Patient-level prediction installation guide

Jenna Reps

2018-05-30

Contents

1 Introduction

1.1 Windows Users

The OHDSI Patient Level Prediction (PLP) package requires installing:

- R (https://cran.cnr.berkeley.edu/) - the latest version is recommended
- Rstudio (https://www.rstudio.com/)
- Java (http://www.java.com)
- RTools (https://cran.r-project.org/bin/windows/Rtools/)
- Anaconda 3.6 (https://www.continuum.io/downloads) - this will require checking your path variable to ensure the correct python is used by R - more instructions below

1.2 Mac/Linux Users

The OHDSI Patient Level Prediction (PLP) package requires installing:

- R (https://cran.cnr.berkeley.edu/) - the latest version is recommended
- Rstudio (https://www.rstudio.com/)
- Java (http://www.java.com)
- Xcode command line tools(run in terminal: xcode-select –install) [MAC USERS ONLY]
- Python 3.6 (https://www.python.org/downloads/) - this will require checking your path variable to ensure this version python is added - more instructions below

1.2.1 Setting up Python for Mac/Linux Users

After installing python 3.6 you can type python3 to open python in a terminal.

To get the package dependencies, in a terminal run:

```
pip3 install --upgrade pip
pip3 install -U NumPy
pip3 install -U SciPy
pip3 install -U scikit-learn
pip3 install --upgrade tensorflow
pip3 install keras
```

2 Checking Python in Path

Both anaconda and python 3.6 change the path by default. You can access the path variable in R using `Sys.getenv('PATH')`. This should contain the location of your anaconda or python 3.6.

If you cannot find the anaconda/python in your path, then you need to add it:

2.1 Windows users need to:

my computer -> system properties -> advanced system settings Then at the bottom right you'll see a button: Environmental Variables, clicking on that will enable you yo edit the PATH variable to add the anaconda location.

2.2 Mac/Linux users need to:

Need to edit the bash profile to add it by running in the terminal: `touch ~/.bash_profile; open ~/.bash_profile;` and adding in the location of python 3.6.

Please note: if you update the path while R is open, you will need to shut R down and reopen before the path is refreshed.

3 Installing PatientLevelPrediction using drat

```
install.packages("drat")
drat::addRepo("OHDSI")
install.packages("PatientLevelPrediction")
# to get the latest PLP then use devtools
install.packages("devtools")
install_github("ohdsi/PatientLevelPrediction")
```

4 Installing PatientLevelPrediction using devtools

If the drat code fails as the repository is not available you can try:

```
install.packages("devtools")
library("devtools")
install_github("ohdsi/SqlRender")
install_github("ohdsi/DatabaseConnectorJars")
install_github("ohdsi/DatabaseConnector")
install_github("ohdsi/FeatureExtraction")
install_github("ohdsi/OhdsiSharing")
install_github("ohdsi/OhdsiRTools")
install_github("ohdsi/BigKnn")
install_github("ohdsi/PatientLevelPrediction")
```

5 Configuring PythonInR Mac/Linux users

Non-windows users need to specify their R environment to get R to use python 3.6 rather than the default python. In a new Rstudio session run this to open the environment file:

```
install.packages(â€~usethisâ€)
usethis::edit_r_environ()::edit_r_environ()
```

In the file that opens add and save: PATH= {The path containing the python 3} USESPECIALPYTHONVERSION=â€œpython3.6"

For example mine was: PATH="/Library/Frameworks/Python.framework/Versions/3.6/bin:/usr/local/bin:/usr/bin:/bin:/usr/
USESPECIALPYTHONVERSION="python3.6"

You now need to compile PythonInR so it uses python 3.6. In a new R studio session run:

```
Sys.setenv(USESPECIALPYTHONVERSION = "python3.6")
devtools::install_bitbucket("Floooo/PythonInR")
```

This should now set the PythonInR package to use your python 3.6.

6 Testing installation

To test the installation (including python) run:

```
library(DatabaseConnector)
connectionDetails <- createConnectionDetails(dbms = "sql_server", user = "username",
    password = "hidden", server = "your server", port = "your port")
PatientLevelPrediction::checkPlpInstallation(connectionDetails = connectionDetails,
    python = T)
```

To test the installation (excluding python) run:

```
library(DatabaseConnector)
connectionDetails <- createConnectionDetails(dbms = "sql_server", user = "username",
    password = "hidden", server = "your server", port = "your port")
PatientLevelPrediction::checkPlpInstallation(connectionDetails = connectionDetails,
    python = F)
```

If the above returns a value other than 1 then there is an issue. You can determine the issue by running: `PatientLevelPrediction::interpretInstallCode(N)` where `N` is the value returned by the plp installation check.

7 Adding Keras for deep learning

To add the R keras interface, in Rstudio run:

```
devtools::install_github("rstudio/keras")
library(keras)
install_keras()
```

8 Common install issues

This will be updated over time.

If you have an error when trying to install a package in R saying 'Dependancy X not available ...' then this cna sometimes be fixed by running `install.packages('X')` and then once that completes trying to reinstall the package that had the error.

I have found that using the github devtools to install packages can be impacted if you have multiple R sessions open as one session with a library open cause cause the library to be looked and this can prevent an install of a package that depends on that library.