labassignment3

February 2, 2023

1 Lab Assignment 3: How to Load, Convert, and Write JSON Files in Python

1.1 DS 6001: Practice and Application of Data Science

1.1.1 Instructions

Please answer the following questions as completely as possible using text, code, and the results of code as needed. Format your answers in a Jupyter notebook. To receive full credit, make sure you address every part of the problem, and make sure your document is formatted in a clean and professional way.

1.2 Problem 0

Import the following libraries:

[1]:
```
import numpy as np
import pandas as pd
import requests
import json
import sys
sys.tracebacklimit = 0 # turn off the error tracebacks
```

1.3 Problem 1

JSON and CSV are both text-based formats for the storage of data. It's possible to open either one in a plain text editor. Given this similarity, why does a CSV file usually take less memory than a JSON formatted file for the same data? Under what conditions could a JSON file be smaller in memory than a CSV file for the same data? (2 points)

Consider a CSV file with the following contents:

[2]:
```
dictionary = {}
for i in range(0, 100):
    column_name = 'c' + str(i)
    column_values = 100 * [np.nan]
    column_values[i] = i
    dictionary[column_name] = column_values
data_frame = pd.DataFrame(dictionary)
csv_data = data_frame.to_csv(index = False, line_terminator = '\n')
```

```
print(csv_data[0:699])
print('characters: ' + str(len(csv_data)))
```

```
c0,c1,c2,c3,c4,c5,c6,c7,c8,c9,c10,c11,c12,c13,c14,c15,c16,c17,c18,c19,c20,c21,c2
2,c23,c24,c25,c26,c27,c28,c29,c30,c31,c32,c33,c34,c35,c36,c37,c38,c39,c40,c41,c4
2,c43,c44,c45,c46,c47,c48,c49,c50,c51,c52,c53,c54,c55,c56,c57,c58,c59,c60,c61,c6
2,c63,c64,c65,c66,c67,c68,c69,c70,c71,c72,c73,c74,c75,c76,c77,c78,c79,c80,c81,c8
2,c83,c84,c85,c86,c87,c88,c89,c90,c91,c92,c93,c94,c95,c96,c97,c98,c99
0.0,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
,,,,,,,,,,,,,,,,,,,,
,1.0,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
,,,,,,,,,,,,,,,,,,,,
,,2.0,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
,,,,,,,,,,,,,,,,,,,,

characters: 10780
```

Such a data set may be represented by the following JSON elements:

[3]:
```
json_element = data_frame.to_json(orient = "records")
print(json_element[0:1091])
print('characters: ' + str(len(json_element)))

json_element = data_frame.to_json(orient = "columns")
print(json_element[0:1091])
print('characters: ' + str(len(json_element)))

json_element = data_frame.to_json(orient = "split")
print(json_element[0:1091])
print('characters: ' + str(len(json_element)))

json_element = data_frame.to_json(orient = "index")
print(json_element[0:1091])
print('characters: ' + str(len(json_element)))

json_element = data_frame.to_json(orient = "values")
print(json_element[0:1091])
print('characters: ' + str(len(json_element)))
```

```
[{"c0":0.0,"c1":null,"c2":null,"c3":null,"c4":null,"c5":null,"c6":null,"c7":null
,"c8":null,"c9":null,"c10":null,"c11":null,"c12":null,"c13":null,"c14":null,"c15
":null,"c16":null,"c17":null,"c18":null,"c19":null,"c20":null,"c21":null,"c22":n
ull,"c23":null,"c24":null,"c25":null,"c26":null,"c27":null,"c28":null,"c29":null
,"c30":null,"c31":null,"c32":null,"c33":null,"c34":null,"c35":null,"c36":null,"c
37":null,"c38":null,"c39":null,"c40":null,"c41":null,"c42":null,"c43":null,"c44"
:null,"c45":null,"c46":null,"c47":null,"c48":null,"c49":null,"c50":null,"c51":nu
ll,"c52":null,"c53":null,"c54":null,"c55":null,"c56":null,"c57":null,"c58":null,
"c59":null,"c60":null,"c61":null,"c62":null,"c63":null,"c64":null,"c65":null,"c6
6":null,"c67":null,"c68":null,"c69":null,"c70":null,"c71":null,"c72":null,"c73":
```

null,"c74":null,"c75":null,"c76":null,"c77":null,"c78":null,"c79":null,"c80":nul
l,"c81":null,"c82":null,"c83":null,"c84":null,"c85":null,"c86":null,"c87":null,"
c88":null,"c89":null,"c90":null,"c91":null,"c92":null,"c93":null,"c94":null,"c95
":null,"c96":null,"c97":null,"c98":null,"c99":null}
characters: 109191
{"c0":{"0":0.0,"1":null,"2":null,"3":null,"4":null,"5":null,"6":null,"7":null,"8
":null,"9":null,"10":null,"11":null,"12":null,"13":null,"14":null,"15":null,"16"
:null,"17":null,"18":null,"19":null,"20":null,"21":null,"22":null,"23":null,"24"
:null,"25":null,"26":null,"27":null,"28":null,"29":null,"30":null,"31":null,"32"
:null,"33":null,"34":null,"35":null,"36":null,"37":null,"38":null,"39":null,"40"
:null,"41":null,"42":null,"43":null,"44":null,"45":null,"46":null,"47":null,"48"
:null,"49":null,"50":null,"51":null,"52":null,"53":null,"54":null,"55":null,"56"
:null,"57":null,"58":null,"59":null,"60":null,"61":null,"62":null,"63":null,"64"
:null,"65":null,"66":null,"67":null,"68":null,"69":null,"70":null,"71":null,"72"
:null,"73":null,"74":null,"75":null,"76":null,"77":null,"78":null,"79":null,"80"
:null,"81":null,"82":null,"83":null,"84":null,"85":null,"86":null,"87":null,"88"
:null,"89":null,"90":null,"91":null,"92":null,"93":null,"94":null,"95":null,"96"
:null,"97":null,"98":null,"99":null},"c1":{"0":null,"1":1.0,"2":null,"3":null,"4
":null,"5":null,"6":null,"7":null,"8":null,"9":null
characters: 99781
{"columns":["c0","c1","c2","c3","c4","c5","c6","c7","c8","c9","c10","c11","c12",
"c13","c14","c15","c16","c17","c18","c19","c20","c21","c22","c23","c24","c25","c
26","c27","c28","c29","c30","c31","c32","c33","c34","c35","c36","c37","c38","c39
","c40","c41","c42","c43","c44","c45","c46","c47","c48","c49","c50","c51","c52",
"c53","c54","c55","c56","c57","c58","c59","c60","c61","c62","c63","c64","c65","c
66","c67","c68","c69","c70","c71","c72","c73","c74","c75","c76","c77","c78","c79
","c80","c81","c82","c83","c84","c85","c86","c87","c88","c89","c90","c91","c92",
"c93","c94","c95","c96","c97","c98","c99"],"index":[0,1,2,3,4,5,6,7,8,9,10,11,12
,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,3
9,40,41,42,43,44,45,46,47,48,49,50,51,52,53,54,55,56,57,58,59,60,61,62,63,64,65,
66,67,68,69,70,71,72,73,74,75,76,77,78,79,80,81,82,83,84,85,86,87,88,89,90,91,92
,93,94,95,96,97,98,99],"data":[[0.0,null,null,null,null,null,null,null,null,null
,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null
,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null
,null,null,null,null,null,null,null,null,null,null,
characters: 51102
{"0":{"c0":0.0,"c1":null,"c2":null,"c3":null,"c4":null,"c5":null,"c6":null,"c7":
null,"c8":null,"c9":null,"c10":null,"c11":null,"c12":null,"c13":null,"c14":null,
"c15":null,"c16":null,"c17":null,"c18":null,"c19":null,"c20":null,"c21":null,"c2
2":null,"c23":null,"c24":null,"c25":null,"c26":null,"c27":null,"c28":null,"c29":
null,"c30":null,"c31":null,"c32":null,"c33":null,"c34":null,"c35":null,"c36":nul
l,"c37":null,"c38":null,"c39":null,"c40":null,"c41":null,"c42":null,"c43":null,"
c44":null,"c45":null,"c46":null,"c47":null,"c48":null,"c49":null,"c50":null,"c51
":null,"c52":null,"c53":null,"c54":null,"c55":null,"c56":null,"c57":null,"c58":n
ull,"c59":null,"c60":null,"c61":null,"c62":null,"c63":null,"c64":null,"c65":null
,"c66":null,"c67":null,"c68":null,"c69":null,"c70":null,"c71":null,"c72":null,"c
73":null,"c74":null,"c75":null,"c76":null,"c77":null,"c78":null,"c79":null,"c80"
:null,"c81":null,"c82":null,"c83":null,"c84":null,"c85":null,"c86":null,"c87":nu
ll,"c88":null,"c89":null,"c90":null,"c91":null,"c92":null,"c93":null,"c94":null,

```
"c95":null,"c96":null,"c97":null,"c98":null,"c99":n
characters: 109681
[[0.0,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null
,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null
,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null
,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null
,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null
,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null
,null,null,null,null],[null,1.0,null,null,null,null,null,null,null,null,null,nul
l,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,nul
l,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,nul
l,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,nul
l,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,nul
l,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,nul
l,null,null,null,null,null,null,null,null],[null,null,2.0,null,null,null,null,nu
ll,null,null,null,null,null,null,null,null,null,nul
characters: 50191
```

A CSV file usually takes less memory than a JSON formatted file for the same data because the number of characters in the CSV file is usually less than the number of characters in each JSON element for the same data.

That being said, consider the following JSON elements:

[4]:
```python
def convert_to_JSON_string(data_frame, orientation):
    if orientation == 'records':
        JSON_array = []
        for i in range(0, len(data_frame.index)):
            JSON_object = {}
            for column_name in data_frame.columns:
                if not pd.isnull(data_frame.at[i, column_name]):
                    JSON_object[column_name] = data_frame.at[i, column_name]
            if JSON_object:
                JSON_array.append(JSON_object)
        return str(JSON_array).replace(' ', '')
    if orientation == 'columns':
        JSON_object = {}
        for column_name in data_frame.columns:
            JSON_subobject = {}
            for i in range(0, len(data_frame.index)):
                if not pd.isnull(data_frame.at[i, column_name]):
                    JSON_subobject[str(i)] = data_frame.at[i, column_name]
            if JSON_subobject:
                JSON_object[column_name] = JSON_subobject
        return str(JSON_object).replace(' ', '')
    if orientation == 'split':
        JSON_object = {}
        JSON_object['columns'] = data_frame.columns.to_list()
```

```
        JSON_object['index'] = data_frame.index.to_list()
        JSON_array = []
        for i in range(0, len(data_frame.index)):
            JSON_subarray = []
            for column in data_frame.columns:
                if not pd.isnull(data_frame.at[i, column]):
                    JSON_subarray.append(data_frame.at[i, column])
                else:
                    JSON_subarray.append('null')
            JSON_array.append(JSON_subarray)
        JSON_object['data'] = JSON_array
        return str(JSON_object).replace(' ', '')
    if orientation == 'index':
        JSON_object = {}
        for i in range(0, len(data_frame.index)):
            JSON_subobject = {}
            for column_name in data_frame.columns:
                if not pd.isnull(data_frame.at[i, column_name]):
                    JSON_subobject[column_name] = data_frame.at[i, column_name]
            if JSON_subobject:
                JSON_object[str(i)] = JSON_subobject
        return str(JSON_object).replace(' ', '')
    if orientation == 'values':
        JSON_array = []
        for i in range(0, len(data_frame.index)):
            JSON_subarray = []
            for column in data_frame.columns:
                if not pd.isnull(data_frame.at[i, column]):
                    JSON_subarray.append(data_frame.at[i, column])
                else:
                    JSON_subarray.append('null')
            JSON_array.append(JSON_subarray)
        return str(JSON_array).replace(' ', '')
    return None
```

[5]:
```
json_element = convert_to_JSON_string(data_frame = data_frame, orientation =␣
 ↪'records')
print(json_element[0:1091])
print('characters: ' + str(len(json_element)))

json_element = convert_to_JSON_string(data_frame = data_frame, orientation =␣
 ↪'columns')
print(json_element[0:1091])
print('characters: ' + str(len(json_element)))

json_element = convert_to_JSON_string(data_frame = data_frame, orientation =␣
 ↪'split')
```

```
print(json_element[0:1091])
print('characters: ' + str(len(json_element)))

json_element = convert_to_JSON_string(data_frame = data_frame, orientation =␣
 ↪'index')
print(json_element[0:1091])
print('characters: ' + str(len(json_element)))

json_element = convert_to_JSON_string(data_frame = data_frame, orientation =␣
 ↪'values')
print(json_element[0:1091])
print('characters: ' + str(len(json_element)))
```

```
[{'c0':0.0},{'c1':1.0},{'c2':2.0},{'c3':3.0},{'c4':4.0},{'c5':5.0},{'c6':6.0},{'
c7':7.0},{'c8':8.0},{'c9':9.0},{'c10':10.0},{'c11':11.0},{'c12':12.0},{'c13':13.
0},{'c14':14.0},{'c15':15.0},{'c16':16.0},{'c17':17.0},{'c18':18.0},{'c19':19.0}
,{'c20':20.0},{'c21':21.0},{'c22':22.0},{'c23':23.0},{'c24':24.0},{'c25':25.0},{
'c26':26.0},{'c27':27.0},{'c28':28.0},{'c29':29.0},{'c30':30.0},{'c31':31.0},{'c
32':32.0},{'c33':33.0},{'c34':34.0},{'c35':35.0},{'c36':36.0},{'c37':37.0},{'c38
':38.0},{'c39':39.0},{'c40':40.0},{'c41':41.0},{'c42':42.0},{'c43':43.0},{'c44':
44.0},{'c45':45.0},{'c46':46.0},{'c47':47.0},{'c48':48.0},{'c49':49.0},{'c50':50
.0},{'c51':51.0},{'c52':52.0},{'c53':53.0},{'c54':54.0},{'c55':55.0},{'c56':56.0
},{'c57':57.0},{'c58':58.0},{'c59':59.0},{'c60':60.0},{'c61':61.0},{'c62':62.0},
{'c63':63.0},{'c64':64.0},{'c65':65.0},{'c66':66.0},{'c67':67.0},{'c68':68.0},{'
c69':69.0},{'c70':70.0},{'c71':71.0},{'c72':72.0},{'c73':73.0},{'c74':74.0},{'c7
5':75.0},{'c76':76.0},{'c77':77.0},{'c78':78.0},{'c79':79.0},{'c80':80.0},{'c81'
:81.0},{'c82':82.0},{'c83':83.0},{'c84':84.0},{'c85
characters: 1281
{'c0':{'0':0.0},'c1':{'1':1.0},'c2':{'2':2.0},'c3':{'3':3.0},'c4':{'4':4.0},'c5'
:{'5':5.0},'c6':{'6':6.0},'c7':{'7':7.0},'c8':{'8':8.0},'c9':{'9':9.0},'c10':{'1
0':10.0},'c11':{'11':11.0},'c12':{'12':12.0},'c13':{'13':13.0},'c14':{'14':14.0}
,'c15':{'15':15.0},'c16':{'16':16.0},'c17':{'17':17.0},'c18':{'18':18.0},'c19':{
'19':19.0},'c20':{'20':20.0},'c21':{'21':21.0},'c22':{'22':22.0},'c23':{'23':23.
0},'c24':{'24':24.0},'c25':{'25':25.0},'c26':{'26':26.0},'c27':{'27':27.0},'c28'
:{'28':28.0},'c29':{'29':29.0},'c30':{'30':30.0},'c31':{'31':31.0},'c32':{'32':3
2.0},'c33':{'33':33.0},'c34':{'34':34.0},'c35':{'35':35.0},'c36':{'36':36.0},'c3
7':{'37':37.0},'c38':{'38':38.0},'c39':{'39':39.0},'c40':{'40':40.0},'c41':{'41'
:41.0},'c42':{'42':42.0},'c43':{'43':43.0},'c44':{'44':44.0},'c45':{'45':45.0},'
c46':{'46':46.0},'c47':{'47':47.0},'c48':{'48':48.0},'c49':{'49':49.0},'c50':{'5
0':50.0},'c51':{'51':51.0},'c52':{'52':52.0},'c53':{'53':53.0},'c54':{'54':54.0}
,'c55':{'55':55.0},'c56':{'56':56.0},'c57':{'57':57.0},'c58':{'58':58.0},'c59':{
'59':59.0},'c60':{'60':60.0},'c61':{'61':61.0},'c62
characters: 1771
{'columns':['c0','c1','c2','c3','c4','c5','c6','c7','c8','c9','c10','c11','c12',
'c13','c14','c15','c16','c17','c18','c19','c20','c21','c22','c23','c24','c25','c
26','c27','c28','c29','c30','c31','c32','c33','c34','c35','c36','c37','c38','c39
','c40','c41','c42','c43','c44','c45','c46','c47','c48','c49','c50','c51','c52',
```

```
'c53','c54','c55','c56','c57','c58','c59','c60','c61','c62','c63','c64','c65','c
66','c67','c68','c69','c70','c71','c72','c73','c74','c75','c76','c77','c78','c79
','c80','c81','c82','c83','c84','c85','c86','c87','c88','c89','c90','c91','c92',
'c93','c94','c95','c96','c97','c98','c99'],'index':[0,1,2,3,4,5,6,7,8,9,10,11,12
,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,3
9,40,41,42,43,44,45,46,47,48,49,50,51,52,53,54,55,56,57,58,59,60,61,62,63,64,65,
66,67,68,69,70,71,72,73,74,75,76,77,78,79,80,81,82,83,84,85,86,87,88,89,90,91,92
,93,94,95,96,97,98,99],'data':[[0.0,'null','null','null','null','null','null','n
ull','null','null','null','null','null','null','null','null','null','null','null
','null','null','null','null','null','null','null',
characters: 70902
{'0':{'c0':0.0},'1':{'c1':1.0},'2':{'c2':2.0},'3':{'c3':3.0},'4':{'c4':4.0},'5':
{'c5':5.0},'6':{'c6':6.0},'7':{'c7':7.0},'8':{'c8':8.0},'9':{'c9':9.0},'10':{'c1
0':10.0},'11':{'c11':11.0},'12':{'c12':12.0},'13':{'c13':13.0},'14':{'c14':14.0}
,'15':{'c15':15.0},'16':{'c16':16.0},'17':{'c17':17.0},'18':{'c18':18.0},'19':{'
c19':19.0},'20':{'c20':20.0},'21':{'c21':21.0},'22':{'c22':22.0},'23':{'c23':23.
0},'24':{'c24':24.0},'25':{'c25':25.0},'26':{'c26':26.0},'27':{'c27':27.0},'28':
{'c28':28.0},'29':{'c29':29.0},'30':{'c30':30.0},'31':{'c31':31.0},'32':{'c32':3
2.0},'33':{'c33':33.0},'34':{'c34':34.0},'35':{'c35':35.0},'36':{'c36':36.0},'37
':{'c37':37.0},'38':{'c38':38.0},'39':{'c39':39.0},'40':{'c40':40.0},'41':{'c41'
:41.0},'42':{'c42':42.0},'43':{'c43':43.0},'44':{'c44':44.0},'45':{'c45':45.0},'
46':{'c46':46.0},'47':{'c47':47.0},'48':{'c48':48.0},'49':{'c49':49.0},'50':{'c5
0':50.0},'51':{'c51':51.0},'52':{'c52':52.0},'53':{'c53':53.0},'54':{'c54':54.0}
,'55':{'c55':55.0},'56':{'c56':56.0},'57':{'c57':57.0},'58':{'c58':58.0},'59':{'
c59':59.0},'60':{'c60':60.0},'61':{'c61':61.0},'62'
characters: 1771
[[0.0,'null','null','null','null','null','null','null','null','null','null','nul
l','null','null','null','null','null','null','null','null','null','null','null',
'null','null','null','null','null','null','null','null','null','null','null','nu
ll','null','null','null','null','null','null','null','null','null','null','null'
,'null','null','null','null','null','null','null','null','null','null','null','n
ull','null','null','null','null','null','null','null','null','null','null','null
','null','null','null','null','null','null','null','null','null','null','null','
null','null','null','null','null','null','null','null','null','null','null','nul
l','null','null','null','null','null','null','null','null'],['null',1.0,'null','
null','null','null','null','null','null','null','null','null','null','null','nul
l','null','null','null','null','null','null','null','null','null','null','null',
'null','null','null','null','null','null','null','null','null','null','null','nu
ll','null','null','null','null','null','null','null','null','null','null','null'
,'null','null','null','null','null','null','null','null','null','null','null','
characters: 69991
```


The number of characters in some of our JSON elements is less than the number of characters in our CSV data.

A JSON file can be smaller in memory than a CSV file when there are many data, many missing values, and omission of missing values from the JSON file.

1.4 Problem 2

NASA has a dataset of all meteorites that have fallen to Earth between the years A.D. 860 and 2013. The data contain the name of each meteorite, along with the coordinates of the place where the meteorite hit, the mass of the meteorite, and the date of the collison. The data is stored as a JSON here: https://data.nasa.gov/resource/y77d-th95.json

Look at the data in your web-browser and explain which strategy for loading the JSON into Python makes the most sense and why.

Then write and run the code that will work for loading the data into Python. (2 points)

Looking at the data in Chrome, I observe that there is nesting but no metadata, and that the data set is a JSON array. Thus, I may:

1. Use `requests.get` to download the raw JSON data,
2. Use `json.loads` on the text attribute of the output from step 1 to register the the data as a list in Python, and
3. Use the `pd.json_normalize` function on the list that is the output of step 2.

[6]:
```
url = 'https://data.nasa.gov/resource/y77d-th95.json'
response = requests.get(url)
response_body_as_string = response.text
list_of_meteorites_that_have_fallen_to_Earth = json.
 ↪loads(response_body_as_string)
data_frame_of_meteorites_that_have_fallen_to_Earth = pd.
 ↪json_normalize(list_of_meteorites_that_have_fallen_to_Earth)
data_frame_of_meteorites_that_have_fallen_to_Earth
```

[6]:

	name	id	nametype	recclass	mass	fall	\
0	Aachen	1	Valid	L5	21	Fell	
1	Aarhus	2	Valid	H6	720	Fell	
2	Abee	6	Valid	EH4	107000	Fell	
3	Acapulco	10	Valid	Acapulcoite	1914	Fell	
4	Achiras	370	Valid	L6	780	Fell	
..	…	…	…	…	…	…	
995	Tirupati	24009	Valid	H6	230	Fell	
996	Tissint	54823	Valid	Martian (shergottite)	7000	Fell	
997	Tjabe	24011	Valid	H6	20000	Fell	
998	Tjerebon	24012	Valid	L5	16500	Fell	
999	Tomakovka	24019	Valid	LL6	600	Fell	

	year	reclat	reclong	geolocation.type	\
0	1880-01-01T00:00:00.000	50.775000	6.083330	Point	
1	1951-01-01T00:00:00.000	56.183330	10.233330	Point	
2	1952-01-01T00:00:00.000	54.216670	-113.000000	Point	
3	1976-01-01T00:00:00.000	16.883330	-99.900000	Point	
4	1902-01-01T00:00:00.000	-33.166670	-64.950000	Point	
..	…	…	…	…	
995	1934-01-01T00:00:00.000	13.633330	79.416670	Point	

```
996  2011-01-01T00:00:00.000   29.481950    -7.611230            Point
997  1869-01-01T00:00:00.000   -7.083330   111.533330            Point
998  1922-01-01T00:00:00.000   -6.666670   106.583330            Point
999  1905-01-01T00:00:00.000   47.850000    34.766670            Point

    geolocation.coordinates :@computed_region_cbhk_fwbd  \
0         [6.08333, 50.775]                          NaN
1       [10.23333, 56.18333]                         NaN
2           [-113, 54.21667]                         NaN
3          [-99.9, 16.88333]                         NaN
4        [-64.95, -33.16667]                         NaN
..                       …                          …
995     [79.41667, 13.63333]                         NaN
996     [-7.61123, 29.48195]                         NaN
997    [111.53333, -7.08333]                         NaN
998    [106.58333, -6.66667]                         NaN
999        [34.76667, 47.85]                         NaN

    :@computed_region_nnqa_25f4
0                           NaN
1                           NaN
2                           NaN
3                           NaN
4                           NaN
..                          …
995                         NaN
996                         NaN
997                         NaN
998                         NaN
999                         NaN

[1000 rows x 13 columns]
```

1.5 Problem 3

The textbook chapter for this module shows, as an example, how to pull data in JSON format from Reddit's top 25 posts on /r/popular. The steps outlined there pull all of the features in the data into the dataframe, resulting in a dataframe with 172 columns.

If we only wanted a few features, then looping across elements of the JSON list itself and extracting only the data we want may be a more efficient approach.

Use looping - and not `pd.read_json()` or `pd.json_normalize()` - to create a dataframe with 25 rows (one for each of the top 25 posts), and only columns for `subreddit`, `title`, `ups`, and `created_utc`. The JSON file exists at http://www.reddit.com/r/popular/top.json, and don't forget to specify `headers = {'User-agent': 'DS6001'}` within `requests.get()`. (3 points)

[8]:
```
url = 'http://www.reddit.com/r/popular/top.json'
the_headers = {'User-agent': 'DS6001'}
response = requests.get(url, headers = the_headers)
response_body_as_string = response.text
body = json.loads(response_body_as_string)
data = body.get('data')
children = data.get('children')
data_frame = pd.DataFrame({'subreddit': [], 'title': [], 'ups': [],␣
 ↪'created_utc': []})
for i in range(0, len(children)):
    child = children[i]
    data = child.get('data')
    list_of_subreddit_title_ups_and_created_utc = [data.get('subreddit'), data.
 ↪get('title'), data.get('ups'), data.get('created_utc')]
    data_frame.loc[len(data_frame.index)] =␣
 ↪list_of_subreddit_title_ups_and_created_utc
data_frame
```

[8]:

	subreddit	title	ups	created_utc
0	nextfuckinglevel	The man climbed out of his eighth floor apartm…	100132	1.675270e+09
1	MadeMeSmile	Last January I started my sobriety and health …		
2	meirl	meirl		
3	gaming	Sonic's Hedgehog		
4	WhitePeopleTwitter	I guess I'm getting rid of Netflix then…		
5	meirl	meirl		
6	wholesomememes	like father like son!		
7	aww	Sleeping in the mother's arms		
8	interestingasfuck	In the 1970s, North Korea ordered 1,000 Volvo …		
9	dankmemes	Is a.i. art banned yet?		
10	todayilearned	TIL: In 1962, a 10 year old found a radioactiv…		
11	interestingasfuck	The last delivered Boeing 747 made a crown wit…		
12	rareinsults	Mayo is no joke		
13	AnimalsBeingBros	Parrot ask his owner if he's alright after he …		
14	therewasanattempt	To massage yourself unnoticed		
15	Unexpected	The Night Train nsfw		
16	news	California police kill double amputee who was …		
17	comics	[OC] Single Player		
18	mildlyinfuriating	Convenience store worker wouldn’t accept this …		
19	politics	Fireworks in House after Democrat says 'insurr…		
20	PublicFreakout	12 Year Old Tiktok prankster throws a dead sna…		
21	WhitePeopleTwitter	Humility		
22	HumansBeingBros	Saving a cow calf from crossing the rainbow road.		
23	antiwork	First the French now the Brits		
24	wallstreetbets	if JPow gives up trying to correct asset (real…		

```
1    91469  1.675255e+09
2    84070  1.675272e+09
3    80431  1.675245e+09
4    79607  1.675261e+09
5    74146  1.675264e+09
6    64840  1.675234e+09
7    63646  1.675259e+09
8    62637  1.675255e+09
9    62167  1.675254e+09
10   59277  1.675245e+09
11   59766  1.675280e+09
12   58187  1.675262e+09
13   53412  1.675270e+09
14   52549  1.675252e+09
15   51793  1.675254e+09
16   50856  1.675258e+09
17   48763  1.675258e+09
18   44390  1.675267e+09
19   45017  1.675284e+09
20   44506  1.675272e+09
21   43642  1.675273e+09
22   43463  1.675270e+09
23   41923  1.675255e+09
24   41043  1.675269e+09
```

1.6 Problem 4

The NBA has saved data on all 30 teams' shooting statistics for the 2014-2015 season here: https://stats.nba.com/js/data/sportvu/2015/shootingTeamData.json. Take a moment and look at this JSON file in your web browser. The structure of this particular JSON is complicated, but see if you can find the team-by-team data. In this problem our goal is to use `pd.json_normalize()` to get the data into a dataframe. The following questions will guide you towards this goal.

1.6.1 Part a

Download the raw text of the NBA JSON file and register it as JSON formatted data in Python's memory. (2 points)

[9]:
```
url = 'https://stats.nba.com/js/data/sportvu/2015/shootingTeamData.json'
response = requests.get(url)
response_body_as_string = response.text
body = json.loads(response_body_as_string)
```

1.6.2 Part b

Describe, in words, the path that leads to the team-by-team data. (2 points)

The response body of the request to https://stats.nba.com/js/data/sportvu/2015/shootingTeamData.json is a JSON object. This JSON object has a field with key `resultSets` and a value of a JSON

array. This JSON array has one JSON object. This JSON object has a field with key `rowSet` and a value of a JSON array of team-by-team data.

1.6.3 Part c

Use the `pd.json_normalize()` function to pull the team-by-team data into a dataframe. This is going to be tricky. You will need to use indexing on the JSON data as well as the `record_path` parameter.

If you are successful, you will have a dataframe with 30 rows and 33 columns. The first row will refer to the Golden State Warriors, the second row will refer to the San Antonio Spurs, and the third row will refer to the Cleveland Cavaliers. The columns will only be named 0, 1, 2, … at this point. (4 points)

[10]:
```
data_frame = pd.json_normalize(body, record_path = ['resultSets', 'rowSet'])
data_frame
```

[10]:

	0	1	2	3	4	5	6	7	8	\
0	1610612744	Golden State	Warriors	GSW		82	48.7	114.9	14.9	
1	1610612759	San Antonio	Spurs	SAS		82	48.3	103.5	14.8	
2	1610612739	Cleveland	Cavaliers	CLE		82	48.7	104.3	16.9	
3	1610612746	Los Angeles	Clippers	LAC		82	48.6	104.5	15.0	
4	1610612760	Oklahoma City	Thunder	OKC		82	48.6	110.2	16.1	
5	1610612737	Atlanta	Hawks	ATL		82	48.6	102.8	19.0	
6	1610612745	Houston	Rockets	HOU		82	48.6	106.5	17.2	
7	1610612757	Portland	Trail Blazers	POR		82	48.5	105.1	17.5	
8	1610612758	Sacramento	Kings	SAC		81	48.4	106.7	18.7	
9	1610612764	Washington	Wizards	WAS		82	48.5	104.1	15.4	
10	1610612748	Miami	Heat	MIA		82	48.6	100.0	17.9	
11	1610612761	Toronto	Raptors	TOR		81	48.5	102.7	23.0	
12	1610612742	Dallas	Mavericks	DAL		82	49.0	102.3	18.2	
13	1610612766	Charlotte	Hornets	CHA		82	48.6	103.4	16.8	
14	1610612762	Utah	Jazz	UTA		82	49.0	97.7	18.1	
15	1610612753	Orlando	Magic	ORL		81	48.7	102.0	18.0	
16	1610612749	Milwaukee	Bucks	MIL		82	48.7	99.0	17.4	
17	1610612740	New Orleans	Pelicans	NOP		82	48.5	102.7	19.9	
18	1610612750	Minnesota	Timberwolves	MIN		82	48.6	102.4	15.1	
19	1610612754	Indiana	Pacers	IND		82	48.8	102.2	13.7	
20	1610612751	Brooklyn	Nets	BKN		82	48.4	98.6	14.4	
21	1610612765	Detroit	Pistons	DET		82	48.7	102.0	17.5	
22	1610612743	Denver	Nuggets	DEN		82	48.6	101.9	15.9	
23	1610612738	Boston	Celtics	BOS		81	48.5	105.6	18.9	
24	1610612741	Chicago	Bulls	CHI		82	48.9	101.6	18.1	
25	1610612755	Philadelphia	76ers	PHI		82	48.6	97.4	19.7	
26	1610612756	Phoenix	Suns	PHX		82	48.4	100.9	15.6	
27	1610612752	New York	Knicks	NYK		82	48.5	98.4	10.4	
28	1610612763	Memphis	Grizzlies	MEM		82	48.6	99.1	16.4	
29	1610612747	Los Angeles	Lakers	LAL		82	48.3	97.3	15.6	

	9	…	23	24	25	26	27	28	29	30	31	32
0	0.498	…	0.478	21.2	42.5	0.497	2.3	6.3	0.363	10.8	25.3	0.429
1	0.481	…	0.506	18.3	39.8	0.460	0.9	2.6	0.341	6.1	15.9	0.381
2	0.481	…	0.473	18.2	40.7	0.447	1.7	5.7	0.299	9.0	23.9	0.378
3	0.497	…	0.480	18.9	42.0	0.450	2.0	6.0	0.334	7.7	20.8	0.373
4	0.480	…	0.497	17.5	38.7	0.451	1.6	5.1	0.321	6.6	18.6	0.356
5	0.463	…	0.483	19.4	44.6	0.435	1.0	3.1	0.311	9.0	25.3	0.355
6	0.433	…	0.472	15.5	36.4	0.426	2.3	7.4	0.318	8.4	23.5	0.355
7	0.441	…	0.447	18.0	39.8	0.453	1.7	5.9	0.295	8.8	22.6	0.389
8	0.452	…	0.473	18.1	39.7	0.454	0.9	3.1	0.276	7.2	19.4	0.372
9	0.480	…	0.483	19.5	44.3	0.439	0.7	2.7	0.254	8.0	21.5	0.371
10	0.488	…	0.490	15.7	35.2	0.445	0.8	2.9	0.282	5.3	15.1	0.347
11	0.462	…	0.461	14.1	32.4	0.436	1.8	5.6	0.327	6.8	17.7	0.384
12	0.473	…	0.464	17.5	41.4	0.423	1.4	5.3	0.273	8.4	23.3	0.360
13	0.459	…	0.449	17.0	39.8	0.427	1.8	6.0	0.297	8.9	23.4	0.379
14	0.445	…	0.468	15.9	37.2	0.426	1.4	4.3	0.318	7.1	19.5	0.363
15	0.456	…	0.475	18.5	42.6	0.435	0.7	2.7	0.249	7.1	19.5	0.363
16	0.463	…	0.477	13.2	29.4	0.448	1.1	4.0	0.270	4.3	11.6	0.370
17	0.458	…	0.460	17.9	41.1	0.434	0.6	2.6	0.247	7.9	21.2	0.374
18	0.464	…	0.471	16.1	35.4	0.455	0.7	2.6	0.272	4.8	13.8	0.350
19	0.453	…	0.465	16.4	38.1	0.431	1.7	5.7	0.299	6.4	17.4	0.368
20	0.457	…	0.464	15.8	36.1	0.438	1.0	3.3	0.303	5.5	15.1	0.363
21	0.464	…	0.452	15.7	37.2	0.422	0.9	4.0	0.227	8.1	22.2	0.366
22	0.406	…	0.448	16.4	37.8	0.434	1.1	4.3	0.264	6.9	19.5	0.354
23	0.453	…	0.451	16.9	39.9	0.424	1.6	5.7	0.274	7.1	20.3	0.350
24	0.458	…	0.442	17.0	38.5	0.441	1.3	3.9	0.332	6.6	17.5	0.380
25	0.445	…	0.449	15.3	37.4	0.409	1.6	5.7	0.281	7.7	21.8	0.354
26	0.440	…	0.447	16.6	39.5	0.421	1.4	5.0	0.288	7.6	20.8	0.363
27	0.447	…	0.439	15.9	36.4	0.438	1.5	4.9	0.305	5.9	16.6	0.358
28	0.440	…	0.459	16.1	38.5	0.418	0.7	2.5	0.278	5.4	16.0	0.340
29	0.441	…	0.420	14.0	34.5	0.406	2.2	7.9	0.278	5.6	16.7	0.335

[30 rows x 33 columns]

1.6.4 Part d

Find the path that leads to the headers (the column names), and extract these names as a list. Then set the `.columns` attribute of the dataframe you created in part c equal to this list. The result should be that the dataframe now has the correct column names. (3 points)

[11]:
```
list_of_result_sets = body.get('resultSets')
result_set = list_of_result_sets[0]
list_of_headers = result_set.get('headers')
data_frame.columns = list_of_headers
data_frame
```

[11]:

	TEAM_ID	TEAM_CITY	TEAM_NAME	TEAM_ABBREVIATION	TEAM_CODE	GP	\
0	1610612744	Golden State	Warriors	GSW		82	
1	1610612759	San Antonio	Spurs	SAS		82	
2	1610612739	Cleveland	Cavaliers	CLE		82	
3	1610612746	Los Angeles	Clippers	LAC		82	
4	1610612760	Oklahoma City	Thunder	OKC		82	
5	1610612737	Atlanta	Hawks	ATL		82	
6	1610612745	Houston	Rockets	HOU		82	
7	1610612757	Portland	Trail Blazers	POR		82	
8	1610612758	Sacramento	Kings	SAC		81	
9	1610612764	Washington	Wizards	WAS		82	
10	1610612748	Miami	Heat	MIA		82	
11	1610612761	Toronto	Raptors	TOR		81	
12	1610612742	Dallas	Mavericks	DAL		82	
13	1610612766	Charlotte	Hornets	CHA		82	
14	1610612762	Utah	Jazz	UTA		82	
15	1610612753	Orlando	Magic	ORL		81	
16	1610612749	Milwaukee	Bucks	MIL		82	
17	1610612740	New Orleans	Pelicans	NOP		82	
18	1610612750	Minnesota	Timberwolves	MIN		82	
19	1610612754	Indiana	Pacers	IND		82	
20	1610612751	Brooklyn	Nets	BKN		82	
21	1610612765	Detroit	Pistons	DET		82	
22	1610612743	Denver	Nuggets	DEN		82	
23	1610612738	Boston	Celtics	BOS		81	
24	1610612741	Chicago	Bulls	CHI		82	
25	1610612755	Philadelphia	76ers	PHI		82	
26	1610612756	Phoenix	Suns	PHX		82	
27	1610612752	New York	Knicks	NYK		82	
28	1610612763	Memphis	Grizzlies	MEM		82	
29	1610612747	Los Angeles	Lakers	LAL		82	

	MIN	PTS	PTS_DRIVE	FGP_DRIVE	…	CFGP	UFGM	UFGA	UFGP	CFG3M	\
0	48.7	114.9	14.9	0.498	…	0.478	21.2	42.5	0.497	2.3	
1	48.3	103.5	14.8	0.481	…	0.506	18.3	39.8	0.460	0.9	
2	48.7	104.3	16.9	0.481	…	0.473	18.2	40.7	0.447	1.7	
3	48.6	104.5	15.0	0.497	…	0.480	18.9	42.0	0.450	2.0	
4	48.6	110.2	16.1	0.480	…	0.497	17.5	38.7	0.451	1.6	
5	48.6	102.8	19.0	0.463	…	0.483	19.4	44.6	0.435	1.0	
6	48.6	106.5	17.2	0.433	…	0.472	15.5	36.4	0.426	2.3	
7	48.5	105.1	17.5	0.441	…	0.447	18.0	39.8	0.453	1.7	
8	48.4	106.7	18.7	0.452	…	0.473	18.1	39.7	0.454	0.9	
9	48.5	104.1	15.4	0.480	…	0.483	19.5	44.3	0.439	0.7	
10	48.6	100.0	17.9	0.488	…	0.490	15.7	35.2	0.445	0.8	
11	48.5	102.7	23.0	0.462	…	0.461	14.1	32.4	0.436	1.8	
12	49.0	102.3	18.2	0.473	…	0.464	17.5	41.4	0.423	1.4	
13	48.6	103.4	16.8	0.459	…	0.449	17.0	39.8	0.427	1.8	

14	49.0	97.7	18.1	0.445	…	0.468	15.9	37.2	0.426	1.4
15	48.7	102.0	18.0	0.456	…	0.475	18.5	42.6	0.435	0.7
16	48.7	99.0	17.4	0.463	…	0.477	13.2	29.4	0.448	1.1
17	48.5	102.7	19.9	0.458	…	0.460	17.9	41.1	0.434	0.6
18	48.6	102.4	15.1	0.464	…	0.471	16.1	35.4	0.455	0.7
19	48.8	102.2	13.7	0.453	…	0.465	16.4	38.1	0.431	1.7
20	48.4	98.6	14.4	0.457	…	0.464	15.8	36.1	0.438	1.0
21	48.7	102.0	17.5	0.464	…	0.452	15.7	37.2	0.422	0.9
22	48.6	101.9	15.9	0.406	…	0.448	16.4	37.8	0.434	1.1
23	48.5	105.6	18.9	0.453	…	0.451	16.9	39.9	0.424	1.6
24	48.9	101.6	18.1	0.458	…	0.442	17.0	38.5	0.441	1.3
25	48.6	97.4	19.7	0.445	…	0.449	15.3	37.4	0.409	1.6
26	48.4	100.9	15.6	0.440	…	0.447	16.6	39.5	0.421	1.4
27	48.5	98.4	10.4	0.447	…	0.439	15.9	36.4	0.438	1.5
28	48.6	99.1	16.4	0.440	…	0.459	16.1	38.5	0.418	0.7
29	48.3	97.3	15.6	0.441	…	0.420	14.0	34.5	0.406	2.2

	CFG3A	CFG3P	UFG3M	UFG3A	UFG3P
0	6.3	0.363	10.8	25.3	0.429
1	2.6	0.341	6.1	15.9	0.381
2	5.7	0.299	9.0	23.9	0.378
3	6.0	0.334	7.7	20.8	0.373
4	5.1	0.321	6.6	18.6	0.356
5	3.1	0.311	9.0	25.3	0.355
6	7.4	0.318	8.4	23.5	0.355
7	5.9	0.295	8.8	22.6	0.389
8	3.1	0.276	7.2	19.4	0.372
9	2.7	0.254	8.0	21.5	0.371
10	2.9	0.282	5.3	15.1	0.347
11	5.6	0.327	6.8	17.7	0.384
12	5.3	0.273	8.4	23.3	0.360
13	6.0	0.297	8.9	23.4	0.379
14	4.3	0.318	7.1	19.5	0.363
15	2.7	0.249	7.1	19.5	0.363
16	4.0	0.270	4.3	11.6	0.370
17	2.6	0.247	7.9	21.2	0.374
18	2.6	0.272	4.8	13.8	0.350
19	5.7	0.299	6.4	17.4	0.368
20	3.3	0.303	5.5	15.1	0.363
21	4.0	0.227	8.1	22.2	0.366
22	4.3	0.264	6.9	19.5	0.354
23	5.7	0.274	7.1	20.3	0.350
24	3.9	0.332	6.6	17.5	0.380
25	5.7	0.281	7.7	21.8	0.354
26	5.0	0.288	7.6	20.8	0.363
27	4.9	0.305	5.9	16.6	0.358
28	2.5	0.278	5.4	16.0	0.340

```
29    7.9  0.278    5.6   16.7  0.335

[30 rows x 33 columns]
```

1.7 Problem 5

Save the NBA dataframe you extracted in problem 4 as a JSON-formatted text file on your local machine. Format the JSON so that it is organized as dictionary with three lists: `columns` lists the column names, `index` lists the row names, and `data` is a list-of-lists of data points, one list for each row. (Hint: this is possible with one line of code) (2 points)

[12]:
```
data_frame.to_json('shooting_statistics.json', orient = 'split')
pd.read_json('shooting_statistics.json', orient = 'split')
```

[12]:

	TEAM_ID	TEAM_CITY	TEAM_NAME	TEAM_ABBREVIATION	TEAM_CODE	GP	\
0	1610612744	Golden State	Warriors	GSW		82	
1	1610612759	San Antonio	Spurs	SAS		82	
2	1610612739	Cleveland	Cavaliers	CLE		82	
3	1610612746	Los Angeles	Clippers	LAC		82	
4	1610612760	Oklahoma City	Thunder	OKC		82	
5	1610612737	Atlanta	Hawks	ATL		82	
6	1610612745	Houston	Rockets	HOU		82	
7	1610612757	Portland	Trail Blazers	POR		82	
8	1610612758	Sacramento	Kings	SAC		81	
9	1610612764	Washington	Wizards	WAS		82	
10	1610612748	Miami	Heat	MIA		82	
11	1610612761	Toronto	Raptors	TOR		81	
12	1610612742	Dallas	Mavericks	DAL		82	
13	1610612766	Charlotte	Hornets	CHA		82	
14	1610612762	Utah	Jazz	UTA		82	
15	1610612753	Orlando	Magic	ORL		81	
16	1610612749	Milwaukee	Bucks	MIL		82	
17	1610612740	New Orleans	Pelicans	NOP		82	
18	1610612750	Minnesota	Timberwolves	MIN		82	
19	1610612754	Indiana	Pacers	IND		82	
20	1610612751	Brooklyn	Nets	BKN		82	
21	1610612765	Detroit	Pistons	DET		82	
22	1610612743	Denver	Nuggets	DEN		82	
23	1610612738	Boston	Celtics	BOS		81	
24	1610612741	Chicago	Bulls	CHI		82	
25	1610612755	Philadelphia	76ers	PHI		82	
26	1610612756	Phoenix	Suns	PHX		82	
27	1610612752	New York	Knicks	NYK		82	
28	1610612763	Memphis	Grizzlies	MEM		82	
29	1610612747	Los Angeles	Lakers	LAL		82	

	MIN	PTS	PTS_DRIVE	FGP_DRIVE	…	CFGP	UFGM	UFGA	UFGP	CFG3M	\

0	48.7	114.9	14.9	0.498	…	0.478	21.2	42.5	0.497	2.3
1	48.3	103.5	14.8	0.481	…	0.506	18.3	39.8	0.460	0.9
2	48.7	104.3	16.9	0.481	…	0.473	18.2	40.7	0.447	1.7
3	48.6	104.5	15.0	0.497	…	0.480	18.9	42.0	0.450	2.0
4	48.6	110.2	16.1	0.480	…	0.497	17.5	38.7	0.451	1.6
5	48.6	102.8	19.0	0.463	…	0.483	19.4	44.6	0.435	1.0
6	48.6	106.5	17.2	0.433	…	0.472	15.5	36.4	0.426	2.3
7	48.5	105.1	17.5	0.441	…	0.447	18.0	39.8	0.453	1.7
8	48.4	106.7	18.7	0.452	…	0.473	18.1	39.7	0.454	0.9
9	48.5	104.1	15.4	0.480	…	0.483	19.5	44.3	0.439	0.7
10	48.6	100.0	17.9	0.488	…	0.490	15.7	35.2	0.445	0.8
11	48.5	102.7	23.0	0.462	…	0.461	14.1	32.4	0.436	1.8
12	49.0	102.3	18.2	0.473	…	0.464	17.5	41.4	0.423	1.4
13	48.6	103.4	16.8	0.459	…	0.449	17.0	39.8	0.427	1.8
14	49.0	97.7	18.1	0.445	…	0.468	15.9	37.2	0.426	1.4
15	48.7	102.0	18.0	0.456	…	0.475	18.5	42.6	0.435	0.7
16	48.7	99.0	17.4	0.463	…	0.477	13.2	29.4	0.448	1.1
17	48.5	102.7	19.9	0.458	…	0.460	17.9	41.1	0.434	0.6
18	48.6	102.4	15.1	0.464	…	0.471	16.1	35.4	0.455	0.7
19	48.8	102.2	13.7	0.453	…	0.465	16.4	38.1	0.431	1.7
20	48.4	98.6	14.4	0.457	…	0.464	15.8	36.1	0.438	1.0
21	48.7	102.0	17.5	0.464	…	0.452	15.7	37.2	0.422	0.9
22	48.6	101.9	15.9	0.406	…	0.448	16.4	37.8	0.434	1.1
23	48.5	105.6	18.9	0.453	…	0.451	16.9	39.9	0.424	1.6
24	48.9	101.6	18.1	0.458	…	0.442	17.0	38.5	0.441	1.3
25	48.6	97.4	19.7	0.445	…	0.449	15.3	37.4	0.409	1.6
26	48.4	100.9	15.6	0.440	…	0.447	16.6	39.5	0.421	1.4
27	48.5	98.4	10.4	0.447	…	0.439	15.9	36.4	0.438	1.5
28	48.6	99.1	16.4	0.440	…	0.459	16.1	38.5	0.418	0.7
29	48.3	97.3	15.6	0.441	…	0.420	14.0	34.5	0.406	2.2

	CFG3A	CFG3P	UFG3M	UFG3A	UFG3P
0	6.3	0.363	10.8	25.3	0.429
1	2.6	0.341	6.1	15.9	0.381
2	5.7	0.299	9.0	23.9	0.378
3	6.0	0.334	7.7	20.8	0.373
4	5.1	0.321	6.6	18.6	0.356
5	3.1	0.311	9.0	25.3	0.355
6	7.4	0.318	8.4	23.5	0.355
7	5.9	0.295	8.8	22.6	0.389
8	3.1	0.276	7.2	19.4	0.372
9	2.7	0.254	8.0	21.5	0.371
10	2.9	0.282	5.3	15.1	0.347
11	5.6	0.327	6.8	17.7	0.384
12	5.3	0.273	8.4	23.3	0.360
13	6.0	0.297	8.9	23.4	0.379
14	4.3	0.318	7.1	19.5	0.363

```
15    2.7  0.249    7.1   19.5  0.363
16    4.0  0.270    4.3   11.6  0.370
17    2.6  0.247    7.9   21.2  0.374
18    2.6  0.272    4.8   13.8  0.350
19    5.7  0.299    6.4   17.4  0.368
20    3.3  0.303    5.5   15.1  0.363
21    4.0  0.227    8.1   22.2  0.366
22    4.3  0.264    6.9   19.5  0.354
23    5.7  0.274    7.1   20.3  0.350
24    3.9  0.332    6.6   17.5  0.380
25    5.7  0.281    7.7   21.8  0.354
26    5.0  0.288    7.6   20.8  0.363
27    4.9  0.305    5.9   16.6  0.358
28    2.5  0.278    5.4   16.0  0.340
29    7.9  0.278    5.6   16.7  0.335

[30 rows x 33 columns]
```

[ ]: