Decomposition Functions

List of functions

Here is a list of functions that are commonly used in biogeochemical models to modify decomposition rates. This list was extracted from Table 2 of Sierra et al. (2015, JAMES 7:335-356, http://onlinelibrary.wiley.com/doi/10.1002/2014MS000358/abstract).

$f(X)$	Model or author name	SoilR name
$f(T) =$ $\left(\frac{T_{\max}-T}{T_{\max}-T_{\text{opt}}}\right)^{0.2} \exp\left(\frac{0.2}{2.63}\left(1-\left(\frac{T_{\max}-T}{T_{\max}-T_{\text{opt}}}\right)^{2.63}\right)\right)$	Century	fT.Century1
$3.439 \exp\left(\frac{0.2}{2.63}\left(1-\left(\frac{T_{\max}-T}{T_{\max}-T_{\text{opt}}}\right)^{2.63}\right)\left(\frac{T_{\max}-T}{T_{\max}-T_{\text{opt}}}\right)^{0.2}\right)$	Century	fT.Century2
$0.8 \exp(0.095 T_s)$	Daycent	fT.Daycent1
$0.56 + (1.46 \arctan(\pi 0.0309(T_s - 15.7)))/\pi$	Daycent	fT.Daycent2
$Q_{10}^{(T-10)/10}$	Q10	Q2, Q1.4
$\exp\left(308.56\left(\frac{1}{56.02}-\frac{1}{(T+273)-227.13}\right)\right)$	Lloyd and Taylor	fT.LandT
$\exp(-3.764 + 0.204T(1 - 0.5T/36.9))$	Kirschbaum	fT.KB
$\exp((\ln(Q_{10})/10)(T-20))$	Demeter	fT.Demeter
$\exp(-(T/(T_{\text{opt}}+T_{\text{lag}}))^{T_{\text{shape}}}) Q_{10}^{(T-10)/10}$	Standcarb	fT.Standcarb
———-	——-	—
$f(W) =$ $\frac{1}{1+30\exp(-8.5W_i)}$	Century	fW.Century
$\left(\frac{W_i-b}{a-b}\right)^{d((b-a)/(a-c))}\left(\frac{W_i-c}{a-c}\right)^{d}$	Daycent	fW.Daycent1
$0.25 + 0.75(W_i)$	Demeter	fW.Demeter
$(1-\exp(-(3/W_{\min})(W_i+a)))^{b}\ \exp(-(W_i/(M_{\max}+c))^{d})$	Standcarb	fW.Standacarb
$4W_i(1-W_i)$ if $W_i \leq 0.5$; 1 if $W_i > 0.5$	Candy	fW.Candy
$\exp(-\exp(a - b\ W_i))$	Gompertz	fW.Gompertz
$bW_i + (1-b)W_i^2$	Myers	NA
$aW_i - bW_i^2$	Moyano	fW.Moyano
$\min[\alpha W_i^f,\ \beta(1-W_i)^g]$	Skopp	fW.Skopp

Generally, the functions are combined to produce a decomposition modifier $\xi(t)$ in models of the form

$$\frac{d\mathbf{C}(t)}{dt} = \mathbf{I}(t) + \xi(t) \cdot \mathbf{A} \cdot \mathbf{C}(t),$$

with

$$\xi(t) = \prod_i f(X_i(t)),$$

where $f(X_i)$ are functions that depend on environmental variables X_i such as temperature and moisture.

Functions in SoilR

Temperature functions

The list of temperature functions above are implemented in the SoilR package and can be used for comparing their behavior.

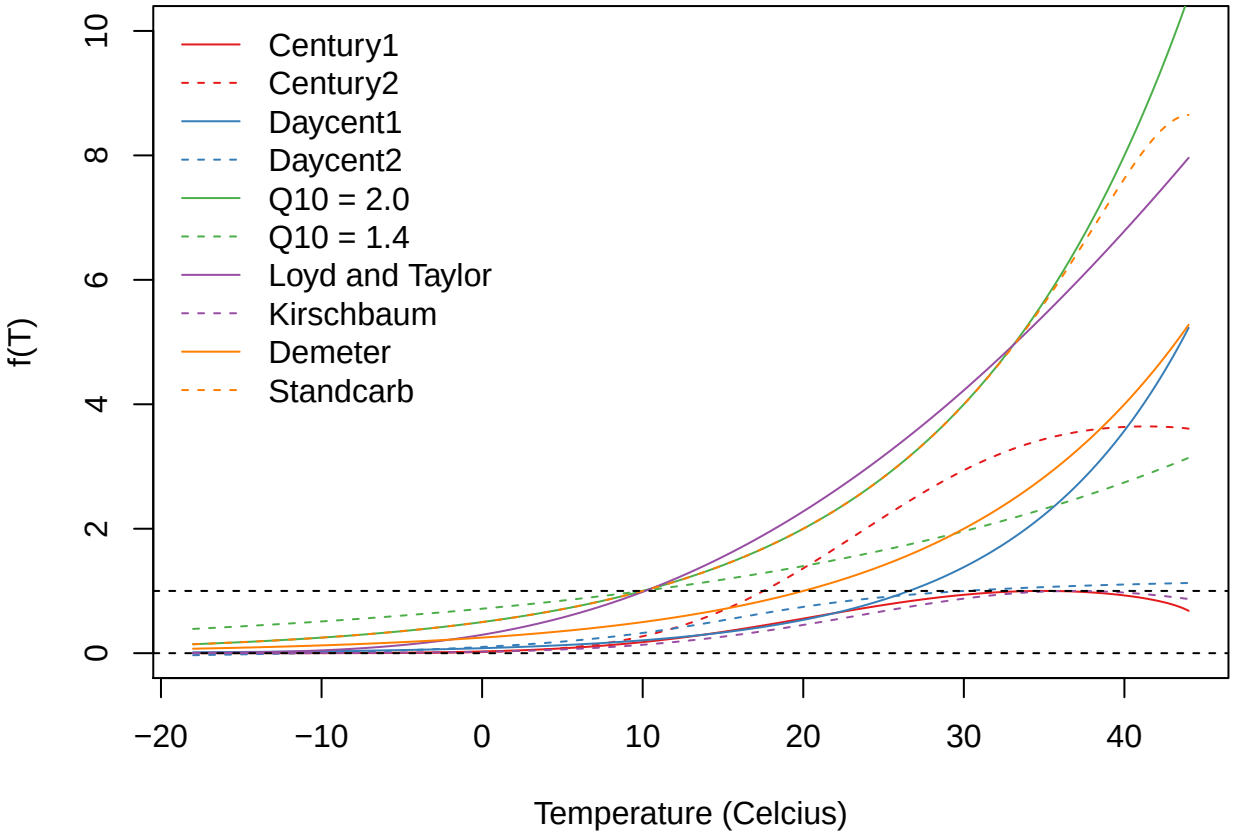
```
library(SoilR)
```

```
## Loading required package: deSolve
##
## Attaching package: 'deSolve'
##
## The following object is masked from 'package:graphics':
##
##     matplot
##
## Loading required package: parallel
## Loading required package: RUnit
```

```
Temp=seq(-18,44,0.1) #Temperature range

fT1=fT.Century1(Temp)
fT2=fT.Century2(Temp)
fT3=fT.Daycent1(Temp)
fT4=fT.Daycent2(Temp)
fT5=fT.Q10(Temp)
fT6=fT.Q10(Temp,Q10=1.4)
fT7=fT.LandT(Temp)
fT8=fT.KB(Temp)
fT9=fT.Demeter(Temp)
fT10=fT.Standcarb(Temp)
```

To plot them

```
library(RColorBrewer) #Colors for plotting
pal=brewer.pal(5,"Set1") #Color palette
par(mar=c(4,4,1,0.5))
plot(Temp,fT1,type="l",xlab="Temperature (Celcius)", ylab="f(T)",ylim=c(0,10),col=pal[1])
lines(Temp,fT2,col=pal[1],lty=2)
lines(Temp,fT3,col=pal[2])
lines(Temp,fT4,col=pal[2],lty=2)
lines(Temp,fT5,col=pal[3])
lines(Temp,fT6,col=pal[3],lty=2)
lines(Temp,fT7,col=pal[4])
lines(Temp,fT8,col=pal[4],lty=2)
lines(Temp,fT9,col=pal[5])
lines(Temp,fT10,col=pal[5],lty=2)
abline(h=1,lty=2)
abline(h=0,lty=2)
legend("topleft",c("Century1","Century2","Daycent1","Daycent2","Q10 = 2.0","Q10 = 1.4","Loyd and Taylor
```



Moisture functions

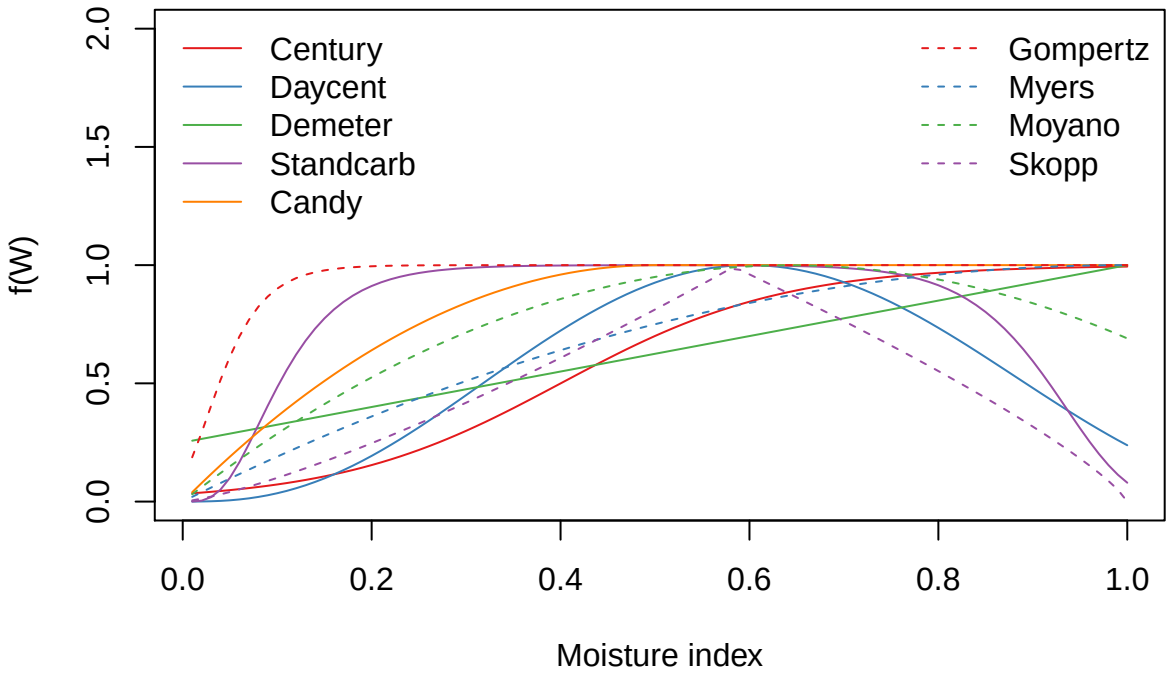
The moisture dependend functions are more tricky to compare because the original authors express moisture using different metrics. Some use soil water content, others the aridity index, and others soil water potential. To deal with this, we can take the range of these metrics and expressed in a range from 0 to 1. In the JAMES paper we called this metric the moisture index W_i. Some moisture functions implemented in SoilR follow the original convention reported in the original papers, so we need to re-implement them here following the W_i convention:

```
# Moisture Functions
W1=seq(0.01,1,0.01)

#Standarize some functions to 'Moisture Index'.
fW.Century1=function (MoistIndex)
{
  1/(1 + 30 * exp(-8.5 * (MoistIndex)))
}

#Need to modify the Daycent function so it doesn't calculate water filled pored space. It is replaced b
fW.Daycent1.1=function (MoistIndex, a = 0.6, b = 1.27, c = 0.0012, d = 2.84)
{
  wfps = MoistIndex
  wfunc = (((wfps - b)/(a - b))^(d * ((b - a)/(a - c))))*((wfps - c)/(a - c))^d
  return(wfunc)
}
```

```
fW.Standcarb1=function (Moist, MatricShape = 5, MatricLag = 0, MoistMin = 30,
                        MoistMax = 350, DiffuseShape = 15, DiffuseLag = 4)
{
  IncreaseRate = 3/MoistMin
  MatricLimit = (1 - exp(-IncreaseRate * (Moist + MatricLag)))^MatricShape
  DiffuseLimit = exp(-1*(Moist/(MoistMax + DiffuseLag))^DiffuseShape)
  MoistDecayIndex = MatricLimit * DiffuseLimit
  return(MoistDecayIndex)
}

fW.Candy=function(MoistIndex){
  Mi=MoistIndex # Moisture index. Original equation arrives here dividing VWC by pore volume
  fw=ifelse(Mi<=0.5, 4*Mi*(1-Mi),1)
  return(fw)
}

fW.Myers=function(MoistIndex,b=2){
  b*MoistIndex+((1-b)*MoistIndex^2)
}

#Run the functions
fW1=fW.Century1(W1)
fW2=fW.Daycent1.1(W1)
fW3=fW.Demeter(W1,Msat=1)
fW4=fW.Standcarb1(W1*100,MoistMin=15,MoistMax=90)
fW5=fW.Candy(W1)
fW6=fW.Gompertz(W1)
fW7=fW.Myers(W1)
fW8=fW.Moyano(W1)
fW9=fW.Skopp(W1)
```

And now we can plot them

```
plot(W1,fW1,type="l",xlab="Moisture index",ylab="f(W)",ylim=c(0,2),col=pal[1])
lines(W1,fW2,col=pal[2])
lines(W1,fW3,col=pal[3])
lines(W1,fW4,col=pal[4])
lines(W1,fW5,col=pal[5])
lines(W1,fW6,col=pal[1],lty=2)
lines(W1,fW7,col=pal[2],lty=2)
lines(W1,fW8,col=pal[3],lty=2)
lines(W1,fW9,col=pal[4],lty=2)
legend("topleft",c("Century","Daycent","Demeter","Standcarb","Candy"),lty=rep(1,5),col=pal[1:5],bty="n")
legend("topright",c("Gompertz","Myers","Moyano","Skopp"),lty=rep(2,4),col=pal[1:4],bty="n")
```



Decomposition potential maps

The original JAMES paper included a couple of global maps combining the predictions of these functions using a global climate dataset. Unfortunately, the reviewers said the paper was too long and we had to take this part out of the paper, but maybe we can use these maps here for this analysis.

First, we load required packages and set the working directory to be able to read the climate datasets stored in the repository

```
library(raster)
```

```
## Loading required package: sp
```

```
library(rasterVis)
```

```
## Loading required package: lattice
## Loading required package: latticeExtra
```

```
setwd("~/soil-metamodel/Litter-decomp-mapping/DecompositionFunctions/")

#Read the NetCDF data of annual means of Temperature, Precipitation and PET
PETb=brick("Data/PET_PT.WATCH.MonthlyMean.1980.2001.nc")
```

```
## Loading required namespace: ncdf
```

```
Pb=brick("Data/Precip.WATCH.MonthlyMean.1950.2001.nc")
Tbair=brick("Data/Tair.WATCH.MonthlyMean.1950.2001.nc")
```

Now we calculate annual means for air temperature, annual precipitation, and annual potential evapotranspiration

```
PETm=calc(PETb,mean)
Pm=calc(Pb,mean)
Tm=calc(Tbair,mean)-273 #Change from Kelvin to Celcius
```

Calculate the moisture index as the ratio between precipitation and potential evapotranspiration$W_i = PPT/PET$

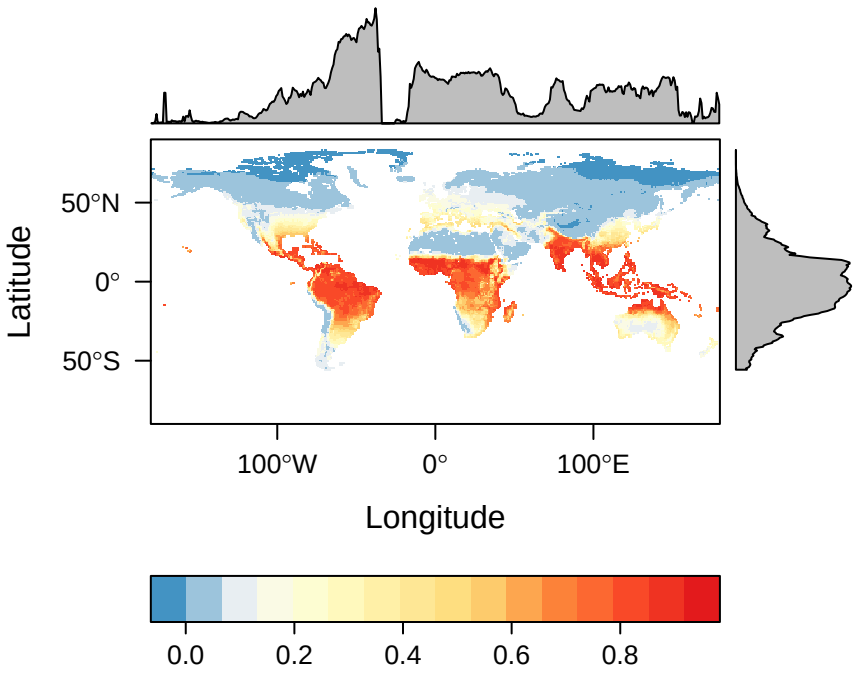
```
Wi=Pm/PETm
Wi[Wi>1]=1 # When PPT>PET, we assume saturation (Wi=1)
```

We can calculate now the value of ξ using the functions fT.Century1 and fW.Century

```
xi.Century=fT.Century1(Tm)*fW.Century1(Wi)
```

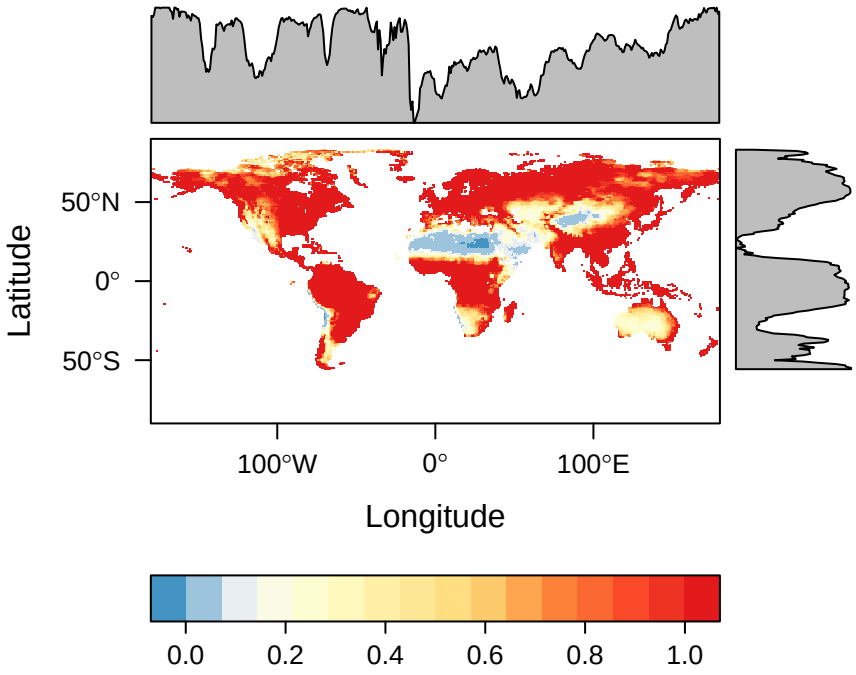
A global map can be plotted as

```
pald2=c(rev(brewer.pal(9, 'RdBu')[c(5,8)]),brewer.pal(9,'YlOrRd')[c(1,2,3,5,6,7)])
levelplot(xi.Century,ylab="Latitude", xlab="Longitude",
          par.settings=RdBuTheme(region=pald2))
```



We can calculate this decomposition index for any combination of the temperature and moisture functions above. For example, we can combine the Lloyd and Taylor function for the temperature dependence and the Moyano function for the moisture dependance and obtain the following map:

```
xi.LandTMoyano=fT.LandT(Tm)*fW.Moyano(Wi)
levelplot(Wi,ylab="Latitude", xlab="Longitude",
          par.settings=RdBuTheme(region=pald2))
```


50°N
0°
50°S
Latitude
100°W
0°
100°E
Longitude
0.0
0.2
0.4
0.6
0.8
1.0