Installing the Rp-Bp pipeline

This document describes detailed installation instructions for the Rp-Bp pipeline. These steps have been primarily tested on Ubuntu.

Prerequisites

The pipelines make use of a number of standard bioinformatics tools. All of these must be installed and available on the `$PATH` for the pipeline to work correctly. All of the pipeline scripts check that the required programs are available before executing. If any required programs cannot be found, the script prints an error message about the missing program and does not continue. The versions used during development and testing are specified below. For most of the tools, any recent version should be sufficient. If problems arise, though, please use the version indicated below

- bowtie2, version 2.2.6
- flexbar, version 2.5
- SAMtools, version 1.2
- STAR, version 2.4.1d

Helper script

While not officially supported, the script here downloads and compiles the necessary prerequisites in `$HOME/install` and installs them in `$HOME/local`.

It uses `apt-get` to install the Intel thread building blocks and CMake. The direct Rp-Bp dependencies mentioned above are simply downloaded and placed in the appropriate location. The relevant locations (`$HOME/local/bin`) must be add to the `$PATH` before the executable files can be found. `sudo` permissions are required for the calls to `apt-get`.

OpenBLAS

If installation fails due to missing `OpenBLAS` dependencies for `scipy`, please follow the instructions here.

Back to top

Simple installation

We recommend installing the application in a virtual environment as described below. If this is not desired for some reason, the following instructions can be used to install the package without sudo access in a user's home directory.

The commands below are presumably executed in a directory like `$HOME/install`. They install the python executables into `$HOME/local/bin` (or wherever the `prefix` option is located). So that directory must be in the `$PATH`. This can be accomplished by adding a line like `export PATH=$HOME/local/bin:$PATH` in the file `.bashrc` on Ubuntu.

N.B. It is very important that pip and wheel are upgraded!

N.B. The `--user` option can also be given to pip3 for installation within the user's home directory. The installation is then compatible with system-wide python3 installations without requiring sudo access.

```
### Lines to add to .bashrc

# for the installation process
export PATH=$HOME/local/bin:$PATH

### Downloading and installing the required software

# Download, extract and install Python 3.
wget https://www.python.org/ftp/python/3.5.1/Python-3.5.1.tgz && tar -xvf Python-3.5.1.tgz &

# Upgrade pip.
pip3 install --upgrade pip wheel

# Clone the git repository.
git clone git@github.com:dieterich-lab/rp-bp.git

# Change into the rp-bp directory and build the package.
cd rp-bp && pip3 install --verbose -r requirements.txt

# or install the package in the user's home directory
cd rp-bp && pip3 install --verbose --user  -r requirements.txt
```

The build process includes compiling several libraries for optimized numerical calculations. Due to the optimized nature of these libraries, the initial installation can take up to an hour.

Due to the `--verbose` flag, much debugging information is printed. In some cases, building some libraries may initially fail; this is due to a known issue with dependency-handling in pip. The following output (or something similar depending on packages already available) indicates installation succeeded:

```
Successfully installed cython-0.25.1 docopt-0.6.2 joblib-0.10.3
numpy-1.11.2 pandas-0.19.0 patsy-0.4.1 psutil-4.4.2 pyfasta-0.5.2
pysam-0.9.1.4 pystan-2.12.0.0 python-dateutil-2.5.3 pytz-2016.7
pyyaml-3.12 rpbp-1.0 scipy-0.18.1 six-1.10.0 statsmodels-0.6.1
tqdm-4.9.0 Cleaning up...
```

Back to top

Virtual environment installation

These instructions explain how to install the software and most dependencies from scratch without required root access. It only requires standard development libraries and tools, like gcc and the gzip development headers. The python build scripts will also output a line like "The necessary bits to build these optional modules were not found" if any optional libraries, developoment headers, etc., are not found.

The commands below are presumably executed in a directory like `$HOME/install`. They install the python executables into `$HOME/local/bin` (or wherever the `prefix` option is located). So that directory must be in the `$PATH`. This can be accomplished by adding a line like `export PATH=$HOME/local/bin:$PATH` in the file `.bashrc` on Ubuntu.

```
### Lines to add to .bashrc

# for the virtual environment
# See http://www.simononsoftware.com/virtualenv-tutorial-part-2/ for more details.
export WORKON_HOME=$HOME/.virtualenvs
source $HOME/local/bin/virtualenvwrapper_lazy.sh

# for the installation process
export PATH=$HOME/local/bin:$PATH

### Downloading and installing the required software

# Download, extract and install Python 2. This is necessary for creating the virtual enviror
wget https://www.python.org/ftp/python/2.7.11/Python-2.7.11.tgz && tar -xvf Python-2.7.11.tg

# Download, extract and install Python 3. This is necessary for the pipelines
wget https://www.python.org/ftp/python/3.5.1/Python-3.5.1.tgz && tar -xvf Python-3.5.1.tgz &

# Upgrade both versions of pip.
pip2 install --upgrade pip && pip3 install --upgrade pip wheel

# Install the virtual environment wrapper for Python 2.
pip2 install --upgrade virtualenvwrapper
```

```
# Create a virtual environment using Python 3.
mkvirtualenv rpbp -p $HOME/local/bin/python3

# Activate the virtual environment
workon rpbp

# Clone the git repository.
git clone git@github.com:dieterich-lab/rp-bp.git

# Change into the rp-bp directory and build the package.
cd rp-bp && pip3 install --verbose  -r requirements.txt
```

The build process includes compiling several libraries for optimized numerical calculations. Due to the optimized nature of these libraries, the initial installation can take up to an hour.

Due to the `--verbose` flag, much debugging information is printed. In some cases, building some libraries may initially fail; this is due to a known issue with dependency-handling in pip. The following output (or something similar depending on packages already available) indicates installation succeeded:

```
Successfully installed cython-0.25.1 docopt-0.6.2 joblib-0.10.3 numpy-1.11.2 pandas-0.19.0 p
Cleaning up...
```

To use the programs in the future, use the `workon` command to ensure the virtual environment is active.

```
# Activate the virtual environment
workon rpbp
```

Back to top

Anaconda installation

The package can also be installed within an anaconda environment. Importantly, the version of gcc included by default does not properly compile the Stan model files, so it must be updated first.

```
# update gcc
conda install -c salford_systems libgcc-6=6.2.0

# create the rpbp anaconda environment
conda create -n rpbp python=3.5 anaconda

# activate it
source activate rpbp
```

```
# clone the repo
git clone git@github.com:dieterich-lab/rp-bp.git

# change into the folder
cd rp-bp/

# install
pip install -r requirements.txt --verbose --log log.txt
```

Back to top