

Turnover model – Millennial-age GDGTs in forested mineral soils

Model

Author(s):

Gies, Hannah

Publication date:

2020

Permanent link:

<https://doi.org/10.3929/ethz-b-000430429>

Rights / license:

[In Copyright - Non-Commercial Use Permitted](#)

Soil model

August 6, 2020

1 Soil Turnover Model

The following script estimates the turnover time of an organic compound based on its radiocarbon signature at a single point in time in different samples based on a single pool model and a two-pool model.

Define the variables here:

```
In [66]: #sampling year of the soil compound
sampling_year = 2014

#sample name
sample_name = ['Bb 0-5', 'Bb 10-20', 'Bb 20-40', 'Ln 0-5', 'Ln 10-20', 'Ln 20-40']

#the compound's name and its fraction modern (Fm) for each sample
compounds = {
    'bulk': [1.007, 0.886, 0.834, 1.117, 1.015, 0.8],
    'isoGDGT': [0.964, 0.869, 0.764, 1.009, 0.821, 0.705],
    'brGDGT': [0.985, 0.851, 0.813, 0.980, 0.868, 0.595],
    'SCFA': [0.998, 0.915, 0.934, 1.172, 1.023, 1.008],
    'C26_FA': [0.922, 0.884, 0.768, 1.164, 0.988, 0.900],
    'C28_FA': [0.935, 0.865, 0.731, 1.181, 1.021, 0.732],
    'Alkane': [0.995, 0.862, 0.732, 1.141, 0.971, 0.517]
}

#the 2-pool-model requires an estimation of the turnover time of the fast-
#fast turnover fraction based on the proportion of the light fraction in t
fast_fraction = [0.897, 0.193, 0.115, 0.162, 0.113, 0.087]

#fast turnover based on light fraction turnover as single fast pool (van o
#fast_turnover = [354, 886, 349, 46, 236, 1181]

#fast turnover based on SCFA single pool turnover
fast_turnover = [344, 921, 761, 33, 242, 299]

#fast turnover based on topsoil SCFA single pool turnover at all depths
#fast_turnover = [344, 344, 344, 33, 33, 33]
```

```

#the output of the two-pool model is saved as two_pool_turnover_{file_info}
#to differentiate between tables based on different fast turnover times
file_info = 'light_fraction_turnover'

```

```

In [67]: import math
import numpy as np
import scipy as sp
from scipy import optimize
import matplotlib as mpl
import matplotlib.pyplot as plt
import pandas as pd
%matplotlib inline

```

atmospheric CO2 data from Hua et al. (2013) and Hammer & Levin (2017)

```

In [68]: #atmospheric CO2 Fm
#1950 - 1986 data from Hua et al. 2013
atmo_1950_1986 = [-26,-26,-26,-24,-23,17,24,98,168,280,228,219,391,827,899]
#1987 - 2016 data from Hammer & Levin 2017
atmo_1987_2016 = [183,169,158,149,138,134,126,120,112,104,100,98,96,94,92,90,88,86,84,82,80,78,76,74,72,70,68,66,64,62,60,58,56,54,52,50,48,46,44,42,40,38,36,34,32,30,28,26,24,22,20,18,16,14,12,10,8,6,4,2,0,-2,-4,-6,-8,-10,-12,-14,-16,-18,-20,-22,-24,-26,-28,-30,-32,-34,-36,-38,-40,-42,-44,-46,-48,-50,-52,-54,-56,-58,-60,-62,-64,-66,-68,-70,-72,-74,-76,-78,-80,-82,-84,-86,-88,-90,-92,-94,-96,-98,-100,-102,-104,-106,-108,-110,-112,-114,-116,-118,-120,-122,-124,-126,-128,-130,-132,-134,-136,-138,-140,-142,-144,-146,-148,-150,-152,-154,-156,-158,-160,-162,-164,-166,-168,-170,-172,-174,-176,-178,-180,-182,-184,-186,-188,-190,-192,-194,-196,-198,-200,-202,-204,-206,-208,-210,-212,-214,-216,-218,-220,-222,-224,-226,-228,-230,-232,-234,-236,-238,-240,-242,-244,-246,-248,-250,-252,-254,-256,-258,-260,-262,-264,-266,-268,-270,-272,-274,-276,-278,-280,-282,-284,-286,-288,-290,-292,-294,-296,-298,-300,-302,-304,-306,-308,-310,-312,-314,-316,-318,-320,-322,-324,-326,-328,-330,-332,-334,-336,-338,-340,-342,-344,-346,-348,-350,-352,-354,-356,-358,-360,-362,-364,-366,-368,-370,-372,-374,-376,-378,-380,-382,-384,-386,-388,-390,-392,-394,-396,-398,-400,-402,-404,-406,-408,-410,-412,-414,-416,-418,-420,-422,-424,-426,-428,-430,-432,-434,-436,-438,-440,-442,-444,-446,-448,-450,-452,-454,-456,-458,-460,-462,-464,-466,-468,-470,-472,-474,-476,-478,-480,-482,-484,-486,-488,-490,-492,-494,-496,-498,-500,-502,-504,-506,-508,-510,-512,-514,-516,-518,-520,-522,-524,-526,-528,-530,-532,-534,-536,-538,-540,-542,-544,-546,-548,-550,-552,-554,-556,-558,-560,-562,-564,-566,-568,-570,-572,-574,-576,-578,-580,-582,-584,-586,-588,-590,-592,-594,-596,-598,-600,-602,-604,-606,-608,-610,-612,-614,-616,-618,-620,-622,-624,-626,-628,-630,-632,-634,-636,-638,-640,-642,-644,-646,-648,-650,-652,-654,-656,-658,-660,-662,-664,-666,-668,-670,-672,-674,-676,-678,-680,-682,-684,-686,-688,-690,-692,-694,-696,-698,-700,-702,-704,-706,-708,-710,-712,-714,-716,-718,-720,-722,-724,-726,-728,-730,-732,-734,-736,-738,-740,-742,-744,-746,-748,-750,-752,-754,-756,-758,-760,-762,-764,-766,-768,-770,-772,-774,-776,-778,-780,-782,-784,-786,-788,-790,-792,-794,-796,-798,-800,-802,-804,-806,-808,-810,-812,-814,-816,-818,-820,-822,-824,-826,-828,-830,-832,-834,-836,-838,-840,-842,-844,-846,-848,-850,-852,-854,-856,-858,-860,-862,-864,-866,-868,-870,-872,-874,-876,-878,-880,-882,-884,-886,-888,-890,-892,-894,-896,-898,-900,-902,-904,-906,-908,-910,-912,-914,-916,-918,-920,-922,-924,-926,-928,-930,-932,-934,-936,-938,-940,-942,-944,-946,-948,-950,-952,-954,-956,-958,-960,-962,-964,-966,-968,-970,-972,-974,-976,-978,-980,-982,-984,-986,-988,-990,-992,-994,-996,-998,-1000]
atmo_1950_2016 = atmo_1950_1986 + atmo_1987_2016

#convert D14C to Fm
def D14C_Fm(D14C, year):
    return (D14C/1000+1)*math.exp((year-1950)/8267)

Fm_atmo_1950_2016 = list(map(lambda x: D14C_Fm(x[1], x[0] + 1950), enumerate(atmo_1950_2016)))

```

```

In [69]: #atmospheric CO2 Fm until sampling
Fm_atmo_model = Fm_atmo_1950_2016[:-(2016-sampling_year)]

```

Definition of Functions for single pool model

```

In [70]: #functions

#find intial soil Fm assuming constant atmospheric Fm = 1 before bomb test
def find_Fini(k):
    Fini = (k/(k + 0.000121))
    return Fini

#find soil Fm of following year
def F_thisyear(k, F_lastyear, F_atmo):
    F_thisyear = k*F_atmo + F_lastyear*(1-k-0.000121)
    return F_thisyear

#find Fraction modern at the end of a list of annual atmospheric values

```

```

def F_sample(k):
    #define F_ini
    F_ini = find_Fini(k)
    #initiate for-loop
    F_lastyear = F_ini
    #for-loop
    for F_atmo in Fm_atmo_model:
        F_sample_thisyear = F_thisyear(k, F_lastyear, F_atmo)
        F_lastyear = F_sample_thisyear
    return F_sample_thisyear

#find turnover time for Fm of sample
def get_turnovertime(Fm_true):
    #loss function
    def loss(k):
        F_modelled = F_sample(k)
        return math.sqrt((Fm_true - F_modelled)** 2)
    result = sp.optimize.least_squares(loss, 0.001)
    optimal_k = result.x[0]
    turnovertime = 1/optimal_k
    return turnovertime

```

Determine the single-pool modelled turnover time for each compound at each depth and save it as a .csv

```

In [71]: Fm_samples = pd.DataFrame(compounds, index=sample_name)
turnover_times = Fm_samples.applymap(get_turnovertime)
pd.set_option("display.precision", 0)
print(turnover_times)
turnover_times.to_csv('single_pool_turnover.csv')

```

	Alkane	C26_FA	C28_FA	SCFA	brGDGT	bulk	isoGDGT
Bb 0-5	359	860	753	344	411	304	539
Bb 10-20	1435	1210	1403	921	1553	1190	1362
Bb 20-40	3098	2576	3113	761	1992	1743	2631
Ln 0-5	36	33	33	33	439	66	295
Ln 10-20	494	395	249	242	1372	271	1895
Ln 60-80	7779	1057	3098	299	5685	2153	3527

Definition of functions for two-pool model

```

In [72]: #find Fraction modern at the end of a list of annual atmospheric values
def F_sample_2pool(k,k_fast,fraction):
    #define F_ini of slow and fast pool
    F_ini_slow = find_Fini(k)
    F_ini_fast = find_Fini(k_fast)
    #initiate for-loop
    Fslow_lastyear = F_ini_slow

```

```

Ffast_lastyear = F_ini_fast
#for-loop
for F_atmo in Fm_atmo_model:
    Fslow_thisyear = F_thisyear(k, Fslow_lastyear, F_atmo)
    Ffast_thisyear = F_thisyear(k_fast, Ffast_lastyear, F_atmo)
    F_sample_thisyear = fraction*Ffast_thisyear+(1-fraction)*Fslow_thisyear
    Fslow_lastyear = Fslow_thisyear
    Ffast_lastyear = Ffast_thisyear
return F_sample_thisyear

def get_turnovertime_combined(Fm_true, k_fast, fraction):

    #loss function
    def loss(k):
        F_modelled = F_sample_2pool(k, k_fast, fraction)
        return math.sqrt((Fm_true - F_modelled)** 2)
    result = sp.optimize.least_squares(loss, 0.001)
    optimal_k = result.x[0]
    turnovertime_slow = 1/optimal_k
    turnovertime_fast = 1/k_fast
    turnovertime = fraction * turnovertime_fast + (1-fraction) * turnovertime_slow
    return turnovertime

def get_turnovertime_column(Fm_true, k_fast, fraction):
    result = []
    for i in range(len(Fm_true)):
        result.append(get_turnovertime_combined(Fm_true[i], k_fast[i], fraction))
    return result

```

Assuming the turnover time and the size of the labile pool the turnover time of the stabilized pool is calculated for each compound and saved as a .csv

```

In [73]: k_fast = [1/x for x in fast_turnover]
turnover_twopool = Fm_samples.apply(lambda col: get_turnovertime_column(col, k_fast, fast_fraction))
turnover_twopool['fast_turnover'] = fast_turnover
turnover_twopool['fast_fraction'] = fast_fraction
print(turnover_twopool)
turnover_twopool.to_csv('two_pool_turnover_{}.csv'.format(file_info))

```

	Alkane	C26_FA	C28_FA	SCFA	brGDGT	bulk	isoGDGT	fast_turnover	\
Bb 0-5	361	2741	1669	344	446	319	739	344	
Bb 10-20	1446	1214	1413	921	1569	1194	1370	921	
Bb 20-40	3198	2638	3214	761	2023	1765	2697	761	
Ln 0-5	36	33	33	33	547	65	357	33	
Ln 10-20	503	399	249	242	1432	271	1991	242	
Ln 60-80	8581	1080	3236	299	6114	2227	3700	299	

fast_fraction

```
Bb 0-5          9e-01
Bb 10-20       2e-01
Bb 20-40       1e-01
Ln 0-5         2e-01
Ln 10-20       1e-01
Ln 60-80       9e-02
```

plot the modelled evolution of a single sample

```
In [74]: #plotting for single pool model
         #enter sample D14C
         Ftrue = 0.995
         #enter fitted turnover time
         t = 359

         k=1/t

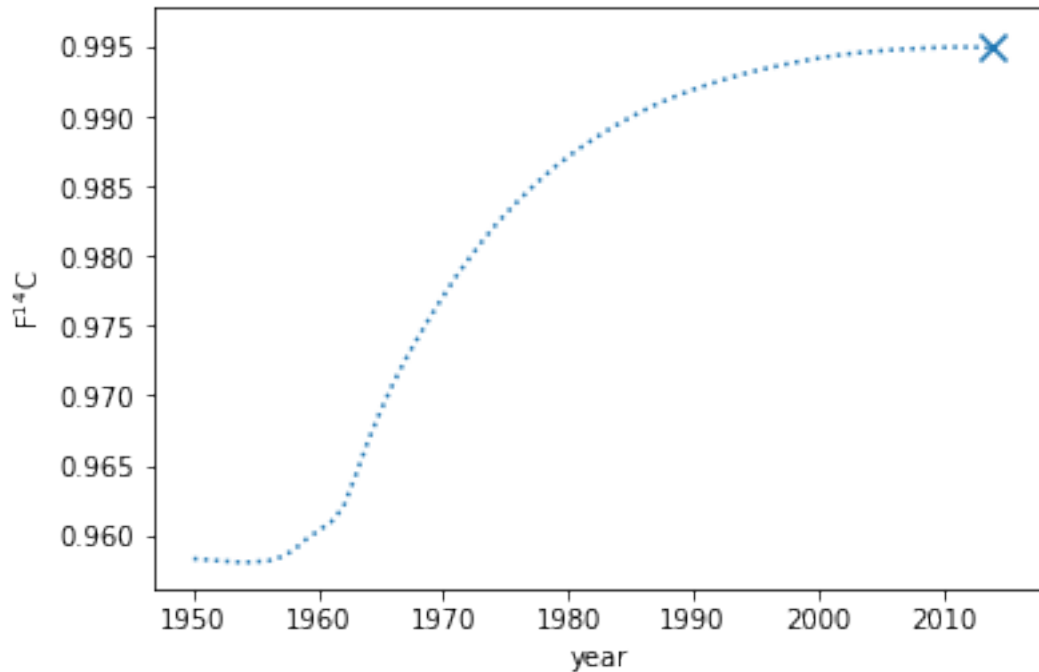
         y=[]

         F_lastyear = find_Fini(k)
         for F_atmo in Fm_atmo_model:
             F_sample_thisyear = F_thisyear(k, F_lastyear, F_atmo)
             F_lastyear = F_sample_thisyear
             y.append(F_sample_thisyear)

         x = list(range(1950, 2015))

         plt.figure
         plt.plot(x,y, linestyle=':')
         plt.xlabel('year')
         plt.ylabel('$\mathregular{F^{14}C}$')
         plt.scatter(2014, Ftrue, marker = 'x', s=100)
         plt.show
```

```
Out[74]: <function matplotlib.pyplot.show>
```



```
In [10]: #plotting for two-pool model
#enter Fm of the sample, the turnover times of the labile and the stable p
Ftrue=1.009
t_slow= 420
t_fast=33
fraction = 0.162

k=1/t_slow
k_fast=1/t_fast

y=[]
y_fast=[]
y_slow=[]

Fslow_lastyear = find_Fini(k)
Ffast_lastyear = find_Fini(k_fast)

for F_atmo in Fm_atmo_model:
    Fslow_thisyear = F_thisyear(k, Fslow_lastyear, F_atmo)
    Ffast_thisyear = F_thisyear(k_fast, Ffast_lastyear, F_atmo)
    F_sample_thisyear = fraction*Ffast_thisyear+(1-fraction)*Fslow_thi
    Fslow_lastyear = Fslow_thisyear
    Ffast_lastyear = Ffast_thisyear
```

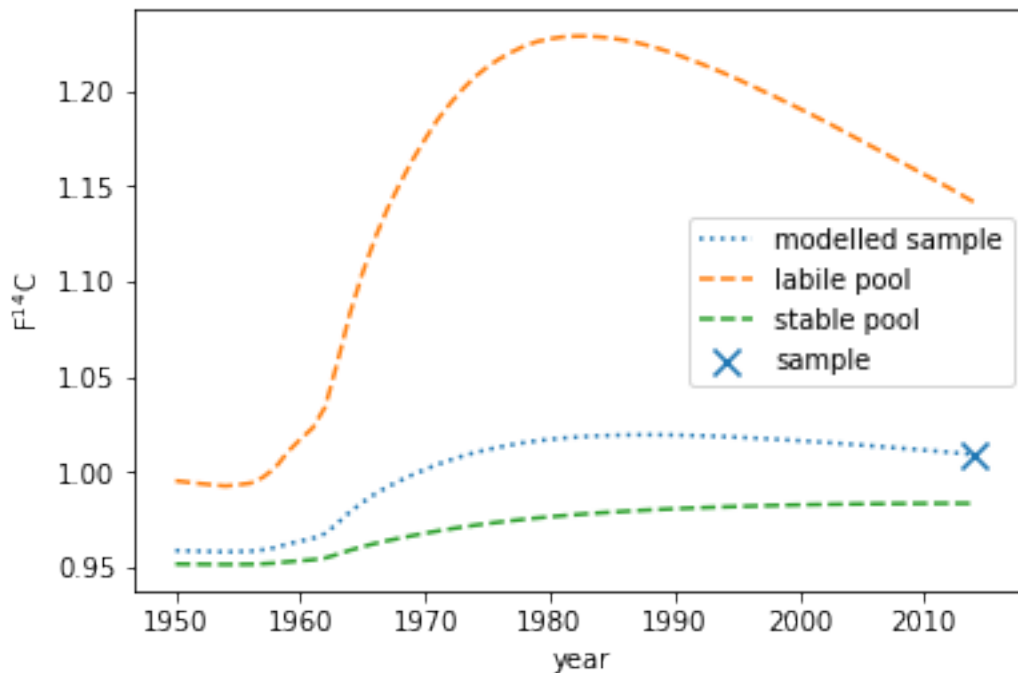
```

y_fast.append(Ffast_thisyear)
y_slow.append(Fslow_thisyear)
y.append(F_sample_thisyear)

x = list(range(1950, 2015))

plt.figure
plt.plot(x, y, linestyle=':', label='modelled sample')
plt.plot(x, y_fast, linestyle='--', label='labile pool')
plt.plot(x, y_slow, linestyle='--', label='stable pool')
plt.xlabel('year')
plt.ylabel('$\mathregular{F^{14}C}$')
plt.scatter(2014, Ftrue, marker = 'x', s=100, label='sample')
plt.legend()
plt.show()

```



Sensitivity of the model to assumptions

```

In [84]: #set the Fm of the sample, assumed fast turnover time and the fraction of
Fm_true = 0.705
t_fast = 299
f_fast = 0.087
#set a range of fast turnover times and fractions to check
t_fast_range = range(1,799)
f_fast_range = np.linspace(0.01, 0.187, 50)

```



```

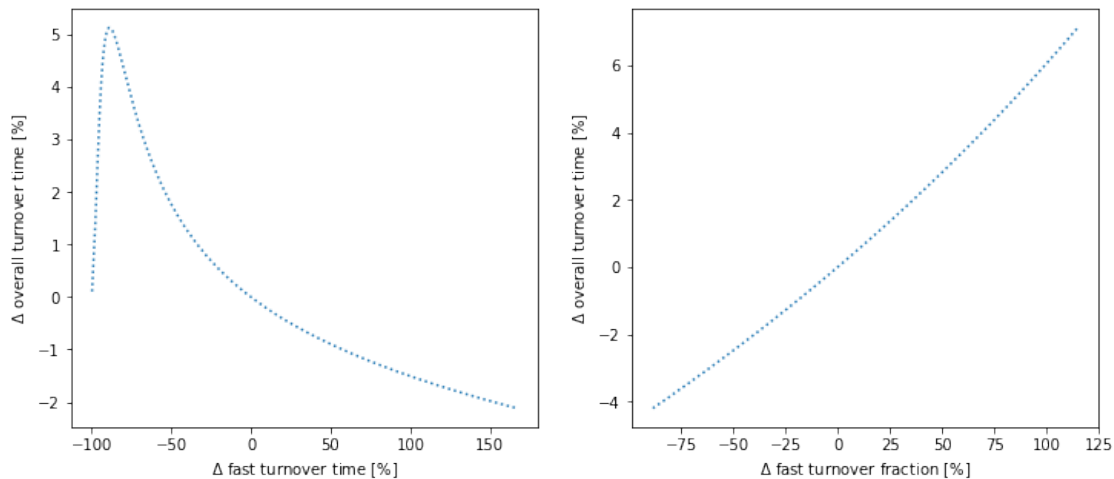
diff_t = []
diff_t_result = []
diff_f = []
diff_f_result = []
tf_ref = get_turnovertime_combined(Fm_true, 1/t_fast, f_fast)

for t in t_fast_range:
    diff_t.append((t-t_fast)/t_fast*100)
    diff_t_result.append((get_turnovertime_combined(Fm_true, 1/t, f_fast)-
tf_ref)/tf_ref*100)

for f in f_fast_range:
    diff_f.append((f-f_fast)/f_fast*100)
    diff_f_result.append((get_turnovertime_combined(Fm_true, 1/t_fast, f)-
tf_ref)/tf_ref*100)

_,axs = plt.subplots(1,2, figsize=(12,5))
plt.sca(axs[0])
plt.plot(diff_t, diff_t_result, linestyle=':')
plt.xlabel('$\Delta$ fast turnover time [%]')
plt.ylabel('$\Delta$ overall turnover time [%]')
plt.sca(axs[1])
plt.plot(diff_f, diff_f_result, linestyle=':')
plt.xlabel('$\Delta$ fast turnover fraction [%]')
plt.ylabel('$\Delta$ overall turnover time [%]')
plt.show()

```



In []:

In []:

In []:

In []: