MODULE / WEEK 03
DATA INTEGRITY WITH PYTHON SCRIPTS & DATA ANALYSIS

QMB4400
DATA ANALYSIS AND OPTIMIZATION

RASMUSSEN UNIVERSITY

1

LIVE CLASSROOM

- The Problem of Data Integrity
- Data Cleaning and Preparation
- JSON
- Python for Data Analysis

RASMUSSEN UNIVERSITY

2

DIRTY DATA

- Multitude of contributing factors
- Carrying data into dirty territory
- Common entry tips and tricks
- Get out of the Data Dumpster

RASMUSSEN UNIVERSITY

3

DIRTY DATA

- Analyst face this challenge
- Connecting aspects of data together via means of manipulation needs done regularly
- Data is dirty!
- Get out of the Data Dumpster

RASMUSSEN UNIVERSITY

4

DIRTY DATA

- Numeric data, pandas uses the floating-point value NaN (Not a Number) to represent missing data

	0	1	2	4
0	1.0	6.5	3.0	NaN
1	1.0	NaN	NaN	NaN
2	NaN	NaN	NaN	NaN
3	NaN	6.5	3.0	NaN

RASMUSSEN UNIVERSITY

5

DIRTY DATA

	0	1	2
0	1.0	6.5	3.0
1	1.0	NaN	NaN
3	NaN	6.5	3.0

RASMUSSEN UNIVERSITY

6

DIRTY DATA

	0	1	2
0	1.0	6.5	3.0

RASMUSSEN UNIVERSITY

7

DIRTY DATA

- Few ways to filter out missing data
  - pandas.isnull
  - Dropna
- Returns the series with only the non-null data & index values

RASMUSSEN UNIVERSITY

8

DIRTY DATA

In [17]: from numpy import nan as NA

In [18]: data = pd.Series([1, NA, 3.5, NA, 7])

In [19]: data.dropna()

Out[19]:

0 1.0

2 3.5

4 7.0

dtype: float64

=

This is equivalent to:

In [20]: data[data.notnull()]

Out[20]:

0 1.0

2 3.5

4 7.0

dtype: float64

RASMUSSEN UNIVERSITY

9

FILLING IN MISSING DATA

- Rather than filtering out missing data (and potentially discarding other data along with it), you may want to fill the "holes" in any number of ways.
- The fillna method is the workhorse function to use
- Call fillna with a constant replaces missing values with that value

RASMUSSEN UNIVERSITY

10

FILLING IN MISSING DATA

In [35]: df.fillna(0)

Out[35]:

	0	1	2
0	-0.204708	0.500000	0.000000
1	-0.555730	0.500000	0.000000
2	0.092908	0.500000	0.769023
3	1.246435	0.500000	-1.296221
4	0.274992	0.228913	1.352917
5	0.886429	-2.001637	-0.371843
6	1.669025	-0.438570	-0.539741

RASMUSSEN UNIVERSITY

11

FILLING IN MISSING DATA

- Calling fillna with a dict, you can use a different fill value for each column:

  In [36]: df.fillna({1: 0.5, 2: 0})

  Out[36]:

	0	1	2
0	-0.204708	0.000000	0.000000
1	-0.555730	0.000000	0.000000
2	0.092908	0.000000	0.769023
3	1.246435	0.000000	-1.296221
4	0.274992	0.228913	1.352917
5	0.886429	-2.001637	-0.371843
6	1.669025	-0.438570	-0.539741

RASMUSSEN UNIVERSITY

12

FILLING IN MISSING DATA

Fillna returns a new object, but you can modify the existing object in-place.

In [37]: _ = df.fillna(0, inplace=True)

In [38]: df

Out[38]:

	0	1	2
0	-0.204708	0.000000	0.000000
1	-0.555730	0.000000	0.000000
2	0.092908	0.000000	0.769023
3	1.246435	0.000000	-1.296221
4	0.274992	0.228913	1.352917
5	0.886429	-2.001637	-0.371843
6	1.669025	-0.438570	-0.539741

RASMUSSEN UNIVERSITY

13

FILLING IN MISSING DATA

The same interpolation methods available for reindexing can be used with fillna:

In [39]: df = pd.DataFrame(np.random.randn(6, 3))
In [40]: df.iloc[2:, 1] = NA In [41]: df.iloc[4:, 2] = NA
In [42]: df Out[42]:

RASMUSSEN UNIVERSITY

14

FILLING IN MISSING DATA

	0	1	2
0	0.476985	3.248944	-1.021228
1	-0.577087	0.124121	0.302614
2	0.523772	NaN	0.769023
3	-0.713544	NaN	-2.370232
4	-1.860761	NaN	NaN
5	1-265934	NaN	NaN

RASMUSSEN UNIVERSITY

15

FILLING IN MISSING DATA

In [43]: df.fillna(method='ffill')

Out[43]:

	0	1	2
0	0.476985	3.248944	-1.021228
1	-0.577087	0.124121	0.302614
2	0.523772	0.124121	0.769023
3	-0.713544	0.124121	-2.370232
4	-1.860761	0.124121	-2.370232
5	1-265934	0.124121	-2.370232

RASMUSSEN UNIVERSITY

16

FILLING IN MISSING DATA

- In [44]: df.fillna(method='ffill', limit=2)
- Out[44]:

	0	1	2
0	0.476985	3.248944	-1.021228
1	-0.577087	0.124121	0.302614
2	0.523772	0.124121	0.769023
3	-0.713544	0.124121	-2.370232
4	-1.860761	NaN	-2.370232
5	1-265934	NaN	-2.370232

RASMUSSEN UNIVERSITY

17

FILLING IN MISSING DATA

FILLNA FUNCTION ARGUMENT

Argument	Description
value	Scalar value or dict-like object to use to fill missing values
method	Interpolation; by default 'ffill' if function called with no other arguments
axis	Axis to fill on; default axis=0
inplace	Modify the calling object without producing a copy
limit	For forward and backward filling, maximum number of consecutive periods to fill

RASMUSSEN UNIVERSITY

18

DATA TRANSFORMATION

Duplicate rows may be found in a DataFrame for any number of reasons.

Example:

```
In [47]: data = pd.DataFrame({'k1': ['one', 'two'] * 3 + ['two'],
   ....:                      'k2': [1, 1, 2, 3, 3, 4, 4]})
In [48]: data
Out[48]:
   k1  k2
0  one   1
1  two   1
2  one   2
3  two   3
4  one   3
5  two   4
6  two   4
```

RASMUSSEN UNIVERSITY

19

DATA TRANSFORMATION

The DataFrame method duplicated returns a Boolean Series indicating whether each row is a duplicate (has been observed in a previous row) or not:

```
In [49]: data.duplicated()
Out[49]:
0    False
1    False
2    False
3    False
4    False
5    False
6     True
dtype: bool
```

RASMUSSEN UNIVERSITY

20

DATA TRANSFORMATION

Drop-duplicates returns a DataFrame where the duplicated array is False:

```
In [50]: data.drop_duplicates()
Out[50]:
   k1  k2
0  one   1
1  two   1
2  one   2
3  two   3
4  one   3
5  two   4
```

RASMUSSEN UNIVERSITY

21

DATA TRANSFORMATION

Both of these methods by default consider all of the column; alternatively, you can specify any subset of them to detect duplicates.

If we had an additional column of values and wanted to filter duplicates only based on 'k1' column:

```
In [51]: data['v1'] = range(7)

In [52]: data.drop_duplicates(['k1'])
Out[52]:
   k1   k2  v1
0  one   1   0
1  two   1   1
```

RASMUSSEN UNIVERSITY

22

DATA TRANSFORMATION

Duplicated and drop_duplicates by default keep the first observed value combination.

Passing keep='last' will return the last one:

```
In [53]: data.drop_duplicates(['k1', 'k2'], keep='last')
Out[53]:
   k1    k2  v1
0  one   1   0
1  two   1   1
2  one   2   2
3  two   3   3
4  one   3   4
6  two   4   6
```

RASMUSSEN UNIVERSITY

23

Transforming Data Using a Function or Mapping

For many datasets, you may wish to perform some transformation based on the values in an array, Series, or column in a DataFrame.

Consider the following hypothetical data collected about various kinds of meat:

```
In [54]: data = pd.DataFrame({'food': ['bacon', 'pulled pork', 'bacon',
   ....:                       'Pastrami', 'corned beef', 'Bacon',
   ....:                       'pastrami', 'honey ham', 'nova lox'],
   ....:                 'ounces': [4, 3, 12, 6, 7.5, 8, 3, 5, 6]})
```

RASMUSSEN UNIVERSITY

24

Transforming Data Using a Function or Mapping

```
In [55]: data
Out[55]:
        food        ounces
0      bacon         4.0
1      pulled pork   3.0
2      bacon        12.0
3      Pastrami      6.0
4      corned beef   7.5
5      Bacon         8.0
6      pastrami      3.0
7      honey ham     5.0
8      nova lox      6.0
```

RASMUSSEN UNIVERSITY

25

Transforming Data Using a Function or Mapping

Suppose you wanted to add a column indicating the type of animal that each food came from.

Write down a mapping of each distinct meat type to the kind of animal.

```
meat_to_animal = {
  'bacon': 'pig',
  'pulled pork': 'pig',
  'pastrami': 'cow',
  'corned beef': 'cow',
  'honey ham': 'pig',
  'nova lox': 'salmon'
}
```

RASMUSSEN UNIVERSITY

26

Transforming Data Using a Function or Mapping

The map method on Series accepts a function or dict-like object containing a mapping, but her we have a small problem in that some of the meats are capitalized and others are not.

```
In [57]: lowercased = data['food'].str.lower()

In [58]: lowercased
Out[58]:
0        bacon
1        pulled pork
2        bacon
3        pastrami
4        corned beef
5        bacon
6        pastrami
7        honey ham
8        nova lox
Name: food, dtype: object
```

RASMUSSEN UNIVERSITY

27

Transforming Data Using a Function or Mapping

```
In [59]: data['animal'] = lowercased.map(meat_to_animal)
In [60]: data
Out[60]:
        food        ounces  animal
0       bacon         4.0    pig
1       pulled pork   3.0    pig
2       bacon        12.0    pig
3       Pastrami      6.0    cow
4       corned beef   7.5    cow
5       Bacon         8.0    pig
6       pastrami      3.0    cow
7       honey ham     5.0    pig
8       nova lox      6.0    salmon
```

RASMUSSEN UNIVERSITY

28

Transforming Data Using a Function or Mapping

We can also have a passed a function that does all the work:

```
In [61]: data['food'].map(lambda x: meat_to_animal[x.lower()])
Out[61]:
0     pig
1     pig
2     pig
3     cow
4     cow
5     pig
6     cow
7     pig
8   salmon
Name: food, dtype: object
```

RASMUSSEN UNIVERSITY

29

Replacing Values

Filling in missing data with the fillna method is a special case of more general value replacement. As you've already seen, map can be used to modify a subset of values in an object but replace provides a simpler and more flexible way to do so. Let's consider this Series:

```
In [62]: data = pd.Series([1., -999., 2., -999., -1000., 3.])

In [63]: data
Out[63]:
0      1.0
1   -999.0
2      2.0
3   -999.0
4  -1000.0
5      3.0
dtype: float64
```

RASMUSSEN UNIVERSITY

30

Replacing Values

The -999 values might be sentinel values for missing data. To replace these with NA values that pandas understands, we can use replace, producing a new Series (unless you pass inplace=True):

```
In [64]: data.replace(-999, np.nan)
Out[64]:
0      1.0
1      NaN
2      2.0
3      NaN
4  -1000.0
5      3.0
dtype: float64
```

31

Replacing Values

If you want to replace multiple values at once, you instead pass a list and then the substitute value:

```
In [65]: data.replace([-999, -1000], np.nan)
Out[65]:
0    1.0
1    NaN
2    2.0
3    NaN
4    NaN
5    3.0
dtype: float64
```

32

Replacing Values

The argument passed can also be a dict:

```
In [67]: data.replace({-999: np.nan, -1000: 0})
Out[67]:
0    1.0
1    NaN
2    2.0
3    NaN
4    0.0
5    3.0
dtype: float64
```

33

Replacing Values

To use a different replacement for each value, pass a list of substitutes:

```
In [66]: data.replace([-999, -1000], [np.nan, 0])
Out[66]:
0    1.0
1    NaN
2    2.0
3    NaN
4    0.0
5    3.0
dtype: float64
```

The `data.replace` method is distinct from `data.str.replace`, which performs string substitution element-wise.

34

JSON DATA IN PYTHON

What is JSON?

- JSON is a standard format for data exchange based on JavaScript programming language
- JSON stands for JavaScript Object Notation.
- JSON is in string or text format.

The syntax of JSON: JSON is written as key and value pair.
{ "Key": "Value", "Key": "Value", }

JSON is very similar to Python dictionary. Python supports JSON, and it has an inbuilt library as a JSON.

35

JSON DATA IN PYTHON

JSON is built on two structures:

- A collection of name/value pairs. This is realized as an object, record, dictionary, hash table, keyed list, or associative array.
- An ordered list of values. This is realized as an array, vector, list, or sequence.

There are a couple of packages that support JSON in Python

- metamagic.json
- Jyson
- Simplejson
- Yajl-Py
- Ultrajson
- json

36

JSON DATA IN PYTHON

```
{
        "article": [
        {
                "id":"01",
                "language": "JSON",
                "edition": "first",
                "author": "Derrick Mwiti"
        },

        {
                "id":"02",
                "language": "Python",
                "edition": "second",
                "author": "Derrick Mwiti"
        }
    ],
    "blog":[
    {
        "name": "Datacamp",
        "URL":"datacamp.com"
    }
    ]
}
```

RASMUSSEN UNIVERSITY

37

JSON DATA IN PYTHON

Converting JSON to Python Objects

We can parse the above JSON string using json.loads() method from the json module. The result is a Python dictionary.

RASMUSSEN UNIVERSITY

38

JSON DATA IN PYTHON

```
import json
my_json_string = """{
        "article": [
        {
                "id":"01",
                "language": "JSON",
                "edition": "first",
                "author": "Derrick Mwiti"
        },

        {
                "id":"02",
                "language": "Python",
                "edition": "second",
                "author": "Derrick Mwiti"
        }
    ],
    "blog":[
    {
        "name": "Datacamp",
        "URL":"datacamp.com"
    }
    ]
}
"""
to_python = json.loads(my_json_string)
```

RASMUSSEN UNIVERSITY

39

JSON DATA IN PYTHON

to_python['blog']

[{'URL': 'Rasmussen.edu', 'name': 'Rasmussen'}]

RASMUSSEN UNIVERSITY

40

JSON DATA IN PYTHON

Converting Python Objects to JSON

json.dumps() – convert Python Objects to JSON

blog = {'URL': 'Rasmussen.edu', 'name': 'Rasmussen'}

to_json = json.dumps(blog)

to_json

'{'URL': 'Rasmussen.edu', 'name': 'Rasmussen'}'

RASMUSSEN UNIVERSITY

41

JSON DATA IN PYTHON

COMPARISON OF DATA TYPES IN PYTHON and JSON

Python	JSON
dict	Object
list	Array
tuple	Array
str	String
int	Number
float	Number
True	true
False	false
None	null

Next you will see how to convert various Python objects to different JSON data types

RASMUSSEN UNIVERSITY

42

JSON DATA IN PYTHON

- Python Tuple to JSON Array
  Tuple_example = ‘Mango’, ‘Banana’, ‘Apple’;
  print(json.dumps(tuple_example));
  [“Mango”, “Banana”, “Apple”]
- Python Tuple to JSON Array
  list_example = [“Mango”, 1, 3, 6, “Oranges”];
  print(json.dumps(list_example));
  [“Mango”, 1, 3, 6, “Oranges”]

RASMUSSEN UNIVERSITY

43

JSON DATA IN PYTHON

- Python String to JSON String
  string_example = “This is a cool example.”
  print(json.dumps(string_example))
  “This is a cool example.”
- Python Boolean Values to JSON Boolean Values
  Boolean_value = False
  print(json.dumps(Boolean_value));
  false

RASMUSSEN UNIVERSITY

44

JSON DATA IN PYTHON

Writing a JSON File

json module allows us to write JSON data into a JSON file
JSON files are saved with the .json extension

```
my_json_string = """{
        "article": [
        {
                "id":"01",
                "language": "JSON",
                "edition": "first",
                "author": "Derrick Mwiti“
        },
```

RASMUSSEN UNIVERSITY

45

JSON DATA IN PYTHON

```
        {
                "id":"02",
                "language": "Python",
                "edition": "second",
                "author": "Derrick Mwiti"
        }
    ],
    "blog":[
    {
        "name": “Rasmussen",
        "URL":“Rasmussen.edu"
    }
    ]
}
“”””
with open(‘test_file.json’, ‘w’) as file:
json.dump(my_json_string)
```

RASMUSSEN UNIVERSITY

46

JSON DATA IN PYTHON

Reading a JSON File

json module allows us to read JSON file
json.load is used to load in files

```
with open(‘test_file.json’, ‘r’) as j:
    json_data = json.load(j)
    print(json_data)

my_json_string = “”””{
        "article": [
        {
                "id":"01",
                "language": "JSON",
                "edition": "first",
                "author": "Derrick Mwiti“
        },
```

RASMUSSEN UNIVERSITY

47

JSON DATA IN PYTHON

```
        {
                "id":"02",
                "language": "Python",
                "edition": "second",
                "author": "Derrick Mwiti"
        }
    ],
    "blog":[
    {
        "name": “Rasmussen",
        "URL":“Rasmussen.edu"
    }
    ]
}
```

RASMUSSEN UNIVERSITY

48

JSON DATA IN PYTHON

json.load vs json.loads

json.load is used when loading a file
json.loads(load string) is used when loading a string

json.dump vs json.dumps

json.dump is used when we want to dump JSON into a file
json.dumps(dump string) is used when we need the JSON data as a string for parsing or printing

RASMUSSEN UNIVERSITY

49

JSON DATA IN PYTHON

Handling JSON Data in Data Science

Need to load data that is in JSON format

- Pandas provides .read_json that enables us to do this
- Data is loaded – convert it into a dataframe using the pandas.DataFrame attribute

Example:

```
import pandas as pd
data = pd.read_json(https://api.github.com/users)
df = pd.DataFrame(data)
df
```

RASMUSSEN UNIVERSITY

50

JSON HELPS

https://www.guru99.com/python-json.html

https://www.tutorialspoint.com/numpy/index.htm

RASMUSSEN UNIVERSITY

51

SEE PYTHON FOR DATA ANALYSIS SLIDES FOR ADDITIONAL INFORMATION

RASMUSSEN UNIVERSITY

52

MODULE 03 DISCUSSION FORUM GRADING RUBRIC

	LEVELS OF ACHIEVEMENT			
Criteria	Advanced	Proficient	Developing	Limited
Quality of Comments Total Points: 4	90 to 100 % Timely and appropriate comments as defined by the instructor. Thoughtful and reflective, responds respectfully to other students' remarks, provokes questions and comments from the group.	75 to 89 % Volunteers comments, most are appropriate and reflect some thoughtfulness as defined by the instructor. Leads to other questions or remarks from students.	60 to 74 % Volunteers comments but lacks depth, may or may not lead to other questions from students. Off topic or irrelevant contributions.	0 to 59 % Did not participate.
Interaction with Course Resources Total Points: 3	90 to 100 % Clear reference to text being discussed and connects it to other text of reference points from previous readings and discussions.	75 to 89 % Has done the reading with some thoroughness, may lack some detail or critical insight.	60 to 74 % Has done the reading; lacks thoroughness or understanding or insight.	0 to 59 % Did not participate.
Active Listening and Participation Total Points: 3	90 to 100 % Comments clearly demonstrate respect and attentiveness to others; creatively builds on others' comments to offer insights. Directly answers the question(s) asked AND provides additional insights. Exceptional level of interaction as defined by the instructor.	75 to 89 % Shows consistency in responding to the comments of others. Stays focused on the stream of discussion rather than own ideas. Directly answers the questions(s) asked. Appropriate level of interaction as defined by the instructor.	60 to 74 % Does not stay focused on others' comments (too busy formulating own) or loses continuity of discussion. Inconsistent in tracking discussion stream. Indirectly answers the assigned question(s). Poor level of interaction as defined by the instructor.	0 to 59 % Did not participate

RASMUSSEN UNIVERSITY

53

MODULE 03 DISCUSSION FORUM - DATA INTEGRITY ISSUES

For this discussion, describe two different ways to correct data integrity issues with the Python script and then compare differences between the two approaches.

Your initial post should be a minimum of at least two fully-formed, well-thought-out scholarly paragraphs (blocks of code examples and graphics are considered additional information beyond the two required paragraphs but are encouraged when appropriate).

RASMUSSEN UNIVERSITY

54

MODULE 03 DISCUSSION FORUM - DATA INTEGRITY ISSUES

For your reply, choose two other student responses and provide additional insights to each of them that add value to their posting. The reply should be at least one fully-formed, well-thought-out scholarly paragraph (blocks of code examples are considered additional information beyond the required paragraph but are encouraged when appropriate). A simple "I agree" or "I disagree" type of post is unacceptable.

Due dates for your initial and response posts can be found by checking the *Course Syllabus* and *Course Calendar.*

RASMUSSEN UNIVERSITY

55

MODULE 03 COURSE PROJECT: GRADING RUBRIC

Criteria	Points
A correct Python script is attached.	50
Output screenshots are attached.	50
Total	100

RASMUSSEN UNIVERSITY

56

MODULE 03 COURSE PROJECT: VALUES

Please download the Noshowappointments.csv dataset from the **Course Files** page.

Then write a Python script that can perform following tasks.

- Step 1: Find out all the occurrences where No-Show record is either blank or Missing
- Step 2: Replace it with NaN values
- Step 3: Ignore NaN values from the calculation and determine PatientID, Count(No-Show=Yes)
- Step 4: Ignore NaN values from the calculation and determine Calculate Neighbourhood, count(No-Shows=Yes)

For your submission, include the following:
•Attach a Python script.
•An output screenshot.

RASMUSSEN UNIVERSITY

57

MODULE 03 COURSE PROJECT: VALUES

Submit these files as a single zipped ".zip" file to the drop box below. Please check the **Course Calendar** for specific due dates.

Note: For help with zipping or compressing your files, visit the How do you zip files? Answers page.

The name of the file should be your first initial and last name, followed by an underscore and the name of the assignment, and an underscore and the date. (Mac users, please remember to append the ".zip" extension to the filename.) An example is shown below:

Jstudent_exampleproblem_101504

RASMUSSEN UNIVERSITY

58

MODULE 03 ASSIGNMENT: Grading Rubric

Criteria	Points
A correct Python script is attached.	50
Output screenshots are attached.	50
Total	100

RASMUSSEN UNIVERSITY

59

MODULE 03 ASSIGNMENT: Zip Codes

You are working as an analytics developer for the United States Government. You need to prepare a JSON file that shows all zip codes with the city name for the entire country.

Please download the current JSON file from the **Course Files** folder. You need to perform following tasks.

- Step 1: Use Pandas to read the JSON file.
- Step 2: Analyze the data.
- Step 3: Update the zip code = blank or null to something meaningful such as **Not Available**.
- Step 4: Write the JSON file using the Pandas.

RASMUSSEN UNIVERSITY

60

MODULE 03 ASSIGNMENT: Zip Codes

You need to submit following things as a part of your submission.

- Python Script
- Output Screenshot
- Modified JSON file
- Submit these files as a single zipped ".zip" file to the drop box below. Please check the **Course Calendar** for specific due dates.

Note: For help with zipping or compressing your files, visit the How do you zip files? Answers page.

RASMUSSEN UNIVERSITY

61

MODULE 03 ASSIGNMENT: Zip Codes

The name of the file should be your first initial and last name, followed by an underscore and the name of the assignment, and an underscore and the date. (Mac users, please remember to append the ".zip" extension to the filename.) An example is shown below:

Jstudent_exampleproblem_101504

RASMUSSEN UNIVERSITY

62