Capstone 1 Data Wrangling

Vicki Brown

The first step in completing your capstone project is to collect data. Depending on your dataset, you may apply some of the data wrangling techniques that you learned in this unit. Some of you may be using standard datasets and sources, such as Kaggle or Yelp, where minimal or no data wrangling is required. Students often find that this part of the project takes a lot longer than they estimated, which is completely normal. The more work you put in, the more you'll learn. Data wrangling is an important tool in a data scientist's toolbox!

Learning Objective

- Collect the data.
- Clean the dataset and address issues like missing values and outliers.
- Apply data wrangling techniques, including standard datasets and sources.

Data Wrangling steps

- Review the data set(s).
- What kind of cleaning steps did you perform?
- How did you deal with missing values, if any?
- Were there outliers, and how did you handle them?

Water Quality Data

Data Acquisition

Access

No API is available. I used the "Expert Query" form, requesting data in three chunks, saved as CSV files:

- Julian Date < 1999001
- 1999001 < Julian Date < 2009001
- Julian Date > 2009001

Note: Water quality data is also available for download from ScienceBase; however, that archive includes fewer parameters and is not as up to date as the database at `sfbay.wr.usgs.gov`.

Files

1. SFBayWaterQuality1969-1998.csv
2. SFBayWaterQuality1999-2008.csv
3. SFBayWaterQuality2009-2019.csv

Review the data set(s)

Data Format

All files are formatted as CSV (comma-separated values).

Water Quality files have two header rows; the second row shows units of measure.

```
Date, Time, Station Number, Distance from 36, Depth, Discrete Chlorophyll, Chlorophyll a/a+PHA,
Fluorescence, Calculated Chlorophyll, Discrete Oxygen, Oxygen Electrode Output, Oxygen Saturation
%, Calculated Oxygen, Discrete SPM, Optical Backscatter, Calculated SPM, Measured Extinction
Coefficient, Calculated Extinction Coefficient, Salinity, Temperature, Sigma-t, Nitrite, Nitrate +
Nitrite, Ammonium, Phosphate, Silicate

MM/DD/YYYY, 24 hr., , [km], [meters], [mg/m3], , [volts], [mg/m3], [mg/L], [volts], , [mg/L],
[mg/L], [volts], [mg/L], [per meter], [per meter], [psu], [°C], [kg/m3], [µM], [µM], [µM], [µM],
[µM]
```

Cleaning

What kind of cleaning steps were performed?

- Combine multiple datasets.
- Handle multi-level index for water quality columns.
- Convert Date and Time columns to DateTime.
- Remove columns that are not useful.

Combine multiple datasets

The data is imported in three files. These files have identical columns, so they can easily be concatenated into one DataFrame and exported to a single file.

Handle multi-level index for water quality columns

The original Water Quality CSV files had two-row column headers. The second level is units.

```
MultiIndex([(                            'Date',          'MM/DD/YYYY'),
            (                            'Time',              '24 hr.'),
            (                  'Station Number',  'Unnamed: 2_level_1'),
            (                'Distance from 36',                '[km]'),
            (                           'Depth',            '[meters]'),
            (            'Discrete Chlorophyll',             '[mg/m3]'),
               ...
            (                        'Silicate',                '[µM]')],
           )
```

It will be easier to work with the data if I save the units into a dictionary and change the DataFrame to only have one level of headers.

Convert Date and Time columns to DateTime

The initial dataset has a Date column and a Time column. We would like to have a single DateTime column.

Issues:

- The initial Date column is type `string` (e.g., `M/D/YYYY`), no leading zeroes on day or month, but possibly a leading space. Conveniently, `pd.to_datetime` is able to convert this to DateTime format without trouble.
- The initial Time column is type `int`, no leading zeroes on the hour. To concatenate this to the Date column, I need it to be type `string`, 0-padded.

Once I have two strings, I can concatenate them into a new DateTime column and convert that to Datetime format.

Remove Columns that are not useful

Optical Backscatter

According to the data dictionary, due to sensor changes and gain differences, this value is only comparable within cruises and may not be comparable between cruises.

Discrete vs Calculated values: chlorophyll, SPM, and O2

The USGS contact suggested that I should ignore "discrete" values for chlorophyll, SPM, and oxygen, using the "calculated" values instead. The latter values are calculated using linear regression between the discrete values and other measurements.

A quick look confirmed that summary statistics for these calculated values match those for the comparable discrete values.

I removed the "discrete" columns from the dataset.

Time

I have a `DateTime` column, so I no longer need the `Time` column. I removed it as well.

Missing Values

How did you deal with missing values, if any?

- Some parameters are not checked for every sample.

Some parameters are not checked for every sample

Some of the water quality parameters (particularly the dissolved "nutrients") are not checked for every sample. However, there are no columns that are entirely missing data and no obvious overlap of columns.

Further investigation indicates that nutrient (e.g., nitrate, phosphate, ammonium) samples are typically only taken near the surface (e.g., 1 - 2 m depth). This was confirmed by the USGS contact for the dataset, who told me that we can assume dissolved nutrient values from surface to bottom are generally the same. "The Bay is well mixed."

Solution: These values are not "missing".

Given that nutrients are sampled for only a subset of records (primarily shallower depths), I created a separate DataFrame in which these samples are present.

Outliers

Were there outliers, and how did you handle them?

- Determine outliers using statistics and plotting.

Determine outliers using statistics and plotting

I calculated summary statistics per sampling station for each remaining column in the dataset. The statistics included `zscore`. I also plotted values in scatter plots by date.

There are a few apparent outliers for several parameters at most stations. However, at this point in the analysis, I do not anticipate that these will cause problems. If I need to remove them at a future date, I will easily be able to locate them.

Station Location Information

Data Acquisition

Access

Location data for "standard" stations is available from ScienceBase.

However, more complete location data is available in tables at sfbay.wr.usgs.gov. These tables include the general location of each station (by geographical landmark) as well as data for "non-standard" stations which are sampled less often.

These tables were copied, pasted into a spreadsheet, then exported as CSV. Header fields were edited to remove newlines and several fields were modified to remove artifacts before exporting to CSV format.

File

SFBayStationLocationsTable.csv

Review the data set

Data Format

The Station Locations file is CSV format with one header row and 5 columns.

```
Station Number, General Location, North Latitude, West Longitude, Depth MLW (meters)
```

Cleaning

What kind of cleaning steps were performed?

I did some cleaning of the data in Numbers (spreadsheet app) before importing into Jupyter Notebooks. I removed

- newlines in the column headers
- non-ASCII characters in the data
- extraneous quotation marks
- an asterix in a station number field

Missing Data

How did you deal with missing values, if any?

- Many records in the table do not include geographic degrees.

Many records in the table do not include geographic degrees

Many records inherit degrees from the station on the row above.

I want to fill in this data.

```
st_df.head()
```

	Station Number	General Location	North Latitude	West Longitude	Depth MLW (m)
0	657	Rio Vista	38 9.1'	-121 41.3'	10.1
1	649	Sacramento River	3.6'	48.0'	10.1
2	2	Chain Island	3.8'	51.1'	11.3
3	3	Pittsburg	3.1'	52.8'	11.3
4	4	Simmons Point	2.9'	56.1'	11.6

I created four new columns, extracting the degrees and minutes from the previously existing `North Latitude` and `West Longitude`columns:

- North Lat Degrees
- North Lat Minutes
- West Long Degrees
- West Ling Minutes

I then filled in missing degrees using the most recent non-null value above.

When the new columns were filled, I dropped the original `North Latitude` and `West Longitude` columns and reordered the DataFrame columns.

Phytoplankton Data

Data Acquisition

Access

Phytoplankton data was downloaded from * ScienceBase: Phytoplankton, 1992-2015 * ScienceBase: Phytoplankton, 2016...

Files

1. Phytoplankton_San_Francisco_Bay_1992_2014.csv
2. Phytoplankton_San_Francisco_Bay_2014_2016.csv
3. Phytoplankton_San_Francisco_Bay_2017-2018.csv

Note: The apparent overlap in dates for Phytoplankton files, `Phytoplankton_San_Francisco_Bay_1992_2014.csv` and `Phytoplankton_San_Francisco_Bay_2014_2016.csv`, is not an error. Data from the first three months of 2014 is included in two separate files. Possible redundant rows from 2014 will need to be investigated.

Review the data set(s)

Data Format

All files are formatted as CSV (comma-separated values).

Phytoplankton files have one header row. Column headers include units of measure. These will be stripped.

```
Taxonomic Identification, Phylum or Class, Date,Station Number, Depth (m), Actual Count, Density (cells/mL), Biovolume (cubic micrometers/mL), Cell Volume (cubic micrometers/cell)
```

Note: The Actual Count parameter was not included before 2014. The recommendation from USGS is to not use this field, as it is essentially "for internal use only".

Cleaning

What kind of cleaning steps were performed?

- Handle file encoding.
- Normalize data formats.
- Combine multiple datasets.
- Handle redundant records.
- Generate DateTime values.
- Remove useless columns.
- Re-order remaining columns.

File Encoding

When reading in the Phytoplankton data

```
ph_df1 = pd.read_csv('Data/Phytoplankton_San_Francisco_Bay_1992_2014.csv', header=0)
```

I got an error:

```
UnicodeDecodeError                        Traceback (most recent call last)

<ipython-input-5-0cbe2ac6151a> in <module>

      1 # Read in Phytoplankton data

----> 2 ph_df1 = pd.read_csv('Data/Phytoplankton_San_Francisco_Bay_1992_2014.csv', header=0)

      3 ph_df2 = pd.read_csv('Data/Phytoplankton_San_Francisco_Bay_2014_2016.csv', header=0)

      4 ph_df3 = pd.read_csv('Data/Phytoplankton_San_Francisco_Bay_2017-2018.csv', header=0)

...

UnicodeDecodeError 'utf-8' codec can't decode byte 0xa0 in position 13: invalid start byte
```

A web search tells me that this is due to file encoding.

Changing the encoding in the `read` call:

```
ph_df1 = pd.read_csv('Data/Phytoplankton_San_Francisco_Bay_1992_2014.csv', header=0,
encoding='latin1')
```

works and does not throw any errors.

Now that I understand file encoding a bit better, I've fixed the encoding of the CSV files on disk to be UTF8.

Combine multiple datasets

Issue - Mis-aligned columns

One of the imported Phytoplankton datasets has one fewer columns than the other two. Thus, if we simply try a concatenation, we get an error:

```
/Local/.../ipykernel_launcher.py:2: FutureWarning: Sorting because non-concatenation axis is not
aligned. A future version of pandas will change to not sort by default.
```

Setting `sort=True` aligns the columns-in-common, allowing concatenation with the newer column not causing problems.

Possibly redundant records in phytoplankton data for 2014

From the filenames, I suspected there might be overlap for part of 2014. An "overlapping" row would match on every field except for Actual Count, which would be either a number or NaN.

I examined some of the 2014 data, using

```
ph_2014_df = ph_df.loc[ph_df['Date'].str.match('[12]/.*/2014')]

ph_2014_df.sort_values(by=['Date', 'Station Number', 'Depth (m)', 'Density (cells/mL)'])
```

I did not see any clear overlap.

- output table, partial:

	Actual Count	Biovolume (cubic micrometers/mL)	Cell Volume (cubic micrometers/cell)	Date	Density (cells/mL)	Depth (m)	Phylum or Class	Station Number	Taxonomic Identification
16451	1.0	182.2	911.1	1/14/2014	0.2	2.0	BACILLARIOPHYTA	6.0	Navicula spp.
16461	1.0	120.4	601.9	1/14/2014	0.2	2.0	CHLOROPHYTA	6.0	Pyramimonas spp.
16277	NaN	1314.2	3285.5	1/14/2014	0.4	2.0	BACILLARIOPHYTA	6.0	Thalassiosira nordenskioeldii
16283	NaN	1221.4	3053.5	1/14/2014	0.4	2.0	DINOPHYCEAE	6.0	Karlodinium veneficum
16275	NaN	251.4	157.1	1/14/2014	1.6	2.0	BACILLARIOPHYTA	6.0	Chaetoceros subtilis
16274	NaN	38706.4	16127.7	1/14/2014	2.4	2.0	BACILLARIOPHYTA	6.0	Actinocyclus normanii
16448	1.0	27033.0	7447.1	1/14/2014	3.6	2.0	BACILLARIOPHYTA	6.0	Cyclotella striata
16449	1.0	6589.2	1815.2	1/14/2014	3.6	2.0	BACILLARIOPHYTA	6.0	Diploneis sp.
16453	1.0	3626.0	998.9	1/14/2014	3.6	2.0	BACILLARIOPHYTA	6.0	Rhoicosphenia marina
16455	1.0	871.7	240.1	1/14/2014	3.6	2.0	BACILLARIOPHYTA	6.0	Thalassionema nitzschioides
16459	1.0	1737.9	478.7	1/14/2014	3.6	2.0	BACILLARIOPHYTA	6.0	Tryblionella granulata
16276	NaN	7068.4	1767.1	1/14/2014	4.0	2.0	BACILLARIOPHYTA	6.0	Cyclotella striata
16282	NaN	1105.7	251.3	1/14/2014	4.4	2.0	DINOPHYCEAE	6.0	Heterocapsa rotundata
16280	NaN	29027.5	2591.7	1/14/2014	11.2	2.0	CRYPTOPHYTA	6.0	Rhodomonas marina
16281	NaN	10052.0	502.6	1/14/2014	20.0	2.0	CRYPTOPHYTA	6.0	Teleaulax amphioxeia
16442	1.0	78746.8	2944.9	1/14/2014	26.7	2.0	BACILLARIOPHYTA	6.0	Actinocyclus sp.
16444	1.0	11780.5	440.6	1/14/2014	26.7	2.0	BACILLARIOPHYTA	6.0	Cocconeis placentula
16450	1.0	26296.7	983.4	1/14/2014	26.7	2.0	BACILLARIOPHYTA	6.0	Epithemia sp.
16463	1.0	43006.3	1608.3	1/14/2014	26.7	2.0	CRYPTOPHYTA	6.0	Chroomonas sp.
16468	1.0	47773.1	1786.6	1/14/2014	26.7	2.0	EUGLENOPHYTA	6.0	Unknown Euglenophyte

DateTime values

The format of the Date column is such that pandas already sees it as a DateTime.

Reorder Columns

Due to the need to sort columns before concatenation, the data columns are now in alphabetical order, left to right.

Before saving, I re-ordered the columns, moving Date, Station number, and Depth to the front.

Missing Data

How did you deal with missing values, if any?

- Drop Actual Count column.
- Remove columns that are not useful.

Drop Actual Count column.

The only missing data is in the Actual Count column. This value was not part of the dataset before 2014.

Actual Count is described as the "Number of phytoplankton cells counted in the sample". However, the sample size is not provided.

Actual Count values seem to be at odds with Density values. Density is described as the "Number of phytoplankton cells per milliliter of water. Most "actual counts" are very small; the mean is 25 cells. But Density is much larger; that mean is 2803.

I asked my USGS contact. Her recommendation is that I ignore the Actual Count field.

- It's primarily a "notation" field; someone saw these cells under a microscope.
- The sample size is indeterminable.
- It's unrelated to the rest of the data.

I removed this column.

Remove columns that are not useful

My USGS contact also recommended removing the Density and Cell Volume columns, concentrating instead on `Biovolume`.I removed these two columns and saved the dataset to disk.

Outliers

Were there outliers, and how did you handle them?

- Outliers are not a concern.

I am not particularly worried about outliers in this dataset as the only numeric value is a calculated value.

Consolidate Biovolume and Merge with Water Quality

The original phytoplankton data has multiple entries for each date and station, separating the phytoplankton by genus and species. Per recommendation from Tara Schraga at USGS, I have consolidated this information for each {date, station, depth} combination, producing a simple sum total of biovolume.

I merged the resulting column into the WQ DataFrame.