Metadata

```
Course:   DS 5001
Module:   07 Homework
Topic:    PCA
Author:   R.C. Alvarado
```

Instructions

In this week's code exercise, you will apply PCA to a new corpus, a collection of gothic `g` and detective `d` novels from English literature. Using the tables `novels-LIB.csv` and `novels-CORPUS.csv` found in `/data/novels`and the notebook from this week (`M07_01_PCA.ipynb`) as well as previous relevant notebooks, perform the following tasks:

1. Import the `LIB` and `CORPUS` tables from the provided CSV files.

- The index `OHCO` for `CORPUS` is `['book_id','chapter_id','para_num','sent_num','token_num']`
- The index for `LIB` is `'book_id'`.

1. Extract a `VOCAB` table and add `max_pos` as a feature.

2. Compute `TFIDF` and `VOCAB['dfidf']` for the `CORPUS` using the following parameters:

- `bag = ['book_id', 'chap_id']`
- `tf_method = 'max'`
- `idf_method = 'standard'`

1. Create a `DOC` table from the `TFIDF` index in which each row represents a bag, i.e. a chapter. In other words, it should have `['book_id', 'chap_id']` as its index. This table should have information from the `LIB` table added to it, so that each chapter is identified with an author, title, and genre. These data will appear in your visualizations. For example, in a scatter plot of documents in the first two principle components, you will want to know the book and chapter of each data point.

2. Create a reduced version of the `TFIDF` table with only the top 1000 nouns (i.e. `NN` and `NNS`) in descending order of `DFIDF`.

- Do not "collapse" table -- keep the index as `(book_id, chap_id)`.

1. Write a function that computes PCA from a given document-term count matrix (this included weighted counts, such a tfidf). It should return three dataframes: `LOADINGS` (the term-component matrix), `DCM` (the document-component matrix), `COMPINF` (the component information table). Give it the following parameters:

- `X # The input matrix`
- `k # The number of components to generate`

- `norm_docs # True or False`
- `center_by_mean # True of False`
- `center_by_variance # True or False`

1. Compute PCA from the feature-reduced `TFIDF` table using your function. Use the following parameter values:

- `X = TFIDF_REDUCED` Or whatever you called your reduced TFIDF table
- `k=10`
- `norm_docs = True`
- `center_by_mean = False`
- `center_by_variance = False`

1. Visualize your results using scatter plots and box plots.

When you have finished these tasks, answer the questions listed below.

Questions

Q1

Looking at the documents plotted against the first principle component (PC), which genre has the more narrow range, i.e. distance between the minimum and maximum values? This can be seen using a box plot.

Answer: Detective `d`

Q2

Looking at the documents plotted against the first PC, which author has the highest absolute value, in terms of both mean and range? In other words, which author is farthest from 0? Again, the box plots of each author are useful here.

Answer: Radcliffe.

Q3

In the third PC, which author has, by far, the maximum range?

Answer: Radcliffe.

Q4

Looking at the loadings for the second PC, how would you characterize the opposition, based on the top three words at each pole?

Answer: family or indoors vs world or outdoors

There are many possible answers here. To be correct, but show evidence of interpreting the words found at each pole.

Q5

Recompute the principle components with `center_by_variance` set to `True`. This will change the words that appear at the extremes of the first PC. Does this change your interpretation in the previous question?

Answer: Yes. The words are sufficiently different in meaning to warrant a change in interpretation.

Solution

Set Up

Config

```
data_home = "../../repo/lessons/data"
local_lib = "../../repo/lessons/data"
data_prefix = 'novels'
OHCO = ['book_id', 'chap_id', 'para_num', 'sent_num', 'token_num']
```

```
bag = OHCO[:2] # chapter
max_terms = 1000
tf_method = 'max'
global_term_sig = 'dfidf'
n_comps = 10
center_by_mean=False
center_by_variance=False
```

Imports

```
import pandas as pd
import numpy as np
from scipy.linalg import norm, eigh
from sklearn.decomposition import PCA
import plotly_express as px
```

Prepare the Data

Get LIB

```
LIB = pd.read_csv(f'{data_home}/{data_prefix}/{data_prefix}-LIB.csv').set_index('book_id')
```

```
LIB.head()
```

```
                genre_id author_id
book_id
secretadversary        d  christie
styles                 d  christie
moonstone              d   collins
```

```
adventures             d     doyle
baskervilles           d     doyle
```

Get CORPUS

```
CORPUS = pd.read_csv(f'{data_home}/{data_prefix}/{data_prefix}-CORPUS.csv').set_index(OHCO)
```

```
CORPUS.head()
```

```
                                                     pos     term_str
book_id          chap_id para_num sent_num token_num
secretadversary 1        0        1        0          DT          the
                                           1         NNP        young
                                           2         NNP  adventurers
                                           3         NNP          ltd
                         1        0        0          JJ        tommy
```

Extract VOCAB

```
VOCAB = CORPUS.term_str.value_counts().to_frame('n').sort_index()
VOCAB.index.name = 'term_str'
```

```
VOCAB['max_pos'] = CORPUS.value_counts(['term_str','pos']).unstack().idxmax(1)
```

```
NOUNS = VOCAB[VOCAB.max_pos.isin(['NN','NNS'])]
```

Create BOW

```
BOW = CORPUS.groupby(bag+['term_str']).term_str.count().to_frame('n')
```

```
BOW
```

```
                                n
book_id    chap_id term_str
adventures 1       a          225
                   abandoned    1
                   abhorrent    1
                   able         3
                   about        9
...                           ...
usher      1       yet         22
                   you          9
                   your         2
                   youth        1
                   zigzag       2

[377222 rows x 1 columns]
```

Create DTM

```
DTM = BOW.n.unstack(fill_value=0)
```

Create TFIDF

```
VOCAB['df'] = DTM.astype('bool').sum()
VOCAB['idf'] = np.log2(len(DTM) / VOCAB.df)
```

```
TFIDF = (DTM.T / DTM.T.max()).T * VOCAB.idf
```

```
# TFIDF
```

Create DFIDF

```
VOCAB['dfidf'] = VOCAB.df * VOCAB.idf
```

```
VOCAB.dfidf
```

```
term_str
a              0.000000
aback         46.368028
abaft          8.321928
abandon       98.408049
abandoned    124.513524
                 ...
à             20.210897
æt             8.321928
ætat           8.321928
ça            14.643856
émeutes        8.321928
Name: dfidf, Length: 27396, dtype: float64
```

```
VIDX = VOCAB[VOCAB.max_pos.isin(['NN','NNS'])].sort_values('dfidf', ascending=False).head(ma
```

Compute PCA

```
# norm?
```

Define Function

```
def get_pca(TFIDF,
            k=10,
            norm_docs=True,
            norm_level=2,
            center_by_mean=True,
            center_by_variance=False):

    if TFIDF.isna().sum().sum():
        TFIDF = TFIDF.fillna(0)
```

```
    if norm_docs:
        TFIDF = TFIDF.apply(lambda x: x / norm(x), 1).fillna(0)

    if center_by_mean:
        TFIDF = TFIDF - TFIDF.mean()

    if center_by_variance:
        TFIDF = TFIDF / TFIDF.std()

    COV = TFIDF.cov()

    eig_vals, eig_vecs = eigh(COV)
    EIG_VEC = pd.DataFrame(eig_vecs, index=COV.index, columns=COV.index)
    EIG_VAL = pd.DataFrame(eig_vals, index=COV.index, columns=['eig_val'])
    EIG_VAL.index.name = 'term_str'

    EIG_IDX = EIG_VAL.eig_val.sort_values(ascending=False).head(k)

    COMPS = EIG_VEC[EIG_IDX.index].T
    COMPS.index = [i for i in range(COMPS.shape[0])]
    COMPS.index.name = 'pc_id'

    LOADINGS = COMPS.T

    DCM = TFIDF.dot(LOADINGS)

    COMPINF = pd.DataFrame(index=COMPS.index)

    for i in range(k):
        for j in [0, 1]:
            top_terms = ' '.join(LOADINGS.sort_values(i, ascending=bool(j)).head(5).index.to
            COMPINF.loc[i, j] = top_terms
    COMPINF = COMPINF.rename(columns={0:'pos', 1:'neg'})

    COMPINF['eig_val'] = EIG_IDX.reset_index(drop=True).to_frame()
    COMPINF['exp_var'] = COMPINF.eig_val / COMPINF.eig_val.sum()

    return LOADINGS, DCM, COMPINF
```

```
LOADINGS, DCM, COMPINF = get_pca(TFIDF[VIDX], norm_docs=True, norm_level=2, center_by_mean=c
```

```
LOADINGS
```

```
pc_id                  0         1         2         3         4         5  \
term_str
yours           0.019484  0.016559 -0.007586 -0.003938 -0.017489  0.025233
```

```
reply         -0.009054  0.007702 -0.010562 -0.019532 -0.012423  0.016606
order         -0.001659  0.001893 -0.033437  0.018477 -0.008462  0.032114
curiosity     -0.018668  0.012928 -0.005848 -0.010643 -0.002937  0.027110
memory        -0.001513 -0.001008  0.018553 -0.013331 -0.008593  0.014673
...                 ...       ...       ...       ...       ...       ...
lad            0.024363 -0.008369 -0.005117 -0.002852 -0.010479  0.006415
enquiry       -0.031909 -0.017446 -0.019341 -0.019341  0.011217  0.005914
bag            0.032418 -0.031064 -0.030680  0.022937  0.006646  0.019362
investigation  0.023187  0.013508 -0.017991 -0.006847  0.057409 -0.003030
inclination   -0.016683  0.027050  0.002015  0.006067 -0.007924 -0.004001

pc_id                 6         7         8         9
term_str
yours          0.003145 -0.008342  0.007351  0.005823
reply         -0.011956  0.013342  0.026030  0.002001
order          0.010416 -0.015883 -0.018002 -0.000714
curiosity      0.005271 -0.002335  0.007111 -0.030269
memory        -0.033355  0.016084 -0.016580  0.017187
...                 ...       ...       ...       ...
lad            0.030052 -0.001212 -0.036436  0.032535
enquiry        0.022170  0.005435 -0.008798  0.009281
bag           -0.089170 -0.056387  0.036524  0.037134
investigation -0.016042  0.013304 -0.047208 -0.016968
inclination    0.028011 -0.033625  0.019295 -0.014957

[1000 rows x 10 columns]
```

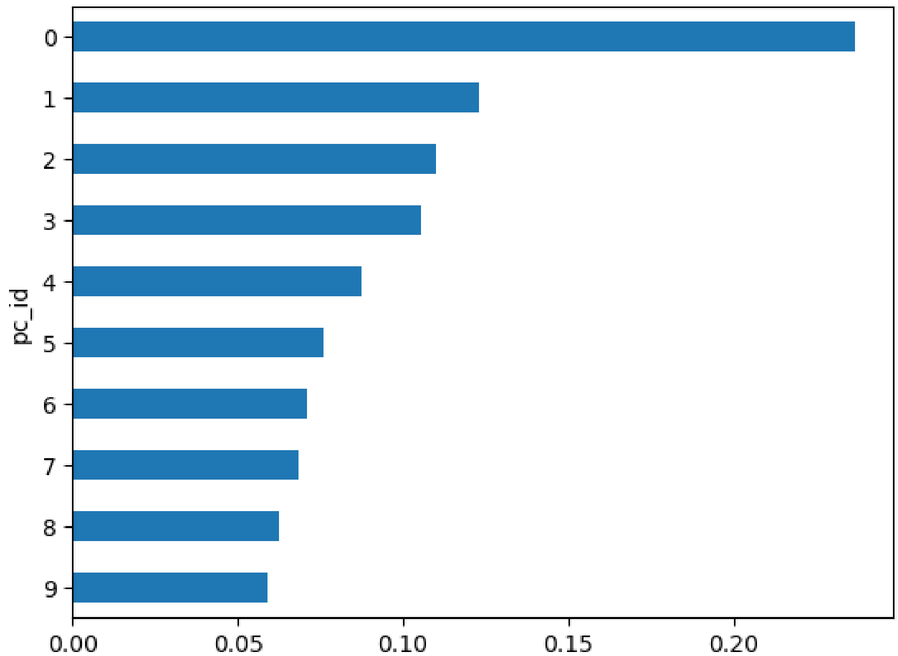
COMPINF

```
                                           pos  \
pc_id
0                    thats youre shes cab lawyer
1            brother engagement father sister son
2             chateau cottage woods sea mountains
3                blood whilst child sleep monster
4                   chateau cab dog inquiry letter
5                chateau convent bed coffee crime
6                     cab brother castle son horse
7               aunt evidence murderer cab murder
8       aunt brother acquaintance engagement box
9                  castle box chateau river whilst

                                         neg   eig_val   exp_var
pc_id
0         chateau castle chamber woods convent  0.042644  0.236543
1             mountains woods sea rocks whilst  0.022226  0.123285
2            castle aunt chamber apartment lamp  0.019867  0.110199
```

```
3                chateau thats aunt youre shes  0.019055  0.105694
4                 thats youre shes guess youve  0.015809  0.087692
5          mountains castle rocks aunt horses  0.013682  0.075895
6            aunt whilst sleep aunts mistress  0.012767  0.070817
7               chateau sleep whilst lady bed  0.012347  0.068490
8     cottage mistress servants lady convent  0.011256  0.062436
9                candle lamp walls bosom bed  0.010628  0.058949
```

```
COMPINF.exp_var.sort_values().plot.barh();
```



Create DOC

```
DOC = pd.DataFrame(index=TFIDF.index).join(LIB)
```

```
DOC
```

```
                    genre_id  author_id
book_id    chap_id
adventures 1               d      doyle
           2               d      doyle
           3               d      doyle
           4               d      doyle
           5               d      doyle
...                      ...        ...
udolpho    54              g  radcliffe
```

```
           55              g  radcliffe
           56              g  radcliffe
           57              g  radcliffe
usher      1               g        poe

[320 rows x 2 columns]
```

```
DOC['label'] = DOC.apply(lambda x: f"{x.author_id.title()} {x.name[0].title()} Ch{x.name[1]
```

```
DOC
```

```
                   genre_id  author_id                   label
book_id    chap_id
adventures 1              d      doyle    Doyle Adventures Ch1-d
           2              d      doyle    Doyle Adventures Ch2-d
           3              d      doyle    Doyle Adventures Ch3-d
           4              d      doyle    Doyle Adventures Ch4-d
           5              d      doyle    Doyle Adventures Ch5-d
...                     ...        ...                       ...
udolpho    54             g  radcliffe  Radcliffe Udolpho Ch54-g
           55             g  radcliffe  Radcliffe Udolpho Ch55-g
           56             g  radcliffe  Radcliffe Udolpho Ch56-g
           57             g  radcliffe  Radcliffe Udolpho Ch57-g
usher      1              g        poe           Poe Usher Ch1-g

[320 rows x 3 columns]
```

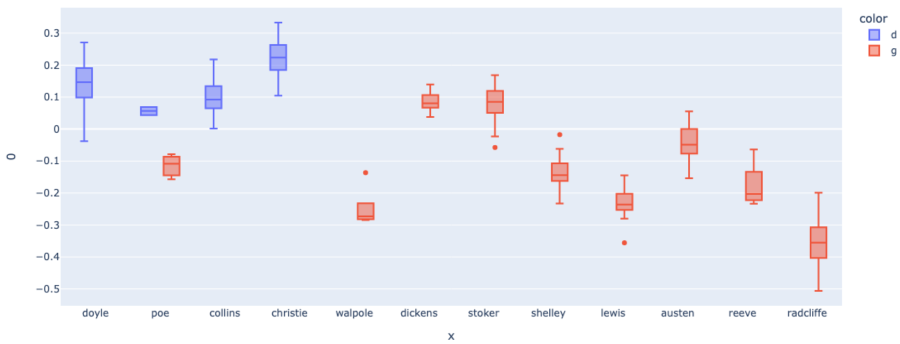
Visualize

Docs PC0 Genre

```
px.box(DCM, x=DOC.genre_id, y=0, height=500, color=DOC.genre_id)
```

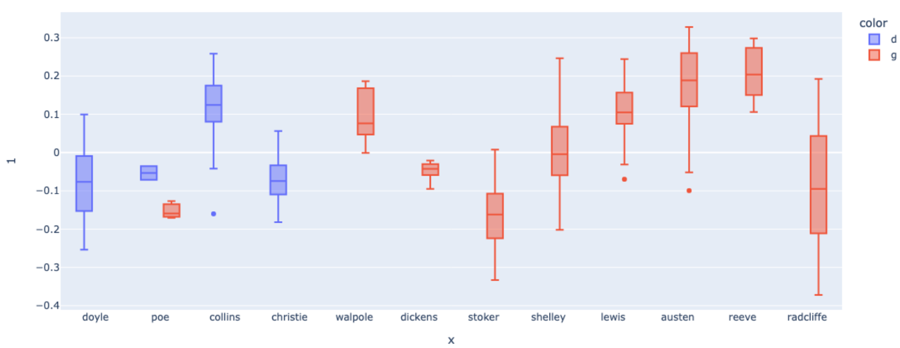
Docs PC0 Author

```
px.box(DCM, x=DOC.author_id, y=0, height=500, color=DOC.genre_id)
```



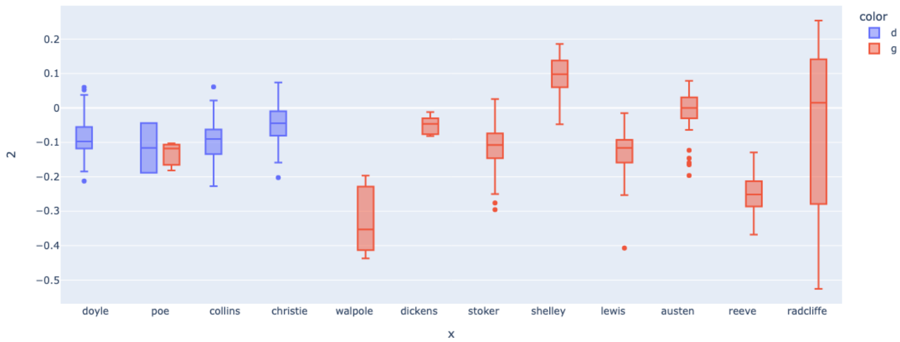
Docs PC1 Author

```
# DCM.join(LIB.author_id).groupby('author_id').mean().idxmax()
# DCM.join(LIB.author_id).groupby('author_id').mean().idxmin()
px.box(DCM, x=DOC.author_id, y=1, height=500, color=DOC.genre_id)
```



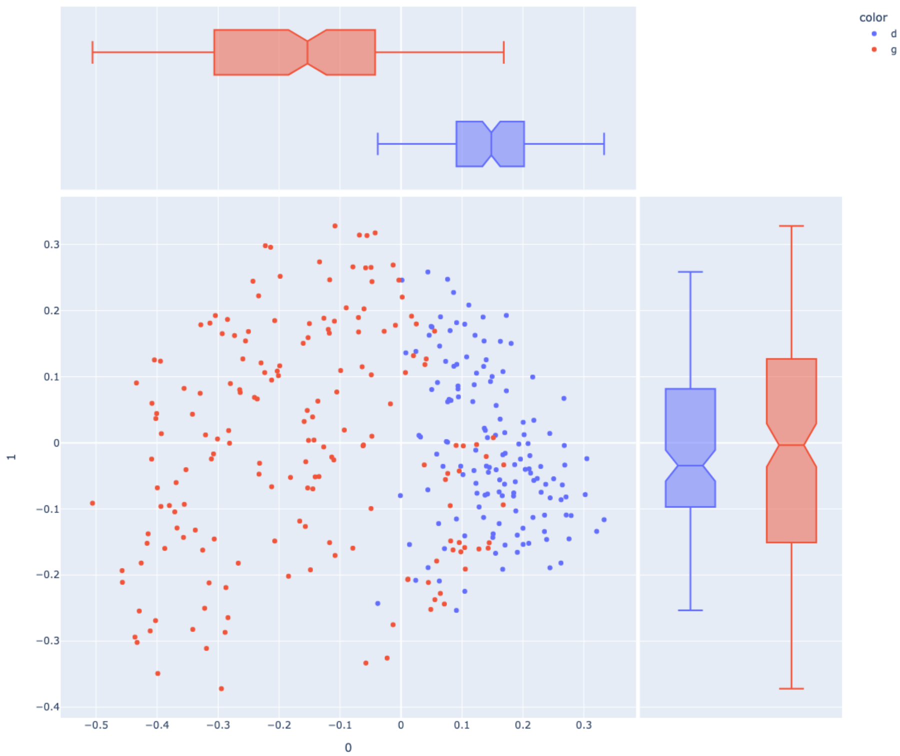
Docs PC2 Author

```
px.box(DCM, x=DOC.author_id, y=2, height=500, color=DOC.genre_id)
```



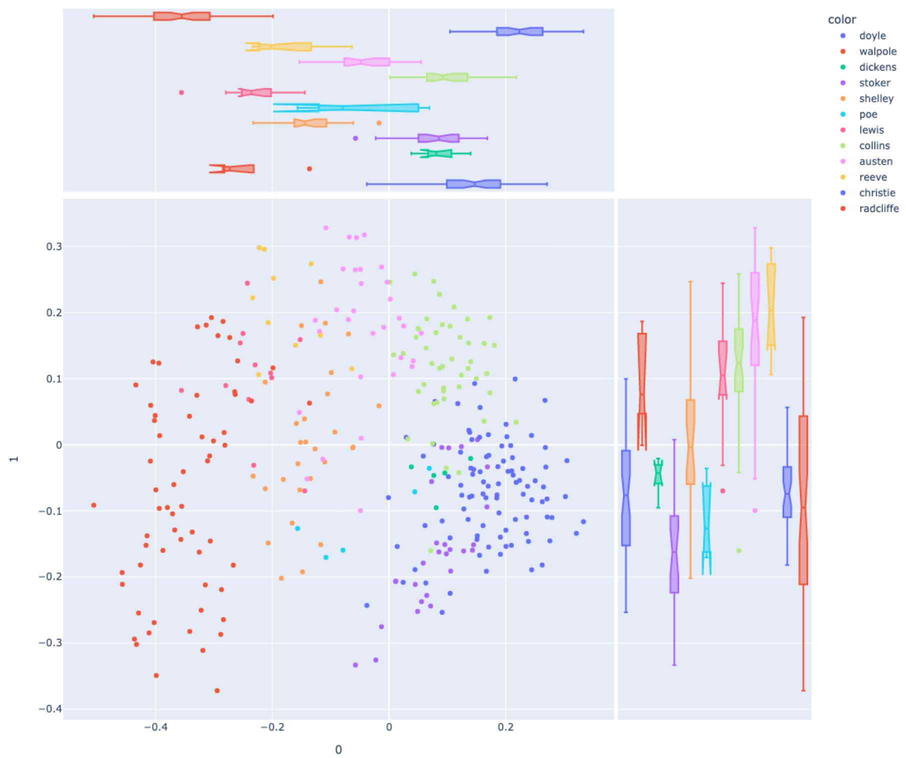
Docs PC 0 and 1 Genre

```
px.scatter(DCM, 0, 1,
           color=DOC.genre_id,
           hover_name=DOC.label,
           marginal_x='box', marginal_y='box', height=1000)
```



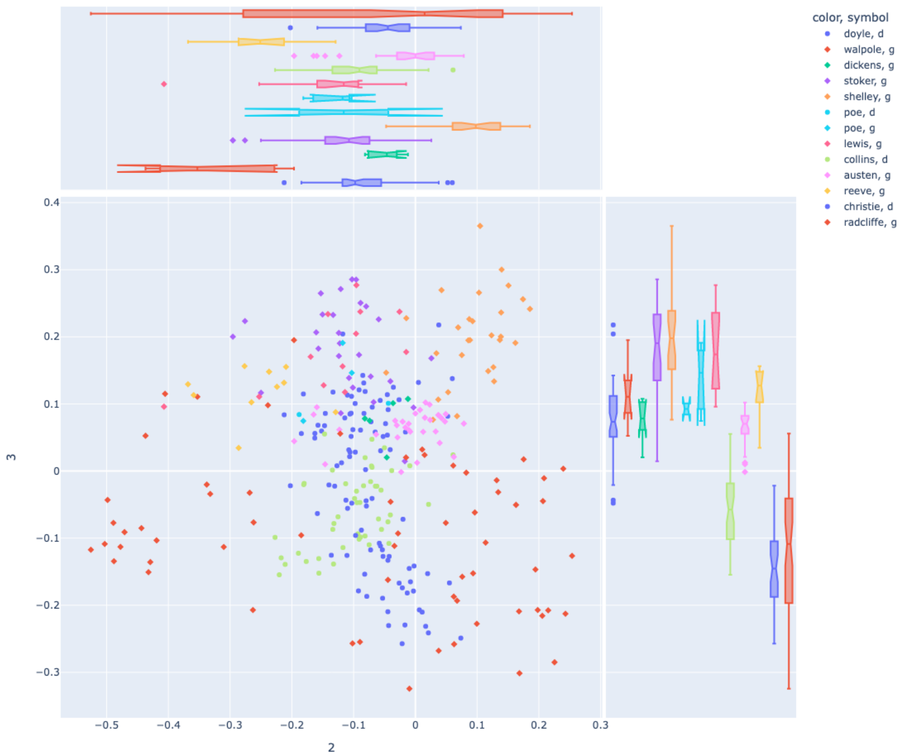
Docs PC 0 and 1 Author

```
px.scatter(DCM, 0, 1,
           color=DOC.author_id,
           hover_name=DOC.label,
           marginal_x='box', marginal_y='box', height=1000)
```



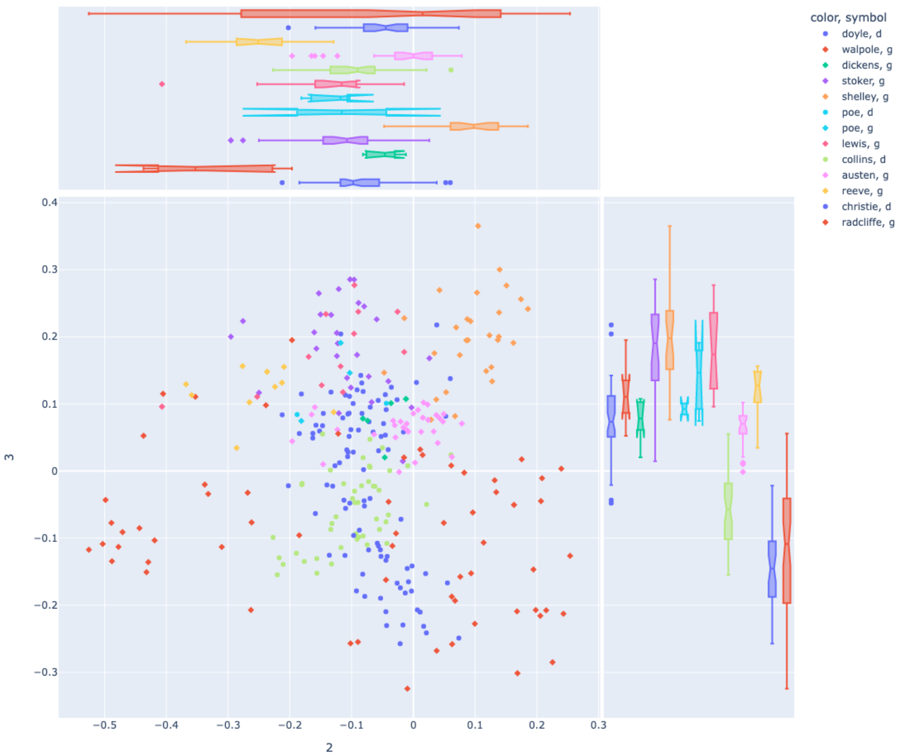
Docs PC 2 and 3 Author

```
px.scatter(DCM, 2, 3,
           color=DOC.author_id,
           symbol=DOC.genre_id,
           hover_name=DOC.label,
           marginal_x='box', marginal_y='box', height=1000)
```



Docs PC 2 and 3 Author

```
px.scatter(DCM, 2, 3,
           color=DOC.author_id,
           symbol=DOC.genre_id,
           hover_name=DOC.label,
           marginal_x='box', marginal_y='box', height=1000)
```




```
# px.scatter_3d(DCM, 0, 1, 2, color=DOC.author_id, height=1000, hover_name=DOC.label)
```

Loadings

```
LOADINGS
```

```
pc_id                 0         1         2         3         4         5  \
term_str
yours          0.019484  0.016559 -0.007586 -0.003938 -0.017489  0.025233
reply         -0.009054  0.007702 -0.010562 -0.019532 -0.012423  0.016606
order         -0.001659  0.001893 -0.033437  0.018477 -0.008462  0.032114
curiosity     -0.018668  0.012928 -0.005848 -0.010643 -0.002937  0.027110
memory        -0.001513 -0.001008  0.018553 -0.013331 -0.008593  0.014673
...                 ...       ...       ...       ...       ...       ...
lad            0.024363 -0.008369 -0.005117 -0.002852 -0.010479  0.006415
enquiry       -0.031909 -0.017446 -0.019341 -0.019341  0.011217  0.005914
bag            0.032418 -0.031064 -0.030680  0.022937  0.006646  0.019362
investigation  0.023187  0.013508 -0.017991 -0.006847  0.057409 -0.003030
inclination   -0.016683  0.027050  0.002015  0.006067 -0.007924 -0.004001
```

```
pc_id                 6         7         8         9
term_str
yours          0.003145 -0.008342  0.007351  0.005823
reply         -0.011956  0.013342  0.026030  0.002001
order          0.010416 -0.015883 -0.018002 -0.000714
curiosity      0.005271 -0.002335  0.007111 -0.030269
memory        -0.033355  0.016084 -0.016580  0.017187
...                 ...       ...       ...       ...
lad            0.030052 -0.001212 -0.036436  0.032535
enquiry        0.022170  0.005435 -0.008798  0.009281
bag           -0.089170 -0.056387  0.036524  0.037134
investigation -0.016042  0.013304 -0.047208 -0.016968
inclination    0.028011 -0.033625  0.019295 -0.014957

[1000 rows x 10 columns]
```

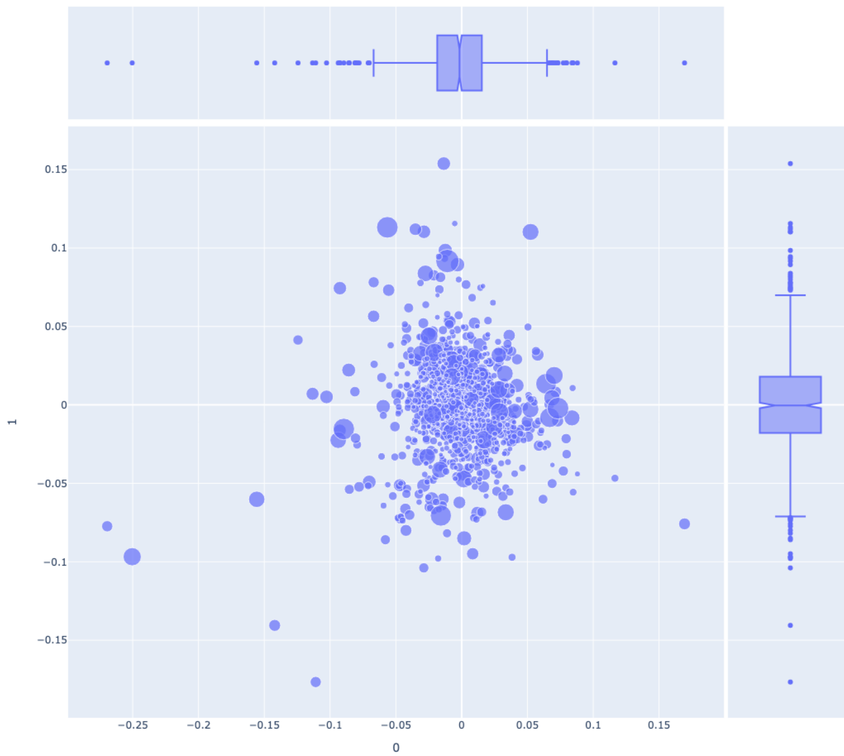
```
VOCAB.loc[VIDX]
```

```
                 n max_pos   df       idf       dfidf
term_str
yours          198      NN  118  1.439285  169.835635
reply          184      NN  118  1.439285  169.835635
order          227      NN  118  1.439285  169.835635
curiosity      208      NN  118  1.439285  169.835635
memory         208      NN  119  1.427110  169.826129
...            ...     ...  ...       ...         ...
lad             49      NN   35  3.192645  111.742578
enquiry         51      NN   35  3.192645  111.742578
bag             72      NN   35  3.192645  111.742578
investigation   53      NN   35  3.192645  111.742578
inclination     43      NN   35  3.192645  111.742578

[1000 rows x 5 columns]
```

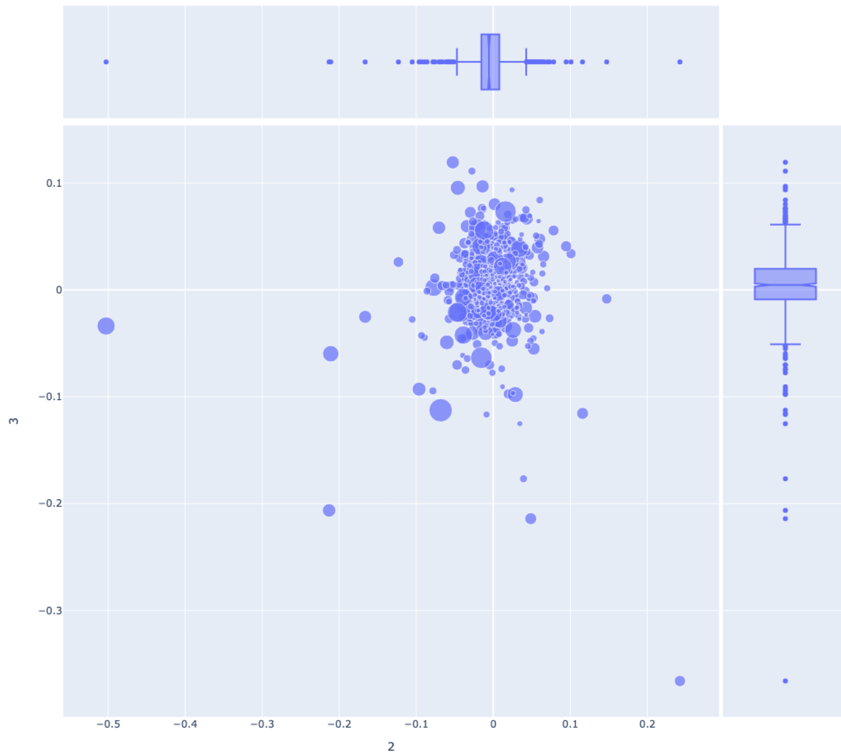
Loadings PC 0 and 1

```
px.scatter(LOADINGS.reset_index(), 0, 1, hover_name='term_str', size=VOCAB.loc[VIDX].n,
           marginal_x='box', marginal_y='box', height=1000)
```



Loadings PC 2 and 3

```
px.scatter(LOADINGS.reset_index(), 2, 3, hover_name='term_str', size=VOCAB.loc[VIDX].n,
           marginal_x='box', marginal_y='box', height=1000)
```


0.1
0
−0.1
−0.2
−0.3
3
−0.5
−0.4
−0.3
−0.2
−0.1
0
0.1
0.2
2