Outline

- Data visualization lite

Cool media art pieces

- Project presentations

Prof. Angela Chang

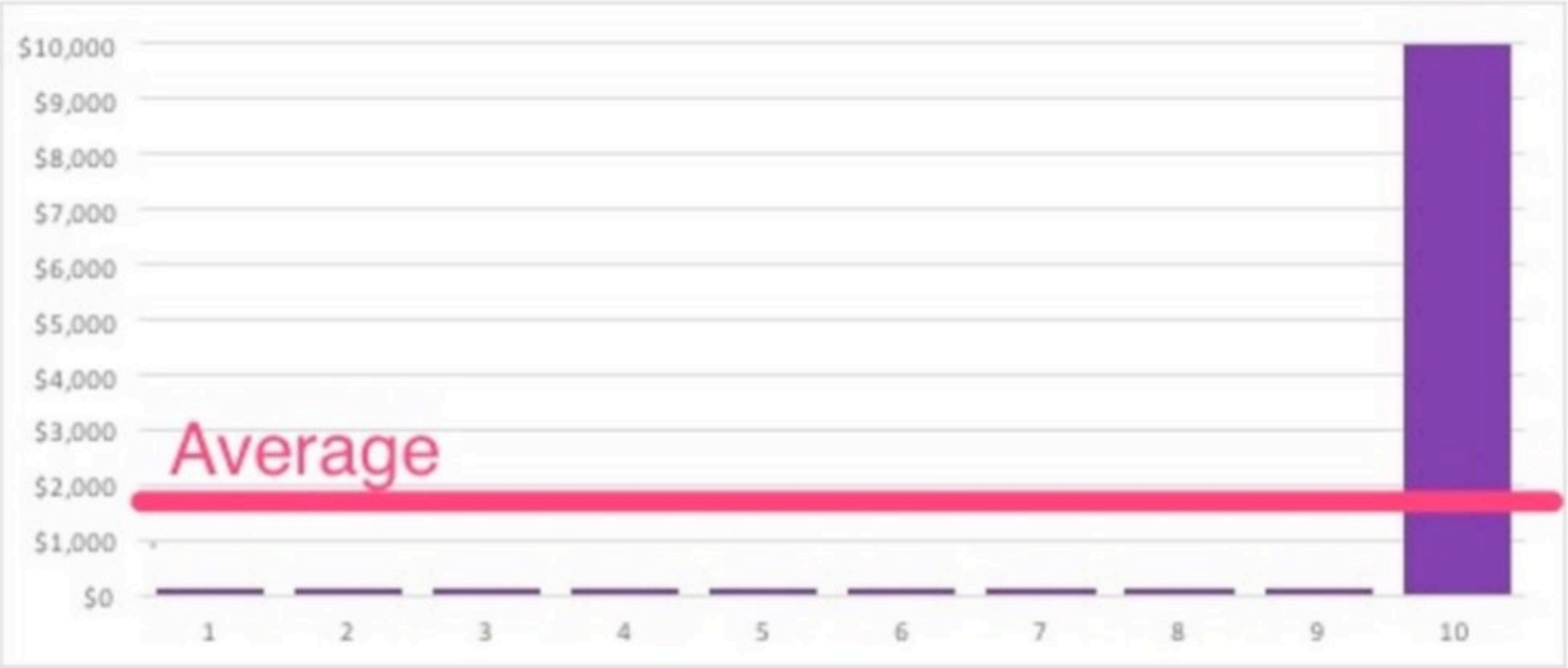
Lecture 24. Data Visualization (Mean, Median, and Mode)

Fall 2017. Dec 4

CODE, CULTURE, AND PRACTICE

Average income in the neighborhood

It’s nowhere near the middle!



http://idatassist.com/most-people-have-more-than-the-average-number-of-feet/

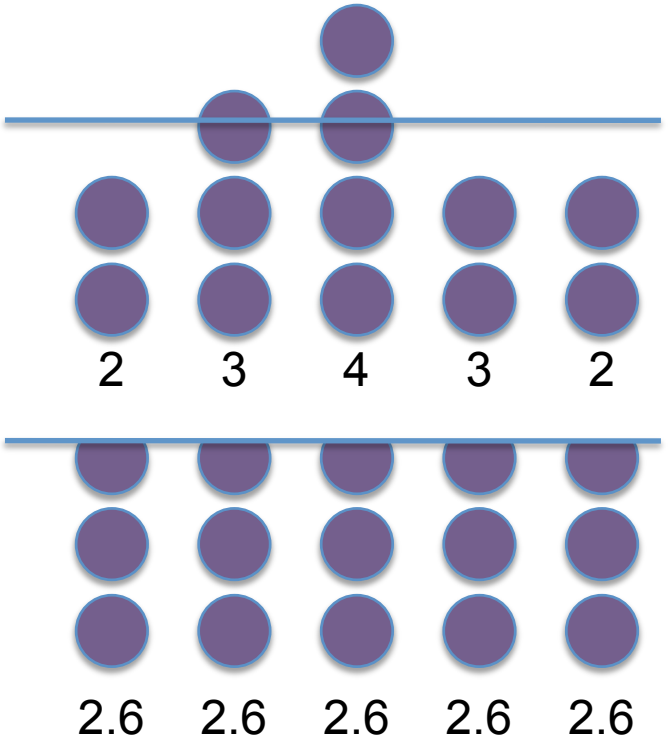
Average (arithmetic mean)

the sum divided by the number of data points

Total 2+3+4+2+2 = 13

5 data sets

e.g. 13 cookies were eaten by 5 people, how many did each person eat?



average = 13 / 5 = 2.6

2.6 across the board

“An average of 2.6 cookies were consumed per person.”

```
def mean(sequence):
    sum = 0.0
    for element in sequence:
        sum = sum + element
    return sum/len(sequence)
```

Statistics_Mean_Median_Mode.ipynb

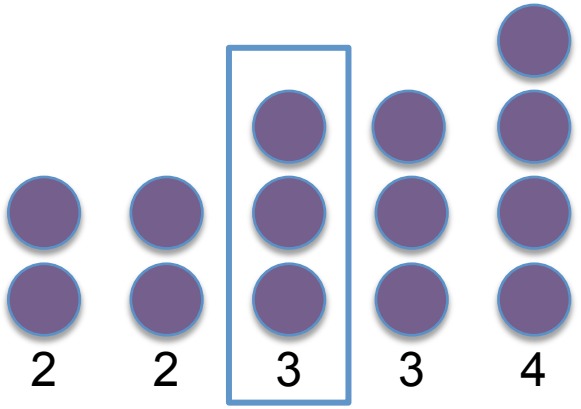
Median

"The middle point"

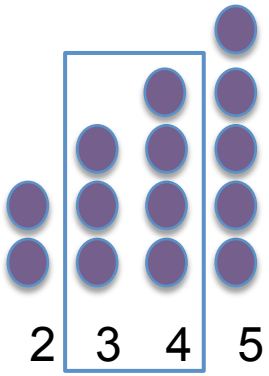
It's the point in the middle of the sorted data.

It's based on the *number* of data sets.

```
def median(sequence):
    sequence.sort()
    half= len(sequence)/2
    print len(sequence)%2
    if (len(sequence)%2 ==0):  #return 2 middle points
        return list([sequence[half-1], sequence[half]])
    else:
        return sequence[half] #return single middle point
```



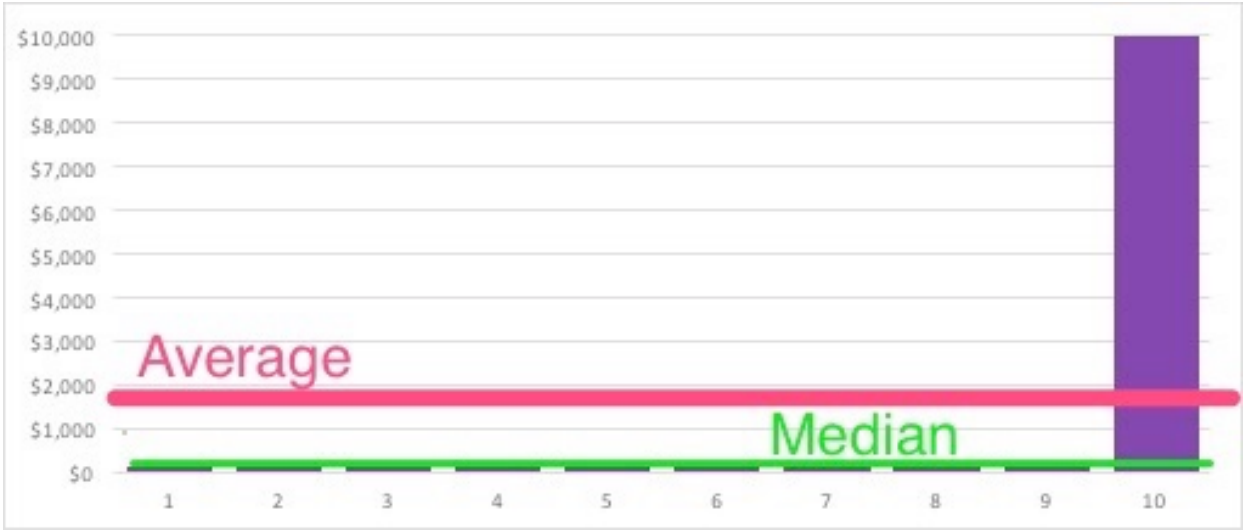
If you have sorted 5 data sets, the median is at data set #3.

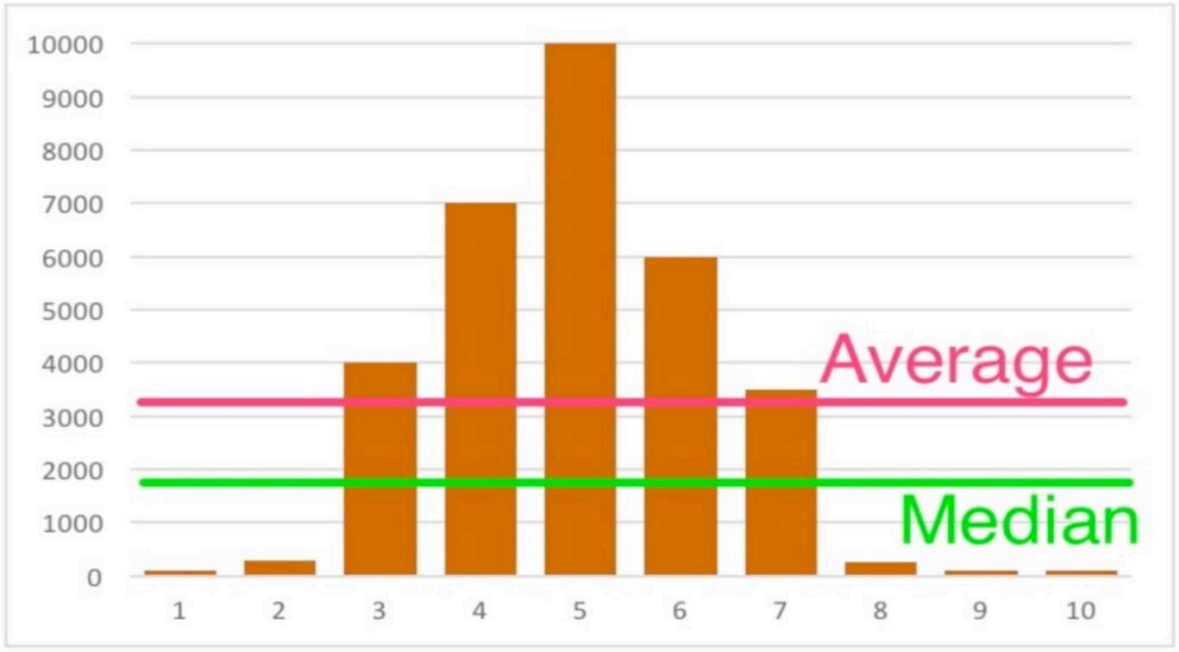


If you have an even number, of points there are two medians. e.g.3,4 above

Statistics_Mean_Median_Mode.ipynb

Median represents “middle” better

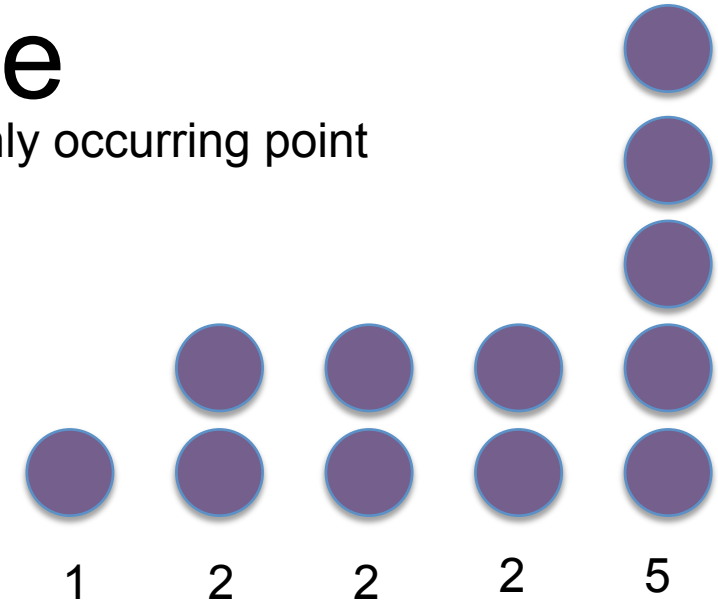




Mode

Most frequent or commonly occurring point

- 12 cookies were eaten by 5 people and *most* of the people ate 2 cookies. The mode is 2.
- Mode represents *popularity* better.




```
def mode(sequence):
    counted = []
    frequent = []
    sequence.sort()
    for num in sequence:
        counted.append( sequence.count(num))
    maximum_occurences = max(counted)
    for ind in range(len(counted)):
        if counted[ind] == maximum_occurences:
            if sequence[ind] not in frequent:
                frequent.append(sequence[ind])
    return frequent
```

Statistics_Mean_Median_Mode.ipynb

Use processing to visualize data

DataVisualization

```
values = [10, 9, 73, 25, 33, 76, 52, 9, 35, 86]

def mean(sequence):
    sum = 0.0
    for element in sequence:
        sum = sum + element
    return sum/len(sequence)

def median(seq):
    seq.sort()
    half = len(seq)/2
    if (len(seq)%2 == 0):
        return list([seq[half-1],seq[half]])
    else:
        return seq[half]

def mode(s):
    counted = []
    frequent = []
    for num in s:
        counted.append( s.count(num))
    print counted
    maximum_occurrences = max(counted)
    print maximum_occurrences
    for ind in range(len(counted)):
        if counted[ind] == maximum_occurrences:
            if s[ind] not in frequent:
                frequent.append(s[ind])
    return frequent
```

the code from the notebook is pasted above

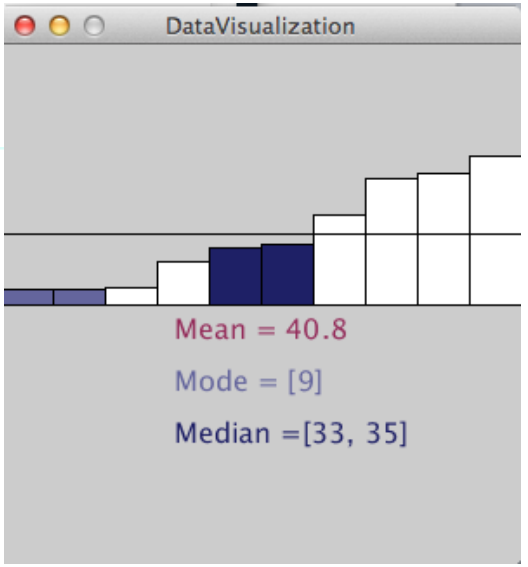
color palette

```
pal = { "avgcol": color(150,40,90),
        "medcol": color(30,30,100),
        "modcol": color(100,100,155) }

### Visualizing the Data
def setup():
    size(300,300)
    #Calculate the results
    avg = mean(values)
    mod = mode(values)
    med = median(values)
    #Draw each data point
    for i in range(len(values)):
        if (values[i] in mod):
            fill(pal["modcol"])
        elif (values[i] == round(avg)):
            fill(pal["avgcol"])
        elif (values[i] in med):
            fill(pal["medcol"])
        else:
            fill(255)
        rect( i * 30, height/2-values[i], 30, values[i] )
    #write the text
    textSize(16)
    fill(pal["avgcol"])
    stroke(2)
    rect(0, height/2-avg, 300, 0)
    text("Mean = "+ str(avg), 100, 170)
    fill(pal["modcol"])
    text("Mode = "+ str(mod), 100,200)
    fill(pal["medcol"])
    text("Median ="+str(med),100,230)
    print values
```

add code to loop over values and draw a rectangle

write text



DataVisualization.pyde

Average equatorial value

create a list for looping over planets

list of planets

```
planetList = [mercuryInfo,venusInfo,earthInfo,marsInfo]
print planetList[2]['name']
print planetList[3]['knownMoons']

radiusSum = 0
for i in range(len(planetList)):
    radiusSum += planetList[i]['eqRadiusKm']
print radiusSum / len(planetList)
# or use the iterator syntax special to python
radiusSum = 0
for p in planetList:
    radiusSum += p['eqRadiusKm']
print radiusSum / len(planetList)
```

element at index 2, which is a dictionary for Venus

the value for that key name in the dictionary (Earth)

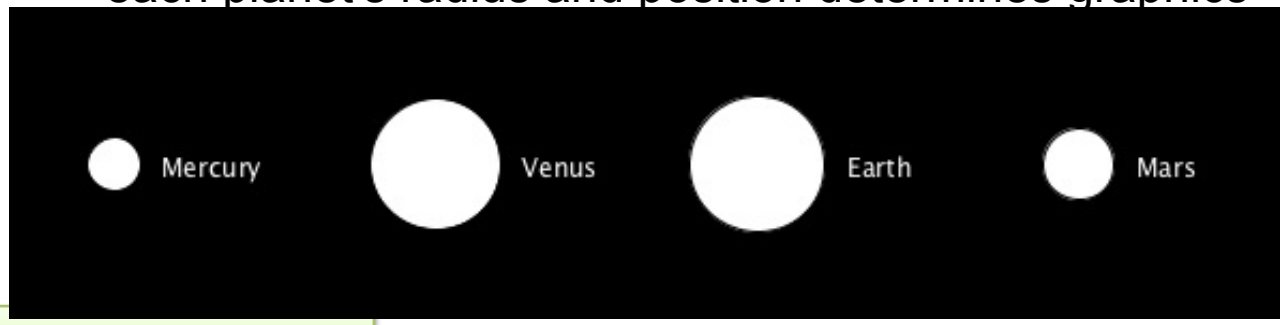
planet dictionary

```
mercuryInfo = {
    "name": "Mercury",
    "knownMoons": 0,
    "eqRadiusKm": 2439.64,
}
venusInfo = {
    "name": "Venus",
    "knownMoons": 0,
    "eqRadiusKm": 6051.59,
}
earthInfo = {
    "name": "Earth",
    "knownMoons": 1,
    "eqRadiusKm": 6387.1,
}
marsInfo = {
    "name": "Mars",
    "knownMoons": 2,
    "eqRadiusKm": 3397.0
}
```

FourPlanets_List.pyde

Visualizing planets (JSON)

each planet's radius and position determines graphics



in **`setup()`** load your data

```
planetList = [
    {'name': 'Mercury', 'eqRadiusKm': 2439.64},
    {'name': 'Venus', 'eqRadiusKm': 6051.59},
    {'name': 'Earth', 'eqRadiusKm': 6387.1},
    {'name': 'Mars', 'eqRadiusKm': 3397.0},
]
```

in **`draw()`** loop through and draw each element

```
planetCount = len(planetList)
for i in range(planetCount):
    #scale radius to be screen friendly
    planetRadius = planetList[i]['eqRadiusKm'] * 0.01
    offset = 50 + ((width/planetCount) *i)
    ellipse(offset,height/2, planetRadius, planetRadius)
    text(planetList[i]['name'], 10+offset+(planetRadius/2), height/2)
```

x offset increases by index position

draw planet

write name

Tip: To load json files, **`import json`** library and use **`json.load()`**

ThePlanets.pyde

Reading csv data from a file

comma separated values data file (in data folder)

ortiz.csv

```
1997,1,6,0.327
1998,9,46,0.277
1999,0,0,0
2000,10,63,0.282
2001,18,48,0.234
2002,20,75,0.272
2003,31,101,0.288
2004,41,139,0.301
2005,47,148,0.3
2006,54,137,0.287
2007,35,117,0.332
2008,23,89,0.264
2009,28,99,0.238
2010,32,102,0.27
2011,29,96,0.309
2012,23,60,0.318
2013,30,103,0.309
2014,35,104,0.263
```

HomeRuns_ReadCSV

use csv library, to open the data file

```
import csv

statsFileHandle = open('ortiz.csv')
statsData = csv.reader(statsFileHandle)
for row in statsData:
    year = row[0]
    homeRuns = row[1]
    rbi = row[2]
    average = row[3]
    print year, homeRuns, rbi, average
```

HomeRuns_ReadCSV

Note: CSV data is read in ‘string’ format.
Convert to numbers for math, use the int() or float()

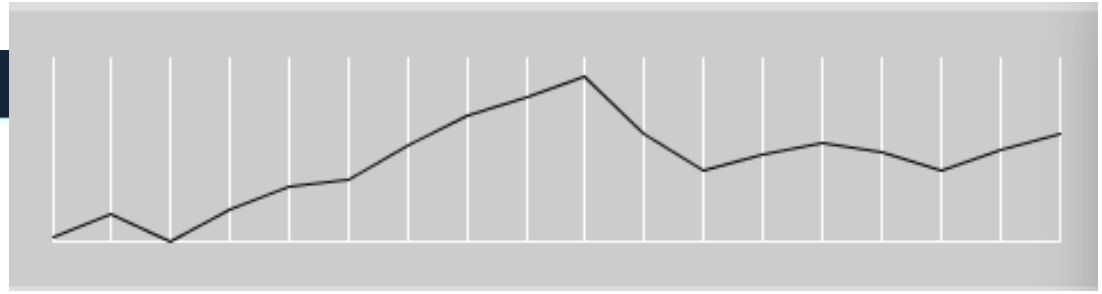
```
homeRunTotal = 0
for row in statsData:
    homeRunTotal += int(row[1])

print homeRunTotal
```

HomeRunsCalculation_CSV

Visualizing data

create a list from the data and then draw it



1997-2014 batting statistics for David Ortiz, Red Sox

HomeRunsTablefromCSV

```
    for row in statsData:
        homeRuns.append(int(row[1]))
    print homeRuns

def draw():
    background(204)
    #Draw background grid for data
    stroke(255)
    line(20, 100, 20, 20)
    line(20, 100, 460, 100)
    for i in range(len(homeRuns)):
        x = map(i, 0, len(homeRuns)-1, 20, 460)
        line(x, 20, x, 100)
    #Draw lines based on home run data
    noFill()
    stroke(0)
    beginShape()
    for i in range(len(homeRuns)):
        x = map(i, 0, len(homeRuns)-1, 20,460)
        y = map(homeRuns[i], 0, 60, 100, 20)
        vertex(x,y)
    endShape()
```

y axis

x axis

loop through the data,
space it out across the width of the window
draw a vertical line

loop through the data,
at each vertical line
map the y value to height of the display (note y is reversed)
draw a connected point at x,y

HomeRunsTablefromCSV

Class participation for today

play with visualization

Modify one of the visualization examples from today's lecture notes by:

- altering the data
- tweaking the visualization parameters (e.g. colors, size, etc.)

---OR----

- Work on your final projects: submit a sample of data from your project
- http://bit.ly/ModData2017

Final Playsentation day

Wednesday Dec. 13th 2017

What's expected by this date:

- prototype of idea that people can experience
- 2-slides describing your project with
  title, your names, project description, a picture

Afterwards

1. code archive (github or zip file)
2. playable demo
3. writeup of what you did, issues you ran into (3 pages)

Homework

1

Work on final projects
and upload any comments or questions you are having

what can I help with, send me code with problems!

2

Class participation today:
upload a modification of a data visualization
upload an example of your data from your final project
http://bit.ly/ModData2017

Summary of today

- Technical practice, reading data formats
  - CSV
  - JSON
  - visualizing data (work from examples)

For Inspiration on your final projects, check out
http://www.coolhunting.com/tag/Data%20Visualization
http://flowingdata.com
http://benfry.com