

CS 242 Assignment 2.1
Manual Test Plan
Yayi Ning

Data-analysis and visualization parts:

By running Main.py you will see:

Chose int value from one of the following:

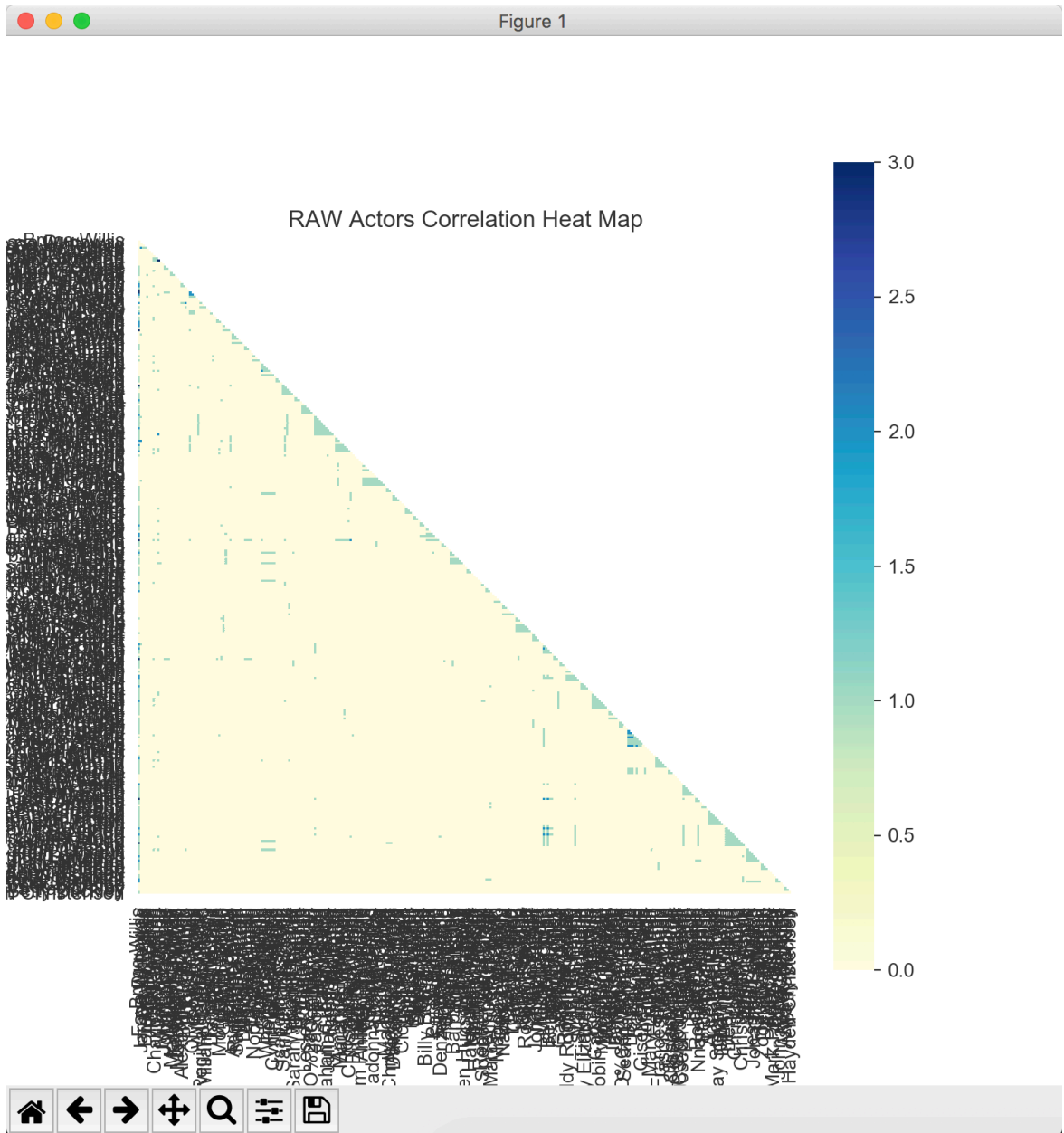
- 1 = Show the hub Star:
- 2 = Display raw (all stars) Star correlation graph
- 3 = Display main hub Stars graph
- 4 = Show which age group have the most gross value?
- 5 = Display the age v.s. gross graph
- 0 = EXIST

Result of Enter 1:

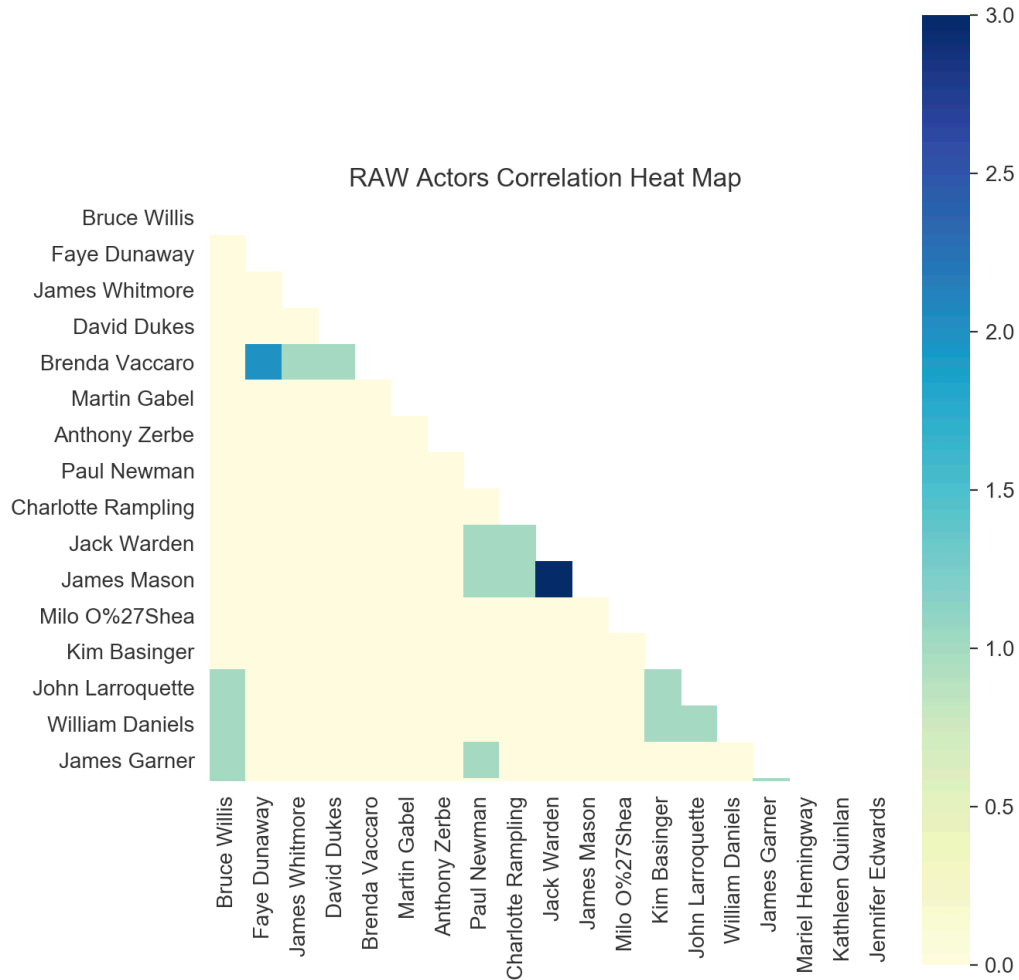
```
1
The hub star is: Bruce Willis

Chose int value from one of the following:
1 = Show the hub Star:
2 = Display raw (all stars) Star correlation graph
3 = Display main hub Stars graph
4 = Show which age group have the most gross value?
5 = Display the age v.s. gross graph
0 = EXIST
```

Result of Enter 2:

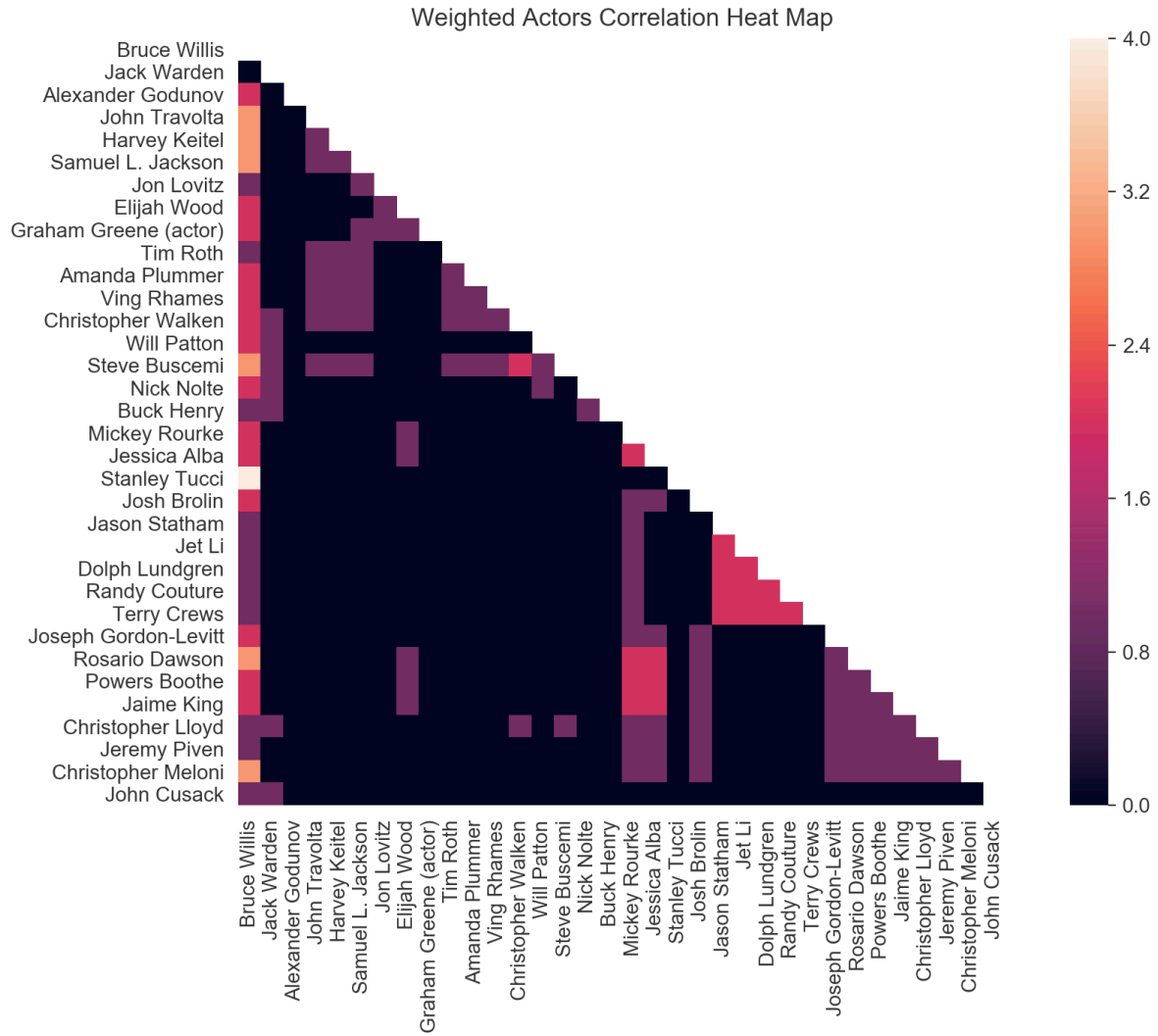


Zoom in:



Enter 3 to see weighted actor correlation map:

Figure 1

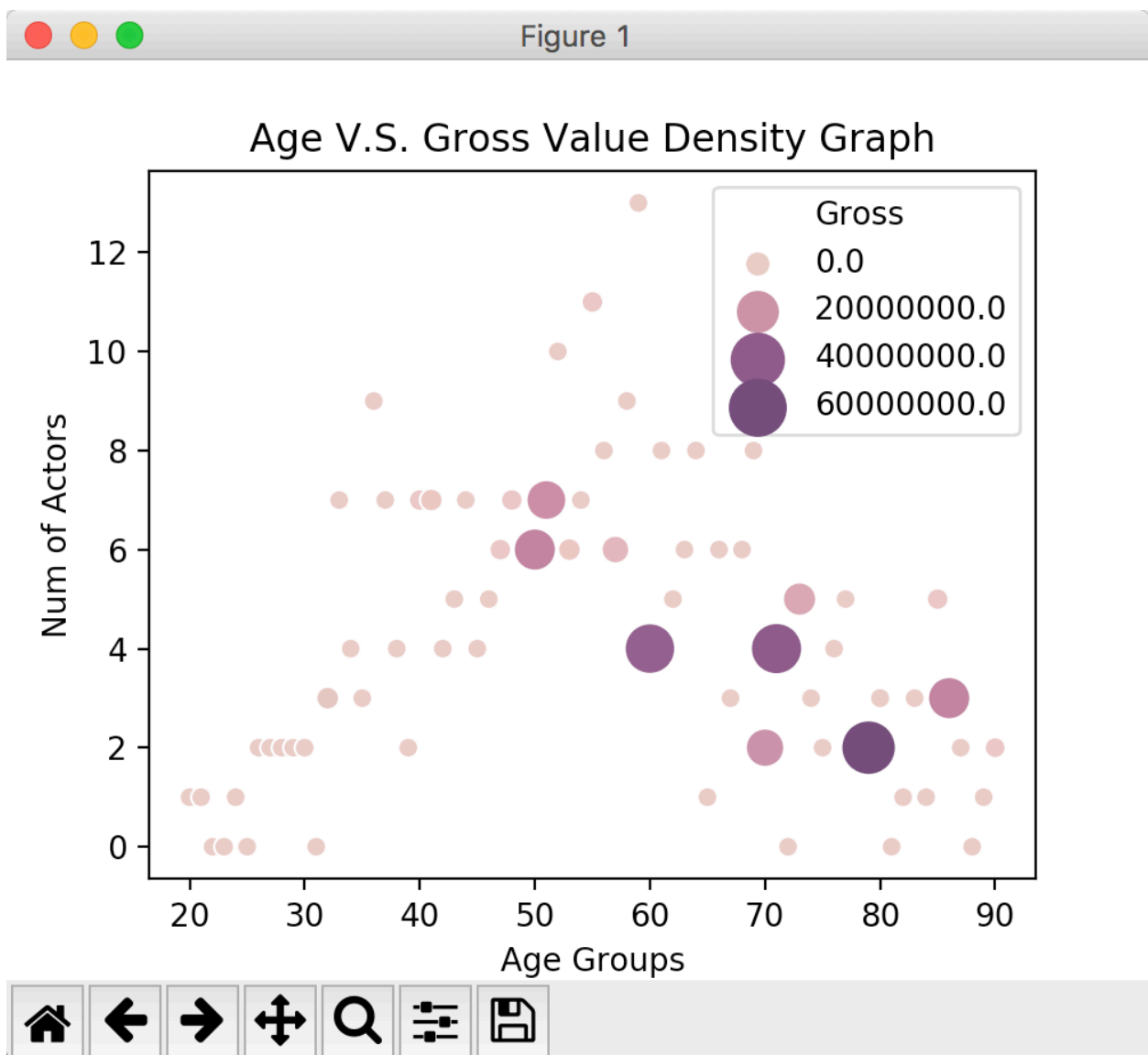


Result of enter 4:

```
4
['20+', '30+', '40+', '50+', '60+', '70+', '80+']
[22212084, 2216132, 178214618, 1027863196, 1005773688, 1001143378, 1150162485]

The age group that have the mac gross value is :80+
With total gross: 6
```

Result of enter 5:



API parts:

Running api.py:

```
* Serving Flask app "app" (lazy loading)
* Environment: production
  WARNING: Do not use the development server in a production environment.
  Use a production WSGI server instead.
* Debug mode: on
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
* Restarting with stat
* Debugger is active!
* Debugger PIN: 306-421-586
```

Open Post Man:

GET:

<http://localhost:5000/actor/all>

```
1 [
2   "Bruce Willis",
3   "Faye Dunaway",
4   "James Whitmore",
5   "David Dukes",
6   "Brenda Vaccaro",
7   "Martin Gabel",
8   "Anthony Zerbe",
9   "Paul Newman",
10  "Charlotte Rampling",
11  "Jack Warden",
12  "James Mason",
13  "Milo O'Shea",
14  "Kim Basinger",
15  "John Larroquette".
```

GET:

<http://localhost:5000/actor/James Whitmore>

```
1 {
2   "age": 87,
3   "json_class": "Actor",
4   "movies": [
5     "Command Decision",
6     "Battleground",
7     "Give 'em Hell, Harry!",
8     "Glory! Glory!",
9     "The Practice",
10    "Here's to Life!",
11    "Mister Sterling"
12  ],
13   "name": "James Whitmore",
14   "total_gross": 6269000
15 }
```

GET:
<http://localhost:5000/actor?name=Faye&age=76>

```
1 {  
2   "age": 76,  
3   "json_class": "Actor",  
4   "movies": [  
5     "The Happening",  
6     "Hurry Sundown",  
7     "Bonnie and Clyde",  
8     "The Thomas Crown Affair",  
9     "A Place for Lovers",  
10    "The Extraordinary Seaman",  
11    "The Arrangement",  
12    "Little Big Man",  
13    "Puzzle of a Downfall Child",
```

GET:
<http://localhost:5000/actor?name=Faye&age=76>

```
1 {  
2   "Brion James": {  
3     "age": -1,  
4     "json_class": "Actor",  
5     "movies": [],  
6     "name": "Brion James",  
7     "total_gross": 0  
8   },  
9   "James Coburn": {  
10    "age": 74,  
11    "json_class": "Actor",  
12    "movies": [  
13      "Ride Lonesome",  
14      "Face of a Fugitive",  
15      "The Magnificent Seven",  
16      "Hell Is for Heroes",  
17      "The Great Escape",
```

PUT:
<http://localhost:5000/actor/update/Charlotte Rampling> {"total_gross":199}

The screenshot shows an HTTP client interface with a PUT request to `http://localhost:5000/actor/update/Charlotte Rampling`. The request body is a JSON object: `{"total_gross":199}`. The interface includes tabs for Authorization, Headers (1), Body (selected), Pre-request Script, and Tests. The Body tab is active, showing the JSON payload. The request is set to be sent as raw JSON (application/json).

Result:

The screenshot shows the response of the PUT request. The status is 200 OK and the time taken is 84 ms. The response body is a JSON array containing an object with a success message and a status code: `[{"result": "Charlotte Rampling's info has been updated successfully!", "status": 201}]`. The interface includes tabs for Body (selected), Cookies, Headers (4), and Test Results. The Body tab is active, showing the JSON response in a pretty-printed format.

In Json data:

```
},
"Charlotte Rampling": {
  "json_class": "Actor",
  "name": "Charlotte Rampling",
  "age": 71,
  "total_gross": 199,
  "movies": []
},
```

POST:

```
http://localhost:5000/actor/add/name_new_1
{
  "age": 73,
  "json_class": "Actor",
  "movies": [],
  "name": "name_new_1",
  "total_gross": 0
}
```


POST http://localhost:5000/actor/add/name_new_1 Params Send Save

Authorization Headers (1) Body Pre-request Script Tests Code

☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary JSON (application/json)

```
1 {
2   "age": 73,
3   "json_class": "Actor",
4   "movies": [],
5   "name": "name_new_1",
6   "total_gross": 0
7 }
```

Result:

Body Cookies Headers (4) Test Results Status: 201 CREATED Time: 71 ms

Pretty Raw Preview JSON

```
1 {
2   "result": {
3     "age": 73,
4     "json_class": "Actor",
5     "movies": [],
6     "name": "name_new_1",
7     "total_gross": 0
8   }
9 }
```

In Json data:

```
},
  "name_new_1": {
    "age": 73,
    "json_class": "Actor",
    "movies": [],
    "name": "name_new_1",
    "total_gross": 0
  }
},
```

DELETE:

http://localhost:5000/actor/delete/name_new_1

The screenshot shows the 'Body' tab of a web browser's developer tools. The response is in JSON format, displayed in 'Pretty' view. The status is '200 OK' and the time taken is '62 ms'. The JSON data is:

```
{
  "result": "name_new_1 deleted successfully!"
}
```

In Json data:

The screenshot shows a code editor with the title 'name_new_1'. The JSON data is as follows:

```
{
  "result": "name_new_1 deleted successfully!"
}
```