'''

INVENTORY UTILITIES

'''

#%% Import modules

import numpy as np

import pandas as pd

import geopandas as gpd

import fiona

import time

import copy

import gc as garc

import matplotlib.pyplot as plt

import scipy.io as spio

import fcgadgets.macgyver.util\_gis as gis

import fcgadgets.macgyver.util\_general as gu

import fcgadgets.cbrunner.cbrun\_util as cbu

import fcgadgets.taz.aspatial\_stat\_models as asm

#%% Import variables

def Process1\_ImportVariables(meta,pNam):

#----------------------------------------------------------------------

# Define regular grid sampling frequency and name of mask

#----------------------------------------------------------------------

rgsf=str(meta['Geos']['RGSF'])

if meta[pNam]['Project']['Name Project']=='BCFCS\_LUC':

mask='BCFCS\_LUC'

elif meta[pNam]['Project']['Name Project']=='BCFCS\_EvalAtPlots':

mask='BCFCS\_EValAtPlots'

elif meta[pNam]['Project']['Name Project']=='BCFCS\_EvalAtCN':

mask='BCFCS\_EvalAtCN'

elif meta[pNam]['Project']['Name Project']=='BCFCS\_NOSEC':

mask='NOSE'

elif meta[pNam]['Project']['Name Project']=='BCFCS\_NMC':

mask='BCFCS\_NMC'

elif meta[pNam]['Project']['Name Project']=='BCFCS\_Eval':

mask='BCFCS\_Eval'

elif meta[pNam]['Project']['Name Project']=='BCFCS\_EvalCoast':

mask='BCFCS\_EvalCoast'

elif meta[pNam]['Project']['Name Project']=='BCFCS\_EvalInterior':

mask='BCFCS\_EvalInterior'

elif meta[pNam]['Project']['Name Project']=='TSA\_DawsonCreek':

mask='TSA\_DawsonCreek'

else:

# The default

mask='Province'

#--------------------------------------------------------------------------

# Define land surface attributes

#--------------------------------------------------------------------------

print('Preparing land surface attributes')

t0=time.time()

lsat={}

# Land cover / land use

vL=['LandCover\_Comp1\_1800','LandCover\_Comp1\_2019',

'LandUse\_Comp1\_2019','LandUse\_Comp1\_2049\_Scn1','LandUse\_Comp1\_2049\_Scn2',

'LandUseChange\_Comp1\_1800to2019\_Year','LandUseChange\_Comp1\_1800to2019\_Type',

'LandUseChange\_Comp1\_2020to2049\_Scn1\_Year','LandUseChange\_Comp1\_2020to2049\_Scn1\_Type',

'LandUseChange\_Comp1\_2020to2049\_Scn2\_Year','LandUseChange\_Comp1\_2020to2049\_Scn2\_Type']

for v in vL:

lsat[v]=gu.ipickle(meta['Paths']['bc1ha'] + '\\Sparse\\RGSF' + rgsf + '\_Mask' + mask + '\_' + v + '.pkl')

# BGC zone (no longer using VRI because there are errors and gaps)

z=gis.OpenGeoTiff(meta['Paths']['bc1ha'] + '\\BEC\_BIOGEOCLIMATIC\_POLY\\ZONE\_GapFilled.tif')

lsat['ID\_BGCZ']=gis.UpdateGridCellsize(z,meta['Geos']['RGSF'])['Data'][ meta['Geos']['iMask'] ]

lsat['ID\_BGCZ'][lsat['ID\_BGCZ']==0]=7

# Region code

lsat['Region Code']=meta['LUT']['Region']['Interior']\*np.ones(meta[pNam]['Project']['N Stand'],dtype='int8')

cd=[meta['LUT']['BEC\_BIOGEOCLIMATIC\_POLY']['ZONE']['CDF'],

meta['LUT']['BEC\_BIOGEOCLIMATIC\_POLY']['ZONE']['CWH'],

meta['LUT']['BEC\_BIOGEOCLIMATIC\_POLY']['ZONE']['MH']]

ind=np.where(np.isin(lsat['ID\_BGCZ'],cd)==True)

lsat['Region Code'][ind]=meta['LUT']['Region']['Coast']

# Age

if 'Custom Age Source' in meta[pNam]['Project']:

z=gis.OpenGeoTiff(meta[pNam]['Project']['Custom Age Source'])

lsat['Age']=gis.UpdateGridCellsize(z,meta['Geos']['RGSF'])['Data'][ meta['Geos']['iMask'] ]

else:

z=gis.OpenGeoTiff(meta['Paths']['bc1ha'] + '\\VRI 2023\\PROJ\_AGE\_1.tif')

lsat['Age']=gis.UpdateGridCellsize(z,meta['Geos']['RGSF'])['Data'][ meta['Geos']['iMask'] ]

# Site index

if 'Custom SI Source' in meta[pNam]['Project']:

z=gis.OpenGeoTiff(meta[pNam]['Project']['Custom SI Source'])

lsat['SI']=gis.UpdateGridCellsize(z,meta['Geos']['RGSF'])['Data'][ meta['Geos']['iMask'] ]

else:

# From BGC Zone table

lsat['SI']=18\*np.ones(meta[pNam]['Project']['N Stand'])

u=np.unique(lsat['ID\_BGCZ'])

for iU in range(u.size):

cd=cbu.lut\_n2s(meta['LUT']['BEC\_BIOGEOCLIMATIC\_POLY']['ZONE'],u[iU])

ind1=np.where(lsat['ID\_BGCZ']==u[iU])[0]

ind2=np.where(meta['Param']['BE']['BGC Zone Averages']['Name']==cd)[0]

lsat['SI'][ind1]=meta['Param']['BE']['BGC Zone Averages']['SI SME'][ind2]

# Natural establishment parameters (by BGC)

lsat['SPH Init Natural']=1500\*np.ones(meta[pNam]['Project']['N Stand'],dtype='int16')

lsat['Regen Delay Natural']=0\*np.ones(meta[pNam]['Project']['N Stand'],dtype='int16')

lsat['Pile Burn Rate']=0\*np.ones(meta[pNam]['Project']['N Stand'],dtype='int16')

u=np.unique(lsat['ID\_BGCZ'])

for iU in range(u.size):

ind0=np.where(lsat['ID\_BGCZ']==u[iU])[0]

ind1=np.where(meta['Param']['BE']['BGC Zone Averages']['Name']==cbu.lut\_n2s(meta['LUT']['BEC\_BIOGEOCLIMATIC\_POLY']['ZONE'],u[iU])[0])[0]

lsat['SPH Init Natural'][ind0]=meta['Param']['BE']['BGC Zone Averages']['Natural Initial Tree Density'][ind1[0]]

lsat['Regen Delay Natural'][ind0]=meta['Param']['BE']['BGC Zone Averages']['Natural Regeneration Delay'][ind1[0]]

lsat['Pile Burn Rate'][ind0]=100\*meta['Param']['BE']['BGC Zone Averages']['Pile Burn Rate'][ind1[0]]

# Species

if 'Custom Species Source' in meta[pNam]['Project']:

# Use custom species

z=gis.OpenGeoTiff(meta[pNam]['Project']['Custom Species Source'])

lsat['Spc1\_ID']=gis.UpdateGridCellsize(z,meta['Geos']['RGSF'])['Data'][ meta['Geos']['iMask'] ]

lsat['Spc2\_ID']=9999\*np.ones(lsat['Spc1\_ID'].size,dtype='int16')

lsat['Spc3\_ID']=9999\*np.ones(lsat['Spc1\_ID'].size,dtype='int16')

lsat['Spc4\_ID']=9999\*np.ones(lsat['Spc1\_ID'].size,dtype='int16')

lsat['Spc5\_ID']=9999\*np.ones(lsat['Spc1\_ID'].size,dtype='int16')

lsat['Spc1\_P']=100\*np.ones(lsat['Spc1\_ID'].size,dtype='int16')

lsat['Spc2\_P']=9999\*np.ones(lsat['Spc1\_ID'].size,dtype='int16')

lsat['Spc3\_P']=9999\*np.ones(lsat['Spc1\_ID'].size,dtype='int16')

lsat['Spc4\_P']=9999\*np.ones(lsat['Spc1\_ID'].size,dtype='int16')

lsat['Spc5\_P']=9999\*np.ones(lsat['Spc1\_ID'].size,dtype='int16')

else:

# Use species from VRI

for s in range(5):

ss=str(s+1)

z=gis.OpenGeoTiff(meta['Paths']['bc1ha'] + '\\VRI 2023\\SPECIES\_CD\_' + ss + '.tif')

lsat['Spc' + ss + '\_ID']=gis.UpdateGridCellsize(z,meta['Geos']['RGSF'])['Data'][ meta['Geos']['iMask'] ]

z=gis.OpenGeoTiff(meta['Paths']['bc1ha'] + '\\VRI 2023\\SPECIES\_PCT\_' + ss + '.tif')

lsat['Spc' + ss + '\_P']=gis.UpdateGridCellsize(z,meta['Geos']['RGSF'])['Data'][ meta['Geos']['iMask'] ]

# Clean species

ind=np.where( (lsat['Spc1\_ID']==0) )[0]

lsat['Spc1\_ID'][ind]=meta['LUT']['VEG\_COMP\_LYR\_R1\_POLY']['SPECIES\_CD\_1']['PL']

lsat['Spc1\_P'][ind]=100

for s in range(5):

ss=str(s+1)

ind=np.where( (lsat['Spc' + ss + '\_ID']==0) )[0]

lsat['Spc' + ss + '\_ID'][ind]=9999

lsat['Spc' + ss + '\_P'][ind]=9999

# Tree density class

z=gis.OpenGeoTiff(meta['Paths']['bc1ha'] + '\\LandCoverUse\\TreeDensityClass\_Current.tif')

lsat['Tree Density Class']=gis.UpdateGridCellsize(z,meta['Geos']['RGSF'])['Data'][ meta['Geos']['iMask'] ]

lsat['Tree Density Class'][lsat['Tree Density Class']==0]=2

# Operational adjustment factors

if 'Custom OAF1' in meta[pNam]['Project']:

lsat['OAF1']=(100\*meta[pNam]['Project']['Custom OAF1'])\*np.ones(meta[pNam]['Project']['N Stand'],dtype='int16')

else:

lsat['OAF1']=(100\*meta['Modules']['GYM']['OAF1 Default'])\*np.ones(meta[pNam]['Project']['N Stand'],dtype='int16')

u=np.unique(lsat['Tree Density Class'])

for iU in range(u.size):

ind0=np.where(lsat['Tree Density Class']==u[iU])[0]

ind1=np.where(meta['Param']['BE']['Tree Density Class Averages']['Name']==cbu.lut\_n2s(meta['LUT']['Derived']['tdc'],u[iU])[0])[0]

lsat['OAF1'][ind0]=100\*meta['Param']['BE']['Tree Density Class Averages']['OAF1'][ind1[0]]

# Wood density

lsat['Wood Density']=meta['Param']['BE']['Biophysical']['Density Wood']\*np.ones(meta[pNam]['Project']['N Stand'],dtype='int16')

u=np.unique(lsat['Spc1\_ID'])

for iU in range(u.size):

ind0=np.where(lsat['Spc1\_ID']==u[iU])[0]

ind1=np.where(meta['Param']['BE']['Wood Density']['Species CD']==cbu.lut\_n2s(meta['LUT']['VEG\_COMP\_LYR\_R1\_POLY']['SPECIES\_CD\_1'],u[iU])[0])[0]

lsat['Wood Density'][ind0]=meta['Param']['BE']['Wood Density']['Wood Density (kg/m3)'][ind1[0]]

# Harvest year

z=gis.OpenGeoTiff(meta['Paths']['bc1ha'] + '\\Disturbances\\Harvest\_Comp2\_Year.tif')

z=gis.UpdateGridCellsize(z,meta['Geos']['RGSF'])['Data'][ meta['Geos']['iMask'] ]

lsat['Harvest Year Comp2']=z

# Harvest first year

z=gis.OpenGeoTiff(meta['Paths']['bc1ha'] + '\\VEG\_CONSOLIDATED\_CUT\_BLOCKS\_SP\\HARVEST\_YEAR\_YearFirst.tif')

z=gis.UpdateGridCellsize(z,meta['Geos']['RGSF'])['Data'][ meta['Geos']['iMask'] ]

lsat['Year Harvest First']=z

# Adjustments to the stocking and fertility of harvested stands

flg=1

if flg==1:

# Occupancy

iH=np.where(lsat['Harvest Year Comp2']>0)

lsat['OAF1'][iH]=92

# Site index

#lsat['SI'][iH]=lsat['SI'][iH]+1

# iH=np.where(lsat['Harvest Year Comp2']==0)

# lsat['SI'][iH]=lsat['SI'][iH]-1

# Harvest probability map

z=gis.OpenGeoTiff(meta['Paths']['bc1ha'] + '\\Disturbances\\HarvestProbability.tif')

lsat['Prob Harvest (%/yr) x 1000']=gis.UpdateGridCellsize(z,meta['Geos']['RGSF'])['Data'][ meta['Geos']['iMask'] ]

# Define timber harvesting land base

lsat=DefineTHLB(meta,pNam,lsat)

print(str(np.round((time.time()-t0)/60,decimals=1)) + ' min')

#--------------------------------------------------------------------------

# Import disturbance and management event chronology

#--------------------------------------------------------------------------

print('Preparing Disturbance/management event chronology')

t0=time.time()

# Initiate disturbance-management event history

dmec0=[None]\*meta[pNam]['Project']['N Stand']

# Initialize a counter for the number of events recorded for each stand

cnt\_e=np.zeros(meta[pNam]['Project']['N Stand'],'int16')

# Initialize each stand with dummy events that weill be truncated after

N\_Events\_Init=50

for iStand in range(meta[pNam]['Project']['N Stand']):

dmec0[iStand]={}

dmec0[iStand]['Year']=9999\*np.ones(N\_Events\_Init,dtype='float')

dmec0[iStand]['Month']=9999\*np.ones(N\_Events\_Init,dtype='int16')

dmec0[iStand]['Day']=9999\*np.ones(N\_Events\_Init,dtype='int16')

dmec0[iStand]['ID Event Type']=9999\*np.ones(N\_Events\_Init,dtype='int16')

dmec0[iStand]['Mortality Factor']=9999\*np.ones(N\_Events\_Init,dtype='int16')

dmec0[iStand]['Growth Factor']=9999\*np.ones(N\_Events\_Init,dtype='int16')

dmec0[iStand]['SILV\_FUND\_SOURCE\_CODE']=9999\*np.ones(N\_Events\_Init,dtype='int16')

dmec0[iStand]['RegenType']=9999\*np.ones(N\_Events\_Init,dtype='int16')

dmec0[iStand]['Planted SPH']=9999\*np.ones(N\_Events\_Init,dtype='int16')

for iSpc in range(6):

dmec0[iStand]['PL\_SPECIES\_CD' + str(iSpc+1)]=9999\*np.ones(N\_Events\_Init,dtype='int16')

dmec0[iStand]['PL\_SPECIES\_PCT' + str(iSpc+1)]=9999\*np.ones(N\_Events\_Init,dtype='int16')

dmec0[iStand]['PL\_SPECIES\_GW' + str(iSpc+1)]=9999\*np.ones(N\_Events\_Init,dtype='int16')

if meta[pNam]['Project']['Special Attribution Method']=='NOSE':

dmec0[iStand]['Index to Event Inciting NOSE']=9999\*np.ones(N\_Events\_Init,dtype='int8')

if meta[pNam]['Project']['DMEC Method']=='From Events':

#----------------------------------------------------------------------

# Events are taken from historical record

#----------------------------------------------------------------------

# Add land use change (1800-2019)

iAffected=np.where(lsat['LandUseChange\_Comp1\_1800to2019\_Year']>0)[0]

for iS in iAffected:

if lsat['LandUseChange\_Comp1\_1800to2019\_Type'][iS]==0:

continue

cd=cbu.lut\_n2s(meta['LUT']['Derived']['lclu\_chng\_comp1'],lsat['LandUseChange\_Comp1\_1800to2019\_Type'][iS])[0]

idType=meta['LUT']['Event'][cd]

dmec0[iS]['Year'][cnt\_e[iS]]=lsat['LandUseChange\_Comp1\_1800to2019\_Year'][iS]

dmec0[iS]['ID Event Type'][cnt\_e[iS]]=idType

dmec0[iS]['Mortality Factor'][cnt\_e[iS]]=100

cnt\_e[iS]=cnt\_e[iS]+1

# Add slashpile burn

dmec0[iS]['Year'][cnt\_e[iS]]=lsat['LandUseChange\_Comp1\_1800to2019\_Year'][iS]+1

dmec0[iS]['ID Event Type'][cnt\_e[iS]]=meta['LUT']['Event']['Slashpile Burn']

dmec0[iS]['Mortality Factor'][cnt\_e[iS]]=100

cnt\_e[iS]=cnt\_e[iS]+1

# Add regen failure

dmec0[iS]['Year'][cnt\_e[iS]]=lsat['LandUseChange\_Comp1\_1800to2019\_Year'][iS]+2

dmec0[iS]['ID Event Type'][cnt\_e[iS]]=meta['LUT']['Event']['Regen Failure']

dmec0[iS]['Mortality Factor'][cnt\_e[iS]]=100

cnt\_e[iS]=cnt\_e[iS]+1

# Add land use change (2020-2049)

iAffected=np.where(lsat['LandUseChange\_Comp1\_2020to2049\_Scn1\_Year']>0)[0]

for iS in iAffected:

if lsat['LandUseChange\_Comp1\_2020to2049\_Scn1\_Year'][iS]==0:

continue

cd=cbu.lut\_n2s(meta['LUT']['Derived']['lclu\_chng\_comp1'],lsat['LandUseChange\_Comp1\_2020to2049\_Scn1\_Type'][iS])[0]

idType=meta['LUT']['Event'][cd]

dmec0[iS]['Year'][cnt\_e[iS]]=lsat['LandUseChange\_Comp1\_2020to2049\_Scn1\_Year'][iS]

dmec0[iS]['ID Event Type'][cnt\_e[iS]]=idType

dmec0[iS]['Mortality Factor'][cnt\_e[iS]]=100

cnt\_e[iS]=cnt\_e[iS]+1

# Add slashpile burn

dmec0[iS]['Year'][cnt\_e[iS]]=lsat['LandUseChange\_Comp1\_2020to2049\_Scn1\_Year'][iS]+1

dmec0[iS]['ID Event Type'][cnt\_e[iS]]=meta['LUT']['Event']['Slashpile Burn']

dmec0[iS]['Mortality Factor'][cnt\_e[iS]]=100

cnt\_e[iS]=cnt\_e[iS]+1

# Add regen failure

dmec0[iS]['Year'][cnt\_e[iS]]=lsat['LandUseChange\_Comp1\_2020to2049\_Scn1\_Year'][iS]+2

dmec0[iS]['ID Event Type'][cnt\_e[iS]]=meta['LUT']['Event']['Regen Failure']

dmec0[iS]['Mortality Factor'][cnt\_e[iS]]=100

cnt\_e[iS]=cnt\_e[iS]+1

# Add wildfire observations

zY=gu.ipickle(meta['Paths']['bc1ha'] + '\\Sparse\\RGSF' + rgsf + '\_Mask' + mask + '\_PROT\_HISTORICAL\_FIRE\_POLYS\_SP\_Year.pkl')

zS=gu.ipickle(meta['Paths']['bc1ha'] + '\\Sparse\\RGSF' + rgsf + '\_Mask' + mask + '\_PROT\_HISTORICAL\_FIRE\_POLYS\_SP\_SevClass.pkl')

for iY in range(6):

iAffected=np.where(zY[iY]>0)[0]

UseBurnSev='On'

mort\_wo\_bs=59

#mort\_wo\_bs=100

if UseBurnSev=='Off':

for iS in iAffected:

dmec0[iS]['Year'][cnt\_e[iS]]=zY[iY][iS]

dmec0[iS]['ID Event Type'][cnt\_e[iS]]=meta['LUT']['Event']['Wildfire']

dmec0[iS]['Mortality Factor'][cnt\_e[iS]]=mort\_wo\_bs

cnt\_e[iS]=cnt\_e[iS]+1

else:

for iS in iAffected:

if zS[iY][iS]==meta['LUT']['Derived']['burnsev\_comp1']['Low']:

dmec0[iS]['Year'][cnt\_e[iS]]=zY[iY][iS]

dmec0[iS]['ID Event Type'][cnt\_e[iS]]=meta['LUT']['Event']['Wildfire']

dmec0[iS]['Mortality Factor'][cnt\_e[iS]]=59

elif zS[iY][iS]==meta['LUT']['Derived']['burnsev\_comp1']['Medium']:

dmec0[iS]['Year'][cnt\_e[iS]]=zY[iY][iS]

dmec0[iS]['ID Event Type'][cnt\_e[iS]]=meta['LUT']['Event']['Wildfire']

dmec0[iS]['Mortality Factor'][cnt\_e[iS]]=59

elif zS[iY][iS]==meta['LUT']['Derived']['burnsev\_comp1']['High']:

dmec0[iS]['Year'][cnt\_e[iS]]=zY[iY][iS]

dmec0[iS]['ID Event Type'][cnt\_e[iS]]=meta['LUT']['Event']['Wildfire']

dmec0[iS]['Mortality Factor'][cnt\_e[iS]]=59

else:

dmec0[iS]['Year'][cnt\_e[iS]]=zY[iY][iS]

dmec0[iS]['ID Event Type'][cnt\_e[iS]]=meta['LUT']['Event']['Wildfire']

dmec0[iS]['Mortality Factor'][cnt\_e[iS]]=mort\_wo\_bs

cnt\_e[iS]=cnt\_e[iS]+1

# Add insect outbreak observations from Pest Compilation 1

# The compilation adds insects into one year and one type file to save time.

zY=gu.ipickle(meta['Paths']['bc1ha'] + '\\Sparse\\RGSF' + rgsf + '\_Mask' + mask + '\_InsectComp1\_Year.pkl')

zT=gu.ipickle(meta['Paths']['bc1ha'] + '\\Sparse\\RGSF' + rgsf + '\_Mask' + mask + '\_InsectComp1\_Type.pkl')

for iY in range(10):

iAffected=np.where( (zY[iY]>0) )[0]

for iS in iAffected:

ind=np.where( (meta['Param']['BE']['InsectComp1']['ID']==zT[iY][iS]) )[0][0]

dmec0[iS]['Year'][cnt\_e[iS]]=zY[iY][iS]

dmec0[iS]['ID Event Type'][cnt\_e[iS]]=meta['LUT']['Event'][meta['Param']['BE']['InsectComp1']['Insect Name'][ind]]

dmec0[iS]['Mortality Factor'][cnt\_e[iS]]=meta['Param']['BE']['InsectComp1']['Mortality (%)'][ind]

cnt\_e[iS]=cnt\_e[iS]+1

# Add harvest observations

zY=gu.ipickle(meta['Paths']['bc1ha'] + '\\Sparse\\RGSF' + rgsf + '\_Mask' + mask + '\_Harvest\_Comp2\_Year.pkl')

iAffected=np.where(zY>0)[0]

for iS in iAffected:

dmec0[iS]['Year'][cnt\_e[iS]]=zY[iS]

dmec0[iS]['ID Event Type'][cnt\_e[iS]]=meta['LUT']['Event']['Harvest']

dmec0[iS]['Mortality Factor'][cnt\_e[iS]]=100

cnt\_e[iS]=cnt\_e[iS]+1

# Pile burning

if np.random.random(1)<lsat['Pile Burn Rate'][iS].astype(float)/100:

dmec0[iS]['Year'][cnt\_e[iS]]=zY[iS]+1

dmec0[iS]['ID Event Type'][cnt\_e[iS]]=meta['LUT']['Event']['Slashpile Burn']

dmec0[iS]['Mortality Factor'][cnt\_e[iS]]=100

cnt\_e[iS]=cnt\_e[iS]+1

# zY=gu.ipickle(meta['Paths']['bc1ha'] + '\\Sparse\\RGSF' + rgsf + '\_Mask' + mask + '\_VEG\_CONSOLIDATED\_CUT\_BLOCKS\_SP\_Year.pkl')

# for iY in range(3):

# iAffected=np.where(zY[iY]>0)[0]

# for iS in iAffected:

# dmec0[iS]['Year'][cnt\_e[iS]]=zY[iY][iS]

# dmec0[iS]['ID Event Type'][cnt\_e[iS]]=meta['LUT']['Event']['Harvest']

# dmec0[iS]['Mortality Factor'][cnt\_e[iS]]=100

# cnt\_e[iS]=cnt\_e[iS]+1

# # Pile burning

# if np.random.random(1)<lsat['Pile Burn Rate'][iS].astype(float)/100:

# dmec0[iS]['Year'][cnt\_e[iS]]=zY[iY][iS]+1

# dmec0[iS]['ID Event Type'][cnt\_e[iS]]=meta['LUT']['Event']['Slashpile Burn']

# dmec0[iS]['Mortality Factor'][cnt\_e[iS]]=100

# cnt\_e[iS]=cnt\_e[iS]+1

# Add planting observations

zY=gu.ipickle(meta['Paths']['bc1ha'] + '\\Sparse\\RGSF' + rgsf + '\_Mask' + mask + '\_PL\_All\_Year.pkl')

zFSC=gu.ipickle(meta['Paths']['bc1ha'] + '\\Sparse\\RGSF' + rgsf + '\_Mask' + mask + '\_PL\_All\_SILV\_FUND\_SOURCE\_CODE.pkl')

zSPH=gu.ipickle(meta['Paths']['bc1ha'] + '\\Sparse\\RGSF' + rgsf + '\_Mask' + mask + '\_PL\_All\_SPH\_Planted.pkl')

zRT=gu.ipickle(meta['Paths']['bc1ha'] + '\\Sparse\\RGSF' + rgsf + '\_Mask' + mask + '\_PL\_All\_RegenType.pkl')

cd={}; pct={}; gw={}

for iSpc in range(6):

cd[iSpc+1]=gu.ipickle(meta['Paths']['bc1ha'] + '\\Sparse\\RGSF' + rgsf + '\_Mask' + mask + '\_PL\_SPECIES\_CD' + str(iSpc+1) + '.pkl')

pct[iSpc+1]=gu.ipickle(meta['Paths']['bc1ha'] + '\\Sparse\\RGSF' + rgsf + '\_Mask' + mask + '\_PL\_SPECIES\_PCT' + str(iSpc+1) + '.pkl')

gw[iSpc+1]=gu.ipickle(meta['Paths']['bc1ha'] + '\\Sparse\\RGSF' + rgsf + '\_Mask' + mask + '\_PL\_SPECIES\_GW' + str(iSpc+1) + '.pkl')

for iY in range(6):

# Remove zeros

iAffected=np.where(cd[iSpc+1][iY]==0)[0]

cd[iSpc+1][iY][iAffected]=9999

for iY in range(6):

iAffected=np.where( (zY[iY]>0) & (zRT[iY]!=meta['LUT']['Derived']['RegenType']['Back-to-back Planting']) )[0]

for iS in iAffected:

dmec0[iS]['Year'][cnt\_e[iS]]=zY[iY][iS]

dmec0[iS]['ID Event Type'][cnt\_e[iS]]=meta['LUT']['Event']['Planting']

dmec0[iS]['Mortality Factor'][cnt\_e[iS]]=0

dmec0[iS]['SILV\_FUND\_SOURCE\_CODE'][cnt\_e[iS]]=zFSC[iY][iS]

dmec0[iS]['RegenType'][cnt\_e[iS]]=zRT[iY][iS]

dmec0[iS]['Planted SPH'][cnt\_e[iS]]=zSPH[iY][iS]

for iSpc in range(6):

dmec0[iS]['PL\_SPECIES\_CD' + str(iSpc+1)][cnt\_e[iS]]=cd[iSpc+1][iY][iS]

dmec0[iS]['PL\_SPECIES\_PCT' + str(iSpc+1)][cnt\_e[iS]]=pct[iSpc+1][iY][iS]

dmec0[iS]['PL\_SPECIES\_GW' + str(iSpc+1)][cnt\_e[iS]]=gw[iSpc+1][iY][iS]

cnt\_e[iS]=cnt\_e[iS]+1

# Add aerial fertilization observations

zY=gu.ipickle(meta['Paths']['bc1ha'] + '\\Sparse\\RGSF' + rgsf + '\_Mask' + mask + '\_FE-CA\_Year.pkl')

zFSC=gu.ipickle(meta['Paths']['bc1ha'] + '\\Sparse\\RGSF' + rgsf + '\_Mask' + mask + '\_FE-CA\_SILV\_FUND\_SOURCE\_CODE.pkl')

for iY in range(3):

iAffected=np.where(zY[iY]>0)[0]

for iS in iAffected:

dmec0[iS]['Year'][cnt\_e[iS]]=zY[iY][iS]

dmec0[iS]['ID Event Type'][cnt\_e[iS]]=meta['LUT']['Event']['Fertilization Aerial']

dmec0[iS]['SILV\_FUND\_SOURCE\_CODE'][cnt\_e[iS]]=zFSC[iY][iS]

dmec0[iS]['Mortality Factor'][cnt\_e[iS]]=0

cnt\_e[iS]=cnt\_e[iS]+1

# Add knockdown

zY=gu.ipickle(meta['Paths']['bc1ha'] + '\\Sparse\\RGSF' + rgsf + '\_Mask' + mask + '\_SP\_KD\_Year.pkl')

zFSC=gu.ipickle(meta['Paths']['bc1ha'] + '\\Sparse\\RGSF' + rgsf + '\_Mask' + mask + '\_SP\_KD\_SILV\_FUND\_SOURCE\_CODE.pkl')

for iY in range(3):

iAffected=np.where(zY[iY]>0)[0]

for iS in iAffected:

dmec0[iS]['Year'][cnt\_e[iS]]=zY[iY][iS]

dmec0[iS]['ID Event Type'][cnt\_e[iS]]=meta['LUT']['Event']['Knockdown']

dmec0[iS]['SILV\_FUND\_SOURCE\_CODE'][cnt\_e[iS]]=zFSC[iY][iS]

dmec0[iS]['Mortality Factor'][cnt\_e[iS]]=0

cnt\_e[iS]=cnt\_e[iS]+1

# Add prescribed burning

zY=gu.ipickle(meta['Paths']['bc1ha'] + '\\Sparse\\RGSF' + rgsf + '\_Mask' + mask + '\_SP-BU-BROAD\_Year.pkl')

for iY in range(3):

iAffected=np.where(zY[iY]>0)[0]

for iS in iAffected:

dmec0[iS]['Year'][cnt\_e[iS]]=zY[iY][iS]

dmec0[iS]['ID Event Type'][cnt\_e[iS]]=meta['LUT']['Event']['Prescribed Burn']

dmec0[iS]['Mortality Factor'][cnt\_e[iS]]=0

cnt\_e[iS]=cnt\_e[iS]+1

elif meta[pNam]['Project']['DMEC Method']=='From Inventory Age':

#----------------------------------------------------------------------

# Carbon stocks in the project year are being constrained by observations

# through input of specified age and productivity.

#----------------------------------------------------------------------

dType=meta['LUT']['Event']['Harvest']\*np.ones(lsat['Age'].size)

dType[np.where(np.random.random(lsat['Age'].size)<0.1)]=meta['LUT']['Event']['Wildfire']

for iStand in range(meta[pNam]['Project']['N Stand']):

dmec0[iStand]['Year'][cnt\_e[iStand]]=meta[pNam]['Project']['Year Project']-lsat['Age'][iStand]

dmec0[iStand]['ID Event Type'][cnt\_e[iStand]]=dType[iStand]

dmec0[iStand]['Mortality Factor'][cnt\_e[iStand]]=100

cnt\_e[iStand]=cnt\_e[iStand]+1

print(str(np.round((time.time()-t0)/60,decimals=1)) + ' min')

#--------------------------------------------------------------------------

# Exclude duplicate events

#--------------------------------------------------------------------------

# if meta[pNam]['Project']['Exclude duplicate events']=='On':

# print('Removing duplicate events from dmec0')

# t0=time.time()

# dmec0=Exclude\_Duplicate\_Events(meta,pNam,dmec0)

# print(str(np.round((time.time()-t0)/60,decimals=1)) + ' min')

#--------------------------------------------------------------------------

# Put events in order

# \*\*\* Must be done before several other processing steps \*\*\*

#--------------------------------------------------------------------------

print('Putting events in order')

t0=time.time()

dmec0=PutEventsInOrder(meta,pNam,dmec0)

print(str(np.round((time.time()-t0)/60,decimals=1)) + ' min')

#--------------------------------------------------------------------------

# Ensure that a stand-replacing disturbance precedes fertilization so that age

#--------------------------------------------------------------------------

# Assume previous disturbance must be missing. Assume fertilization occurs at age

# 35. Assume previous disturbance was harvest.

# \*\*\* Must be in order of calendar date first \*\*\*

# \*\*\* This will not work if the previous harvest or fire had a severity < 100. \*\*\*

# Only applies to cbrunner when fertilization is simulated from TIPSY

if meta[pNam]['Project']['Ensure aerial fert is preceded by disturbance']=='On':

print('Ensure stand-replacing disturbance precedes fertilization')

t0=time.time()

dmec0=Ensure\_Fert\_Preceded\_By\_Disturbance(meta,pNam,dmec0,lsat)

print(str(np.round((time.time()-t0)/60,decimals=1)) + ' min')

#--------------------------------------------------------------------------

# Ensure knockdown events are followed by slashpile events

# \*\*\* Not written. \*\*\*

#--------------------------------------------------------------------------

if meta[pNam]['Project']['Revise some knockdown to include slashpile burn']=='On':

print('Add slashpile burns after knockdown')

#dmec0=Revise\_Knockdown\_to\_Include\_Slashpile\_Burning(meta,dmec0)

#--------------------------------------------------------------------------

# Ensure every stand has a modern disturbance

# So that there is at least one event

#--------------------------------------------------------------------------

if meta[pNam]['Project']['Ensure every stand has a modern disturbance']=='On':

name\_dist='Wildfire'

severity=100

print('Ensure every stand has a disturbance in the modern era')

t0=time.time()

dmec0=Ensure\_Every\_Stand\_Has\_Modern\_Disturbance(meta,pNam,dmec0,name\_dist,severity)

print(str(np.round((time.time()-t0)/60,decimals=1)) + ' min')

#--------------------------------------------------------------------------

# Gap-fill stands with no event history based on age from inventory input

# So that there is at least one event

#--------------------------------------------------------------------------

if meta[pNam]['Project']['Gap-fill DMEC with inventory age']=='On':

print('Gap-fill DMEC with inventory age')

t0=time.time()

dmec0=GapFill\_DMEC\_WithAge(meta,pNam,dmec0,lsat)

print(str(np.round((time.time()-t0)/60,decimals=1)) + ' min')

#--------------------------------------------------------------------------

# IDW - Western Spruce Budworm - Fix severity

# The dmec was populated with the numeric severity ID. Mortality only occurs

# following repeated outrbreak years.

#--------------------------------------------------------------------------

# if meta[pNam]['Project']['Fix severity of western spruce budworm']=='On':

# print('Adjust severity of IDW')

# t0=time.time()

# dmec0=IDW\_Fix\_Severity(meta,dmec0)

# print(str(np.round((time.time()-t0)/60,decimals=1)) + ' min')

#--------------------------------------------------------------------------

# Reduce number of growth curves by adjusting site index

#--------------------------------------------------------------------------

if meta[pNam]['Project']['Revise SI to reduce num of growth curves']=='On':

print('Reducing the number of growth curves by lowering the precision of site index')

t0=time.time()

lsat=ReduceVariationInSiteIndex(meta,pNam,lsat)

print(str(np.round((time.time()-t0)/60,decimals=1)) + ' min')

#--------------------------------------------------------------------------

# Clean species composition - TIPSY will not recognize certain codes

# Decomissioned - now cleaned upon import

#--------------------------------------------------------------------------

# print('Cleaning species composition')

# t0=time.time()

# #meta,dmec0,vri,fcinv=Clean\_Species\_Composition(meta,dmec0,vri,fcinv)

# print(str(np.round((time.time()-t0)/60,decimals=1)) + ' min')

#--------------------------------------------------------------------------

# Put events in order

# \*\*\* Must be done before NOSE types are defined \*\*\*

#--------------------------------------------------------------------------

print('Putting events in order')

t0=time.time()

dmec0=PutEventsInOrder(meta,pNam,dmec0)

print(str(np.round((time.time()-t0)/60,decimals=1)) + ' min')

#--------------------------------------------------------------------------

# Define type of non-obligation stand establishment

#--------------------------------------------------------------------------

if meta[pNam]['Project']['Special Attribution Method']=='NOSE':

print('Defining types of non-obligation stand establishment')

t0=time.time()

meta,dmec0=Define\_NOSE\_ProjectType(meta,pNam,dmec0)

print(str(np.round((time.time()-t0)/60,decimals=1)) + ' min')

#--------------------------------------------------------------------------

# Put events in order

# \*\*\* Must be done before NOSE types are defined \*\*\*

#--------------------------------------------------------------------------

print('Putting events in order')

t0=time.time()

dmec0=PutEventsInOrder(meta,pNam,dmec0)

print(str(np.round((time.time()-t0)/60,decimals=1)) + ' min')

#--------------------------------------------------------------------------

# Remove excess size from each stand

#--------------------------------------------------------------------------

for iStand in range(meta[pNam]['Project']['N Stand']):

ind=np.where(dmec0[iStand]['Year']!=9999)

for k in dmec0[iStand].keys():

dmec0[iStand][k]=dmec0[iStand][k][ind]

#--------------------------------------------------------------------------

# Exclude unidentified activities or disturbances

#--------------------------------------------------------------------------

if meta[pNam]['Project']['Exclude unidentified events']=='On':

print('Removing unrecognized events from DMEC')

t0=time.time()

dmec0=Exclude\_Unidentified\_Events(meta,pNam,dmec0)

print(str(np.round((time.time()-t0)/60,decimals=1)) + ' min')

#--------------------------------------------------------------------------

# Expand DMEC to each scenario

#--------------------------------------------------------------------------

print('Expanding DMEC for each scenario')

t0=time.time()

dmec=[]

for iScn in range(meta[pNam]['Project']['N Scenario']):

dmec.append(copy.deepcopy(dmec0))

del dmec0

print(str(np.round((time.time()-t0)/60,decimals=1)) + ' min')

return meta,lsat,dmec

#%% Define type of stand establishment (salvage, knockdown, underplanting, NSR backlog)

# \*\*\* Decomissioned \*\*\*

# Create index to the inciting disturbance:

# It is critical that the above steps (ensuring there is a previous

# stand-replacing disturbance) work before this will work properly.

# If it is a salvage logging project, change the inciting disturbance (harvest)

# to be "Harvest Salvage" so that it removes a higher percentage of snags. I

# checked that a combo of "Harvest Salvage" + "Slashpile Burn" is consistent with

# the custom harvest (with slashpile burn) used in the salvage demo (March 2021).

# The mortality corrections can be overridden by the adjustment of species-specific

# mortality (Adjust species-specific mortality='Off' to avoid this)

def Define\_NOSE\_ProjectType(meta,pNam,dmec0):

# Initialize project type

meta[pNam]['Project']['RegenType']=np.zeros(meta[pNam]['Project']['N Stand'],dtype='int8')

meta[pNam]['Project']['Strata']['Project Type']['ID']=np.zeros(meta[pNam]['Project']['N Stand'],dtype='int8')

for iStand in range(meta[pNam]['Project']['N Stand']):

# Index to stand establishment events (exclude direct seeding)

iNOSE=np.where( (dmec0[iStand]['ID Event Type']==meta['LUT']['Event']['Planting']) & (np.isin(dmec0[iStand]['SILV\_FUND\_SOURCE\_CODE'],meta['Param']['BE']['FSC']['NO List ID'])==True) )[0]

if iNOSE.size==0:

continue

for i in iNOSE:

# Define open spaces if an event needs to be added

iOpen=np.where(dmec0[iStand]['Year']==9999)[0]

if iOpen.size>1:

iOpen=iOpen[0]

yr0=dmec0[iStand]['Year'][i]

if dmec0[iStand]['RegenType'][i]==meta['LUT']['Derived']['RegenType']['Salvage and Planting']:

#--------------------------------------------------------------

# Salvage

#--------------------------------------------------------------

# For non-ob salvage, assume high mortality in the natural disturbance inciting salvage

meta[pNam]['Project']['RegenType'][iStand]=meta['LUT']['Derived']['RegenType']['Salvage and Planting']

meta[pNam]['Project']['Strata']['Project Type']['ID'][iStand]=meta['LUT']['Derived']['RegenType']['Salvage and Planting']

iH=np.where( (dmec0[iStand]['ID Event Type']==meta['LUT']['Event']['Harvest']) & (dmec0[iStand]['Year']<=yr0) | (dmec0[iStand]['ID Event Type']==meta['LUT']['Event']['Harvest Salvage']) & (dmec0[iStand]['Year']<=yr0) )[0]

if iH.size==0:

# Small number of instances

# I think this is happing because NTEMS was considered in the RegenType classification

# but it is not being pulled into the model. Can be fixed.

yr\_lag=yr0-3

dmec0[iStand]['Year'][iOpen]=yr\_lag

dmec0[iStand]['ID Event Type'][iOpen]=meta['LUT']['Event']['Harvest Salvage']

dmec0[iStand]['Mortality Factor'][iOpen]=100

iH=np.where( (dmec0[iStand]['ID Event Type']==meta['LUT']['Event']['Harvest Salvage']) & (dmec0[iStand]['Year']==yr\_lag) )[0]

iOpen=iOpen+1

if iH.size>1:

iH=iH[-1]

iInc=np.where( (dmec0[iStand]['ID Event Type']==meta['LUT']['Event']['Wildfire']) & (dmec0[iStand]['Year']<dmec0[iStand]['Year'][iH]) | \

(dmec0[iStand]['ID Event Type']==meta['LUT']['Event']['Mountain Pine Beetle']) & (dmec0[iStand]['Year']<dmec0[iStand]['Year'][iH]) )[0]

if iInc.size==0:

dmec0[iStand]['Year'][iOpen]=dmec0[iStand]['Year'][iH]-3

dmec0[iStand]['ID Event Type'][iOpen]=meta['LUT']['Event']['Mountain Pine Beetle']

dmec0[iStand]['Mortality Factor'][iOpen]=100

dmec0[iStand]['Index to Event Inciting NOSE'][i]=iOpen

else:

if iInc.size>1:

iInc=iInc[-1]

dmec0[iStand]['Index to Event Inciting NOSE'][i]=iInc

if dmec0[iStand]['Mortality Factor'][iInc]<75:

dmec0[iStand]['Mortality Factor'][iInc]=90

elif dmec0[iStand]['RegenType'][i]==meta['LUT']['Derived']['RegenType']['Straight-to-planting Post Wildfire']:

#--------------------------------------------------------------

# Straight fire

#--------------------------------------------------------------

meta[pNam]['Project']['RegenType'][iStand]=meta['LUT']['Derived']['RegenType']['Straight-to-planting Post Wildfire']

meta[pNam]['Project']['Strata']['Project Type']['ID'][iStand]=meta['LUT']['Derived']['RegenType']['Straight-to-planting Post Wildfire']

iInc=np.where( (dmec0[iStand]['ID Event Type']==meta['LUT']['Event']['Wildfire']) & (dmec0[iStand]['Year']<yr0) )[0]

if iInc.size==0:

dmec0[iStand]['Year'][iOpen]=yr0-1

dmec0[iStand]['ID Event Type'][iOpen]=meta['LUT']['Event']['Wildfire']

dmec0[iStand]['Mortality Factor'][iOpen]=100

dmec0[iStand]['Index to Event Inciting NOSE'][i]=iOpen

else:

if iInc.size>1:

iInc=iInc[-1]

dmec0[iStand]['Index to Event Inciting NOSE'][i]=iInc

if dmec0[iStand]['Mortality Factor'][iInc]<100:

dmec0[iStand]['Mortality Factor'][iInc]=100

elif dmec0[iStand]['RegenType'][i]==meta['LUT']['Derived']['RegenType']['Straight-to-planting Post Beetles']:

#--------------------------------------------------------------

# Straight beetle

#--------------------------------------------------------------

meta[pNam]['Project']['RegenType'][iStand]=meta['LUT']['Derived']['RegenType']['Straight-to-planting Post Beetles']

meta[pNam]['Project']['Strata']['Project Type']['ID'][iStand]=meta['LUT']['Derived']['RegenType']['Straight-to-planting Post Beetles']

iInc=np.where( (dmec0[iStand]['ID Event Type']==meta['LUT']['Event']['Mountain Pine Beetle']) & (dmec0[iStand]['Year']<yr0) )[0]

if iInc.size==0:

dmec0[iStand]['Year'][iOpen]=yr0-1

dmec0[iStand]['ID Event Type'][iOpen]=meta['LUT']['Event']['Mountain Pine Beetle']

dmec0[iStand]['Mortality Factor'][iOpen]=100

dmec0[iStand]['Index to Event Inciting NOSE'][i]=iOpen

else:

if iInc.size>1:

iInc=iInc[-1]

dmec0[iStand]['Index to Event Inciting NOSE'][i]=iInc

if dmec0[iStand]['Mortality Factor'][iInc]<100:

dmec0[iStand]['Mortality Factor'][iInc]=100

elif dmec0[iStand]['RegenType'][i]==meta['LUT']['Derived']['RegenType']['Knockdown and Planting']:

#--------------------------------------------------------------

# Knockdown

#--------------------------------------------------------------

meta[pNam]['Project']['RegenType'][iStand]=meta['LUT']['Derived']['RegenType']['Knockdown and Planting']

meta[pNam]['Project']['Strata']['Project Type']['ID'][iStand]=meta['LUT']['Derived']['RegenType']['Knockdown and Planting']

iInc=np.where( (dmec0[iStand]['ID Event Type']==meta['LUT']['Event']['Knockdown']) & (dmec0[iStand]['Year']<yr0) )[0]

if iInc.size==0:

continue

if iInc.size>1:

iInc=iInc[-1]

dmec0[iStand]['Index to Event Inciting NOSE'][i]=iInc

if dmec0[iStand]['Mortality Factor'][iInc]<100:

dmec0[iStand]['Mortality Factor'][iInc]=100

elif dmec0[iStand]['RegenType'][i]==meta['LUT']['Derived']['RegenType']['Harvest and Planting NSR Backlog']:

#--------------------------------------------------------------

# NSR backlog

#--------------------------------------------------------------

meta[pNam]['Project']['RegenType'][iStand]=meta['LUT']['Derived']['RegenType']['Harvest and Planting NSR Backlog']

meta[pNam]['Project']['Strata']['Project Type']['ID'][iStand]=meta['LUT']['Derived']['RegenType']['Harvest and Planting NSR Backlog']

iInc=np.where( (dmec0[iStand]['ID Event Type']==meta['LUT']['Event']['Harvest']) & (dmec0[iStand]['Year']<yr0) | (dmec0[iStand]['ID Event Type']==meta['LUT']['Event']['Harvest Salvage']) & (dmec0[iStand]['Year']<yr0) )[0]

if iInc.size>1:

iInc=iInc[-1]

if iInc.size>0:

dmec0[iStand]['Index to Event Inciting NOSE'][i]=iInc

else:

dmec0[iStand]['Year'][iOpen]=yr0-10

dmec0[iStand]['ID Event Type'][iOpen]=meta['LUT']['Event']['Harvest']

dmec0[iStand]['Mortality Factor'][iOpen]=100

dmec0[iStand]['Index to Event Inciting NOSE'][i]=iOpen

elif dmec0[iStand]['RegenType'][i]==meta['LUT']['Derived']['RegenType']['Replanting']:

#--------------------------------------------------------------

# Replanting

#--------------------------------------------------------------

meta[pNam]['Project']['RegenType'][iStand]=meta['LUT']['Derived']['RegenType']['Replanting']

meta[pNam]['Project']['Strata']['Project Type']['ID'][iStand]=meta['LUT']['Derived']['RegenType']['Replanting']

dmec0[iStand]['Year'][iOpen]=yr0-1+0.9

dmec0[iStand]['ID Event Type'][iOpen]=meta['LUT']['Event']['Mountain Pine Beetle']

dmec0[iStand]['Mortality Factor'][iOpen]=100

dmec0[iStand]['Index to Event Inciting NOSE'][i]=iOpen

elif dmec0[iStand]['RegenType'][i]==meta['LUT']['Derived']['RegenType']['Road Rehabilitation']:

#--------------------------------------------------------------

# Road rehab

#--------------------------------------------------------------

meta[pNam]['Project']['RegenType'][iStand]=meta['LUT']['Derived']['RegenType']['Road Rehabilitation']

meta[pNam]['Project']['Strata']['Project Type']['ID'][iStand]=meta['LUT']['Derived']['RegenType']['Road Rehabilitation']

dmec0[iStand]['Year'][iOpen]=yr0-10

dmec0[iStand]['ID Event Type'][iOpen]=meta['LUT']['Event']['Forest to Road']

dmec0[iStand]['Mortality Factor'][iOpen]=100

dmec0[iStand]['Index to Event Inciting NOSE'][i]=iOpen

iOpen=iOpen+1

dmec0[iStand]['Year'][iOpen]=yr0-9

dmec0[iStand]['ID Event Type'][iOpen]=meta['LUT']['Event']['Regen Failure']

dmec0[iStand]['Mortality Factor'][iOpen]=100

return meta,dmec0

#%% Process project inputs 2

def Process2\_PrepareGrowthCurves(meta,pNam,lsat,dmec):

#--------------------------------------------------------------------------

# Indicate which scenario is affected by events

# \*\*\* Make sure scenarios have been defined as "baseline" or "actual" \*\*\*

#--------------------------------------------------------------------------

print('Applying exclusion rules to baseline scenario.')

for iScn in range(meta[pNam]['Project']['N Scenario']):

for iStand in range(meta[pNam]['Project']['N Stand']):

# Initialize indicator for each scenario

dmec[iScn][iStand]['Scenario Affected']=np.zeros(dmec[iScn][iStand]['Year'].size,dtype='int16')

if meta[pNam]['Project']['Special Attribution Method']=='Off':

# All events occur in all scenarios

for iT in range(dmec[iScn][iStand]['Year'].size):

dmec[iScn][iStand]['Scenario Affected'][iT]=1

elif meta[pNam]['Project']['Special Attribution Method']=='TDAF':

meta[pNam]['Project']['Activities To Exclude From Baseline']=np.array([meta['LUT']['Event']['Direct Seeding'],

meta['LUT']['Event']['Knockdown'],

meta['LUT']['Event']['Mechanical Site Prep'],

meta['LUT']['Event']['Harvest'],

meta['LUT']['Event']['Harvest Salvage'],

meta['LUT']['Event']['Thinning'],

meta['LUT']['Event']['Aerial BTK Spray'],

meta['LUT']['Event']['Planting'],

meta['LUT']['Event']['Fertilization Aerial'],

meta['LUT']['Event']['Slashpile Burn'],

meta['LUT']['Event']['Prescribed Burn']])

for iT in range(dmec[iScn][iStand]['Year'].size):

if np.isin(iScn,meta[pNam]['Project']['Baseline Indices'])==False:

# All events impact

dmec[iScn][iStand]['Scenario Affected'][iT]=1

elif np.isin(iScn,meta[pNam]['Project']['Baseline Indices'])==True:

if np.isin(dmec[iScn][iStand]['ID Event Type'][iT],meta[pNam]['Project']['Activities To Exclude From Baseline'])==False:

# Events not in the list are added

dmec[iScn][iStand]['Scenario Affected'][iT]=1

else:

pass

elif meta[pNam]['Project']['Special Attribution Method']=='LUC':

meta[pNam]['Project']['Activities To Exclude From Baseline']=np.array([meta['LUT']['Event']['FL-CL'],

meta['LUT']['Event']['FL-PA'],

meta['LUT']['Event']['FL-RC'],

meta['LUT']['Event']['FL-TR'],

meta['LUT']['Event']['FL-EM'],

meta['LUT']['Event']['Slashpile Burn'],

meta['LUT']['Event']['Regen Failure']])

for iT in range(dmec[iScn][iStand]['Year'].size):

if np.isin(iScn,meta[pNam]['Project']['Baseline Indices'])==False:

# All events impact

dmec[iScn][iStand]['Scenario Affected'][iT]=1

elif np.isin(iScn,meta[pNam]['Project']['Baseline Indices'])==True:

if np.isin(dmec[iScn][iStand]['ID Event Type'][iT],meta[pNam]['Project']['Activities To Exclude From Baseline'])==False:

# Events not in the list are added

dmec[iScn][iStand]['Scenario Affected'][iT]=1

else:

pass

elif meta[pNam]['Project']['Special Attribution Method']=='BAU':

meta[pNam]['Project']['Activities To Exclude From Baseline']=np.array([

meta['LUT']['Event']['Direct Seeding'],

meta['LUT']['Event']['Knockdown'],

meta['LUT']['Event']['Mechanical Site Prep'],

meta['LUT']['Event']['Harvest'],

meta['LUT']['Event']['Harvest Salvage'],

meta['LUT']['Event']['Thinning'],

meta['LUT']['Event']['Aerial BTK Spray'],

meta['LUT']['Event']['Planting'],

meta['LUT']['Event']['Fertilization Aerial'],

meta['LUT']['Event']['Slashpile Burn'],

meta['LUT']['Event']['Prescribed Burn']])

meta[pNam]['Project']['Activities To Exclude From Actual']=np.array([

meta['LUT']['Event']['Mechanical Site Prep'],

meta['LUT']['Event']['Thinning'],

meta['LUT']['Event']['Aerial BTK Spray'],

meta['LUT']['Event']['Fertilization Aerial'],

meta['LUT']['Event']['Prescribed Burn']])

for iT in range(dmec[iScn][iStand]['Year'].size):

# Non-obligation status

StatusNO=np.isin(dmec[iScn][iStand]['SILV\_FUND\_SOURCE\_CODE'][iT],meta['Param']['BE']['FSC']['NO List ID'])

if (np.isin(iScn,meta[pNam]['Project']['Actual Indices'])==True) & (np.isin(dmec[iScn][iStand]['ID Event Type'][iT],meta[pNam]['Project']['Activities To Exclude From Actual'])==False) & (StatusNO==False):

# All but excluded events

dmec[iScn][iStand]['Scenario Affected'][iT]=1

elif (np.isin(iScn,meta[pNam]['Project']['Baseline Indices'])==True) & (np.isin(dmec[iScn][iStand]['ID Event Type'][iT],meta[pNam]['Project']['Activities To Exclude From Baseline'])==False):

# Events not in the list are added

dmec[iScn][iStand]['Scenario Affected'][iT]=1

else:

pass

elif meta[pNam]['Project']['Special Attribution Method']=='NOSE':

# Non-obligation stand establishment

# List of activities that will be excluded from baseline

if meta[pNam]['Project']['RegenType'][iStand]==meta['LUT']['Derived']['RegenType']['Harvest and Planting NSR Backlog']:

eL=['Planting','Direct Seeding','Knockdown','Mechanical Site Prep']

else:

eL=['Planting','Direct Seeding','Harvest','Harvest Salvage','Knockdown','Slashpile Burn','Mechanical Site Prep']

meta[pNam]['Project']['Activities To Exclude From Baseline']=np.zeros(len(eL))

for i in range(len(eL)):

meta[pNam]['Project']['Activities To Exclude From Baseline'][i]=meta['LUT']['Event'][eL[i]]

# Index to stand establishment events

#StatusNO=np.isin(dmec[iScn][iStand]['SILV\_FUND\_SOURCE\_CODE'],meta['Param']['BE']['FSC']['NO List ID'])

iEstabNO=np.where( (dmec[iScn][iStand]['Index to Event Inciting NOSE']!=9999) )[0]

#iEstabNO=np.where( (dmec[iScn][iStand]['ID Event Type']==meta['LUT']['Event']['Planting']) & (StatusNO==True) | \

# (dmec[iScn][iStand]['ID Event Type']==meta['LUT']['Event']['Direct Seeding']) & (StatusNO==True) )[0]

if iEstabNO.size>0:

# If multiple planting events, focus on the first instance

iEstabNO=iEstabNO[0]

# Index to inciting event

iIncitingEvent=dmec[iScn][iStand]['Index to Event Inciting NOSE'][iEstabNO]

for iT in range(dmec[iScn][iStand]['Year'].size):

if np.isin(iScn,meta[pNam]['Project']['Baseline Indices'])==True:

if (iT<=iIncitingEvent) | (np.isin(dmec[iScn][iStand]['ID Event Type'][iT],meta[pNam]['Project']['Activities To Exclude From Baseline'])==False):

# Baseline scenario, some events excluded after the inciting event

dmec[iScn][iStand]['Scenario Affected'][iT]=1

else:

# Project scenario, everything occurs in all scenarios

dmec[iScn][iStand]['Scenario Affected'][iT]=1

elif meta[pNam]['Project']['Special Attribution Method']=='Nutrient Management':

# Nutrient management

for iT in range(dmec[iScn][iStand]['Year'].size):

if (np.isin(iScn,meta[pNam]['Project']['Actual Indices'])==True):

dmec[iScn][iStand]['Scenario Affected'][iT]=1

else:

if dmec[iScn][iStand]['ID Event Type'][iT]!=meta['LUT']['Event']['Fertilization Aerial']:

dmec[iScn][iStand]['Scenario Affected'][iT]=1

#--------------------------------------------------------------------------

# Parameterize growth curves

#--------------------------------------------------------------------------

print('Preparing growth curves')

t0=time.time()

# Initialize list

gc=[None]\*meta[pNam]['Project']['N Scenario']

for iScn in range(meta[pNam]['Project']['N Scenario']):

gc[iScn]=[None]\*meta[pNam]['Project']['N Stand']

for iStand in range(meta[pNam]['Project']['N Stand']):

# Index to growth curve

cnt\_gc=0

# Initialize growth curve identifiers in DMEC

dmec[iScn][iStand]['ID\_GC']=1\*np.ones(dmec[iScn][iStand]['Year'].size)

# Initialize growth curve info

gc[iScn][iStand]={}

for key in meta['Modules']['GYM']['GC\_Variable\_List']:

if np.isin(key,['ID\_Stand','ID\_Scn','ID\_GC','s1','p1','s2','p2','s3','p3','s4','p4','s5','p5','s6','p6','init\_density','regen\_delay','regeneration\_method','bec\_zone','FIZ'])==True:

if key=='ID\_Stand':

# For big projects, this needs to be 32 bits

gc[iScn][iStand][key]=9999\*np.ones(12,dtype='int32')

else:

gc[iScn][iStand][key]=9999\*np.ones(12,dtype='int16')

else:

gc[iScn][iStand][key]=9999\*np.ones(12)

if lsat['Region Code'][iStand]==meta['LUT']['Region']['Coast']:

fiz=meta['LUT']['TIPSY']['FIZ']['C']

else:

fiz=meta['LUT']['TIPSY']['FIZ']['I']

#--------------------------------------------------------------------------

# Add pre-contact growth curve

#--------------------------------------------------------------------------

gc[iScn][iStand]['ID\_Stand'][cnt\_gc]=iStand

gc[iScn][iStand]['ID\_Scn'][cnt\_gc]=iScn

gc[iScn][iStand]['ID\_GC'][cnt\_gc]=1

gc[iScn][iStand]['regeneration\_method'][cnt\_gc]=meta['LUT']['TIPSY']['regeneration\_method']['N']

gc[iScn][iStand]['s1'][cnt\_gc]=lsat['Spc1\_ID'][iStand]

gc[iScn][iStand]['p1'][cnt\_gc]=lsat['Spc1\_P'][iStand]

gc[iScn][iStand]['i1'][cnt\_gc]=lsat['SI'][iStand]

gc[iScn][iStand]['s2'][cnt\_gc]=lsat['Spc2\_ID'][iStand]

gc[iScn][iStand]['p2'][cnt\_gc]=lsat['Spc2\_P'][iStand]

gc[iScn][iStand]['s3'][cnt\_gc]=lsat['Spc3\_ID'][iStand]

gc[iScn][iStand]['p3'][cnt\_gc]=lsat['Spc3\_P'][iStand]

gc[iScn][iStand]['s4'][cnt\_gc]=lsat['Spc4\_ID'][iStand]

gc[iScn][iStand]['p4'][cnt\_gc]=lsat['Spc4\_P'][iStand]

gc[iScn][iStand]['s5'][cnt\_gc]=lsat['Spc5\_ID'][iStand]

gc[iScn][iStand]['p5'][cnt\_gc]=lsat['Spc5\_P'][iStand]

gc[iScn][iStand]['init\_density'][cnt\_gc]=lsat['SPH Init Natural'][iStand]

gc[iScn][iStand]['regen\_delay'][cnt\_gc]=lsat['Regen Delay Natural'][iStand]

gc[iScn][iStand]['oaf1'][cnt\_gc]=lsat['OAF1'][iStand].astype(float)/100

gc[iScn][iStand]['oaf2'][cnt\_gc]=meta['Modules']['GYM']['OAF2 Default']

gc[iScn][iStand]['bec\_zone'][cnt\_gc]=lsat['ID\_BGCZ'][iStand]

gc[iScn][iStand]['FIZ'][cnt\_gc]=fiz

cnt\_gc=cnt\_gc+1

# Index to existing GC IDs

iExistingGC=np.where(gc[iScn][iStand]['ID\_GC']<9999)[0]

#--------------------------------------------------------------------------

# Add events from disturbance/management event history

#--------------------------------------------------------------------------

for iYr in range(dmec[iScn][iStand]['Year'].size):

# Index to existing GC IDs

iExistingGC=np.where(gc[iScn][iStand]['ID\_GC']<9999)[0]

# Calculate planting density

PlantingDensity=int(dmec[iScn][iStand]['Planted SPH'][iYr])

if lsat['Region Code'][iStand]==meta['LUT']['Region']['Interior']:

PlantingDensity=np.minimum(2400,np.maximum(1400,PlantingDensity))

else:

PlantingDensity=np.minimum(2400,np.maximum(900,PlantingDensity))

PlantingDensity=int(PlantingDensity)

# Create a flag that indicates whether there are back-to-back planting

# Back to back planting I think occurs in some cases because they go back

# and add a bit. You can tell by looking at the treatment area - if the

# second planting treatment area is tiny compared to the first, you could

# ignore it I guess. Certainly not ideal, but I don't see a work around.

# We are going to ignore the second planting for now.

Flag\_PlantingBackToBack=0

if iYr>0:

if (dmec[iScn][iStand]['ID Event Type'][iYr]==meta['LUT']['Event']['Planting']):

Yr=dmec[iScn][iStand]['Year'][iYr]

indPrevPL=np.where( (dmec[iScn][iStand]['Year']==Yr-1) & (dmec[iScn][iStand]['ID Event Type']==meta['LUT']['Event']['Planting']) )[0]

if indPrevPL.size>0:

Flag\_PlantingBackToBack=1

# Index to previous disturbance for fertilization

#IndPrevDistForFert=int(dmec[iStand]['IndPrevDistForFert'][iYr])

# Non-obligation status

StatusNO=np.isin(dmec[iScn][iStand]['SILV\_FUND\_SOURCE\_CODE'][iYr],meta['Param']['BE']['FSC']['NO List ID'])

if (meta[pNam]['Project']['Special Attribution Method']=='Off') | (meta[pNam]['Project']['Special Attribution Method']=='TDAF') | (meta[pNam]['Project']['Special Attribution Method']=='Nutrient Management'):

#--------------------------------------------------------------

# Off or TDAF or Nutrient Management

#--------------------------------------------------------------

if (dmec[iScn][iStand]['Scenario Affected'][iYr]==1) & (dmec[iScn][iStand]['ID Event Type'][iYr]==meta['LUT']['Event']['Planting']):

dmec[iScn][iStand]['ID\_GC'][iYr:]=np.max(gc[iScn][iStand]['ID\_GC'][iExistingGC])+1

gc[iScn][iStand]['ID\_Stand'][cnt\_gc]=iStand

gc[iScn][iStand]['ID\_Scn'][cnt\_gc]=iScn

gc[iScn][iStand]['ID\_GC'][cnt\_gc]=dmec[iScn][iStand]['ID\_GC'][iYr]

gc[iScn][iStand]['regeneration\_method'][cnt\_gc]=meta['LUT']['TIPSY']['regeneration\_method']['P']

gc[iScn][iStand]['init\_density'][cnt\_gc]=PlantingDensity

gc[iScn][iStand]['regen\_delay'][cnt\_gc]=0

gc[iScn][iStand]['i1'][cnt\_gc]=lsat['SI'][iStand]

if (dmec[iScn][iStand]['PL\_SPECIES\_CD1'][iYr]!=9999) & (dmec[iScn][iStand]['PL\_SPECIES\_CD1'][iYr]!=0) & (dmec[iScn][iStand]['PL\_SPECIES\_PCT1'][iYr]!=9999):

# \*\*\* Adjust site index if it is energy production \*\*\*

if (dmec[iScn][iStand]['PL\_SPECIES\_CD1'][iYr]==meta['LUT']['VEG\_COMP\_LYR\_R1\_POLY']['SPECIES\_CD\_1']['AT']):

gc[iScn][iStand]['i1'][cnt\_gc]=30

gc[iScn][iStand]['init\_density'][cnt\_gc]=2000

# Using planting info if it exists

gc[iScn][iStand]['s1'][cnt\_gc]=dmec[iScn][iStand]['PL\_SPECIES\_CD1'][iYr]

gc[iScn][iStand]['p1'][cnt\_gc]=dmec[iScn][iStand]['PL\_SPECIES\_PCT1'][iYr]

gc[iScn][iStand]['gain1'][cnt\_gc]=dmec[iScn][iStand]['PL\_SPECIES\_GW1'][iYr]

gc[iScn][iStand]['selage1'][cnt\_gc]=10

gc[iScn][iStand]['s2'][cnt\_gc]=dmec[iScn][iStand]['PL\_SPECIES\_CD2'][iYr]

gc[iScn][iStand]['p2'][cnt\_gc]=dmec[iScn][iStand]['PL\_SPECIES\_PCT2'][iYr]

gc[iScn][iStand]['gain2'][cnt\_gc]=dmec[iScn][iStand]['PL\_SPECIES\_GW2'][iYr]

gc[iScn][iStand]['selage2'][cnt\_gc]=10

gc[iScn][iStand]['s3'][cnt\_gc]=dmec[iScn][iStand]['PL\_SPECIES\_CD3'][iYr]

gc[iScn][iStand]['p3'][cnt\_gc]=dmec[iScn][iStand]['PL\_SPECIES\_PCT3'][iYr]

gc[iScn][iStand]['gain3'][cnt\_gc]=dmec[iScn][iStand]['PL\_SPECIES\_GW3'][iYr]

gc[iScn][iStand]['selage3'][cnt\_gc]=10

gc[iScn][iStand]['s4'][cnt\_gc]=dmec[iScn][iStand]['PL\_SPECIES\_CD4'][iYr]

gc[iScn][iStand]['p4'][cnt\_gc]=dmec[iScn][iStand]['PL\_SPECIES\_PCT4'][iYr]

gc[iScn][iStand]['gain4'][cnt\_gc]=dmec[iScn][iStand]['PL\_SPECIES\_GW4'][iYr]

gc[iScn][iStand]['selage4'][cnt\_gc]=10

gc[iScn][iStand]['s5'][cnt\_gc]=dmec[iScn][iStand]['PL\_SPECIES\_CD5'][iYr]

gc[iScn][iStand]['p5'][cnt\_gc]=dmec[iScn][iStand]['PL\_SPECIES\_PCT5'][iYr]

gc[iScn][iStand]['gain5'][cnt\_gc]=dmec[iScn][iStand]['PL\_SPECIES\_GW5'][iYr]

gc[iScn][iStand]['selage5'][cnt\_gc]=10

else:

# Otherwise assume best-available inventory spc. comp.

gc[iScn][iStand]['s1'][cnt\_gc]=lsat['Spc1\_ID'][iStand]

gc[iScn][iStand]['p1'][cnt\_gc]=lsat['Spc1\_P'][iStand]

gc[iScn][iStand]['s2'][cnt\_gc]=lsat['Spc2\_ID'][iStand]

gc[iScn][iStand]['p2'][cnt\_gc]=lsat['Spc2\_P'][iStand]

gc[iScn][iStand]['s3'][cnt\_gc]=lsat['Spc3\_ID'][iStand]

gc[iScn][iStand]['p3'][cnt\_gc]=lsat['Spc3\_P'][iStand]

gc[iScn][iStand]['s4'][cnt\_gc]=lsat['Spc4\_ID'][iStand]

gc[iScn][iStand]['p4'][cnt\_gc]=lsat['Spc4\_P'][iStand]

gc[iScn][iStand]['s5'][cnt\_gc]=lsat['Spc5\_ID'][iStand]

gc[iScn][iStand]['p5'][cnt\_gc]=lsat['Spc5\_P'][iStand]

gc[iScn][iStand]['oaf1'][cnt\_gc]=lsat['OAF1'][iStand].astype(float)/100

gc[iScn][iStand]['oaf2'][cnt\_gc]=meta['Modules']['GYM']['OAF2 Default']

gc[iScn][iStand]['bec\_zone'][cnt\_gc]=lsat['ID\_BGCZ'][iStand]

gc[iScn][iStand]['FIZ'][cnt\_gc]=fiz

# Update counter

cnt\_gc=cnt\_gc+1

elif (meta[pNam]['Project']['Special Attribution Method']=='BAU'):

#--------------------------------------------------------------

# Harvesting

# Exclude incremental improvements in silviculture, including:

# - genetic gains

#--------------------------------------------------------------

if (dmec[iScn][iStand]['Scenario Affected'][iYr]==1) & (dmec[iScn][iStand]['ID Event Type'][iYr]==meta['LUT']['Event']['Planting']):

dmec[iScn][iStand]['ID\_GC'][iYr:]=np.max(gc[iScn][iStand]['ID\_GC'][iExistingGC])+1

gc[iScn][iStand]['ID\_Stand'][cnt\_gc]=iStand

gc[iScn][iStand]['ID\_Scn'][cnt\_gc]=iScn

gc[iScn][iStand]['ID\_GC'][cnt\_gc]=dmec[iScn][iStand]['ID\_GC'][iYr]

gc[iScn][iStand]['regeneration\_method'][cnt\_gc]=meta['LUT']['TIPSY']['regeneration\_method']['P']

gc[iScn][iStand]['init\_density'][cnt\_gc]=PlantingDensity

gc[iScn][iStand]['regen\_delay'][cnt\_gc]=0

gc[iScn][iStand]['i1'][cnt\_gc]=lsat['SI'][iStand]

if (dmec[iScn][iStand]['PL\_SPECIES\_CD1'][iYr]!=9999) & (dmec[iScn][iStand]['PL\_SPECIES\_CD1'][iYr]!=0) & (dmec[iScn][iStand]['PL\_SPECIES\_PCT1'][iYr]!=9999):

# Using planting info if it exists

gc[iScn][iStand]['s1'][cnt\_gc]=dmec[iScn][iStand]['PL\_SPECIES\_CD1'][iYr]

gc[iScn][iStand]['p1'][cnt\_gc]=dmec[iScn][iStand]['PL\_SPECIES\_PCT1'][iYr]

gc[iScn][iStand]['s2'][cnt\_gc]=dmec[iScn][iStand]['PL\_SPECIES\_CD2'][iYr]

gc[iScn][iStand]['p2'][cnt\_gc]=dmec[iScn][iStand]['PL\_SPECIES\_PCT2'][iYr]

gc[iScn][iStand]['s3'][cnt\_gc]=dmec[iScn][iStand]['PL\_SPECIES\_CD3'][iYr]

gc[iScn][iStand]['p3'][cnt\_gc]=dmec[iScn][iStand]['PL\_SPECIES\_PCT3'][iYr]

gc[iScn][iStand]['s4'][cnt\_gc]=dmec[iScn][iStand]['PL\_SPECIES\_CD4'][iYr]

gc[iScn][iStand]['p4'][cnt\_gc]=dmec[iScn][iStand]['PL\_SPECIES\_PCT4'][iYr]

gc[iScn][iStand]['s5'][cnt\_gc]=dmec[iScn][iStand]['PL\_SPECIES\_CD5'][iYr]

gc[iScn][iStand]['p5'][cnt\_gc]=dmec[iScn][iStand]['PL\_SPECIES\_PCT5'][iYr]

else:

# Otherwise assume best-available inventory spc. comp.

gc[iScn][iStand]['s1'][cnt\_gc]=lsat['Spc1\_ID'][iStand]

gc[iScn][iStand]['p1'][cnt\_gc]=lsat['Spc1\_P'][iStand]

gc[iScn][iStand]['s2'][cnt\_gc]=lsat['Spc2\_ID'][iStand]

gc[iScn][iStand]['p2'][cnt\_gc]=lsat['Spc2\_P'][iStand]

gc[iScn][iStand]['s3'][cnt\_gc]=lsat['Spc3\_ID'][iStand]

gc[iScn][iStand]['p3'][cnt\_gc]=lsat['Spc3\_P'][iStand]

gc[iScn][iStand]['s4'][cnt\_gc]=lsat['Spc4\_ID'][iStand]

gc[iScn][iStand]['p4'][cnt\_gc]=lsat['Spc4\_P'][iStand]

gc[iScn][iStand]['s5'][cnt\_gc]=lsat['Spc5\_ID'][iStand]

gc[iScn][iStand]['p5'][cnt\_gc]=lsat['Spc5\_P'][iStand]

gc[iScn][iStand]['oaf1'][cnt\_gc]=lsat['OAF1'][iStand].astype(float)/100

gc[iScn][iStand]['oaf2'][cnt\_gc]=meta['Modules']['GYM']['OAF2 Default']

gc[iScn][iStand]['bec\_zone'][cnt\_gc]=lsat['ID\_BGCZ'][iStand]

gc[iScn][iStand]['FIZ'][cnt\_gc]=fiz

# Update counter

cnt\_gc=cnt\_gc+1

elif (meta[pNam]['Project']['Special Attribution Method']=='NOSE'):

#--------------------------------------------------------------

# Non-obligation stand establishment

#--------------------------------------------------------------

# Index to event that incited NO stand establishment

iIncitingNOSE=dmec[iScn][iStand]['Index to Event Inciting NOSE'][iYr]

#----------------------------------------------------------------------

# Planting (non-obligation)

#----------------------------------------------------------------------

if (dmec[iScn][iStand]['ID Event Type'][iYr]==meta['LUT']['Event']['Planting']) & (StatusNO==True) & (iIncitingNOSE!=9999):

gc[iScn][iStand]['ID\_Stand'][cnt\_gc]=iStand

gc[iScn][iStand]['ID\_Scn'][cnt\_gc]=iScn

gc[iScn][iStand]['i1'][cnt\_gc]=lsat['SI'][iStand]

gc[iScn][iStand]['oaf1'][cnt\_gc]=meta['Modules']['GYM']['OAF1 Default']

gc[iScn][iStand]['oaf2'][cnt\_gc]=meta['Modules']['GYM']['OAF2 Default']

gc[iScn][iStand]['bec\_zone'][cnt\_gc]=lsat['ID\_BGCZ'][iStand]

gc[iScn][iStand]['FIZ'][cnt\_gc]=fiz

if np.isin(iScn,meta[pNam]['Project']['Baseline Indices'])==True:

# Baseline scenarios:

# Growth curve update (at the time of inciting event)

dmec[iScn][iStand]['ID\_GC'][iIncitingNOSE:]=dmec[iScn][iStand]['ID\_GC'][iIncitingNOSE]+1

gc[iScn][iStand]['ID\_GC'][cnt\_gc]=dmec[iScn][iStand]['ID\_GC'][iYr]

gc[iScn][iStand]['regeneration\_method'][cnt\_gc]=meta['LUT']['TIPSY']['regeneration\_method']['N']

if meta[pNam]['Project']['RegenType'][iStand]==meta['LUT']['Derived']['RegenType']['Straight-to-planting Post Wildfire']:

gc[iScn][iStand]['init\_density'][cnt\_gc]=200

gc[iScn][iStand]['regen\_delay'][cnt\_gc]=5

elif meta[pNam]['Project']['RegenType'][iStand]==meta['LUT']['Derived']['RegenType']['Straight-to-planting Post Beetles']:

gc[iScn][iStand]['init\_density'][cnt\_gc]=500

gc[iScn][iStand]['regen\_delay'][cnt\_gc]=5

elif meta[pNam]['Project']['RegenType'][iStand]==meta['LUT']['Derived']['RegenType']['Salvage and Planting']:

gc[iScn][iStand]['init\_density'][cnt\_gc]=1400

gc[iScn][iStand]['regen\_delay'][cnt\_gc]=2

elif meta[pNam]['Project']['RegenType'][iStand]==meta['LUT']['Derived']['RegenType']['Knockdown and Planting']:

gc[iScn][iStand]['init\_density'][cnt\_gc]=1400

gc[iScn][iStand]['regen\_delay'][cnt\_gc]=0

elif meta[pNam]['Project']['RegenType'][iStand]==meta['LUT']['Derived']['RegenType']['Harvest and Planting NSR Backlog']:

gc[iScn][iStand]['init\_density'][cnt\_gc]=500

gc[iScn][iStand]['regen\_delay'][cnt\_gc]=5

elif meta[pNam]['Project']['RegenType'][iStand]==meta['LUT']['Derived']['RegenType']['Replanting']:

gc[iScn][iStand]['init\_density'][cnt\_gc]=500

gc[iScn][iStand]['regen\_delay'][cnt\_gc]=5

gc[iScn][iStand]['s1'][cnt\_gc]=lsat['Spc1\_ID'][iStand]

gc[iScn][iStand]['p1'][cnt\_gc]=lsat['Spc1\_P'][iStand]

gc[iScn][iStand]['s2'][cnt\_gc]=lsat['Spc2\_ID'][iStand]

gc[iScn][iStand]['p2'][cnt\_gc]=lsat['Spc2\_P'][iStand]

gc[iScn][iStand]['s3'][cnt\_gc]=lsat['Spc3\_ID'][iStand]

gc[iScn][iStand]['p3'][cnt\_gc]=lsat['Spc3\_P'][iStand]

gc[iScn][iStand]['s4'][cnt\_gc]=lsat['Spc4\_ID'][iStand]

gc[iScn][iStand]['p4'][cnt\_gc]=lsat['Spc4\_P'][iStand]

gc[iScn][iStand]['s5'][cnt\_gc]=lsat['Spc5\_ID'][iStand]

gc[iScn][iStand]['p5'][cnt\_gc]=lsat['Spc5\_P'][iStand]

else:

# Project scenarios:

# Growth curve update: Project scenarios with planting at iYr

dmec[iScn][iStand]['ID\_GC'][iYr:]=np.max(gc[iScn][iStand]['ID\_GC'][iExistingGC])+1

gc[iScn][iStand]['ID\_GC'][cnt\_gc]=dmec[iScn][iStand]['ID\_GC'][iYr]

gc[iScn][iStand]['regeneration\_method'][cnt\_gc]=meta['LUT']['TIPSY']['regeneration\_method']['P']

gc[iScn][iStand]['init\_density'][cnt\_gc]=PlantingDensity

gc[iScn][iStand]['regen\_delay'][cnt\_gc]=0

if (dmec[iScn][iStand]['PL\_SPECIES\_CD1'][iYr]!=9999) & (dmec[iScn][iStand]['PL\_SPECIES\_CD1'][iYr]!=0) & (dmec[iScn][iStand]['PL\_SPECIES\_PCT1'][iYr]!=9999):

# Using planting layer information

gc[iScn][iStand]['s1'][cnt\_gc]=dmec[iScn][iStand]['PL\_SPECIES\_CD1'][iYr]

gc[iScn][iStand]['p1'][cnt\_gc]=dmec[iScn][iStand]['PL\_SPECIES\_PCT1'][iYr]

gc[iScn][iStand]['gain1'][cnt\_gc]=int(dmec[iScn][iStand]['PL\_SPECIES\_GW1'][iYr])

gc[iScn][iStand]['selage1'][cnt\_gc]=10

gc[iScn][iStand]['s2'][cnt\_gc]=dmec[iScn][iStand]['PL\_SPECIES\_CD2'][iYr]

gc[iScn][iStand]['p2'][cnt\_gc]=dmec[iScn][iStand]['PL\_SPECIES\_PCT2'][iYr]

gc[iScn][iStand]['gain2'][cnt\_gc]=int(dmec[iScn][iStand]['PL\_SPECIES\_GW2'][iYr])

gc[iScn][iStand]['selage2'][cnt\_gc]=10

gc[iScn][iStand]['s3'][cnt\_gc]=dmec[iScn][iStand]['PL\_SPECIES\_CD3'][iYr]

gc[iScn][iStand]['p3'][cnt\_gc]=dmec[iScn][iStand]['PL\_SPECIES\_PCT3'][iYr]

gc[iScn][iStand]['gain3'][cnt\_gc]=int(dmec[iScn][iStand]['PL\_SPECIES\_GW3'][iYr])

gc[iScn][iStand]['selage3'][cnt\_gc]=10

gc[iScn][iStand]['s4'][cnt\_gc]=dmec[iScn][iStand]['PL\_SPECIES\_CD4'][iYr]

gc[iScn][iStand]['p4'][cnt\_gc]=dmec[iScn][iStand]['PL\_SPECIES\_PCT4'][iYr]

gc[iScn][iStand]['gain4'][cnt\_gc]=int(dmec[iScn][iStand]['PL\_SPECIES\_GW4'][iYr])

gc[iScn][iStand]['selage4'][cnt\_gc]=10

gc[iScn][iStand]['s5'][cnt\_gc]=dmec[iScn][iStand]['PL\_SPECIES\_CD5'][iYr]

gc[iScn][iStand]['p5'][cnt\_gc]=dmec[iScn][iStand]['PL\_SPECIES\_PCT5'][iYr]

gc[iScn][iStand]['gain5'][cnt\_gc]=int(dmec[iScn][iStand]['PL\_SPECIES\_GW5'][iYr])

gc[iScn][iStand]['selage5'][cnt\_gc]=10

else:

# Planting info not available, using inventory

gc[iScn][iStand]['s1'][cnt\_gc]=lsat['Spc1\_ID'][iStand]

gc[iScn][iStand]['p1'][cnt\_gc]=lsat['Spc1\_P'][iStand]

gc[iScn][iStand]['s2'][cnt\_gc]=lsat['Spc2\_ID'][iStand]

gc[iScn][iStand]['p2'][cnt\_gc]=lsat['Spc2\_P'][iStand]

gc[iScn][iStand]['s3'][cnt\_gc]=lsat['Spc3\_ID'][iStand]

gc[iScn][iStand]['p3'][cnt\_gc]=lsat['Spc3\_P'][iStand]

gc[iScn][iStand]['s4'][cnt\_gc]=lsat['Spc4\_ID'][iStand]

gc[iScn][iStand]['p4'][cnt\_gc]=lsat['Spc4\_P'][iStand]

gc[iScn][iStand]['s5'][cnt\_gc]=lsat['Spc5\_ID'][iStand]

gc[iScn][iStand]['p5'][cnt\_gc]=lsat['Spc5\_P'][iStand]

# Update counter

cnt\_gc=cnt\_gc+1

#----------------------------------------------------------------------

# Planting (obligation)

#----------------------------------------------------------------------

if (dmec[iScn][iStand]['Scenario Affected'][iYr]==1) & (dmec[iScn][iStand]['ID Event Type'][iYr]==meta['LUT']['Event']['Planting']) & (Flag\_PlantingBackToBack==0) & (StatusNO==False):

dmec[iScn][iStand]['ID\_GC'][iYr:]=np.max(gc[iScn][iStand]['ID\_GC'][iExistingGC])+1

gc[iScn][iStand]['ID\_Stand'][cnt\_gc]=iStand

gc[iScn][iStand]['ID\_Scn'][cnt\_gc]=iScn

gc[iScn][iStand]['ID\_GC'][cnt\_gc]=dmec[iScn][iStand]['ID\_GC'][iYr]

gc[iScn][iStand]['regeneration\_method'][cnt\_gc]=meta['LUT']['TIPSY']['regeneration\_method']['P']

gc[iScn][iStand]['i1'][cnt\_gc]=lsat['SI'][iStand]

gc[iScn][iStand]['init\_density'][cnt\_gc]=PlantingDensity

gc[iScn][iStand]['regen\_delay'][cnt\_gc]=0

gc[iScn][iStand]['oaf1'][cnt\_gc]=meta['Modules']['GYM']['OAF1 Default']

gc[iScn][iStand]['oaf2'][cnt\_gc]=meta['Modules']['GYM']['OAF2 Default']

gc[iScn][iStand]['bec\_zone'][cnt\_gc]=lsat['ID\_BGCZ'][iStand]

gc[iScn][iStand]['FIZ'][cnt\_gc]=fiz

if (dmec[iScn][iStand]['PL\_SPECIES\_CD1'][iYr]!=9999) & (dmec[iScn][iStand]['PL\_SPECIES\_CD1'][iYr]!=0) & (dmec[iScn][iStand]['PL\_SPECIES\_PCT1'][iYr]!=9999):

gc[iScn][iStand]['s1'][cnt\_gc]=dmec[iScn][iStand]['PL\_SPECIES\_CD1'][iYr]

gc[iScn][iStand]['p1'][cnt\_gc]=dmec[iScn][iStand]['PL\_SPECIES\_PCT1'][iYr]

gc[iScn][iStand]['gain1'][cnt\_gc]=dmec[iScn][iStand]['PL\_SPECIES\_GW1'][iYr]

gc[iScn][iStand]['selage1'][cnt\_gc]=10

gc[iScn][iStand]['s2'][cnt\_gc]=dmec[iScn][iStand]['PL\_SPECIES\_CD2'][iYr]

gc[iScn][iStand]['p2'][cnt\_gc]=dmec[iScn][iStand]['PL\_SPECIES\_PCT2'][iYr]

gc[iScn][iStand]['gain2'][cnt\_gc]=dmec[iScn][iStand]['PL\_SPECIES\_GW2'][iYr]

gc[iScn][iStand]['selage2'][cnt\_gc]=10

gc[iScn][iStand]['s3'][cnt\_gc]=dmec[iScn][iStand]['PL\_SPECIES\_CD3'][iYr]

gc[iScn][iStand]['p3'][cnt\_gc]=dmec[iScn][iStand]['PL\_SPECIES\_PCT3'][iYr]

gc[iScn][iStand]['gain3'][cnt\_gc]=dmec[iScn][iStand]['PL\_SPECIES\_GW3'][iYr]

gc[iScn][iStand]['selage3'][cnt\_gc]=10

gc[iScn][iStand]['s4'][cnt\_gc]=dmec[iScn][iStand]['PL\_SPECIES\_CD4'][iYr]

gc[iScn][iStand]['p4'][cnt\_gc]=dmec[iScn][iStand]['PL\_SPECIES\_PCT4'][iYr]

gc[iScn][iStand]['gain4'][cnt\_gc]=dmec[iScn][iStand]['PL\_SPECIES\_GW4'][iYr]

gc[iScn][iStand]['selage4'][cnt\_gc]=10

gc[iScn][iStand]['s5'][cnt\_gc]=dmec[iScn][iStand]['PL\_SPECIES\_CD5'][iYr]

gc[iScn][iStand]['p5'][cnt\_gc]=dmec[iScn][iStand]['PL\_SPECIES\_PCT5'][iYr]

gc[iScn][iStand]['gain5'][cnt\_gc]=dmec[iScn][iStand]['PL\_SPECIES\_GW5'][iYr]

gc[iScn][iStand]['selage5'][cnt\_gc]=10

else:

gc[iScn][iStand]['s1'][cnt\_gc]=lsat['Spc1\_ID'][iStand]

gc[iScn][iStand]['p1'][cnt\_gc]=lsat['Spc1\_P'][iStand]

gc[iScn][iStand]['s2'][cnt\_gc]=lsat['Spc2\_ID'][iStand]

gc[iScn][iStand]['p2'][cnt\_gc]=lsat['Spc2\_P'][iStand]

gc[iScn][iStand]['s3'][cnt\_gc]=lsat['Spc3\_ID'][iStand]

gc[iScn][iStand]['p3'][cnt\_gc]=lsat['Spc3\_P'][iStand]

gc[iScn][iStand]['s4'][cnt\_gc]=lsat['Spc4\_ID'][iStand]

gc[iScn][iStand]['p4'][cnt\_gc]=lsat['Spc4\_P'][iStand]

gc[iScn][iStand]['s5'][cnt\_gc]=lsat['Spc5\_ID'][iStand]

gc[iScn][iStand]['p5'][cnt\_gc]=lsat['Spc5\_P'][iStand]

# Update counter

cnt\_gc=cnt\_gc+1

# Add a planted stand if missing:

# Some stands may have no recorded historical management, but then they

# may be harvested on-the-fly. Add a second growth curve for the on-the-fly

# planting.

if cnt\_gc==1:

gc[iScn][iStand]['ID\_Stand'][cnt\_gc]=iStand

gc[iScn][iStand]['ID\_Scn'][cnt\_gc]=iScn

gc[iScn][iStand]['ID\_GC'][cnt\_gc]=np.max(gc[iScn][iStand]['ID\_GC'][iExistingGC])+1

gc[iScn][iStand]['regeneration\_method'][cnt\_gc]=meta['LUT']['TIPSY']['regeneration\_method']['P']

gc[iScn][iStand]['i1'][cnt\_gc]=lsat['SI'][iStand]

gc[iScn][iStand]['init\_density'][cnt\_gc]=int(1500)

gc[iScn][iStand]['regen\_delay'][cnt\_gc]=0

gc[iScn][iStand]['oaf1'][cnt\_gc]=meta['Modules']['GYM']['OAF1 Default']

gc[iScn][iStand]['oaf2'][cnt\_gc]=meta['Modules']['GYM']['OAF2 Default']

gc[iScn][iStand]['bec\_zone'][cnt\_gc]=lsat['ID\_BGCZ'][iStand]

gc[iScn][iStand]['FIZ'][cnt\_gc]=fiz

# The natural species composition may not be realistic. Use regional

# default planting composition

gain=15

if (lsat['ID\_BGCZ'][iStand]==meta['LUT']['BEC\_BIOGEOCLIMATIC\_POLY']['ZONE']['CWH']) | (lsat['ID\_BGCZ'][iStand]==meta['LUT']['BEC\_BIOGEOCLIMATIC\_POLY']['ZONE']['CDF']):

# Coastal

gc[iScn][iStand]['s1'][cnt\_gc]=meta['LUT']['VEG\_COMP\_LYR\_R1\_POLY']['SPECIES\_CD\_1']['FDC']

gc[iScn][iStand]['p1'][cnt\_gc]=70

gc[iScn][iStand]['gain1'][cnt\_gc]=gain

gc[iScn][iStand]['selage1'][cnt\_gc]=10

gc[iScn][iStand]['s2'][cnt\_gc]=meta['LUT']['VEG\_COMP\_LYR\_R1\_POLY']['SPECIES\_CD\_1']['CW']

gc[iScn][iStand]['p2'][cnt\_gc]=15

gc[iScn][iStand]['gain2'][cnt\_gc]=gain

gc[iScn][iStand]['selage2'][cnt\_gc]=10

gc[iScn][iStand]['s3'][cnt\_gc]=meta['LUT']['VEG\_COMP\_LYR\_R1\_POLY']['SPECIES\_CD\_1']['BA']

gc[iScn][iStand]['p3'][cnt\_gc]=15

gc[iScn][iStand]['gain3'][cnt\_gc]=gain

gc[iScn][iStand]['selage3'][cnt\_gc]=10

elif (lsat['ID\_BGCZ'][iStand]==meta['LUT']['BEC\_BIOGEOCLIMATIC\_POLY']['ZONE']['ICH']):

# Interior wetbelt

gc[iScn][iStand]['s1'][cnt\_gc]=meta['LUT']['VEG\_COMP\_LYR\_R1\_POLY']['SPECIES\_CD\_1']['CW']

gc[iScn][iStand]['p1'][cnt\_gc]=70

gc[iScn][iStand]['gain1'][cnt\_gc]=gain

gc[iScn][iStand]['selage1'][cnt\_gc]=10

gc[iScn][iStand]['s2'][cnt\_gc]=meta['LUT']['VEG\_COMP\_LYR\_R1\_POLY']['SPECIES\_CD\_1']['HW']

gc[iScn][iStand]['p2'][cnt\_gc]=15

gc[iScn][iStand]['gain2'][cnt\_gc]=gain

gc[iScn][iStand]['selage2'][cnt\_gc]=10

else:

# Interior

gc[iScn][iStand]['s1'][cnt\_gc]=meta['LUT']['VEG\_COMP\_LYR\_R1\_POLY']['SPECIES\_CD\_1']['PL']

gc[iScn][iStand]['p1'][cnt\_gc]=60

gc[iScn][iStand]['gain1'][cnt\_gc]=gain

gc[iScn][iStand]['selage1'][cnt\_gc]=10

gc[iScn][iStand]['s2'][cnt\_gc]=meta['LUT']['VEG\_COMP\_LYR\_R1\_POLY']['SPECIES\_CD\_1']['SW']

gc[iScn][iStand]['p2'][cnt\_gc]=40

gc[iScn][iStand]['gain2'][cnt\_gc]=gain

gc[iScn][iStand]['selage2'][cnt\_gc]=10

if gc[iScn][iStand]['s1'][cnt\_gc]==0:

print(iScn)

print(iStand)

print(iYr)

# Get rid of rows with no info

for iScn in range(meta[pNam]['Project']['N Scenario']):

for iStand in range(meta[pNam]['Project']['N Stand']):

# Don't use ID\_Stand, because there could be a legit ID\_Stand=9999

ind=np.where(gc[iScn][iStand]['ID\_GC']!=9999)[0]

for key in meta['Modules']['GYM']['GC\_Variable\_List']:

gc[iScn][iStand][key]=gc[iScn][iStand][key][ind]

print(str(np.round((time.time()-t0)/60,decimals=1)) + ' min')

#--------------------------------------------------------------------------

# Adjust mortality factors that only affect specific tree species

#--------------------------------------------------------------------------

# if meta[pNam]['Project']['Adjust species-specific mortality']=='On':

# print('Adjusting mortality based on species-specific pests')

# t0=time.time()

# dmec=AdjustSpeciesSpecificMortality(meta,pNam,dmec,gc,meta[pNam]['Project']['Actual Indices'][0])

# print(str(np.round((time.time()-t0)/60,decimals=1)) + ' min')

#--------------------------------------------------------------------------

# Extract a set of unique growth curves

# Decompose the full set of stands into a subset of unique stand types.

# Exclude the first three columns, as they are all different.

#--------------------------------------------------------------------------

print('Extracting unique growth curves')

t0=time.time()

ugc=ExtractUniqueGrowthCurves(meta,pNam,gc)

print(str(np.round((time.time()-t0)/60,decimals=1)) + ' min')

#--------------------------------------------------------------------------

# Export to BatchTIPSY parameter spreadsheet

#--------------------------------------------------------------------------

print('Exporting BatchTIPSY parameters to spreadsheet')

t0=time.time()

cbu.Write\_BatchTIPSY\_Input\_Spreadsheet(meta,pNam,ugc)

print(str(np.round((time.time()-t0)/60,decimals=1)) + ' min')

#--------------------------------------------------------------------------

# Populate BatchTIPSY.exe input variable (.dat) file

#--------------------------------------------------------------------------

print('Creating BatchTIPSY.exe input varialbe (.dat) file')

t0=time.time()

cbu.Write\_BatchTIPSY\_Input\_File(meta,pNam)

print(str(np.round((time.time()-t0)/60,decimals=1)) + ' min')

return gc,ugc,lsat,dmec

#%% Process project inputs 3

def Process3\_PrepInputsByBatch(meta,pNam,lsat,dmec,gc,ugc):

#--------------------------------------------------------------------------

# Prepare inventory

#--------------------------------------------------------------------------

print('Preparing inventory input files')

t0=time.time()

for iScn in range(meta[pNam]['Project']['N Scenario']):

# Loop through batches, saving inventory to file

for iBat in range(meta[pNam]['Project']['N Batch']):

# Index to batch

indBat=cbu.IndexToBatch(meta[pNam],iBat)

N\_StandsInBatch=len(indBat)

# Initilize

lsat1={}

# Populate with arrays

for k in lsat.keys():

# Exclude nested dictionaries

if (k=='THLB'):

continue

lsat1[k]=np.zeros((1,N\_StandsInBatch),dtype='int16')

lsat1[k][0,:]=lsat[k][indBat]

# Timber harvesting landbase (1=yes, 0=no)

sLCLU=meta[pNam]['Scenario'][iScn]['Land Cover/Land Use Scenario']

lsat1['THLB']=lsat['THLB'][sLCLU][:,indBat]

# Temperature will be updated automatically

lsat1['MAT']=4\*np.ones((1,N\_StandsInBatch))

# Probability of harvest (%/yr) from spatial map

lsat1['Prob Harvest (%/yr) x 1000']=lsat['Prob Harvest (%/yr) x 1000'][indBat]

# Wood density (kg/m3)

lsat1['Wood Density']=lsat['Wood Density'][indBat]

# Sawtooth species-region samples

if meta[pNam]['Project']['Biomass Module']=='Sawtooth':

lsat1['Srs1\_ID']=meta['LUT']['Spc'][meta[pNam]['Scenario'][iScn]['SRS1\_CD']]\*np.ones((1,N\_StandsInBatch),dtype='int16')

else:

lsat1['Srs1\_ID']=9999\*np.ones((1,N\_StandsInBatch),dtype='int16')

# Save

gu.opickle(meta['Paths'][pNam]['Input Scenario'][iScn] + '\\Inventory\_Bat' + cbu.FixFileNum(iBat) + '.pkl',lsat1)

print(str(np.round((time.time()-t0)/60,decimals=1)) + ' min')

#--------------------------------------------------------------------------

# Simulate wildfires

#--------------------------------------------------------------------------

# if (meta[pNam]['Scenario'][iScn]['Wildfire Status Pre-modern']=='On') | (meta[pNam]['Scenario'][iScn]['Wildfire Status Modern']=='On') | (meta[pNam]['Scenario'][iScn]['Wildfire Status Future']=='On'):

# if meta[pNam]['Project']['Frozen Ensembles Status']=='Off':

# print('Generating wildfire information')

# t0=time.time()

# asm.SimulateWildfireFromAAO(meta,pNam,lsat)

# print(str(np.round((time.time()-t0)/60,decimals=1)) + ' min')

#--------------------------------------------------------------------------

# Simulate IBM

#--------------------------------------------------------------------------

# if (meta[pNam]['Scenario'][iScn]['MPB Status Pre-modern']=='On') | (meta[pNam]['Scenario'][iScn]['MPB Status Modern']=='On') | (meta[pNam]['Scenario'][iScn]['MPB Status Future']=='On'):

# if meta[pNam]['Project']['Frozen Ensembles Status']=='Off':

# print('Generating MPB information')

# t0=time.time()

# asm.SimulateIBMFromAAO(meta,pNam,lsat)

# print(str(np.round((time.time()-t0)/60,decimals=1)) + ' min')

#--------------------------------------------------------------------------

# Prepare disturbance/management event chronology

#--------------------------------------------------------------------------

print('Preparing DMEC input files')

t0=time.time()

for iEns in range(meta[pNam]['Project']['N Ensemble']):

for iScn in range(meta[pNam]['Project']['N Scenario']):

#if (meta[pNam]['Scenario'][iScn]['Wildfire Status Pre-modern']=='On') | (meta[pNam]['Scenario'][iScn]['Wildfire Status Modern']=='On') | (meta[pNam]['Scenario'][iScn]['Wildfire Status Future']=='On'):

# # Import wildfire from aspatial stats model

# if meta[pNam]['Project']['Frozen Ensembles Status']=='Off':

# wf\_sim=gu.ipickle(meta['Paths'][pNam]['Data'] + '\\Inputs\\Ensembles\\wf\_sim\_Scn' + cbu.FixFileNum(iScn) + '\_Ens' + cbu.FixFileNum(iEns) + '.pkl')

# else:

# wf\_sim=gu.ipickle(meta[pNam]['Project']['Frozen Ensembles Path'] + '\\wf\_sim\_Scn' + cbu.FixFileNum(iScn) + '\_Ens' + cbu.FixFileNum(iEns) + '.pkl')

# if 'idx' in wf\_sim:

# idx=wf\_sim['idx']

# tmp=wf\_sim.copy()

# for v in ['Occurrence','Mortality']:

# wf\_sim[v]=np.zeros((meta[pNam]['Project']['N Time'],meta[pNam]['Project']['N Stand']),dtype='int8')

# wf\_sim[v][idx[0],idx[1]]=tmp[v]

# del tmp

# # Import wildfire from onset-spread model

# if 'Use Wildfire from OSM' in meta[pNam]['Project']:

# idx=gu.ipickle(meta['Paths'][pNam]['Data'] + '\\Inputs\\Ensembles\\wf\_sim\_osm\_Scn' + cbu.FixFileNum(iScn) + '\_Ens' + cbu.FixFileNum(iEns) + '.pkl')

# wf\_osm={}

# wf\_osm['Raw']=np.zeros( (lsc['Scenarios'][0]['Cover'].shape) ,dtype='int8')

# wf\_osm['Raw'][idx]=1

# wf\_osm['Raw']=np.reshape(wf\_osm['Raw'],(lsc['tv'].size,meta[pNam]['Project']['N Stand']))

# wf\_osm['Occurrence']=np.zeros((meta[pNam]['Project']['N Time'],meta[pNam]['Project']['N Stand']),dtype='int8')

# wf\_osm['Mortality']=np.zeros((meta[pNam]['Project']['N Time'],meta[pNam]['Project']['N Stand']),dtype='int8')

# for iT in range(lsc['tv'].size):

# if lsc['tv'][iT]<2022:

# continue

# indT=np.where(meta[pNam]['Year']==lsc['tv'][iT])[0]

# indS=np.where(wf\_osm['Raw'][iT,:]==1)[0]

# wf\_osm['Occurrence'][indT,indS]=1

# wf\_osm['Mortality'][indT,indS]=100

# if (meta[pNam]['Scenario'][iScn]['MPB Status Pre-modern']=='On') | (meta[pNam]['Scenario'][iScn]['MPB Status Modern']=='On') | (meta[pNam]['Scenario'][iScn]['MPB Status Future']=='On'):

# # Import simulated mountain pine beetle

# if meta[pNam]['Project']['Frozen Ensembles Status']=='Off':

# ibm\_sim=gu.ipickle(meta['Paths'][pNam]['Data'] + '\\Inputs\\Ensembles\\ibm\_sim\_Scn' + cbu.FixFileNum(iScn) + '\_Ens' + cbu.FixFileNum(iEns) + '.pkl')

# else:

# ibm\_sim=gu.ipickle(meta[pNam]['Project']['Frozen Ensembles Path'] + '\\ibm\_sim\_Scn' + cbu.FixFileNum(iScn) + '\_Ens' + cbu.FixFileNum(iEns) + '.pkl')

# if 'idx' in ibm\_sim:

# idx=ibm\_sim['idx']

# tmp=ibm\_sim.copy()

# for v in ['Occurrence','Mortality']:

# ibm\_sim[v]=np.zeros((meta[pNam]['Project']['N Time'],meta[pNam]['Project']['N Stand']),dtype='int8')

# ibm\_sim[v][idx[0],idx[1]]=tmp[v]

# del tmp

for iBat in range(meta[pNam]['Project']['N Batch']):

# Index to batch

indBat=cbu.IndexToBatch(meta[pNam],iBat)

# Initialize dictionary

ec={}

ec['ID Event Type']=np.zeros((meta[pNam]['Year'].size,indBat.size,meta['Core']['Max Events Per Year']),dtype='int16')

ec['Mortality Factor']=np.zeros((meta[pNam]['Year'].size,indBat.size,meta['Core']['Max Events Per Year']),dtype='int16')

ec['Growth Factor']=np.zeros((meta[pNam]['Year'].size,indBat.size,meta['Core']['Max Events Per Year']),dtype='int16')

ec['ID Growth Curve']=np.zeros((meta[pNam]['Year'].size,indBat.size,meta['Core']['Max Events Per Year']),dtype='int16')

tv=np.arange(meta[pNam]['Project']['Year Start'],meta[pNam]['Project']['Year End']+1,1)

#----------------------------------------------------------

# Spinup with constant return interval

#----------------------------------------------------------

if (meta[pNam]['Project']['Spinup Status']=='On'):

lsat0=gu.ipickle(meta['Paths'][pNam]['Input Scenario'][iScn] + '\\Inventory\_Bat' + cbu.FixFileNum(iBat) + '.pkl')

for iS in range(indBat.size):

# Index to stand

iStandFull=indBat[iS]

# Pre-industrial disturbance interval

# Old: regional

#if lsat0['Region Code'][0,iS]==meta['LUT']['Region']['Coast']:

# ivl\_pi=300

#else:

# ivl\_pi=125

if meta[pNam]['Project']['Return Interval Source']=='Custom':

# From custom input

ivl\_pi=meta[pNam]['Project']['Custom Return Interval']

elif meta[pNam]['Project']['Return Interval Source']=='BGC Zone':

# BGC Zone values

cd=cbu.lut\_n2s(meta['LUT']['BEC\_BIOGEOCLIMATIC\_POLY']['ZONE'],lsat0['ID\_BGCZ'][0,iS])[0]

ind=np.where(meta['Param']['BE']['BGC Zone Averages']['Name']==cd)[0]

ivl\_pi=meta['Param']['BE']['BGC Zone Averages']['Disturbance Return Interval'][ind]

else:

print('Spin-up return interval source incorrect.')

# Timing of transition between pre-industrial and modern periods

try:

YearRef=dmec[iScn][iStandFull]['Year'][0]

except:

YearRef=np.random.randint(1775,high=1920,size=1,dtype=int)

AgeRef=ivl\_pi

YrRegCyc=np.arange(YearRef-AgeRef-100\*ivl\_pi,YearRef-AgeRef+ivl\_pi,ivl\_pi)

Year=YrRegCyc[np.where(YrRegCyc>=meta[pNam]['Year'][0])[0]]

ID\_Type=meta['LUT']['Event']['Wildfire']\*np.ones(Year.size)

MortF=100\*np.ones(Year.size)

GrowthF=0\*np.ones(Year.size)

ID\_GrowthCurve=1\*np.ones(Year.size)

ec=cbu.CompileEvents(ec,tv,iS,ID\_Type,Year,MortF,GrowthF,ID\_GrowthCurve)

#------------------------------------------------------------------

# Add modern era events

#------------------------------------------------------------------

for iS in range(indBat.size):

# Index to stand

iStandFull=indBat[iS]

ind=np.where(dmec[iScn][iStandFull]['Scenario Affected']==1)[0]

if ind.size>0:

ID\_Type=dmec[iScn][iStandFull]['ID Event Type'][ind]

# Remove historical wildfire if it is being simulated

if meta[pNam]['Scenario'][iScn]['Wildfire Scn Obs Period']!=-9999:

ind2=np.where(ID\_Type!=meta['LUT']['Event']['Wildfire'])[0]

ind=ind[ind2]

ID\_Type=ID\_Type[ind2]

Year=dmec[iScn][iStandFull]['Year'][ind]

MortF=dmec[iScn][iStandFull]['Mortality Factor'][ind]

GrowthF=dmec[iScn][iStandFull]['Growth Factor'][ind]

ID\_GrowthCurve=dmec[iScn][iStandFull]['ID\_GC'][ind]

ec=cbu.CompileEvents(ec,tv,iS,ID\_Type,Year,MortF,GrowthF,ID\_GrowthCurve)

# #------------------------------------------------------------------

# # Add simulated wildfire

# #------------------------------------------------------------------

# if (meta[pNam]['Scenario'][iScn]['Wildfire Status Pre-modern']=='On') | (meta[pNam]['Scenario'][iScn]['Wildfire Status Modern']=='On') | (meta[pNam]['Scenario'][iScn]['Wildfire Status Future']=='On'):

# Occ=wf\_sim['Occurrence'][:,indBat].copy()

# Mort=wf\_sim['Mortality'][:,indBat]

# for iEPY in range(meta['Core']['Max Events Per Year']):

# # Index to available spots with simulated wildfire mortality

# ind=np.where( (ec['ID Event Type'][:,:,iEPY]==0) & (Occ==1) )

# # Populate

# ec['ID Event Type'][ind[0],ind[1],iEPY]=meta['LUT']['Event']['Wildfire']

# ec['Mortality Factor'][ind[0],ind[1],iEPY]=Mort[ind[0],ind[1]]

# ec['Growth Factor'][ind[0],ind[1],iEPY]=9999

# ec['ID Growth Curve'][ind[0],ind[1],iEPY]=1

# # Eliminate occurrence so that it is not populated again as loop

# # through events per year continues

# Occ[ind[0],ind[1]]=0

# # Add simulated wildfire from onset-spread model

# if 'Use Wildfire from OSM' in meta[pNam]['Project']:

# Occ=wf\_osm['Occurrence'][:,indBat].copy()

# Mort=wf\_osm['Mortality'][:,indBat]

# for iEPY in range(meta['Core']['Max Events Per Year']):

# # Index to available spots with simulated wildfire mortality

# ind=np.where( (ec['ID Event Type'][:,:,iEPY]==0) & (Occ==1) )

# # Populate

# ec['ID Event Type'][ind[0],ind[1],iEPY]=meta['LUT']['Event']['Wildfire']

# ec['Mortality Factor'][ind[0],ind[1],iEPY]=Mort[ind[0],ind[1]]

# ec['Growth Factor'][ind[0],ind[1],iEPY]=9999

# ec['ID Growth Curve'][ind[0],ind[1],iEPY]=1

# # Eliminate occurrence so that it is not populated again as loop

# # through events per year continues

# Occ[ind[0],ind[1]]=0

#------------------------------------------------------------------

# Add simulated MPB

#------------------------------------------------------------------

# if (meta[pNam]['Scenario'][iScn]['MPB Status Pre-modern']=='On') | (meta[pNam]['Scenario'][iScn]['MPB Status Modern']=='On') | (meta[pNam]['Scenario'][iScn]['MPB Status Future']=='On'):

# Occ=ibm\_sim['Occurrence'][:,indBat].copy()

# Mort=ibm\_sim['Mortality'][:,indBat]

# for iEPY in range(meta['Core']['Max Events Per Year']):

# # Index to available spots with simulated wildfire mortality

# ind=np.where( (ec['ID Event Type'][:,:,iEPY]==0) & (Occ==1) )

# # Populate

# ec['ID Event Type'][ind[0],ind[1],iEPY]=meta['LUT']['Event']['Mountain Pine Beetle']

# ec['Mortality Factor'][ind[0],ind[1],iEPY]=Mort[ind[0],ind[1]]

# ec['Growth Factor'][ind[0],ind[1],iEPY]=9999

# ec['ID Growth Curve'][ind[0],ind[1],iEPY]=1

# # Eliminate occurrence so that it is not populated again as loop

# # through events per year continues

# Occ[ind[0],ind[1]]=0

#------------------------------------------------------------------

# Add future scheduled NOSE

#------------------------------------------------------------------

if 'NOSE Future' in meta[pNam]['Project']:

c,ia,ib=np.intersect1d(indBat,meta[pNam]['Project']['NOSE Future']['Stand Index'],return\_indices=True)

if ia.size>0:

for iP in range(ia.size):

YearIncite=meta[pNam]['Project']['NOSE Future']['Year'][ib[iP]]

iT=np.where(tv==YearIncite)[0]

iAvailable=np.where(ec['ID Event Type'][iT,ia[iP],:]==0)[0]

ec['ID Event Type'][iT,ia[iP],iAvailable]=meta['LUT']['Event']['GC Switch']

ec['Mortality Factor'][iT,ia[iP],iAvailable]=0

ec['Growth Factor'][iT,ia[iP],iAvailable]=-10

ec['ID Growth Curve'][iT,ia[iP],iAvailable]=1

if np.isin(iScn,meta[pNam]['Project']['Actual Indices'])==True:

TimeBetweenFireAndPlant=2

iT=np.where(tv==YearIncite+TimeBetweenFireAndPlant)[0]

iAvailable=np.where(ec['ID Event Type'][iT,ia[iP],:]==0)[0]

ID\_GC=np.max(gc[iScn][indBat[ia[iP]]]['ID\_GC'])

ec['ID Event Type'][iT,ia[iP],iAvailable]=meta['LUT']['Event']['Planting']

ec['Mortality Factor'][iT,ia[iP],iAvailable]=0

ec['Growth Factor'][iT,ia[iP],iAvailable]=10

ec['ID Growth Curve'][iT,ia[iP],iAvailable]=ID\_GC

#------------------------------------------------------------------

# Compress by indexing into the elements with information

#------------------------------------------------------------------

ec['idx']=np.where(ec['ID Event Type']>0)

ec['ID Event Type']=ec['ID Event Type'][ec['idx']]

ec['Mortality Factor']=ec['Mortality Factor'][ec['idx']]

ec['Growth Factor']=ec['Growth Factor'][ec['idx']]

ec['ID Growth Curve']=ec['ID Growth Curve'][ec['idx']]

gu.opickle(meta['Paths'][pNam]['Input Scenario'][iScn] + '\\Events\_Ens' + cbu.FixFileNum(iEns) + '\_Bat' + cbu.FixFileNum(iBat) + '.pkl',ec)

print(str(np.round((time.time()-t0)/60,decimals=1)) + ' min')

#--------------------------------------------------------------------------

# Prepare growth curves

#--------------------------------------------------------------------------

print('Preparing growth curve input files')

t0=time.time()

cbu.PrepGrowthCurvesUniqueForCBR(meta,pNam,ugc)

print(str(np.round((time.time()-t0)/60,decimals=1)) + ' min')

#--------------------------------------------------------------------------

# Save data

#--------------------------------------------------------------------------

print('Saving input files')

t0=time.time()

gu.opickle(meta['Paths'][pNam]['Data'] + '\\Inputs\\Metadata.pkl',meta)

gu.opickle(meta['Paths'][pNam]['Data'] + '\\Inputs\\Metadata\_backup.pkl',meta)

gu.opickle(meta['Paths'][pNam]['Data'] + '\\Inputs\\dmec.pkl',dmec)

gu.opickle(meta['Paths'][pNam]['Data'] + '\\Inputs\\lsat.pkl',lsat)

gu.opickle(meta['Paths'][pNam]['Data'] + '\\Inputs\\gc.pkl',gc)

gu.opickle(meta['Paths'][pNam]['Data'] + '\\Inputs\\ugc.pkl',ugc)

print(str(np.round((time.time()-t0)/60,decimals=1)) + ' min')

#--------------------------------------------------------------------------

# Delete all output files

#--------------------------------------------------------------------------

print('Deleting any output files')

t0=time.time()

cbu.DeleteAllOutputFiles(meta,pNam)

print(str(np.round((time.time()-t0)/60,decimals=1)) + ' min')

return meta,lsat,dmec

#%% Timber harvesting land base

def DefineTHLB(meta,pNam,lsat):

# Initialize THLB flags for each land cover/land use change scenario (THLB=1,Non-THLB=0)

thlb={}

# Scenario 0: No REARs are removed from THLB historically or in the future

thlb['S0']=np.ones((meta[pNam]['Project']['N Time'],meta[pNam]['Project']['N Stand']),dtype='int8')

# Scenario 1: Historical REARs are removed from THLB, but there is no

# no subequent establishment of REARs

thlb['S1']=np.ones((meta[pNam]['Project']['N Time'],meta[pNam]['Project']['N Stand']),dtype='int8')

# Scenario 2: Historical REARs are removed from THLB, with subsequent establishment

# of REARs also removed from THLB

thlb['S2']=np.ones((meta[pNam]['Project']['N Time'],meta[pNam]['Project']['N Stand']),dtype='int8')

# Index to stands that are uneconomic

iUneconomic=np.where(lsat['SI']<=5)[0]

if iUneconomic.size>0:

# Remove uneconomic stands from THLB

for k in thlb.keys():

thlb[k][:,iUneconomic]=0

# Idenify stands that have been harvested

# has\_been\_harvested=np.zeros(meta[pNam]['Project']['N Stand'])

# for iStand in range(len(dmec)):

# ind=np.where( (dmec[iScn][iStand]['ID Event Type']==meta['LUT']['Event']['Harvest']) | (dmec[iScn][iStand]['ID Event Type']==meta['LUT']['Event']['Harvest Salvage']) )[0]

# if ind.size>0:

# has\_been\_harvested[iStand]=1

# Index to stands that have not been harvested

iNoHarv=np.where( (lsat['Harvest Year Comp2']==0) & (lsat['SI']>5) )[0]

# Use the ratio of THLB to non-THLB as an indicator of what will be harvested

# among remaining primary forest

ratio\_thlb=22/62 # ratio of THLB to total forest (SOF)

# Ratio of uneconomic to total (needed to adjust probability)

corr=iUneconomic.size/meta[pNam]['Project']['N Stand']

# Probability of evading harvest

if iNoHarv.size>0:

p\_evade=(1-ratio\_thlb-corr)\*(meta[pNam]['Project']['N Stand']/iNoHarv.size)

else:

p\_evade=(1-ratio\_thlb-corr)

# Random prediction of whether it will evade harvesting

iRem=np.where(np.random.random(iNoHarv.size)<p\_evade)[0]

for k in thlb.keys():

thlb[k][:,iNoHarv[iRem]]=0

# np.sum(thlb['Actual'][0,:])/meta[pNam]['Project']['N Stand']

#--------------------------------------------------------------------------

# Transitions

#--------------------------------------------------------------------------

# Initialize year of transition

thlb\_YearTransitionOut=np.zeros(meta[pNam]['Project']['N Stand'])

# Historical transitions

ind=np.where( (lsat['LandCover\_Comp1\_2019']==meta['LUT']['Derived']['lc\_comp1']['Forest']) & (lsat['LandUse\_Comp1\_2019']==meta['LUT']['Derived']['lu\_comp1']['Conservation Natural']) )

thlb\_YearTransitionOut[ind]=lsat['LandUseChange\_Comp1\_1800to2019\_Year'][ind]

ind=np.where( (lsat['LandCover\_Comp1\_2019']==meta['LUT']['Derived']['lc\_comp1']['Forest']) & (lsat['LandUse\_Comp1\_2019']==meta['LUT']['Derived']['lu\_comp1']['Conservation Consistent']) )

thlb\_YearTransitionOut[ind]=lsat['LandUseChange\_Comp1\_1800to2019\_Year'][ind]

# Apply historical transitions to Scenario 1 and 2

for j in range(thlb\_YearTransitionOut.size):

if thlb\_YearTransitionOut[j]>0:

it=np.where( (meta[pNam]['Year']>=thlb\_YearTransitionOut[j]) )[0]

thlb['S1'][it,j]=0

# Add future transitions to historical transitions

ind=np.where( (lsat['LandCover\_Comp1\_2019']==meta['LUT']['Derived']['lc\_comp1']['Forest']) & (lsat['LandUse\_Comp1\_2019']!=meta['LUT']['Derived']['lu\_comp1']['Conservation Natural']) & (lsat['LandUse\_Comp1\_2049\_Scn2']==meta['LUT']['Derived']['lu\_comp1']['Conservation Natural']) )

thlb\_YearTransitionOut[ind]=lsat['LandUseChange\_Comp1\_2020to2049\_Scn2\_Year'][ind]

ind=np.where( (lsat['LandCover\_Comp1\_2019']==meta['LUT']['Derived']['lc\_comp1']['Forest']) & (lsat['LandUse\_Comp1\_2019']!=meta['LUT']['Derived']['lu\_comp1']['Conservation Consistent']) & (lsat['LandUse\_Comp1\_2049\_Scn2']==meta['LUT']['Derived']['lu\_comp1']['Conservation Consistent']) )

thlb\_YearTransitionOut[ind]=lsat['LandUseChange\_Comp1\_2020to2049\_Scn2\_Year'][ind]

# Apply historical and future transitions to Scenario 2

for j in range(thlb\_YearTransitionOut.size):

if thlb\_YearTransitionOut[j]>0:

it=np.where( (meta[pNam]['Year']>=thlb\_YearTransitionOut[j]) )[0]

thlb['S2'][it,j]=0

# #--------------------------------------------------------------------------

# # Baselines

# #--------------------------------------------------------------------------

# # Adjust the baseline so that simulated harvesting between 1995 and 2022 only

# # occurs in areas where the THLB was affected by value diversification

# for year in range(1990,2023,1):

# iT=np.where(meta[pNam]['Year']==year)[0]

# iS=np.where( (thlb['Baseline'][iT,:]==1) & (thlb['Actual'][iT,:]==1) )[1]

# thlb['Baseline'][iT,iS]=0

# iS=np.where( (thlb['Scn1 Baseline'][iT,:]==1) & (thlb['Scn1 Actual'][iT,:]==1) )[1]

# thlb[iScn]['Scn1 Baseline'][iT,iS]=0

lsat['THLB']=thlb

return lsat

# def DefineTHLB(meta,pNam,lsat,dmec,lsc):

# thlb=[{}]\*meta[pNam]['Project']['N Scenario']

# for iScn in range(meta[pNam]['Project']['N Scenario']):

# #--------------------------------------------------------------------

# # Initialize THLB flags (THLB=1,Non-THLB=0)

# #--------------------------------------------------------------------

# # Initially assume everything is in the THLB

# thlb[iScn]['Actual']=np.ones((meta[pNam]['Project']['N Time'],meta[pNam]['Project']['N Stand']),dtype='int8')

# thlb[iScn]['Baseline']=thlb[iScn]['Actual'].copy()

# thlb[iScn]['Scn1 Actual']=thlb[iScn]['Actual'].copy()

# thlb[iScn]['Scn1 Baseline']=thlb[iScn]['Actual'].copy()

# # Index to stands that are uneconomic

# iUneconomic=np.where(lsat['SI']<=5)[0]

# if iUneconomic.size>0:

# # Remove uneconomic stands from THLB

# thlb[iScn]['Actual'][:,iUneconomic]=0

# thlb[iScn]['Baseline'][:,iUneconomic]=0

# thlb[iScn]['Scn1 Actual'][:,iUneconomic]=0

# thlb[iScn]['Scn1 Baseline'][:,iUneconomic]=0

# # Idenify stands that have been harvested

# has\_been\_harvested=np.zeros(meta[pNam]['Project']['N Stand'])

# for iStand in range(len(dmec)):

# ind=np.where( (dmec[iScn][iStand]['ID Event Type']==meta['LUT']['Event']['Harvest']) | (dmec[iScn][iStand]['ID Event Type']==meta['LUT']['Event']['Harvest Salvage']) )[0]

# if ind.size>0:

# has\_been\_harvested[iStand]=1

# # Index to stands that have not been harvested

# iNoHarv=np.where( (has\_been\_harvested==0) & (lsat['SI']>5) )[0]

# # Use the ratio of THLB to non-THLB as an indicator of what will be harvested

# # among remaining primary forest

# ratio\_thlb=22/60 # ratio of THLB to total forest (SOF)

# # Ratio of uneconomic to total (needed to adjust probability)

# corr=iUneconomic.size/meta[pNam]['Project']['N Stand']

# # Probability of evading harvest

# if iNoHarv.size>0:

# p\_evade=(1-ratio\_thlb-corr)\*(meta[pNam]['Project']['N Stand']/iNoHarv.size)

# else:

# p\_evade=(1-ratio\_thlb-corr)

# # Random prediction of whether it will evade harvesting

# iRem=np.where(np.random.random(iNoHarv.size)<p\_evade)[0]

# thlb[iScn]['Actual'][:,iNoHarv[iRem]]=0

# thlb[iScn]['Baseline'][:,iNoHarv[iRem]]=0

# thlb[iScn]['Scn1 Actual'][:,iNoHarv[iRem]]=0

# thlb[iScn]['Scn1 Baseline'][:,iNoHarv[iRem]]=0

# # np.sum(thlb['Actual'][0,:])/meta[pNam]['Project']['N Stand']

# #------------------------------------------------------------------------------

# # Actual (New based on REAR layer)

# #------------------------------------------------------------------------------

# # Initialize year of transition

# thlb\_YearTransitionOut=np.zeros(meta[pNam]['Project']['N Stand'])

# ind=np.where(lsat['THLB Layer']==1)[0]

# thlb\_YearTransitionOut[ind]=1990

# # Conservation from land surface classification

# name=meta[pNam]['Scenario'][iScn]['Land Surface Scenario']

# if name!='Off':

# idx=LSC\_Scenario\_Crosswalk(lsc,name)

# Use=np.reshape(lsc['Scenarios'][idx]['Use'].copy(),(lsc['tv'].size,meta[pNam]['Project']['N Stand']))

# ind=np.where( Use==meta['LUT']['LSC']['Use']['Conservation Consistent'] )

# if ind[0].size>0:

# for i in range(ind[0].size):

# thlb\_YearTransitionOut[ind[1][i]]=lsc['tv'][ind[0][i]]

# # Apply transition to actual THLB

# for j in range(thlb\_YearTransitionOut.size):

# if thlb\_YearTransitionOut[j]>0:

# it=np.where( (meta[pNam]['Year']>=thlb\_YearTransitionOut[j]) )[0]

# thlb[iScn]['Actual'][it,j]=0

# thlb[iScn]['Scn1 Actual'][it,j]=0

# #------------------------------------------------------------------------------

# # Scn1 Actual (with deferrals + random areas to achieve 30 by 30)

# #------------------------------------------------------------------------------

# thlb\_YearTransitionOut=np.zeros(meta[pNam]['Project']['N Stand'])

# ind=np.where(lsat['THLB Layer']==2)[0]

# thlb\_YearTransitionOut[ind]=2023

# # Apply transition to actual THLB

# for j in range(thlb\_YearTransitionOut.size):

# if thlb\_YearTransitionOut[j]>0:

# it=np.where( (meta[pNam]['Year']>=thlb\_YearTransitionOut[j]) )[0]

# thlb[iScn]['Scn1 Actual'][it,j]=0

# #------------------------------------------------------------------------------

# # Baselines

# #------------------------------------------------------------------------------

# # Adjust the baseline so that simulated harvesting between 1995 and 2022 only

# # occurs in areas where the THLB was affected by value diversification

# for year in range(1990,2023,1):

# iT=np.where(meta[pNam]['Year']==year)[0]

# iS=np.where( (thlb[iScn]['Baseline'][iT,:]==1) & (thlb[iScn]['Actual'][iT,:]==1) )[1]

# thlb[iScn]['Baseline'][iT,iS]=0

# iS=np.where( (thlb[iScn]['Scn1 Baseline'][iT,:]==1) & (thlb[iScn]['Scn1 Actual'][iT,:]==1) )[1]

# thlb[iScn]['Scn1 Baseline'][iT,iS]=0

# return thlb

#%% Get index to a scenario in the land surface class list

# def LSC\_Scenario\_Crosswalk(lsc,name):

# if 'Scenarios' not in lsc:

# return

# for i in range(len(lsc['Scenarios'])):

# if lsc['Scenarios'][i]['Name']==name:

# return i

#%% Get unique growth curves

def ExtractUniqueGrowthCurves(meta,pNam,gc):

ugc={}

ugc['GC\_Variable\_List']=np.array(meta['Modules']['GYM']['GC\_Variable\_List'])[3:]

# Calculate unique stand types

ugc['Full']=np.zeros((int(10e6),len(meta['Modules']['GYM']['GC\_Variable\_List'])))

cnt=0

for iScn in range(meta[pNam]['Project']['N Scenario']):

for iStand in range(meta[pNam]['Project']['N Stand']):

gc0=gc[iScn][iStand]

for iGC in range(gc0['ID\_GC'].size):

for k in range(len(meta['Modules']['GYM']['GC\_Variable\_List'])):

key=meta['Modules']['GYM']['GC\_Variable\_List'][k]

ugc['Full'][cnt,k]=gc0[key][iGC]

cnt=cnt+1

ugc['Full']=ugc['Full'][0:cnt,:]

# Unique growth curves

# The 'Inverse' variable acts as the crosswalk between the full and unique gc arrays

ugc['Unique'],ugc['Index'],ugc['Inverse']=np.unique(ugc['Full'][:,3:],return\_index=True,return\_inverse=True,axis=0)

return ugc

#%% Add changes in land surface classfication to DMEC

def AddLandSurfaceChangesToDMEC(meta,pNam,dmec,lsc):

for iScn in range(meta[pNam]['Project']['N Scenario']):

# Name of LS scenario for each scenario

sNam=meta[pNam]['Scenario'][iScn]['Land Surface Scenario']

# Index to LSC scenario

for i in range(len(lsc['Scenarios'])):

if lsc['Scenarios'][i]['Name']==sNam:

idx=i

break

if sNam!='Off':

#Cover=np.reshape(lsc['Scenarios'][idx]['Cover'].copy(),(lsc['tv'].size,meta[pNam]['Project']['N Stand']))

Use=np.reshape(lsc['Scenarios'][idx]['Use'].copy(),(lsc['tv'].size,meta[pNam]['Project']['N Stand']))

#----------------------------------------------------------------------

# Fuel breaks

#----------------------------------------------------------------------

nam='Fuel Break'

indS=np.unique(np.where( Use==meta['LUT']['LSC']['Use'][nam] )[1])

if indS.size>0:

for i in range(indS.size):

iS=indS[i]

iT=np.where(Use[:,iS]==meta['LUT']['LSC']['Use'][nam])[0][0]

# Add harvest

dmec[iScn][iS]['Year']=np.append(dmec[iScn][iS]['Year'],lsc['tv'][iT])

dmec[iScn][iS]['ID Event Type']=np.append(dmec[iScn][iS]['ID Event Type'],meta['LUT']['Event']['Harvest'])

dmec[iScn][iS]['Mortality Factor']=np.append(dmec[iScn][iS]['Mortality Factor'],100)

dmec[iScn][iS]['Growth Factor']=np.append(dmec[iScn][iS]['Growth Factor'],9999)

if 'Index to Event Inciting NOSE' in dmec[iScn][iS]:

dmec[iScn][iS]['Index to Event Inciting NOSE']=np.append(dmec[iScn][iS]['Index to Event Inciting NOSE'],9999)

for v in meta['Core']['StringsToFill']:

dmec[iScn][iS][v]=np.append(dmec[iScn][iS][v],9999)

# Add slashpile burn

dmec[iScn][iS]['Year']=np.append(dmec[iScn][iS]['Year'],lsc['tv'][iT]+1)

dmec[iScn][iS]['ID Event Type']=np.append(dmec[iScn][iS]['ID Event Type'],meta['LUT']['Event']['Slashpile Burn'])

dmec[iScn][iS]['Mortality Factor']=np.append(dmec[iScn][iS]['Mortality Factor'],100)

dmec[iScn][iS]['Growth Factor']=np.append(dmec[iScn][iS]['Growth Factor'],9999)

if 'Index to Event Inciting NOSE' in dmec[iScn][iS]:

dmec[iScn][iS]['Index to Event Inciting NOSE']=np.append(dmec[iScn][iS]['Index to Event Inciting NOSE'],9999)

for v in meta['Core']['StringsToFill']:

dmec[iScn][iS][v]=np.append(dmec[iScn][iS][v],9999)

# Add planting

dmec[iScn][iS]['Year']=np.append(dmec[iScn][iS]['Year'],lsc['tv'][iT]+2)

dmec[iScn][iS]['ID Event Type']=np.append(dmec[iScn][iS]['ID Event Type'],meta['LUT']['Event']['Planting'])

dmec[iScn][iS]['Mortality Factor']=np.append(dmec[iScn][iS]['Mortality Factor'],0)

dmec[iScn][iS]['Growth Factor']=np.append(dmec[iScn][iS]['Growth Factor'],9999)

if 'Index to Event Inciting NOSE' in dmec[iScn][iS]:

dmec[iScn][iS]['Index to Event Inciting NOSE']=np.append(dmec[iScn][iS]['Index to Event Inciting NOSE'],9999)

for v in meta['Core']['StringsToFill']:

dmec[iScn][iS][v]=np.append(dmec[iScn][iS][v],9999)

dmec[iScn][iS]['PL\_SPECIES\_CD1'][-1]=meta['LUT']['VEG\_COMP\_LYR\_R1\_POLY']['SPECIES\_CD\_1']['AT']

dmec[iScn][iS]['PL\_SPECIES\_PCT1'][-1]=100

# Add harvest

RotationLength=14

for iR in range(1,10):

dmec[iScn][iS]['Year']=np.append(dmec[iScn][iS]['Year'],lsc['tv'][iT]+2+iR\*RotationLength)

dmec[iScn][iS]['ID Event Type']=np.append(dmec[iScn][iS]['ID Event Type'],meta['LUT']['Event']['Harvest'])

dmec[iScn][iS]['Mortality Factor']=np.append(dmec[iScn][iS]['Mortality Factor'],100)

dmec[iScn][iS]['Growth Factor']=np.append(dmec[iScn][iS]['Growth Factor'],9999)

if 'Index to Event Inciting NOSE' in dmec[iScn][iS]:

dmec[iScn][iS]['Index to Event Inciting NOSE']=np.append(dmec[iScn][iS]['Index to Event Inciting NOSE'],9999)

for v in meta['Core']['StringsToFill']:

dmec[iScn][iS][v]=np.append(dmec[iScn][iS][v],9999)

#----------------------------------------------------------------------

# Energy Production

#----------------------------------------------------------------------

nam='Energy Production'

indS=np.unique(np.where( Use==meta['LUT']['LSC']['Use'][nam] )[1])

if indS.size>0:

for i in range(indS.size):

iS=indS[i]

iT=np.where(Use[:,iS]==meta['LUT']['LSC']['Use'][nam])[0][0]

# Add harvest

dmec[iScn][iS]['Year']=np.append(dmec[iScn][iS]['Year'],lsc['tv'][iT])

dmec[iScn][iS]['ID Event Type']=np.append(dmec[iScn][iS]['ID Event Type'],meta['LUT']['Event']['Harvest'])

dmec[iScn][iS]['Mortality Factor']=np.append(dmec[iScn][iS]['Mortality Factor'],100)

dmec[iScn][iS]['Growth Factor']=np.append(dmec[iScn][iS]['Growth Factor'],9999)

if 'Index to Event Inciting NOSE' in dmec[iScn][iS]:

dmec[iScn][iS]['Index to Event Inciting NOSE']=np.append(dmec[iScn][iS]['Index to Event Inciting NOSE'],9999)

for v in meta['Core']['StringsToFill']:

dmec[iScn][iS][v]=np.append(dmec[iScn][iS][v],9999)

# Add slashpile burn

dmec[iScn][iS]['Year']=np.append(dmec[iScn][iS]['Year'],lsc['tv'][iT]+1)

dmec[iScn][iS]['ID Event Type']=np.append(dmec[iScn][iS]['ID Event Type'],meta['LUT']['Event']['Slashpile Burn'])

dmec[iScn][iS]['Mortality Factor']=np.append(dmec[iScn][iS]['Mortality Factor'],100)

dmec[iScn][iS]['Growth Factor']=np.append(dmec[iScn][iS]['Growth Factor'],9999)

if 'Index to Event Inciting NOSE' in dmec[iScn][iS]:

dmec[iScn][iS]['Index to Event Inciting NOSE']=np.append(dmec[iScn][iS]['Index to Event Inciting NOSE'],9999)

for v in meta['Core']['StringsToFill']:

dmec[iScn][iS][v]=np.append(dmec[iScn][iS][v],9999)

# Add planting

dmec[iScn][iS]['Year']=np.append(dmec[iScn][iS]['Year'],lsc['tv'][iT]+2)

dmec[iScn][iS]['ID Event Type']=np.append(dmec[iScn][iS]['ID Event Type'],meta['LUT']['Event']['Planting'])

dmec[iScn][iS]['Mortality Factor']=np.append(dmec[iScn][iS]['Mortality Factor'],0)

dmec[iScn][iS]['Growth Factor']=np.append(dmec[iScn][iS]['Growth Factor'],9999)

if 'Index to Event Inciting NOSE' in dmec[iScn][iS]:

dmec[iScn][iS]['Index to Event Inciting NOSE']=np.append(dmec[iScn][iS]['Index to Event Inciting NOSE'],9999)

for v in meta['Core']['StringsToFill']:

dmec[iScn][iS][v]=np.append(dmec[iScn][iS][v],9999)

dmec[iScn][iS]['PL\_SPECIES\_CD1'][-1]=meta['LUT']['VEG\_COMP\_LYR\_R1\_POLY']['SPECIES\_CD\_1']['AT']

dmec[iScn][iS]['PL\_SPECIES\_PCT1'][-1]=100

# Add harvest

RotationLength=14

for iR in range(1,10):

dmec[iScn][iS]['Year']=np.append(dmec[iScn][iS]['Year'],lsc['tv'][iT]+2+iR\*RotationLength)

dmec[iScn][iS]['ID Event Type']=np.append(dmec[iScn][iS]['ID Event Type'],meta['LUT']['Event']['Harvest'])

dmec[iScn][iS]['Mortality Factor']=np.append(dmec[iScn][iS]['Mortality Factor'],100)

dmec[iScn][iS]['Growth Factor']=np.append(dmec[iScn][iS]['Growth Factor'],9999)

if 'Index to Event Inciting NOSE' in dmec[iScn][iS]:

dmec[iScn][iS]['Index to Event Inciting NOSE']=np.append(dmec[iScn][iS]['Index to Event Inciting NOSE'],9999)

for v in meta['Core']['StringsToFill']:

dmec[iScn][iS][v]=np.append(dmec[iScn][iS][v],9999)

return dmec

#%% Exclude unidentified events

def Exclude\_Unidentified\_Events(meta,pNam,dmec0):

N\_Removed=0

for iStand in range(meta[pNam]['Project']['N Stand']):

if dmec0[iStand]==None:

continue

ind=np.where(dmec0[iStand]['ID Event Type']!=9999)[0]

N\_Removed=N\_Removed+ind.size

for key in dmec0[iStand]:

dmec0[iStand][key]=dmec0[iStand][key][ind]

print(str(N\_Removed) + ' unidentifiable events removed.')

return dmec0

#%% Adjust species-specific mortality

# Make sure iScn\_Actual is the index to the actual inventory. If you run it with

# a counterfactual scenario, it may encounter stands with only one GC. Yet, this

# method requires a previous GC.

# def AdjustSpeciesSpecificMortality(meta,pNam,dmec,gc,iScn\_Actual):

# # Species affected sets

# Pest\_List=['Mountain Pine Beetle','Balsam Beetle','Spruce Beetle','Douglas-fir Beetle','Western Spruce Budworm']

# SA\_List=[None]\*len(Pest\_List)

# for iPest in range(len(Pest\_List)):

# ind=np.where( (meta['Param']['BE']['DistBySC']['Name']==Pest\_List[iPest]) )[0][0]

# SA\_List[iPest]=np.array([meta['Param']['BE']['DistBySC']['SpcCD1'][ind],meta['Param']['BE']['DistBySC']['SpcCD2'][ind],

# meta['Param']['BE']['DistBySC']['SpcCD3'][ind],meta['Param']['BE']['DistBySC']['SpcCD4'][ind],

# meta['Param']['BE']['DistBySC']['SpcCD5'][ind],meta['Param']['BE']['DistBySC']['SpcCD6'][ind]])

# for iScn in range(meta[pNam]['Project']['N Scenario']):

# for iStand in range(meta[pNam]['Project']['N Stand']):

# for iYr in range(dmec[iScn][iStand]['Year'].size):

# for iPest in range(len(Pest\_List)):

# if dmec[iScn][iStand]['ID Event Type'][iYr]==meta['LUT']['Event'][Pest\_List[iPest]]:

# ind\_GC=int(dmec[iScn\_Actual][iStand]['ID\_GC'][iYr]-1)

# scd=[None]\*4

# try:

# scd[0]=cbu.lut\_n2s(meta['LUT']['VEG\_COMP\_LYR\_R1\_POLY']['SPECIES\_CD\_1'],gc[iScn\_Actual][iStand]['s1'][ind\_GC])

# except:

# print('Crashing 1')

# print(gc[iScn\_Actual][iStand]['s1'])

# print(iYr)

# print(dmec[iScn\_Actual][iStand]['ID\_GC'])

# print(ind\_GC)

# try:

# scd[1]=cbu.lut\_n2s(meta['LUT']['VEG\_COMP\_LYR\_R1\_POLY']['SPECIES\_CD\_1'],gc[iScn\_Actual][iStand]['s2'][ind\_GC])

# scd[2]=cbu.lut\_n2s(meta['LUT']['VEG\_COMP\_LYR\_R1\_POLY']['SPECIES\_CD\_1'],gc[iScn\_Actual][iStand]['s3'][ind\_GC])

# scd[3]=cbu.lut\_n2s(meta['LUT']['VEG\_COMP\_LYR\_R1\_POLY']['SPECIES\_CD\_1'],gc[iScn\_Actual][iStand]['s4'][ind\_GC])

# except:

# print('Crashing 2')

# print(gc[iScn\_Actual][iStand]['s1'])

# print(gc[iScn\_Actual][iStand]['p1'])

# print(gc[iScn\_Actual][iStand]['s2'])

# print(gc[iScn\_Actual][iStand]['p2'])

# print(iYr)

# print(dmec[iScn\_Actual][iStand]['ID\_GC'])

# print(ind\_GC)

# spct=[None]\*4

# spct[0]=gc[iScn\_Actual][iStand]['p1'][ind\_GC]

# spct[1]=gc[iScn\_Actual][iStand]['p2'][ind\_GC]

# spct[2]=gc[iScn\_Actual][iStand]['p3'][ind\_GC]

# spct[3]=gc[iScn\_Actual][iStand]['p4'][ind\_GC]

# PercentAffected=0

# for i in range(4):

# if np.isin(scd[i],SA\_List[iPest])==True:

# PercentAffected=PercentAffected+spct[i]

# dmec[iScn][iStand]['Mortality Factor'][iYr]=(PercentAffected/100)\*dmec[iScn][iStand]['Mortality Factor'][iYr]

# return dmec

#%% Exclude duplicate events from DMEC

def Exclude\_Duplicate\_Events(meta,pNam,dmec):

for iStand in range(meta[pNam]['Project']['N Stand']):

if dmec[iStand]==None:

continue

for key in meta['LUT']['Event'].keys():

indType=np.where( (dmec[iStand]['ID Event Type']==meta['LUT']['Event'][key]) )[0]

if indType.size==0:

continue

uYear=np.unique(np.floor(dmec[iStand]['Year'][indType]))

for iYear in range(uYear.size):

ind=np.where( (dmec[iStand]['ID Event Type']==meta['LUT']['Event'][key]) & (np.floor(dmec[iStand]['Year'])==uYear[iYear]) )[0]

dmec[iStand]['ID Event Type'][ind[1:]]=9999

return dmec

#%% Put DMEC events in order

def PutEventsInOrder(meta,pNam,dmec):

def count\_lists(l):

return sum(1 + count\_lists(i) for i in l if isinstance(i,list))

if count\_lists(dmec)==0:

for iStand in range(meta[pNam]['Project']['N Stand']):

d=dmec[iStand].copy()

ord=np.argsort(d['Year'])

# Fix index to inciting events

if 'Index to Event Inciting NOSE' in d.keys():

indOld=np.where(d['Index to Event Inciting NOSE']>=0)[0]

indNew=np.where(ord==d['Index to Event Inciting NOSE'][indOld])[0]

if indNew.size>0:

d['Index to Event Inciting NOSE'][indOld]=indNew

for key in d.keys():

d[key]=d[key][ord]

dmec[iStand]=d.copy()

else:

for iScn in range(meta[pNam]['Project']['N Scenario']):

for iStand in range(meta[pNam]['Project']['N Stand']):

d=dmec[iScn][iStand].copy()

ord=np.argsort(d['Year'])

# Fix index to inciting events

if 'Index to Event Inciting NOSE' in d.keys():

indOld=np.where(d['Index to Event Inciting NOSE']>=0)[0]

indNew=np.where(ord==d['Index to Event Inciting NOSE'][indOld])[0]

if indNew.size>0:

d['Index to Event Inciting NOSE'][indOld]=indNew

for key in d.keys():

d[key]=d[key][ord]

dmec[iScn][iStand]=d.copy()

return dmec

#%% Ensure disturbance prededes fertilization so age is specified

# So that age at fert is specified.

def Ensure\_Fert\_Preceded\_By\_Disturbance(meta,pNam,dmec,ba):

ListOfTestedDist=[meta['LUT']['Event']['Wildfire'],

meta['LUT']['Event']['Harvest'],

meta['LUT']['Event']['Knockdown'],

meta['LUT']['Event']['Harvest Salvage'],

meta['LUT']['Event']['Mountain Pine Beetle'],

meta['LUT']['Event']['Balsam Beetle'],

meta['LUT']['Event']['Douglas-fir Beetle'],

meta['LUT']['Event']['Spruce Beetle']]

for iStand in range(meta[pNam]['Project']['N Stand']):

if dmec[iStand]==None:

continue

iFert=np.where( (dmec[iStand]['ID Event Type']==meta['LUT']['Event']['Fertilization Aerial']) )[0]

if iFert.size==0:

continue

iFert=iFert[0]

# Index to events prior to first fertilization with 100% mortality

ind=np.where( (dmec[iStand]['Year']<dmec[iStand]['Year'][iFert]) & (dmec[iStand]['Mortality Factor']==100) & np.isin(dmec[iStand]['ID Event Type'],ListOfTestedDist) )[0]

if ind.size>0:

continue

print('Adding a harvest before nutrient application')

# Assume mean of 38 + random variation (planting is 2 years after harvest)

r=38+np.random.randint(-6,high=6)

Year=dmec[iStand]['Year'][iFert]-r

# Add harvest

for k in dmec[iStand].keys():

dmec[iStand][k]=np.append(dmec[iStand][k],9999)

dmec[iStand]['Year'][-1]=Year

dmec[iStand]['ID Event Type'][-1]=meta['LUT']['Event']['Harvest']

dmec[iStand]['Mortality Factor'][-1]=100

dmec[iStand]['Growth Factor'][-1]=9999

# Add slashpile burn

for k in dmec[iStand].keys():

dmec[iStand][k]=np.append(dmec[iStand][k],9999)

dmec[iStand]['Year'][-1]=Year+1

dmec[iStand]['ID Event Type'][-1]=meta['LUT']['Event']['Slashpile Burn']

dmec[iStand]['Mortality Factor'][-1]=100

dmec[iStand]['Growth Factor'][-1]=9999

# Add planting

for k in dmec[iStand].keys():

dmec[iStand][k]=np.append(dmec[iStand][k],9999)

dmec[iStand]['Year'][-1]=Year+2

dmec[iStand]['ID Event Type'][-1]=meta['LUT']['Event']['Planting']

dmec[iStand]['Mortality Factor'][-1]=0

dmec[iStand]['Growth Factor'][-1]=9999

return dmec

#%% Reduce number of growth curves by adjusting site index

def ReduceVariationInSiteIndex(meta,pNam,lsat):

trig=0

for i in range(1,55,2):

ind=np.where(lsat['SI']==i)[0]

if trig==0:

lsat['SI'][ind]=lsat['SI'][ind]+1

trig=1

else:

lsat['SI'][ind]=lsat['SI'][ind]-1

trig=0

return lsat

#%% Ensure every stand has a modern disturbance event

def Ensure\_Every\_Stand\_Has\_Modern\_Disturbance(meta,pNam,dmec,name\_dist,severity):

for iStand in range(meta[pNam]['Project']['N Stand']):

if dmec[iStand]==None:

continue

if dmec[iStand]['Year'].size==0:

#print(iStand)

#break

r=np.random.randint(1700,2000)

dmec[iStand]['Year']=np.append(dmec[iStand]['Year'],r)

dmec[iStand]['ID Event Type']=np.append(dmec[iStand]['ID Event Type'],meta['LUT']['Event'][name\_dist])

dmec[iStand]['Mortality Factor']=np.append(dmec[iStand]['Mortality Factor'],np.array(severity,dtype='int16'))

dmec[iStand]['Growth Factor']=np.append(dmec[iStand]['Growth Factor'],np.array(0,dtype='int16'))

if 'FCI Funded' in dmec[iStand]:

dmec[iStand]['FCI Funded']=np.append(dmec[iStand]['FCI Funded'],np.array(0,dtype='int16'))

for v in meta['Core']['StringsToFill']:

dmec[iStand][v]=np.append(dmec[iStand][v],9999)

return dmec

#%% Generate DMEC from estimate of stand age

def GapFill\_DMEC\_WithAge(meta,pNam,dmec,lsat):

for iStand in range(meta[pNam]['Project']['N Stand']):

if dmec[iStand]==None:

continue

Age=np.minimum(300,lsat['Age'])

# if dmec[iStand]['Mortality Factor'].size>0:

# if np.max(dmec[iStand]['Mortality Factor']==100):

# # If there is a stand-replacing disturbance on record, no need to proceed

# continue

# Allowing gap-filling of really young stands can be very problematic

th=30

if Age[iStand]>=th:

r=np.random.random(1)

if r<0.33:

type=meta['LUT']['Event']['Wildfire']

else:

type=meta['LUT']['Event']['Mountain Pine Beetle']

for k in dmec[iStand].keys():

dmec[iStand][k]=np.append(dmec[iStand][k],9999)

dmec[iStand]['Year'][-1]=meta[pNam]['Project']['Year Project']-Age[iStand]

dmec[iStand]['ID Event Type'][-1]=type

dmec[iStand]['Mortality Factor'][-1]=np.array(100,dtype='int16')

dmec[iStand]['Growth Factor'][-1]=9999

return dmec