Data 8R
Summer 2017

Plotting Functions

Discussion 7: July 20, 2017

1 Midterm Review

Question 4 Part 1, Question 5 Part 1, Question 8.

2 Review: Defining Functions

Define the following functions, according to what their docstrings say:

```
def feet_to_cm(feet):
    """
    1 foot is 30.48 cm.
    """


    return feet * 30.48



def first_and_last(array)
    """
    Given an array, returns a new array with only the first and last element
    """


    return make_array(array.item(0), array.item(len(array)-1))



def max_diff(array):
    """
    Given an array, return the biggest difference between adjacent elements of the array.
    >>> max_diff(make_array(10, 3, 5, 6))
    7
    """


    return max(np.diff(array))



def percent_change(initial, new):
    """
```

```
    Returns the percentage change between the initial and new values.
    >>> percent_difference(40,50)
    25.0
    """


    difference = new - initial
    return 100 * difference / initial
```

```
def challenge(table):
    """
    Given a table that has the columns "initial" and "new", return a new table with an additional column, "
     percent change".
    """
```

change = table.apply(percent$_c$*hange*,"*initial*","*new*")*new*$_t$*able* = *table.with*$_c$*olumn*("*percentchange*", *change*)*returnnew*$_t$*able*

3 Applying Functions onto Tables

Imagine that we have a table about physics data called **physics_data**:

Time	Distance (ft)
1	5
2	5
9	8
...	...

In physics calculations, we often want to have the data in terms of centimeters. Create a table called **cm_table** that has the original data and a new column called **Distance (cm)**.

```
cm_table = ...
```

```
= physics_data.with_column("Distance (cm)", physics_data.apply(feet_to_cm, "Distance (ft)"))
```

Now, calculate the average velocity by defining a general function that calculates velocity, and then applying that function onto the table: *Hint: Velocity = distance / time*

```
def velocity(distance, time):
```

return distance / time

```
table_with_velocity = ...
```

cm$_t$*able.with*$_c$*olumn*("*velocity*", *cm*$_t$*able.apply*(*velocity*,"*distance*","*time*"))