Access the slides and files here:

https://github.com/j-berg/bioinformatics_bootcamp

#4.1

Importing a datatable

Making a gene dictionary

Mapping gene IDs and gene names

Reusable script

Modularity

- Plan how to break up complex tasks using pseudocode and make each task a function

Goals:

- Use an organism’s GTF file as input
- Parse out matching gene IDs and names
- Import user datatable
- Map gene dictionary to row names
- Output modified table

Getting a user's inputs

```python
import sys
import pandas as pd

# Set global variables
type_col = 2
metadata_col = 8

# Parse user arguments
def parse_args():
    if len(sys.argv) != 3:
        raise Exception("Too few or too many arguments provided.")

    print("Running script:", sys.argv[0])
    print("With options:")
    print("GTF file:", sys.argv[1])
    print("Datatable file:", sys.argv[2])
    gtf_url = sys.argv[1]
    df_url = sys.argv[2]

    return gtf_url, df_url
```

Importing data

```
22  # Use an organism's GTF file as input
23  def read_gtf(gtf_url):
24      gtf = pd.read_csv(
25          str(gtf_url),
26          sep = '\t',
27          header = None,
28          comment = '#',
29          low_memory = False
30      )
31      return gtf
32
33  # Import user datatable
34  def read_df(df_url):
35      data = pd.read_csv(
36          str(df_url),
37          sep = '\t',
38          index_col = 0,
39          low_memory = False
40      )
41      return data
42
```

Making a gene dictionary

```
43  # Parse out matching gene IDs and names
44  def make_dictionary(gtf, type_id="gene"):
45
46      gtf_c = gtf.copy()
47      gtf_c = gtf_c.loc[gtf_c[type_col] == type_id]
48
49      gtf_c['intermediate_id'] = gtf_c[8].str.split('gene_id \"').str[1]
50      gtf_c['id'] = gtf_c['intermediate_id'].str.split('\"; ').str[0]
51
52      gtf_c['intermediate_name'] = gtf_c[8].str.split('gene_name \"').str[1]
53      gtf_c['name'] = gtf_c['intermediate_name'].str.split('\"; ').str[0]
54
55      gene_dict = pd.Series(gtf_c['name'].values,index=gtf_c['id']).to_dict()
56
57      return gene_dict
58
```

Mapping gene IDs and gene names

```
59  # Map gene dictionary to row names
60  def map_index(data, gene_dict):
61
62      data_c = data.copy()
63      data_c['new_name'] = data_c.index.to_series().map(gene_dict)
64      data_c['new_name'] = data_c['new_name'].fillna(data_c.index.to_series())
65      data_c = data_c.set_index('new_name')
66      data_c.index.name = None
67
68      return data_c
69
```

Output Data

```
70  # Output modified table
71  def output_modified(data, data_url):
72
73      output_name = data_url.split('.')[0]
74      suffix = data_url.split('.')[1]
75      save_name = str(output_name) + '_renamed.' + str(suffix)
76      data.to_csv(
77          str(save_name),
78          sep='\t'
79      )
80
```

Tying it all together

```
81  # Main
82  #
83  gtf_url, df_url = parse_args()
84  gtf = read_gtf(gtf_url)
85  data = read_df(df_url)
86  gene_dict = make_dictionary(gtf)
87  data_mapped = map_index(data, gene_dict)
88  output_modified(data_mapped, df_url)
89
```

Reusable scripts

```
(base) jordan-berg:class_4_1 jordan$ python gene_dictionary.py Saccharomyces_cerevisiae.R64-1-1.100.gtf SCE_data_table.tsv
Running script: gene_dictionary.py
With options:
GTF file: Saccharomyces_cerevisiae.R64-1-1.100.gtf
Datatable file: SCE_data_table.tsv
(base) jordan-berg:class_4_1 jordan$
```

Homework

- Take your RNA-seq data table that was mapped to gene IDs from before and remap to gene names

- if you haven’t gotten that far, an example data table is available in the folder for this lesson on the GitHub repository