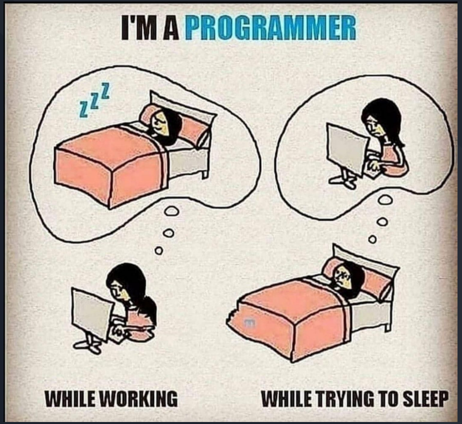
COGS 108 Week 5 A04/A07

Oct 30, 2023



Slides Adapted from Previous Quarter

slido

Do you have any communication issues with your group mates?

WHAT'S FOR TODAY

01 LOGISTICS

02 DISCUSSION LAB 4

LOGISTICS

- Project Proposal Due Wednesday Nov 3 @ 11:59 PM
- D4 Due Friday Nov 5 @ 11:59 PM

DISCUSSION LAB 4 DESCRIPTIVE AND EXPLORATORY DATA ANALYSIS

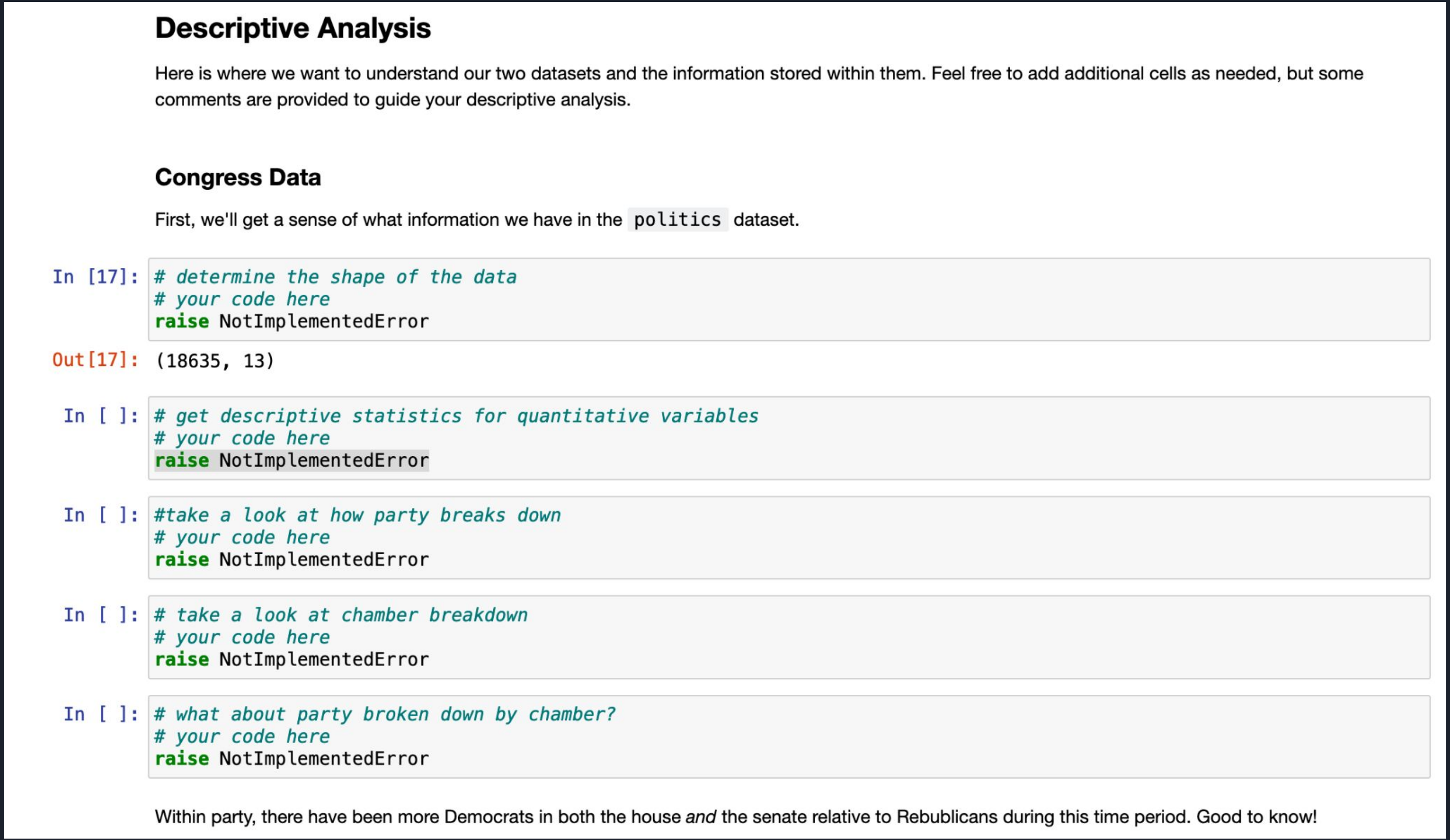
Web scraping tools

```
# packages helpful for webscraping
import requests
```

The requests library is the de facto standard for making HTTP requests in Python. It abstracts the complexities of making requests behind a beautiful, simple API so that you can focus on interacting with services and consuming data in your application.

```
import bs4
from bs4 import BeautifulSoup
```

Beautiful Soup is a Python library for pulling data out of HTML and XML files.



Descriptive Analysis

Here is where we want to understand our two datasets and the information stored within them. Feel free to add additional cells as needed, but some comments are provided to guide your descriptive analysis.

Congress Data

First, we'll get a sense of what information we have in the `politics` dataset.

```
In [17]: # determine the shape of the data
         # your code here
         raise NotImplementedError

Out[17]: (18635, 13)
```

```
In [ ]: # get descriptive statistics for quantitative variables
        # your code here
        raise NotImplementedError
```

```
In [ ]: #take a look at how party breaks down
        # your code here
        raise NotImplementedError
```

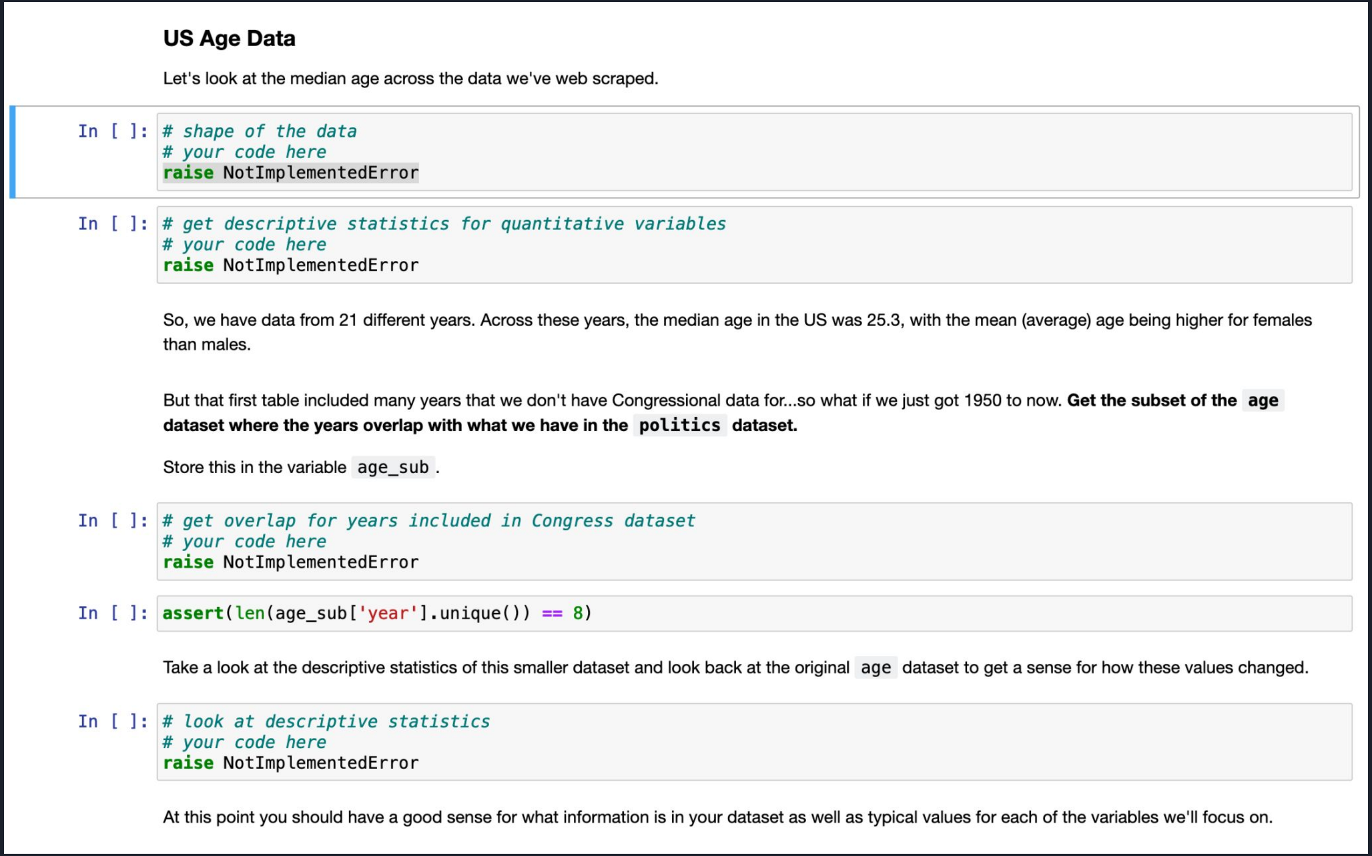
```
In [ ]: # take a look at chamber breakdown
        # your code here
        raise NotImplementedError
```

```
In [ ]: # what about party broken down by chamber?
        # your code here
        raise NotImplementedError
```

Within party, there have been more Democrats in both the house *and* the senate relative to Rebublicans during this time period. Good to know!

PART II: DESCRIPTIVE ANALYSIS

- Determine the shape of the data: shape
- Get descriptive statistics for quantitative variables: describe()
- Take a look at how party breaks down : value_counts()
- Take a look at chamber breakdown
- What about party broken down by chamber?



US Age Data

Let's look at the median age across the data we've web scraped.

```
In [ ]: # shape of the data
        # your code here
        raise NotImplementedError
```

```
In [ ]: # get descriptive statistics for quantitative variables
        # your code here
        raise NotImplementedError
```

So, we have data from 21 different years. Across these years, the median age in the US was 25.3, with the mean (average) age being higher for females than males.

But that first table included many years that we don't have Congressional data for...so what if we just got 1950 to now. **Get the subset of the `age` dataset where the years overlap with what we have in the `politics` dataset.**

Store this in the variable `age_sub`.

```
In [ ]: # get overlap for years included in Congress dataset
        # your code here
        raise NotImplementedError
```

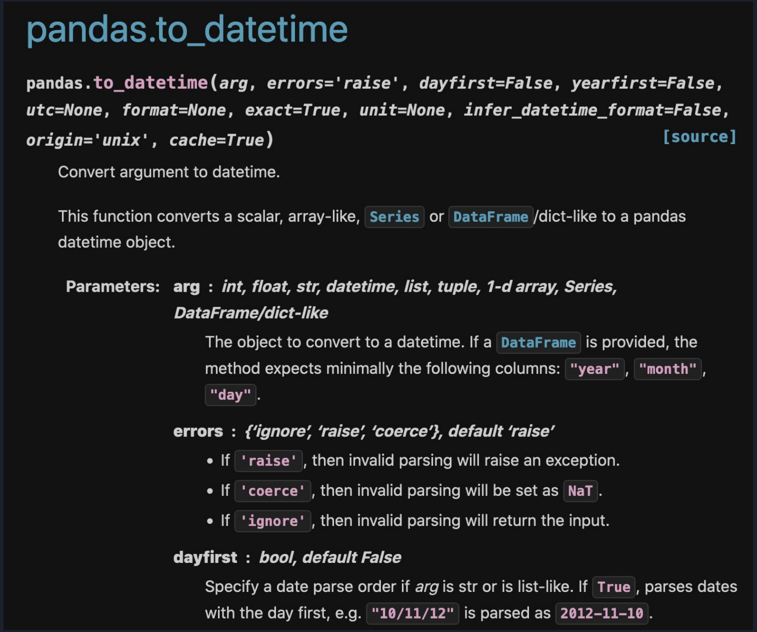
```
In [ ]: assert(len(age_sub['year'].unique()) == 8)
```

Take a look at the descriptive statistics of this smaller dataset and look back at the original `age` dataset to get a sense for how these values changed.

```
In [ ]: # look at descriptive statistics
        # your code here
        raise NotImplementedError
```

At this point you should have a good sense for what information is in your dataset as well as typical values for each of the variables we'll focus on.

EDA - date and time



pandas.to_datetime

`pandas.to_datetime(arg, errors='raise', dayfirst=False, yearfirst=False, utc=None, format=None, exact=True, unit=None, infer_datetime_format=False, origin='unix', cache=True)` [source]

Convert argument to datetime.

This function converts a scalar, array-like, `Series` or `DataFrame`/dict-like to a pandas datetime object.

Parameters: **arg** : *int, float, str, datetime, list, tuple, 1-d array, Series, DataFrame/dict-like*

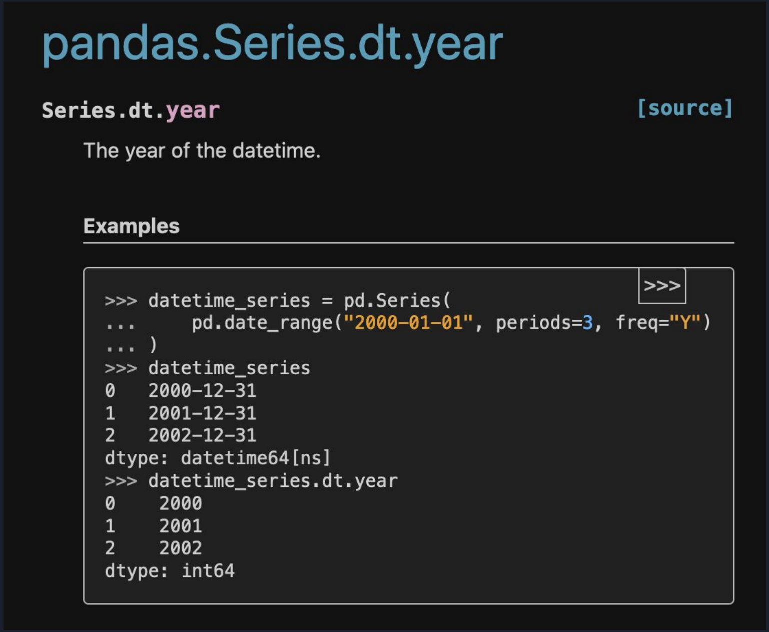
The object to convert to a datetime. If a `DataFrame` is provided, the method expects minimally the following columns: `"year"`, `"month"`, `"day"`.

errors : *{'ignore', 'raise', 'coerce'}, default 'raise'*

- If `'raise'`, then invalid parsing will raise an exception.
- If `'coerce'`, then invalid parsing will be set as `NaT`.
- If `'ignore'`, then invalid parsing will return the input.

dayfirst : *bool, default False*

Specify a date parse order if *arg* is str or is list-like. If `True`, parses dates with the day first, e.g. `"10/11/12"` is parsed as `2012-11-10`.



pandas.Series.dt.year

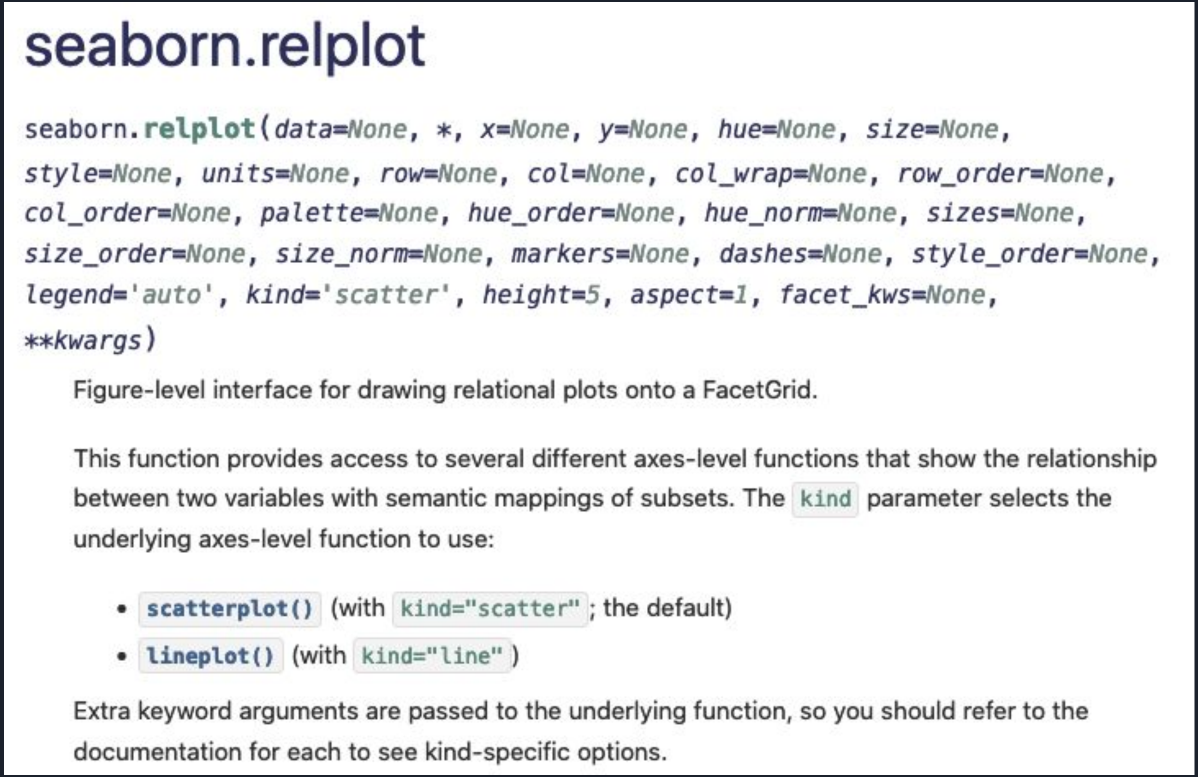
`Series.dt.year` [source]

The year of the datetime.

Examples

```
>>> datetime_series = pd.Series(
...     pd.date_range("2000-01-01", periods=3, freq="Y")
... )
>>> datetime_series
0   2000-12-31
1   2001-12-31
2   2002-12-31
dtype: datetime64[ns]
>>> datetime_series.dt.year
0    2000
1    2001
2    2002
dtype: int64
```

Relational Plots



seaborn.relplot

```
seaborn.relplot(data=None, *, x=None, y=None, hue=None, size=None,
style=None, units=None, row=None, col=None, col_wrap=None, row_order=None,
col_order=None, palette=None, hue_order=None, hue_norm=None, sizes=None,
size_order=None, size_norm=None, markers=None, dashes=None, style_order=None,
legend='auto', kind='scatter', height=5, aspect=1, facet_kws=None,
**kwargs)
```

Figure-level interface for drawing relational plots onto a FacetGrid.

This function provides access to several different axes-level functions that show the relationship between two variables with semantic mappings of subsets. The `kind` parameter selects the underlying axes-level function to use:

- `scatterplot()` (with `kind="scatter"`; the default)
- `lineplot()` (with `kind="line"`)

Extra keyword arguments are passed to the underlying function, so you should refer to the documentation for each to see kind-specific options.

THANKS!

Questions on Campuswire or office hours

Office hours: Tue/Thu, 4-5 PM