Supplementary material III

The Yeast Pathway Kit: a method for metabolic pathway assembly with automated documentation through simulation software.

Filipa Pereira[1], Flávio de Azevedo, Nadia Skorupa Parachin[2,3], Bärbel Hahn-Hägerdal[2], Marie-Francoise Gorwa-Grauslund [2] and Björn Johansson[1]*

[1] CBMA - Center of Molecular and Environmental Biology, Department of Biology, University of Minho, Campus de Gualtar, Braga, 4710-057, Portugal

[2] Department of Applied Microbiology, Lund University, SE-22100 Lund, Sweden

* To whom correspondence should be addressed. Tel: +351 253601517. Fax: +351 253678980. Email: bjorn_johansson@bio.uminho.pt.

[3] Present Address: Nadia Skorupa Parachin, Departamento de Biologia Celular, Instituto de Ciências Biológicas, Universidade de Brasília, Brasília, 70790-190-Brasília-DF, Brazil

The ypkpathway software

Ypkpathway is a software tool for the automated planning of metabolic pathway assemblies using the Yeast Pathway Kit protocol. Ypkpathway can be installed by first installing the free Anaconda Scientific Python distribution from Continuum analytics. This will ensure that all requirements are automatically installed, and the whole installation is located in a folder in the user's directory in a folder that is easily (re)movable if necessary. Anaconda and ypkpathway are available for the platforms Windows, MacOSX and Ubuntu linux.

Anaconda installation on Windows 7, MacOSX or Ubuntu.

Go to the website of Anaconda (Fig 1). Download the Anaconda installation file for your operating system and follow the installation instructions and select default options whenever applicable.

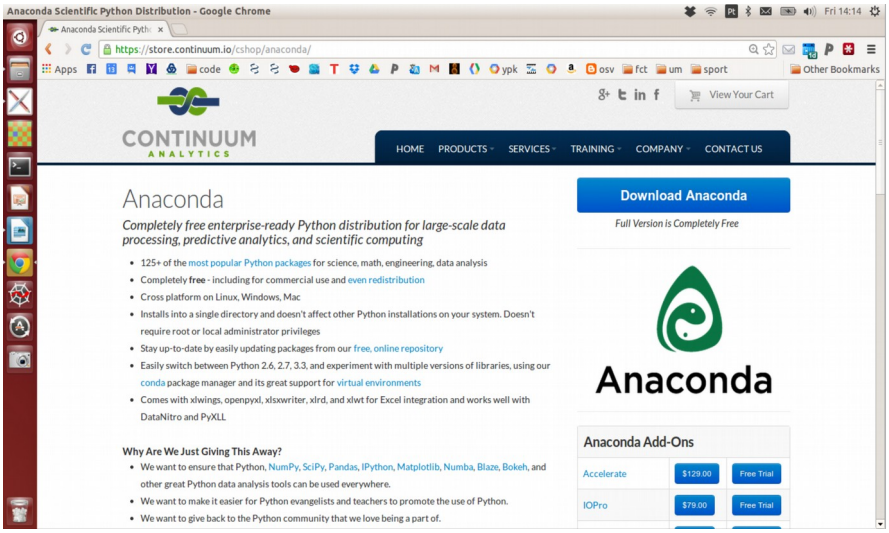

Fig 1: Anaconda download site

The installation is large, but files are only installed in one directory in the users private folder. The next step is to start the Anaconda Launcher. How to do this may differ between platforms. The MacOSX installer leaves a green Anaconda shortcut on the desktop (Fig 2).

Fig 2: The Anaconda Launcher icon is placed on the MacOSX desktop in the upper left corner.

The windows start menu will have a Launcher shortcut as well. For Ubuntu linux it is necessary to update launcher by the command “conda update launcher” at the command line.

The launcher should look like the picture in Fig 3. The appearance might vary slightly depending on platform.

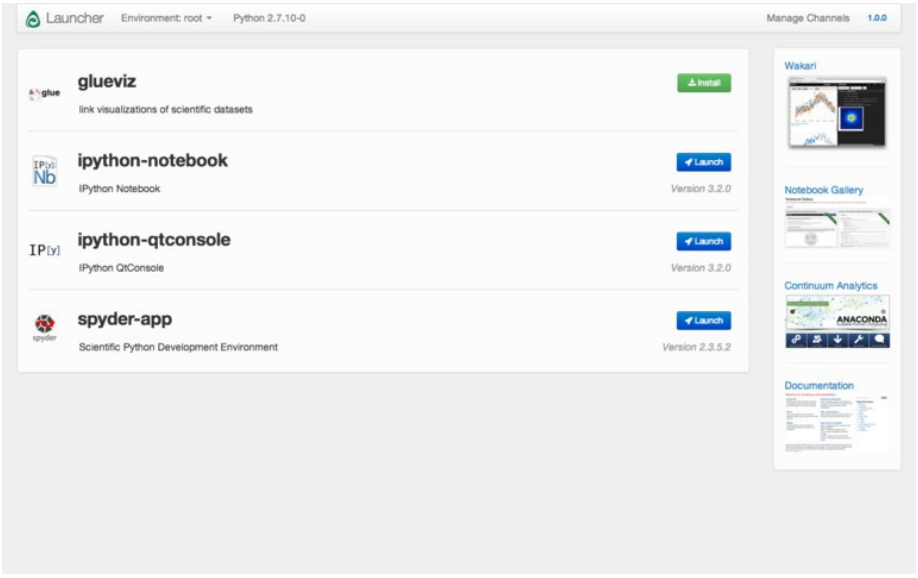

Fig 3: Anaconda Launcher on MacOSX

Adding a software channel

It is necessary the tell Anaconda where to find ypkpathway. We can do this by adding a software channel to Launcher. Click on the “**Manage channels**” link in the upper right corner of the window (Fig 3). Add the channel “**BjornFJohansson**” as shown in Fig 4 and click **Add channel** followed by **submit**.

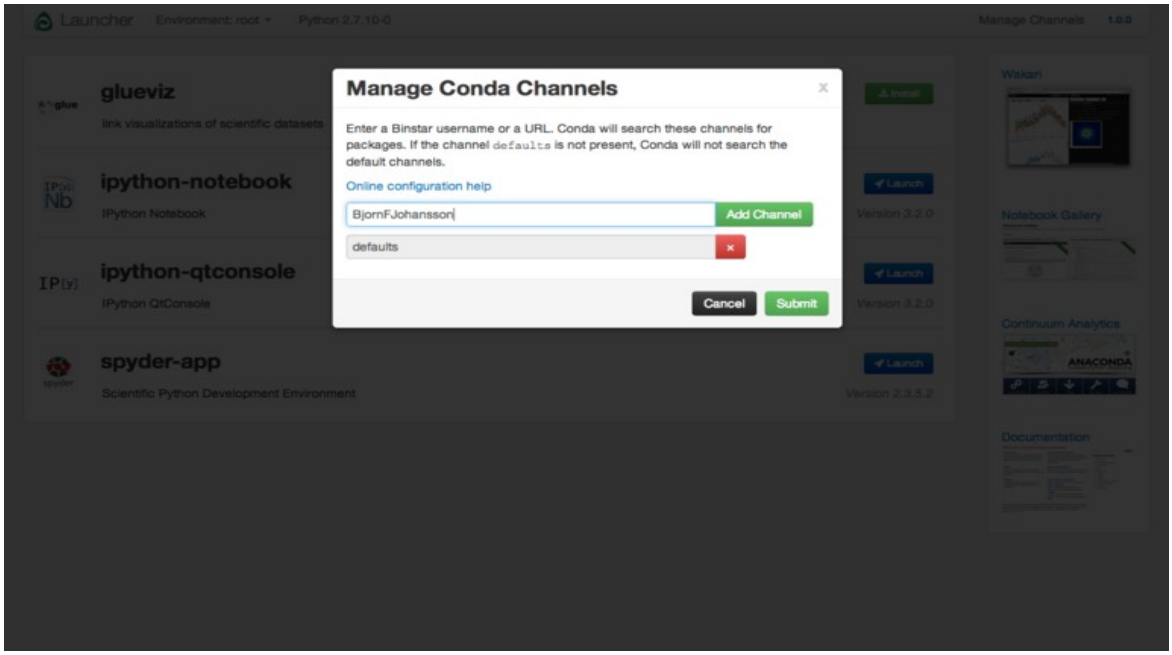

Fig 4: Adding the Anaconda software channel **BjornFJohansson**

When the channel was successfully added, a “**ypkpathway**” entry should be visible in the list of apps (Fig 5). Click on the link to install, after which the link should turn blue and change from “Install” to “Launch”. Click on the blue “Launch” button to start ypkpathway.

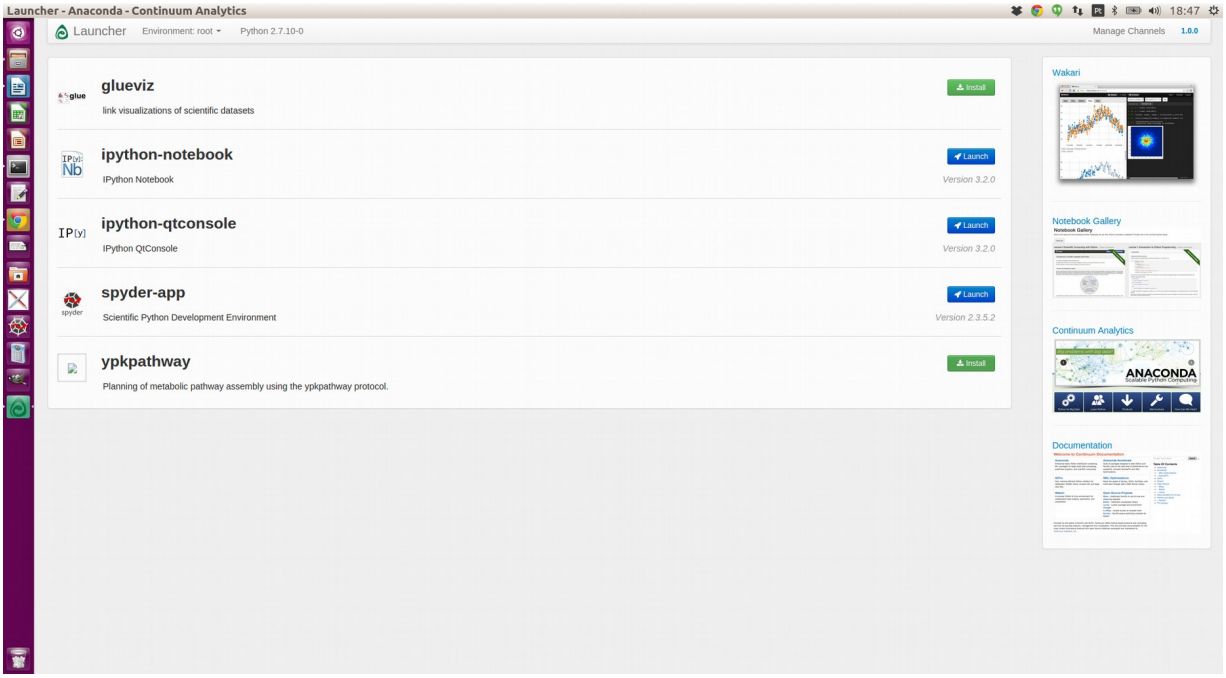

Fig 5: Ypkpathway app in launcher list on the Ubuntu platform

After successfully starting ypkpathway, the main window should have the appearance in Fig 6 .

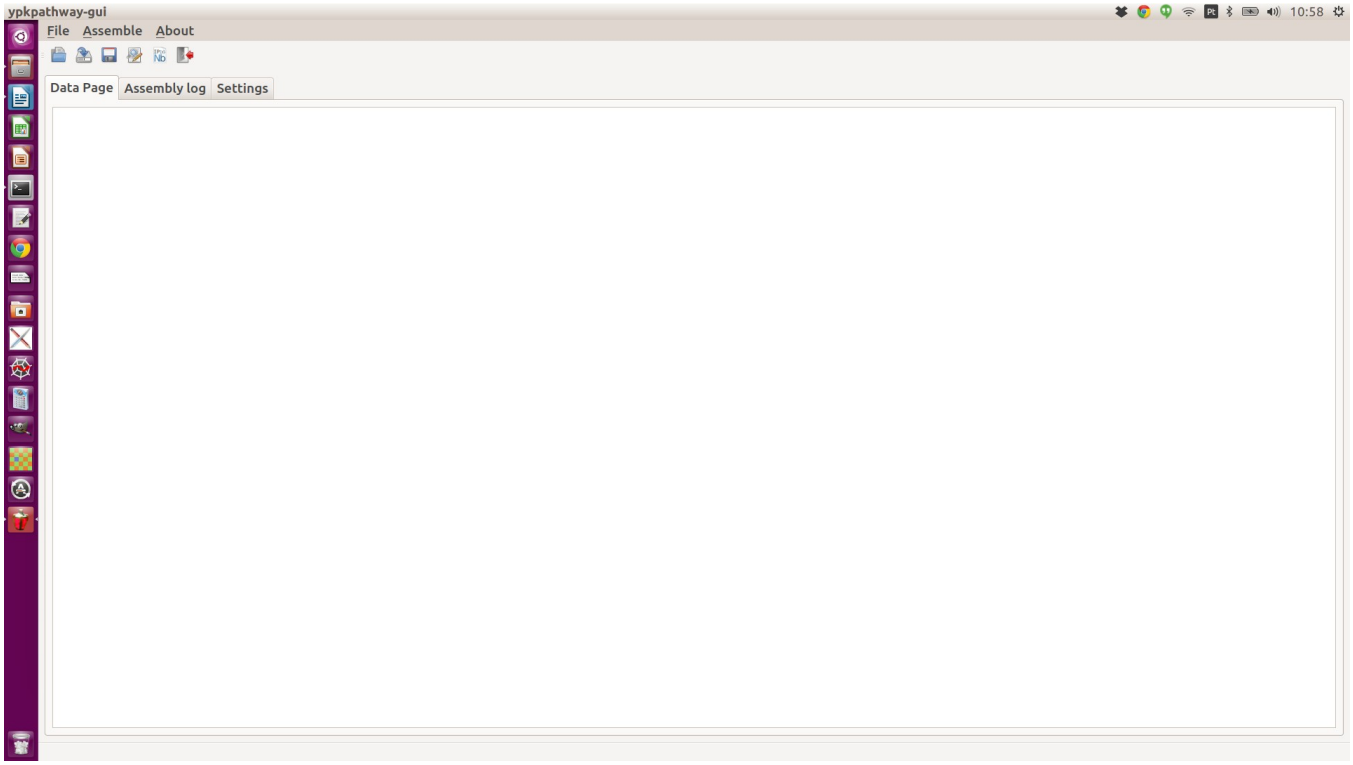

Fig 6: The ypkpathway main window on the Ubuntu platform

Using ypkpathway

The ypkpathway uses a text file for indata, alternatively text files can be pasted directly in the text window underneath the “Data page” tab. The supplementary data comes with seven data files named pth1.txt - pth7.txt. The text file pth6.txt was opened in ypkpathway in Fig 7.

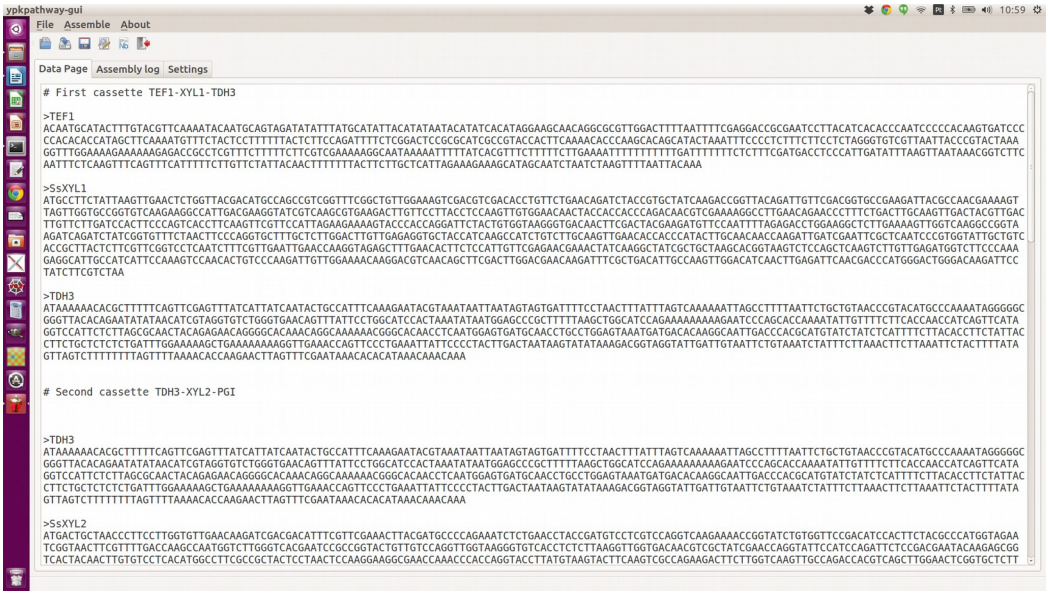

Fig 7: Text file pth6.txt in ypkpathway

The text file is simply a list of sequences in either FASTA or Genbank format (mixed formats are allowed) that reflect the sequences to be assembled. The details of the file syntax will be dealt with in a later section (**File formatting**). Assembly is started by choosing the Assemble alternative under the Assemble menu or by clicking on the “Assemble” button (Fig 8)

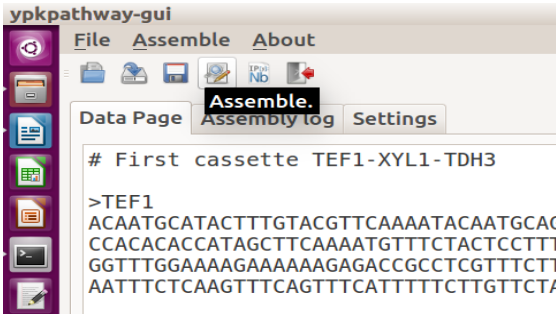

Fig 8: Start assembly by pressing the assembly button

The assembly process might take some time depending on the capability of the computer. An assembly log is written to the “Assembly log” window. The assembly process creates a folder with the same name as the text file (without the “.txt” extension) in the same location. When the assembly is finished (Fig 9), open the main Ipython notebook file describing the pathway by clicking the “Open pathway” icon or select the “Open pathway.” alternative under the Assemble menu (Fig 10).

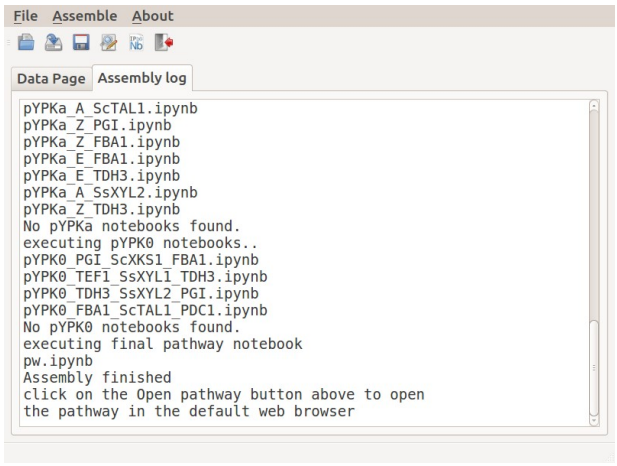

Fig 9: Assembly finished

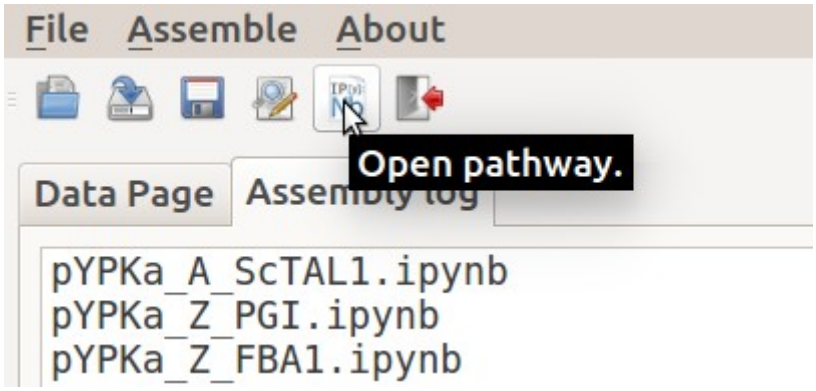

Fig 10: Opening the pathway in the default web browser

The Pathway folder

The Pathway folder contains a series of IPython notebook and sequence files describing the assembly. There is a notebook file describing each plasmid constructed. The notebook “pw.ipynb” describe the final pathway assembly and is also the notebook that is opened by the “Open pathway” link (Fig 11).

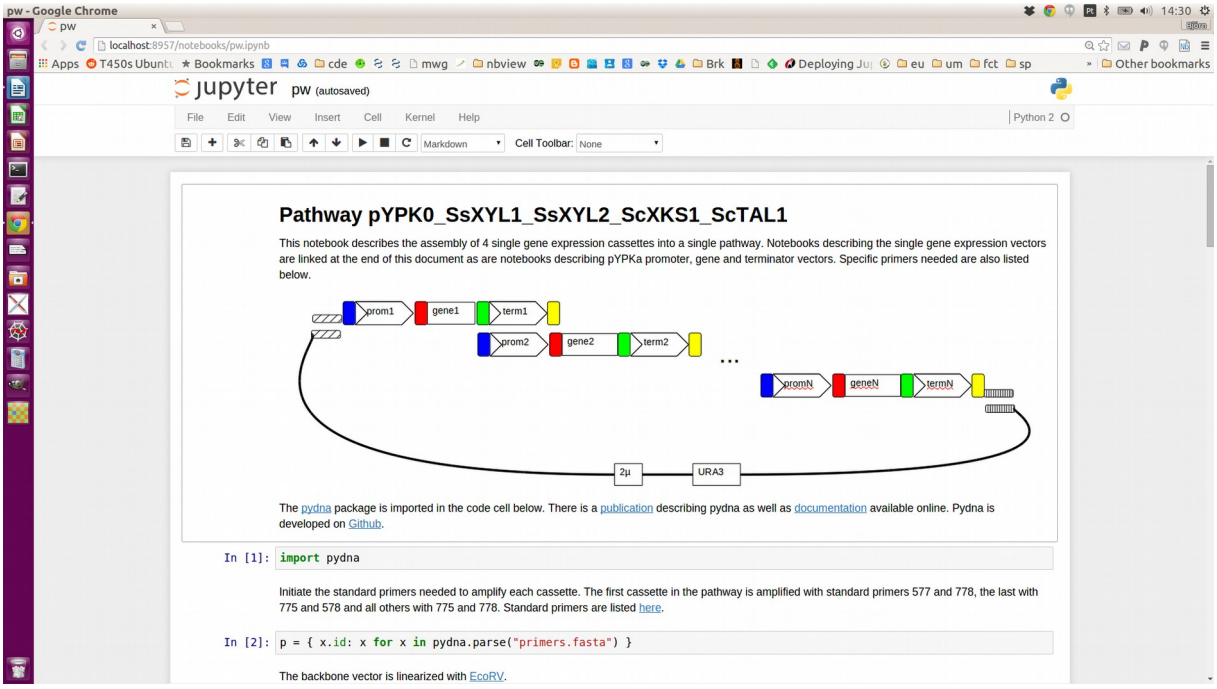

Fig 11: pw.ipynb for pth6.txt opened in the google chrome browser on Ubuntu linux

At this stage the ypkpathway program is no longer needed and can be closed. The raw data from the Data page as well as the assembly log is copied to the pathway folder, so the folder is a self contained description of the assembly.

Interpretation of the main pathway notebook

The IPython notebooks describing the pathway and intermediate vectors contain comments images and executable python code in separate numbered cells (Fig 12). The Ypkpathway software generates the notebook files, but the actual assembly strategy is described in pydna code within the notebooks themselves. The notebook pw.ipynb describe the assembly of the pathway.

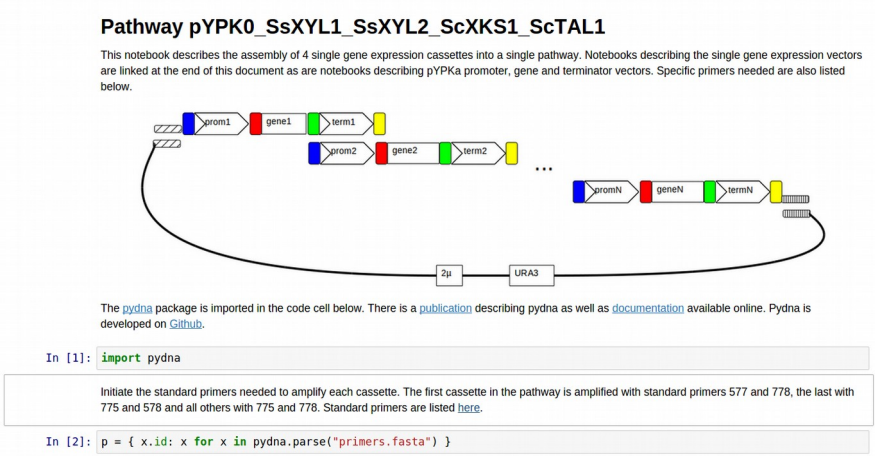

Fig 12: IPython notebook document in pw.ipynb after simulation of the pth6.txt example showing two text cells and two code cells

The final pathway assembly is simulated by first reading the sequences of single gene expression constructs from local files (Figure 13).

```
In [5]: cas_vectors ='''
                    pYPK0_TEF1_SsXYL1_TDH3.gb
                    pYPK0_TDH3_SsXYL2_PGI.gb
                    pYPK0_PGI_ScXKS1_FBA1.gb
                    pYPK0_FBA1_ScTAL1_PDC1.gb'''.splitlines()
        template_vectors = [pydna.read(v.strip()) for v in cas_vectors if v.strip()]
        template_vectors
```

Fig 13: single gene expression vector sequences read from files

The assembly simulation is visualized in a simple text based figure (Fig 14). In this case, four single gene expression constructs are assembled.

```
In [11]: candidate.figure()

Out[11]:  -|pYPKpw|124
          |         \/
          |         /\
          |         124|2524bp_PCR_prod|711
          |                           \/
          |                           /\
          |                           711|2924bp_PCR_prod|1013
          |                                             \/
          |                                             /\
          |                                             1013|3567bp_PCR_prod|643
          |                                                                 \/
          |                                                                 /\
          |                                                                 643|3009bp_PCR_prod|242
          |                                                                                   \/
          |                                                                                   /\
          |                                                                                   242-
          |                                                                                     |
          ------------------------------------------------------------------------------------------
```

Fig 14: Visualization of the assembly process described by pth6.txt

Single gene expression vector notebooks

Download pYPK0_TEF1_SsXYL1_TDH3

```
In [28]: import pydna
         reloaded = pydna.read("pYPK0_TEF1_SsXYL1_TDH3.gb")
         reloaded.verify_stamp()

Out[28]: cSEGUID_M2q6_Lxm8JLLUJDAn3vRRNk5KtA
```

pYPKa vector notebooks

The cut and paste cloning of

All primers needed for assembly are listed in a text cell in the end of the notebook. Primers needed for the assembly are listed in the end of the notebook (Figure 15).

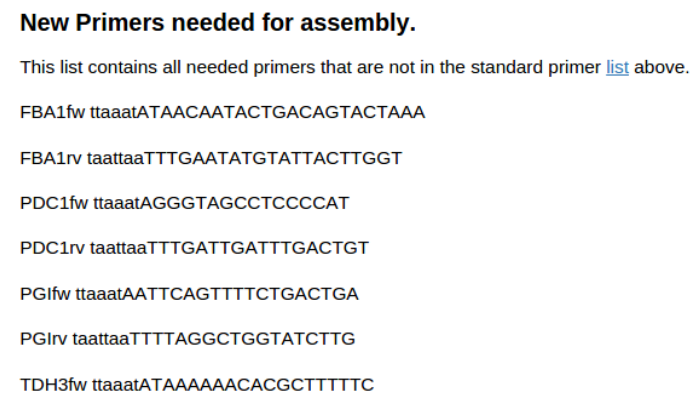

Fig 15: A part of the primer list in the end of the notebook

The construction strategies for intermediate vectors needed for the assembly can be inspected by the hyper links in the end of the notebook (Fig 16).

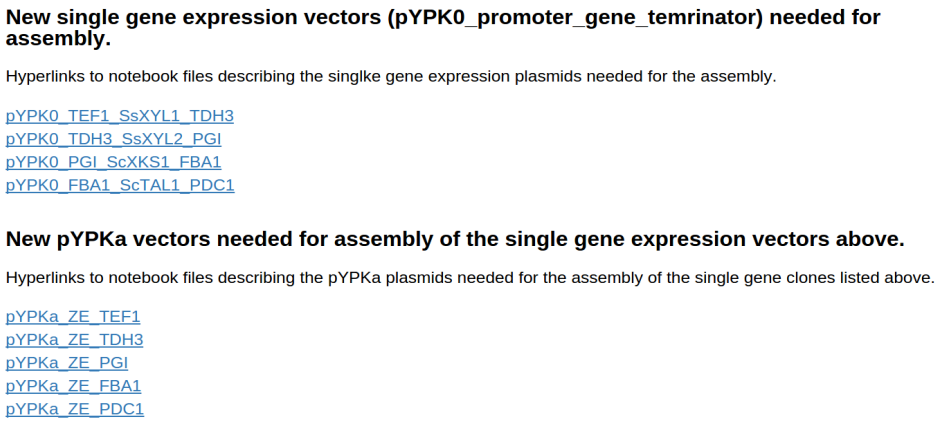

Fig 16: List of links to constructs needed for the assembly

File formatting

The data file is simply a list of terminator-promoter fragments and genes that should be assembled. Sequences has be in either FASTA or Genbank format in the order they should be assembled. Formats can be mixed, but there must be at least one blank line between each sequence. The pth6.txt datafile in the previous example has the structure depicted below, sequences are truncated for clarity and only the first six out of twelve sequences are shown for clarity:

```
>TEF1
ACAATGC...AAA

>SsXYL1
atgatc...taa

>TDH3
ATAAAAA...AAA


>TDH3
ATAAAAA...AAA

>SsXYL2
atgcac...tag

>PGI
TGTTTAA...AAA
```

Note that the TDH3 sequence is repeated, first as terminator and then as promoter of the next cassette. The pathway folder will contain a description of the construction of the pYPKa vectors needed for the assembly of the single gene cassette vectors:

```
pYPKa_Z_TEF1
pYPKa_A_SsXYL1
pYPKa_E_TDH3
pYPKa_Z_TDH3
pYPKa_A_SsXYL2
pYPKa_E_PGI
```

There will also be a description of the assembly of the single yeast expression vectors **pYPK0_TEF1_gene1_TDH3** and **pYPK0_TDH3_gene2_PGI** by gap repair. The final pathway will be described by a IPython notebook file called **pw.ipynb.**

One of the main points of the Yeast Pathway Kit is the reuse of cloned genetic parts, especially terminator-promoter and genes cloned in the pYPKa vector as in the example above.

Sequences can also be given to the ypkpathway program in the form of the sequences of the entire plasmids that were created in previous experiments. These plasmid sequences will be automatically recognized by ypkpathway, and the construction step of these will be omitted. Below is an example of feeding the ypkpathway algorithm with six pYPKa sequences (truncated for clarity):

```
>pYPKa_Z_TEF1tp
tcgcgcgttt...ACAATG...AAA...ctttcgtc
>pYPKa_A_gene1
tcgcgcgttt...atgatc...taa...ctttcgtc
>pYPKa_E_TDH3tp
tcgcgcgttt...ATAAAA...AAA...ctttcgtc
```

```
>pYPKa_Z_TDH3tp
tcgcgcgttt...ATAAAAA...AAA...ctttcgtc

>pYPKa_A_gene2
tcgcgcgttt...atgcac...tag...ctttcgtc

>pYPKa_E_TPI1tp
tcgcgcgttt...TGTTTAA...AAA...ctttcgtc
```

In this case, no pYPKa vector construction will be simulated. Similarly, sequences of single gene expression vectors (**pYPK0_promoter_gene_terminator** vectors are also recognized. The indata below has two such single gene expression cassettes:

```
>pYPK0_TEF1tp_gene1_TDH3tp
tcgcgcgttt...ACAATGC...AAA...atgatc...taa...ATAAAAA...AAA...ctttcgtc

>pYPK0_TDH3tp_gene2_TPI1tp
tcgcgcgttt...ATAAAAA...AAA...atgcac...tag...TGTTTAA...AAA...ctttcgtc
```

Only the final pathway will be assembled in this case. The real usefulness of this automatic sequence recognition comes from the mixing of the three kinds of sequences. In the example below, two pYPKa sequences were supplied, one for the first promoter and one for the first gene. We will assume that the TPI1 terminator is not available in a **pYPKa_E** clone and is therefore supplied as a linear sequence. The **pYPK0_TPI1tp_gene2_TPI1tp** vector single gene expression vector was presumably constructed in a previous experiment and its sequence is given in the tet file.

```
>pYPKa_Z_TEF1tp
tcgcgcgttt...ACAATGC...AAA...ctttcgtc
>pYPKa_A_gene1
tcgcgcgttt...atgatc...taa...ctttcgtc
>TPI1tp
ATAAAAA...AAA

>pYPK0_TPI1tp_gene2_TPI1tp
tcgcgcgttt...ATAAAAA...AAA...atgcac...tag...TGTTTAA...AAA...ctttcgtc
```

Supplementary Figure 15. Representation of cloning plan for the final sequence.

The second “(plan)” link (Supplementary Figure 13-3) shows a plan for the construction of the first pYPK0_tp_gene_tp clone. These clones are assembled from three pYPKa derived PCR products for each element and linearized pYPKpw (Supplementary Figure 16).