Data challenges

- Non-standardized data
- Inconveniently structured data
 - Tidy-ing data
 - Data with multiple factors
- Duplicate data
- Incorrect values
- Missing values



Standardizing data

Raw Year	Standardized
2019	2019
'19	2019

Raw Medication	Standardized
azithromycin	azithromycin
Zithromax	azithromycin

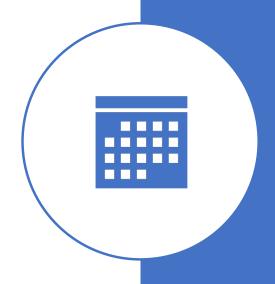
Raw Name	Standardized
McDougal	McDougal
mcdougal	McDougal

Raw Unit	Standardized
micron	μm
μm	μm

On dates

- ISO 8601 is an international standard for date-time information.
 - 20191001T182618+0000
- A challenge with this is that it requires dates be parsed again to do calculations.
 - It may be easier to store this as separate fields: year, month, day, hour, minute, seconds.
- Can use pd.to_datetime to convert to DateTime objects to allow subtraction, comparison.
- An alternative: Unix time
 - Number of seconds since 1 January 1970 UTC.
 - Returned by time.time() or e.g.

```
((pd.to_datetime('June 10, 2020') -
  pd.to_datetime('January 1, 1970')) /
pd.Timedelta('1 sec'))
```



Tidy Data

A data structuring approach.

Every variable has its own column.

Every observation has its own row.

Every value has its own cell.

TODO: Melting

• Example and code



"8-mg Zofran"

Dosage (mg): 8

Medicine: Zofran



"effusion of the right knee"

Condition: "knee effusion"

Side: "right"



"white male"

Gender: male

Ethnicity: Caucasian

Data with multiple factors

Duplicate data

- Sometimes a data point (row) may be listed more than once, especially if manual entry was involved.
- But be careful: depending on how your data is structured, it may also be the case that data should appear more than once.
 - Imagine, e.g. a patient sent home from the hospital only to return later that day with the same conditions.

See duplicate rows:

Pid should uniquely identify a patient.

Date and pid *almost* uniquely identifies an encounter.

See duplicate rows:

See duplicate rows:

Drop duplicate rows

```
>>> deduplicated_data = data.drop_duplicates()
>>> deduplicated_data
         last first
                                            condition
   pid
                             date
                    age
  1002 Smith John 42
                          20191001
                                             diabetes
  4261 Smith Jane 46
                                   pulmonary embolism
1
                          20190510
3
  4171 Smith Janet 16
                          20190909
                                                acne
```

 Both duplicated and drop_duplicates take an optional subset keyword argument specifying which columns to pay attention to.

```
>>> data duplicated(subset=['last'])
0    False
1    True
2    True
3    True
dtype: bool
```

Incorrect values

- Define and check ranges
 - If a person is 57 years old, that is plausible. If a person is 577 years old, then maybe there is something wrong.
- Check categorical values
 - e.g. is the "State" field correct? We know the list of all possible states.
- Look for inconsistencies
 - e.g. City: "New Haven", Zip: "90210"
- Look at outliers
 - If only one person has a disease, it could be very rare... or it could be a typo.
- Validate when possible.