HW3 (Score: 28.5 / 30.0)

1. Task (Score: 5.0 / 5.0)
2. Task (Score: 10.0 / 10.0)
3. Task (Score: 13.5 / 15.0)
4. Comment

DATA 601: HW3

Fall 2019

Due: Wed. Oct. 2, 2019 (by 23:55)

Learning Objectives

- Work with realworld datasets that can be represented using tabular data structures.
- Gain experience wrangling and organizing data using `pandas`.
- Produce visualizations summarizing information from tabular data.

This is an individual homework assignment.

Please complete this homework assignment within the Jupypter notebook environment, making use of Markdown and Code cells to properly format your answers. Please provide solutions where asked.

Your completed Jupyter notebook is to be submitted via the HW2 dropbox on D2L.

Warm up

- Please review class slides on `pandas`.
- Please also review the Calgary Rainfall Jupyter notebook. In this homework, we will use the Calgary Rainfall dataset. Please download the dataset if you already haven't done so. You may use the data provided in class or download it directly from Open Calgary (https://data.calgary.ca/Environment/Historical-Rainfall/d9kv-swk3).

(Top)

Task 1 (5 points)

Cleanup and organization

- Use `pandas` to read in the data set. Do not discard the datetime information in the columns. Convert the 'TIMESTAMP' column to a `datetime` object (You can use `pandas.to_datetime()` (https://pandas.pydata.org/pandas-docs/stable/generated/pandas.to_datetime.html) to accomplish this).
- You may notice that 'YEAR' column is now redundant. Additionally, for this homework, we won't make use of the 'ID' column. Please discard it (you can use `pandas.DataFrame.drop` (https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.drop.html) for this). You can also discard any rows where the channel is not active.
- Display the head (`pandas.DataFrame.head` (https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.head.html)), tail (`pandas.DataFrame.tail` (https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.tail.html)) and description (`pandas.DataFrame.describe` (https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.describe.html)) of the resulting dataframe.

Please provide your solution by inserting appropriate code and markdown cells below this cell.

In [1]:

```
import pandas as pd
import numpy as np
import datetime as dt

import matplotlib as mpl
import matplotlib.pyplot as plt
mpl.style.use('ggplot')
```

In [2]:

```
# Import data as 'rainfall'
rainfall = pd.read_csv("Historical_Rainfall.csv")

# Check format of TIMESTAMP by viewing the head of the data
display(rainfall[0:6:])

# Convert TIMESTAMP to datetime object using the format '%Y/%m/%d %H:%M:%S %p'
rainfall['TIMESTAMP'] = pd.to_datetime(rainfall['TIMESTAMP'], format='%Y/%m/%d %H:%M:%S %p')

# Check the info of the rainfall data to ensure the TIMESTAMP column has converted to a datetime object
rainfall.info()
```

	CHANNEL	YEAR	TIMESTAMP	RAINFALL	RG_ACTIVE	ID
0	42	2019	2019/09/14 05:05:00 AM	0.2	Y	2019-09-14T05:05:00-42
1	42	2019	2019/09/14 01:45:00 AM	0.4	Y	2019-09-14T01:45:00-42
2	42	2019	2019/09/14 01:40:00 AM	0.4	Y	2019-09-14T01:40:00-42
3	48	2019	2019/09/13 10:45:00 AM	0.2	Y	2019-09-13T10:45:00-48
4	21	2019	2019/09/13 10:40:00 AM	0.2	Y	2019-09-13T10:40:00-21
5	2	2019	2019/09/12 05:40:00 PM	0.2	Y	2019-09-12T17:40:00-02

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 889780 entries, 0 to 889779
Data columns (total 6 columns):
CHANNEL      889780 non-null int64
YEAR         889780 non-null int64
TIMESTAMP    889780 non-null datetime64[ns]
RAINFALL     889780 non-null float64
RG_ACTIVE    889780 non-null object
ID           889780 non-null object
dtypes: datetime64[ns](1), float64(1), int64(2), object(2)
memory usage: 40.7+ MB
```

In [3]:

```
# Drop columns using pandas.DataFrame.drop
rainfall_dropped = rainfall.drop(['YEAR', 'ID', 'RG_ACTIVE'], axis = 1)
```

In [4]:

```
#Display the head, tail and description
display(rainfall_dropped.head())
display(rainfall_dropped.tail())
display(rainfall_dropped.describe())
```

	CHANNEL	TIMESTAMP	RAINFALL
0	42	2019-09-14 05:05:00	0.2
1	42	2019-09-14 01:45:00	0.4
2	42	2019-09-14 01:40:00	0.4
3	48	2019-09-13 10:45:00	0.2
4	21	2019-09-13 10:40:00	0.2

	CHANNEL	TIMESTAMP	RAINFALL
889775	8	1988-05-20 06:45:00	0.2
889776	21	1988-05-20 06:45:00	0.2
889777	8	1988-05-20 06:25:00	0.2
889778	15	1988-05-20 06:15:00	0.2
889779	17	1988-05-20 12:55:00	0.2

	CHANNEL	RAINFALL
count	889780.000000	889780.000000
mean	17.493284	0.317491
std	11.116896	0.418900
min	1.000000	0.100000
25%	8.000000	0.200000
50%	16.000000	0.200000
75%	26.000000	0.200000
max	99.000000	43.200000

(Top)

Task 2 (10 points)

Restructure and determine rainfall daily totals per channel

- We are interested in the daily rainfall totals per channel. Restructure and aggregate your table so that entries now contain *daily totals per channel*.

The precise details of how you accomplish this are up to you. You can for example build a hierarchical index for the rows with the year, month and day. You can also have a hierarchical index on the columns based on the channels. Please make use of pandas grouping and aggregation facilities to accomplish this.

Please provide your solution by inserting appropriate code and markdown cells below this cell.

In [5]:

```
# Create DAY, MONTH and YEAR columns
rainfall_dropped['YEAR'] = rainfall_dropped['TIMESTAMP'].dt.year
rainfall_dropped['MONTH'] = rainfall_dropped['TIMESTAMP'].dt.month
rainfall_dropped['DAY'] = rainfall_dropped['TIMESTAMP'].dt.day

# Group by 'YEAR', 'MONTH', 'DAY' and 'CHANNEL' and use the sum function to find daily rainfall totals
rainfall_daily_total = rainfall_dropped.groupby(['YEAR', 'MONTH', 'DAY', 'CHANNEL']).sum()
display(rainfall_daily_total.head())
```

				RAINFALL
YEAR	**MONTH**	**DAY**	**CHANNEL**	
			8	0.4
			15	0.2
1988	5	20	17	0.2
			18	0.2
			19	0.2

(Top)

Task 3 (15 points)

Visualization

Produce visualizations that show:

- Rainiest day of the year for the years 1989 through 2018. For each day, show the date and the total rainfall. Note that you will need to aggregate over the channels. Use the maximum over the channels for this, i.e. the channel that recorded the most rainfall.
- Average number of rainy days per month. Average over the years 1989 through 2018.
- Rainfall *monthly* statistics such as mean, median, min and max. Aggregate over the years 1989 through 2018 (years for which the data is complete). You will need to aggregate over the channels as well.

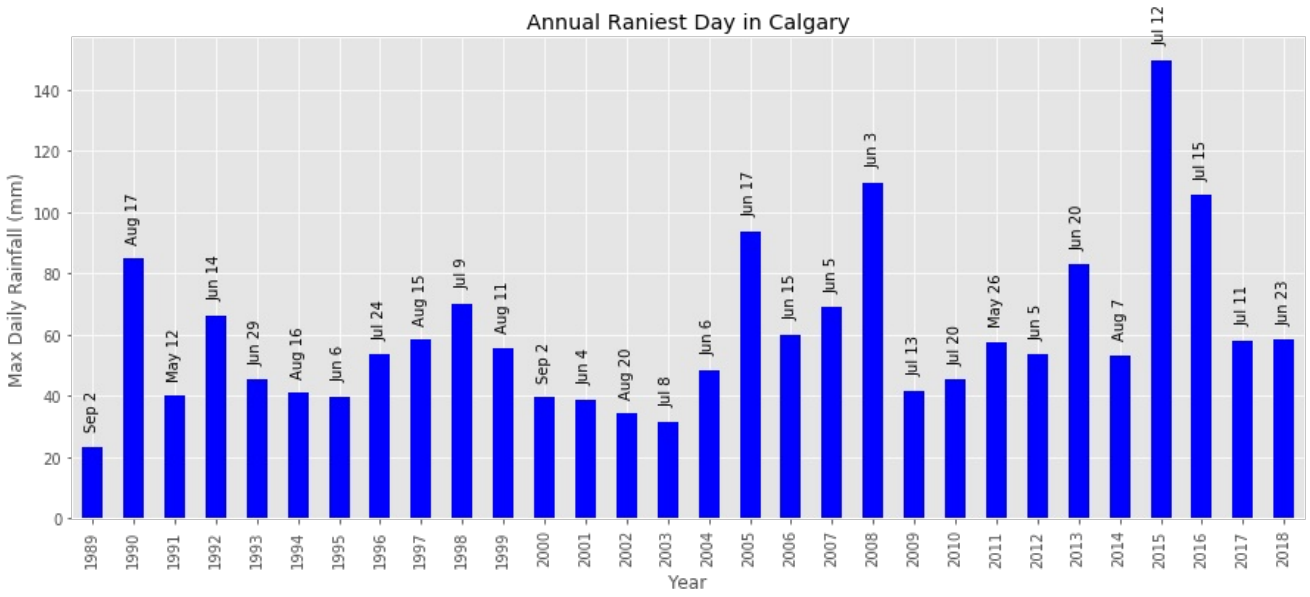
The details of what visualizations to use are not spelled out. Please choose a visualization that is appropriate for each of the above tasks and *clearly* shows the requested information. Please also ensure that you provide appropriate labels/legends/colorbars so that your visualizations are readable and self-contained.

Please provide your solution by inserting appropriate code and markdown cells below this cell.

Comments:
The x-axis of your second and third graphs use numbers in place of month names. These require some mental juggling to translate, and so you should add a few lines of code to convert the numbers to strings.
Bonus points for the comments and citations, though.

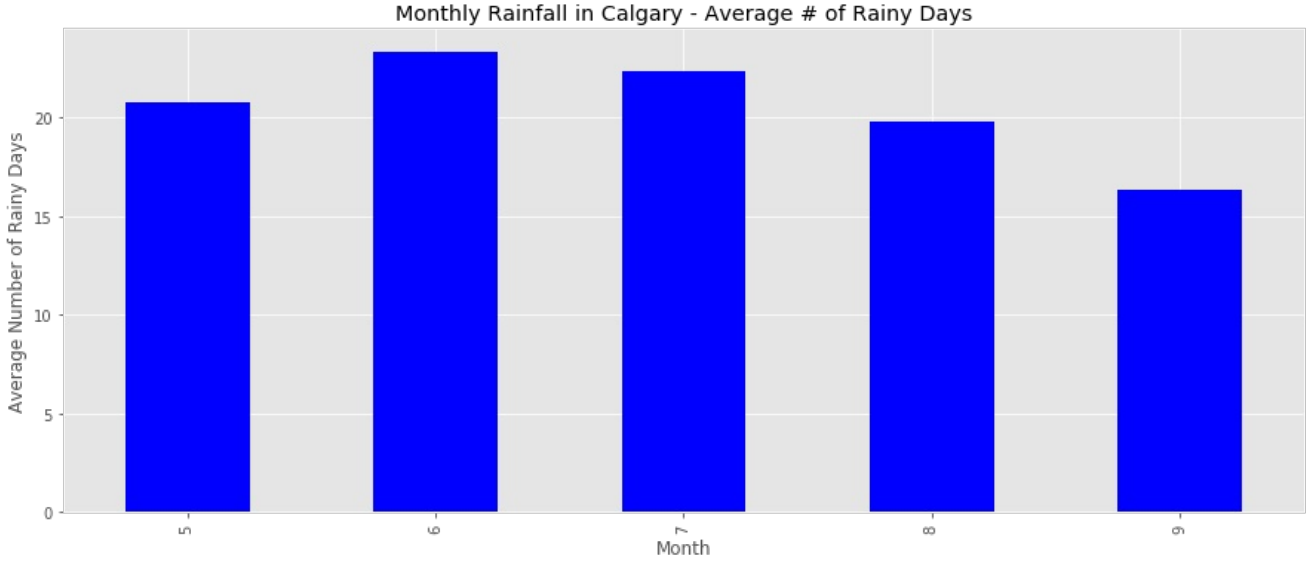
In [6]:

```
# Create daily rainfall total object without 2019 and 1988 data
rainfall_daily_total_2018 = rainfall_dropped.groupby(['YEAR', 'MONTH', 'DAY', 'CHANNEL']).sum().loc[1989:2018]

# Determine yearly max rainfall amounts
rainfall_yearly_max = rainfall_daily_total_2018.groupby(['YEAR']).max()

# Determine which day the yearly max rainfall amounts occurs on
find_yearly_max_day = pd.merge(rainfall_yearly_max, rainfall_daily_total_2018.reset_index(), on = ['YEAR', 'RAINFAL
L'], how = 'left')

# Create a list of numeric pairs that represent the month and day of the yearly max rainfall
yearly_max_day_labels_list = list(zip(find_yearly_max_day['MONTH'], find_yearly_max_day['DAY']))

# Create labels to be used in bar chart
labels = [dt.date(1900, m, 1).strftime('%b') + ' ' + str(d) for m,d in yearly_max_day_labels_list]

# Create figure
fig = plt.figure()
axis = fig.add_subplot(1,1,1)
yearly_bar = rainfall_yearly_max.plot(kind='bar', color='b', grid=True, ax = axis)
yearly_bar.set_ylabel("Max Daily Rainfall (mm)")
yearly_bar.set_xlabel("Year")
yearly_bar.set_title('Annual Raniest Day in Calgary')
axis.get_legend().remove()

# Label bars with labels list
## Borrowed code from stackoverflow:
## https://stackoverflow.com/questions/28931224/adding-value-labels-on-a-matplotlib-bar-chart

rects = yearly_bar.patches
for rect, label in zip(rects, labels):
    height = rect.get_height()
    yearly_bar.text(rect.get_x() + rect.get_width() / 2,
                    height + 5,
                    label,
                    ha = 'center',
                    va = 'bottom',
                    rotation = 90)

fig.set_size_inches(15,6)
plt.show()
```



In [7]:

```
# Record daily rainfall max using the 'max' function over all the channels per day
rainfall_daily_max = rainfall_daily_total_2018.groupby(['YEAR', 'MONTH', 'DAY']).max()

# Group by 'YEAR' and 'MONTH' to get counts of monthly rainfall days
# followed by unstack which will ungroup the 'YEAR' and 'MONTH' to allow
# the mean function to average the monthly rainfall days from 1988 to 2018
monthly_average_rainy_days = rainfall_daily_max.groupby(['YEAR', 'MONTH']).count().unstack().mean()

# Plot the data
fig = plt.figure()
axis = fig.add_subplot(1,1,1)
monthly_bar = monthly_average_rainy_days.unstack().transpose().plot(kind='bar', color='b', grid=True, ax = axis)
monthly_bar.set_ylabel("Average Number of Rainy Days")
monthly_bar.set_xlabel("Month")
monthly_bar.set_title('Monthly Rainfall in Calgary - Average # of Rainy Days')
fig.set_size_inches(15,6)
axis.get_legend().remove()
plt.show()
```

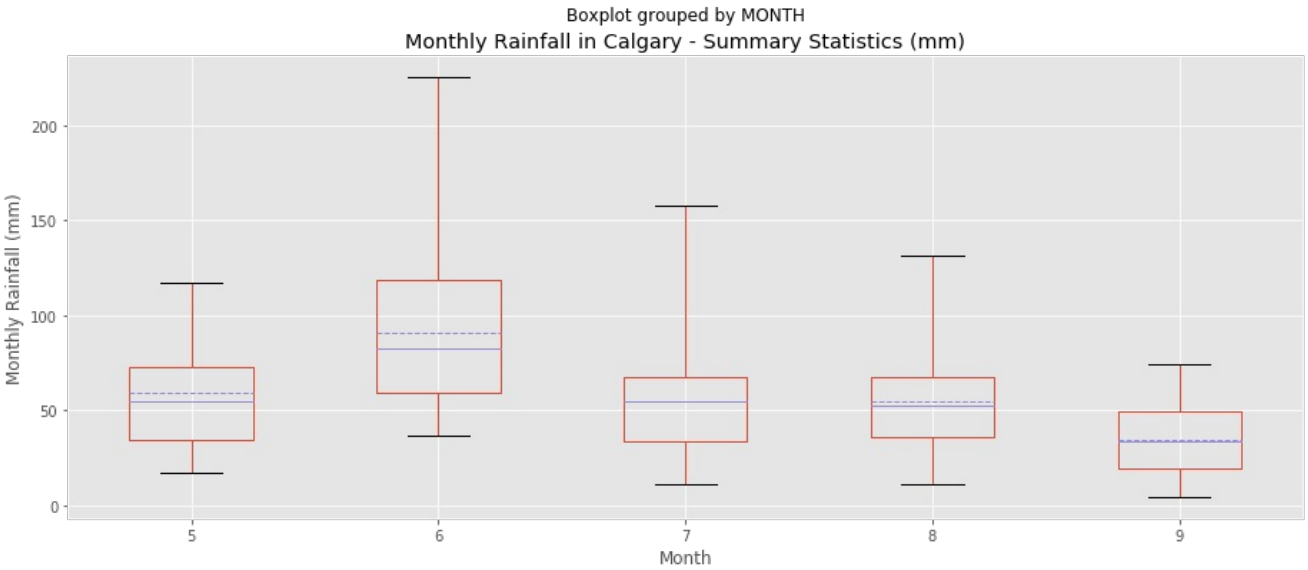


In [8]:

```
# Begin by using the rainfall_daily_total_2018 that includes total daily rainfall per channel from 1989 to 2018

# Sum the monthly rainfall at each channel
rainfall_monthly_per_channel = rainfall_daily_total_2018.groupby(['YEAR', 'MONTH', 'CHANNEL']).sum()

# Average the monthly rainfall at each channel to get a monthly rainfall average
rainfall_monthly = rainfall_monthly_per_channel.groupby(['YEAR', 'MONTH']).mean()

# Check to see if there are any months with any NaNs recorded
display(rainfall_monthly.unstack())

# It looks good. Use the describe function to check our summary statistics
stats = rainfall_monthly.groupby(['MONTH']).describe()
display(stats)

# Plot the summary statistics using 'boxplot' and double check with summary statistics
box_plot = rainfall_monthly.boxplot(by = 'MONTH',
                                    whis = 'range',
                                    meanline = True,
                                    showmeans = True,
                                    figsize = (15, 6))
box_plot.set_xlabel('Month')
box_plot.set_ylabel('Monthly Rainfall (mm)')
box_plot.set_title('Monthly Rainfall in Calgary - Summary Statistics (mm)')

# I couldn't figure out the fancy formating :(
# https://matplotlib.org/3.1.1/gallery/statistics/boxplot_color.html
# The sample notebook worked in my environment so it's something about my boxplot that
# is causing the error. Could it be because I am using a dataframe versus an array?

plt.show()
```

	RAINFALL				
MONTH	5	6	7	8	9
YEAR					
1989	24.900000	36.240000	11.071429	47.153846	34.123077
1990	71.052632	47.800000	55.345455	62.947826	4.352381
1991	73.663636	95.041667	31.765217	39.704348	21.260870
1992	38.086957	171.563636	56.295652	35.834783	39.491667
1993	64.150000	112.058333	67.541667	86.658333	21.275000
1994	67.108333	67.125000	20.441667	81.633333	7.833333
1995	86.172414	55.068966	87.455172	46.227586	36.848276
1996	51.675862	59.951724	55.200000	33.751724	47.758621
1997	104.786207	96.379310	18.785714	65.807143	35.207143
1998	64.020690	95.455172	87.875862	21.924138	18.414286
1999	37.978571	82.907143	98.628571	61.770370	12.315385
2000	23.733333	76.992000	49.460870	48.903704	48.496296
2001	17.814286	89.478571	35.828571	11.078571	13.200000
2002	28.515385	48.275862	28.868966	67.337931	53.324138
2003	29.924138	63.613793	32.862069	26.213793	34.020690
2004	53.900000	65.735714	67.358621	56.427586	30.668966
2005	16.888889	225.928571	21.100000	83.192308	61.038095
2006	35.524138	131.289655	50.717241	55.634483	67.986207
2007	100.860000	120.600000	24.371429	50.857143	53.421429
2008	114.870968	125.606452	60.493750	48.918750	27.125000
2009	34.188235	40.700000	75.188235	77.223529	4.141176
2010	64.105556	69.794444	58.061111	54.505556	61.227778
2011	116.935294	82.177143	80.862857	67.976471	8.329412
2012	56.122222	145.422222	41.788889	36.554286	6.508571
2013	114.858824	133.685714	57.542857	17.262857	49.560000
2014	105.937143	132.683333	54.062857	88.028571	66.434286
2015	41.410811	58.821622	76.345946	131.672222	73.940541
2016	65.263415	46.785000	157.952381	77.728571	28.900000
2017	42.342857	46.985714	46.847619	21.933333	22.057143
2018	32.962791	93.009091	36.750000	29.895455	33.213636

	RAINFALL							
	count	mean	std	min	25%	50%	75%	max
MONTH								
5	30.0	59.325119	31.060460	16.888889	34.522211	55.011111	73.010885	116.935294
6	30.0	90.572528	43.380681	36.240000	59.104147	82.542143	118.464583	225.928571
7	30.0	54.895689	29.800217	11.071429	33.603695	54.631429	67.495905	157.952381
8	30.0	54.491952	26.030515	11.078571	36.014658	52.681349	67.816836	131.672222
9	30.0	34.082447	20.572835	4.141176	19.125932	33.617163	49.294074	73.940541


Boxplot grouped by MONTH
Monthly Rainfall in Calgary - Summary Statistics (mm)
Monthly Rainfall (mm)
200
150
100
50
0
5
6
7
8
9
Month

In [ ]: