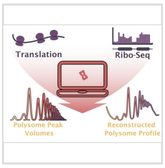


Jul 03, 2020

Using polyan: a Python package for modelling polysome profiles from ribosome density data

Tobias von der Haar[1]

[1]University of Kent

In Development This protocol is published without a DOI.

Tobias von der Haar
University of Kent

PROTOCOL CITATION

Tobias von der Haar 2020. Using polyan: a Python package for modelling polysome profiles from ribosome density data. **protocols.io**
https://protocols.io/view/using-polyan-a-python-package-for-modelling-polyso-bh4yj8xw

CREATED

Jul 01, 2020

LAST MODIFIED

Jul 03, 2020

PROTOCOL INTEGER ID

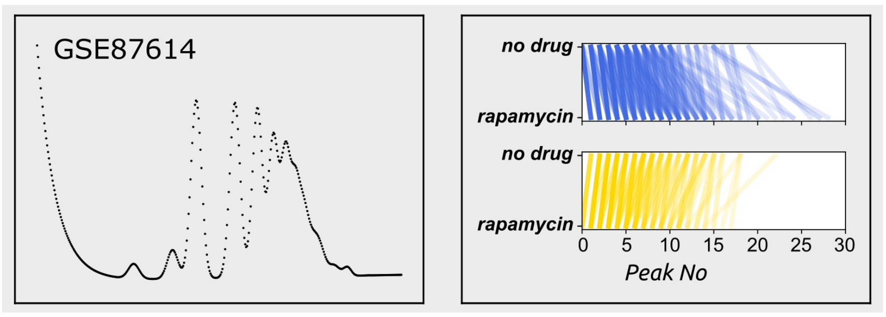
38776

General Functionality

1 polyan provides a number of Python functions for modelling polysome profiles from input data that relate to the transcript abundance and Ribosome occupancy for an organism. Typically, such data are intended to be the result of ribosome footprinting experiments, but in principle any experimental or computational technique that yields information on how many rbosomes are bound to the transcripts of an organisms can be used as input.

polyan currently provides three functions: fp2poly() calculates peak volume information for a modelled polysome profile, based on a single input dataset. plot_poly() generates plotting coordinates for the modelled profile, which can then be plotted using standard Python plotting functions. compare_profiles() reads in ribosome density data for two conditions and maps the principal shifts between modelled polysome peaks between these two conditions, thereby providing a road map of translational control under these conditions.

2 Example output from polyan is shown below, including a modelled polysome profile calculated from NCBI GEO dataset GSE87614 from Zinshteyn et al. 2017 (left), and the compare_profile() output for two datasets from Nedialkova and Leidel 2015 (right), mapping the major upward and downward movement of mRNAs upon treatment with rapamycin.



Prerequisites

3 polyan has been tested on Windows and Linux systems running Python 3.7.6, which was installed as part of the Anaconda package. However, polyan should run in all Python 3-based environments.

polyan has a number of dependencies on other packages, all of which should be available as part of any standard python installation, including pandas, numpy, pyplot, scipy and json.

Installation

4 The polyan package is available from pypi using the pip installer, which is included with most Python distributions. For installation , execute

```
pip install polyan
```

Where Python was installed as part of Anaconda in windows, this command may need to be executed in an Anaconda prompt window rather than a general command prompt window.

5 Import polyan into your Python environment by executing

```
import polyan
```

Data preparation

6 polyan requires the source data to be formatted as a pandas dataframe, which needs to contain at least two columns for gene names and Ribosome binding data, and may contain a third column for RNA abundance data. Where no RNA abundance data are specified, generic RNA abundance data are used.

An example dataset is attached (**GSE87614.csv**), which if present in a working directory can be loaded into a dataframe by executing

```
import pandas as pd
df = pd.read_csv('GSE87614.csv')
```

polyan provides support datasets for *Saccharomyces cereisiae* and for human HEK293 cells. It is possible to use

7 polyan with any other organism, but then support datasets need to be provided by the user including a reference list of gene names with corresponding transcript lengths.

8 The default column names of the dataframe are ORF, RNA_Prints and Ribo_Prints. Other column names can be used but then columns containing the data need to be explicitly listed when the polyan functions are executed. The ORF column must list systematic gene names for S. cerevisiae (eg YAL004W), and the ENSEMBL gene name for human genes (eg ENSG00000210100). RNA_Prints and RIbo_Prints must contain data proportional to the number of observed prints in Next Generation Sequencing (NGS) experiments, ie must not contain RPKM data.

Modelling polysome profiles

9 Polysome profiles can be modelled by consecutive execution of the fp2poly() and plot_poly() functions, followed by plotting of the x and y coordinates provided by plot_poly() using general plotting packages such as pyplot. The default settings of fp2poly() work on yeast data conforming to the requirements explained in steps 6-8, as is the case for the dataset loaded in step 6. provided that polyan has been installed and imported correctly, peak volumes for this dataset can be generated by executing

```
peak_vols = polyan.fp2poly(df)
```
python 3

and coordinates for the polysome trace can be calculated from peak_vols by executing

```
trace = polyan.plot_poly(peak_vols)
```

"trace" is a list containing two vectors corresponding to the x- and y-axis coordinates, respectively.
These coordinates can then be plotted via any suitable plotting package. The example output shown in step 2 can be reproduced using pyplot by executing

```
import matplotlib.pyplot as plt
fig,ax = plt.subplots()
ax scatter(trace[0],trace[1],s=2,c='black')
plt.show()
```

Mapping movement between polysome peaks

10 Changes in ribosome density between two conditions can be mapped using the compare_profiles() function of polyan. the results are trturned as pyplot fig and ax objects, which can be visualised using the plt.show() function of pyplot. To compare how transcripts shift between modelled polysome peaks eg when comparing two datasets stored in two dataframes df1 and df2, execute

```
polyan.compare_profiles(df1,df2)
```

The example otuput shown in step 2 was generated by comparing datasets **GSE67387_WT.csv** and **GSE67387_rapamycin.csv** , from the data by Nedialkova and Leidel (2015).

Changing parameters for fp2poly()

11 Where column names in the dataframes differ from the pre-specified "ORF","RNA_Prints" and "Ribo_Prints", column names to be used can be specified for both fp2poly() and compare profiles. For example where gene names are given in

col1, Ribo_Prints in col2 and RNA_Prints in col3:

```
polyan.fp2poly(df, df_columns=['col1','col2','col3')
```

The order of the column names must be Gene names, then Ribosome density data, then RNA abundance data (if used).

12 fp2_poly currently provides context data (gene lengths and generic RNA abundance data) for *s.cerevisiae* and for HEK293 cells. which parameter set to use is specified by the parset parameter, which can either be parset = 'Scer' (default) or parset = 'HEK'. To calculate peak volumes for a HEK293 dataset with standard column names, the command is

```
peak_vols = polyan.fp2poly(df,parset='HEK')
```