marp: true

Working with the Supreme Court Database

UIC CS 111 Law, October 2019

Profs. Bob Sloan and Richard Warner

Primary author orignal version: Dr. Daniel M. Katz

Downloading Data

We will be using data from the Supreme Court Database available at: http://scdb.wustl.edu/index.php

The first file we want (MODERN Database: 2019 Release 01, Justice Centered Data, Cases Organized by Supreme Court Citation, CSV file type) is at:

http://scdb.wustl.edu/_brickFiles/2019_01/SCDB_2019_01_justiceCentered_Citation.csv.zip

(Post to Piazza or visit any TA or lab tutor if you don't know how to unzip a zip file.)

Importing modules

Good practice to put *all* modules you're using at top of code.

We will be using (at least):

- the open-source Python Data Analysis Library pandas (http://pandas.pydata.org/)
- the standard Python plotting library, matplotlib.pyplot

So at top of your Python file (if working in a file) or at console before anything else, let's put

```
import pandas
import matplotlib.pyplot as plt
```

Loading data into Python from file

Open as usual, *except* file's characters in relatively rare ISO-8859-1 encoding, not Python's default of ASCII or UTF-8 Unicode:

```
fileref = open('SCDB_2019_01_justiceCentered_Citation.csv',
                encoding='ISO-8859-1')
```

Could use readlines (or other methods) with fileref as usual, but we'd like to

- exploit the CSV format, and
- *do data analytics with pandas*, so

```
# Read from file with pandas preparing to exploit csv format
scdb = pandas.read_csv(fileref)
```

Some CS: Data Representations

For encoding lovers: Recall **ISO-8859-1** 1-byte-only encoding for a 256 character subset of Unicode, sufficient for alphabets of Western European Languages, English, and some common *non*-ASCII symbols, including cent sign, Pound sign, Yen sign, and section symbol §. (Supreme Court DB and lots of legal writing uses §.)

CSV = Comman Separated Values: Text format for Excel style data. Adjacent cells separated by commas; rows separated by newlines.

- Universal format for sharing data files for data science.

- Python has standard built-in module that helps with CSV, and so does Pandas.

- Maybe more about details later in semester; don't need more for now.

First Look at the Data

Data dimensions:

```
>>> print(scdb.shape)

(78233, 61)
```

First 5 rows:

```
>>> scdb.head()
     caseId     docketId    caseIssuesId               voteId dateDecision  \
0  1946-001  1946-001-01  1946-001-01-01  1946-001-01-01-01-01   11/18/1946
1  1946-001  1946-001-01  1946-001-01-01  1946-001-01-01-01-02   11/18/1946
2  1946-001  1946-001-01  1946-001-01-01  1946-001-01-01-01-03   11/18/1946
3  1946-001  1946-001-01  1946-001-01-01  1946-001-01-01-01-04   11/18/1946
4  1946-001  1946-001-01  1946-001-01-01  1946-001-01-01-01-05   11/18/1946

<deleting some so it will fit on one slide>

        ...         majVotes  minVotes justice   justiceName vote opinion  \
0       ...                8         1      86      HHBurton  2.0     1.0
1       ...                8         1      84     RHJackson  1.0     1.0
2       ...                8         1      81     WODouglas  1.0     1.0
3       ...                8         1      80  FFrankfurter  4.0     2.0
4       ...                8         1      79        SFReed  1.0     1.0
<deleting more because it goes on>

[5 rows x 61 columns]
>>>
```

More selecting subsets of rows

- Last 5 rows:

```
scdb.tail()
```

- A specific row:

```
>>> scdb.loc[10]
caseId                                        1946-002
docketId                                   1946-002-01
caseIssuesId                            1946-002-01-01
...
chief                                           Vinson
docket                                              12
caseName                    CLEVELAND v. UNITED STATES
...
justiceName                                  RHJackson
vote                                                 2
...
```

Visualizing Data: matplotlib

"Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter notebook, web application servers, and four graphical user interface toolkits. ...

For simple plotting the pyplot module provides a MATLAB-like interface, particularly when combined with IPython." http://matplotlib.org/

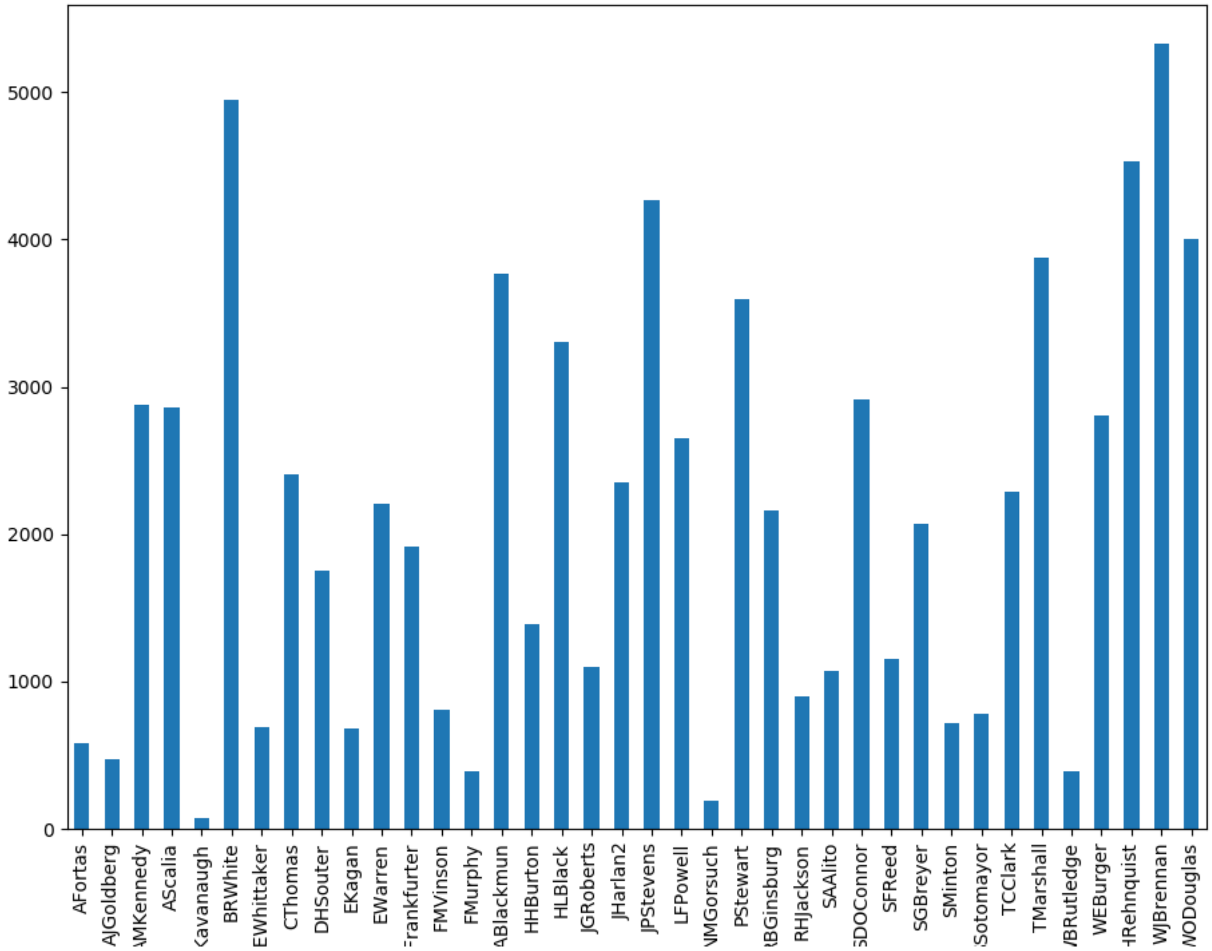
(The Spyder console is an IPython shell.)

Learn More: http://matplotlib.org/

How Many Decisions Did Each Justice Participate in?

Let's plot the number of decisions by Justice in alphabetical order

```
In [1]: f = plt.figure(figsize=(11,8))  #  11 x 8 inch fig.
In [2]: scdb.groupby("justiceName")["docketId"].count().plot(
                        kind="bar")
Out[2]:
<matplotlib.axes._subplots.AxesSubplot object at 0x11773dc18>
```

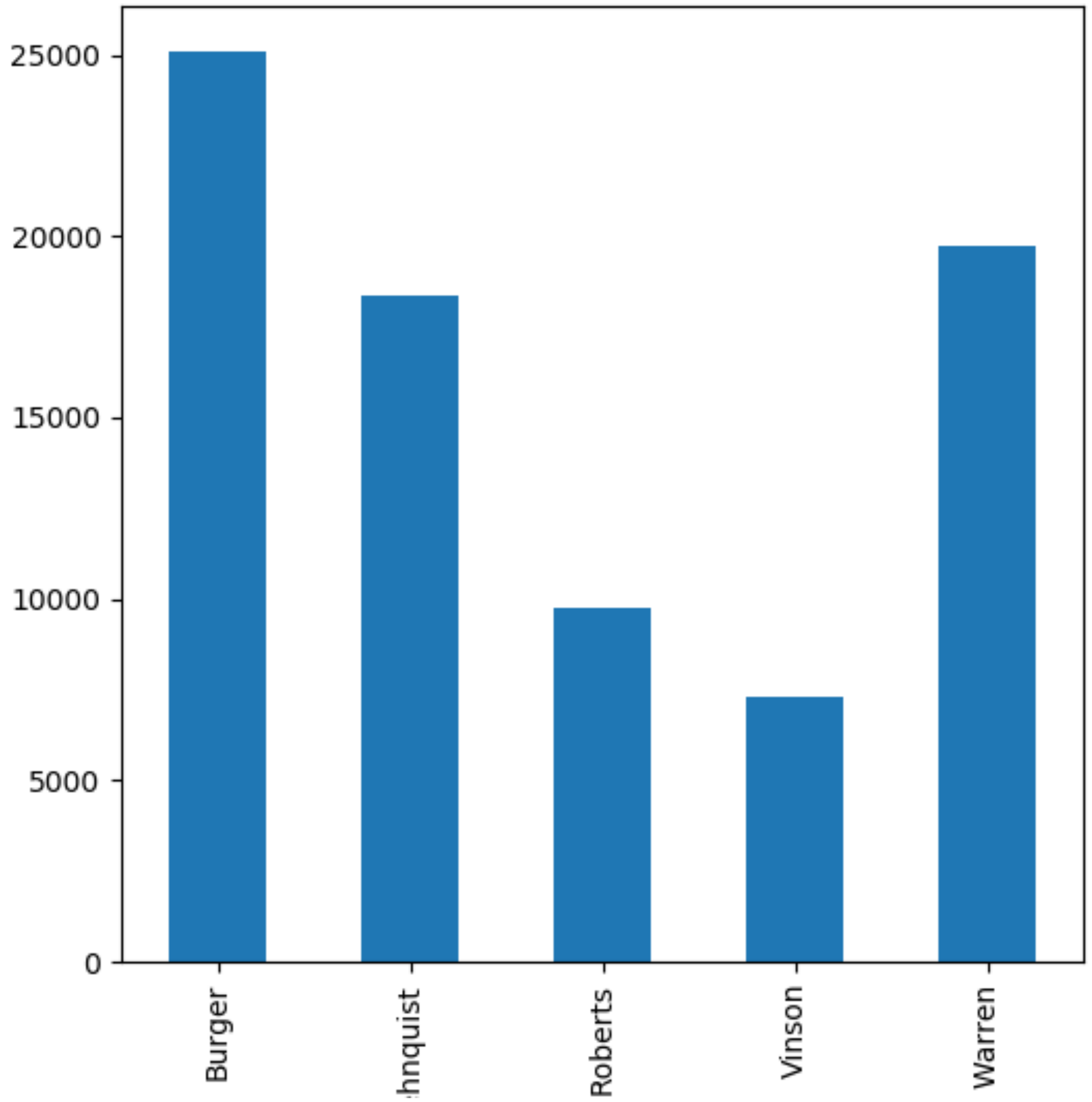


What About Chief Justices Only?

We can try to change the groupby from "justiceName"" to "chief"

```
f = plt.figure(figsize=(6,6))       # A little smaller
scdb.groupby('chief')['docketId'].count().plot(kind='bar')
```

Chiefs?



Data Science ABC

Always be checking (your outputs)

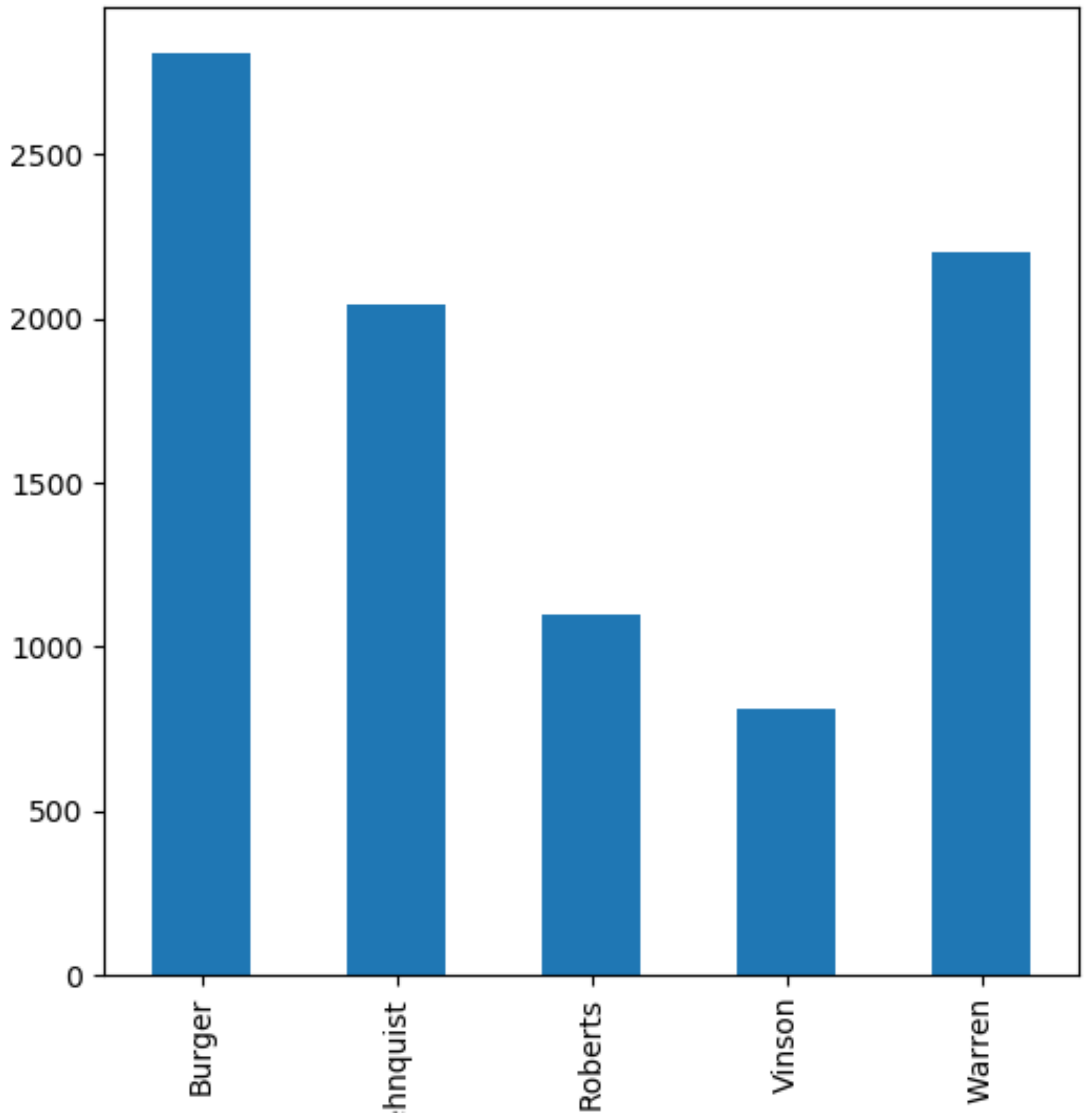
Look carefully at that graph and compare to earlier graph with all justices. It's not really what we want, is it?

The numbers are a bit off.

To get the number of decisions for each chief justice, we need to use .nunique rather than .count.

```
f = plt.figure(figsize=(6,6))
scdb.groupby('chief')['docketId'].nunique().plot(kind='bar')
```

Correct graph for Chief Justices



What are we counting?

Why do we use .count() for the number of decisions by each justice, but .nunique() for the number of decisions by each chief justice?

Let's take a closer look at the data to see what is going on

```
>>> scdb_peek = scdb.loc[0:20,
            {'docketId', 'chief', 'justiceName'}]
>>> scdb_peek
     justiceName   chief     docketId
0        HHBurton  Vinson  1946-001-01
1       RHJackson  Vinson  1946-001-01
2       WODouglas  Vinson  1946-001-01
3    FFrankfurter  Vinson  1946-001-01
4          SFReed  Vinson  1946-001-01
5         HLBlack  Vinson  1946-001-01
6      WBRutledge  Vinson  1946-001-01
7         FMurphy  Vinson  1946-001-01
8        FMVinson  Vinson  1946-001-01
9        HHBurton  Vinson  1946-002-01
10      RHJackson  Vinson  1946-002-01
11      WODouglas  Vinson  1946-002-01
12   FFrankfurter  Vinson  1946-002-01
13         SFReed  Vinson  1946-002-01
14        HLBlack  Vinson  1946-002-01
15     WBRutledge  Vinson  1946-002-01
16        FMurphy  Vinson  1946-002-01
17       FMVinson  Vinson  1946-002-01
18       HHBurton  Vinson  1946-003-01
19      RHJackson  Vinson  1946-003-01
20      WODouglas  Vinson  1946-003-01
```

What's going on with the SC DB

The first two dockets ("1946-001-01" and "1946-002-01") each have 9 rows (or records in data science speak).

Each row or record represents a vote by the identified justiceName and each justice votes only once per docketId.

The "chief" column represents the Chief Justice for the given docketId and there is only one chief per docketId.

Thus, when we counted the *number of rows* for each Chief Justice we were counting the number of votes cast for all the docketId's that each Chief Justice presided over, including their own vote, giving us the number of decisions rendered by each Chief Justice * approx. 9

count()

".count()" gives us the number of *rows* for each chief. This is the plot above that was incorrect:

```
>>> scdb.groupby('chief')['docketId'].count()
chief
Burger       25094
Rehnquist    18358
Roberts       9774
Vinson        7307
Warren       19736
Name: docketId, dtype: int64
```

nunique()

".nunique()" gives us the number of unique docketId's for each chief, rather than the number of rows.

```
>>> scdb.groupby('chief')['docketId'].nunique()
chief
Burger       2809
Rehnquist    2044
Roberts      1096
Vinson        812
Warren       2205
Name: docketId, dtype: int64
```

Direct comparison

Let's run the count and nunique figures by justiceName to compare.

Display only the last 5 rows which contain two justices that were chief justices (Burger and Rehnquist) so we can easily compare the results.

```
justice_count = scdb.groupby('justiceName')['docketId'].count()
>>> justice_count.tail()
justiceName
WBRutledge      387
WEBurger       2807
WHRehnquist    4529
WJBrennan      5325
WODouglas      4001
Name: docketId, dtype: int64
```

justiceName

Unlike chief, grouping by justiceName gives us the same result for .count and .nunique because each row represents a justice's vote and each justice only votes once per docketId.

Hopefully you now understand the scdb data structure well enough that you see why `scdb.groupby('chief')['docketId'].nunique().plot(kind='bar')` gave the proper plot above.

What if we were only interested in cases from certain terms?

```
#The 2010 Term
scdb_subset = scdb[scdb.term == 2010]



#Terms 2010 to current: Let's use this one
scdb_subset = scdb[scdb.term >= 2010]
```

Data Exploration - Descriptive Statistics

Since the 2010 term, how many cases (caseId) has the court reviewed?

```
# We use "nunique" rather than "count" because our data
# has 1 row for each voting Justice (usually 9 per case) but
#  we want to know the number of distinct caseId's, not rows.
>>> scdb_subset.caseId.nunique()
684

# See the difference with count?
>>> scdb_subset.caseId.count()
6068
```

Data Exploration - Descriptive Statistics (cont.)

How many cases for each term from 2010 on?

```
>>> scdb_subset.groupby('term').caseId.nunique()
term
2010    84
2011    77
2012    79
2013    75
2014    70
2015    81
2016    69
2017    76
2018    73
Name: caseId, dtype: int64
```

Data Exploration - Descriptive Statistics (cont.)

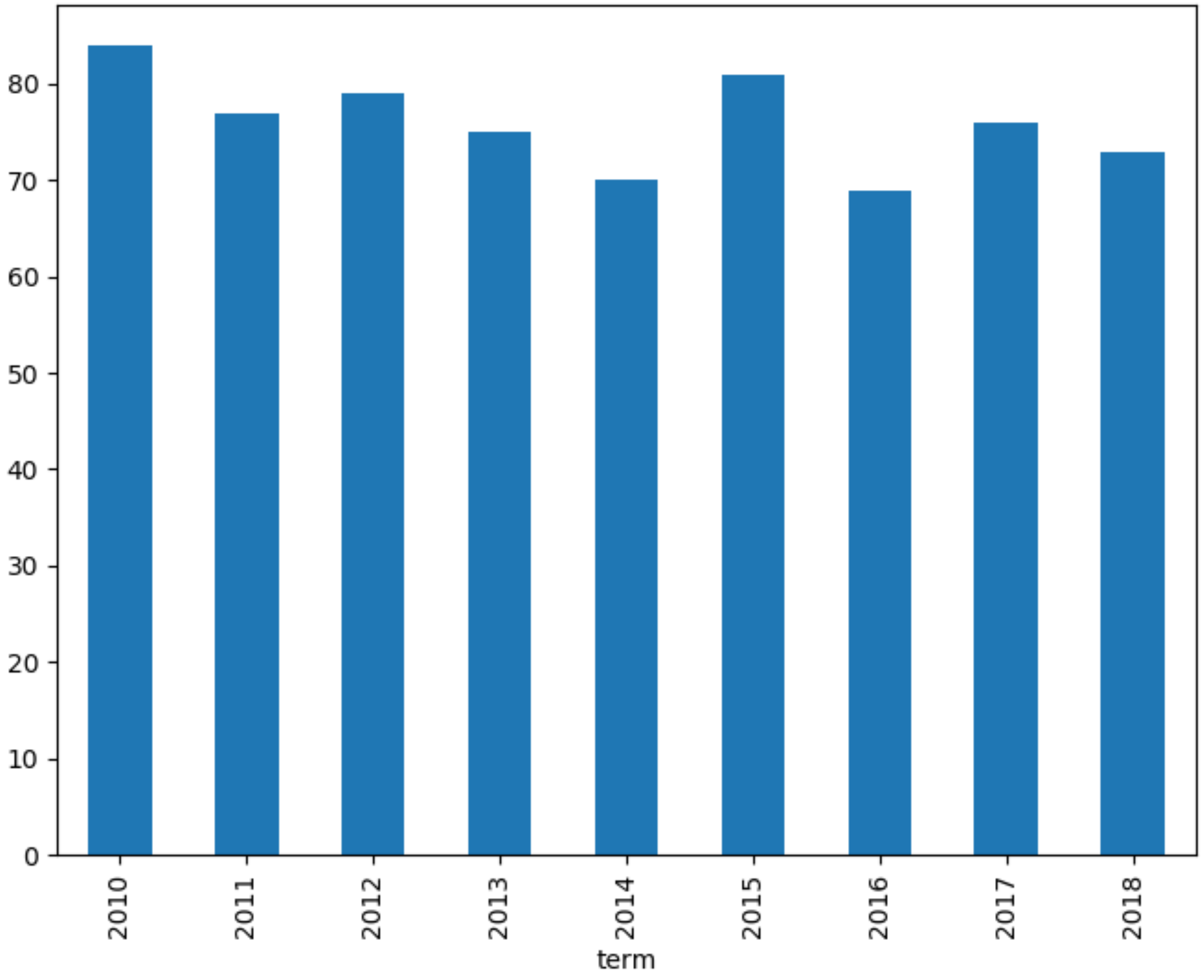
- What is the average number of cases per term?

```
>>> scdb_subset.groupby('term').caseId.nunique().mean()
76.0
```

Data Exploration - Bar Plots again

- Visualize the number of cases for each term

```
#Plot the number of cases for each term
f = plt.figure(figsize=(8,6))
scdb_subset.groupby('term')['caseId'].nunique().plot(
                        kind="bar")
```

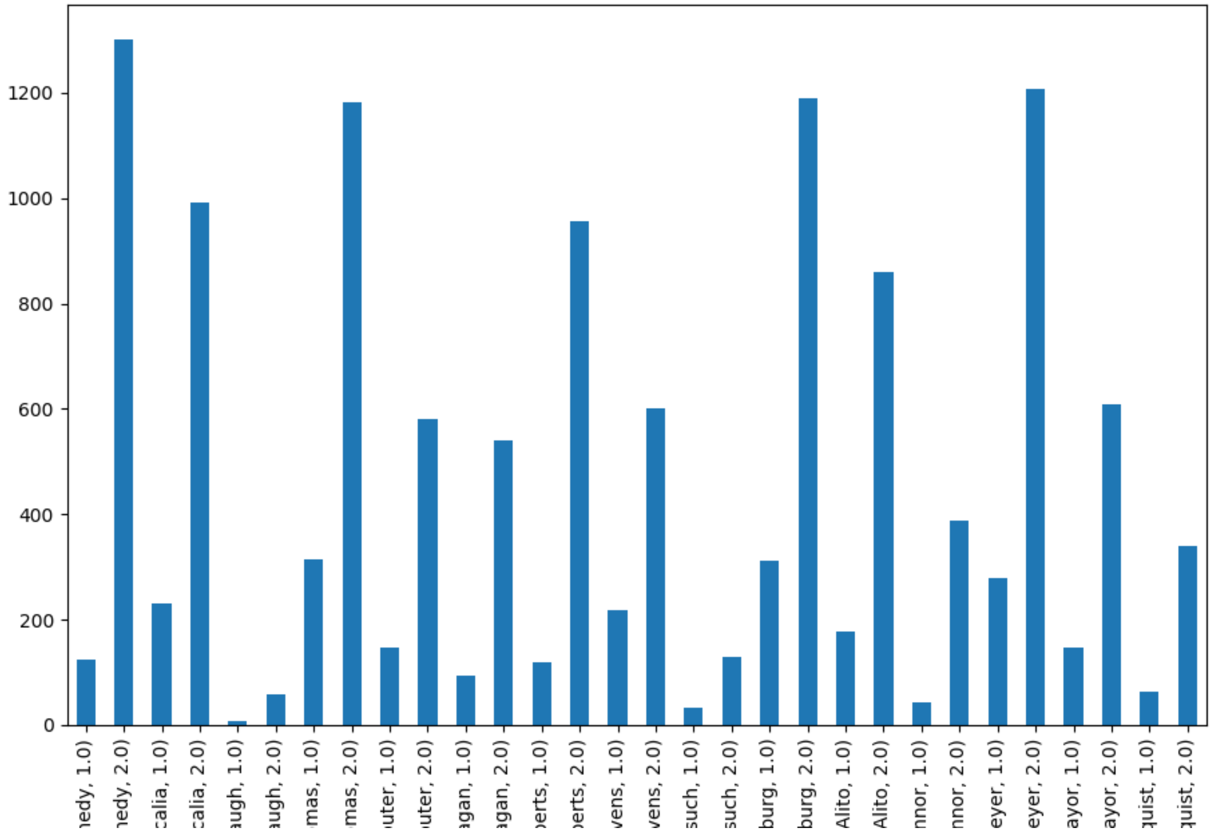


Subsetting Data again, this time inline

- Since the 2000 term, see how many times each justice has voted for the dissent and majority.

- And just to demo it, we'll do the subset of the terms inline

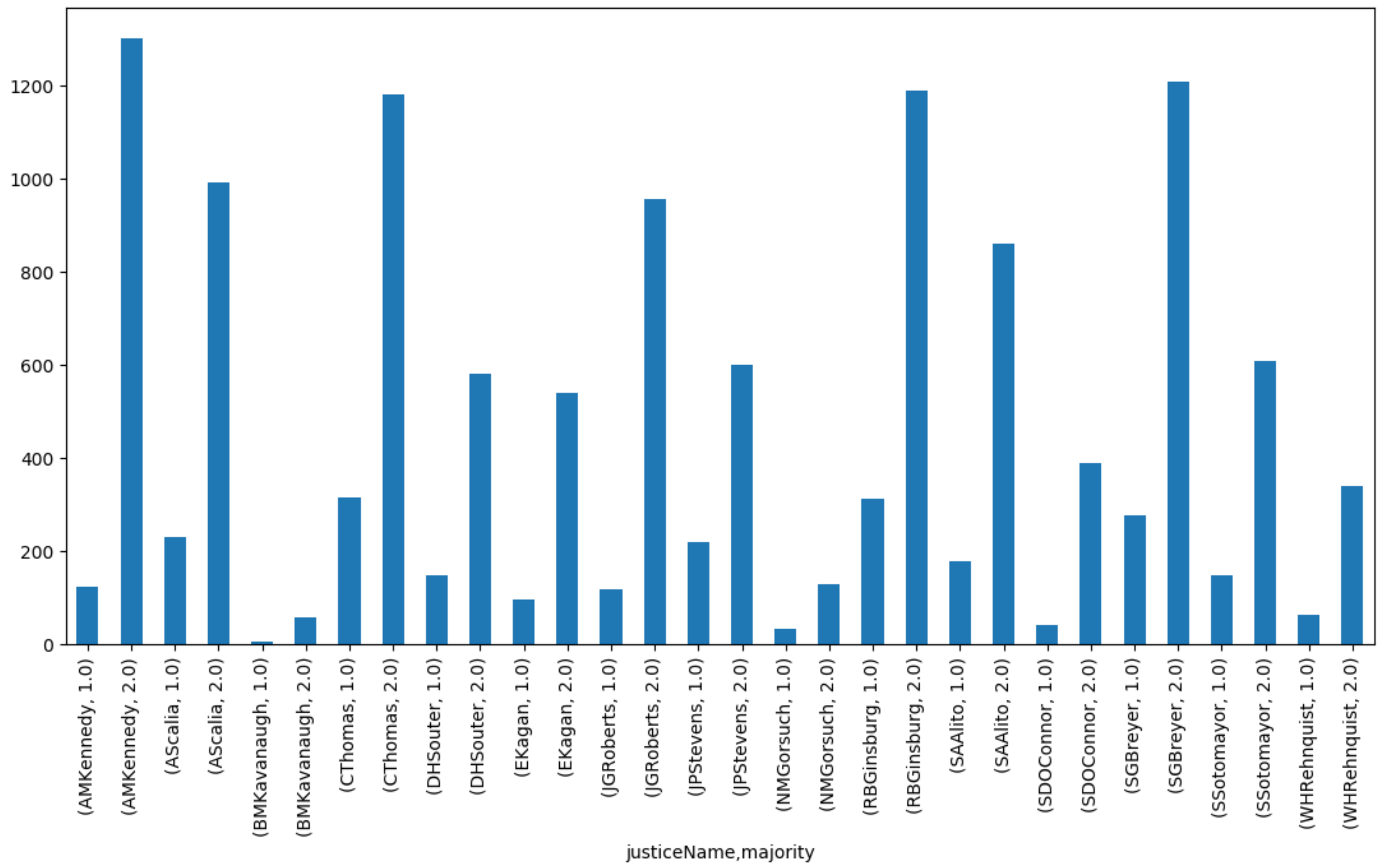
```
f = plt.figure(figsize=(11,7))
scdb[scdb.term >= 2000].groupby(['justiceName', 'majority'])[
            'caseId'].nunique().plot(kind="bar")
```



Problem with Some Plots Including Last One

- The names of the Justices ran off bottom of page in that plot!
- Because some of Justices' names are so long
- Happens sometimes
- Fix is: tight_layout()

```
f = plt.figure(figsize=(11,7))
scdb[scdb.term >= 2000].groupby(['justiceName', 'majority'])[
            'caseId'].nunique().plot(kind="bar")
f.tight_layout()
```



Assignment (Project)

1. Create the subset of all cases *from the 2000 term to the most recently completed term* (i.e., the end of what is in the database) to use for problems 2 and 3
2. Calculate and plot the number of cases for each term
3. For each justice, calculate and plot the number of votes in each "direction" (total, not by term)
4. Download Case centered data from the Supreme Court Database site. It has cases organized by Supreme Court decision. Restrict to cases from the 2006-2018 terms for problems 5 and 6
5. Calculate and plot the number of cases for each caseDisposition type
6. Create and briefly describe your own plot

Renaming Columns

First, let's create a subset called "scdb_subset" with the first 3 columns

```
scdb_subset = scdb.iloc[:,0:3]
scdb_subset.head()
>>> scdb_subset.head()
     caseId     docketId    caseIssuesId
0  1946-001  1946-001-01  1946-001-01-01
1  1946-001  1946-001-01  1946-001-01-01
2  1946-001  1946-001-01  1946-001-01-01
3  1946-001  1946-001-01  1946-001-01-01
4  1946-001  1946-001-01  1946-001-01-01
```

Rename a specific column

```
scdb_subset.rename(columns = {'caseId': 'case_Identifier'},
                    inplace = True)

>>> scdb_subset.head()
  case_Identifier     docketId    caseIssuesId
0        1946-001  1946-001-01  1946-001-01-01
1        1946-001  1946-001-01  1946-001-01-01
2        1946-001  1946-001-01  1946-001-01-01
3        1946-001  1946-001-01  1946-001-01-01
4        1946-001  1946-001-01  1946-001-01-01
```

Rename all columns:

```
scdb_subset.columns = ['case_identifier', 'docket_identifier',
            'caseIssues_identifier']



>>> scdb_subset.head()
  case_identifier docket_identifier caseIssues_identifier
0        1946-001       1946-001-01        1946-001-01-01
1        1946-001       1946-001-01        1946-001-01-01
2        1946-001       1946-001-01        1946-001-01-01
3        1946-001       1946-001-01        1946-001-01-01
4        1946-001       1946-001-01        1946-001-01-01
```