Patient-level prediction installation guide

Jenna Reps

2018-05-15

Contents

1 Introduction

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/) [WINDOWS ONLY]
- 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

2 Checking Python [Important for Mac or Linux users]

To check python is correctly configured you need to make sure the anaconda bin is in the environmental path variable (and placed before any the default python bin). This can be tested by typing python in the terminal/command line and making sure anaconda loads. If it does not load anaconda, then R will also not load anaconda and this will cause the python models to fail. You can access the path variable in R using `Sys.getenv('PATH')`.

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

```r
install.packages("drat")
drat::addRepo("OHDSI")
install.packages("PatientLevelPrediction")
```

4 Installing PatientLevelPrediction using devtools

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

```r
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 Testing installation

To test the installation (including python) run:

```r
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:

```r
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.

6 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.