Pandas II

group by

Andrew Bray

Agenda

1. Review
2. Stuff
3. Group by

US Elections Data

- From MIT Elections Lab
- County-level Data
- President elections 2000 - 2016

```
# remotes::install_github("andrewpbray/boxofdata")
library(boxofdata)
library(tidyverse)
data(uselections)
dim(uselections)
```

```
## [1] 50524    11
```

```
names(uselections)
```

```
##  [1] "year"           "state"          "state_po"       "county"
##  [5] "FIPS"           "office"         "candidate"      "party"
##  [9] "candidatevotes" "totalvotes"     "version"
```

US Elections Data, cont.

year	state	county	candidate	party	candidatevotes	totalvotes
2000	Alabama	Autauga	Al Gore	democrat	4942	17208
2000	Alabama	Autauga	George W. Bush	republican	11993	17208
2000	Alabama	Autauga	Ralph Nader	green	160	17208
2000	Alabama	Autauga	Other	NA	113	17208
2000	Alabama	Baldwin	Al Gore	democrat	13997	56480
2000	Alabama	Baldwin	George W. Bush	republican	40872	56480
2000	Alabama	Baldwin	Ralph Nader	green	1033	56480
2000	Alabama	Baldwin	Other	NA	578	56480

Into Python

```
import pandas as pd
uselections = r.uselections
uselections.shape
```

```
## (50524, 11)
```

```
uselections.columns
```

```
## Index(['year', 'state', 'state_po', 'county', 'FIPS', 'office', 'candidate'
##        'party', 'candidatevotes', 'totalvotes', 'version'],
##       dtype='object')
```

```
uselections.dtypes
```

```
## year                 float64
## state                 object
## state_po              object
## county                object
## FIPS                 float64
## office                object
## candidate             object
## party                 object
```

Select columns

Method 1: Pass a *string* into `[ ]`...

```
uselections["county"]
```

```
## 0              Autauga
## 1              Autauga
## 2              Autauga
## 3              Autauga
## 4              Baldwin
##                ...
## 50519    District 40
## 50520    District 40
## 50521    District 99
## 50522    District 99
## 50523    District 99
## Name: county, Length: 50524, dtype: object
```

... get out a series.

Select columns

Method 2: Pass a *list* into `[ ]`…

```
uselections[["county"]]
```

```
##                county
## 0             Autauga
## 1             Autauga
## 2             Autauga
## 3             Autauga
## 4             Baldwin
## ...               ...
## 50519   District 40
## 50520   District 40
## 50521   District 99
## 50522   District 99
## 50523   District 99
##
## [50524 rows x 1 columns]
```

… get out a data frame.

Slicing rows

Method 1: Pass a *slice* into [ ]...

```
uselections[0:5]
```

```
##       year    state state_po  ... candidatevotes  totalvotes     version
## 0   2000.0  Alabama       AL  ...         4942.0     17208.0  20191203.0
## 1   2000.0  Alabama       AL  ...        11993.0     17208.0  20191203.0
## 2   2000.0  Alabama       AL  ...          160.0     17208.0  20191203.0
## 3   2000.0  Alabama       AL  ...          113.0     17208.0  20191203.0
## 4   2000.0  Alabama       AL  ...        13997.0     56480.0  20191203.0
##
## [5 rows x 11 columns]
```

... get out a slice data frame. Sound familiar?

```
slice(uselections, 1:2)
```

```
## # A tibble: 2 x 11
##    year state state_po county   FIPS office candidate party candidatevotes
##   <dbl> <chr> <chr>    <chr>   <dbl> <chr>  <chr>     <chr>          <dbl>
## 1  2000 Alab… AL       Autau…   1001 Presi… Al Gore   demo…           4942
## 2  2000 Alab… AL       Autau…   1001 Presi… George W… repu…          11993
## # … with 2 more variables: totalvotes <dbl>, version <dbl>
```

Selecting and slicing

Method 1: (preferred) access labels with `.loc`.

```
uselections.loc[0:5, ["county"]]
```

```
##     county
## 0  Autauga
## 1  Autauga
## 2  Autauga
## 3  Autauga
## 4  Baldwin
## 5  Baldwin
```

Selecting and slicing

Method 2: access integer indices with `.iloc`.

```
uselections.columns
```

```
## Index(['year', 'state', 'state_po', 'county', 'FIPS', 'office', 'candidate'
##        'party', 'candidatevotes', 'totalvotes', 'version'],
##       dtype='object')
```

```
uselections.iloc[0:5, 3]
```

```
## 0    Autauga
## 1    Autauga
## 2    Autauga
## 3    Autauga
## 4    Baldwin
## Name: county, dtype: object
```

Filtering rows

You can apply a Boolean series as a mask.

```
mask = uselections["year"] == 2016
uselections[mask]
```

```
##          year    state state_po  ... candidatevotes  totalvotes     version
## 40517  2016.0  Alabama       AL  ...         5936.0     24973.0  20191203.0
## 40518  2016.0  Alabama       AL  ...        18172.0     24973.0  20191203.0
## 40519  2016.0  Alabama       AL  ...          865.0     24973.0  20191203.0
## 40520  2016.0  Alabama       AL  ...        18458.0     95215.0  20191203.0
## 40521  2016.0  Alabama       AL  ...        72883.0     95215.0  20191203.0
## ...       ...      ...      ...  ...            ...         ...         ...
## 50519  2016.0   Alaska       AK  ...         1377.0      4610.0  20191203.0
## 50520  2016.0   Alaska       AK  ...          895.0      4610.0  20191203.0
## 50521  2016.0   Alaska       NA  ...          274.0      5056.0  20191203.0
## 50522  2016.0   Alaska       NA  ...           40.0      5056.0  20191203.0
## 50523  2016.0   Alaska       NA  ...           28.0      5056.0  20191203.0
##
## [9474 rows x 11 columns]
```

Filtering rows and selecting columns

Boolean mask plus a list of columns.

```
mask = uselections["year"].isin([2012, 2016])
uselections[mask, ["county", "state"]]
```

Will this run?

> Need to use `.loc`

```
mask = uselections["year"].isin([2012, 2016])
uselections.loc[mask, ["county", "state"]]
```

```
##              county    state
## 31166       Autauga  Alabama
## 31167       Autauga  Alabama
## 31168       Autauga  Alabama
## 31169       Baldwin  Alabama
## 31170       Baldwin  Alabama
## ...             ...      ...
## 50519   District 40   Alaska
## 50520   District 40   Alaska
## 50521   District 99   Alaska
## 50522   District 99   Alaska
```

Let's shine that up.

1. Form data frame.
2. Apply `.agg()` method.
3. Pass as the aggregation function the string method to `.join`.

Pandas Inventory

Now we know how to:

1. Select columns
2. Slice rows
3. Do both simultaneously
4. Filter rows using boolean masks
5. Add columns

Practice: Question 1

Practice: Question 1

Extract the first three rows where the candidate got more than 90% of the vote.

```
uselections["prop"] = uselections["candidatevotes"]/uselections["totalvo
```

Handy utility functions

`sort_values()`

```
uselections.sort_values("state")
```

`value_counts()`

```
uselections["year"].value_counts()
```

`unique()`

`sample()`

Practice: Question 2

Practice: Question 2

Which candidates were on the ballot in California in 2016?

```
uselections["candidate"].unique()
```

```
## array(['Al Gore', 'George W. Bush', 'Ralph Nader', 'Other', 'John Kerry',
##        'Barack Obama', 'John McCain', 'Mitt Romney', 'Hillary Clinton',
##        'Donald Trump'], dtype=object)
```

```
mask = (uselections["year"] == 2016) & (uselections["state_po"] == "CA"
uselections[mask]["candidate"].unique()
```

```
## array(['Hillary Clinton', 'Donald Trump', 'Other'], dtype=object)
```

Practice: Question 3

Which were the top 5 counties in California in 2016 in the proportion of the vote won by Hillary Clinton?

```
mask = (uselections["year"] == 2016) & (uselections["state_po"] == "CA"
df = uselections[mask]
df.sort_values("prop", ascending = False)["county"].head(5)
```

```
## 41099    San Francisco
## 40988          Alameda
## 41048            Marin
## 41108        San Mateo
## 41117       Santa Cruz
## Name: county, dtype: object
```

Groupby

For separate operations on subsets of the data frame, use *grouped* operations.

```
uselections.groupby("year")
```

```
## <pandas.core.groupby.generic.DataFrameGroupBy object at 0x7f8e001917f0>
```

```
uselections.groupby("year").agg(sum)
```

```
##                 FIPS  candidatevotes   totalvotes       version        prop
## year
## 2000.0  382202048.0     105411375.0  421645500.0  2.549745e+11  3152.000000
## 2004.0  286675578.0     122320549.0  366961647.0  1.912915e+11  3154.000000
## 2008.0  286675578.0     131187337.0  393562011.0  1.912915e+11  3154.000000
## 2012.0  286675578.0     129094316.0  387949353.0  1.912915e+11  3155.000000
## 2016.0  286675578.0     136495547.0  409605462.0  1.912915e+11  3152.918886
```

Practice: Question 4

For each county in California in 2016, calculate the proportion of votes for each major party candidate.