Who is this gene and what does it do? A toolkit for munging transcriptomics data in python

Charles K. Fisher,[*] Aaron M. Smith, and Jonathan R. Walsh

Unlearn.AI, Inc., San Francisco, CA 94108

(Dated: April 5, 2018)

Abstract

Transcriptional regulation is extremely complicated. Unfortunately, so is working with transcriptional data. Genes can be referred to using a multitude of different identifiers and are assigned to an ever increasing number of categories. Gene expression data may be available in a variety of units (e.g, counts, RPKMs, TPMs). Batch effects dominate signal, but metadata may not be available. Most of the tools are written in R. Here, we introduce a library, **`genemunge`**, that makes it easier to work with transcriptional data in python. This includes translating between various types of gene names, accessing Gene Ontology (GO) information, obtaining expression levels of genes in healthy tissue, correcting for batch effects, and using prior knowledge to select sets of genes for further analysis. Code for **`genemunge`** is freely available on Github.

[*] drckf@unlearn.ai; authors listed alphabetically.

I. OVERVIEW

> munge: verb
>
> 1. to manipulate (raw data), especially to convert (data) from one format to another.
>
> *www.dictionary.com/browse/munge*

Like any area that uses big data, transcriptomics data requires extensive munging – rote but critical tasks such as cleaning data, selecting relevant data, structuring metadata, and making labels interpretable. These tasks often need to be repeated on a given project as the data and aims evolve, and tend to be similar between different analyses. To face these challenges, a library of data munging tools can be extraordinarily useful. Such a library can provide reliable and tested tools to cleanly separate munging tasks from analysis, making it easier to start new projects and data processing pipelines less fragile.

This note introduces `genemunge`, a library of tools for working with human transcriptomics data. `genemunge` is written in python and is available as a package through `PyPI`. This initial version, `v0.0`, contains tools for tasks such as:

- Translating between conventions for gene symbols [1].
- Accessing Gene Ontology (GO) metadata [2–4].
- Using prior knowledge of biological and molecular processes to select gene sets [5, 6].
- Retrieving statistics on gene expression in healthy tissue [7–10].
- Converting expression data to TPM from counts or RPKM [11].
- Correcting for unobserved, and uninteresting, factors of variation (i.e., batch effects). [12, 13].

The goal of `genemunge`, and its current use case for the authors, is to serve as a resource for gene information that can return useful data structures and be integrated into processing pipelines. The next section gives a few example use cases of the library.

II. EXAMPLE USE CASES OF GENEMUNGE

We consider a simple analysis where **genemunge** is useful. Suppose we want to find genes associated with the immune system, select those with larger expression in the small intestine than the stomach, and then retrieve basic information about those genes.

The following code snippets use the API from **genemunge v0.0**. We begin by importing libraries required for the example.

```
import json
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

import genemunge
```

A. Searching the gene ontology for relevant genes

The Gene Ontology (GO) contains basic descriptors for each ontology entry [2–4]. Since we want genes related to the immune system, we will do a keyword search for *immune* and retrieve GO identifiers with this keyword. We can then obtain the associated genes.

```
# set up an object to search the gene ontology
searcher = genemunge.search.Searcher()

# get all of the GO identifiers associated with the word ’immune’
# set exact = False to walk through the ontology and grab all child terms
immune_identifiers = searcher.keyword_search([’immune’], exact=False)

# get all of the genes assigned to the immune_identifiers
immune_genes = searcher.get_genes(immune_identifiers)
```

We will then use prior knowledge to remove housekeeping genes. A list of housekeeping genes curated by [6] is stored in **genemunge**.

```
# get a list of housekeeping genes
housekeeping = searcher.get_housekeeping_genes()

# keep all of the immune related genes that are not housekeeping genes
variable_immune_genes = list(set(immune_genes) - set(housekeeping))
```

B. Obtaining statistics about gene expression

We can use **genemunge** to access summary statistics from the GTEx project (through recount) about expression levels in healthy tissue [7–10]. The median expression value can be used to find genes that are more expressed in the small intestine than the stomach.

```
# set up an object to describe genes
describer = genemunge.describe.Describer('symbol')

# find the absolute and relative expression levels of each gene of interest
expression_data = pd.DataFrame(index=variable_immune_genes,
                               columns=['expression', 'log ratio'])

# get the expression levels in healthy tissue (in TPM units)
stats = describer.tissue_stats['median'].reindex(variable_immune_genes)
expression_data['expression'] = stats['Small Intestine']

# control the log with a small pseudocount
pseudocount = 1.0
expression_data['log ratio'] = np.log10(
  (pseudocount + stats['Small Intestine']) / (pseudocount + stats['Stomach']))
```

A scatter plot of the relative expression in the small intestine to the stomach against the absolute expression in the small intestine is shown in Figure 1.

```
# plot the gene expression fraction
fig, ax = plt.subplots(figsize=(12, 8))
```

```
plt.xticks(fontsize=16)
plt.yticks(fontsize=16)

ax.scatter(expression_data['expression'], expression_data['log ratio'])
ax.set_xlabel('Small Intestine expression [TPM]', fontsize=20)
ax.set_ylabel('log10 ratio (Small Intestine / Stomach)', fontsize=20)
ax.set_xscale('log')
ax.set_xlim([0.001, 1e5])
ax.set_ylim([-3, 4])
plt.savefig('small_intestine_example.png', bbox_inches='tight', dpi=300)
plt.show()
```

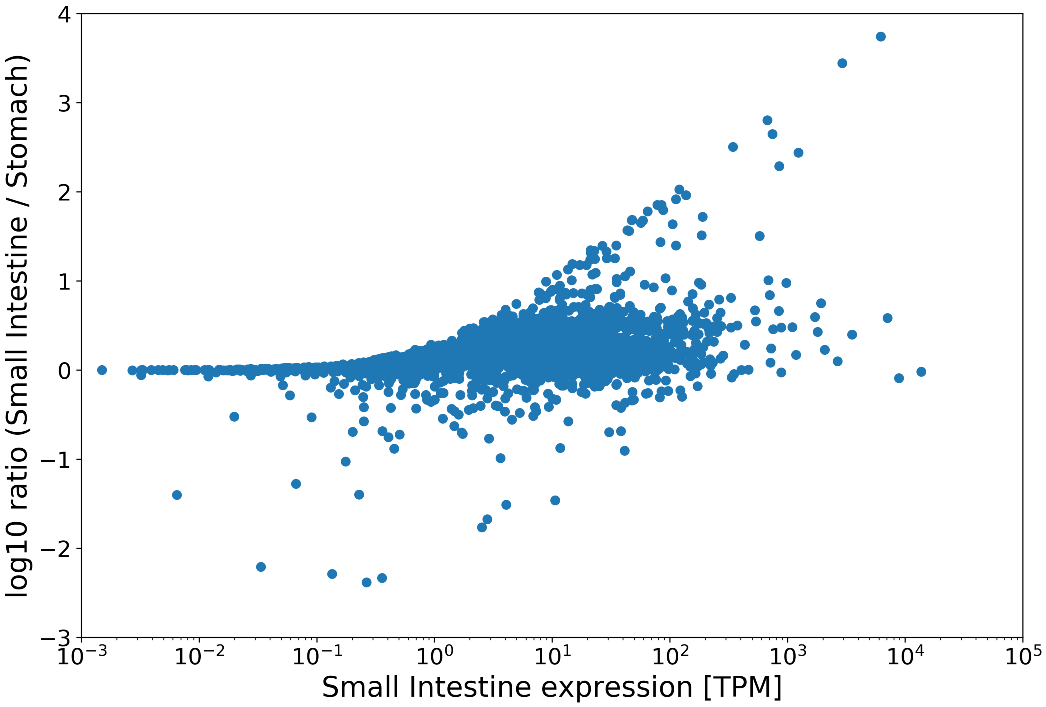

FIG. 1: Scatter plot showing the relative expression between the small intestine and the stomach versus the absolute expression in the small intestine. Genes of interest are in the upper arm.

C. Converting between gene identifier types

In genemunge, the base representation of genes is in terms of their Ensembl ID (without a version number). We will want to see the gene symbol in the results, so we convert the gene symbols.

```
# set up an object to convert from ensembl to symbol
ensembl_to_symbol = genemunge.convert.IDConverter('ensembl_gene_id', 'symbol')

# convert the immune identifiers to gene symbols
variable_immune_symbols = ensembl_to_symbol.convert_list(variable_immune_genes)

# reset the index of the dataframe
expression_data.index = variable_immune_symbols
```

We can then select genes with high relative expression. Of course, one should be careful with this type of thing and do a differential expression analysis, but we'll just wing it.

```
# select genes with high relative expression
target_genes = expression_data[expression_data['log ratio'] > 1]
target_genes = target_genes.sort_values(by=['expression'], ascending=False)
```

D. Displaying information about a gene

Finally, we can look at metadata and gene expression data from GTEx on one of these genes (in this case, the most highly expressed gene).

```
# get some basic information about one of the immune genes
print(json.dumps(describer.get_gene_info(target_genes.index[0]), indent=2))

# make a plot of the expression of one of the immune genes across tissues
describer.plot_tissue_expression(target_genes.index[0], sortby='median',
                                 filename='gene_expr_example.png')
```

The genemunge output of the gene info is

```
"ensembl":  "ENSG00000164816",
  "symbol": "DEFA5",
  "name": "defensin alpha 5",
  "ontology": {
    "GO:0005576": "extracellular region",
    "GO:0050830": "defense response to Gram-positive bacterium",
    "GO:0050832": "defense response to fungus",
    "GO:0031640": "killing of cells of other organism",
    "GO:0045087": "innate immune response",
    "GO:0019730": "antimicrobial humoral response",
    "GO:0050829": "defense response to Gram-negative bacterium",
    "GO:0061844": "antimicrobial humoral immune response mediated by antimicrobial peptide",
    "GO:1905710": "positive regulation of membrane permeability",
    "GO:0051673": "membrane disruption in other organism",
    "GO:0005796": "Golgi lumen",
    "GO:0005615": "extracellular space",
    "GO:0002227": "innate immune response in mucosa",
    "GO:0034774": "secretory granule lumen",
    "GO:0042803": "protein homodimerization activity",
    "GO:0019731": "antibacterial humoral response",
    "GO:0030133": "transport vesicle",
    "GO:0071222": "cellular response to lipopolysaccharide"
  }
```

and the gene expression profile from GTEx is shown in Figure 2. As expected, `DEFA5` is highly expressed in the small intestine and, in this case, nowhere else.

E. Other stuff we aren't covering

This example does not use all of the features of `genemunge`. For example, one could also use `genemunge` to identify transcription factors, or look up the lengths of genes and use them to convert data from counts to TPM units. You can even perform unsupervised correction of batch effects using the Remove Unwanted Variation (RUV) algorithm [12, 13]. You can read more about all of the amazing things `genemunge` can do in the documentation.

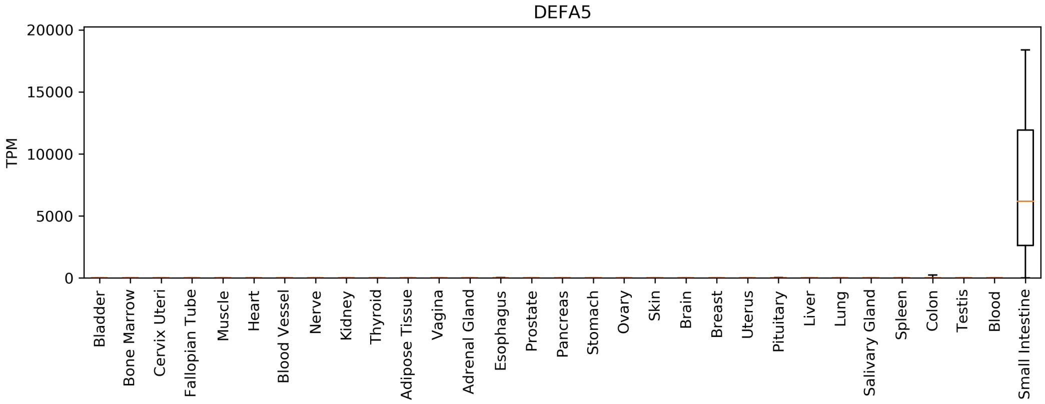

FIG. 2: Gene expression profile in healthy tissue from GTEx for an example gene.

III. SUMMARY

`genemunge` was built to make working with transcriptomics data easier. It provides a simple way to select genes of interest in an analysis and return useful metadata about them. We find that it is a useful component of a larger analysis and data processing pipeline. Our intent on open sourcing the package is to engage with the computational biology community and build it into a broadly useful tool. We welcome feedback, feature requests, and contributions on `genemunge` through `GitHub`.

[1] E. A. Bruford, M. J. Lush, M. W. Wright, T. P. Sneddon, S. Povey, and E. Birney, Nucleic acids research **36**, D445 (2007).

[2] M. Ashburner, C. A. Ball, J. A. Blake, D. Botstein, H. Butler, J. M. Cherry, A. P. Davis, K. Dolinski, S. S. Dwight, J. T. Eppig, *et al.*, Nature genetics **25**, 25 (2000).

[3] G. O. Consortium, Nucleic acids research **32**, D258 (2004).

[4] S. Carbon, A. Ireland, C. J. Mungall, S. Shu, B. Marshall, S. Lewis, A. Hub, and W. P. W. Group, Bioinformatics **25**, 288 (2008).

[5] K. Chawla, S. Tripathi, L. Thommesen, A. Lægreid, and M. Kuiper, Bioinformatics **29**, 2519 (2013).

[6] E. Eisenberg and E. Y. Levanon, Trends in Genetics **29**, 569 (2013).

[7] L. J. Carithers, K. Ardlie, M. Barcus, P. A. Branton, A. Britton, S. A. Buia, C. C. Compton, D. S. DeLuca, J. Peter-Demchok, E. T. Gelfand, *et al.*, Biopreservation and biobanking **13**,

311 (2015).

[8] L. J. Carithers and H. M. Moore, "The genotype-tissue expression (gtex) project," (2015).

[9] A. C. Frazee, B. Langmead, and J. T. Leek, BMC bioinformatics **12**, 449 (2011).

[10] L. Collado-Torres, A. Nellore, K. Kammers, S. E. Ellis, M. A. Taub, K. D. Hansen, A. E. Jaffe, B. Langmead, and J. T. Leek, Nature biotechnology **35**, 319 (2017).

[11] G. P. Wagner, K. Kin, and V. J. Lynch, Theory in biosciences **131**, 281 (2012).

[12] J. A. Gagnon-Bartsch and T. P. Speed, Biostatistics **13**, 539 (2012).

[13] L. Jacob, J. A. Gagnon-Bartsch, and T. P. Speed, Biostatistics **17**, 16 (2015).