
# retrieves the response for the URL and parameters we are sending
response = requests.get(api_search_url)

# print the response
print(response)
# the response is 200 - which means all is OK

# retrieve the JSON from the response variable and add to the json variable
json = response.json()
```

The json-file is then unpacked, flattened and put into a dataframe using the provided function and the pandas module.

```
def flatten_json(y):
    out = {}

    def flatten(x, name=''):
        if type(x) is dict:
            for a in x:
                flatten(x[a], name + a + '_')
        elif type(x) is list:
            i = 0
            for a in x:
                flatten(a, name + str(i) + '_')
                i += 1
        else:
            out[name[:-1]] = x

    flatten(y)
    return out


dic_flattened = [flatten_json(d) for d in json['items']]
df= json_normalize(dic_flattened)
```

When flattening the data into a dataframe, we get 200 rows and 249 different columns, which each represents a different set of metadata. The data frame contains 49800 cells in total.

```
#Show the amount of rows and columns in the entire dataframe
print(df.shape)

#Show the amount of cells in the dataframe
print(df.size)
```

```
(200, 249)
49800
```

Screenshots of the entire list of columns and the data types are in appendix A.

Many of the columns have missing values, since many elements have different metadata compared to the other elements. We obviously can not analyse a dataset containing many cells with missing data, and many of the columns are not suited for analysis simply due to information they contain and the varying metadata for each element. For instance, we have no need for the column containing the date of the elements' creation in the API. To avoid having to sort through each of the 249 columns manually, we made a rough clean-up of the data, by removing every column that had more than 25 rows with missing data.

```
# Remove coloumns that have don't have atleast 175 rows with non-Nan types
# This gives us an easy overview of columns that we could be interested in
df.dropna(axis=1, thresh=175,inplace=True)
```

This is done to remove the columns with information, that only a limited number of elements had such as *distinguishing features* and *labels,* as we are not interested in analysing data that only concern a limited number of elements. The rough clean-up gave us a dataframe with 24 columns, were the majority of the rows in each column (175 out of 200) contained actual data.

```
print(df.dtypes)
```

```
acquisition_date_precision            object
created                               object
credit_line_0                         object
frame_notes_0                         object
frame_notes_1                         object
has_image                               bool
id                                    object
iiif_manifest                         object
modified                              object
object_names_0_name                   object
object_number                         object
on_display                              bool
production_0_creator                  object
production_0_creator_date_of_birth    object
production_0_creator_lref             object
production_0_creator_nationality      object
production_date_0_end                 object
production_date_0_period              object
production_date_0_start               object
public_domain                           bool
responsible_department                object
rights                                object
titles_0_language                     object
titles_0_title                        object
dtype: object
```

This smaller set of data is easier to grasp and thus selecting specific columns that we find useful in our analysis is possible. We made a new dataframe only consisting of the columns we were interested in.

```
# Creating a new dataframe only consisting of the columns we want
nydf = df[["credit_line_0","has_image",'production_0_creator','production_0_
```

We then changed the names of our columns to better match the information they contained.

```
# renaming columns to more fitting names
nydf.rename(columns={'credit_line_0':'creator','production_0_creator':'name','production_0_creator_date_of_birth':'birthyear','pr
print(nydf.columns)
```

```
Index(['creator', 'has_image', 'name', 'birthyear', 'nationality', 'py_start',
       'py_end', 'public_domain', 'language', 'title'],
      dtype='object')
```

The columns ‘py_start’, ‘py_end’ and ‘birthyear’ contain dates that look like this:

```
                birthyear
1943-12-25T00:00:00.000Z
```

Since we are only interested in the specific years, we slice the data in those columns to only contain the first four characters, which is the year.

```
# Slicing the data of coloumns with dates, to only have a year
nydf.loc[:, 'py_start'] = nydf['py_start'].str[:4]
nydf.loc[:, 'py_end'] = nydf['py_end'].str[:4]
nydf.loc[:, 'birthyear'] = nydf['birthyear'].str[:4]
```

The column ‘creator’ contained a ‘©’-symbol in front the creators name, which we also removed.

```
# Slicing the © and 'space' away from the creator columns
nydf.loc[:, 'creator'] = nydf['creator'].str[2:]
```

The final step of our clean-up process was to change the types the ‘py_start’, ‘py_end’ and ‘birthyear’-columns to be numeric. This would allow us to perform statistical analysis of those columns.

```
# Changing datatype of 3 columns to numeric:
nydf.loc[:,'py_start'] = pd.to_numeric(nydf['py_start'])
nydf.loc[:,'py_end'] = pd.to_numeric(nydf['py_end'])
nydf.loc[:,'birthyear'] = pd.to_numeric(nydf['birthyear'])
```

Our cleaned up dataframe ended up looking like this:

```
creator              object
has_image              bool
name                 object
birthyear           float64
nationality          object
py_start            float64
py_end              float64
public_domain          bool
language             object
title                object
Age_when_created    float64
dtype: object
```

As stated earlier, the information about each element in the dataset varies a lot. The amount of elements in each column that had missing data is shown here:

```
print(nydf.isna().sum())
```

```
creator             24
has_image            0
name                18
birthyear           19
nationality         22
py_start            18
py_end              18
public_domain        0
language            13
title               13
Age_when_created    19
dtype: int64
```

4. Analysis and results

After processing and cleaning our data, we made a few statistical findings. Firstly we used value counts on our language column to extract the language of each sculpture title. Not surprisingly danish dominated the sculpture title language with 177 titles being danish. This is as a result of SMK being a danish institute which also means that most art found is danish. English, german and french are other languages that are,

```
#Printing info about the languages
print('Amount of sculpture titles in a specific language:\n',nydf.language.value_counts())
print('Amount of sculptures without a specified language:\n',nydf.language.isna().sum())
```

```
Amount of sculpture titles in a specific language:
 da-DK    177
en-US      6
de-DE      2
fr-FR      2
Name: language, dtype: int64
Amount of sculptures without a specified language:
 13
```

Afterwards we wanted to produce a chart that would visualize the year our artist were bount. For this we make a new dataframe containing the birthyear of each artist and count the year they were born. Afterwards we create an x-axis which is the birth year of the artists, and a y-axis that consists of the amount of artists born in that specific birth year, which we extract from our count value.

```
# New dataframe with count of the artists' birthyear
dfb = pd.DataFrame(nydf['birthyear'].value_counts().reset_index().values,columns=['Birthyear',"Count"])
dfb = dfb.sort_index(axis = 0, ascending=True)
dfb = dfb.astype(int)
```

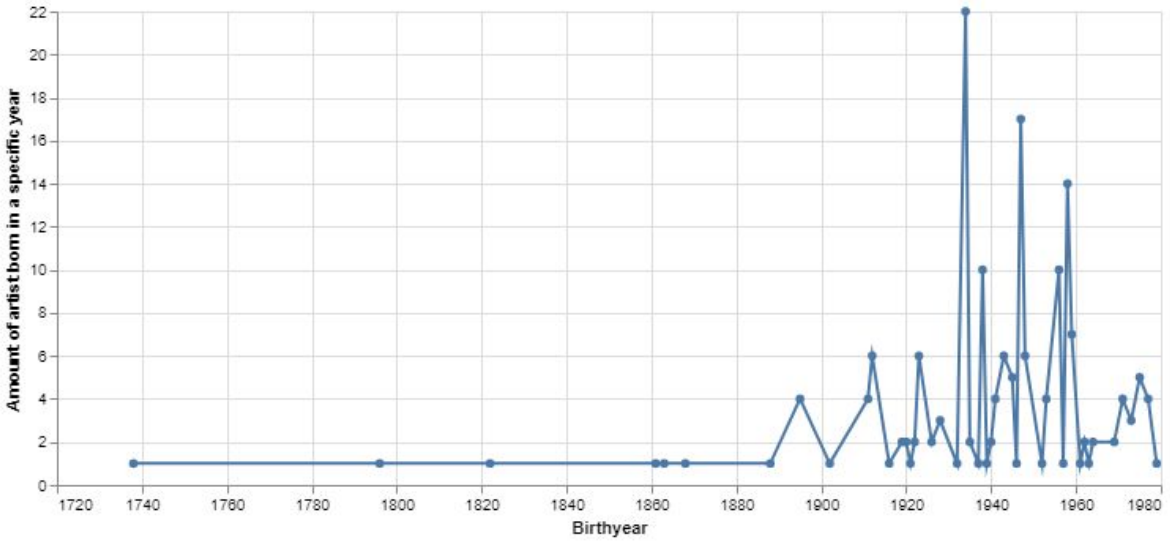
```
# import the altair code library
import altair as alt
alt.renderers.enable('notebook') # hvis I bruger Jupyter Notebook

# create a line plot
alt.Chart(dfb).mark_line(point=True).encode(
        # Birthyear on the X axis with encoding data type Q for quantitative and format c to display without decimal
        x=alt.X('Birthyear:Q', axis=alt.Axis(format='c', title='Birthyear')),

        # Number of artist born in a specific year on the Y axis  with encoding data type Q for quantitative
        y=alt.Y('Count:Q', axis=alt.Axis(title='Amount of artist born in a specific year')),

    # determine the width and hight
    ).properties(width=700, height=300)
```

What we found was that only one artist was born from the years of about 1740, 1800, 1820,1860,1870. It began to spike about year 1890 where we found 4 artists were born in that timeframe. Afterwards the chart dips and rises until about the year 1930-1940 where the chart reaches its peak, as there at one year was born 22 artists. After that point it again dips and rises, however from 1930 the general amount of artists is a tremendous upswing compared to before 1930. We argue that the reason the amount of artists born in the 20th century compared to the 19th and 18th century is as a result of it being much more accessible to gather data on more recent sculptures and artists. SMK open is an ongoing project as well and will be finished in 2020 (SMK Open, n.d) which indicates that the data is not complete and that perhaps more sculptures from before the 20th century is still in the progress of being added. Furthermore, we count artists of each sculpture and some sculptures are made from the same artist which is why the peak of the chart plausibly consists of one or a few artists born in that year that have created 22 sculptures in 1930.



We examined what age the artists had when they created the sculptures. First the age was calculated by taking the year in which production started (py_start) of each element and subtract the birth year of that elements' artist. We added the ages to our dataframe in a new column.

```
# Calculate the artists' age when they created a sculpture
age_when_created = nydf.py_start - nydf.birthyear
# Add the ages as a new column in out dataframe
nydf['Age_when_created'] = age_when_created
print(nydf.head())
```

We then used this column to create a new dataframe containing the frequency of each of the different ages that an artist had when creating a sculpture.

```
# New dataframe with count of the artists' age when the sculpture was created
dfb = pd.DataFrame(nydf['Age_when_created'].value_counts().reset_index().values,columns=['Age_when_created',"Count"])
dfb = dfb.sort_index(axis = 0, ascending=True)
dfb = dfb.astype(int)
```

The top of the frequency table shows that the 5 most frequent ages were 68 with 19 occurrences, 36 with 17 occurrences, 37 with 13 occurrences, 58 with 11 occurrences and 35 with 9 occurrences. A likely reason for many occurrences of the same age, could be that the same artist, therefore the same age, is represented multiple times in SMKs collection.

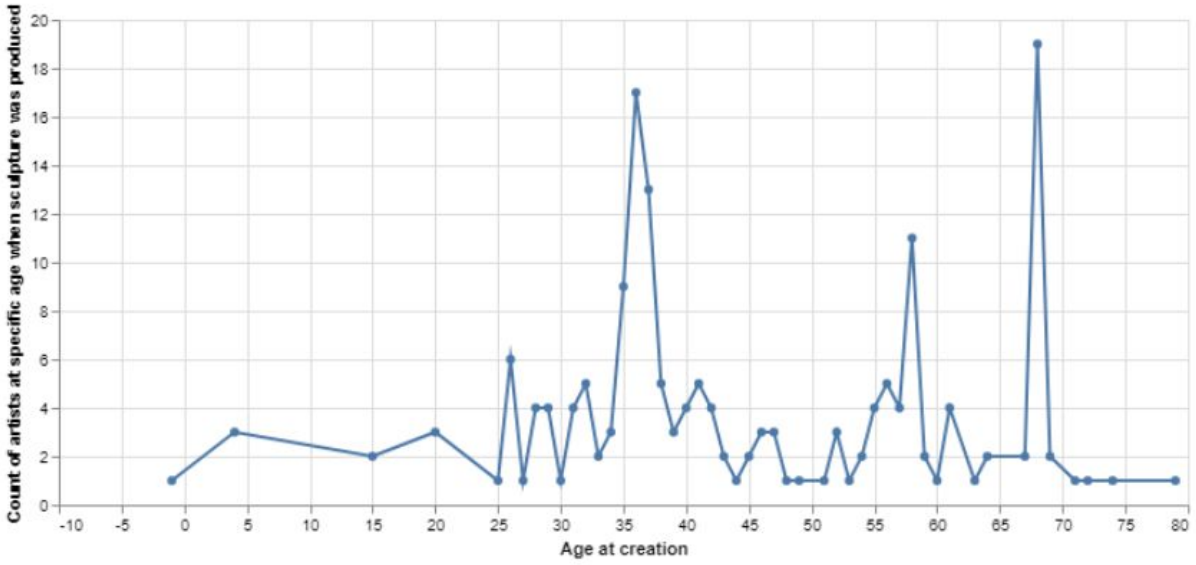
	Age_when_created	Count
0	68	19
1	36	17
2	37	13
3	58	11
4	35	9

We also used the frequency table to create a graph.

```
# create a line plot
alt.Chart(dfb).mark_line(point=True).encode(
        # Age_when_created on the X axis with encoding data type Q for quantitative and format c to display without decimal
        x=alt.X('Age_when_created:Q', axis=alt.Axis(format='c', title='Age at creation')),

        # Number of artist with a specific age when creating a sculpture on the Y axis  with encoding data type Q for quantitati
        y=alt.Y('Count:Q', axis=alt.Axis(title='Count of artists at specific age when sculpture was produced')),

    # determine the width and hight
    ).properties(width=700, height=300)
```



The 3 spikes on the graph clearly show the 5 most frequent ages mentioned above. It also shows that no other age occurs more than 6 times, and most ages occur 4 or less times. The graph shows that the oldest an artist has been when creating a sculpture was 79 years old.

With this, one also notices the anomalies, -1, 4 and 15. The “Age at creation” graph, is dependant on the “production year start” and “birth year”. Meaning the anomalies occur when there is either missing data or faulty data in these. For example, one of Jørgen Gudmundsen-Holmgreens sculptures started production in 1899, but he was born in 1895 and the production was finished in 1952. This would mean that he was 4 years old when he

started his production and that it took 53 years to finish it. Therefore one could point to the production start and end being incorrect.

To further investigate the anomalies, we decided to calculate the age of the artists when they produced a sculpture by using the year in which production finished (py_end), instead of when production began. The ages were added to our dataframe ('Age_when_created2') and we then created a new dataframe with a frequency count of those ages.

```
# Calculate the artists' age when they created a sculpture
age_when_created2 = nydf.py_end - nydf.birthyear
# Add the ages as a new column in out dataframe
nydf['Age_when_created2'] = age_when_created2
# New dataframe with count of the artists' age when the sculpture was created
dfb2 = pd.DataFrame(nydf['Age_when_created2'].value_counts().reset_index().values,columns=['Age_when_created2',"Count"])
dfb2 = dfb2.sort_index(axis = 0, ascending=True)
dfb2 = dfb2.astype(int)
```

The 5 most frequent ages, based on this calculation show a major difference from the other one. For instance, the most frequent age is now 36 with 27 occurrences.

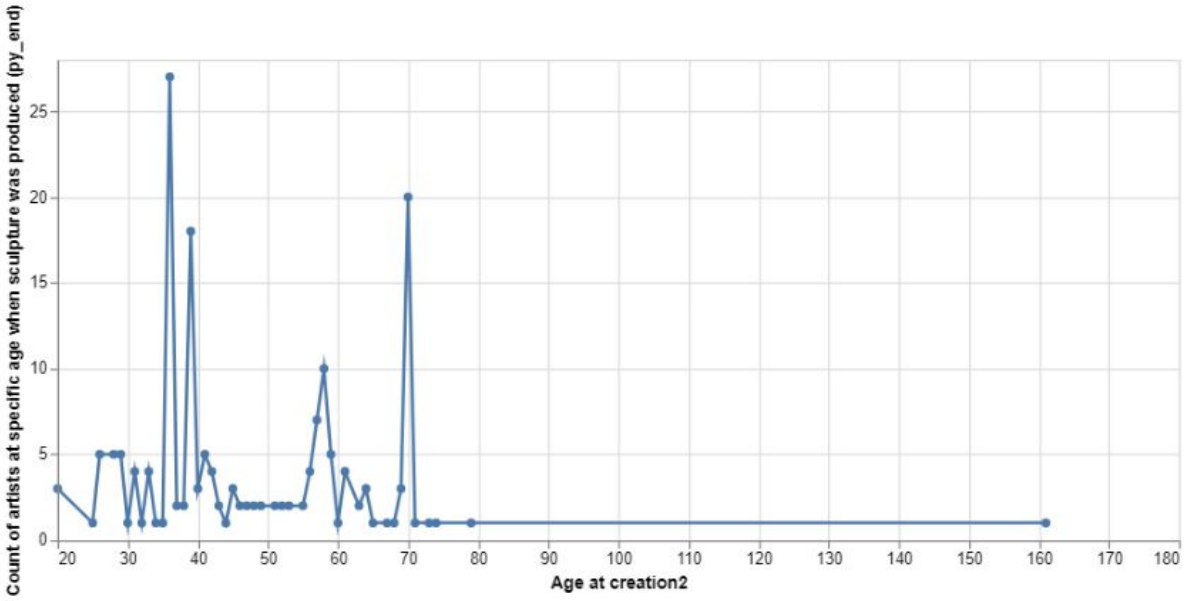
	Age_when_created2	Count
0	36	27
1	70	20
2	39	18
3	58	10
4	57	7

The following graph further illustrates these differences, but also sheds light on another anomaly: One artist being 161 years old when production of the sculpture finished.

```
# create a line plot
alt.Chart(dfb2).mark_line(point=True).encode(
        # Age_when_created on the X axis with encoding data type Q for quantitative and format c to display without decimal
        x=alt.X('Age_when_created2:Q', axis=alt.Axis(format='c', title='Age at creation2')),

        # Number of artist with a specific age when creating a sculpture on the Y axis  with encoding data type Q for quantitati
        y=alt.Y('Count:Q', axis=alt.Axis(title='Count of artists at specific age when sculpture was produced (py_end)')),

    # determine the width and hight
    ).properties(width=700, height=300)
```



Finally, we decided to calculate the time it has taken to produce each sculpture, by subtracting the production start year from the production end year. We added the production time to our dataframe. We print the rows that have a longer production time than 5 years, as those items most likely are the anomalies. In the screenshot below we have highlighted some of the most obvious ones.

```
# Calculate the artists' age when they created a sculpture
years_to_create = nydf.py_end - nydf.py_start
# Add the ages as a new column in out dataframe
nydf['time_to_produce'] = years_to_create
print(nydf.loc[nydf['time_to_produce'] > 5])
```

	creator	has_image	name
22	Robert Jacobsen	True	Jacobsen, Robert
70	NaN	True	Maillol, Aristide
102	Jørgen Gudmundsen-Holmgreen	False	Gudmundsen-Holmgreen, Jørgen
107	Jørgen Gudmundsen-Holmgreen	False	Gudmundsen-Holmgreen, Jørgen
117	Willy Ørskov	False	Ørskov, Willy
121	Jørgen Gudmundsen-Holmgreen	False	Gudmundsen-Holmgreen, Jørgen
124	Jørgen Gudmundsen-Holmgreen	False	Gudmundsen-Holmgreen, Jørgen
150	Richard Winther	True	Winther, Richard
151	Bjørn Nørgaard	True	Nørgaard, Bjørn
164	Richard Winther	False	Winther, Richard
190	Sonja Ferlov Mancoba	False	Ferlov Mancoba, Sonja

	birthyear	nationality	py_start	py_end	public_domain	language
22	1912.0	dansk	1927.0	1981.0	False	da-DK
70	1861.0	fransk	1895.0	1909.0	True	da-DK
102	1895.0	dansk	1899.0	1952.0	False	da-DK
107	1895.0	dansk	1894.0	1965.0	False	da-DK
117	1920.0	dansk	1967.0	1985.0	False	da-DK
121	1895.0	dansk	1899.0	1946.0	False	da-DK
124	1895.0	dansk	1899.0	1952.0	False	da-DK
150	1926.0	dansk	1980.0	1989.0	False	da-DK
151	1947.0	dansk	1962.0	1996.0	False	da-DK
164	1926.0	dansk	1987.0	2087.0	False	da-DK
190	1911.0	dansk	1964.0	1974.0	False	da-DK

	title	Age_when_created
22	Abstrakt figur	15.0
70	Portræt af Madame Maillol. Maske	34.0
102	Udkast til Ikaros	4.0
107	Uden titel	-1.0
117	Tre trommer	47.0
121	Udkast til Araberpige, der bærer et barn	4.0
124	Udkast til Ikaros	4.0
150	Uden titel	54.0
151	Erik Fischer	15.0
164	Uden titel	61.0
190	Maske (Bordmaske eller Maskens fødsel)	53.0

	Age_when_created2	time_to_produce
22	69.0	54.0
70	48.0	14.0
102	57.0	53.0
107	70.0	71.0
117	65.0	18.0
121	51.0	47.0
124	57.0	53.0
150	63.0	9.0
151	49.0	34.0
164	161.0	100.0
190	63.0	10.0

Our own reflection of the three assignments and future work

We have throughout our three portfolio assignments worked with, processed, analyzed and visualized a huge amount of data. In just a few month's time, this was possible with the use of python and its many modules. If we think back and reflect on the amount of work it would take to process and visualize the three datasets manually by ourselves, it would require much more time and would be deemed an excessive waste of time. The tools that we have learned and adapted throughout the course and assignments are in a way invaluable as data is and will always be a crucial part of any workplace, study or business. It is a toolset that once learned can be utilized in almost any capacity. Companies, for instance, have a huge amount of data flow and therefore we could with in the future apply our knowledge in python and process some of this data. We already have experienced in our limited capacity of work that our experience with open data science has been valuable in our respective workplaces.

On the other hand, there is also a lot to apply in terms of future courses and studies. We can in an easier fashion provide descriptive statistics or empirical substance to confirm or debunk certain theoretical viewpoints. Of course, we cant use our newfound toolset in every single study, however, there are certainly a few key areas where we can utilize what we have learned throughout the course. For instance, if we were to conduct interviews, we could make a token analysis or in general, develop a code that could extract key topics and opinions regarding those topics.

Ethical considerations

As we have worked with data it is relevant to discuss the ethical considerations when processing and collecting data. According to the danish body of authority Datatilsynet, regular sensitive information is described as being identification information such as name, address, economic-information, family, work and social information (Datatilsynet, n.d). There are certain elements that would be considered sensitive information such as names and price-class in the Titanic dataset or the names and languages of the artists in the SMK dataset. However, we would argue that the datasets of both SMK and Titanic follow the core guiding principles for scientific data management of FAIR. The principle of FAIR being findable, accessible, interoperable and reusable (Wilkinson, 2016, pp. 4-5). Both datasets are easily findable and accessible to the public, and the danish news corp dataset does not contain any data that is sensitive. If we did process sensitive information in future studies or

our workplaces it is an absolute necessity to follow GDPR and data laws. The most important GDPR laws one should follow are simply asking for permission to process sensitive data, keeping the data secure, allowing the data on people to be accessible if need be (GDPR-info, 2018) and much more but these are what we find to be the most important. The core issue with working with sensitive data is to distinguish whether or not they are classified as sensitive and hence, to ask for permission to process said data. Furthermore to keep that data secure and in the confines of the agreement with the data owners.

In continuation of data laws, we would also argue that one should follow a basic code of ethics for computing professionals. The association for computing machinery has provided such a code, hence we would highlight a couple of key areas that are relevant for us. The first aspect we would like to highlight is to be " honest and trustworthy" and "Fair and take action not to discriminate. (ASM, 2018). We find this important to our portfolios as we would not want to make misleading claims or provide dubious data evidence. This is especially hard as we develop the code and the research questions so there will always be some aspect of bias. Lastly, we want to highlight the aspect of respecting privacy (ASM, 2018). As we described we have not worked with private or sensitive information as the datasets are open and available to the public, however, it is still important to value privacy no matter the data and this is especially important to consider in future work.

	creator	has_image	name
22	Robert Jacobsen	True	Jacobsen, Robert
70	NaN	True	Maillol, Aristide
102	Jørgen Gudmundsen-Holmgreen	False	Gudmundsen-Holmgreen, Jørgen
107	Jørgen Gudmundsen-Holmgreen	False	Gudmundsen-Holmgreen, Jørgen
117	Willy Ørskov	False	Ørskov, Willy
121	Jørgen Gudmundsen-Holmgreen	False	Gudmundsen-Holmgreen, Jørgen
124	Jørgen Gudmundsen-Holmgreen	False	Gudmundsen-Holmgreen, Jørgen
150	Richard Winther	True	Winther, Richard
151	Bjørn Nørgaard	True	Nørgaard, Bjørn
164	Richard Winther	False	Winther, Richard
190	Sonja Ferlov Mancoba	False	Ferlov Mancoba, Sonja

	birthyear	nationality	py_start	py_end	public_domain	language
22	1912.0	dansk	1927.0	1981.0	False	da-DK
70	1861.0	fransk	1895.0	1909.0	True	da-DK
102	1895.0	dansk	1899.0	1952.0	False	da-DK
107	1895.0	dansk	1894.0	1965.0	False	da-DK
117	1920.0	dansk	1967.0	1985.0	False	da-DK
121	1895.0	dansk	1899.0	1946.0	False	da-DK
124	1895.0	dansk	1899.0	1952.0	False	da-DK
150	1926.0	dansk	1980.0	1989.0	False	da-DK
151	1947.0	dansk	1962.0	1996.0	False	da-DK
164	1926.0	dansk	1987.0	2087.0	False	da-DK
190	1911.0	dansk	1964.0	1974.0	False	da-DK

	title	Age_when_created
22	Abstrakt figur	15.0
70	Portræt af Madame Maillol. Maske	34.0
102	Udkast til Ikaros	4.0
107	Uden titel	-1.0
117	Tre trommer	47.0
121	Udkast til Araberpige, der bærer et barn	4.0
124	Udkast til Ikaros	4.0
150	Uden titel	54.0
151	Erik Fischer	15.0
164	Uden titel	61.0
190	Maske (Bordmaske eller Maskens fødsel)	53.0

	Age_when_created2	time_to_produce
22	69.0	54.0
70	48.0	14.0
102	57.0	53.0
107	70.0	71.0
117	65.0	18.0
121	51.0	47.0
124	57.0	53.0
150	63.0	9.0
151	49.0	34.0
164	161.0	100.0
190	63.0	10.0

Bibliography

ACM (2018). Code of Ethics and Professional Conduct. Retrieved the 15th of December 2019 on:

https://www.acm.org/code-of-ethics

Codebeautify (n.d). JSON Viewer. Retrieved the 29th of November 2019 from: https://codebeautify.org/jsonviewer.

Datatilsynet (n.d). Hvad er personoplysninger. Retrieved the 15th of December 2019 on:

https://www.datatilsynet.dk/generelt-om-databeskyttelse/hvad-er-personoplysninger/

DCIM Usage Board (2012). Dublin Core Metadata Element Set, Version 1.1: Reference Description. Retrieved the 29th of November 2019 from: https://business.twitter.com/en/help/campaign-setup/create-a-tweet-engagement-campaign.html

GDPR-info (2018). General Data Protection Regulation. Retrieved the 15th of December 2019 on:

SMK (n.d.) SMK Open. Retrieved the 6th of November 2019 from: https://www.smk.dk/article/smk-open/

SMK (2019). Kunstig intelligens organiserer Danmarks største kunstsamling. Retrieved the 6th of November 2019 from: https://www.smk.dk/article/kunstig-intelligens-organiserer-danmarks-stoerste-kunstsamling/

Wilkinson, M. D. et al. (2016) The FAIR Guiding Principles for scientific data management and stewardship. Sci. Data 3:160018

Appendix:

Appendix A: Dataframe columns and type

acquisition_date	object
acquisition_date_precision	object
alternative_images_0_height	float64
alternative_images_0_iiif_id	object
alternative_images_0_iiif_info	object
alternative_images_0_mime_type	object
alternative_images_0_native	object
alternative_images_0_orientation	object
alternative_images_0_size	float64
alternative_images_0_thumbnail	object
alternative_images_0_width	float64
alternative_images_1_height	float64
alternative_images_1_iiif_id	object
alternative_images_1_iiif_info	object
alternative_images_1_mime_type	object
alternative_images_1_native	object
alternative_images_1_orientation	object
alternative_images_1_size	float64
alternative_images_1_thumbnail	object
alternative_images_1_width	float64
alternative_images_2_height	float64
alternative_images_2_iiif_id	object
alternative_images_2_iiif_info	object
alternative_images_2_mime_type	object
alternative_images_2_native	object
alternative_images_2_orientation	object
alternative_images_2_size	float64
alternative_images_2_thumbnail	object
alternative_images_2_width	float64
alternative_images_3_height	float64
alternative_images_3_iiif_id	object
alternative_images_3_iiif_info	object
alternative_images_3_mime_type	object
alternative_images_3_native	object
alternative_images_3_orientation	object
alternative_images_3_size	float64
alternative_images_3_thumbnail	object
alternative_images_3_width	float64
alternative_images_4_height	float64
alternative_images_4_iiif_id	object
alternative_images_4_iiif_info	object
alternative_images_4_mime_type	object
alternative_images_4_native	object
alternative_images_4_orientation	object
alternative_images_4_size	float64
alternative_images_4_thumbnail	object
alternative_images_4_width	float64
alternative_images_5_height	float64
alternative_images_5_iiif_id	object
alternative_images_5_iiif_info	object
alternative_images_5_mime_type	object
alternative_images_5_native	object
alternative_images_5_orientation	object
alternative_images_5_size	float64
alternative_images_5_thumbnail	object
alternative_images_5_width	float64
alternative_images_6_height	float64
alternative_images_6_iiif_id	object
alternative_images_6_iiif_info	object
alternative_images_6_mime_type	object
alternative_images_6_native	object
alternative_images_6_orientation	object
alternative_images_6_size	float64
alternative_images_6_thumbnail	object
alternative_images_6_width	float64
alternative_images_7_height	float64
alternative_images_7_iiif_id	object
alternative_images_7_iiif_info	object
alternative_images_7_mime_type	object
alternative_images_7_native	object
alternative_images_7_orientation	object
alternative_images_7_size	float64
alternative_images_7_thumbnail	object
alternative_images_7_width	float64
alternative_images_8_height	float64
alternative_images_8_iiif_id	object
alternative_images_8_iiif_info	object
alternative_images_8_mime_type	object
alternative_images_8_native	object
alternative_images_8_orientation	object
alternative_images_8_size	float64
alternative_images_8_thumbnail	object
alternative_images_8_width	float64
alternative_images_9_height	float64
alternative_images_9_iiif_id	object
alternative_images_9_iiif_info	object
alternative_images_9_mime_type	object
alternative_images_9_native	object
alternative_images_9_orientation	object
alternative_images_9_size	float64
alternative_images_9_thumbnail	object
alternative_images_9_width	float64
colors_0	object
colors_1	object
colors_2	object
colors_3	object
colors_4	object
content_description_0	object
created	object
credit_line_0	object
current_location_name	object
dimensions_0_notes	object
dimensions_0_part	object
dimensions_0_type	object
dimensions_0_unit	object
dimensions_0_value	object
dimensions_1_notes	object
dimensions_1_part	object
dimensions_1_type	object
dimensions_1_unit	object
dimensions_1_value	object
dimensions_2_notes	object
dimensions_2_part	object
dimensions_2_type	object
dimensions_2_unit	object
dimensions_2_value	object
dimensions_3_notes	object
dimensions_3_part	object
dimensions_3_type	object
dimensions_3_unit	object

```
dimensions_3_value                  object
dimensions_4_notes                  object
dimensions_4_part                   object
dimensions_4_type                   object
dimensions_4_unit                   object
dimensions_4_value                  object
dimensions_5_notes                  object
dimensions_5_part                   object
dimensions_5_type                   object
dimensions_5_unit                   object
dimensions_5_value                  object
distinguishing_features_0           object
distinguishing_features_1           object
exhibitions_0_date_end              object
exhibitions_0_date_start            object
exhibitions_0_exhibition            object
frame_notes_0                       object
frame_notes_1                       object
has_image                             bool
id                                  object
iiif_manifest                       object
image_cropped                       object
image_height                       float64
image_iiif_id                       object
image_iiif_info                     object
image_mime_type                     object
image_native                        object
image_orientation                   object
image_size                         float64
image_thumbnail                     object
```

```
image_width                        float64
labels_0_date                       object
labels_0_source                     object
labels_0_text                       object
labels_0_type                       object
labels_1_date                       object
labels_1_source                     object
labels_1_text                       object
labels_1_type                       object
materials_0_material                object
materials_1_material                object
materials_2_material                object
materials_3_material                object
materials_4_material                object
materials_5_material                object
materials_6_material                object
materials_7_material                object
modified                            object
notes_0                             object
notes_1                             object
notes_2                             object
notes_3                             object
notes_4                             object
number_of_parts                    float64
object_history_note_0               object
object_names_0_name                 object
object_number                       object
on_display                            bool
part_of_0                           object
part_of_1                           object
```

```
part_of_2                           object
part_of_3                           object
part_of_4                           object
part_of_5                           object
part_of_6                           object
parts_0                             object
parts_1                             object
parts_10                            object
parts_11                            object
parts_12                            object
parts_13                            object
parts_14                            object
parts_15                            object
parts_16                            object
parts_17                            object
parts_18                            object
parts_19                            object
parts_2                             object
parts_20                            object
parts_21                            object
parts_22                            object
parts_3                             object
parts_4                             object
parts_5                             object
parts_6                             object
parts_7                             object
parts_8                             object
parts_9                             object
production_0_creator                object
production_0_creator_date_of_birth  object
```

```
production_0_creator_date_of_death  object
production_0_creator_history        object
production_0_creator_lref           object
production_0_creator_nationality    object
production_0_place                  object
production_1_creator                object
production_1_creator_date_of_birth  object
production_1_creator_date_of_death  object
production_1_creator_history        object
production_1_creator_lref           object
production_1_creator_nationality    object
production_date_0_end               object
production_date_0_period            object
production_date_0_start             object
production_date_1_end               object
production_date_1_period            object
production_date_1_start             object
production_dates_notes_0            object
production_dates_notes_1            object
public_domain                         bool
related_objects_0_notes             object
responsible_department              object
rights                              object
techniques_0_technique              object
titles_0_language                   object
titles_0_notes                      object
titles_0_title                      object
titles_0_translation                object
titles_0_type                       object
titles_1_language                   object
```

```
titles_1_notes                      object
titles_1_title                      object
titles_1_translation                object
titles_1_type                       object
titles_2_language                   object
titles_2_notes                      object
titles_2_title                      object
titles_2_translation                object
titles_2_type                       object
dtype: object
```