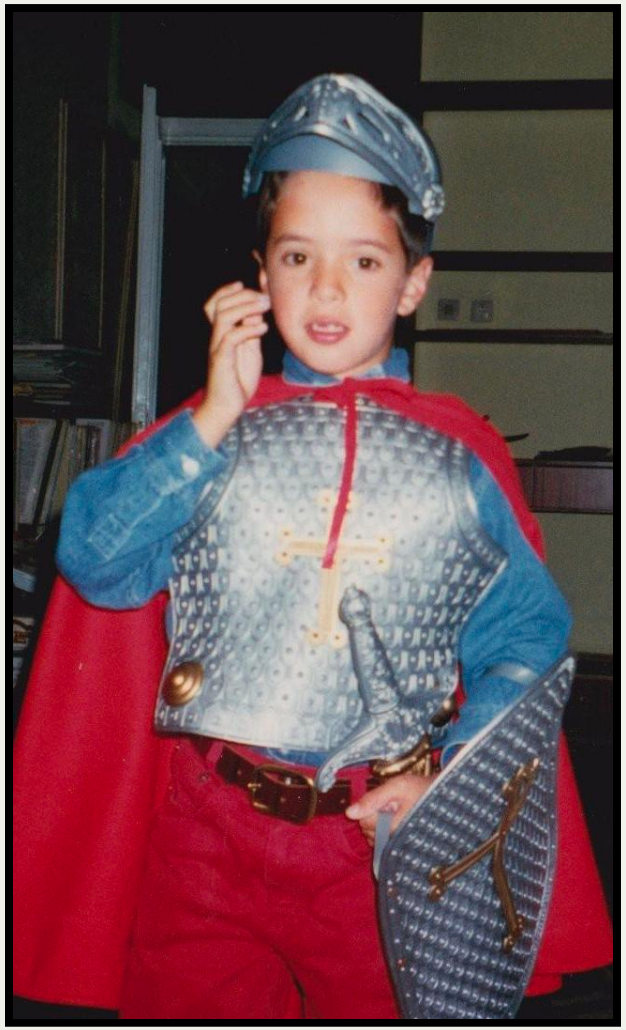
Hell is Other People's Data

Mali Akmanalp / @makmanalp



Mali Akmanalp / @makmanalp

THE ATLAS OF

Deseo recibir información de las Planillas en su celular vía Mensaje de Texto (SMS)	☐
Desea recibir información de las planillas vía correo?	☐
Actividad económica	Lista de actividades económicas

MAPPING PATHS TO PROSPERITY

Ricardo Hausmann, César A. Hidalgo, et al.

ACTIVIDADES ECÓNOMICAS

CÓDIGO	DESCRIPCIÓN
111	EMPRESAS DEDICADAS A LA PRODUCCION ESPECIALIZADA DEL CAFÉ
112	EMPRESAS DEDICADAS A LA PRODUCCION ESPECIALIZADA DE FLOR DE CORTE BAJO CUBIERTA Y AL AIRE LIBRE, INCLUYE SOLAMENTE LOS INVERNADEROS, CULTIVO FLORICULTURA.
113	EMPRESAS DEDICADAS A LA PRODUCCION ESPECIALIZADA DE BANANO
114	EMPRESAS DEDICADAS A LA PRODUCCION ESPECIALIZADA DE CAÑA DE AZUCAR
115	EMPRESAS DEDICADAS A LA PRODUCCION ESPECIALIZADA DE CEREALES Y OLEAGINOSAS, EMPRESAS DEDICADA A LA PRODUCCION DE ACEITE DE PALMA
116	EMPRESAS DEDICADAS A LA PRODUCCION ESPECIALIZADA DE HORTALIZAS Y LEGUMBRES
117	EMPRESAS DEDICADAS A LA PRODUCCION ESPECIALIZADA DE FRUTAS, NUECES, PLANTAS BEBESTIBLES Y ESPECIAS, INCLUYE EL TOSTADO Y-BENEFICIO DEL CACAO
118	EMPRESAS DEDICADAS A LA PRODUCCION AGRICOLA NCP EN UNIDADES ESPECIALIZADAS, INCLUYE LAS EMPRESAS DE BENEFICIO DE TABACO. PRODUCCION AGRICOLA NCP EN UNIDADES ESPECIALIZADAS INCLUYE SOLAMENTE A EMPRESAS DEDICADAS A LA INDUSTRIA DE LA PRODUCCION DE CAUCHO NAT
119	EMPRESAS DEDICADAS A LA PRODUCCION AGRICOLA EN UNIDADES NO ESPECIALIZADAS, INCLUYE LA AGRICULTURA NO MECANIZADA NI CONTEMPLADA EN OTRAS EMPRESAS DEDICADAS A ACTIVIDADES (SIEMBRA, CULTIVO Y/O RECOLECCION)


```
import pandas as pd
```

Military surplus gear given to law enforcement

https://github.com/TheUpshot/Military-Surplus-Gear

Reading Data In

```
In [1]: pd.read_<tab>
pd.read_clipboard  pd.read_fwf        pd.read_html       pd.read_pickle
pd.read_sql_table  pd.read_csv        pd.read_gbq        pd.read_json
pd.read_sql        pd.read_stata      pd.read_excel      pd.read_hdf
pd.read_msgpack    pd.read_sql_query  pd.read_table
```

Encoding Issues

```
In [5]: pd.read_stata("state_names.dta")
Out[5]:
    r                          r_name
0   1                  Aguascalientes
...
14  15                         MÃ©xico
15  16            MichoacÃ¡n de Ocampo
```

```
In [6]: pd.read_stata("state_names.dta", encoding="utf-8")
Out[6]:
     r                          r_name
0    1                  Aguascalientes
...
14  15                          México
15  16              Michoacán de Ocampo
```

```
>>> import chardet
>>> chardet.detect(rawdata)
{'encoding': 'EUC-JP', 'confidence': 0.99}
```

```
$ file boston_python_is_awesome.tsv
boston_python_is_awesome.tsv: UTF-8 Unicode English text
```

```
>>> from unidecode import unidecode
>>> print u'H\xeb\xe4vy M\xebt\xe4l'
Hëävy Mëtäl
>>> unidecode(u'H\xeb\xe4vy M\xebt\xe4l')
'Heavy Metal'
```

Taking a peek

```
In [89]: df = pd.read_excel("1033-program-foia-may-2014.xlsx")
```

```
In [24]: df.columns
Out[24]: Index([u'State', u'County', u'NSN', u'Item Name', u'Quantity',
u'UI', u'Acquisition Cost', u'Ship Date'], dtype='object')
```

```
In [6]: df.describe()
Out[6]:
           Quantity  Acquisition Cost
count  73028.000000      73028.000000
mean      15.006792       7967.575490
std      384.623930     197293.243356
min        1.000000          0.000000
25%        1.000000         58.710000
50%        1.000000        200.000000
75%        5.000000        499.000000
max    91000.000000   18000000.000000
```

```
In [90]: df.info()
<class 'pandas.core.frame.DataFrame'>
Int64Index: 73028 entries, 0 to 73027
Data columns (total 8 columns):
State                73028 non-null object
County               73028 non-null object
NSN                  72983 non-null object
Item Name            71732 non-null object
Quantity             73028 non-null int64
UI                   73028 non-null object
Acquisition Cost     73028 non-null float64
Ship Date            73028 non-null object
dtypes: float64(1), int64(1), object(6)
```

Types are important!

```
5 != '5'
```

5 != 5.0

Adding nulls make int columns floats!

```
In [32]: x = pd.read_csv(StringIO("a,b\n5,6.0\n,"))

In [33]: x
Out[33]:
    a   b
0   5   6
1 NaN NaN

In [34]: x.info()
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2 entries, 0 to 1
Data columns (total 2 columns):
a    1 non-null float64
b    1 non-null float64
dtypes: float64(2)
memory usage: 48.0 bytes
```

Merging on different types makes data disappear

Converting types

```
pd.read_csv(..., dtype={"column_1": int, "column_2": object})
# or
df.column = df.column.astype(int)
```

Kinds of nothing

- No data available: None
- Nully values: "", 0

The Semantics of Types

Identifiers

(Not numbers!)

- 617-555-1234
- 721-07-1426
- 01605

Categoricals

(Not strings!)

- red, green, blue, orange
- A, B, AB, 0
- noun, verb, adjective, adverb

Ordinals

(Also not strings!)

- low, medium, high
- ★, ★★, ★★★, ★★★★
- high school, undergrad, graduate

Dealing with missing data

Converting custom N/A values

```
pd.read_csv(..., na_values=["N/A", "Unknown"])
# or
df.replace("N/A", None)
```

Dropping nulls

```
Out[65]:
     a  b   c   d
0    1  2 NaN NaN
1  NaN  3 NaN NaN
2    4  5 NaN NaN
```

```
In [68]: df.dropna(axis=1)
Out[68]:
   b
0  2
1  3
2  5
```

```
In [69]: df.dropna(axis=1, how="all")
Out[69]:
    a  b
0   1  2
1 NaN  3
2   4  5
```

```
In [86]: df.fillna(method="bfill")
Out[86]:
   a  b   c   d
   0  1  2 NaN NaN
   1  4  3 NaN NaN
   2  4  5 NaN NaN
```

```
In [90]: df.interpolate()
Out[90]:
     a  b   c   d
     0  1.0  2 NaN NaN
     1  2.5  3 NaN NaN
     2  4.0  5 NaN NaN
```

Playing with data

Selecting columns

```
In [50]: df.State
Out[50]:
0      AK
1      AK
2      AK
...

In [51]: df["Item Name"]
Out[51]:
0      RIFLE,5.56 MILLIMETER
1      RIFLE,5.56 MILLIMETER
2      RIFLE,5.56 MILLIMETER
3      RIFLE,5.56 MILLIMETER
4      RIFLE,5.56 MILLIMETER
...
```

Selecting multiple columns

```
In [54]: df[["State", "County"]]
Out[54]:
      State                      County
0        AK  ANCHORAGE
1        AK  ANCHORAGE
2        AK  ANCHORAGE
```

Queries

```
In [58]: df["Acquisition Cost"] > 100000
Out[58]:
0      False
1      False
2      False
3      False
4      False
5      False
...
```

Queries

```
In [59]: df[df["Acquisition Cost"] > 100000]
Out[59]:
      State                    County                NSN  \
146      AK  ANCHORAGE                    2355-DS-COM-BTV2
515      AL  BALDWIN                      2320-01-047-8754
663      AL  BLOUNT                       2320-01-047-8754
674      AL  BLOUNT                       2320-01-230-0304
693      AL  BLOUNT                       2355-01-555-0908
...
In [60]: df[df["Acquisition Cost"] > 100000].count()
Out[60]:
State                646
County               646
NSN                  645
Item Name            641
...
```

What were the highest cost items?

```
In [38]: df.sort("Acquisition Cost", ascending=False)\
[["Item Name", "Acquisition Cost"]]
Out[38]:
                                                Item Name  Acquisition Cos
65754                              AIRCRAFT, ROTARY WING          1800000
65753                              AIRCRAFT, ROTARY WING          1800000
65760                              AIRCRAFT, ROTARY WING          1800000
65759                              AIRCRAFT, ROTARY WING          1800000
65758                              AIRCRAFT, ROTARY WING          1800000
65757                              AIRCRAFT, ROTARY WING          1800000
65756                              AIRCRAFT, ROTARY WING          1800000
65755                              AIRCRAFT, ROTARY WING          1800000
48074                              AIRCRAFT, ROTARY WING           650000
19524                          AIRPLANE,CARGO-TRANSPORT           534000
65678                               AIRCRAFT, FIXED WING           431700
2150                         RADAR SURVEILLANCE CENTRAL           152670
```

What were the largest quantities of items?

```
In [15]: df.sort("Quantity", ascending=False)\
[["Item Name", "Quantity", "UI"]]
Out[15]:
```

	Item Name	Quantity	UI
52530	WIRE,ELECTRICAL	91000	Foot
36221	SCREW,CAP,SOCKET HEAD	43822	Each
52399	STRAP,TIEDOWN,ELECTRICAL COMPONENTS	6000	Each
52536	CABLE COAX	6000	FT
39189	RUBBER SHEET,SOLID	6000	Each

value_counts() is awesome

```
In [16]: df.UI.value_counts()
Out[16]:
Each         51581
EA           13370
Pair          1381
PR            1285
Unknown        982
Kit            732
Set            463
KT             441
SE             360
```

```
In [17]: df = df.replace({"UI":["EA", "EACH", "PR" ...]},
                         {"UI":["Each", "Each", "Pair" ...]})
In [18]: df = df.replace({"UI":"Unknown"}, {"UI":np.NaN})
```

```
In [22]: df.UI.value_counts()[20:]
Out[22]:
Assortment     10
GL              9
Can             8
OT              7
BD              7
...
JR              1
E4              1
Skein           1
UU              1
SP              1
Length: 54, dtype: int64
```

```
dontcare = df.UI.value_counts()[20:]
df = df[~df.UI.isin(dontcare.index)]
```

```
In [46]: df.sort("Quantity", ascending=False)\
   ....: [["Item Name", "Quantity", "UI"]]
Out[46]:
                                      Item Name  Quantity       UI
52530                          WIRE,ELECTRICAL     91000     Foot
36221                SCREW,CAP,SOCKET HEAD         43822     Each
52399        STRAP,TIEDOWN,ELECTRICAL COMPONENTS      6000     Each
52536                               CABLE  COAX      6000     Foot
```

Regexes and other string functions

```
>>> df.column[df.column.str.contains("(\d{4}-\d{2}-\d{2})")]
>>> df["Item Name"].str.lower()
```

Pivoting

```
In [1]: df
Out[1]:
         date variable     value
0  2000-01-03        A  0.469112
1  2000-01-04        A -0.282863
2  2000-01-05        A -1.509059
3  2000-01-03        B -1.135632
4  2000-01-04        B  1.212112
5  2000-01-05        B -0.173215
....
```

Pivoting

```
In [3]: df.pivot(index='date', columns='variable', values='value')
Out[3]:
variable            A         B         C         D
date
2000-01-03  0.469112 -1.135632  0.119209 -2.104569
2000-01-04 -0.282863  1.212112 -1.044236 -0.494929
2000-01-05 -1.509059 -0.173215 -0.861849  1.071804
```

Transforming with functions

```
In [128]: df4
Out[128]:
        one     three       two
a -0.626544       NaN -0.351587
b -0.138894 -0.177289  1.136249
c  0.011617  0.462215 -0.448789
d       NaN  1.124472 -1.101558

In [129]: f = lambda x: len(str(x))

In [130]: df4['one'].map(f)
Out[130]:
a    14
b    15
c    15
d     3
Name: one, dtype: int64
```

Merging Datasets

Bane of everyone's existence

- Make sure the types match
- Check dataframe size before and after
- Try with how=inner and how=outer

```
In [83]: ages
Out[83]:
   age    cats
0    1   simon
1    2  phoebe
2    3  norman
In [84]: weights
Out[84]:
     cats  weight
0   simon       1
1  norman       2
```

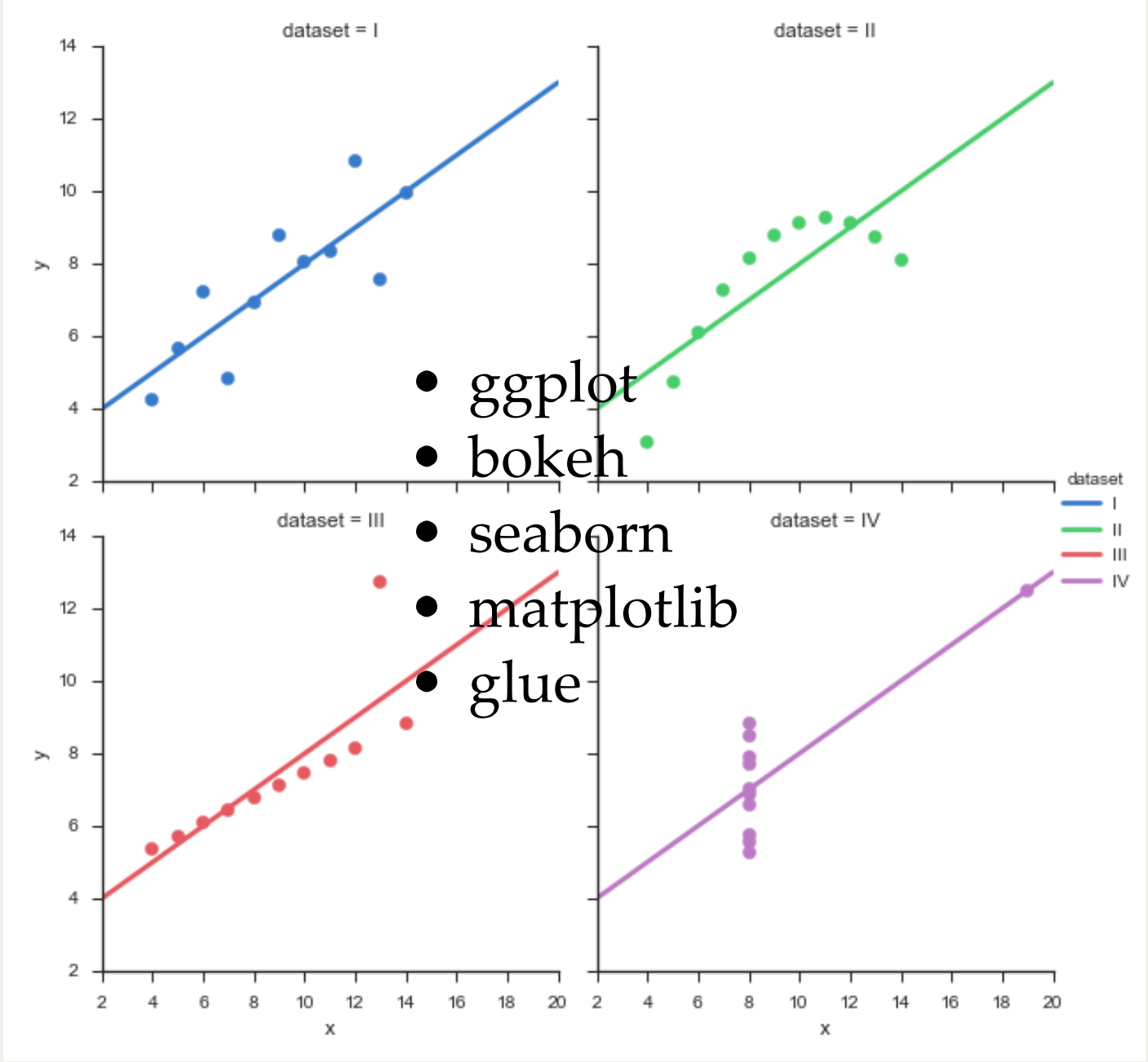
```
In [86]: pd.merge(ages, weights, left_on="cats", right_on="cats")
Out[86]:
   age    cats  weight
0    1   simon       1
1    3  norman       2
```

```
In [88]: pd.merge(ages, weights, left_on="cats", right_on="cats",
                  how="outer")
Out[88]:
   age    cats  weight
0    1   simon       1
1    2  phoebe     NaN
2    3  norman       2
```

fuzzywuzzy and jellyfish

```
>>> from fuzzywuzzy import fuzz
>>> fuzz.ratio("this is a test", "this is a test!")
    96
```

Graphs!


dataset = I
dataset = II
dataset = III
dataset = IV
dataset
I
II
III
IV
x
y
- ggplot
- bokeh
- seaborn
- matplotlib
- glue

PDFs suck!

Monthly Employment Situation for Veterans

Released Monthly on the day of the U.S. Employment Situation

Economic Information & Analysis, Illinois Department of Employment Security

Release date: December 6, 2013

National Data (Only data from the Monthly Report, Table A-5 will be updated monthly)

The U.S. Bureau of Labor Statistics reports monthly the employment status of the civilian population 18 years and over by veteran status, period of service and sex, not seasonally adjusted. (Table A-5) **In November 2013, the unemployment rate for all Veterans over 18 was 6.7%, up from 6.6% in November 2012. The unemployment rate for Gulf War-era II Veterans over 18 was 9.9% in November 2013, down from 10.0% in November 2012.** These numbers are not seasonally adjusted; they come from a small sample and are highly volatile from one month to the next. The December employment figures will be available on January 10, 2014. Here is a link to the November U.S. Employment Situation http://www.bls.gov/schedule/archives/empsit_nr.htm#current .

U.S. Bureau of Labor Statistics released a detailed report on the Employment Situation of Veterans-2012 on March 20, 2013 (annual release each March). The table below shows some key figures from this report. Here is a link to the report: http://www.bls.gov/news.release/vet.nr0.htm

Unemployment Rates, annual averages	2012	2011
Total Veterans, 18 & older	7.0	8.3
Gulf War-era II Veterans	9.9	12.1
Nonveterans	7.9	8.7
Gulf War era II		
Total, 18 & Over	9.9	12.1
18 - 24	20.4	30.2
25 - 34	10.6	13.0
35 - 44	5.0	6.9
45 - 54	7.7	4.1
55 - 64	5.8	7.8

Please note, because the estimates for the veterans' annual average unemployment provided in the Veterans' Employment Situation – 2012 are more accurate than the other estimates discussed here, this is likely the best data available for users to reference in public presentations, grant applications, etc.

State Data

Employment and unemployment statistics by Veteran status for the state of Illinois are only available annually. These annual averages are produced by the U.S. Bureau of Labor Statistics for a special request from the Veteran's Employment and Training Services (VETS) in the Department of Labor. These figures are used by VETS in their State allocations under the Jobs for Veterans Act. **The Illinois unemployment rate for Veterans aged 20 years and**

Scaling up

Advanced Options

Download All Data

Clear All Selections

Multi-Select Mode

Help

Don't let notebooks get in the way of reusable code

Sometimes repeatability matters

- Build tools: Make, tup
- OKFN Bubbles, ETLs, Hadoop, Storm, etc

Be conscious about what you load into memory

```
pd.read_csv(..., usecols=["blah"])
pd.read_csv(..., iterator=True, chunksize=100000)
```

Push things down into the pandas / numpy layer

Takeaways

Use the proper types for things

Data has a tendency to be used in unanticipated ways

Documentation matters

Fix data before you need it fixed!

Data cleaning is a necessary *evil*

Thank you!

@makmanalp