Case Studies in Bioinformatics

Cyril Matthey-Doret, December 2016

Background

Ontogeny is the development of an organism from egg fertilization to its mature state. It is driven by expression of different genes at each stage of development, associated with phenotypic changes. There is a lot of discussion on how these sets of genes differ between species. The earliest studies of embryogenesis, in the 19th century, suggested that the most similar state of development between related species happens at the earliest stages of embryogenesis. This lead to what is called the funnel model, proposing that related species are very similar at the onset of embryogenesis and diverge as they develop. However, more recent work suggested that this convergence, also known as the phylotypic stage, happens later, earliest stages being more different between species. This gave rise to the hourglass model. Both the funnel and hourglass model were based on morphological observations, but it is now possible to test these models by combining gene expression data at different stages of embryogenesis with phylogenetic information for the concerned genes to better study the similarity between species. Here, we investigate a study (Domazet-Lošo et Tautz, 2010) relying on an approach termed phylostratigraphy, which consists in tracing the origin of specific genes through similarity searches between species. This allows to build a phylogenetic tree where each gene has a rank that estimates their age. These ranks can be used to get a metric of age for the whole transcriptome at each developmental stage, called Transcriptome Age Index (TAI). Results obtained in this study support the hourglass model, focusing on expression datasets from the zebrafish *Danio rerio*, Drosophilia, the mosquito *Anopheles* and the nematode *Caenorhabditis elegans*. In this report, we use forensic bioinformatics to repeat the analysis from Domazet-Lošo et Tautz, 2010 in zebrafish to have a critical view of each step and a better understanding of their results. Finally, we propose improvements for the analysis, explaining how it would affect the results.

Methods

We reproduced the analysis from the paper using python 2.7 with the modules panda, numpy, matplotlib and GEOparse. We used expression data from *Danio rerio* (zebrafish) to reproduce the results shown in figure 1a of Domazet-Lošo et Tautz, 2010. Expression data for zebrafish was made available by the authors and phylostrata dataset were obtained from the University of Lausanne Computational Biology Group website. The first step was to load the expression data from microarray experiments and extract meaningful informations from the metadata for each sample (developmental stage, age and sex). The expression data was then formatted and annotated with microarray spot ID using the GEO platform (GPL) annotation file. Throughout the whole report, "expression" refers to the microarray signal intensity value and therefore has no unit.

```
#!pip install GEOparse

import matplotlib.pylab as plt
import numpy as np
import pandas as pd
import GEOparse

try:
    gse = GEOparse.get_GEO(filepath="./GSE24616.soft.gz", silent=True)
except IOError:
    gse = GEOparse.get_GEO("GSE24616",destdir="./", silent=True)
# Using GEOparse inbuilt tools to parse the Gene series expression data for zebrafish.
# If the file is not found, it is downloaded first, otherwise it is parsed directly
```

```
char = {"stage": [], "time": [], "sex": [], "sample_name": []}
# Initializing dictionary object to store metadata

for gsm_name, gsm in sorted(gse.gsms.iteritems()):
    char["stage"].append(gsm.metadata['characteristics_ch1'][1].split(": ")[1])
    char["time"].append(gsm.metadata['characteristics_ch1'][2].split(": ")[1])
    char["sex"].append(gsm.metadata['characteristics_ch1'][3].split(": ")[1])
    char["sample_name"].append(gsm.name)
# Formatting parsed metadata in a structured dictionary object

GPL = gse.gpls.values()[0]
# Getting spots IDs from platform annotation file
pivoted_samples = gse.pivot_samples('VALUE')
# Extract expression values with samples as columns
pivoted_samples.set_index(GPL.table.SPOT_ID, inplace=True)
# Setting indexes as microarray spot ID
```

Next, we imported the phylostrata and merged it with expression data by spot ID. Since the phylostrata data file contains both spots ID and associated genes ID, there is no need for an additional conversion step. As certain genes were represented multiple times, we used the mean expression for these duplicate genes in each sample. We then generated time stamps corresponding to the different ages of samples. There are 61 time stamps represented by integers assigned in chronological order, matching all of the unique age values from samples in the dataset. For later use, we computed the mean expression of each gene across all samples at each time stamp.

```
strata = pd.read_csv("./phylostrata.txt", sep="\t", header=None)
strata.columns = ["GeneID", "ProbeID", "age"]
strata.set_index("ProbeID", inplace=True)
# Loading and formatting phylostrata with matching genes ID, probe ID and gene age

matched_data = pivoted_samples.join(strata, how="inner").groupby(level=0).last()
# Matching expression with phylostrata by probe ID, adding columns for GENE ID and age.
# Only probes present in both tables are kept.

unique_data = matched_data.groupby("GeneID").mean()
# Taking the mean expression values for all genes.

char_pd = pd.DataFrame(char, index=char["sample_name"])
# New dataframe object containing meta-data with sample names as row indexes.
mixed = char_pd[char_pd.sex == "mixed"].sample_name.tolist()
mixed += char_pd[char_pd.sex == "female"].sample_name.tolist()
# Excluding males from the analysis to simplify the analysis.

char_pd["timing_number"] = 0
# Initiating new column to store developmental time stamps
time_stamps = char_pd.time.unique()
for i in xrange(len(time_stamps)):
    char_pd.loc[char_pd.time == time_stamps[i], "timing_number"] = i + 1
# Assigning developmental time stamps

experiment_index = char_pd[char_pd.index.isin(mixed)].reset_index().groupby(
    "timing_number")["index"].apply(lambda x: np.array(x))
# Dataframe with samples grouped by timestamps
```

```
set_mean = {}
stages = []
for d, col_list in experiment_index.iteritems():
    set_mean[d] = unique_data[col_list].mean(axis=1)
    # Storing mean expression of all samples at with a given time stamp
    stages.append(char_pd[char_pd.index.isin(col_list)].stage[0])
    # Storing developmental stage matching time stamp

mean_data = pd.DataFrame(set_mean)
# Builds dataframe containing mean expression of all genes at each time stamp
```

Results

From this data structure containing the mean expression of every gene at a given time stamp, we calculated the transcriptome age index (TAI) at each time stamp "s", as described in the paper methods:

$$TAI_s = \frac{\sum_{i=1}^{n} ps_i e_i}{\sum_{i=1}^{n} e_i}$$

where ps_i is the phylostratum of gene i and e_i is its expression.

```
age_indices = unique_data.age
# Extracting phylostrata (age rank).
expression_data = mean_data.values
# Extracting mean expression values for all time stamps.
product = np.dot(expression_data.T, age_indices)
# Computing the weighted sum of phylostrata for each time stamp using dot-product.
mean_expression = expression_data.T.sum(1)
# Summing all expression values for each time stamp.
TAI = np.divide(product, mean_expression)
# For each time stamp, dividing weighted sums of phylostrata
# by respective sums of expressions to obtain TAI.

plt.plot(TAI)
plt.xlabel("Time stamp")
plt.ylabel("TAI")
my_colorlist = ['#ff0000', '#ff6a00', '#ffb700', '#91ff00', '#00ff59',
                '#00ffbf', '#00bbff', '#aa00ff', '#e205ff','#ff05c9',
                '#940047']
def order_uniq(seq):
    # Function allowing to get each unique list element and preserve the order
    seen = set()
    seen_add = seen.add
    return [x for x in seq if not (x in seen or seen_add(x))]

col_count = 0
for t in order_uniq(stages):
    plt.axvline(x=stages.index(t), color=my_colorlist[col_count], label=t)
    # Adding vertical line representing start of developmental stages
    col_count += 1
```

```
legend = plt.legend(loc=(0.29,0.5), shadow=False, fontsize='x-small')
legend.get_frame().set_facecolor('#dddddd')
```

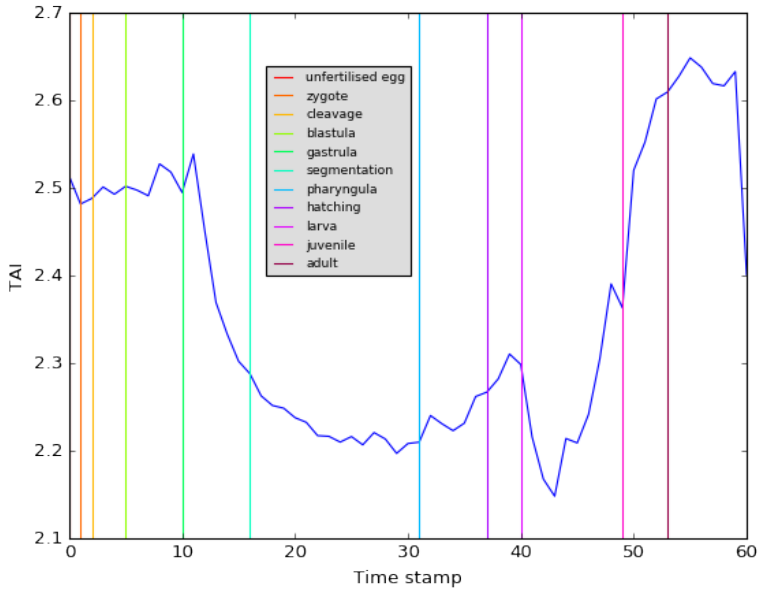

Figure 1: TAI as calculated in the original paper. Time stamps represent each unique value for the age of samples in chronological order. These values are ranks and the age does not increase with them in a linear manner. The vertical lines represent the start of each developmental stage defined as the first time stamp at which the stage was observed.

When plotting the TAIs calculated as described in the paper (Figure 1), we obtained rather similar results. On the figure presented in this report, the vertical lines represent the earliest time stamp at which a developmental stage was observed. This figure depicts what one would expect from the hourglass model, as the lowest TAIs (i.e. the transcriptome with the oldest genes, or phylotypic stage) are located in the middle, from the segmentation to the larval state.

However, there is a bias in the way gene expression values affect the TAI: Distributions of gene expressions are usually made of many lowly expressed genes, and a few outliers that are highly expressed. Since the TAI directly depends on the product of gene expression by phylostrata, those few outliers will dictate the TAIs.

As expected, when plotting the distribution of the whole gene expression data in the zebrafish dataset (Figure 2), most genes have low expression values, and there are outliers with much higher values (the maximum expression observed was 388'818.64).

A common way to solve the outlier problem is to log-transform the expression values (Figure 3). This reduces the distance with outliers and would be interesting here, as this would reduce the weight of strongly expressed genes on the TAI.

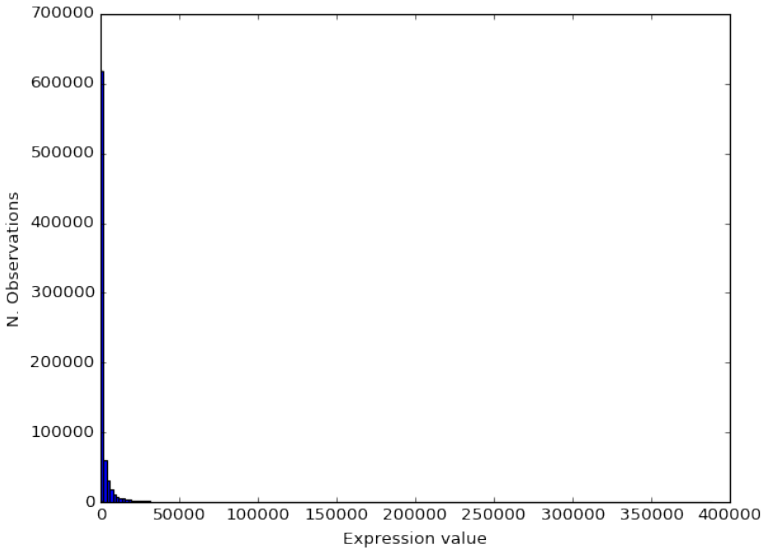

Figure 2: Distribution of gene expression across all samples.

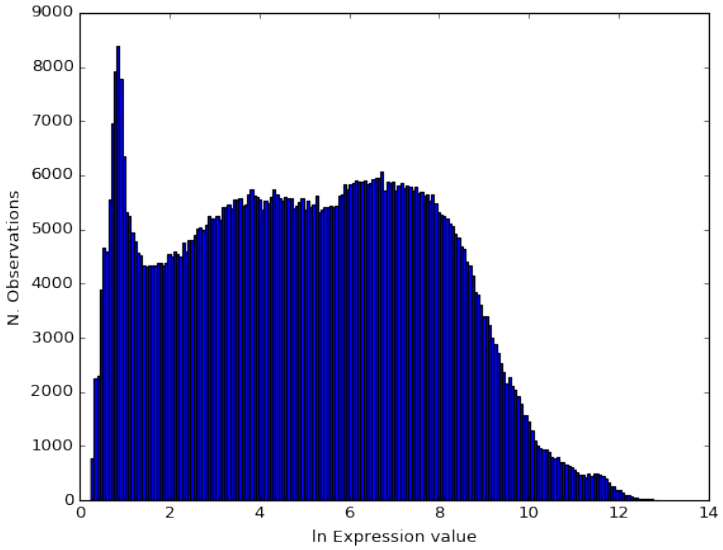

Figure 3: Distribution of log-transformed gene expression across all samples.

Finally, we recalculated the TAI using this log-transformed expression data:

```
age_indices = unique_data.age
# Extracting phylostrata (age rank).
expression_data = np.log(mean_data.values)
# Extracting ln of mean expression values for all time stamps.
product = np.dot(expression_data.T, age_indices)
# Computing the weighted sum of phylostrata for each time stamp using dot-product.
mean_expression = expression_data.T.sum(1)
# Summing all log-transformed expression values for each time stamp.
TAI = np.divide(product, mean_expression)
# For each time stamp, dividing weighted sums of phylostrata by
# respective sums of expressions to obtain TAI.

plt.plot(TAI)
plt.xlabel("Time stamp")
plt.ylabel("TAI")

col_count = 0
for t in order_uniq(stages):
    plt.axvline(x=stages.index(t), color=my_colorlist[col_count], label=t)
    # Adding vertical line representing start of developmental stages
    col_count += 1

legend = plt.legend(loc=(0.29,0.5), shadow=False, fontsize='x-small')
legend.get_frame().set_facecolor('#dddddd')
```

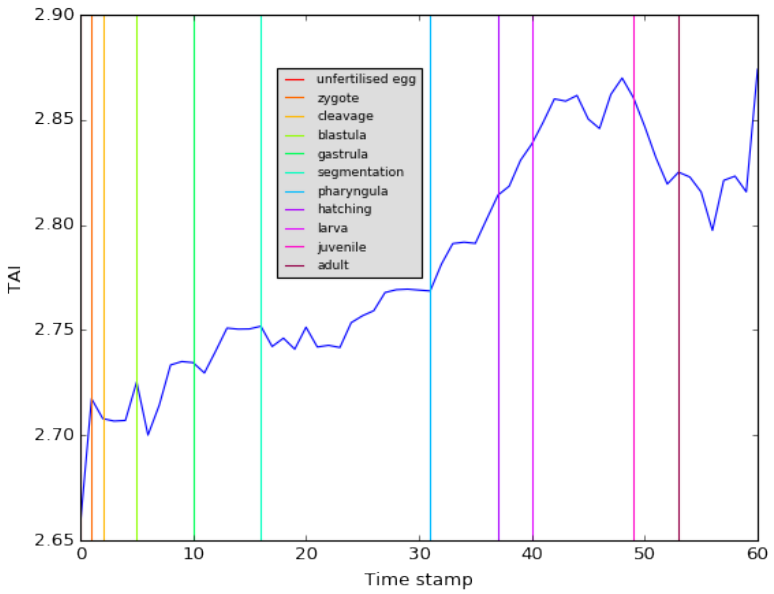

Figure 4: TAI calculated from log-transformed expression. Time stamps represent each unique value for the age of samples in chronological order. These values are ranks and the age does not increase with them in a linear manner. The vertical lines represent the start of each developmental stage defined as the first time stamp at which the stage was observed.

Log-transforming the expression dramatically changes the results (Figure 4), showing these outliers genes had a strong impact on the TAI value. The lowest TAI are now at the very begining of development and the transcriptome becomes younger with time. This would be in accordance with the funnel model, proposing that the phylotypic stage is at the very beginning of ontogeny.

Discussion

This analysis shows the impact data normalization and processing steps can have on the results. In this particular case, it is interesting because the original study supports the hourglass model, while the same analysis with normalized data supports the funnel model.

Calculating TAI without log-transforming yields a biased estimation of transcriptome age as it will be dominated by a few outlier genes. Therefore, the original results would suggest that strongly expressed genes follow the hourglass model, which is also interesting, however this does not apply when giving more weight to lowly expressed genes.

Since the publication of this paper, the bias discussed here has been addressed using a more detailed approach (Piasecka et al., 2013) where sets of genes over-expressed at the same developmental stage are identified and analysed independently. This approach is robust to strongly expressed genes and allows a more detailed analysis. The authors analyzed how different genomic features in the gene sets were constraint throughout development and found that only sequence conservation follows the hourglass model (sequence is more conserved for genes expressed in the mid-developmental stages), whereas expression, age, regulatory elements and orthology relationship are more constraint in early development, thus following the funnel model. This suggests the models can co-exist in different features.

References

Šestak, M. S., Božičević, V., Bakarić, R., Dunjko, V., & Domazet-Lošo, T. (2013). Phylostratigraphic profiles reveal a deep evolutionary history of the vertebrate head sensory systems. Frontiers in Zoology, 10, 18. http://doi.org/10.1186/1742-9994-10-18

Domazet-Lošo T, Tautz D. (2010). A phylogenetically based transcriptome age index mirrors ontogenetic divergence patterns. Nature, 468(7325):815-8. http://doi.org/10.1038/nature09632

Piasecka B, Lichocki P, Moretti S, Bergmann S, Robinson-Rechavi M (2013) The Hourglass and the Early Conservation Models—Co-Existing Patterns of Developmental Constraints in Vertebrates. PLOS Genetics 9(4): e1003476. doi: 10.1371/journal.pgen.1003476

Data files

Zebrafish expression data: https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE24616

Phylostrata: http://www2.unil.ch/cbg/index.php?title=Module_1:_Is_the_hourglass_model_for_gene_expression_really_supported_by_the_data%3F