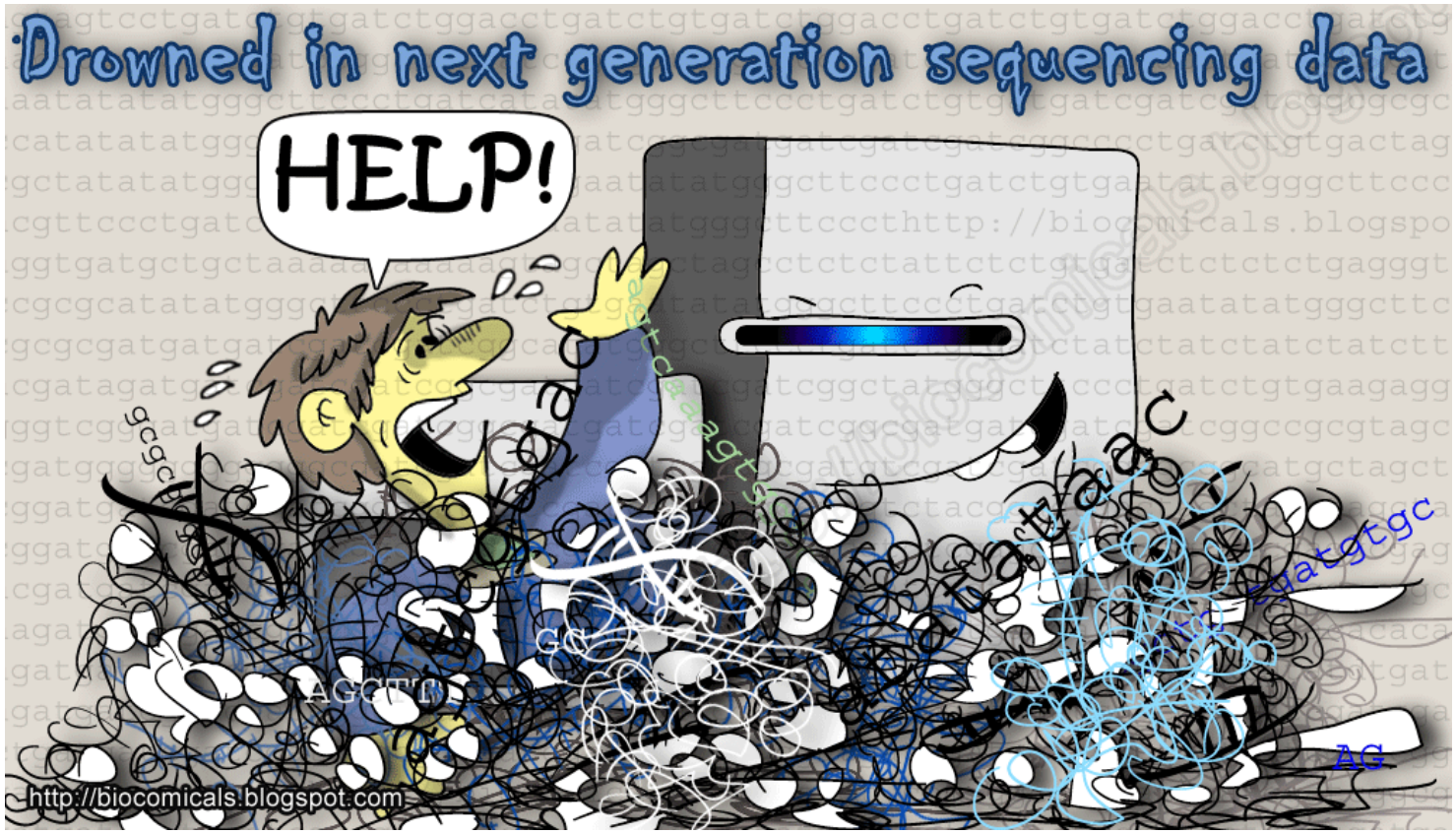
Assignment 3



Mayank Choudhary
Bio5488
5th February 2016

Adapted from www.biocomicals.com, Alper Uzun, PhD.

Assignment 3: Mapping cDNA Reads

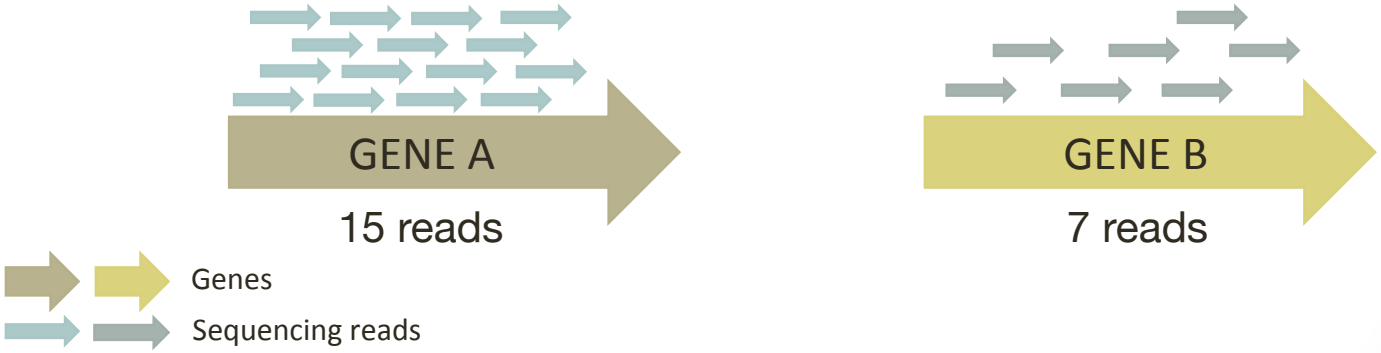
- Goal
  - Align RNA-seq reads to a set of genes to estimate their expression
- Input
  - FASTA file of genes of interest (scott_mouse_cDNAs.fa)

```
[mayank@genomic:~/assignment3]$ head scott_mouse_cDNAs.fa
>CCDS25507.1|Gfap|Mus musculus
ATGGAGCGGAGACGCATCACCTCTGCGCGCCGCTCCTATGCCTCCGAGACGGTGGTCAGGGGCCTCGG
GAGCGAGCGTGCAGAGATGATGGAGCTCAATGACCGCTTTGCTAGCTACATCGAGAAGGTCCGCTTCC
CGGCCAACAGTGCCCGGCTGGAGGTGGAGAGGGACAACTTTGCACAGGACCTCGGCACCCTGAGGCAG
GAGGAGATCCAGTTCTTAAGGAAGATCTATGAGGAGGAAGTTCGAGAACTCCGGGAGCAGCTGGCCCA
GTCTAAGTTTGCAGACCTCACAGACGCTGCGTCCCGCAACGCAGAGCTGCTCCGCCAAGCCAAGCACG
CCAGTTACCAGGAGGCACTTGCTCGGCTGGAGGAGGAGGGCCAAAGCCTCAAGGAGGAGATGGCCCGC
AACCTCCAGATCCGAGAAACCAGCCTGGACACCAAATCCGTGTCAGAAGGCCACCTCAAGAGGAACAT
>CCDS22757.1|Tubb3|Mus musculus
ATGAGGGAGATCGTGCACATCCAGGCCGGCCAGTGCGGCAACCAGATAGGGGCCAAGTTCTGGGAGGT
```

- Sequencing reads (mckinley_raw_reads.txt)

```
[mayank@genomic:~/assignment3]$ head mckinley_raw_reads.txt
AGACACACCCCATCTACACCCCGGA
AAACGAGCCCAGAGGCTCAAGTTCC
CAAACCCCACCACCAAAAAAAAAAA
AAAAAACCGGTTTGTTTTAAAAATA
AACGGGTCAAGGCTTGTCCCAGCTC
```

- Output
  - # of mapped reads for each gene ∝ gene expression



Short-read alignment

- Problem statement
  - Given a reference genome and a short read, find the locations in the reference genome that match the read

Close analogy: (approx.) string matching

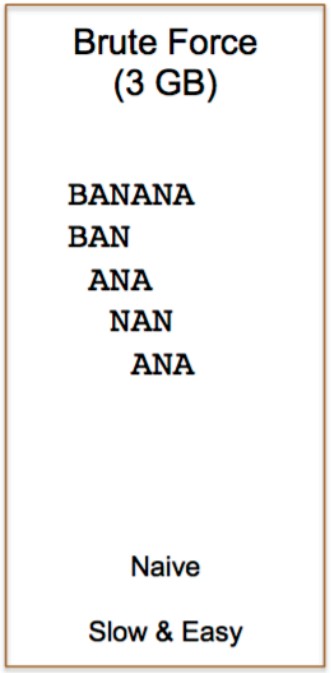
- The goal is to find a pattern (the short-read) in a large text or corpus (the genome), allowing for mismatches and indels.

- Naively, you can scan the text for the pattern but this is inefficient (think of trying to find an address in a phone book where the names were all mixed up).

- There are techniques to pre-process (or index) the text to make queries fast and also that can even compress the size of the text.

Short-read alignment

- Problem statement
  - Given a reference genome and a short read, find the locations in the reference genome that match the read
- There are several classes of algorithms to solve this:



Genomes and reads, however, are too large for direct approaches like dynamic programming.

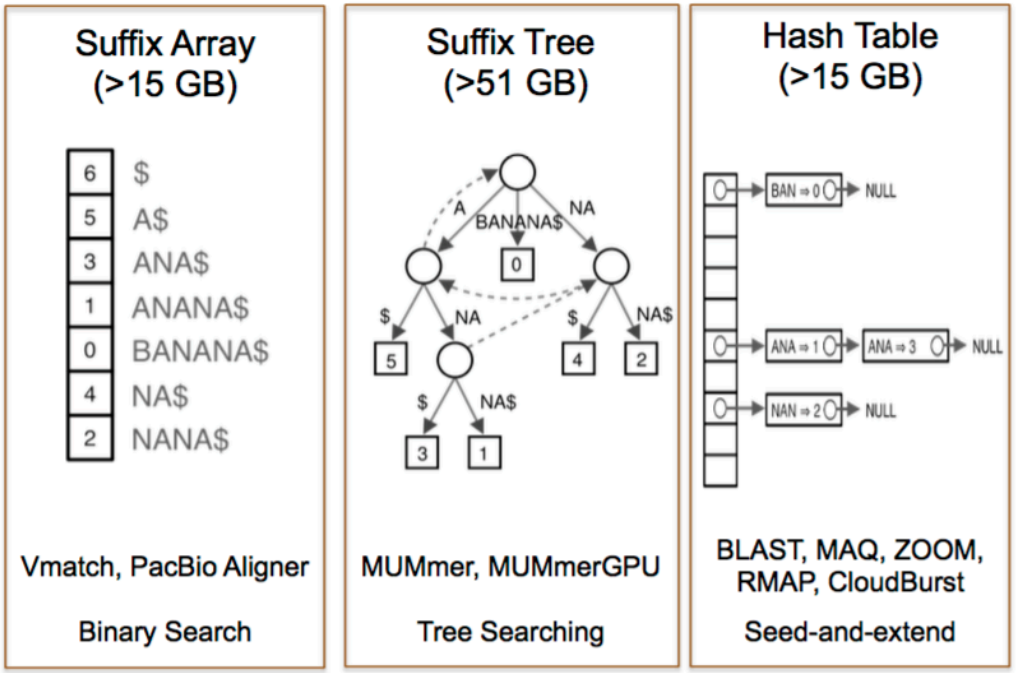
Indexing is required.

Illustrations adapted from Ben Langmead's Computational Genomics (Fall 2015)

Short-read alignment

- Problem statement
  - Given a reference genome and a short read, find the locations in the reference genome that match the read
- There are several classes of algorithms to solve this:

Choice of index is key to performance



Illustrations adapted from Ben Langmead's Computational Genomics (Fall 2015)

Short-read alignment with hash tables

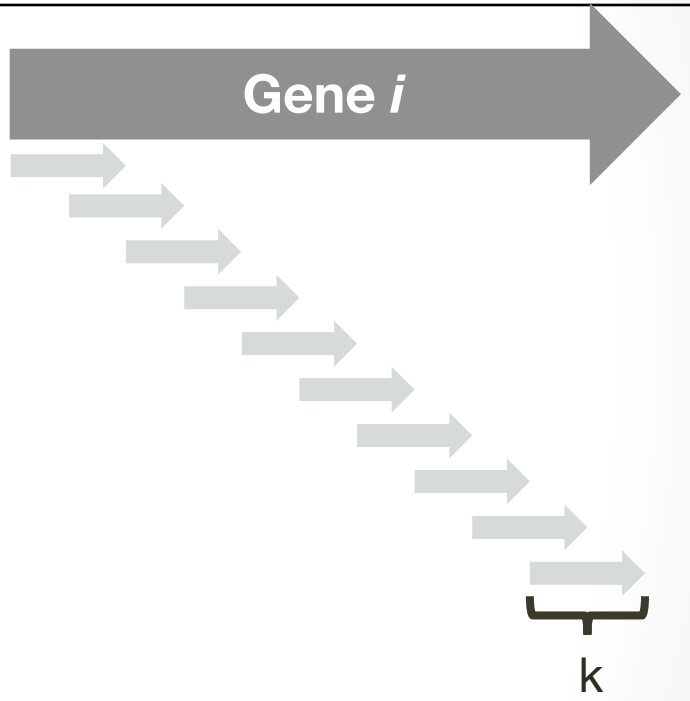
- Some popular alignment algorithms that use hash tables:
  - BLAST
  - MAQ
  - ELAND
- Advantages
  - Easy to implement
  - Fast
- Disadvantages
  - For you to think about ☺ (See Assignment3 questions)

Hash table alignment algorithm

Step 1

Create **a hash table**/dictionary of every substring of length k of the reference.

- Key = k-mer; value = gene name



K-mer sequence (keys)	Gene (values)
CCCCCC	Gene 1
AAAAAA	Gene 1
TTTTTT	Gene 2

Hash table alignment algorithm

Step 2

For each sequencing read of length k:

- Determine if the sequencing read exists in the hash table
- If the read exists in the hash table, then the read aligned
  - The name of the gene the read aligned to is stored as the value for that read key
- Else, the read did not align to any gene

Get the next sequencing read

AAAAA

Is the sequencing read in the hash table?

K-mer sequence (keys)	Gene (values)
CCCCC	Gene 1
AAAAA	Gene 1
TTTTT	Gene 2

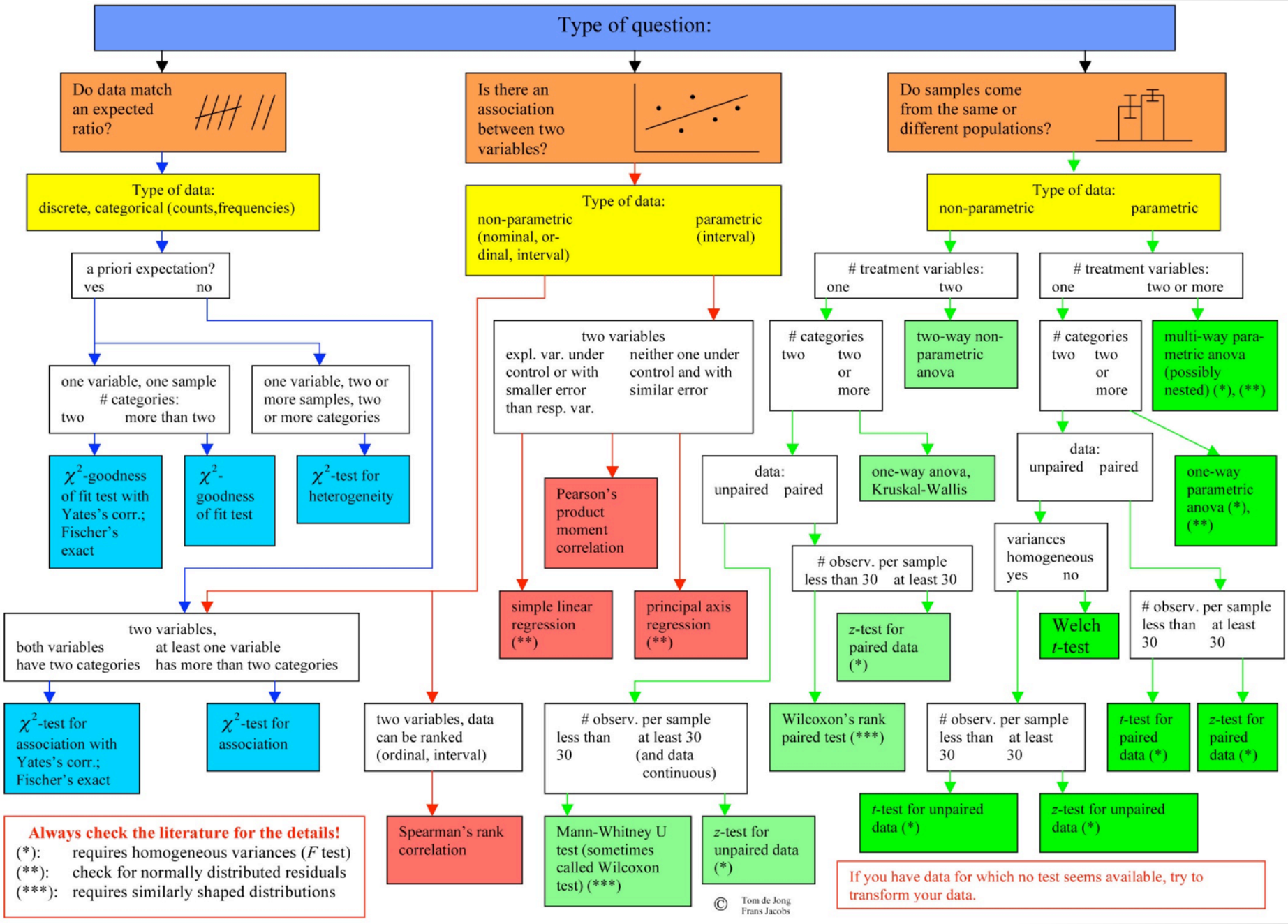
Yes, it aligned to gene 1

Your TODOs: Part 1

- Finish coding the starter script map_sequence_starter.py
  - Usage: `python3 map_sequence_starter.py <cDNA_file> <seq_reads_file>`
  - The sections of code that you will write are commented for you
  - Generate a 25-mer hash table of the gene cDNA sequences
    - **Key**: 25-mer sequence
    - **Value**: name of gene where 25-mer sequence was taken from
  - Create a dictionary of the gene names and cDNA sequences:
    - **Key**: gene name
    - **Value**: entire gene sequence
  - Calculate the reverse complement of a DNA sequence
    - Why this is needed: if a sequencing read does not map to the reference, its reverse complement may map

Your TODOs: Part 2

- Comment the code that's already written
  - Look up and define all functions used
  - https://docs.python.org/3/library/index.html
- Answer questions
  - Hypothesis testing



Adapted from http://fwncwww14.wks.gorlaeus.net/images/home/news/Flowchart2011.jpg

Assignment 3 requirements

- Starter scripts and input files located in `/home/assignments/assignment3/`
  - **IMPORTANT**: don't copy the input data files to work, just reference the full path, e.g.,

```
$ python3 map_sequence_starter.py /home/assignments/assignment3/scott_mouse_cDNAs.fa /home/assignments/assignment3/mckinley_raw_reads.txt
```

- Due Wednesday (2nd Feb '16) at 10:00 AM
- Your submission folder should contain:
  - Modified map_sequence_starter.py
  - Modified README.txt

Assignment 3 suggestions:
a.k.a. how to maintain your sanity

- Read the documentation
  - Get a gist of what Python is capable of

Topic	Python.org link
Functions	https://docs.python.org/3/library/functions.html
Integers & floating points numbers	https://docs.python.org/3/library/stdtypes.html#numeric-types-int-float-complex
Strings	https://docs.python.org/3/library/stdtypes.html#str
Lists	https://docs.python.org/3/library/stdtypes.html#list
Dictionaries	https://docs.python.org/3/library/stdtypes.html#mapping-types-dict

- Google, google, google
- Work in groups
- Use the discussion boards on BB
- Come to office hours (M after class)
- Come to the tutoring sessions (T 3:15)

Python dictionaries: what are they?

- A common data structure
  - Collection of key-value pairs
    - The collection is unordered
    - A key-value pair is called an item
  - Keys
    - Can be a strings or numbers
    - Must be unique. A dictionary cannot have duplicate keys
  - Values
    - Can be anything: strings, numbers, dictionaries
    - Do not have to be unique
- Also called hash tables & associative arrays
- Why are dictionaries useful?
  - Allow you to map/associate something with something else

Python dictionaries: what can I do with them?

- Create empty dictionaries
- Add key-value pairs
- Remove key-value pairs
- Update a key’s value
- Check if a key exists in a dictionary
- Iterate through all key-value pairs

Python dictionaries: where can I learn more?

- Python.org tutorial: https://docs.python.org/3.4/tutorial/datastructures.html#dictionaries

- Python.org documentation: https://docs.python.org/3.4/library/stdtypes.html#mapping-types-dict

Python functions: what are they?

- A block of reusable code used to perform a specific task
  - Take in arguments (optional)
  - Do something
  - Return something (optional)
- Similar to mathematical functions

  $f(x) = x^2$

  $f(2) = 4$
- Why are functions useful?
  - Allow you to reuse the same code
- Two types
  - Built-in
    - Examples
      - `print`: print something to the terminal
      - `float`: convert something to a floating point number
  - User-defined
    - You create your own functions

Python functions: how can I call a function?

Call a function that takes no arguments

`<function_name>()`	`print_greeting()`

Call a function that takes argument(s)

`<function_name>(<arg>)`	`len("Fred")`

Python functions: how can I define my own function?

IMPORTANT: Functions must be defined after the import statements but before the main code!

Define a function without arguments

```
def <function_name>():
    # Do something
```

```
def print_greeting():
    print("Howdy!")
```

Define a function with arguments

```
def <function_name>(<arg>):
    # Do something
```

```
def print_greeting(name):
    print("Howdy " + name + "!")
```

Define a function that returns a value

```
def <function_name>():
    # Do something
    return(<value_to_return>)
```

```
def print_greeting():
    greeting = "Howdy!"
    return(greeting)
```

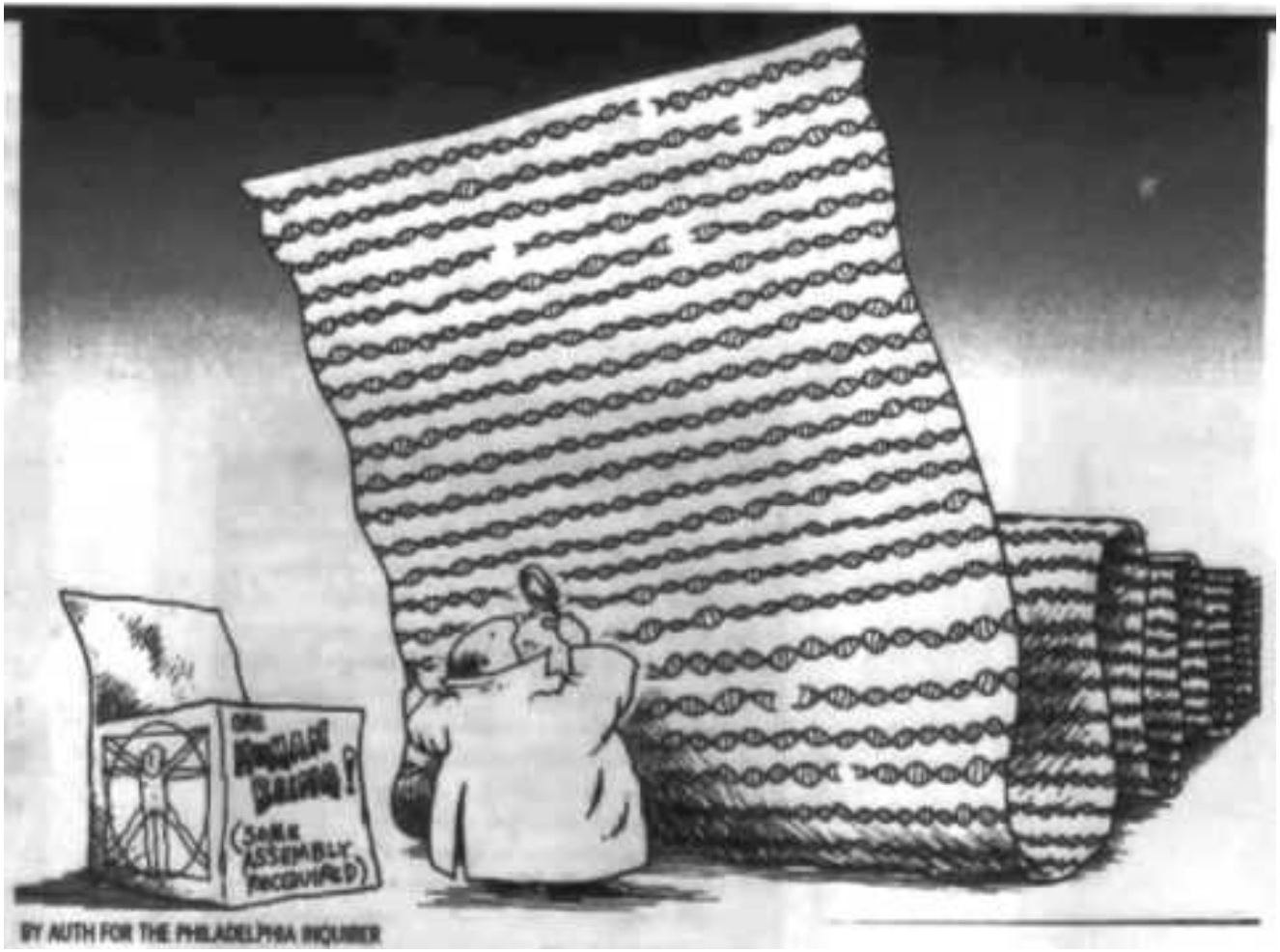
Python functions: where can I learn more?

- Python.org tutorial
  - User-defined functions: https://docs.python.org/3/tutorial/controlflow.html#defining-functions

- Python.org documentation
  - Built-in functions: https://docs.python.org/3/library/functions.html

Questions?

Good luck!



Adapted from Allen Van Deynze's talk at the Tomato Disease Workshop 2010