The euptf package

Mélanie Weynants and Brigitta Tóth
mweynants@gmail.com

March 9, 2016

The euptf package provides pedotransfer functions (PTFs) for the prediction of hydraulic properties in European soils. The PTFs were trained and validated on subsets of the European Hydropedological Data Inventory (Weynants et al., 2013, EU-HYDI,). the methodology and the results are published in the European Journal of Soil Science (Tóth et al., 2014).
This vignette is a short tutorial explaining how to use the PTFs and how to export the results outside R for non-R users.

1 Installation

A working R installation is needed to use the PTFs in euptf. Please refer to http://cran.r-project.org/ for the installation of the latest R version.
The euptf package is available from the European soil portal: http://eusoils.jrc.ec.europa.eu. To install euptf, open an R session, set the working directory to where you downloaded the package using setwd() and type the following command in the console window:

```
> # install dependencies if need be
> IP <- installed.packages()[,1]
> if (!"rpart" %in% IP){install.packages("rpart")}
> if (!"gWidgets" %in% IP){install.packages("gWidgets")}
> if (!"gWidgetstcltk" %in% IP){install.packages("gWidgetstcltk")}
> if (!"raster" %in% IP){install.packages("raster")}
> # install package
> install.packages("euptf_1.4.tar.gz", repos=NULL, type="source")
> require(euptf)
> help(euptf)
```

The last line opens the package help in the default internet browser.

2 Getting started: data preparation

Before using the PTFs you need to format your data so that the package's functions can access them. An example data frame is provided:

```
> data(ptf.data)
> str(ptf.data)

'data.frame':        60 obs. of  10 variables:
 $ SAMPLE_ID: int  1 2 3 4 5 6 7 8 9 10 ...
 $ TOPSOIL  : Factor w/ 2 levels "sub","top": 1 1 1 1 2 1 2 2 1 2 ...
 $ USSAND   : num  88.4 98.6 84.4 77.6 77.4 ...
 $ USSILT   : num  3.6 1.2 14.1 20.4 16.9 ...
 $ USCLAY   : num  8 0.2 1.5 2 5.7 8.5 2.5 14.1 15 4 ...
 $ OC       : num  0.11 0.06 0.34 0.4 0.86 2.24 0.29 0.5 0.33 3.93 ...
```

```
 $ BD       : num  1.68 1.49 1.55 1.3 1.66 1.7 1.63 1.58 1.7 1.13 ...
 $ PH_H2O   : num  10.5 5.8 6.6 5.6 7.2 ...
 $ CEC      : num  29.4 1.1 2.67 5.95 6.18 ...
 $ CACO3    : num  15 NA NA NA 0 0 0.1 1.4 0 NA ...
```

When creating a new data frame, the available variables should be named as in the example. If one of the variables has no observations, it can be omitted from the data frame. Missing values should be assigned value `NA`. Besides, if the class PTFs are to be used, the data frame should have additional columns `TEXT_FAO_MOD` and/or `TEXT_US`. If the particle size distribution (sand (0.5-2 mm), silt (0.002-0.5 mm), clay (<0.002 mm)) is available, they can be obtained using the following commands:

```
> # set FAO texture class
> ptf.data$TEXT_FAO_MOD <- psd2classFAO_MOD(ptf.data$USSAND, ptf.data$USSILT, ptf.data$USCLAY,
+     ptf.data$OC, option=TRUE)
> # set USDA texture class
> ptf.data$TEXT_US <- psd2classUS(ptf.data$USSAND, ptf.data$USSILT, ptf.data$USCLAY,
+     ptf.data$OC, option=TRUE)
```

It is important to set `option = TRUE` otherwise the PTFs will fail and produce only `NA` values.

For non R users, the data can also be prepared in an other software and imported into R. In that case, the names of the columns have to be respected in the source file. For example, data prepared in an Excel workbook named myworkbook.xlsx could be imported either by first saving the relevant Excel worksheet in comma separated table, for instance myworksheet.csv or by importing them directly:

```
> ## import data from csv file:
> mydata <- read.csv("myworksheet.csv")
> ## or with package xlsx:
> # install the package if not yet done:
> if (!"xlsx" %in% IP){install.packages("xlsx")}
> # load the package:
> require("xlsx")
> # import data from xlsx file:
> mydata <- read.xlsx("myworkbook.xlsx", sheetName="myworksheet")
> # see ?read.xlsx for more options
```

3 Choosing the right PTF

Typing `ChoosePTF()` in the R console will open a window helping to choose the right PTF based on the available input data and the desired output. Checking the available variables and clicking on the "What PTF?" button will update the PTF associated to each output.

4 Running the PTFs

Once the data are ready and the PTF has been chosen, it can be called with function `predict.ptf`. The output of the function varies from one PTF to another, depending if it is a point or a parameter PTF.

Let's take three examples that can illustrate the different cases and run them on the example dataset `ptf.data`.

Example 1 We want to predict the water content at field capacity, based on Topsoil/Subsoil distinction, sand, silt and clay contents, organic carbon content and bulk density. Using `ChoosePTF`, we see that we should use `PTF09`.

```
> ?PTF09
```

The help page of PTF09 tells us it is a linear regression based PTF predicting the water content at field capacity (FC) from PSD and OC.

```
> ex1 <- predict.ptf(newdata = ptf.data, ptf = "PTF09")
> str(ex1)

 Named num [1:60] 0.1262 0.0715 0.1476 0.1708 0.2062 ...
 - attr(*, "names")= chr [1:60] "1" "2" "3" "4" ...
```

The output is a vector of 60 elements giving the predicted water content at field capacity (330 cm of suction head) for each row in `ptf.data`.

Example 2 If we want to predict the water content at another matric potential, for instance $h = -100$ cm, we need to predict it with MRC PTF. With the same input data, `ChoosePTF` tells us that we have to use `PTF21`.

```
> ?PTF21
```

The help page of PTF21 tells us it is a class PTF giving Mualem-van Genuchten parameters.

```
> ex2 <- predict.ptf(newdata = ptf.data, ptf = "PTF21")
> str(ex2)

 num [1:60, 1:4] 0.0407 0.0407 0.0407 0.0407 0.0407 ...
 - attr(*, "dimnames")=List of 2
  ..$ : NULL
  ..$ : chr [1:4] "thr" "ths" "alp" "n"
```

The output is now a 60x4 matrix. It is not what we want. `PTF21` gives us the 4 parameters of the van Genuchten model (assuming $m = 1 - 1/n$), but we would like it to be evaluated at a suction head of 100 cm. So we add this request to the call of `predict.ptf`.

```
> ex2 <- predict.ptf(newdata = ptf.data, ptf = "PTF21", h = 100)
> str(ex2)

 num [1, 1:4, 1:60] 100 0.19 0.469 NA 100 ...
 - attr(*, "dimnames")=List of 3
  ..$ : NULL
  ..$ : chr [1:4] "h" "theta" "Se" "K"
  ..$ : NULL
```

But now we have a 1x4x60 array. What does it mean? Looking at the attributes of this array, we see that the names of the 4 elements in the second dimension are `"h" "theta" "Se" "K"`. We are interested in the water content values `theta` for our 60 observations.

```
> ex2 <- ex2[,2,]
> str(ex2)

 num [1:60] 0.19 0.146 0.194 0.203 0.207 ...
```

Example 3 If we are interested in the parameters of both the moisture retention and the hydraulic conductivity curves, `ChoosePTF` tells us that we have to use `PTF19`.

```
> ?PTF19
```

The help page of PTF19 says it is a class PTF giving Mualem-van Genuchten parameters for combinations of USDA texture classes and topsoil/subsoil distinction.
As a consequence, we have to make sure the USDA texture class is in the input data before running the PTF.

```
> ptf.data$TEXT_US <- psd2classUS(ptf.data$USSAND, ptf.data$USSILT, ptf.data$USCLAY,
+     ptf.data$OC, option=TRUE)
> ex3 <- predict.ptf(newdata = ptf.data, ptf = "PTF19")
> str(ex3)

 num [1:60, 1:7] 0.0373 0.0342 0.0373 0.0373 0.052 ...
 - attr(*, "dimnames")=List of 2
  ..$ : chr [1:60] "15" "22" "16" "17" ...
  ..$ : chr [1:7] "thr" "ths" "alp" "n" ...
```

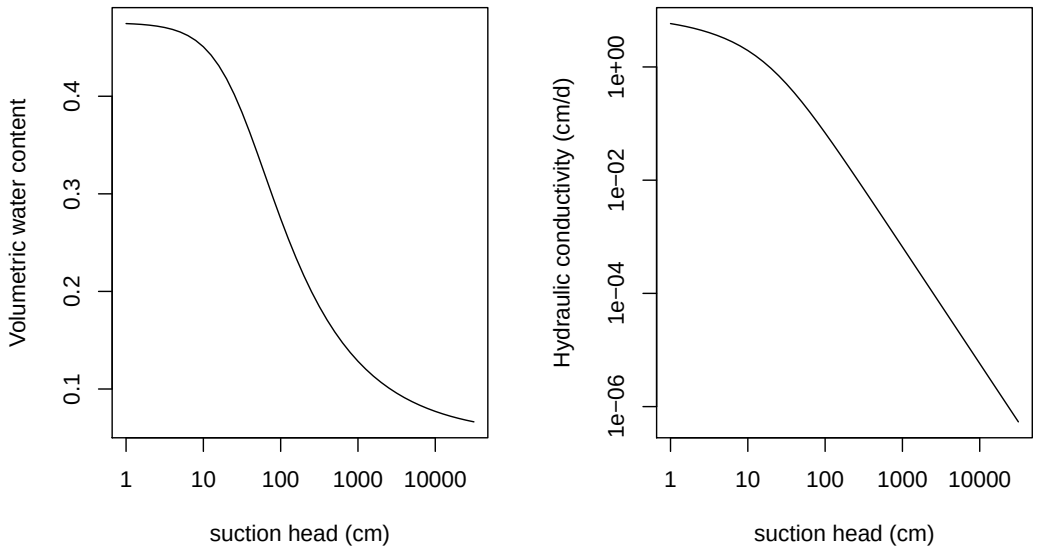
The output is a 60x7 matrix, the 7 parameters of the Mualem-van Genuchten model for each of the 60 observations in `data.ptf`. As in example 2, we can evaluate the model at given matric potentials.

```
> ex3 <- predict.ptf(newdata = ptf.data, ptf = "PTF19", h = 10^(seq(0,4.5,by = 0.1)))
> str(ex3)

 num [1:46, 1:4, 1:60] 1 1.26 1.58 2 2.51 ...
 - attr(*, "dimnames")=List of 3
  ..$ : NULL
  ..$ : chr [1:4] "h" "theta" "Se" "K"
  ..$ : NULL
```

The retention and conductivity curves obtained can be plotted. For example, for the fifth observation in `ptf.data`, we can do:

```
> # divide the figure into 2 plotting areas
> par(mfrow=c(1,2))
> # select calculated values for the fifth sample
> i <- 5
> # plot h vs theta in the first plotting area
> plot(ex3[,1,i], ex3[,2,i], type = "l", log = "x", xlab = "suction head (cm)",
+     ylab = "Volumetric water content")
> # plot h vs K in the second plotting area
> plot(ex3[,1,i], ex3[,4,i], type = "l", log = "xy", xlab = "suction head (cm)",
+     ylab = "Hydraulic conductivity (cm/d)")
```



In conclusion, if `n` is the number of observations in the input data, `predict.ptf` returns:

- a `nx1` vector, for point PTFs
- a `nx4` or `nx7` matrix for parameters PTFs
- a `mx4xn` array for parameters PTFs that were given `m` suction head values at which to be evaluated.

5 Exporting the results

For non-R users, the results can be exported for use with other software, either as text files or directly in Excel. For example, the two ways of predicting the field capacity can be exported as the columns of a new table.

```
> FC <- data.frame(FC1 = ex1, FC2 = predict.ptf(newdata = ptf.data, ptf = "PTF21", h = 330)[,2,])
> str(FC)
> ## example to save the predicted values to a csv file:
> write.csv(FC, file = "myFC.csv")
> # see ?write.csv for additional options
> ## example to save predicted values to xlsx file:
> require("xlsx")
> write.xlsx(x = FC, file = "myoutput.xlsx", sheetName = "myFC", row.names = FALSE)
```

6 Running the PTFs on spatial data

Function `predict.ptf.raster` makes it possible to run the PTFs on a `RasterStack` object, created with R package `raster`. The layers of the RasterStack must be named with valid names, like the column names in the example dataset `data(ptf.data)`. The output of the PTF is saved as a `RasterLayer` or a `RasterBrick` and written in a raster file.

References

Tóth, B., Weynants, M., Nemes, A., Makó, A., Bilas, G., Tóth, G., 2014. New generation of hydraulic pedotransfer functions for Europe. European Journal Of Soil Science in Press.

Weynants, M., Tóth, G., Montanarella, L., *al.*, *et.*, 2013. European Hydropedological Inventory (EU-HYDI). Publications Office of the European Union, Luxembourg.