

Data Tidying and Cleaning

Preparing data
for knowledge extraction

Yordan Darakchiev

Technical Trainer

iordan93@gmail.com





sli.do

#DataScience

Table of Contents

- Messy and tidy data
 - Tidying up messy data
- Operations on datasets
- Cleaning data
 - Validation
 - Transformation
 - Error correction
 - Features
- Data tidying and cleaning as a process

Data Tidying

Arranging data
in a meaningful manner

Tidy Data

- Most important rules when creating (or using) datasets
 - Columns – attributes (features, variables)
 - Rows – observations
 - Cells – values (one observation of one feature)
 - All other data is called **messy data**
- Empirical rule for testing whether a dataset is tidy
 - Adding one more observation should create one new row
 - No new columns
 - No multiple rows
 - No partial rows, no changes to other rows
- pandas allows us to read, tidy up and transform datasets
 - Data modelling requires a tidy and clean dataset in order to work well (garbage in – garbage out)

Messy Data

What we want

country	year	cases	population
Afghanistan	2000	2666	20095360
Brazil	1999	31737	17200362
Brazil	2000	80488	17400898
China	1999	212258	127201272
China	2000	212266	128002583

variables

country	year	cases	population
Afghanistan	2000	2666	20095360
Brazil	1999	31737	17200362
Brazil	2000	80488	17400898
China	1999	212258	127201272
China	2000	212266	128002583

observations

country	year	cases	population
Afghanistan	2000	2666	20095360
Afghanistan	2000	2666	20095360
Brazil	1999	31737	17200362
Brazil	1999	31737	17200362
Brazil	2000	80488	17400898
China	1999	212258	127201272
China	1999	212258	127201272
China	2000	212266	128002583
China	2000	212266	128002583

values

What we get instead

<data endTime="2014-04-23 23:27:29" generated="2014-04-24 00:08:14" handle="46608654" radarStatus="online" startTime="2014-04-23 23:27:00" type="data"> noise,20140423232700,4,4,COMMUNITY_NOISE,48.5 noise,20140423232700,60,60,COMMUNITY_NOISE,54.3 noise,20140423232700,3,3,COMMUNITY_NOISE,50.2 noise,20140423232700,6,6,COMMUNITY_NOISE,46.9 noise,20140423232700,61,61,COMMUNITY_NOISE,44.5 noise,20140423232701,4,4,COMMUNITY_NOISE,49.4 noise,20140423232701,60,60,COMMUNITY_NOISE,52.3 noise,20140423232701,3,3,COMMUNITY_NOISE,49.3 noise,20140423232701,6,6,COMMUNITY_NOISE,47.2 noise,20140423232701,61,61,COMMUNITY_NOISE,44.5 noise,20140423232701,2,2,COMMUNITY_NOISE,34.2 noise,20140423232702,4,4,COMMUNITY_NOISE,48.8 noise,20140423232702,60,60,COMMUNITY_NOISE,50.7 noise,20140423232702,3,3,COMMUNITY_NOISE,49.3 noise,20140423232702,6,6,COMMUNITY_NOISE,46.5 noise,20140423232702,61,61,COMMUNITY_NOISE,44.7 noise,20140423232702,2,2,COMMUNITY_NOISE,34.6 noise,20140423232703,4,4,COMMUNITY_NOISE,49.3 noise,20140423232703,60,60,COMMUNITY_NOISE,50.4 noise,20140423232703,3,3,COMMUNITY_NOISE,48.9 noise,20140423232703,6,6,COMMUNITY_NOISE,48.8 noise,20140423232703,61,61,COMMUNITY_NOISE,45.6 noise,20140423232703,2,2,COMMUNITY_NOISE,37.8 radar,20140423232704,11272847,A,MEL,A320,JST951,J,2296.2,33267.9,110,67,VHVL,1111,CNS,MEL,,ScheduledFlight,242 radar,20140423232704,11272848,A,MEL,A333,QFA776,J,-11074.4,8118.1,933,125,VHQP,3253,PER,MEL,,ScheduledFlight,1065 noise,20140423232704,4,4,COMMUNITY_NOISE,49.3 noise,20140423232704,60,60,COMMUNITY_NOISE,51.3 noise,20140423232704,3,3,COMMUNITY_NOISE,49 noise,20140423232704,6,6,COMMUNITY_NOISE,49.9 noise,20140423232704,61,61,COMMUNITY_NOISE,45.4 noise,20140423232704,2,2,COMMUNITY_NOISE,35.4 noise,20140423232705,4,4,COMMUNITY_NOISE,48.8 noise,20140423232705,60,60,COMMUNITY_NOISE,52.2 noise,20140423232705,3,3,COMMUNITY_NOISE,49.4 noise,20140423232705,6,6,COMMUNITY_NOISE,49.2 noise,20140423232705,61,61,COMMUNITY_NOISE,47.7 noise,20140423232706,3,3,COMMUNITY_NOISE,48.4 noise,20140423232707,4,4,COMMUNITY_NOISE,48.7 noise,20140423232707,61,61,COMMUNITY_NOISE,46.6 noise,20140423232708,3,3,COMMUNITY_NOISE,49.3 radar,20140423232709,11272847,A,MEL,A320,JST951,J,2296.2,33267.9,110,67,VHQP,3253,PER,MEL,,ScheduledFlight,1065 noise,20140423232709,3,3,COMMUNITY_NOISE,50 noise,20140423232710,4,4,COMMUNITY_NOISE,49.3 noise,20140423232710,61,61,COMMUNITY_NOISE,47 noise,20140423232711,3,3,COMMUNITY_NOISE,49.3 </data>

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	aa	ab	ac	ad	ae	af	ag	ah	ai	aj	ak	al	am	an	ao	ap	aq	ar	as	at	au	av	aw	ax	ay	az	ba	bb	bc	bd	be	bf	bg	bh	bi	bj	bk	bl	bm	bn	bo	bp	bq	br	bs	bt	bu	bv	bw	bx	by	bz	ca	cb	cc	cd	ce	cf	cg	ch	ci	cj	ck	cl	cm	cn	co	cp	cq	cr	cs	ct	cu	cv	cw	cx	cy	cz	da	db	dc	dd	de	df	dg	dh	di	dj	dk	dl	dm	dn	do	dp	dq	dr	ds	dt	du	dv	dw	dx	dy	dz	ea	eb	ec	ed	ee	ef	eg	eh	ei	ej	ek	el	em	en	eo	ep	eq	er	es	et	eu	ev	ew	ex	ey	ez	fa	fb	fc	fd	fe	ff	fg	fh	fi	fj	fk	fl	fm	fn	fo	fp	fq	fr	fs	ft	fu	fv	fw	fx	fy	fz	ga	gb	gc	gd	ge	gf	gg	gh	gi	gj	gk	gl	gm	gn	go	gp	gq	gr	gs	gt	gu	gv	gw	gx	gy	gz	ha	hb	hc	hd	he	hf	hg	hh	hi	hj	hk	hl	hm	hn	ho	hp	hq	hr	hs	ht	hu	hv	hw	hx	hy	hz	ia	ib	ic	id	ie	if	ig	ih	ii	ij	ik	il	im	in	io	ip	iq	ir	is	it	iu	iv	iw	ix	iy	iz	ja	jb	jc	jd	je	jf	jg	jh	ji	jj	jk	jl	jm	jn	jo	jp	jq	jr	js	jt	ju	jv	jw	jx	ky	kz	la	lb	lc	ld	le	lf	lg	lh	li	lj	lk	ll	lm	ln	lo	lp	lq	lr	ls	lt	lu	lv	lw	lx	ly	lz	ma	mb	mc	md	me	mf	mg	mh	mi	mj	mk	ml	mm	mn	mo	mp	mq	mr	ms	mt	mu	mv	mw	mx	my	mz	na	nb	nc	nd	ne	nf	ng	nh	ni	nj	nk	nl	nm	nn	no	np	nq	nr	ns	nt	nu	nv	nw	nx	ny	nz	oa	ob	oc	od	oe	of	og	oh	oi	oj	ok	ol	om	on	oo	op	oq	or	os	ot	ou	ov	ow	ox	oy	oz	pa	pb	pc	pd	pe	pf	pg	ph	pi	pj	pk	pl	pm	pn	po	pp	pq	pr	ps	pt	pu	pv	pw	px	py	pz	qa	qb	qc	qd	qe	qf	qg	qh	qi	qj	qk	ql	qm	qn	qo	qp	qq	qr	qs	qt	qu	qv	qw	qx	qy	qz	ra	rb	rc	rd	re	rf	rg	rh	ri	rj	rk	rl	rm	rn	ro	rp	rq	rr	rs	rt	ru	rv	rw	rx	ry	rz	sa	sb	sc	sd	se	sf	sg	sh	si	sj	sk	sl	sm	sn	so	sp	sq	sr	ss	st	su	sv	sw	sx	sy	sz	ta	tb	tc	td	te	tf	tg	th	ti	tj	tk	tl	tm	tn	to	tp	tq	tr	ts	tt	tu	tv	tw	tx	ty	tz	ua	ub	uc	ud	ue	uf	ug	uh	ui	uj	uk	ul	um	un	uo	up	uq	ur	us	ut	uu	uv	uw	ux	uy	uz	va	vb	vc	vd	ve	vf	vg	vh	vi	vj	vk	vl	vm	vn	vo	vp	vq	vr	vs	vt	vu	vv	vw	wx	wy	wz	xa	xb	yc	yd	ze	zf	zg	zh	zi	zj	zk	zl	zm	zn	zo	zp	zq	zr	zs	zt	zu	zv	zw	zx	zy	zz
Company I Website U First Name Last Name Email Street Add City Postal / Zi State / Pro Cou	Vandelay I http://van George Costanza g.costance3661 Hinkl Binghamtc 13901 New York USA	Thathertoi http://king Heather Hall heather.ha 1683 Jasper Edmonton T5J 3N6 Alberta Can	Western E http://ww Erich Thomson eric.t@we 745 Stroof Atlanta 30305 Georgia USA	Mammoth http://tiny Michael Blake michael.bl 4984 Clair Morgan 76671 Texas USA	Powell Mc http://ww Harbert Powell harbert@ 1125 Cros Michigan 48607 Detroit USA	Urban Son http://ww Garcia Est Orosco garcia@ur 16221 E 4C Denver 80239 Colorado USA	Wayne Enl http://way Bruce Wayne b.wayne@ 3881 Mich Chicago 60631 Illinois USA	Kramerica http://www Cosmo Kramer ckramer@ 22 Marion New York 66059 New York Ame	Bluth Com http://thel Bluth George george@b 36 Southw Fairfield 60042 New Jerse USA	Flowers By http://ww Elliott Irene irene@irer 2197 Tato Buffalo Gr 60089 Illinois USA	Spacely Sp http://spa Spacely Cosmo cosmo@sr 89 Davids New York 66412 New York USA	Dunder Mi http://ww Dudner Bob bob@dunc 25 Spring C Waterloo N3E 0R4 Ontario Can	Sixty Secoi http://www Brown Darren darren@si 35 Brentw San Franci 94210 California USA	Charles To http://ww Angel Charles c.angel@ti 20 Dora Cr The Bay 94212 California USA	Spade and http://www Archer Christoph c.a@spade 68 Cambri Waterloo N2N 2J8 Ontario Canada	Atlantic N http://www Thomas Won won@atla 94 Walters Vancouver K2C 4CJ BC Canada	Tessier-Asl http://ww Tessier Marie marie@te 2 Bourke C Melbourne 3418 Australia	Southern C http://www Van Houte Kirk kirk@allie 67 Gadd A Springfield 63012 Illinois USA	Initech http://www Charles Deborah d.charles@ 35 Quintin Waterloo N3E 0R4 Ontario Canada	McMahon http://www Tate Charles tate@man 87 Wright Palo Alto 94304 California USA	Widget Inc http://www Wilbert Gregg gregg@wic 3094 Boul Quebec G1R 1B8 Quebec Canada	Acme Cor http://www Harris Mike mike@acn 59 South S London N1 9EW UK	LexCorp http://www Luthor Lex lex@lex.cc 29 Fergus Toronto N2N 2K2 Ontario Canada																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																														

BioLegato: bldna - DNA sequence tasks

File Edit Documentat DNARNA RNA Struc Similarity Database Pattern Alignment Primers Help

KJ939334

MM 001112307

KJ551546

KJ551545

KJ551544

KJ551543

KJ551542

KJ551541

KJ551540

KJ551539

KJ551538

KJ551537

agcagatatcattacagtggaactactaatgaagccatggaggaagaatcacatagctggcttatgcttcccttcttggtaagcagaggtttttttgacatataaaatgagtggtgactatcacattgaagaagccacaaagcctagacaatctttataagtagcacaaatccaccagtagcaaaagccagtagcgcgaagcagagtttagtgcgtccacagcccgccggttcacaatgtggaggaagtagattcacatccccctttaccccttccacacattgaagtggtcgtatcacaaagtttctccctgcgctgtacaaicccagataaagtgtacataaactcaggggtcggttttagcagataaacaagaaggaagtaggaggtatcacaaagctcttaggtcttaaatctgtacataaactctcttgcgtttttcaggaatactcaggaattctccttcaggggtgtgtggctccacagcaccctgacttccaaagctgtagtggaaccgatccatggaggttcacgacaggtctctgcggcggtcttccatccgtctaccatagaaccagctcgagagccgaaggtttcacatttacatataaatctcttattctttcacaaagcaacggcagcagctccatcacaccacaaacctgatacctcataccagcagtaatcgatgctgatggaggtctacgcgaagccacagctcaggtttcacattcacaaactctcttcccttcacaaaggaacgcaggt

Tidy and Messy Data

- A very good [paper](#) on tidy data
- Example: several datasets
 - Same information, different ease of use

	country	year	cases	population
1	Afghanistan	1999	745	19987071
2	Afghanistan	2000	2666	20595360
3	Brazil	1999	37737	172006362
4	Brazil	2000	80488	174504898
5	China	1999	212258	1272915272
6	China	2000	213766	1280428583

Tidy dataset

	country	year	rate
1	Afghanistan	1999	745/19987071
2	Afghanistan	2000	2666/20595360
3	Brazil	1999	37737/172006362
4	Brazil	2000	80488/174504898
5	China	1999	212258/1272915272
6	China	2000	213766/1280428583

	country	year	key	value
1	Afghanistan	1999	cases	745
2	Afghanistan	1999	population	19987071
3	Afghanistan	2000	cases	2666
4	Afghanistan	2000	population	20595360
5	Brazil	1999	cases	37737
6	Brazil	1999	population	172006362
7	Brazil	2000	cases	80488
8	Brazil	2000	population	174504898
9	China	1999	cases	212258
10	China	1999	population	1272915272
11	China	2000	cases	213766
12	China	2000	population	1280428583

Messy to Tidy Data

1. The table header contains values
 - Identify the variables and distribute (unpivot) the values
- Read the `pew.csv` dataset
 - Distribution of income by religion
- Show the first 5 values (use the `head()` function)
 - Also see the number of variables and observations (`shape`)
 - This will also ensure that you've read the dataset correctly
 - **Variables:** religion, income, frequency
- Transform the dataset to make it tidy ([docs](#))

```
pew = pd.read_csv("pew.csv")
pew_tidy = pd.melt(pew,
    id_vars = ["religion"], # Identifier variables (all others are "unpivoted")
    var_name = "income", # Variable
    value_name = "frequency") # Value
```


Messy to Tidy Data (2)

2. Multiple variables stored in one column

- Identify and split the variables into separate columns
- Read the `tb.csv` dataset
 - Tuberculosis cases
 - `m04`, `m514`, `m1524`, etc. contain two variables (gender and age)
 - male, 0-4 years old; male, 5-14 years old, etc.
 - There's also a problem with missing values (NaN)
- Tidying process
 - First, melt all columns (they are values and should not be)
 - Next, split the column names and extract the gender and age information
 - Add the new info to the dataset
 - Remove all missing values

Messy to Tidy Data (3)

```
tb = pd.read_csv("tb.csv")
# Melt the values
tb = pd.melt(tb, id_vars = ["iso2", "year"],
             var_name = "sex_and_age", value_name = "cases")

# Separate the columns and merge back
parts = tb["sex_and_age"].str.extract("(\D)(\d+)(\d{2})", expand = True)
parts.columns = ["sex", "age_lower", "age_upper"]
parts["age"] = parts["age_lower"] + "-" + parts["age_upper"]
tb = pd.concat([tb, parts], axis = 1)

# Remove missing values and sort them
tb = tb.drop(["sex_and_age", "age_lower", "age_upper"], axis = 1)
tb = tb.dropna()
tb = tb.sort_values(by = ["iso2", "year", "sex", "age", "cases"])

# The index is now wrong, reindex to make it better
tb = tb.reset_index()
del tb["index"]
```

Messy to Tidy Data (4)

3. Variables are stored in both rows and columns
 - Identify and split the variables
 - Read the `weather.csv` dataset
 - Daily weather records in Mexico in 2010
 - `d1`, `d2`, etc. are the days of a month; also `tmin` and `tmax` should be columns
 - Make a new column with the date: `[date, tmin, tmax]`
 - Tidying process
 - Melt all days
 - Create days based on date, month and year
 - Pivot the `tmin` and `tmax` columns

Messy to Tidy Data (5)

```
temp_data = pd.read_csv("weather.csv")

temp_data = pd.melt(temp_data,
    id_vars = ["id", "year", "month", "element"],
    var_name = "day")
temp_data["day"] = temp_data["day"].str.extract("(\d+)",
    expand = True).astype(np.int64)

# Remove missing / invalid days (e.g. 31st April) and dates with no records
temp_data = temp_data.dropna()
temp_data["date"] = pd.to_datetime(temp_data[["year", "month", "day"]])
temp_data = temp_data.drop(["year", "month", "day"], axis = 1)

# Pivot the elements back to their own columns
temp_data = temp_data.pivot_table(index = ["id", "date"],
    columns = "element", values = "value")

# Pivoting returns a multi-indexed element, go back to a flat DataFrame
temp_data.reset_index(inplace = True)
temp_data.columns.name = ""
```

Messy to Tidy Data (6)

4. One type in multiple tables

- Merge the tables into one
 - Read all tables, add the new columns
 - Often the filename should be in its own column (if it's important)
 - Melt and tidy if necessary

5. Multiple types in one table

- Split into more tables
 - If necessary, introduce relations (similar to a relational database)
- Each table should be responsible for one type of measurement
- * Read the `billboard.csv` dataset and apply those transformations

Operations on Datasets

**Basic tools to get started
working with messy data**

Subsetting Rows

- Selecting only some rows (aka **selection**)
- First / last n records (observations)

```
temp_data.head(10)  
temp_data.tail() # 5 by default
```

- Random n records

```
temp_data.sample(n = 10)  
temp_data.sample() # 1 random record by default
```

- Smallest / largest n records in a given column

```
temp_data.nsmallest(3, "tmax")  
temp_data.nlargest(3, "tmax")
```

- Subsetting by a Boolean expression (predicate)
 - Returns only rows where the expression returns True

```
temp_data[temp_data.tmax > 30]
```

Subsetting Columns

- Selecting only some columns (aka **projection**)
- Single column (returns a Series object)

```
temp_data["tmax"]  
temp_data.tmax # Possible in most cases
```

- More than one column (returns a DataFrame object)

```
temp_data[["tmin", "tmax"]]
```

- Combining filters

```
temp_data[temp_data.date > "2010-08-01"][["date", "tmax"]]  
temp_data.loc[temp_data.date > "2010-08-01", ["date", "tmax"]]
```

- A note on Boolean expressions

- "and", "or", "not" are &, |, ~
- **Always** put parentheses around the individual expressions

```
temp_data[(temp_data.date > "2010-08-01") & (temp_data.date < "2010-09-01")]
```


Summary Statistics and Grouping

- These methods work by columns
 - If multiple columns are passed, they are applied to each column individually

```
print("Count:", temp_data.tmin.count()) # number of non-null values
print("Min:", temp_data.tmin.min())
print("Max:", temp_data.tmin.max())
print("Mean:", temp_data.tmin.mean())
print("Median:", temp_data.tmin.median())
print("Standard deviation:", temp_data.tmin.std())
```

- Grouping
 - Splits the data into several groups based on the values of a column
 - We have to apply a method after grouping
 - Or iterate over the groups (using a for-loop)
 - Example: Average number of people for each income group

```
pew_tidy.groupby("income").mean()
```

Cleaning Data

You've got the data... now what?

Cleaning Data

- No common way of doing this
- We have to rely on intuition and some common patterns
 - Tidy up the dataset
 - You have to know the dataset documentation first
 - Treat nulls / NaNs: either remove them or replace them
 - Replacing values might be dangerous
 - If done properly, it will affect the data in a positive way
 - Identify and fix errors (also dangerous)
 - Melt and pivot datasets
 - Merge (join) and separate datasets
 - Subset variables and / or observations
 - Summarize and group variables
 - [Pandas cheat sheet](#)

Example: Weather Data

- Since there's no common way of cleaning, we'll explore and clean a dataset, showing steps and examples as we go
- Dataset (weather data, courtesy of [synesthesiam@github](#))
- Read the dataset (you don't need to download it)
 - See how many variables and observations are there
 - Display the first and last few rows to get a sense of the data
 - Check the data types (to see if something's wrong with the reading)
 - E.g. numbers recognized as strings
 - See a subset of the columns
 - Summarize (describe) the dataset

Example: Weather Data (2)

- The column names don't look good
 - Make them "pythonic" (lowercase_with_underscores)
 - This will make selecting them easier (weather.mean_temp)

```
weather.columns = ["date", "max_temp", "mean_temp", "min_temp", "max_dew",  
                  "mean_dew", "min_dew", "max_humidity", "mean_humidity",  
                  "min_humidity", "max_pressure", "mean_pressure",  
                  "min_pressure", "max_visibility", "mean_visibility",  
                  "min_visibility", "max_wind", "mean_wind", "min_wind",  
                  "precipitation", "cloud_cover", "events", "wind_dir"]
```

- What are the ranges of data?
 - E. g. temperature, pressure, humidity
 - Use the min() and max() methods
- * Try to explore the data a bit
 - Plot a few histograms and / or boxplots to see the distributions

Example: Weather Data (3)

- Convert the dates to a datetime object
 - To make performing time-dependent analysis easier
 - Use `apply()` to perform a function on every row

```
from datetime import datetime
def string_to_date(date_string):
    return datetime.strptime(date_string, "%Y-%m-%d")
```

```
weather.date = weather.date.apply(string_to_date)
```

- It's even better to use dates as indices (when we need to subset date ranges or perform other time-dependent tasks)

```
weather.index = weather.date
weather = weather.drop("date", axis = 1) # We don't need it twice,
# axis = 1 tells pandas to search for a column (axis = 0 -> row)

print(weather.loc[datetime(2012, 8, 19)]) # or weather.loc["2012-08-19"]
```

- Also see why precipitation is not a float and edit it

Example: Weather Data (4)

- Remove or replace missing values
 - In this case, replacing is better because removing takes away an entire row

```
weather_with_events = weather.dropna(subset = ["events"])  
weather.events = weather.events.fillna("") # Better
```

- Try to see how variables interact – group the data
 - E.g. by cloud cover and events
 - Print the number of days each combination of {cover, events} occurred

```
for (cover, events), group_data in weather.groupby(["cloud_cover", "events"]):  
    print("Cover: {0}, Events: {1}, Count: {2}"  
          .format(cover, events, len(group_data)))  
# Or: weather.groupby(["cloud_cover", "events"]).size()
```

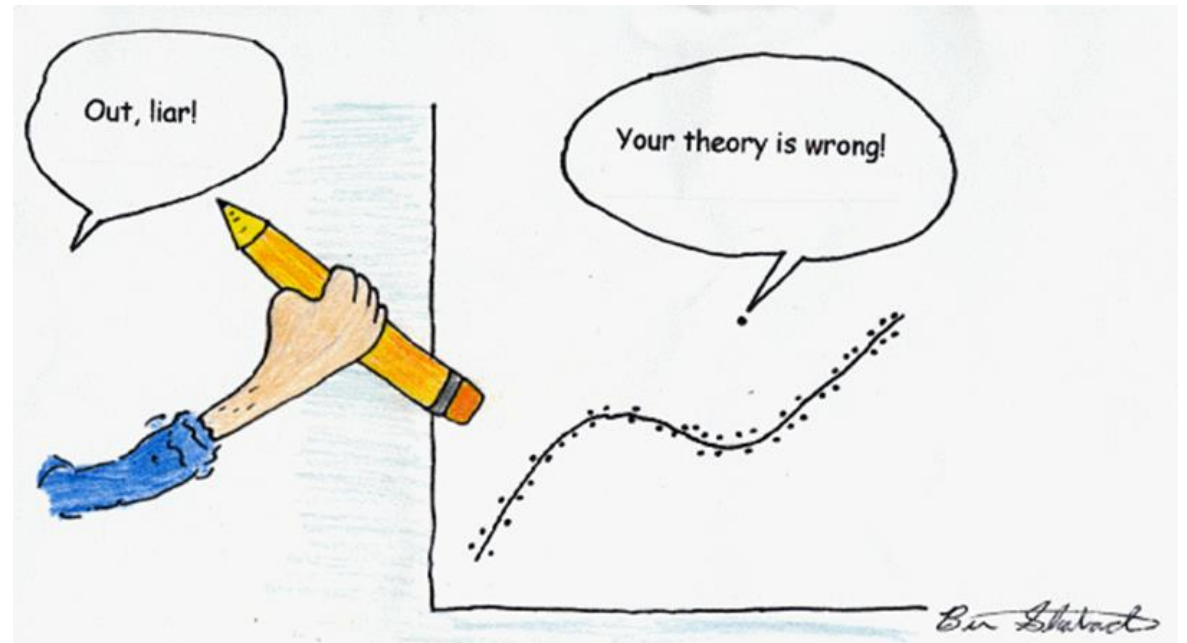
- Plot data – next time

Example: Weather Data (5)

- If needed, perform transformations
 - Math operations: log, square root, addition, multiplication, etc.
 - Be careful as you'll get results in different dimensions
 - Normalizing scores (such as using Z-scores) is recommended in most cases
 - It's much better for ML algorithms to have data of similar scales
 - You can do that manually or use a library (such as [sklearn.preprocessing](https://scikit-learn.org/stable/modules/preprocessing.html))
 - By convention, calculated columns are added to the dataset
- **Describe all operations as you're doing them**
 - Describe what you're doing and why
 - Useful to check your work later (or allow others to do that)
 - If needed, save the resulting dataset into a file
 - Supply your data transformation log with it
 - Provide a dataset description

Outliers and Errors

- **Outliers** – values which are far from their expected range
 - Or having a very low probability of happening (assuming a model)
- Many possible cases
 - Wrong data entry (e.g. an adult weighing 5kg might be 50kg or something else)
 - Wrong assumptions (the data is correct, our view isn't)
- What to do?
 - Inspect the data point
 - Try to figure out what happened
 - If needed, remove the row or try to replace the value
 - Try a transformation
 - If possible, perform analysis with and without the outlier(s) and compare your results



Transformations on Features

- The quality of our results depends strongly on the features we use
 - "Garbage in – garbage out"
- Dimensionality reduction
 - Reducing the number of variables (features)
 - We can do this manually or use algorithms
 - Feature selection
 - Selecting only columns that are useful
 - Feature extraction
 - Transforming non-structured to structured data
 - Examples: images, audio, text
 - Getting meaningful features
- Feature engineering
 - Using our knowledge of the data to create meaningful features
 - Involves a lot of brainstorming and testing

Next Steps (Optional)

- Have a look at `scikit-learn`'s "Dataset Transformations" module
 - It describes the most common operations
 - Data cleaning
 - Dimensionality reduction
 - Feature extraction
- There are many algorithms based on
 - Data types (e.g. text or numerical data, labelled vs. not labelled)
 - Model types (how we want to present our data, e.g. linear model)
 - Algorithm types (e.g. finding similar news articles, recommending movies to users, classifying, etc.)
- No "hard and fast rule", use your intuition
 - Knowing more tools / models / algorithms -> better performance

Summary

- Messy and tidy data
 - Tidying up messy data
- Operations on datasets
- Cleaning data
 - Validation
 - Transformation
 - Error correction
 - Features
- Data tidying and cleaning as a process

The image features a white background with two blue decorative bars. The top bar is a solid blue strip. The bottom bar is a gradient blue strip that transitions from a lighter blue on the left to a darker blue on the right. The word "Questions?" is centered in a blue, sans-serif font.

Questions?