Introduction to Data Science

Data Wrangling

Joanna Bieri DATA101

Important Information

- Email: joanna_bieri@redlands.edu
- Office Hours: Duke 209 Click Here for Joanna's Schedule

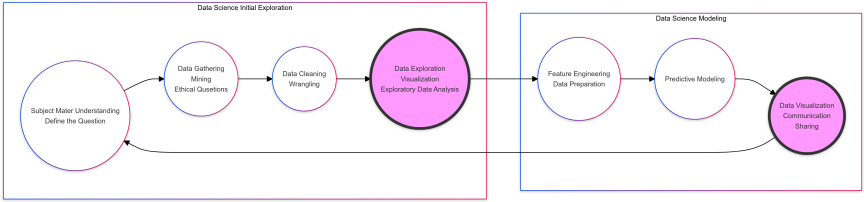
Announcements

Please come to office hours to get help!

Remember to come to lab!

The Data Science Lifecycle

Today we are talking about **Data Wrangling**



Tidy Data

What is Clean or Tidy Data?

Characteristics of Tidy Data:

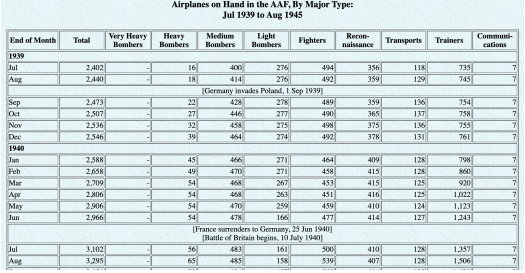
- Each variable forms a column
- Each observation forms a row
- Observational units form a table

Untidy Data:

- Anything other kind of mess.

Some Examples of Data

Air force Data: Army Air Forces Statistical Digest, WW II



Airplanes on Hand in the AAF, By Major Type: Jul 1939 to Aug 1945

End of Month	Total	Very Heavy Bombers	Heavy Bombers	Medium Bombers	Light Bombers	Fighters	Recon-naissance	Transports	Trainers	Communi-cations
1939										
Jul	2,402	-	16	400	276	494	356	118	735	7
Aug	2,440	-	18	414	276	492	359	129	745	7
[Germany invades Poland, 1 Sep 1939]										
Sep	2,473	-	22	428	278	489	359	136	754	7
Oct	2,507	-	27	446	277	490	365	137	758	7
Nov	2,536	-	32	458	275	498	375	136	755	7
Dec	2,546	-	39	464	274	492	378	131	761	7
1940										
Jan	2,588	-	45	466	271	464	409	128	798	7
Feb	2,658	-	49	470	271	458	415	128	860	7
Mar	2,709	-	54	468	267	453	415	125	920	7
Apr	2,806	-	54	468	263	451	416	125	1,022	7
May	2,906	-	54	470	259	459	410	124	1,123	7
Jun	2,966	-	54	478	166	477	414	127	1,243	7
[France surrenders to Germany, 25 Jun 1940] [Battle of Britain begins, 10 July 1940]										
Jul	3,102	-	56	483	161	500	410	128	1,357	7
Aug	3,295	-	65	485	158	539	407	128	1,506	7

Figure 1: Data Image

- Merged cells to represent date
- There are historical markers

Some Examples of Data

	A	AA	AB	AC	AD	AE	AF	AG	AH
1	Estimated HIV Prevalence% - (Ages 15-49)	2004	2005	2006	2007	2008	2009	2010	2011
2	Abkhazia								
3	Afghanistan						0.06	0.06	0.06
4	Akrotiri and Dhekelia								
5	Albania								
6	Algeria	0.1	0.1	0.1	0.1	0.1			
7	American Samoa								
8	Andorra								
9	Angola	1.9	1.9	1.9	1.9	2	2.1	2.1	2.1
10	Anguilla								
11	Antigua and Barbuda								
12	Argentina	0.4	0.4	0.4	0.4	0.5	0.4	0.4	0.4
13	Armenia	0.1	0.1	0.1	0.1	0.1	0.2	0.2	0.2
14	Aruba								
15	Australia	0.1	0.1	0.1	0.1	0.1	0.2	0.2	0.2
16	Austria	0.2	0.2	0.2	0.3	0.3	0.3	0.4	0.4
17	Azerbaijan	0.06	0.06	0.06	0.1	0.1	0.1	0.1	0.1
18	Bahamas	3	3	3	3.1	3.1	2.9	2.8	2.8

Figure 2: Data Image

- The orange cell is a title for the data set instead of a variable name.

Some Examples of Data

Subject	United States			
	Estimate	Margin of Error	Percent	Percent Margin of Error
EMPLOYMENT STATUS				
Population 16 years and over	255,797,692	+/-17,051	255,797,692	(X)
In labor force	162,184,325	+/-135,158	63.4%	+/-0.1
Civilian labor force	161,159,470	+/-127,501	63.0%	+/-0.1
Employed	150,599,165	+/-138,066	58.9%	+/-0.1
Unemployed	10,560,305	+/-27,385	4.1%	+/-0.1
Armed Forces	1,024,855	+/-10,363	0.4%	+/-0.1
Not in labor force	93,613,367	+/-126,007	36.6%	+/-0.1
Civilian labor force	161,159,470	+/-127,501	161,159,470	(X)
Unemployment Rate	(X)	(X)	6.6%	+/-0.1
Females 16 years and over	131,092,196	+/-11,187	131,092,196	(X)
In labor force	76,493,327	+/-75,824	58.4%	+/-0.1
Civilian labor force	76,350,498	+/-75,238	58.2%	+/-0.1
Employed	71,451,559	+/-79,007	54.5%	+/-0.1
Own children of the householder under 6 years	22,939,897	+/-14,240	22,939,897	(X)
All parents in family in labor force	14,957,537	+/-36,506	65.2%	+/-0.1
Own children of the householder 6 to 17 years	47,007,147	+/-19,644	47,007,147	(X)
All parents in family in labor force	33,238,793	+/-49,036	70.7%	+/-0.1

Figure 3: Data Image

- This data is at the aggregate level.

Displaying Data vs. Summarizing Data

Raw Data - Star Wars

	name	height	mass	hair_color	skin_color	eye_color
0	Luke Skywalker	172.0	77.0	blond	fair	blue
1	C-3PO	167.0	75.0	NaN	gold	yellow
2	R2-D2	96.0	32.0	NaN	white, blue	red
3	Darth Vader	202.0	136.0	none	white	yellow
4	Leia Organa	150.0	49.0	brown	light	brown
...	...	...	...	...	...	...
82	Finn	NaN	NaN	black	dark	dark
83	Rey	NaN	NaN	brown	light	hazel
84	Poe Dameron	NaN	NaN	brown	light	brown
85	BB8	NaN	NaN	none	none	black
86	Captain Phasma	NaN	NaN	none	none	unknown

Displayed Data

Focus on only two columns. Is this Tidy?

	name	height	mass
0	Luke Skywalker	172.0	77.0
1	C-3PO	167.0	75.0
2	R2-D2	96.0	32.0
3	Darth Vader	202.0	136.0
4	Leia Organa	150.0	49.0
...	...	...	...
82	Finn	NaN	NaN
83	Rey	NaN	NaN
84	Poe Dameron	NaN	NaN
85	BB8	NaN	NaN
86	Captain Phasma	NaN	NaN

Important Ideas - Data

- Start with **Raw Data**
- Data needs to be **Tidy**
- Most of the time, our data will not be tidy.
- We will have to **Clean** and **Wrangle**
- We must keep a record of what we do at every step.
- **Summarizing Data** is an important part of data science.

Data Wrangling

Data Wrangling

- We start with raw data and save it in a data frame
- The data frame stays on the left
- We can apply operations (functions) on the left
- The operations will not overwrite the data - so you can be experimental here and not lose your original data.

Load the raw data:

- Data from two hotels: one resort and one city hotel
- Observations: Each row represents a hotel booking
- Goal for original data collection: Development of prediction models to classify a hotel booking's likelihood to be canceled

Print to Screen

Two ways:

```
DF_raw_hotels
```

```
show(DF_raw_hotels)
```

What is the difference?

Explore Variables

What do these command do?

```
DF_raw_hotels.shape()
```

```
columns_list = list(DF_raw_hotels.keys())
print(columns_list)
```

```
print(len(columns_list))
```

Print variable names more nicely

```
for column in columns_list:
    print(column)
```

Focus on a specific columns

You just need to grab the column name (exactly spelled) and then tell the data frame to show only that column.

ONE:

```
DF_raw_hotels[['lead_time']]
```

TWO:

```
my_columns = ['hotel','lead_time']
DF_raw_hotels[my_columns].sort_values('lead_time')
```

Sorting the data

Ascending Order

Use the **sort_values()** command:

```
my_columns = ['hotel','lead_time']
DF_raw_hotels[my_columns].sort_values('lead_time')
```

Descending order

Just add an optional flag. Can you spot the difference in the code?

```
my_columns = ['hotel','lead_time']
DF_raw_hotels[my_columns].sort_values('lead_time',
                                      ascending=False)
```

One Big Command

```
DF_leadtime_sorted = DF_raw_hotels[my_columns].sort_values('lead_time', ascending=False).copy()
```

The Data Sorted

	hotel	lead_time
1	Resort Hotel	737
4182	Resort Hotel	709
65240	City Hotel	629
65249	City Hotel	629
65245	City Hotel	629
...	...	...
85725	City Hotel	0
85726	City Hotel	0
85758	City Hotel	0
85759	City Hotel	0
20017	Resort Hotel	0

Sorting Categorical Data

What should this command do?

```
DF_raw_hotels[["hotel","arrival_date_month"]].sort_values("arrival_date_month")
```

Sorting Categorical Data

	hotel	arrival_date_month
87641	City Hotel	April
87091	City Hotel	April
87090	City Hotel	April
87089	City Hotel	April
87088	City Hotel	April
...	...	...
8398	Resort Hotel	September
8397	Resort Hotel	September
8396	Resort Hotel	September
8404	Resort Hotel	September
98043	City Hotel	September

Sorting Categorical Data (extra)

We can tell python that our categorical data has an order:

```
DF_raw_hotels["arrival_date_month"] = pd.Categorical(DF_raw_hotels["
                                                     categories=['Ja
                                                                 'Ap
                                                                 'Ju
                                                                 'Oc
                                                      ordered=True)

DF_raw_hotels[["hotel","arrival_date_month"]].sort_values("arrival_d
```

Sorting Categorical Data (extra)

	hotel	arrival_date_month
13803	Resort Hotel	January
9942	Resort Hotel	January
9943	Resort Hotel	January
9944	Resort Hotel	January
9945	Resort Hotel	January
...	...	...
31141	Resort Hotel	December
15143	Resort Hotel	December
31142	Resort Hotel	December
31135	Resort Hotel	December
103323	City Hotel	December

Sub selecting specific categorical data

- Maybe my analysis only concerns specific hotels.
- Creating a **boolean mask** - a fancy way to say a bunch of True and False.

```
DF_leadtime_sorted['hotel']== 'Resort Hotel'
```

Notice the double equals ($==$) this is telling Python to test the truth of the statement. What does this do

Boolean Mask

```
1         True
4182      True
65240    False
65249    False
65245    False
         ...
85725    False
85726    False
85758    False
85759    False
20017     True
Name: hotel, Length: 119390, dtype: bool
```

Boolean Mask

Now we can give this mask to the data frame and it will only show us observations (rows) where "Resort Hotel" is true.

```
DF_leadtime_sorted[mask]
```

Boolean Mask - subselected data

	hotel	lead_time
1	Resort Hotel	737
4182	Resort Hotel	709
8397	Resort Hotel	542
8421	Resort Hotel	542
8422	Resort Hotel	542
...	...	...
3603	Resort Hotel	0
3602	Resort Hotel	0
3601	Resort Hotel	0
3600	Resort Hotel	0
20017	Resort Hotel	0

Summary of selecting, sorting, and masking.

What did we just do?

```
# Here are my parameters: column names and variable info
my_columns = ['hotel','lead_time']
sort_column = 'lead_time'
hotel_type = 'Resort Hotel'

# Sub select my columns and then sort descending
DF_sorted = DF_raw_hotels[my_columns].sort_values(sort_column,
                                                  ascending=Fa

# Do a mask so I am left with only Resort Hotels
mask = DF_leadtime_sorted['hotel']== 'Resort Hotel'
DF_final = DF_sorted[mask].copy()
```

Using Logical Operators

Basic Operators

Operator	Definition
$<$	less than
$>$	greater than
$<=$	less than or equal to
$>=$	greater than or equal to
$==$	exactly equal to
$!=$	not equal to

Advanced Operators

Operator	Definition
and	check if two things are both true
or	check if one of two things is true
in	checks if something is in another thing
!	not checks if something is false

Examples

3<10

3<10 and 2<10

'cat' == 'cat'

'CAT'=='cat'

my_numbers = [3,6,18,42] 42 in my_numbers

Fancy Stuff (EXTRA)

Excluding variables

Without typing in all the column names

```
# First get all of them
my_columns = list(DF_raw_hotels.keys())
# Then remove the one you don't want
my_columns.remove('agent')
# Then get the new data frame
DF_raw_hotels[my_columns]
```

Starts with

You can use the command **startswith()** to check if a variable starts with a word or part of a word. Here is code to get only columns that start with the word "arrival":

```
columns_list = list(DF_raw_hotels.keys())
column_mask = [column.startswith('arrival') for column in columns_li
my_columns = list(DF_raw_hotels.keys()[column_mask])
```

['arrival_date_year', 'arrival_date_month', 'arrival_date_week_number', 'arrival_date_day_of_month']

Ends with

You can use the command **endswith()** to check if a variable ends with a word or part of a word. Here is code to get only columns that end with the word "type":

```
columns_list = list(DF_raw_hotels.keys())
column_mask = [column.endswith('type') for column in columns_list]
my_columns = list(DF_raw_hotels.keys()[column_mask])
```

['reserved_room_type', 'assigned_room_type', 'deposit_type', 'customer_type']

IN

You can use the Python check **is in** to check if a variable contains a word or part of a word. Here is code to get only columns that contain the word "date":

```
columns_list = list(DF_raw_hotels.keys())
column_mask = ["date" in column for column in columns_list]
my_columns = list(DF_raw_hotels.keys()[column_mask])
```

['arrival_date_year', 'arrival_date_month', 'arrival_date_week_number', 'arrival_date_day_of_month', 'reservation_status_date']