Introduction to data science

Patrick Shafto

Department of Math and Computer Science

Plan for today

- Tidy data!
- In-class / HW for Tues

Tidy data?

- Wickham, 2014

Tidy data?

- It is often said that 80% of data analysis is spent on the process of cleaning and preparing the data (Dasu and Johnson 2003).
- Data preparation is not just a first step, but must be repeated many over the course of analysis as new problems come to light or new data is collected.
- Despite the amount of time it takes, there has been surprisingly little research on how to clean data well.

Tidy data?

- Part of the challenge is the breadth of activities it encompasses:
  - from outlier checking,
  - to date parsing,
  - to missing value imputation.
- To get a handle on the problem, focus on a small, but important, aspect of data cleaning called data tidying: structuring datasets to facilitate analysis.

Tidy data?

- Two types of data:
  - Tidy data:
    - Variables on the columns, observations on the rows, each observational unit forms a table
  - Messy data:
    - Anything else!

Tidy data?

	treatmenta	treatmentb
John Smith	—	2
Jane Doe	16	11
Mary Johnson	3	1

Table 1: Typical presentation dataset.

	John Smith	Jane Doe	Mary Johnson
treatmenta	—	16	3
treatmentb	2	11	1

Table 2: The same data as in Table 1 but structured differently.

Tidy data?

- A dataset is a collection of values, usually either numbers (if quantitative) or strings (if qualitative).
- Values are organized in two ways.
- Every value belongs to a variable and an observation.
- A variable contains all values that measure the same underlying attribute (like height, temperature, duration) across units.
- An observation contains all values measured on the same unit (like a person, or a day, or a race) across attributes.

Tidy data?

Table 3 reorganises Table 1 to make the values, variables and obserations more clear. The dataset contains 18 values representing three variables and six observations. The variables are:

1. `person`, with three possible values (John, Mary, and Jane).
2. `treatment`, with two possible values (a and b).
3. `result`, with five or six values depending on how you think of the missing value (-, 16, 3, 2, 11, 1).

	treatmenta	treatmentb
John Smith	—	2
Jane Doe	16	11
Mary Johnson	3	1

Table 1: Typical presentation dataset.

Tidy data?

name	trt	result
John Smith	a	—
Jane Doe	a	16
Mary Johnson	a	3
John Smith	b	2
Jane Doe	b	11
Mary Johnson	b	1

Table 3: The same data as in Table 1 but with variables in columns and observations in rows.

- The experimental design tells us more about the structure of the observations.

- In this experiment, every combination of of person and treatment was measured, a completely crossed design.

- The experimental design also determines whether or not missing values can be safely dropped.

- In this experiment, the missing value represents an observation that should have been made, but wasn't, so it's important to keep it.

- Structural missing values, which represent measurements that can't be made (e.g., the count of pregnant males) can be safely removed.

Tidy data?

name	trt	result
John Smith	a	—
Jane Doe	a	16
Mary Johnson	a	3
John Smith	b	2
Jane Doe	b	11
Mary Johnson	b	1

Table 3: The same data as in Table 1 but with variables in columns and observations in rows.

- For a given dataset, it's usually easy to figure out what are observations and what are variables, but it is surprisingly difficult to precisely define variables and observations in general.

- For example, if the columns in the Table 1 were height and weight we would have been happy to call them variables. If the columns were height and width, it would be less clear cut, as we might think of height and width as values of a dimension variable.

- If the columns were home phone and work phone, we could treat these as two variables, but in a fraud detection environment we might want variables phone number and number type because the use of one phone number for multiple people might suggest fraud.

- In a given analysis, there may be multiple levels of observation.
- For example, in a trial of new allergy medication we might have three observational types:
    - demographic data collected from each person (age, sex, race),
    - medical data collected from each person on each day (number of sneezes, redness of eyes), and
    - meterological data collected on each day (temperature, pollen count).

Tidy data?

- Two types of data:
  - Tidy data:
    - Variables on the columns, observations on the rows, each observational unit forms a table
    - Codd's 3rd normal form (1990)

	treatmenta	treatmentb
John Smith	—	2
Jane Doe	16	11
Mary Johnson	3	1

Table 1

name	trt	result
John Smith	a	—
Jane Doe	a	16
Mary Johnson	a	3
John Smith	b	2
Jane Doe	b	11
Mary Johnson	b	1

Table 3

- Tidy data makes it easy for an analyst or a computer to extract needed variables because it provides a standard way of structuring a dataset.

- Compare Table 3 to Table 1: in Table 1 you need to use different strategies to extract different variables. This slows analysis and invites errors. If you consider how many data analysis operations involve all of the values in a variable (every aggregation function), you can see how important it is to extract these values in a simple, standard way.

- While the order of variables and observations does not affect analysis, a good ordering makes it easier to scan the raw values.

- One way of organizing variables is by their role in the analysis: are values fixed by the design of the data collection, or are they measured during the course of the experiment?

    - Fixed variables describe the experimental design and are known in advance. Computer scientists often call fixed variables dimensions, and statisticians usually denote them with subscripts on random variables.

    - Measured variables are what we actually measure in the study.

- Fixed variables should come first, followed by measured variables, each ordered so that related variables are contiguous.

- Rows can then be ordered by the first variable, breaking ties with the second and subsequent (fixed) variables. This is the convention adopted by all tabular displays in this paper.

- The five most common problems with messy datasets, along with their remedies
    - Column headers are values, not variable names
    - Multiple variables are stored in one column.
    - Variables are stored in both rows and columns.
    - Multiple types of observational units are stored in the same table.
    - A single observational unit is stored in multiple tables.
- Surprisingly, most messy datasets, including types of messiness not explicitly described above, can be tidied with a small set of tools: melting, string splitting, and casting.

Melting

- messy = pd.DataFrame({'row' : ['A', 'B', 'C'],
- 'a' : [1, 2, 3],
- 'b' : [4, 5, 6],
- 'c' : [7, 8, 9]})

	a	b	c	row
0	1	4	7	A
1	2	5	8	B
2	3	6	9	C

Melting

- pd.melt(messy, id_vars='row')

	a	b	c	row
0	1	4	7	A
1	2	5	8	B
2	3	6	9	C

	row	variable	value
0	A	a	1
1	B	a	2
2	C	a	3
3	A	b	4
4	B	b	5
5	C	b	6
6	A	c	7
7	B	c	8
8	C	c	9

Melting

- tidy = pd.melt(messy, id_vars='row', var_name='dimension', value_name='length')

	row	variable	value
0	A	a	1
1	B	a	2
2	C	a	3
3	A	b	4
4	B	b	5
5	C	b	6
6	A	c	7
7	B	c	8
8	C	c	9

	row	dimension	length
0	A	a	1
1	B	a	2
2	C	a	3
3	A	b	4
4	B	b	5
5	C	b	6
6	A	c	7
7	B	c	8
8	C	c	9

Melting

- messy1 = tidy.pivot(index='row',columns='dimension',values='length')

	row	dimension	length
0	A	a	1
1	B	a	2
2	C	a	3
3	A	b	4
4	B	b	5
5	C	b	6
6	A	c	7
7	B	c	8
8	C	c	9

dimension	a	b	c
row			
A	1	4	7
B	2	5	8
C	3	6	9

Column headers are values not variable names

- The melt function is key, but it is not always sufficient, as we will see with additional examples used by Wickham.

	religion	<$10k	$10-20k	$20-30k	$30-40k	$40-50k	$50-75k
0	Agnostic	27	34	60	81	76	137
1	Atheist	12	27	37	52	35	70
2	Buddhist	27	21	30	34	33	58
3	Catholic	418	617	732	670	638	1116
4	Don't know/refused	15	14	15	11	10	35
5	Evangelical Prot	575	869	1064	982	881	1486
6	Hindu	1	9	7	9	11	34
7	Historically Black Prot	228	244	236	238	197	223
8	Jehovah's Witness	20	27	24	24	21	30
9	Jewish	19	19	25	25	30	95

Column headers are values not variable names

- tidy = pd.melt(messy, id_vars = ['religion'], var_name='income', value_name='freq')
- tidy.sort_values(by=['religion'], inplace=True)
- tidy.head()

	religion	<$10k	$10-20k	$20-30k	$30-40k	$40-50k	$50-75k
0	Agnostic	27	34	60	81	76	137
1	Atheist	12	27	37	52	35	70
2	Buddhist	27	21	30	34	33	58
3	Catholic	418	617	732	670	638	1116
4	Don't know/refused	15	14	15	11	10	35
5	Evangelical Prot	575	869	1064	982	881	1486
6	Hindu	1	9	7	9	11	34
7	Historically Black Prot	228	244	236	238	197	223
8	Jehovah's Witness	20	27	24	24	21	30
9	Jewish	19	19	25	25	30	95

	religion	income	freq
0	Agnostic	<$10k	27
30	Agnostic	$30-40k	81
40	Agnostic	$40-50k	76
50	Agnostic	$50-75k	137
10	Agnostic	$10-20k	34

Multiple values stored in the same column

	country	year	m014	m1524	m2534	m3544	m4554	m5564	m65	mu	f014
10	AD	2000	0	0	1	0	0	0	0	NaN	NaN
36	AE	2000	2	4	4	6	5	12	10	NaN	3
60	AF	2000	52	228	183	149	129	94	80	NaN	93
87	AG	2000	0	0	0	0	0	0	1	NaN	1
136	AL	2000	2	19	21	14	24	19	16	NaN	3

Multiple values stored in the same column

- molten = pd.melt(messy, id_vars=['country', 'year'], value_name='cases')
- molten.sort_values(by=['year', 'country'], inplace=True)
- molten.head(10)

	country	year	m014	m1524	m2534	m3544	m4554	m5564	m65	mu	f014
10	AD	2000	0	0	1	0	0	0	0	NaN	NaN
36	AE	2000	2	4	4	6	5	12	10	NaN	3
60	AF	2000	52	228	183	149	129	94	80	NaN	93
87	AG	2000	0	0	0	0	0	0	1	NaN	1
136	AL	2000	2	19	21	14	24	19	16	NaN	3

	country	year	variable	cases
0	AD	2000	m014	0
201	AD	2000	m1524	0
402	AD	2000	m2534	1
603	AD	2000	m3544	0
804	AD	2000	m4554	0
1005	AD	2000	m5564	0
1206	AD	2000	m65	0
1407	AD	2000	mu	NaN
1608	AD	2000	f014	NaN
1809	AD	2000	f1524	NaN

Multiple values stored in the same column

```
tidy = molten[molten['variable'] != 'mu'].copy()

def parse_age(s):
    s = s[1:]
    if s == '65':
        return '65+'
    else:
        return s[:-2]+'-'+s[-2:]


tidy['sex'] = tidy['variable'].apply(lambda s: s[:1])

tidy['age'] = tidy['variable'].apply(parse_age)

tidy = tidy[['country', 'year', 'sex', 'age', 'cases']]

tidy.head(10)
```

Multiple values stored in the same column

	country	year	variable	cases
0	AD	2000	m014	0
201	AD	2000	m1524	0
402	AD	2000	m2534	1
603	AD	2000	m3544	0
804	AD	2000	m4554	0
1005	AD	2000	m5564	0
1206	AD	2000	m65	0
1407	AD	2000	mu	NaN
1608	AD	2000	f014	NaN
1809	AD	2000	f1524	NaN

	country	year	sex	age	cases
0	AD	2000	m	0-14	0
201	AD	2000	m	15-24	0
402	AD	2000	m	25-34	1
603	AD	2000	m	35-44	0
804	AD	2000	m	45-54	0
1005	AD	2000	m	55-64	0
1206	AD	2000	m	65+	0
1608	AD	2000	f	0-14	NaN
1809	AD	2000	f	15-24	NaN
2010	AD	2000	f	25-34	NaN

Multiple types in the same table

	year	artist	track	time	date entered	wk1	wk2	wk3
0	2000	2,Pac	Baby Don't Cry	4:22	2000-02-26	87	82	72
1	2000	2Ge+her	The Hardest Part Of ...	3:15	2000-09-02	91	87	92
2	2000	3 Doors Down	Kryptonite	3:53	2000-04-08	81	70	68
3	2000	98^0	Give Me Just One Nig...	3:24	2000-08-19	51	39	34
4	2000	A*Teens	Dancing Queen	3:44	2000-07-08	97	97	96
5	2000	Aaliyah	I Don't Wanna	4:15	2000-01-29	84	62	51
6	2000	Aaliyah	Try Again	4:03	2000-03-18	59	53	38
7	2000	Adams,Yolanda	Open My Heart	5:30	2000-08-26	76	76	74

Multiple types in the same table

```
molten = pd.melt(messy,
                 id_vars=['year','artist','track','time','date entered'],
                 var_name = 'week',
                 value_name = 'rank',
                )

molten.sort_values(by=['date entered','week'], inplace=True)

molten.head()
```

Multiple types in the same table

	year	artist	track	time	date entered	week	rank
5	2000	Aaliyah	I Don't Wanna	4:15	2000-01-29	wk1	84
13	2000	Aaliyah	I Don't Wanna	4:15	2000-01-29	wk2	62
21	2000	Aaliyah	I Don't Wanna	4:15	2000-01-29	wk3	51
0	2000	2,Pac	Baby Don't Cry	4:22	2000-02-26	wk1	87
8	2000	2,Pac	Baby Don't Cry	4:22	2000-02-26	wk2	82

Multiple types in the same table

```
from datetime import datetime, timedelta



def increment_date(row):

    date = datetime.strptime(row['date entered'], "%Y-%m-%d")

    return date + timedelta(7) * (row['week'] - 1)

molten['week'] = molten['week'].apply(lambda s: int(s[2:]))

molten['date'] = molten.apply(increment_date, axis=1)

molten.drop('date entered', axis=1, inplace=True)

molten.head()
```

Multiple types in the same table

	year	artist	track	time	date entered	week	rank
5	2000	Aaliyah	I Don't Wanna	4:15	2000-01-29	wk1	84
13	2000	Aaliyah	I Don't Wanna	4:15	2000-01-29	wk2	62
21	2000	Aaliyah	I Don't Wanna	4:15	2000-01-29	wk3	51
0	2000	2,Pac	Baby Don't Cry	4:22	2000-02-26	wk1	87
8	2000	2,Pac	Baby Don't Cry	4:22	2000-02-26	wk2	82

	year	artist	track	time	week	rank	date
5	2000	Aaliyah	I Don't Wanna	4:15	1	84	2000-01-29
13	2000	Aaliyah	I Don't Wanna	4:15	2	62	2000-02-05
21	2000	Aaliyah	I Don't Wanna	4:15	3	51	2000-02-12
0	2000	2,Pac	Baby Don't Cry	4:22	1	87	2000-02-26
8	2000	2,Pac	Baby Don't Cry	4:22	2	82	2000-03-04

Multiple types in the same table

```
tidy_track = molten[['year','artist','track','time']]\

        .groupby(['year','artist','track'])\

        .first()

tidy_track.reset_index(inplace=True)

tidy_track.rename(columns = {'index':'id'}, inplace=True)

tidy_track
```

Multiple types in the same table

	id	year	artist	track	time
0	0	2000	2,Pac	Baby Don't Cry	4:22
1	1	2000	2Ge+her	The Hardest Part Of ...	3:15
2	2	2000	3 Doors Down	Kryptonite	3:53
3	3	2000	98^0	Give Me Just One Nig...	3:24
4	4	2000	A*Teens	Dancing Queen	3:44
5	5	2000	Aaliyah	I Don't Wanna	4:15
6	6	2000	Aaliyah	Try Again	4:03
7	7	2000	Adams,Yolanda	Open My Heart	5:30

Multiple types in the same table

```
tidy_rank = pd.merge(molten, tidy_track, on='track')

tidy_rank = tidy_rank[['id', 'date', 'rank']]

tidy_rank.head()
```

Multiple types in the same table

	id	date	rank
0	5	2000-01-29	84
1	5	2000-02-05	62
2	5	2000-02-12	51
3	0	2000-02-26	87
4	0	2000-03-04	82

HW for monday

- Replicate this analysis in your own jupiter notebook
- http://tomaugspurger.github.io/modern-5-tidy.html
- Add comments explaining what exactly the code is doing
- Stop at “Mini Project: Home Court Advantage?”

In class / homework

- Do a comparable set of analyses as the previous one on this data:
- http://www.baseball-reference.com/teams/BOS/2016-schedule-scores.shtml#team_schedule::none
- your analyses can stop at the header “stack/unstack” (just the first three analyses)