

# Repository and Mining of Temporal Data

---

Jessica Nguy  
Siomara Nieves

Dr. Philip Chan

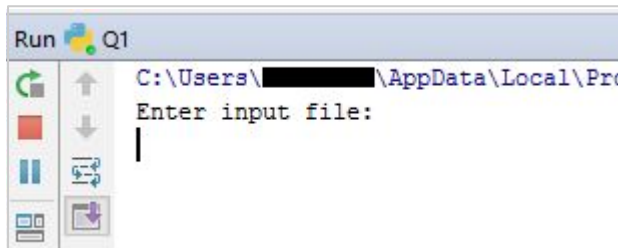
# Progress of Current Milestone

<b>Task</b>	<b>Completion %</b>	<b>Jessica</b>	<b>Siomara</b>	<b>To-Do</b>
CSV User Input catch cases	95%	0%	95%	Add more test cases
Q1	90%	80%	10%	Test, x-axis for figures 2 and 3
Website	100%	80%	20%	Completion deferred to later Milestone
Data Processing	N/A	N/A	N/A	Deferred to Milestone 3
Database setup for Meta-data, Meta-data inputs	80%	0%	80%	Switch Database to SQLite; Deferred to Milestone 3
Evaluation Document, Presentation	100%	60%	40%	None

# Discussion of Each Accomplished Task

- .CSV User Input Test Cases
- Q1
- Website
- Data Processing
- Database Setup
- Evaluation Document, Presentation

# parseCSV



```
Run Q1
C:\Users\██████████\AppData\Local\Pro
Enter input file:
|
```

```
1 import csv
2
3 def read_file():
4
5     #userIn = input("Enter input file: ")
6     print("Enter input file:")
7     #userIn = input() + ".csv"
8     with open(input()) as csvFile:
9         readCSV = csv.reader(csvFile, delimiter=",") # csv = comma separated
10        ...
11
12        dates = []
13        info = []
14
15        for row in readCSV:
16            time = row[0]
17            data = row[1]
18
19            if time == "." or data == ".": # where there's a date with no data put ' . '
20                continue
21
22            elif time == " " or data == " ": # testing for blanks
23                continue
24
25            else: # only add them in here
26                dates.append(time)
27                info.append(data)
28
29        return (info, dates)
```

# Q1

```
26 # ----- Part 1 -----
27 # |
28 # | Convert csv file to readable |
29 # | format for x-axis and y-axis |
30 # |
31 # -----
32
33 # Takes in csv file for x-axis and y-axis from csvReader
34
35 yaxisI, xaxisI = read_file()
36
37 xTitle = xaxisI.pop(0)
38 yTitle = yaxisI.pop(0)
39
40 xaxis_datetime = []
41 for x in xaxisI:
42     xaxis_datetime.append(datetime.strptime(x, '%Y-%m-%d'))
43     ...
```

```
56 # ----- Part 1 -----
57 # |
58 # | Graph x-values versus y-values |
59 # |
60 # -----
61
62
63 titleString = "Q1: Time versus "
64 titleString = titleString + yTitle
65
66 # Graph Plot
67 fig1 = plt.figure()
68 fig1 = plt.plot(xaxis, yaxis)
69 plt.title(titleString) # Append variable
70 plt.xlabel(xTitle)
71 plt.ylabel(yTitle)
```

# Q1

```
75 # ----- Part 2 -----
76 # |
77 # | Graph the change in x-values |
78 # | Calculate z-scores for dataset |
79 # |           Color-Code Range |
80 # |
81 # -----
107 meanY = np.mean(deltaY)
108 stdDevY = np.std(deltaY)
109
110 topRangeZ = (meanY + (3.0 * stdDevY))
111 highRangeZ = (meanY + (2.5 * stdDevY))
112 zeroRangeZ = (meanY + (0 * stdDevY))
113 lowRangeZ = (meanY + (-2.5 * stdDevY))
114 bottomRangeZ = (meanY + (-3.0 * stdDevY))
115
116 # -----
117 # Convert lists into numpy-readable
118 x = np.array(deltaX)
119 y = np.array(deltaY)
120
121
122 cmap = ListedColormap(['r', 'y', 'g', 'y', 'r'])
123 norm = BoundaryNorm([negInf, bottomRangeZ, lowRangeZ,
124                    highRangeZ, topRangeZ, posInf], cmap.N)
125 points = np.array([x, y]).T.reshape(-1, 1, 2)
126 segments = np.concatenate([points[:-1], points[1:]], axis=1)
```

```
149 # ----- Part 3 -----
150 # |
151 # | Graph x-values versus z-score |
152 # |           Color-Code Range |
153 # |
154 # -----
155 #graph x values z-score
156 yzaxis = []
157 yzaxis = stats.zscore(yaxis)
164 # ----- Set Line Colors -----
165 fig3 = plt.figure()
166
167 xzaxis = np.array(xaxis)
168 yzaxis = np.array(yzaxis)
184 # ----- Plot Graph -----
185
186 plt.title(begTitle)
187 plt.axhline(y=3.0, c='k', linestyle='--')
188 plt.axhline(y=2.5, c='k', linestyle='--')
189 plt.axhline(y=0, c='k', linestyle='--')
190 plt.axhline(y=-2.5, c='k', linestyle='--')
191 plt.axhline(y=-3.0, c='k', linestyle='--')
192 plt.xlabel(xTitle)
193 plt.ylabel(midTitle)
194
195 #fig3_size = plt.rcParams["figure.figsize"] = [8,8]
196
197 plt.show()
```

# Website

```
MINGW64 ~/documents/school/djstorage/mysite
$ source ~/.virtualenvs/djangodev/Scripts/activate
(djangodev)
MINGW64 ~/documents/school/djstorage/mysite
$ python manage.py runserver
Performing system checks...

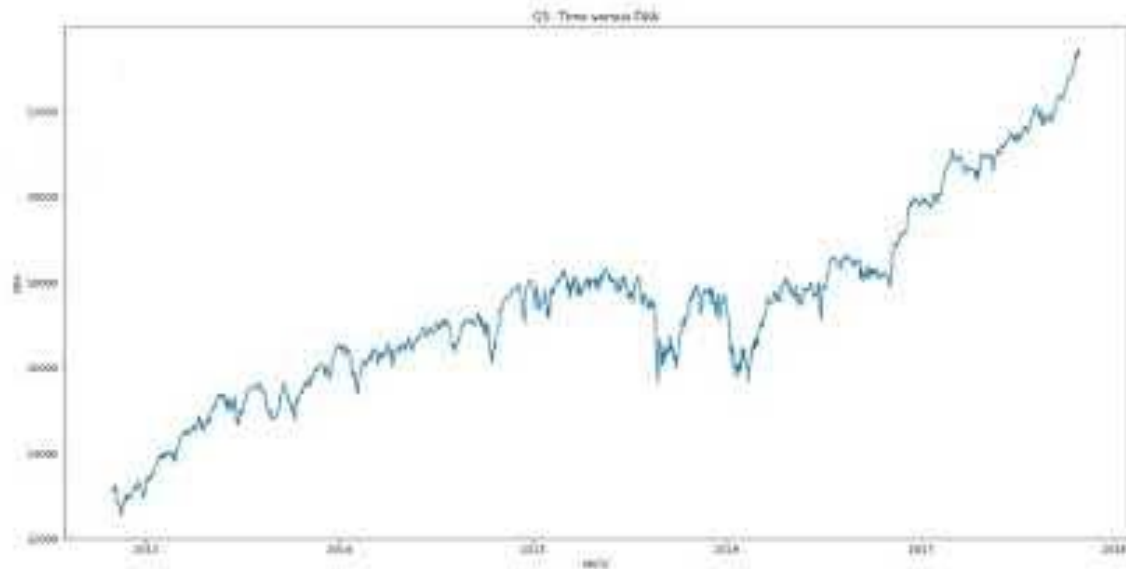
System check identified no issues (0 silenced).
October 30, 2017 - 10:15:49
Django version 2.1.dev20171010132034, using settings 'mysite.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```



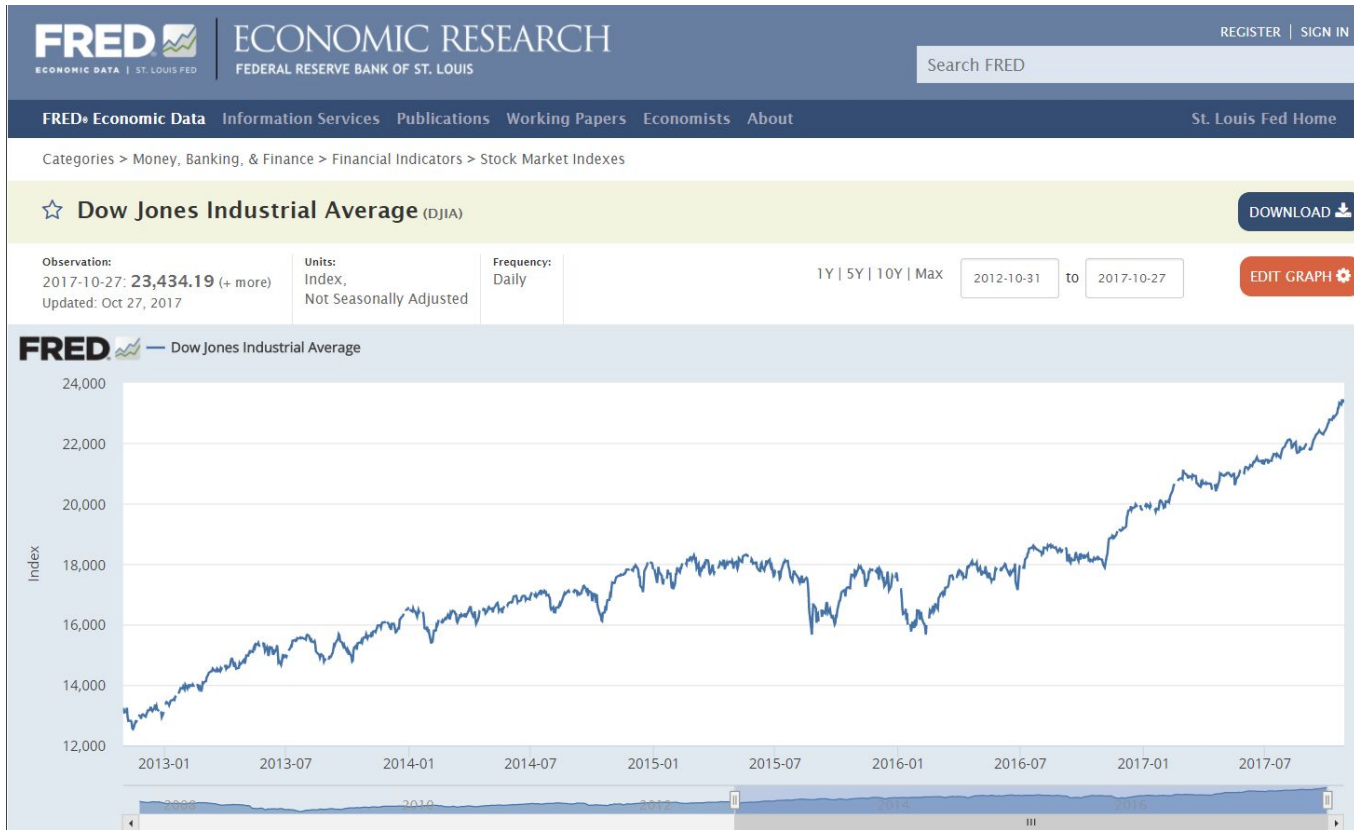
- **Data Processing:** not used for Q1, deferred to a later Milestone.
- **Database Setup:** Database is complete but will be changed from SQL to SQLite and further enhanced on in Milestone 3. The current database saves the metadata (tags, description, start/end timestamps, public/private) that will be provided by the user for Q2.
- **Evaluation Document, Presentation**



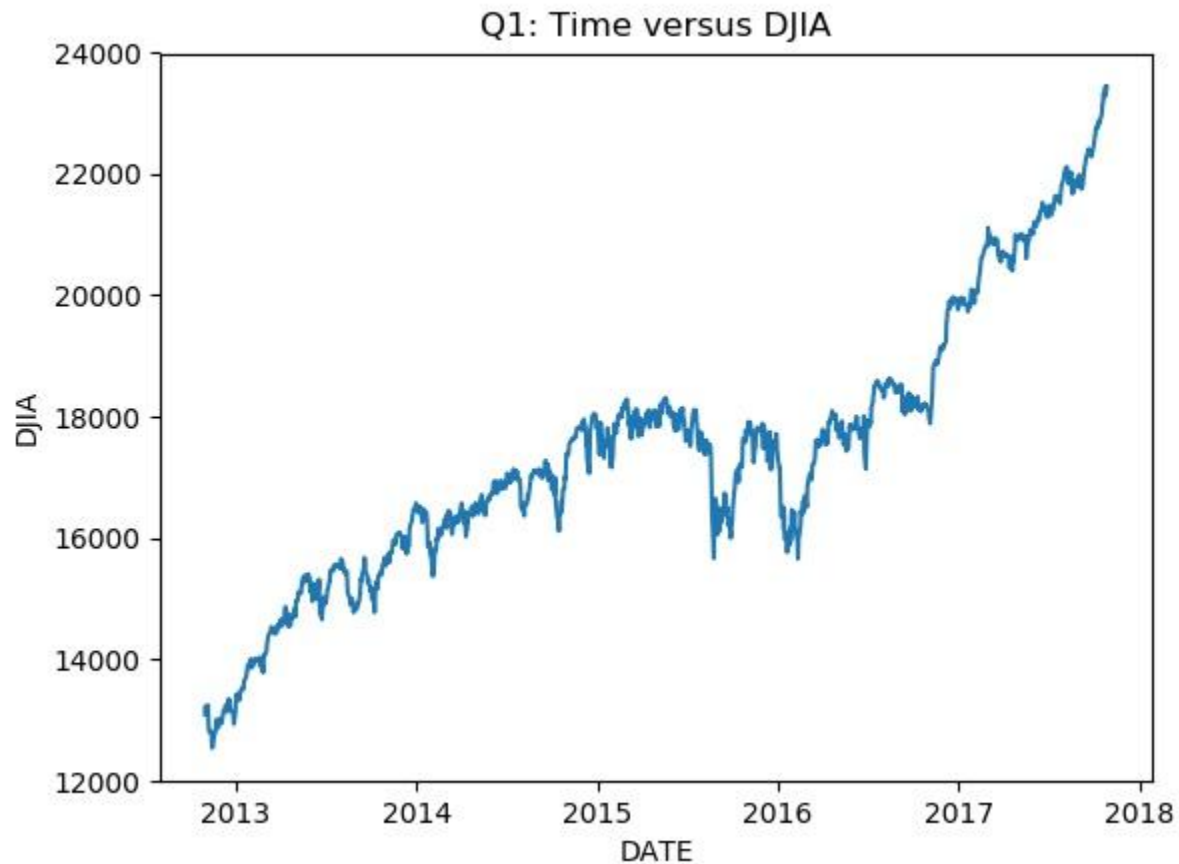
# Demo



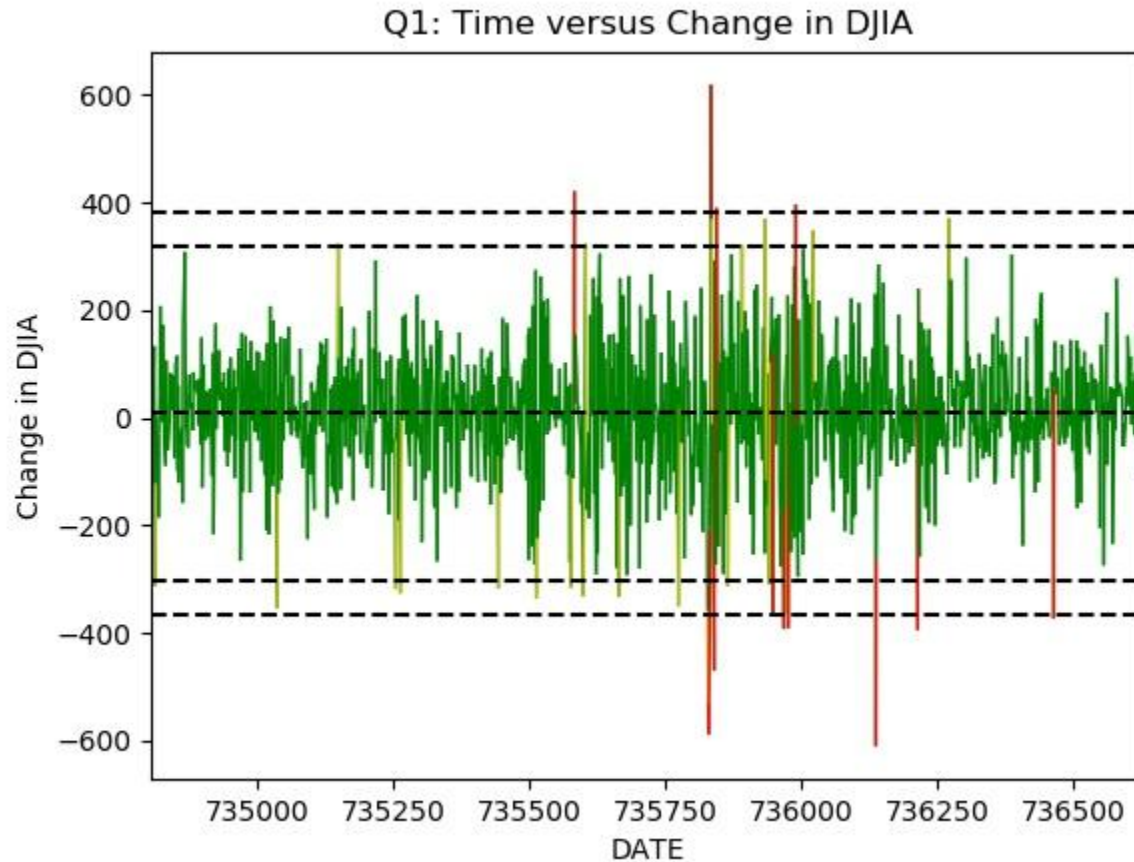
# Comparison to FRED Economic Data



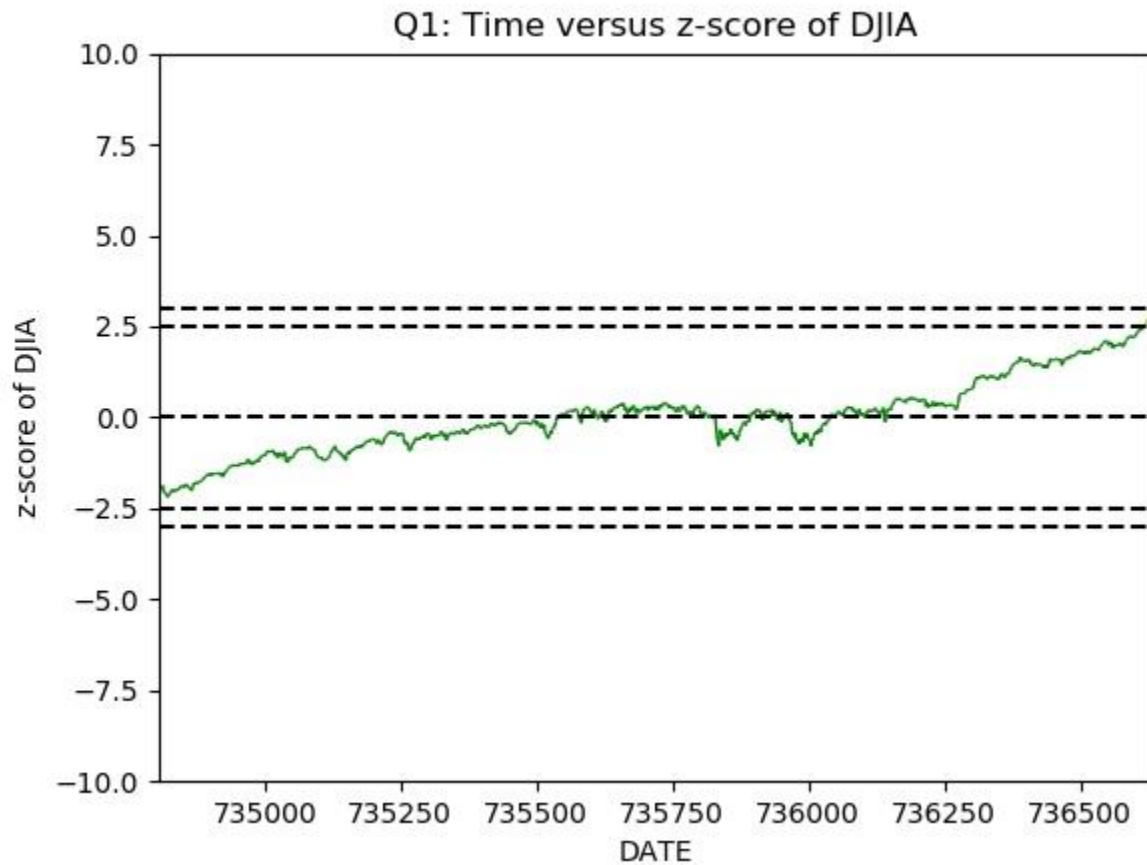
# Q1: Part 1



# Q1: Part 2



# Q1: Part 3



# Plan for Next Milestone

<b>Task</b>	<b>Jessica</b>	<b>Siomara</b>
1.) Q2	Calculations for Pearson correlation, cross correlation, visualizations, writing main code	File storage, file conversion, visualizations
2.) Narrow Data	Research and create program to narrow data. Integration with database may be needed. Create NarrowData.py	Research how to implement a search for queries depending on the metadata provided by other users.
3.) Database Processing	Make sure that code can use information from the database	Linking database to Django, running test cases for queries and selections
4.) Meta-Data, Meta-Data inputs	Create Input.py, be able to retrieve metadata and use it to run Narrow Data.py	Asking user for metadata input and storing into the database
5.) Improve existing code	Ensure that changes in .csv are carried over and do not break Q1.py and Q2.py	Creating more catch cases for errors in file, x-axis plotting for figures 2 and 3
6.) Evaluation Document, Presentation	Write evaluation document, Create presentation, put code onto GitHub repository.	Write evaluation document, Create presentation, put code onto GitHub repository.