adan Package Vignette

Pau Carrió, Manuel Pastor

March 17, 2014

This vignette presents **adan** package through an example session. The aim of the package is assessing the reliability of predictions obtained by in silico methods. More details on the method REF.

Contents

1 Installation

Requirements

- The file adan_X.X.tar.gz.
- R installed with at least version 2.15.3 `http://www.r-project.org/`.
- R packages that should be already installed: Rcpp, pcaMethods, pls, ggplot2 , reshape, knitr. You can install them and their dependecies with commands

```
source("http://bioconductor.org/biocLite.R")
biocLite(c("Rcpp","pls","ggplot2","reshape","knitr",
"pcaMethods","Biobase","BiocGenerics","digest","plyr",
"gtable","reshape2","scales","proto","MASS","stringr",
"RColorBrewer","munsell","colorspace","dichromat","labeling"))
```

Installation

The package can be esaly installed directly from RStudio [1] or your prefered graphical user interface.

To install the package from an R terminal run[2]:

```
#
# in LINUX - MAC
#
install.packages("/PATH/TO/adan_X.X.tar.gz",
  repos=NULL,type="source")
#
# in WINDOWS
#
# check that Sys.getenv("TMPDIR") contains a full path in UTF-8
# change TMPDIR with Sys.setenv(TMPDIR="C:\\PATH\\TO\\PROPER\\TMPDIR")
# ( note the double backslash and full path)
#
install.packages("C:\\PATH\\TO\\adan_X.X.tar.gz",
```

[1] RStudio -> Tools -> Install Packages -> Install from file.

[2] Any text from a # character to the end of the line is taken to be a comment.

```
  repos = NULL,type="source")
#
```

For further details on the instalation process visit `http://cran.r-project.org/doc/manuals/R-admin.html#Installing-packages`.

2 Example Session

In this section we will walk through an example session to get an overview of the two main functions of **adan** package: `adan.build` and `adan.test`. For this session we will use a sample data set included in **adan** package. The sample data set is the solubility data set based on Delaney (DOI: `http://dx.doi.org/10.1021/ci034243x`). The solubility data set contains Pentacle molecular descriptors and solubility values. This sample data was splited into train and test series for demonstrative purposes only. When used in production, `adan.build` must be trained with all data available while `adan.test` is used for new compounds. A model was already build and their predictions are also available. Note that adan does not need any information on the model used to obtain the predictions.

First, we load the package with command:

```
library(adan)
```

Next, we load the data sets with command:

```
data(solubility.adan)
names(solubility)
[1] "MD"       "AqSol"    "set"      "train.md" "train.a"
[6] "test.md"  "test.a"   "train.p"  "test.p"
```

The list `solubility` has nine elements. We are only interested in `train.md`, `test.md`, `train.a`, `train.p` and `test.p`. (i.e. molecular descriptors for train and test sets, predictions for train and test set and activities for only train set.)

Next, we build an `adan` object with the data used to train the model[3]. The argument `scale.md` controls if molecular descriptors need to be scaled or not. In our example we do not scale the molecular descriptors.

```
adan.model <-
  adan.build (
              train.md = solubility$train.md,
              train.a  = solubility$train.a,
              train.p  = solubility$train.p,
              scale.md = FALSE
              )
```

`adan.model` is the object that should be queried to assess the prediction reliability for query compounds (here the test set). We call `test.adan` function.

```
adan.output <-
  adan.test (
              adan.model = adan.model,
              query.md = solubility$test.md,
              query.p  = solubility$test.p
             )
```

The output is a list with three elements:

- `categories`: A data.frame with the assigned `adan` category.

[3]The sign `$` is used to extract named elements out of an R list.

```
head(adan.output$categories)
   categories
2           0
4           0
5           0
6           0
7           0
10          0
```

- `errorCI`: A vector with the extrem values of the error confidence interval. If category is above 3 then a NA value is returned since the prediction is not reliable.

```
head(adan.output$errorCI)
   errorCI
1 1.059562
2 1.059562
3 1.059562
4 1.059562
5 1.059562
6 1.059562
```

- `classification`: A data frame with information on which rules where broken.

```
 head(adan.output$classification)
   d2cXTe d2nXTe d2mTe d2cYTe d2nYTe d2nSDEP
2       0      0     0      0      0       0
4       0      0     0      0      0       0
5       0      0     0      0      0       0
6       0      0     0      0      0       0
7       0      0     0      0      0       0
10      0      0     0      0      0       0
```

3 "How to" with your data

In this section we show you how to use adan with your data step by step.

Prerequisites

- You have the molecular descriptors, activities and predictions for the train set and molecular descriptors and predictions for the query compounds in CSV files. Each row in each file corresponds to a compound.

Step by step

Start an R session and load `adan` package:

```
library(adan)
```

Load train and query data

```
train.md <- read.table( "my_train_md.csv" ,        header = FALSE, sep = ",",
  stringsAsFactors = FALSE )
# if activities are stored as columns use the brackets to
# select the column. Do not forget the comma inside the brackets.
train.a <- read.table( "my_train_activities.csv" , header = FALSE, sep = ",",
  stringsAsFactors = FALSE )[,1]
train.p <- read.table( "my_train_predictions.csv", header = FALSE, sep = ",",
  stringsAsFactors = FALSE )[,1]
#
```

```
#
query.md <- read.table( "my_query_md.csv" ,        header = FALSE, sep = ",",
  stringsAsFactors = FALSE )
query.p <- read.table( "my_query_predictions.csv", header = FALSE, sep = ",",
  stringsAsFactors = FALSE )[,1]
```

Next, we build an adan model. If your molecular descriptors need to be scaled set the argument `scale.md` accordingly.

```
adan.model <-
  adan.build (
              train.md = train.md,
              train.a  = train.a,
              train.p  = train.p,
              scale.md = FALSE
              )
```

Now you are ready to query your adan model with your query compounds.

```
adan.output <-
  adan.test (
              adan.model = adan.model,
              query.md = query.md,
              query.p  = query.p
             )
```

You can save the adan output with

```
write.table( adan.output$categories,  "my_categories.csv", sep =",",row.names=FALSE)
write.table( adan.output$errorCI,     "my_errorCI.csv",    sep =",",row.names=FALSE)
write.table( adan.output$classification,
                                      "my_classification.csv", sep =",",row.names=FALSE)
```