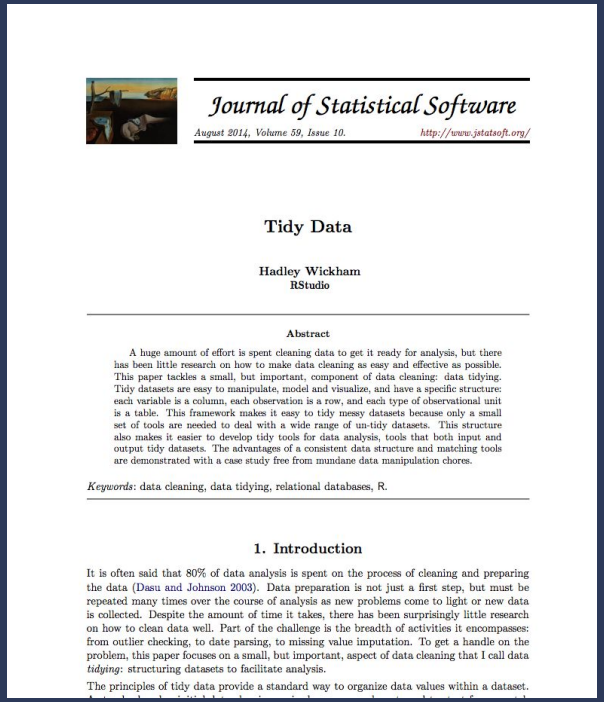
Aviv Rotman | Taboola | PyCon 2018

Tidy Data in Python

Where this all started



Journal of Statistical Software

August 2014, Volume 59, Issue 10. http://www.jstatsoft.org/

Tidy Data

Hadley Wickham
RStudio

Abstract

A huge amount of effort is spent cleaning data to get it ready for analysis, but there has been little research on how to make data cleaning as easy and effective as possible. This paper tackles a small, but important, component of data cleaning: data tidying. Tidy datasets are easy to manipulate, model and visualize, and have a specific structure: each variable is a column, each observation is a row, and each type of observational unit is a table. This framework makes it easy to tidy messy datasets because only a small set of tools are needed to deal with a wide range of un-tidy datasets. This structure also makes it easier to develop tidy tools for data analysis, tools that both input and output tidy datasets. The advantages of a consistent data structure and matching tools are demonstrated with a case study free from mundane data manipulation chores.

Keywords: data cleaning, data tidying, relational databases, R.

1. Introduction

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 times 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. 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, this paper focuses on a small, but important, aspect of data cleaning that I call data *tidying*: structuring datasets to facilitate analysis.

The principles of tidy data provide a standard way to organize data values within a dataset.

Why Should I Tidy Data?

80% of time spent on cleaning Data

Familiar data structure is easy to view and manipulate

Standardization makes tidying a familiar task

Makes it possible to use and design tools for exploratory data analysis

What is Tidy Data?

This is not a Tidy dataset:

	Treatment A	Treatment B
John Smith	-	2
Jane Doe	16	11
Mary Johnson	3	1

What is Tidy Data?

Tidy data is based on 3 basic rules

- Each variable forms a column
- Each observation forms a row
- Each type of observational unit forms a table

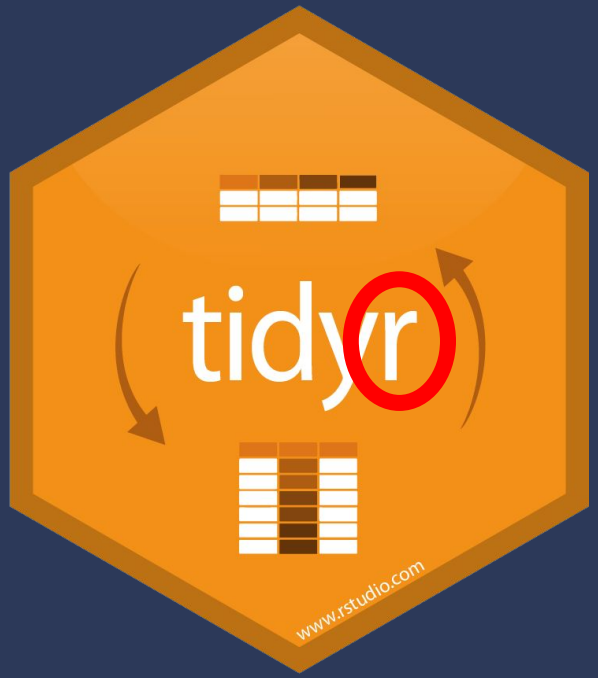
What is Tidy Data?

This is a Tidy dataset:

Name	Treatment	Result
John Smith	a	-
Jane Doe	a	16
Mary Johnson	a	3
John Smith	b	2
Jane Doe	b	11
Mary Johnson	b	1

How Should I Tidy Data

Someone has already done the job for us



The Five Sins of Messy Data

The Five Deadly Sins of Messy Data

> “Happy families are all alike; every unhappy family is unhappy in its own way”
>
> Leo Tolstoy

The Five Deadly Sins of Messy Data

1. **Column headers are values, not variable names**

	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	Dont 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	Jehovahs Witness	20	27	24	24	21	30
9	Jewish	19	19	25	25	30	95

Relationship between income and religion in the US (Pew Research Center)

The Five Deadly Sins of Messy Data

1. **Column headers are values, not variable names**
   Get the most common income in each religion

	religion	mode_income
0	Agnostic	$50-75k
1	Atheist	$50-75k
2	Buddhist	$50-75k
3	Catholic	$50-75k
4	Dont know/refused	$50-75k
5	Evangelical Prot	$50-75k
6	Hindu	$50-75k
7	Historically Black Prot	$10-20k
8	Jehovahs Witness	$50-75k
9	Jewish	$50-75k

The Five Deadly Sins of Messy Data

1. **Column headers are values, not variable names**
   Get the most common revenue in each religion

```
max_freq = df.iloc[:,1:].max(axis=1)
incomes = []
for i, value in enumerate(max_freq):
    current_series = df.iloc[i,:]
    index = current_series[current_series == value].index[0]
    incomes.append(index)
mode_series = pd.Series(incomes)
new_df = df.iloc[:,0:1]
new_df['mode_income'] = mode_series
new_df
```

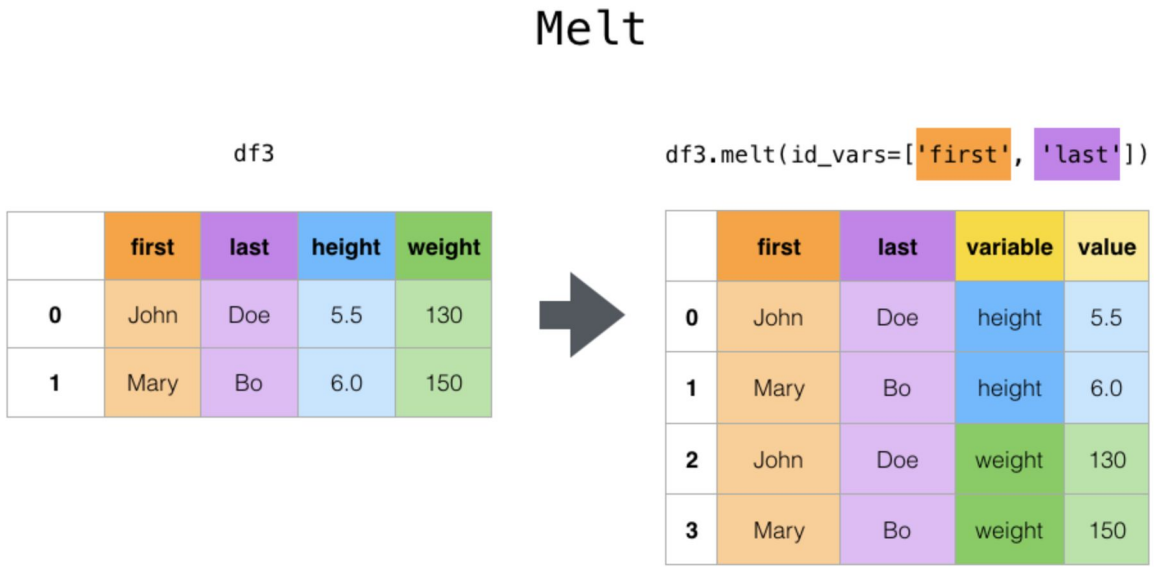
The Five Deadly Sins of Messy Data

1. **Column headers are values, not variable names**

	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	Dont 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	Jehovahs Witness	20	27	24	24	21	30
9	Jewish	19	19	25	25	30	95

Relationship between income and religion in the US (Pew Research Center)

Melt



	first	last	height	weight
0	John	Doe	5.5	130
1	Mary	Bo	6.0	150

	first	last	variable	value
0	John	Doe	height	5.5
1	Mary	Bo	height	6.0
2	John	Doe	weight	130
3	Mary	Bo	weight	150

The Five Deadly Sins of Messy Data

1. **Column headers are values, not variable names**

```
pd.melt(df,["religion"], var_name="income", value_name="freq")
```

The Five Deadly Sins of Messy Data

1. **Column headers are values, not variable names**

	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
20	Agnostic	$20-30k	60
41	Atheist	$40-50k	35
21	Atheist	$20-30k	37
11	Atheist	$10-20k	27
31	Atheist	$30-40k	52

The Five Deadly Sins of Messy Data

1. **Column headers are values, not variable names**
   Get the most common revenue in each religion

	religion	mode_income
0	Agnostic	$50-75k
1	Atheist	$50-75k
2	Buddhist	$50-75k
3	Catholic	$50-75k
4	Dont know/refused	$50-75k
5	Evangelical Prot	$50-75k
6	Hindu	$50-75k
7	Historically Black Prot	$10-20k
8	Jehovahs Witness	$50-75k
9	Jewish	$50-75k

The Five Deadly Sins of Messy Data

1. **Column headers are values, not variable names**
   Get the most common revenue in each religion

```
ids = formatted_df.groupby('religion')['freq'].transform(max) == formatted_df['freq']
formatted_df[ids]
```

The Five Deadly Sins of Messy Data

1. **Column headers are values, not variable names**

	year	artist.inverted	track	time	genre	date.entered	date.peaked	x1st.week	x2nd.week	x3rd.week	...	x67th.week
0	2000	Destiny's Child	Independent Women Part I	3:38	Rock	2000-09-23	2000-11-18	78	63.0	49.0	...	NaN
1	2000	Santana	Maria, Maria	4:18	Rock	2000-02-12	2000-04-08	15	8.0	6.0	...	NaN
2	2000	Savage Garden	I Knew I Loved You	4:07	Rock	1999-10-23	2000-01-29	71	48.0	43.0	...	NaN
3	2000	Madonna	Music	3:45	Rock	2000-08-12	2000-09-16	41	23.0	18.0	...	NaN
4	2000	Aguilera, Christina	Come On Over Baby (All I Want Is You)	3:38	Rock	2000-08-05	2000-10-14	57	47.0	45.0	...	NaN
5	2000	Janet	Doesn't Really Matter	4:17	Rock	2000-06-17	2000-08-26	59	52.0	43.0	...	NaN
6	2000	Destiny's Child	Say My Name	4:31	Rock	1999-12-25	2000-03-18	83	83.0	44.0	...	NaN
7	2000	Iglesias, Enrique	Be With You	3:36	Latin	2000-04-01	2000-06-24	63	45.0	34.0	...	NaN
8	2000	Sisqo	Incomplete	3:52	Rock	2000-06-24	2000-08-12	77	66.0	61.0	...	NaN
9	2000	Lonestar	Amazed	4:25	Country	1999-06-05	2000-03-04	81	54.0	44.0	...	NaN

Date and rankings of songs in the Billboard top 100 chart

The Five Deadly Sins of Messy Data

1. **Column headers are values, not variable names**

```
id_vars = ["year","artist.inverted","track","time","genre","date.entered","date.peaked"]
df = pd.melt(frame=df,id_vars=id_vars, var_name="week", value_name="rank")
```

The Five Deadly Sins of Messy Data

1. **Column headers are values, not variable names**

	year	artist.inverted	track	time	genre	week	rank	date
246	2000	2 Pac	Baby Don't Cry (Keep Ya Head Up II)	4:22	Rap	1	87	2000-02-26
563	2000	2 Pac	Baby Don't Cry (Keep Ya Head Up II)	4:22	Rap	2	82	2000-03-04
880	2000	2 Pac	Baby Don't Cry (Keep Ya Head Up II)	4:22	Rap	3	72	2000-03-11
1197	2000	2 Pac	Baby Don't Cry (Keep Ya Head Up II)	4:22	Rap	4	77	2000-03-18
1514	2000	2 Pac	Baby Don't Cry (Keep Ya Head Up II)	4:22	Rap	5	87	2000-03-25
1831	2000	2 Pac	Baby Don't Cry (Keep Ya Head Up II)	4:22	Rap	6	94	2000-04-01
2148	2000	2 Pac	Baby Don't Cry (Keep Ya Head Up II)	4:22	Rap	7	99	2000-04-08
287	2000	2Ge+her	The Hardest Part Of Breaking Up (Is Getting Ba...	3:15	R&B	1	91	2000-09-02
604	2000	2Ge+her	The Hardest Part Of Breaking Up (Is Getting Ba...	3:15	R&B	2	87	2000-09-09
921	2000	2Ge+her	The Hardest Part Of Breaking Up (Is Getting Ba...	3:15	R&B	3	92	2000-09-16

The Five Deadly Sins of Messy Data

2. **Multiple types of observational units are stored in the same table**

	year	artist.inverted	track	time	genre	week	rank	date
246	2000	2 Pac	Baby Don't Cry (Keep Ya Head Up II)	4:22	Rap	1	87	2000-02-26
563	2000	2 Pac	Baby Don't Cry (Keep Ya Head Up II)	4:22	Rap	2	82	2000-03-04
880	2000	2 Pac	Baby Don't Cry (Keep Ya Head Up II)	4:22	Rap	3	72	2000-03-11
1197	2000	2 Pac	Baby Don't Cry (Keep Ya Head Up II)	4:22	Rap	4	77	2000-03-18
1514	2000	2 Pac	Baby Don't Cry (Keep Ya Head Up II)	4:22	Rap	5	87	2000-03-25
1831	2000	2 Pac	Baby Don't Cry (Keep Ya Head Up II)	4:22	Rap	6	94	2000-04-01
2148	2000	2 Pac	Baby Don't Cry (Keep Ya Head Up II)	4:22	Rap	7	99	2000-04-08
287	2000	2Ge+her	The Hardest Part Of Breaking Up (Is Getting Ba...	3:15	R&B	1	91	2000-09-02
604	2000	2Ge+her	The Hardest Part Of Breaking Up (Is Getting Ba...	3:15	R&B	2	87	2000-09-09
921	2000	2Ge+her	The Hardest Part Of Breaking Up (Is Getting Ba...	3:15	R&B	3	92	2000-09-16

The Five Deadly Sins of Messy Data

2. **Multiple types of observational units are stored in the same table**

```
songs_cols = ["year", "artist.inverted", "track", "time", "genre"]
songs = billboard[songs_cols].drop_duplicates()
songs = songs.reset_index(drop=True)
songs["song_id"] = songs.index
```

The Five Deadly Sins of Messy Data

2. **Multiple types of observational units are stored in the same table.**

	year	artist.inverted	track	time	genre	song_id
0	2000	2 Pac	Baby Don't Cry (Keep Ya Head Up II)	4:22	Rap	0
1	2000	2Ge+her	The Hardest Part Of Breaking Up (Is Getting Ba...	3:15	R&B	1
2	2000	3 Doors Down	Kryptonite	3:53	Rock	2
3	2000	3 Doors Down	Loser	4:24	Rock	3
4	2000	504 Boyz	Wobble Wobble	3:35	Rap	4
5	2000	98°	Give Me Just One Night (Una Noche)	3:24	Rock	5
6	2000	A*Teens	Dancing Queen	3:44	Pop	6
7	2000	Aaliyah	I Don't Wanna	4:15	Rock	7
8	2000	Aaliyah	Try Again	4:03	Rock	8
9	2000	Adams, Yolanda	Open My Heart	5:30	Gospel	9

The Five Deadly Sins of Messy Data

2. **Multiple types of observational units are stored in the same table.**

```
ranks = pd.merge(billboard, songs, on=["year","artist.inverted", "track", "time", "genre"])
ranks = ranks[["song_id", "date","rank"]]
```

	song_id	date	rank
0	0	2000-02-26	87
1	0	2000-03-04	82
2	0	2000-03-11	72
3	0	2000-03-18	77
4	0	2000-03-25	87
5	0	2000-04-01	94

The Five Deadly Sins of Messy Data

3. Multiple variables are stored in one column

	country	year	m014	m1524	m2534	m3544	m4554	m5564	m65	mu	f014
0	AD	2000	0.0	0.0	1.0	0.0	0	0	0.0	NaN	NaN
1	AE	2000	2.0	4.0	4.0	6.0	5	12	10.0	NaN	3.0
2	AF	2000	52.0	228.0	183.0	149.0	129	94	80.0	NaN	93.0
3	AG	2000	0.0	0.0	0.0	0.0	0	0	1.0	NaN	1.0
4	AL	2000	2.0	19.0	21.0	14.0	24	19	16.0	NaN	3.0
5	AM	2000	2.0	152.0	130.0	131.0	63	26	21.0	NaN	1.0
6	AN	2000	0.0	0.0	1.0	2.0	0	0	0.0	NaN	0.0
7	AO	2000	186.0	999.0	1003.0	912.0	482	312	194.0	NaN	247.0
8	AR	2000	97.0	278.0	594.0	402.0	419	368	330.0	NaN	121.0
9	AS	2000	NaN	NaN	NaN	NaN	1	1	NaN	NaN	NaN

Counts of tuberculosis cases by country, year, and demographic (World Health Organisation)

The Five Deadly Sins of Messy Data

3. **Multiple variables are stored in one column**

```
df = pd.melt(df, id_vars=["country","year"], value_name="cases", var_name="sex_and_age"

# Extract Sex, Age lower bound and Age upper bound group
tmp_df = df["sex_and_age"].str.extract("(\D)(\d+)(\d{2})", expand=False)

# Name columns
tmp_df.columns = ["sex", "age_lower", "age_upper"]

# Create `age`column based on `age_lower` and `age_upper`
tmp_df["age"] = tmp_df["age_lower"] + "-" + tmp_df["age_upper"]

# Merge
df = pd.concat([df, tmp_df], axis=1)
```

The Five Deadly Sins of Messy Data

3. **Multiple variables are stored in one column**

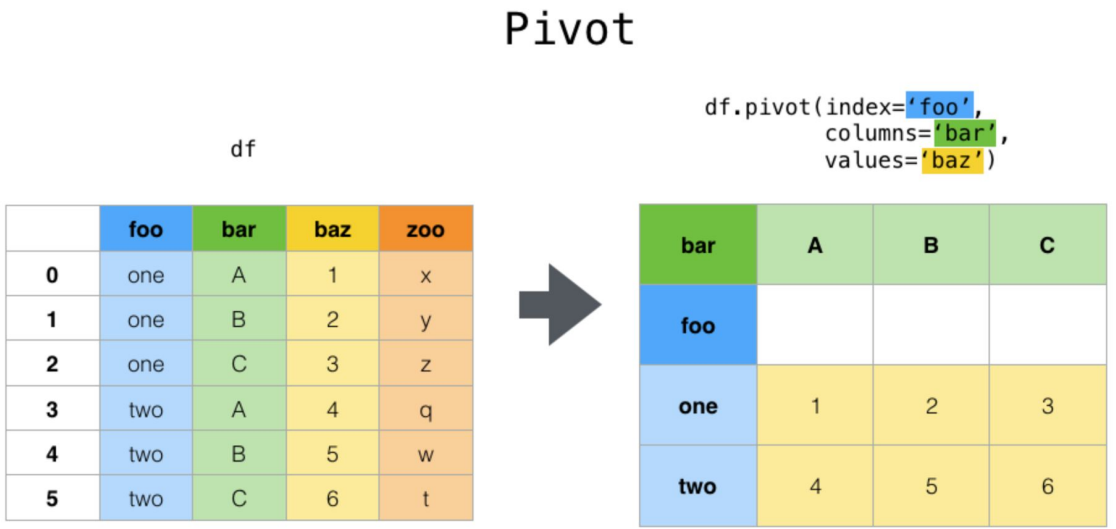
	country	year	cases	sex	age
0	AD	2000	0.0	m	0-14
10	AD	2000	0.0	m	15-24
20	AD	2000	1.0	m	25-34
30	AD	2000	0.0	m	35-44
40	AD	2000	0.0	m	45-54
50	AD	2000	0.0	m	55-64
81	AE	2000	3.0	f	0-14
1	AE	2000	2.0	m	0-14
11	AE	2000	4.0	m	15-24
21	AE	2000	4.0	m	25-34

The Five Deadly Sins of Messy Data

4. **Variables are stored in both rows and columns**

	id	year	month	element	d1	d2	d3	d4	d5	d6	d7	d8
0	MX17004	2010	1	tmax	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	MX17004	2010	1	tmin	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	MX17004	2010	2	tmax	NaN	27.3	24.1	NaN	NaN	NaN	NaN	NaN
3	MX17004	2010	2	tmin	NaN	14.4	14.4	NaN	NaN	NaN	NaN	NaN
4	MX17004	2010	3	tmax	NaN	NaN	NaN	NaN	32.1	NaN	NaN	NaN
5	MX17004	2010	3	tmin	NaN	NaN	NaN	NaN	14.2	NaN	NaN	NaN
6	MX17004	2010	4	tmax	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
7	MX17004	2010	4	tmin	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
8	MX17004	2010	5	tmax	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
9	MX17004	2010	5	tmin	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

Pivot



	foo	bar	baz	zoo
0	one	A	1	x
1	one	B	2	y
2	one	C	3	z
3	two	A	4	q
4	two	B	5	w
5	two	C	6	t

bar	A	B	C
foo			
one	1	2	3
two	4	5	6

The Five Deadly Sins of Messy Data

4. **Variables are stored in both rows and columns**

	id	element	value	date
12	MX17004	tmax	27.3	2010-02-02
13	MX17004	tmin	14.4	2010-02-02
22	MX17004	tmax	24.1	2010-02-03
23	MX17004	tmin	14.4	2010-02-03
44	MX17004	tmax	32.1	2010-03-05
45	MX17004	tmin	14.2	2010-03-05

Daily weather data for one weather station in Mexico for five months in 2010 (Global Historical Climatology Network)

The Five Deadly Sins of Messy Data

4. **Variables are stored in both rows and columns**

```
# Unmelting column "element"
df = df.pivot_table(index=["id","date"], columns="element", values="value")
df.reset_index(drop=False, inplace=True)
```

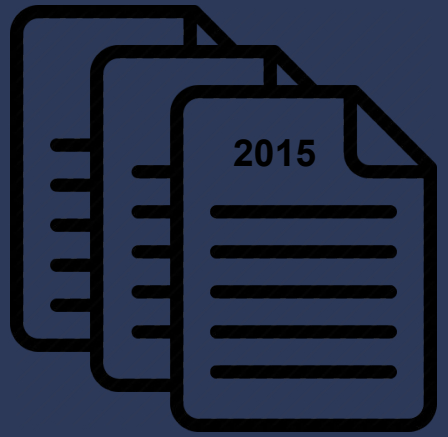
id	date	tmax	tmin
MX17004	2010-02-02	27.3	14.4
MX17004	2010-02-03	24.1	14.4
MX17004	2010-03-05	32.1	14.2

The Five Deadly Sins of Messy Data

5. **A single observational unit is stored in multiple tables**

	rank	name	frequency	sex
0	1	Noah	863	Male
1	2	Liam	709	Male
2	3	Alexander	703	Male
3	4	Jacob	650	Male
4	5	William	618	Male

Illinois Male Baby Names Frequency for the year 2014/2015



The Five Deadly Sins of Messy Data

	rank	name	frequency	sex	year
0	1	Noah	863	Male	2015
1	2	Liam	709	Male	2015
2	3	Alexander	703	Male	2015
3	4	Jacob	650	Male	2015
4	5	William	618	Male	2015

The Five Deadly Sins of Messy Data

1. **Column headers are values, not variable names**
2. **Multiple types of observational units are stored in the same table**
3. **Multiple variables are stored in one column**
4. **Variables are stored in both rows and columns**
5. **A single observational unit is stored in multiple tables**

Why Tidy Data is Awesome

Manipulating Data

- Filter
- Select
- Mutate
- Summarise

Why Tidy Data is Awesome

Manipulating Data

```
df.groupby('year')['frequency'].agg(['mean', 'sum', 'count'])
```

	mean	sum	count
year			
2014	333.693069	33703	101
2015	329.290000	32929	100

Why Tidy Data is Awesome

Manipulating Data

```
df[df['frequency'] > 400].groupby('name')['frequency'].agg(["mean","count"]).sort_values('mean', ascending=False)
```

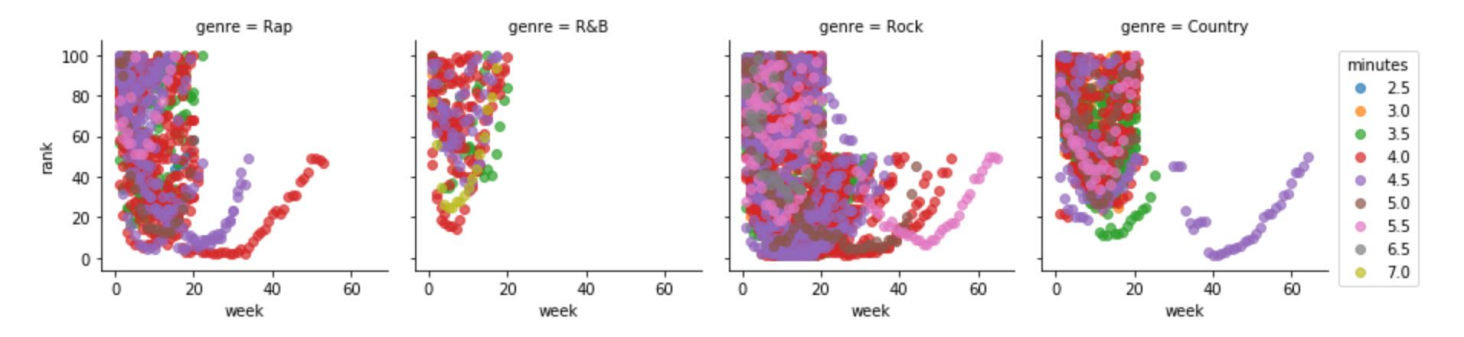
	mean	count
name		
Noah	850.0	2
Alexander	725.0	2
Liam	689.5	2
William	652.5	2
Jacob	652.0	2
Michael	648.5	2
Benjamin	632.5	2
Mason	599.0	2
Daniel	597.0	2
Logan	580.5	2

Why Tidy Data is Awesome

Visualizing Data

- Allows using Grammar of Graphics

```
g = sns.FacetGrid(billboard_genre, col="genre", hue="minutes")
g.map(plt.scatter, "week", "rank", alpha=.7)
g.add_legend();
```



Bibliography

Tidy Data, Hadley Wickham, Journal of Statistical Software 2014

Tidy Data in Python, Jean-Nicholas Hould

Visualizing Pandas' Pivoting and Reshaping Functions, Jay Alammar


Thank You