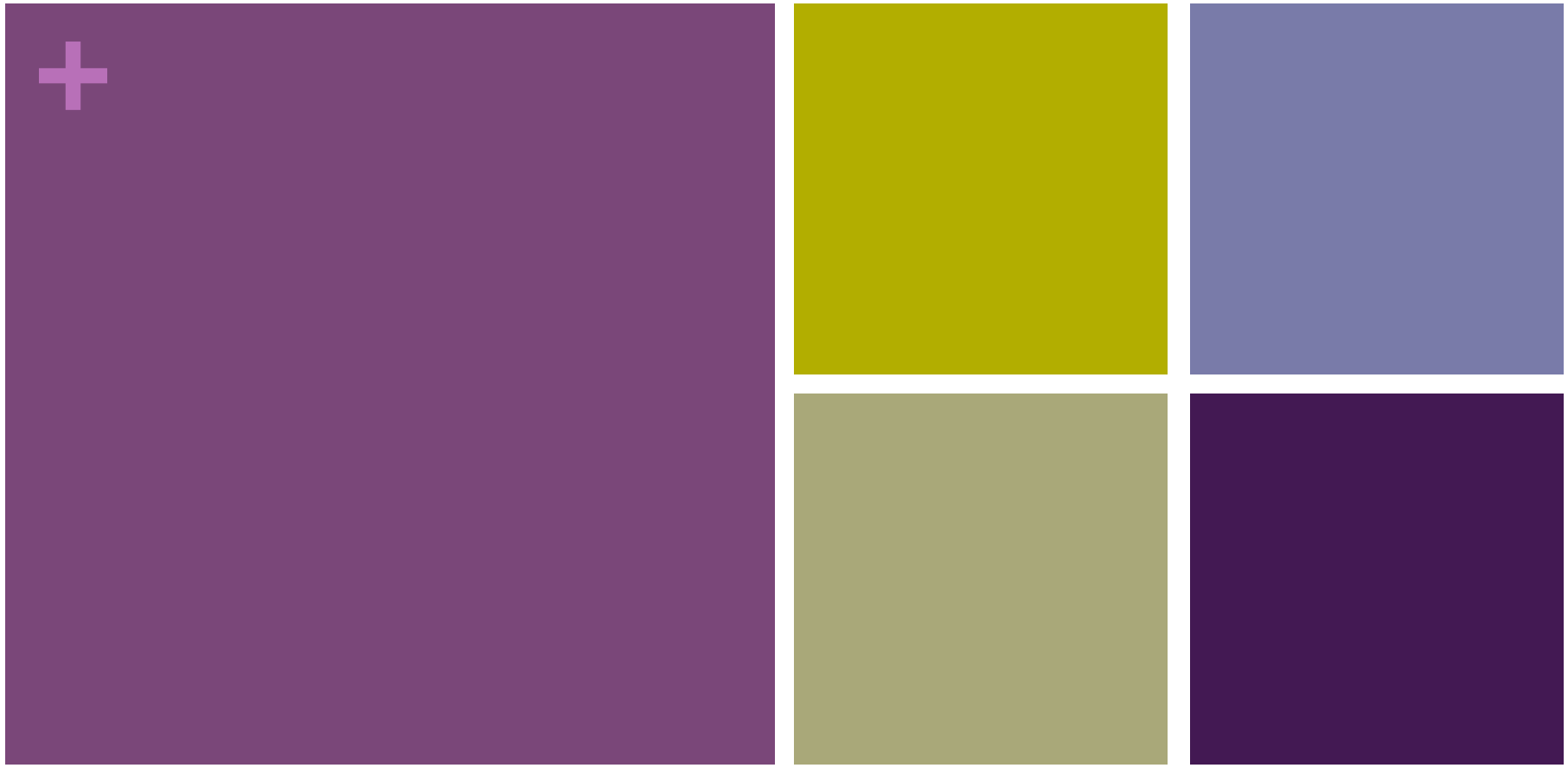


# Sorting for WordClouds

# Text Processing

## Data Visualization Process

- Acquire - Obtain the data from some source
- Parse - Give the data some structure, clean up
- Filter - Remove all but the data of interest
- Mine - Use the data to derive interesting properties
- Represent - Chose a visual representation
- Refine – Improve to make it more visually engaging
- Interact - Make it interactive

## Text Visualization

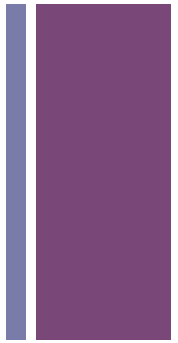
- Source = Document
- Parse = Words
- Filter = Word Set with counts
- Mine = Get relevant words
- Represent = Fonts/Placement
- Refine/Interact

# Acquire data: Source = Document

```
// Sketch 7-1: Parsing an input text file
String inputTextFile = "Obama.txt";
String [] fileContents;
fileContents = loadStrings(inputTextFile);
```

- fileContents has the source!
- What next?

# Parse



- How do we turn `fileContents` into words?
- join array into one long string

```
String rawText;
rawText = join(fileContents, " ");
```

- make all same case

```
rawText = rawText.toLowerCase();
```

- remove symbols and split string into words

```
String delimiters = " ,./?<>;:'\"[{]}\\|=+-_()*&^%$#@!~";
tokens = splitTokens(rawText, delimiters);
```

# Display the words

- Let's start by displaying all of the words:

```
for (String t : tokens) {
    //textSize(15);
    if(random(100) > 40) { // more red than green
      fill(random(150,250),0, 0,190); // make red
    } else {
      fill(0,random(150,250), 0,190); // make green
    }
    text(t, random(0,width-50), random(20,height));
} // for
```

# Count the words (second way)

- Use a HashMap ( a dictionary from words → counts)
- ```
  HashMap <String,Integer> wordCountSet =
        new HashMap<String,Integer>();
  ```
- to add a new word:
  - ```
    wordCountSet.put(word,1); // initial count is 1
    ```
- to get the frequency of a word:
  - ```
    Integer frequency =
          wordCountSet.get(word); // if null, then none
    ```
- to update the frequency of a word:
  - ```
    wordCountSet.put(word, frequeny + 1);
    ```

# Count the words (second way)

```
HashMap<String,Integer> wordCountSet =
   new HashMap<String,Integer>();

for (String t : tokens) {
  Integer wordFrequency = wordCountSet.get(t);
  if(wordFrequency != null) {
    wordCountSet.put(t,wordFrequency+1);
  } else {
    wordCountSet.put(t,1);
  }
}
print(wordCountSet);
```

# Display the UNIQUE words

- Instead of tokens, we want the keys of the HashMap:
  - wordCountSet.keySet()

```
for (String t : wordCountSet.keySet()) {
    //Let's change the text size based on the frequency
    //textSize(<what goes here?>);
    if(random(100) > 40) { // more red than green
      fill(random(150,250),0, 0,190); // make red
    } else {
      fill(0,random(150,250), 0,190); // make green
    }
    text(t, random(0,width-50), random(20,height));
} // for
```

# Display the UNIQUE words

- Instead of tokens, we want the keys of the HashMap:
  - wordCountSet.keySet()

```
for (String t : wordCountSet.keySet()) {
    //Let's change the text size based on the frequency
    textSize(wordCountSet.get(t));
    if(random(100) > 40) { // more red than green
      fill(random(150,250),0, 0,190); // make red
    } else {
      fill(0,random(150,250), 0,190); // make green
    }
    text(t, random(0,width-50), random(20,height));
} // for
```

# Display the most frequent words

- Lazy way
  - check all frequencies of words in set and only display words above a threshold frequency.
  - First find the threshold (loop once)
  - Next use the threshold (loop second time)
- Systematic way
  - sort word set by frequency
  - only display top N words

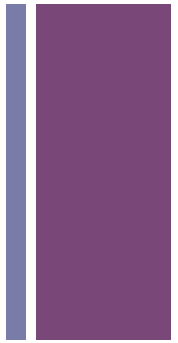
# Code from class (max frequency)

```
// get max
int max = 0;
for (Integer val : wordCountSet.values ()) {
  if (max < val) {
    max = val;
  }
}
```

# Filter and size text using max frequency and map()

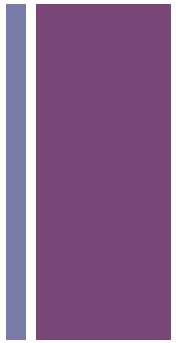
```
for (String t : wordCountSet.keySet ()) {
  int frequency = wordCountSet.get(t);
  if (frequency > max/20) { // filter based on max
    float tSize = map(frequency, 1, max, 12, 150);
    textSize(tSize);
    if (random(100) > 40) {
      fill(random(150, 250), 0, 0, 190);
    } else {
      fill(0, random(150, 250), 0, 190);
    }
    text(t, random(0, width-50), random(20, height));
  }
} // for
```

# Sorting



- Any process of arranging items in sequence
- Build-in `sort()`
  - Works on arrays of simple types, i.e. `int`, `float` and `String`
  - `float[] a = { 3.4, 3.6, 2, 0, 7.1 };`
  - `a = sort(a); // sort all elements in place`
  - `String[] s = { "deer", "elephant", "bear", "aardvark", "cat" };`
  - `s = sort(s, 3); // sort the first three elements`
- Convenient, but not very flexible

# Sorting (implement your own)



- Easy to code (but slow)
  - Selection Sort
  - Bubble Sort
  - Insertion Sort
- Animations
  - https://www.cs.usfca.edu/~galles/visualization/ComparisonSort.html
  - http://www.sorting-algorithms.com/

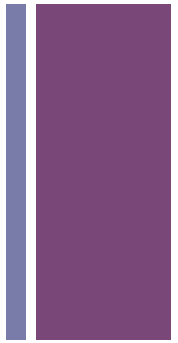
# Selection sort

- Basic idea:
  - step forward on each item of the array starting with the first item, if there is a smallest item in front of the item being stepped on, then swap the two items. Repeat until you've stepped on every item.
- Implementation:
  - nested loop
    - first loop marks the current item
      - inner loop finds the smallest item between the current item and the last item inclusively, then swaps the items
- Time Complexity?

# Bubble sort

- Basic idea:
  - start with the first item in the array compare adjacent items if they are not sorted, swap them, go to the next item and repeat until you get to the end.
  - repeat the above process until sorted
- Implementation:
  - nested loop
    - first loop checks if the array is sorted
      - inner compares and swaps
- Time Complexity?

# Insertion Sort



- Basic idea:
  - start with a sorted subarray, insert the next item from your unsorted list into the right position of the sorted list.
  - When you get to the end of the unsorted list, you are done
- Implementation:
  - nested loop
    - first loop gets next item to insert
      - inner compares, copies and makes space
      - inserts into space
- Time Complexity?