

Additional functions for transforming soil particlesize distributions

Wei Shangguan

September 13, 2010

1 Load the soiltexture package

The soiltexture package can be installed from CRAN with the following commands:

```
install.packages("soiltexture")
```

And loaded with the following commands:

```
require("soiltexture")
require("drc")
```

'drc' has been loaded.

Please cite R and 'drc' if used for a publication,
for references type 'citation()' and 'citation('drc')'.

2 Transforming soil texture data using many Particle- Size Distribution models (from 3 or more particle size classes)

`TT.text.transf.Xm()` is used to transform soil texture data from 3 or more particle size classes using various Particle-Size Distribution (PSD) models. The `drc` package and its associate packages(`lattice`,`magic`,`nlme`, `plotrix`) are required in the PSD model fitting. Compared to `TT.text.transf()`, the following check is not needed (and not done) :

- When the 1st value of input `tri.data` and output particle size classes limits is 0, The 2nd value of the output particle size classes limits must be higher or equal to the 2nd value of the input particle size classes limits."

We need first to create a dummy dataset with more than 3 particle size classes:

```
my.text4 <- data.frame(
  "CLAY" = c(05,60,15,05,25,05,25,45,65,75,13,47),
  "FSILT" = c(02,04,10,15,25,40,35,20,10,05,10,20),
```

```

"CSILT" = c(03,04,05,10,30,45,30,25,05,10,07,23),
"SAND"  = c(90,32,70,70,20,10,10,10,20,10,70,10)
) #

```

Transform this data frame from 4 particle size classes to 3 particle size classes:

```

res <- TT.text.transf.Xm(
  tri.data      = my.text4,
  base.ps.lim   = c(0,1,50,2000),
  dat.ps.lim    = c(0,2,30,60,2000),
  psdmodel      = "AD"
) #
#
round( res[,1:6], 3 )

      0-1    1-50 50-2000 f0:(Intercept) b:(Intercept)
[1,]  4.341  4.651  91.007          0.584          0.364
[2,] 59.657  6.931  33.412          0.807          0.148
[3,] 13.657 14.860  71.483          0.763          0.477
[4,]  3.408 23.472  73.119          0.571          0.412
[5,] 24.116 49.480  26.403          0.619          0.265
[6,]  4.472 81.324  14.283          0.521          0.318
[7,] 24.365 62.043  13.594          0.620          0.255
[8,] 44.507 41.646  13.848          0.721          0.189
[9,] 63.849 14.739  21.412          0.833          0.171
[10,] 74.779 11.981  13.240          0.874          0.087
[11,] 11.934 15.826  72.239          0.611          0.361
[12,] 46.495 39.835  13.677          0.731          0.183
      c:(Intercept)
[1,]          4.276
[2,]          3.211
[3,]          1.314
[4,]          1.630
[5,]          4.298
[6,]          9.084
[7,]          6.745
[8,]          5.920
[9,]          1.102
[10,]          4.801
[11,]          1.989
[12,]          5.533

#
round( res[,7:ncol(res)], 3 )

      r0:(Intercept)    dev
[1,]          0.613 0.783
[2,]          0.138 0.000
[3,]          0.773 0.003
[4,]          0.179 0.000
[5,]          0.039 0.000

```

[6,]	0.032	0.018
[7,]	0.031	0.000
[8,]	0.035	0.000
[9,]	0.090	0.000
[10,]	0.052	0.000
[11,]	0.231	0.000
[12,]	0.035	0.000

The first 3 columns are the predicted values with a sum not equal to 100% (can be normalised by `TT.normalise.sum.X()`). The following 4 columns are the fitted PSD model parameters. And the last column is the Residual Sum of Squares (deviance). Note that the transforming results may be slightly different even with the same function parameters. This is caused by the nature of `drc` package in fitting dose-response models.

Sometimes, the fitting will fail for the iteration is not converged or some errors and warnings happened. These can be ignored, as you can get the transforming results.

The following PSD models are implemented: Anderson (AD), Fredlund4P (F4P), Fredlund3P (F3P), modified logistic growth (ML), Offset-Nonrenormalized Lognormal (ONL), Offset-Renormalized Lognormal (ORL), Skaggs (S), van Genuchten type(VG), van Genuchten modified, Weibull (W), Logarithm(L), Logistic growth (LG), Simple Lognormal (SL), Shiozawa and Campbell (SC). The performance of PSD models is influenced by many aspects like soil texture class, number and position (or closeness) of observation points, clay content etc. The latter four PSD models perform worse than the former ten. The AD, F4P, S, and W model is recommended for most of texture classes. And it will be even better to compare several different PSD models and using the results of the model with the minimum residual sum of squares. Except S and W models, all the PSD models could be used to predict the content below the minimum input limit. The "psdmodel" option could be changed to any other of the above models:

```
res <- TT.text.transf.Xm(
  tri.data      = my.text4,
  base.ps.lim   = c(0,1,50,2000),
  dat.ps.lim    = c(0,2,30,60,2000),
  psdmodel      = "ML"
) #
#
round( res[,1:6], 3 )
```

	0-1	1-50	50-2000	a:(Intercept)	b:(Intercept)
[1,]	4.941	3.947	91.113	19.476	13.341
[2,]	59.849	6.848	33.302	0.675	5.740
[3,]	14.721	13.805	70.984	6.473	4.910
[4,]	4.413	22.512	72.511	53.869	7.420
[5,]	24.466	46.833	28.700	3.162	62.763
[6,]	4.376	76.265	19.359	26.148	57.096

```

[7,] 24.185 58.851 16.964      3.259      72.585
[8,] 44.788 38.541 16.671      1.238     180.302
[9,] 64.027 14.560 21.321      0.615       4.399
[10,] 74.978 11.682 13.340      0.334     249.075
[11,] 12.405 15.396 71.925      8.171       5.553
[12,] 46.747 37.139 16.114      1.146     144.538

```

```

c:(Intercept)
[1,]      1.013
[2,]      0.983
[3,]      0.549
[4,]      0.304
[5,]      1.140
[6,]      0.834
[7,]      1.090
[8,]      1.534
[9,]      0.563
[10,]     1.927
[11,]     0.527
[12,]     1.467

```

```
#
```

```
round( res[,7:ncol(res)], 3 )
```

```

[1] 0.000 0.000 1.669 1.071 0.000 0.000 0.000 0.000 0.011 0.000
[11] 0.205 0.000

```

Because the current PSD model fitting is quite time-consuming and some models are not always successful for all soils, you can change the PSD model, or optimization method potentially at the cost of some accuracy. The default "omethod" option (i.e. "all") is to run all methods and choose the best results with minimum residual sum of squares. The optional methods are "Nelder-Mead", "BFGS", "CG", "L-BFGS-B", "SANN" (see `optim()` for details.)

```

res <- TT.text.transf.Xm(
  tri.data = my.text4,
  base.ps.lim = c(0,1,50,2000),
  dat.ps.lim = c(0,2,30,60,2000),
  psdmodel = "ML",
  omethod = "SANN"
)
#
#

```

```
round( res[,1:6], 3 )
```

```

      0-1    1-50 50-2000 a:(Intercept) b:(Intercept)
[1,]  4.731  4.231 91.027      20.873       7.161
[2,] 59.849  6.849 33.302       0.675      5.740
[3,] 14.721 13.805 70.985       6.473      4.911
[4,]  4.415 22.509 72.513     53.781      7.420
[5,] 24.466 46.834 28.700       3.162     62.764
[6,]  4.376 76.264 19.360     26.146     57.091
[7,] 24.186 58.850 16.964       3.259     72.608

```

```

[8,] 44.788 38.541 16.671      1.238      180.303
[9,] 64.026 14.561 21.321      0.615        4.396
[10,] 74.978 11.682 13.340      0.334      249.151
[11,] 12.410 15.390 71.926      8.166        5.552
[12,] 46.745 37.142 16.113      1.146      144.522

```

```

c:(Intercept)

```

```

[1,]      0.767
[2,]      0.983
[3,]      0.549
[4,]      0.304
[5,]      1.140
[6,]      0.834
[7,]      1.090
[8,]      1.534
[9,]      0.563
[10,]     1.927
[11,]     0.527
[12,]     1.467

```

```

#

```

```

round( res[,7:ncol(res)], 3 )

```

```

[1] 0.091 0.000 1.669 1.071 0.000 0.000 0.000 0.000 0.011 0.000
[11] 0.205 0.000

```

3 Normalizing soil texture data (sum of X texture classes)

TT.normalise.sum.X() is similar to TT.normalise.sum(). But it normalize the sum of the X (X>1) texture classes instead of 3. The option tri.data should be a data.frame with only soil texture data (no additional extra columns should be present).

```

my.text5 <- data.frame(
  "CLAY" = c(05,60,15,04.9,25,05,25,45,65,75,13,47),
  "FSILT" = c(02,04.3,10,15,25,40,35,20,10,05,10,20),
  "CSILT" = c(03,04,05,10,30,45,30,25,05,10,07.2,23.3),
  "SAND" = c(90.5,32,70,70,20.3,10.9,9.3,9.4,20,10,70,10)
) #
#
res <- TT.normalise.sum.X(
  tri.data = my.text5,
  residuals = TRUE
) #

[1] 100.5 100.3 100.0 99.9 100.3 100.9 99.3 99.4 100.0 100.0
[11] 100.2 100.3

#
res

```

	CLAY	FSILT	CSILT	SAND	residuals
[1,]	4.975124	1.990050	2.985075	90.049751	0.5
[2,]	59.820538	4.287139	3.988036	31.904287	0.3
[3,]	15.000000	10.000000	5.000000	70.000000	0.0
[4,]	4.904905	15.015015	10.010010	70.070070	-0.1
[5,]	24.925224	24.925224	29.910269	20.239282	0.3
[6,]	4.955401	39.643211	44.598612	10.802775	0.9
[7,]	25.176234	35.246727	30.211480	9.365559	-0.7
[8,]	45.271630	20.120724	25.150905	9.456740	-0.6
[9,]	65.000000	10.000000	5.000000	20.000000	0.0
[10,]	75.000000	5.000000	10.000000	10.000000	0.0
[11,]	12.974052	9.980040	7.185629	69.860279	0.2
[12,]	46.859422	19.940179	23.230309	9.970090	0.3