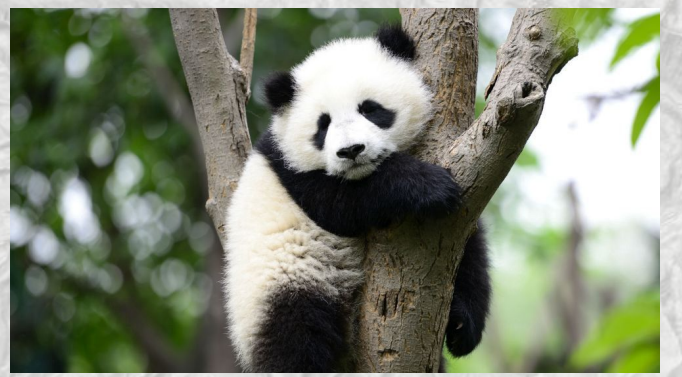
Putting Pandas to Work:
Solving GIS Problems in Python

Erik Neemann

UGIC 2020

Overview

- Big Picture
  - Pandas
  - GeoPandas
- Pandas Basics
- Examples
- Real World



UTAH AGRC
LOCATION MATTERS

Big Picture

Pandas Python library

- Open source Python library that provides "high-performance, easy-to-use data structures and data analysis tools"
- Closely tied to NumPy, SciPy, Matplotlib
- Core structure is the "Dataframe" (df)
  - Think "mini spreadsheet" within Python
  - Good for working with tabular data


```
import pandas as pd
```

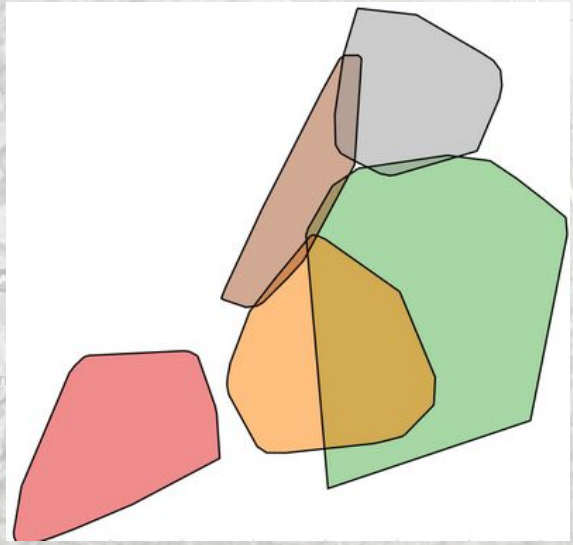
UTAH AGRC
LOCATION MATTERS

@pandas_dev

pandas.pydata.org

Big Picture

GeoPandas Python library



- Open source library "to make working with geospatial data in Python easier"
- Built upon Shapely, Fiona libraries
- Uses Matplotlib and Descartes for plotting
- Core structure is the "GeoDataframe" (gdf)
  - Think "feature class or shapefile" within Python
- Combines Pandas dataframe with geometry and enable spatial operations without a spatial database

```
import geopandas as gpd
```

UTAH AGRC
LOCATION MATTERS

@geopandas

geopandas.org

Pandas Basics

10 Minutes to Pandas: https://pandas.pydata.org/pandas-docs/stable/user_guide/10min.html
Pandas Intro Tutorial: https://www.youtube.com/watch?v=e60ItwlZTKM&t=1s

Creating data

- From Scratch
- From CSV
- From Excel
- From Shapefile
- From PostGIS (using psycopg2 library)

```
df = pd.DataFrame({'A': [1,2,3], 'B': [4,5,6], 'C': [7,8,9]})
df = pd.read_csv(r'path_to_file.csv')
df = pd.read_excel(r'path_to_file.xlsx')
gdf = gpd.read_file(r'path_to_file.shp')
```

```
con = psycopg2.connect(database="opensgid", user="agrc",
    password="agrc", host="opensgid.agrc.utah.gov")
sql = "select * from opensgid.boundaries.county_boundaries"
gdf = gpd.GeoDataFrame.from_postgis(sql, con, geom_col='shape')
```

UTAH AGRC
LOCATION MATTERS

Open SGID: gis.utah.gov/introducing-open-sgid/

Pandas Basics

Viewing data and properties

- Shape - dimensions of dataframe ➔ df.shape
- Head - first 5 rows of dataframe ➔ df.head()
- Tail - last 5 rows of dataframe ➔ df.tail()
- Datatypes - datatype of each column ➔ df.dtypes
- Columns - list column names ➔ df.columns
- Describe - display basics statistics ➔ df.describe()
- Sorting
  - df.sort_values('column_name', ascending=True, inplace=True)
  - sorted_df = df.sort_values('column_name', ascending=True)
    - Options exist for more complex sorting (multiple columns, location of NULLs, etc)

UTAH AGRC
LOCATION MATTERS

Pandas Basics

Getting data (slicing and selecting)

- **By Label** `df.loc[row, col_name]`
  - Treats labels as non-integers
    - `single_value = df.loc[2, 'col_name']`
    - `all_rows = df.loc[:, 'col_name']`
    - `all_cols = df.loc[2, :]`
    - `df.loc[2:6, ['name', 'countynbr', 'poplastcensus']]`
- **By Index** `df.iloc[row, col]`
  - Treats indexes as integers
  - Typical python behavior for slicing, excludes last
    - `single_value = df.iloc[2, 4]`
    - `all_rows = df.iloc[:, 4]`
    - `all_cols = df.iloc[2, :]`
    - `df.loc[2:6, [2, 1, 11]`

UTAH AGRC
LOCATION MATTERS

Pandas Basics

Filtering and subsetting data

- **Boolean indexing - uses true/false conditions to filter data into a subset**
  - Operators include '&' (and), '|' (or), '~' (not)
  - Extract rows where column A is greater than 2
    - `df[df['A'] > 2]`
  - Extract rows where column A is not greater than 2
    - `df[~(df['A'] > 2)]`
- **Other conditions**
  - isin() Method
    - `df[df['A'].isin([1, 2])]`
  - str.contains() Method
    - `city = munis[munis['name'].str.contains('City')]`

UTAH AGRC
LOCATION MATTERS

Pandas Basics

Adding new columns and assigning data

- Adding columns
  - `df['D'] = df['A'] + df['C']`
  - `df['F'] = [20, 30, 40]`
- Assigning values - very similar to getting values
  - Set negative values equal to 0
    - `df[df < 0] = 0`
  - Assign by entire columns or rows
    - `df.loc[:, 'F'] = 9999`
    - `df.iloc[2, :] = 5555`
  - Assign for individual cells
    - `df.loc[1, 'F'] = 33`
    - `df.iloc[2, 1] = 'new value'`

UTAH AGRC
LOCATION MATTERS

Pandas Basics

Cleaning and working with data

- Drop columns
  - `df.drop(columns=['Name', 'Age', 'Birthday'],  inplace=True)`
- Drop duplicates
  - `df.drop_duplicates('Name', inplace=True)`
- Iterate over rows
  - Caution: not efficient for large dataframes, shouldn't be used to modify data
    - `for index, row in df.iterrows():`
    - `    print(row['column1'], row[column2])`
- Apply function to every row (axis=1) or column (axis=0)
  - `sums = df.apply(np.sum, axis=1)`
- Strip whitespace from strings
  - `clean = dirty_df.apply(lambda x: x.str.strip())`

UTAH AGRC
LOCATION MATTERS

Pandas Basics

Handling missing data

- Drop NULLs (can choose axis - row is default)
  - Remove rows that have any NULL values
    - df.dropna(how='any', subset=['col1', 'col2'], inplace=True)
- Fill NULLs
  - Replace NULL values with specific value
    - df[int_fields] = df[int_fields].fillna(9999)
    - df['str_field'].fillna('N', inplace=True)
- Other Methods
  - Fill NULLs with an average value in column
  - Fill NULLs with average of surrounding cells

UTAH AGRC
LOCATION MATTERS

Pandas Basics

Merging data

- Concatenate - combine multiple dataframes
  - Axis can be specified (rows or columns)
    - result = pd.concat([df1, df2, df3])
- Append - limited case of concatenate
  - Add rows from a dataframe(s) to another
    - result = df1.append(df2)
- Merge - all standard database-style joins (left, right, outer, inner)
  - result = pd.merge(left, right, on='key')
- Join - convenient database-style join using index or key column
  - result = left.join(right)
  - result = left.join(right.set_index('right_key'), on='left_key', how='inner')

UTAH AGRC
LOCATION MATTERS

Pandas Basics

Writing out data

- CSV
  - `df.to_csv(r'path_to_file.csv')`
- Excel
  - `df.to_excel(r'path_to_file.xlsx', sheet_name='Sheet1')`
  - Multiple sheets:
    - `with pd.ExcelWriter('output.xlsx') as writer:`
    - `    df1.to_excel(writer, sheet_name='Sheet1')`
    - `    df2.to_excel(writer, sheet_name='Sheet2')`
- Shapefile - using geopandas
  - `gdf.to_file(driver = 'ESRI Shapefile', filename='path_to_file.shp')`

UTAH AGRC
LOCATION MATTERS

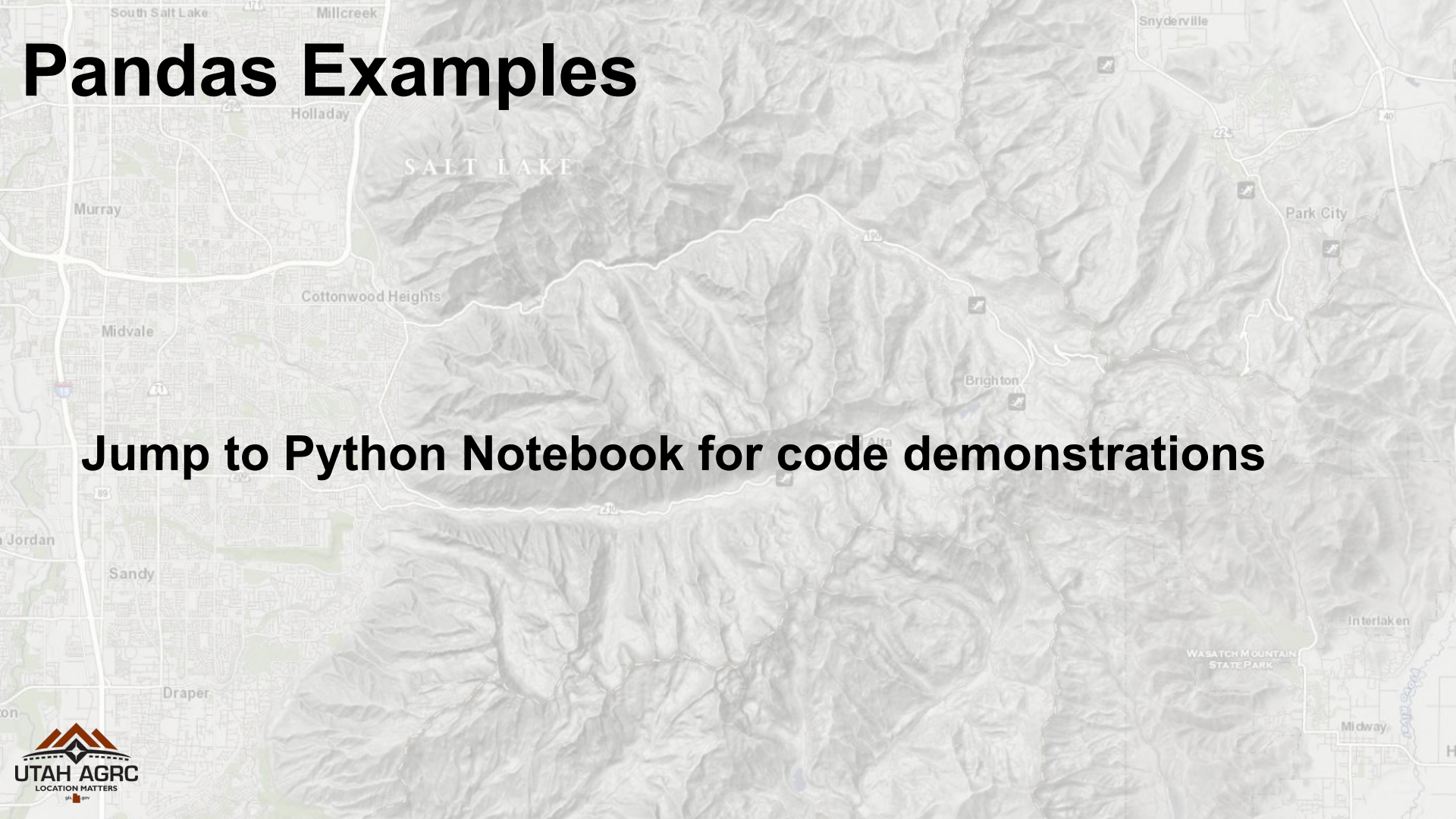
Pandas Basics

Additional capabilities

- Groupby
  - Very similar to SQL functionality
  - Group and aggregate data
- Plotting
  - Simple and streamlined with MatPlotlib integration
  - Geopandas allows for easy, simple mapping (uses Descartes)
  - https://geopandas.org/mapping.html
- Pivot Tables
- Categorical Data
- Time Series Data

UTAH AGRC
LOCATION MATTERS

Pandas Examples



Jump to Python Notebook for code demonstrations

UTAH AGRC
LOCATION MATTERS

Pandas Real World Applications

Build dispatch street segments from a spreadsheet

B18

	A	B	C	D	E	F	G	H
1	STREET	CITYCD	BEG	END	Y START	X START	Y END	X END
2	E TANOAK DR	STG	3055	3190	37.048661	-113.520661	37.049286	-113.517907
3	E TANOAK CIR	STG	3054	3021	37.048641	-113.521491	37.048661	-113.520661
4	BOONE PARK CIR	SAN	1423	1495	37.135281	-113.670061	37.133161	-113.670091
5	E DESERT CANYONS PKWY	STG	3661	3891	37.005375	-113.508266	37.006641	-113.503571
6	N RIMVIEW DR	STG	1886	1906	37.155245	-113.463212	37.156831	-113.463281
7	W CANYON VIEW DR	STG	1521	1600	37.091911	-113.615588	37.091789	-113.617363
8	S HIGHWAY 59	HUR	2489	2490	37.128451	-113.218291	37.128449	-113.218281
9	S SANDHILL DR	WAS	1189	1316	37.114581	-113.482351	37.112648	-113.481966
10	S 2050 E	APP	1200	1400	37.053531	-113.060311	37.050721	-113.060321
11								

```
for index, row in df.iterrows():
    array = arcpy.Array([arcpy.Point(float(row['X START']), float(row['Y START'])),
                         arcpy.Point(float(row['X END']), float(row['Y END']))])
    shape = arcpy.Polyline(array, spatial_reference)
    values = [row['STREET'],
          row['CITYCD'],
          row['BEG'],
          row['END'],
          row['X START'],
          row['Y START'],
          row['X END'],
          row['Y END'],
          shape]

    # add line to FC
    print('Adding line to feature class...')
    with arcpy.da.InsertCursor(segments, fields) as iCur:
        iCur.insertRow(values)
```

- Iterate through df rows
- Build list of field values
- Insert row into feature class with ArcPy Insert Cursor

Pandas Real World Applications

Compare address points to nearby street segments

- Create near table in ArcPy
- Convert near table to dataframe, same with address points and roads
  - `near_arr = arcpy.da.TableToNumPyArray(neartable, '*')`
  - `near_df = pd.DataFrame(data = near_arr)`
- Join data from address points and road segments
  - `near_df.join(addpts_df.set_index('OBJECTID'), on='IN_FID')`
  - `near_df.join(streets_df.set_index('OBJECTID'), on='NEAR_FID')`
- Perform QA checks with .apply() `near_df.apply(QA_checks, axis=1)`
  - `if row['Road_StName'] == row['Addpt_StName']:`
  - `    goodstreet = True`
- Join data back to address point feature class

UTAH AGRC
LOCATION MATTERS

Pandas Real World Applications

Compare address points to nearby street segments

- Create near table in ArcPy
- Conv… …s and roads
  - n…
  - n…
- Join d…
  - n…
  - n… …ID')
- Perfo… …ecks, axis=1)
  - if …
  - goodstreet = True
- Join data back to address point feature class

	A	B	C	D	E	F
1		OBJECTID	IN_FID	NEAR_FID	NEAR_DIST	NEAR_RANK
2	0	1	90	19864	67.8876476352821	1
3	1	2	90	19863	111.569007639289	2
4	2	3	90	19866	111.569007639289	2
5	3	4	90	19867	117.941941947675	3
6	4	5	90	19870	151.398857182606	4
7	5	6	169	19347	20.0672922399109	1
8	6	7	169	10963	28.2418644608854	2
9	7	8	169	11319	85.5317444876772	3
10	8	9	169	17493	102.02164353072	4
11	9	10	169	11503	106.107876914198	5
12	10	11	286	20484	15.5836864179079	1

UTAH AGRC
LOCATION MATTERS

Pandas Real World Applications

Update COVID dashboard based on Google Sheet changes

- Needed to move updated numbers from G Sheet into ArcGIS online layer
- Read Google Sheet (csv) in as dataframe `pd.read_csv('GSheet.csv')`
- ID new rows and send addresses to geocoder `new.apply(geocode, axis=1)`
- Insert new rows into AGOL layer `new.apply(insert_row, axis=1)`
- Use arcpy.da.UpdateCursor on AGOL layer
  - Create temporary dataframe for each iteration on a row
    - `temp_df = updates.loc[updates['UniqueID'] == row[0]]`
  - Check AGOL values against updated dataframe values
  - Update AGOL when differences found
    - `row[5] = temp_df.iloc[0]['Positive_Patients']`

UTAH AGRC
LOCATION MATTERS

Questions?

UTAH AGRC

LOCATION MATTERS

gis.[utah].gov

Erik Neemann (eneemann@utah.gov)