CLEANING DATA IN PYTHON

Tidy data

Tidy data

- "Tidy Data" paper by Hadley Wickham, PhD

- Formalize the way we describe the shape of data

- Gives us a goal when formatting our data

- "Standard way to organize data values within a dataset"

Motivation for tidy data

	name	treatment a	treatment b
0	Daniel	-	42
1	John	12	31
2	Jane	24	27

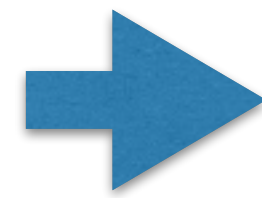|	**0**	**1**	**2**
name	Daniel	John	Jane
treatment a	-	12	24
treatment b	42	31	27

Principles of tidy data

- Columns represent separate variables

- Rows represent individual observations

- Observational units form tables

	name	treatment a	treatment b
0	Daniel	-	42
1	John	12	31
2	Jane	24	27

Converting to tidy data

	name	treatment a	treatment b
0	Daniel	-	42
1	John	12	31
2	Jane	24	27

	name	treatment	value
0	Daniel	treatment a	-
1	John	treatment a	12
2	Jane	treatment a	24
3	Daniel	treatment b	42
4	John	treatment b	31
5	Jane	treatment b	27

- Better for reporting vs. better for analysis

- Tidy data makes it easier to fix common data problems

Converting to tidy data

- The data problem we are trying to fix:

  - Columns containing values, instead of variables

- Solution: pd.melt()

Melting

```
In [1]: pd.melt(frame=df, id_vars='name',
   ...:         value_vars=['treatment a', 'treatment b'])
Out[1]:
     name       variable    value
0  Daniel   treatment a       _
1    John   treatment a      12
2    Jane   treatment a      24
3  Daniel   treatment b      42
4    John   treatment b      31
5    Jane   treatment b      27
```

Melting

```
In [2]: pd.melt(frame=df, id_vars='name',
   ...:         value_vars=['treatment a', 'treatment b'],
   ...:         var_name='treatment', value_name='result')
Out[2]:
     name       treatment      result
0  Daniel   treatment a          _
1    John   treatment a         12
2    Jane   treatment a         24
3  Daniel   treatment b         42
4    John   treatment b         31
5    Jane   treatment b         27
```

CLEANING DATA IN PYTHON

Let's practice!

Pivoting data

Pivot: un-melting data

- Opposite of melting

- In melting, we turned columns into rows

- Pivoting: turn unique values into separate columns

- Analysis friendly shape to reporting friendly shape

- Violates tidy data principle: rows contain observations

  - Multiple variables stored in the same column

Pivot: un-melting data

	date	element	value
0	2010-01-30	tmax	27.8
1	2010-01-30	tmin	14.5
2	2010-02-02	tmax	27.3
3	2010-02-02	tmin	14.4
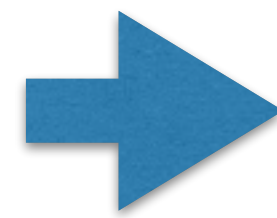
element	tmax	tmin
date		
2010-01-30	27.8	14.5
2010-02-02	27.3	14.4

Pivot

```
In [1]: weather_tidy = weather.pivot(index='date',
   ...:                                columns='element',
   ...:                                values='value')

In [2]: print(weather_tidy)
element      tmax tmin
date
2010-01-30  27.8 14.5
2010-02-02  27.3 14.4
```

Pivot

	date	element	value
0	2010-01-30	tmax	27.8
1	2010-01-30	tmin	14.5
2	2010-02-02	tmax	27.3
3	2010-02-02	tmin	14.4

	date	element	value
0	2010-01-30	tmax	27.8
1	2010-01-30	tmin	14.5
2	2010-02-02	tmax	27.3
3	2010-02-02	tmin	14.4
4	2010-02-02	tmin	16.4

Using pivot when you have duplicate entries

```
In [3]: import numpy as np

In [4]: weather2_tidy = weather.pivot(values='value',
   ...:                               index='date',
   ...:                               columns='element')
Out[4]:
------------------------------------------------------------------
ValueError                              Traceback (most recent call last)
<ipython-input-9-2962bb23f5a3> in <module>()
     1 weather2_tidy = weather2.pivot(values='value',
     2                                index='date',
----> 3                                columns='element')
ValueError: Index contains duplicate entries, cannot reshape
```

Pivot table

- Has a parameter that specifies how to deal with duplicate values

- Example: Can aggregate the duplicate values by taking their average

Pivot table

```
In [5]: weather2_tidy = weather.pivot_table(values='value',
    ...:                                      index='date',
    ...:                                      columns='element',
    ...:                                      aggfunc=np.mean)
Out[5]:
element      tmax tmin
date
2010-01-30   27.8 14.5
2010-02-02   27.3 15.4
```

CLEANING DATA IN PYTHON

Let's practice!

Beyond melt and pivot

Beyond melt and pivot

- Melting and pivoting are basic tools

- Another common problem:

  - Columns contain multiple bits of information

Beyond melt and pivot

	country	year	m014	m1524
0	AD	2000	0	0
1	AE	2000	2	4
2	AF	2000	52	228

Melting and parsing

```
In [1]: pd.melt(frame=tb, id_vars=['country', 'year'])
Out[1]:
   country    year  variable  value
0       AD    2000     m014      0
1       AE    2000     m014      2
2       AF    2000     m014     52
3       AD    2000    m1524      0
4       AE    2000    m1524      4
5       AF    2000    m1524    228
```

- Nothing inherently wrong about original data shape

- Not conducive for analysis

Melting and parsing

```
In [2]: tb_melt['sex'] = tb_melt.variable.str[0]

In [3]: tb_melt
Out[3]:
    country    year variable value  sex
0        AD    2000     m014     0    m
1        AE    2000     m014     2    m
2        AF    2000     m014    52    m
3        AD    2000    m1524     0    m
4        AE    2000    m1524     4    m
5        AF    2000    m1524   228    m
```

CLEANING DATA IN PYTHON

Let's practice!