Ecography

ECOG-03491
Miller, A. D., Thompson, J. R., Tepley, A. J. and Anderson-Teixeira, K. J. 2018. Alternative stable equilibria and critical thresholds created by fire regimes and plant responses in a fire-prone community. – Ecography doi: 10.1111/ecog.03491

Supplementary material

Appendix 1: Supplementary Figures

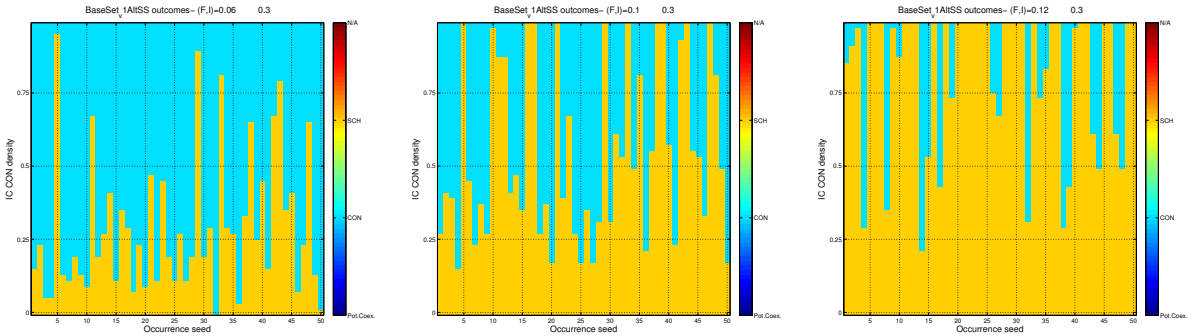

Figure A.1: Competitive outcomes for specific initial conditions and fire occurrence processes, within a region of alternative stable equilibria . All parameters are as in Fig. 3. Changes to fire frequency from low (left panel) to high (right panel) change the relative likelihood of dominance. Generally, point near the center of the alternative stable equilibria region will be least predictable, while points in the alternative stable equilibria region that are close to a boundary with a conifer dominance region will have conifer more likely to persist.

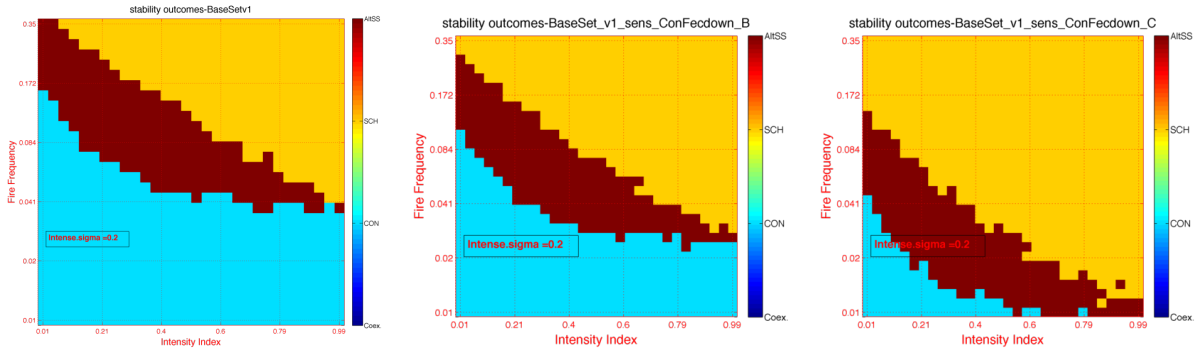

Figure A.2: Effect of decreasing conifer fecundity. Left panel as in Fig. 3. In all appendix figures, axes labels use abbreviations CON=conifer, SCH=shrubland, AltSS=alternative stable equilibria/alternative stable states, Coex.=coexistence.

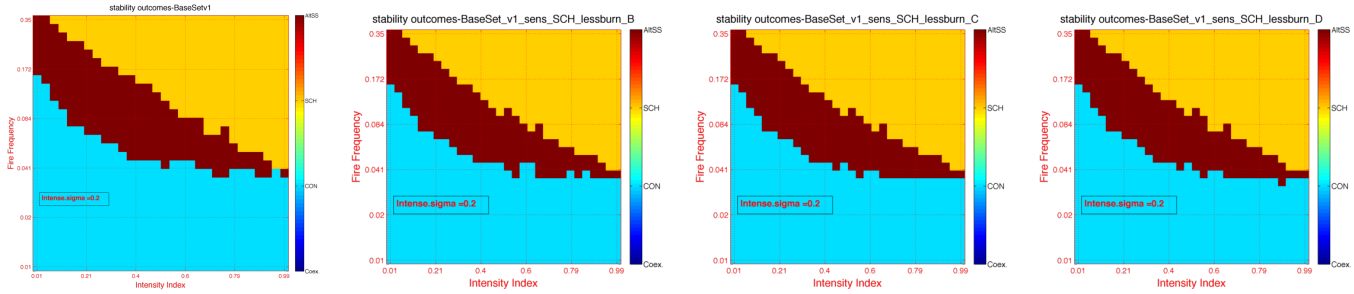

Figure A.3: Effect of increasing survival rate left to right. Shrub communities with a higher percentage of resprouting species would have a higher mean survival rate. Left panel as in Fig. 3.

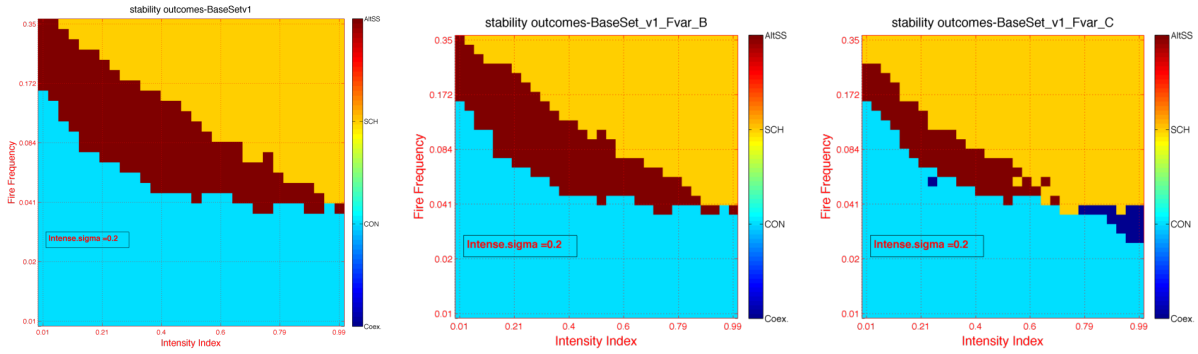

Figure A.4: Effect of increasing *independent* fluctuation in fecundity (an example of niche separation). Generally promotes coexistence, but requires very high levels.

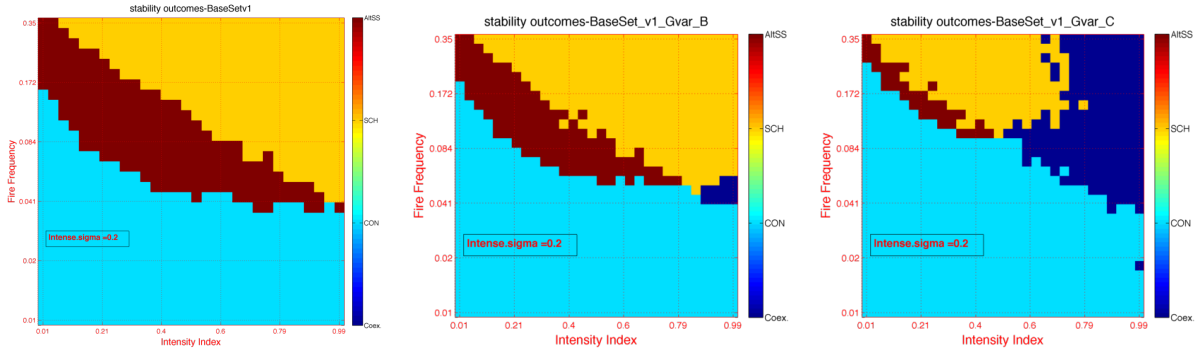

Figure A.5: Effect of increasing *independent* fluctuation in Germination rate (an example of niche separation). Generally promotes coexistence.

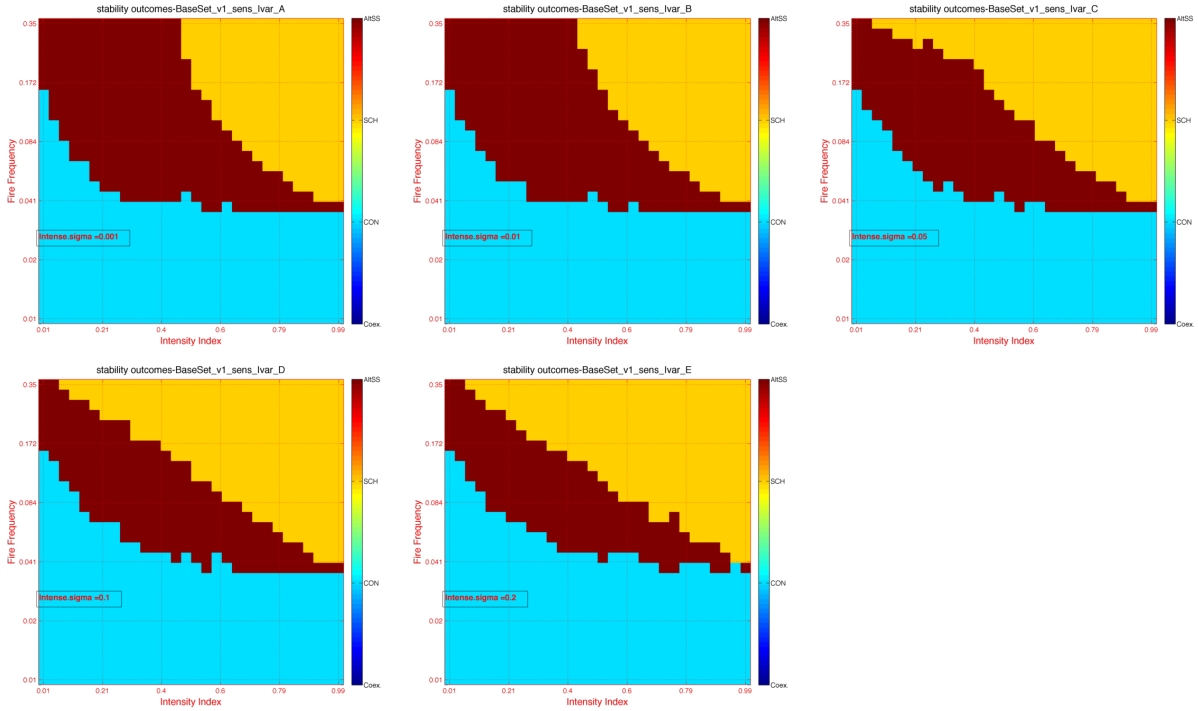

Figure A.6: Effect of increasing variation in intensity σ_I left to right, top to bottom. This decreases the overall area of the alternative stable equilibria region, which is mostly turned in to forest collapse by higher variation.

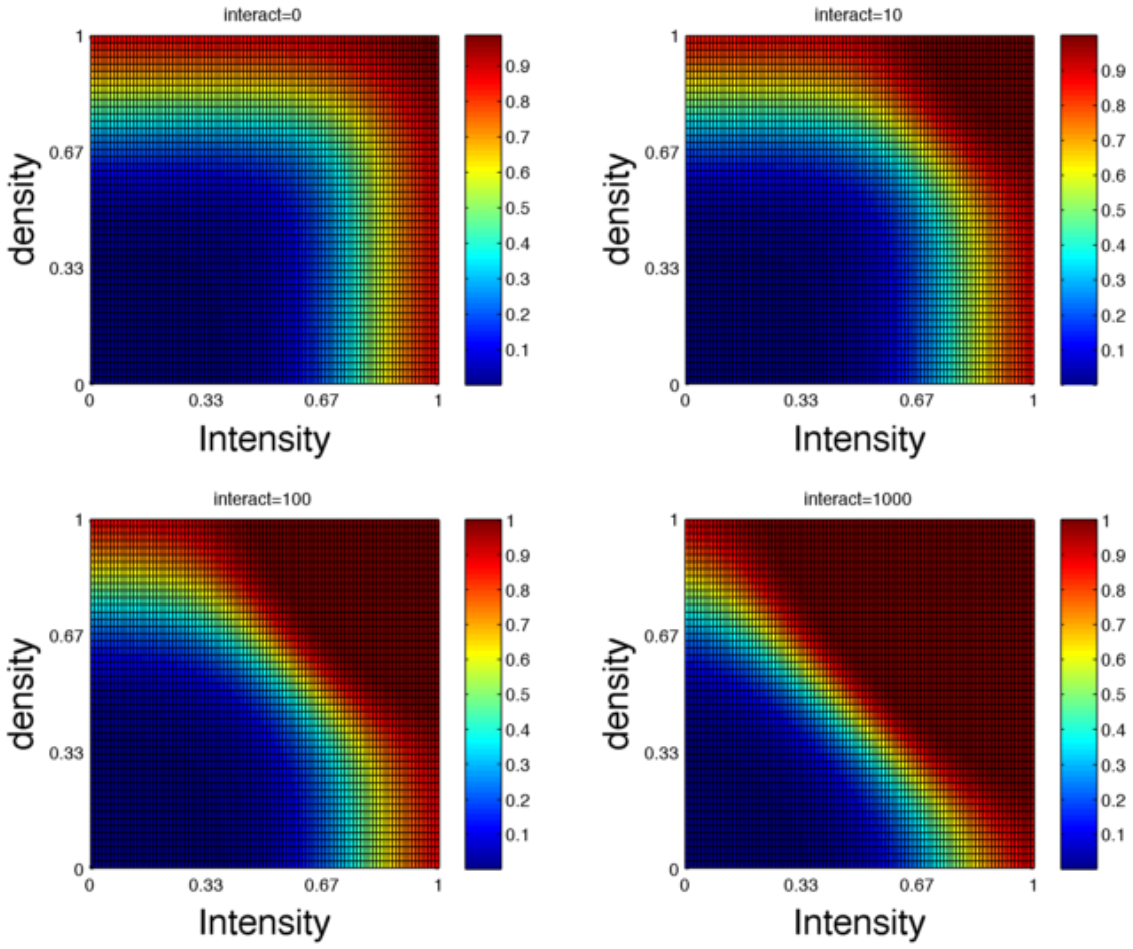

Figure A.7: Interaction strength: top-level parameter that governs how much the ladder effect interacts with the intensity effect in determining conifer mortality.

Appendix 2: Supplementary tables

Klamath community feature	**Model design element**
Fire mortality is highly variable; regime is classified as mixed severity (Halofsky et al. 2011).	Fire regimes have independent frequency and intensity, intensity variation is a free parameter.
Fire mortality depends on size class, PFT, and fire intensity (Odion et al. 2010; Thompson et al. 2011).	Each PFT and size has an independent sigmoidal functional response to the fire intensity index, with separate parameters controlling the steepness of the change and the intensity of the inflection point. Sigmoid response as defined by Eq. 9
Most shurb species recruit well after fire, due to long-lived seedbanks or re-sprouting structures (Donato et al. 2009).	shrubs have high seed longevity. Re-sprouting is not explicitly modeled, seedbank longevity is increased as a proxy.
Most conifer species do not recruit well after fires, due to dispersal limitations, competition from shrubemergence, and a short-lived seedbank. (Thompson and Spies 2009; Odion et al. 2010)	Dispersal is not explicitly modeled. Conifers have very low seed longevity, and thus do not recruit well after fire in the model, due to competition with shrub species.
Understory, shade-intolerant shrub species are excluded as forest canopy closes (Azuma et al. 2004).	Mature shrub non-fire mortality increases with mature conifer density to represent canopy closure, sigmoid response as defined by Eq. 11
Shrubs and small conifers act as ladder fuels (Thompson and Spies 2009)	Mature conifer fire mortality depends on density of mature shruband conifer saplings, response as defined by Eq. 10

Table A.1: Model features designed to incorporate important plant biology and fire dynamics of the Klamath community.

symbol	related	description
N_{jt}	$\left(S_{jt}, J_{jt} M_{jt}\right)$	Density vector (seeds, juvenile, mature)
Λ_{jt}	$\lambda_{(\cdot)}$	time evolution matrix for type j at time t
$\tilde{\lambda}_i$		long-term, low-density growth rate of each PFT i acting as invader
G_{jt}	σ_G	Germination rate and variation
f_{jt}	σ_f	Fecundity and variation
T_j		Size transition rate from juvenile to mature for type j.
δ_{jkt}		Mortality rate for type j size k at time t
m_{jk}	$m_{\cdot,\cdot,l},\ m_{\cdot,\cdot,s}$	base mortality, location and steepness parameters
$S_l,\ S_s$	$L_l,\ L_s$	location and steepness parameters for shading and ladder fuel effect
W_{Ljk}		Ladder weight for type j size k
W_{Sjk}		Shading weight for type j size k
I_t	μ_I, σ_I	Fire Intensity index, mean and variation
F		Mean fire frequency
Int		Interaction strength of ladder fuel and intensity index

Table A.2: List of symbols.

Appendix 3: Computation information

We implement the procedures described in section "Formal Model Specification", using MATLAB R2012a (The MathWorks Inc., Natick, MA, 2012), with source code provided below. First, we generate all stochastic processes for fire occurrence and variation in germination and fecundity, constructing a 30x30 grid of fire regimes across a 0.01-0.99 range of intensity indices and 0.01-0.35 range of frequencies.

Under each of the 900 combinations of fire frequency and intensity fire regimes, we iterate the dynamics of each PFT by itself for 10,000 years, to get the competitive effects of each PFT when treated as an established resident. Using the same stochastic processes, we compute $\tilde{\lambda}_i$ for each PFT as an invader by finding its growth rate at each time step, and then taking the mean over time, excluding the first 100 time steps to minimize the influence of any resident transient dynamics. In this manner, we construct the figures and our quantitative results presented in the main text.

Klamath_fire_shrub_stability_analysis_MAIN.m

```
%% Klamath model, Miller et al 2018, Ecography
%% This script runs all the other code.
clear
tic

%% Load new inputs from script if desired
%run declare_inputs

%% Load saved inputs if desired
load('BaseSetv1.mat') % best to set up a directory structure for inputs/outputs/...
    figure/code etc.
mo.freq_sample_flag='lin'; % sets the sample rate on frequency, 'log' or 'lin

%% start up a pool of four cores if one is not yet open
% if matlabpool('size')==0, %%% This syntax is deprecated, but left for older ...
    versions of matlab
% matlabpool(4)%
% else
% end
if isempty(gcp('nocreate'))
parpool(4)
end
rng(mo.SD) % set seed
%% Generate germination and fecundity
display('generating germ rates and fecundities')
[germs,fecs]=germ_fec_gen_func(mo,pf);

%% Generate fire process
display('generating fire')
```

```
[occ,intensity]=fire_gen(mo); % depends on "model" params only, i.e. environmental ...
    stuff not PFT params

%% compute sigmoidal mortality response
display('seeing what burns...')
[mort_raw]=mort_response_func(mo,pf,intensity,occ);

%% generate resident distributions
display('generating resident distributions')
%[resident_densities]=res_dens_func(mo,pf,mort_raw,germs,fecs);
[resident_densities]=density_dynamics_func(mo,pf,mort_raw,germs,fecs,occ,1); % ...
    final slot is flag, 1-> resident dynamics, 0-> patch dynamics

%% Modify mortality for density feedbacks
[mort_fedback]=mortality_feedbacks_func(resident_densities,mort_raw,occ,pf);

%% Assign Lambdas
display('Arranging invader Lambdas')
[Lambda]=Lambda_assigner(mo,pf,germs,fecs,mort_fedback); % this does NOT assign ...
    Lambda(:,2,1,...), the competition term
    free_space=NaN(size(fecs));
    avail_seed=NaN(size(fecs));
    free_space(1,:,:,:)=squeeze(sum(mort_fedback(1,2:3,:,:,:).*resident_densities...
        (1,2:3,:,:,:),2));
    free_space(2,:,:,:)=squeeze(sum(mort_fedback(2,2:3,:,:,:).*resident_densities...
        (2,2:3,:,:,:),2));
    avail_seed(1,:,:,:)=squeeze(germs(1,:,:,:)).*squeeze(resident_densities...
        (1,1,:,:,:));
    avail_seed(2,:,:,:)=squeeze(germs(2,:,:,:)).*squeeze(resident_densities...
        (2,1,:,:,:));
    % assign comp
    competition=avail_seed./free_space;
    clear avail_seed free_space
```

```
    %assign comp lambdas
    Lambda(1,2,1,:,:,:)=germs(1,:,:,:)./competition(2,:,:,:); % watch the switch-a-...
        roo! Lambda 1 takes germ 1 and comp 2, because PFT 1 only competes with 2 ...
        when 1 is invader
    Lambda(2,2,1,:,:,:)=germs(2,:,:,:)./competition(1,:,:,:);
    clear competition

%% All set, now find eigenvalues
display('Computing eigenvalues')

dom_eigs=NaN(2,mo.f_steps,mo.i_steps,mo.sim_time);

    for step_ind2=1:mo.i_steps
        parfor t_ind=1:mo.sim_time

            dom_eigs(1,:,step_ind2,t_ind)=max(eig3(squeeze(Lambda(1,:,:,:,step_ind2...
                ,t_ind))));
        end
    end

    for step_ind2=1:mo.i_steps
        parfor t_ind=1:mo.sim_time
            dom_eigs(2,:,step_ind2,t_ind)=max(eig3(squeeze(Lambda(2,:,:,:,step_ind2...
                ,t_ind))));
        end
    end
display('done with eigenvalues')

Lambda_tildes=squeeze(mean(dom_eigs(:,:,:,floor(0.01*end):end-1),4)); %throw out ...
    first 1%! this should allow resident dists to settle down

clear dom_eigs
```

```
clear Lambda
clear resident_densities
%% compute outcomes from Lambda_tildes
outcomes.coex=double(squeeze(Lambda_tildes(1,:,:)>1)&squeeze(Lambda_tildes(2,:,:)...
    >1));
outcomes.ASS=double(squeeze(Lambda_tildes(1,:,:)<1)&squeeze(Lambda_tildes(2,:,:)<1)...
    );
outcomes.CONwin=double(squeeze(Lambda_tildes(1,:,:)<1)&squeeze(Lambda_tildes(2,:,:)...
    >1));
outcomes.SCHwin=double(squeeze(Lambda_tildes(1,:,:)>1)&squeeze(Lambda_tildes(2,:,:)...
    <1));

outcome=outcomes.coex+2*outcomes.CONwin+3*outcomes.SCHwin+4*outcomes.ASS; % so 1-> ...
    coex, 2->Conwin, etc

%% run patch dynamics, if desired.
%display('starting patch dynamics')
%[patch_densities]=density_dynamics_func(mo,pf,mort_raw,germs,fecs,occ,0); % 0 ...
    means use shared comp for patch dynamics

warning('off','MATLAB:Figure:SetPosition') % turn off warnings for positioning ...
    docked figs

run plot_stability_outcomes
%export_fig(['./figures/OutcomePlots/',name,'SA_outcomes.png'],'-transparent')
%close gcf

%run plot_outcome_diffs_DD_stability
%export_fig(['./figures/OutcomeDiffs/',name,'outcome_diff.png'],'-transparent')
%close gcf

toc
beep
```

```
display('done with simulation')


% E.g. saving figs if desireed
%save(['./figures/Output/',name],'outcome','mo','pf')
%save(['./figures/Output/',name],'outcome','outcome_DD','mo','pf')
```

declare_inputs.m

```
%% Script for declaring model inputs, including simulation configuration, ...
    disturbance stuff, and PFT life history paramters.
% Klamath model, Miller et al 2018, Ecography

%% Model parameters
% general stuff
name='Example_name'
mo.SD=10;
mo.sim_time=10000;
mo.i_steps=30; % # steps for intensity mean
mo.f_steps=30; % # steps for frequency
% window on (i,f) plane
mo.f_start=0.01;
mo.f_stop=0.35;
mo.i_start=0.01;
mo.i_stop=0.99;
%
mo.i_var=.20; % sigma controlling variation in intensity
%
mo.germ_fec_flag='same'; % only 'same' for now
mo.freq_sample_flag='log'; % log or lin

%% PFT parameters
% PFT 1 is SCH, PFT 2 is CON
pf.fec_base=[30000,30000];
pf.germ_rate=[0.4,0.9];
pf.max_germ=0.9999; % seems sensible to avoid germ_rate=1

% sigmas
pf.fec_sigma=[0.2 0.2];
```

```
pf.germ_sigma=[0.3, 0.1]; % use these to generate IID lognormal germ rates
%
pf.size_transition=[0.5, 0.025];

% mort stuff
pf.mort_rate_base=[0.001 .35, .25;
                0.95 0.08,0.015];% uses convention (PFT,size),
pf.mort_curve_ramps=[NaN 0.1, 0.25
                 NaN  0.1, 0.5]; % these are the intensity locations where the ...
                     sigmoid mort response has max slope
pf.mort_curve_steepness=   [NaN 22, 15
                         NaN 15, 7]; % this controlls the slope at the steepest ...
                             part of sigmoid mort

% ladder/smother
pf.ladder_steepness=11; % this is the steepness of the ladder fuel effect of SCH on...
     CON mortality in fire
pf.ladder_ramp=0.35; % this is the inflection point for the ladder fuel effect. ...
    should probably be higher for 2 than 1
pf.ladder_wt=[NaN 0 0.5;
              NaN .5 0]; %(pft,size) indexing



pf.smother_steepness=6;
pf.smother_ramp=0.65;
pf.smother_wt=[NaN 0 0;
               NaN .4 .6];

pf.interaction_strength=100; % for ladder x intensity interaction

% good idea to save inputs for future loading/reference
%save(['./inputs/',name])
```

density_dynamics_func.m

```
function [patch_densities]=density_dynamics_func(mo,pf,mort_raw,germs,fecs,occ,...
    resident_flag)
% Computes desnisty dynamics
% Klamath model, Miller et al 2018, Ecography
[Lambda_res]=Lambda_assigner(mo,pf,germs,fecs,mort_raw);
%initialize densities
patch_densities=NaN(2,3,mo.f_steps,mo.i_steps,mo.sim_time);
% assign initial densities
patch_densities(1,:,:,:,1)=repmat([100000,1/2,1/2]',[1,mo.f_steps,mo.i_steps]);
patch_densities(2,:,:,:,1)=repmat([100000,1/2,1/2]',[1,mo.f_steps,mo.i_steps]);

mort_cur=mort_raw;
free_space=NaN(2,mo.f_steps,mo.i_steps);
avail_seed=NaN(2,mo.f_steps,mo.i_steps);

if resident_flag==1
display('starting resident loop')
elseif resident_flag==0
    display('starting patch dynamics loop')
else
error('resident_flag not properly set!')
end
for t_ind=1:mo.sim_time-1

%% competition always depends on density for all cases
    if resident_flag==1,
        %% NOT ALL LADDER/SMOTHER feedbacks go into effect here, but some do...
        fireplaces=squeeze(logical(occ(:,:,t_ind)));
        ladder_density...% ONLY PFT 2=CON affects CON res
            =squeeze(pf.ladder_wt(2,3).*patch_densities(2,3,:,:,t_ind)...
```

```
            +pf.ladder_wt(2,2).*patch_densities(2,2,:,:,t_ind));

sig1=squeeze(mort_raw(2,3,fireplaces));
sig2=squeeze(sigmf(ladder_density(fireplaces),[pf.ladder_steepness,...
    pf.ladder_ramp]));
% assign mort
mort_cur(2,3,fireplaces)= 1-(1-sig1).*(1-sig2).*(1-sig1.*sig2).^...
    pf.interaction_strength;

%now smother
smother_density... %ONLY PFT1=SCH smothers SCH res
    =squeeze(pf.smother_wt(1,3).*patch_densities(1,3,:,:,t_ind)...
            +pf.smother_wt(1,2).*patch_densities(1,2,:,:,t_ind));

mort_cur(1,3,:,:,t_ind)=(1-squeeze(mort_raw(1,3,:,:,t_ind)))...
                        .*(sigmf(smother_density,...
                        [pf.smother_steepness,pf.smother_ramp]))...
                        +squeeze(mort_raw(1,3,:,:,t_ind));


% reassign lambdas for new mort(:,3)
Lambda_res(1,3,3,:,:,t_ind)=squeeze(1-mort_cur(1,3,:,:,t_ind));
Lambda_res(2,3,3,:,:,t_ind)=squeeze(1-mort_cur(2,3,:,:,t_ind)); % 2 mature ...
    persists

%% for resident densities, competition is different for each PFT, it only ...
    sees itself.
free_space(1,:,:)=squeeze(mort_cur(1,2,:,:,t_ind).*...
    patch_densities(1,2,:,:,t_ind)+mort_cur(1,3,:,:,t_ind).*...
    patch_densities(1,3,:,:,t_ind));
free_space(2,:,:)=squeeze(mort_cur(2,2,:,:,t_ind).*...
    patch_densities(2,2,:,:,t_ind)+mort_cur(2,3,:,:,t_ind).*...
    patch_densities(2,3,:,:,t_ind));
```

```
    avail_seed(1,:,:)=squeeze(germs(1,:,:,t_ind)).*...
        squeeze(patch_densities(1,1,:,:,t_ind));
    avail_seed(2,:,:)=squeeze(germs(2,:,:,t_ind)).*...
        squeeze(patch_densities(2,1,:,:,t_ind));
    comp_res=avail_seed./free_space;
    % assign lambda
    Lambda_res(1,2,1,:,:,t_ind)=squeeze(germs(1,:,:,t_ind))./squeeze(comp_res...
        (1,:,:));
    Lambda_res(2,2,1,:,:,t_ind)=squeeze(germs(2,:,:,t_ind))./squeeze(comp_res...
        (2,:,:));

    % compute next time step
    patch_densities(1,:,:,:,t_ind+1)=multiprod(squeeze...
        (Lambda_res(1,:,:,:,:,t_ind)),...
        squeeze(patch_densities(1,:,:,:,t_ind)),[1 2], 1);
    patch_densities(2,:,:,:,t_ind+1)=multiprod(squeeze...
        (Lambda_res(2,:,:,:,:,t_ind)),...
        squeeze(patch_densities(2,:,:,:,t_ind)),[1 2], 1);

elseif resident_flag==0,
    %% ALL ladder/smother effects work here
    %% DO LADDER/SMOTHER MODS
    fireplaces=logical(occ(:,:,t_ind));
    ladder_density...
        =squeeze(pf.ladder_wt(1,3).*patch_densities(1,3,:,:,t_ind)...
                +pf.ladder_wt(1,2).*patch_densities(1,2,:,:,t_ind)...
                +pf.ladder_wt(2,3).*patch_densities(2,3,:,:,t_ind)...
                +pf.ladder_wt(2,2).*patch_densities(2,2,:,:,t_ind));

    sig1=squeeze(mort_raw(2,3,fireplaces));
    sig2=squeeze(sigmf(ladder_density(fireplaces),[pf.ladder_steepness,...
        pf.ladder_ramp]));
```

```
mort_cur(2,3,fireplaces)= 1-(1-sig1).*(1-sig2).*(1-sig1.*sig2).^...
    pf.interaction_strength;
%now smother
smother_density...
    =squeeze(pf.smother_wt(1,3).*patch_densities(1,3,:,:,t_ind)...
            +pf.smother_wt(1,2).*patch_densities(1,2,:,:,t_ind)...
            +pf.smother_wt(2,3).*patch_densities(2,3,:,:,t_ind)...
            +pf.smother_wt(2,2).*patch_densities(2,2,:,:,t_ind));
mort_cur(1,3,:,:,t_ind)=(1-squeeze(mort_raw(1,3,:,:,t_ind)))...
                        .*(sigmf(smother_density,[pf.smother_steepness,...
                            pf.smother_ramp]))...
                        +squeeze(mort_raw(1,3,:,:,t_ind));
% reassign lambdas for new mort(:,3)
Lambda_res(1,3,3,:,:,t_ind)=squeeze(1-mort_cur(1,3,:,:,t_ind));
Lambda_res(2,3,3,:,:,t_ind)=squeeze(1-mort_cur(2,3,:,:,t_ind)); % 2 mature ...
    persists


%% same as above, except free space/competition are SHARED
f_s=squeeze(mort_cur(1,2,:,:,t_ind).*patch_densities(1,2,:,:,t_ind)...
    +mort_cur(1,3,:,:,t_ind).*patch_densities(1,3,:,:,t_ind))...
    +squeeze(mort_cur(2,2,:,:,t_ind).*patch_densities(2,2,:,:,t_ind)...
    +mort_cur(2,3,:,:,t_ind).*patch_densities(2,3,:,:,t_ind));

avail_seed =squeeze(germs(1,:,:,t_ind)).*squeeze(patch_densities(1,1,:,:,...
    t_ind))...
    +squeeze(germs(2,:,:,t_ind)).*squeeze(patch_densities(2,1,:,:,t_ind));
comp=avail_seed./f_s;
% assign lambda
Lambda_res(1,2,1,:,:,t_ind)=squeeze(germs(1,:,:,t_ind))./comp;
Lambda_res(2,2,1,:,:,t_ind)=squeeze(germs(2,:,:,t_ind))./comp;

% compute next time step
```

```
        patch_densities(1,:,:,:,t_ind+1)=multiprod(squeeze(...
            Lambda_res(1,:,:,:,:,t_ind)),...
            squeeze(patch_densities(1,:,:,:,t_ind)),[1 2], 1);
        patch_densities(2,:,:,:,t_ind+1)=multiprod(squeeze...
            (Lambda_res(2,:,:,:,:,t_ind)),...
            squeeze(patch_densities(2,:,:,:,t_ind)),[1 2], 1);


    else
        error('resident flag not set properly!')
    end

end

end
```

eig3.m

```
function D = eig3(A)
% function D = eig3(A)
%
% Compute in one shot the eigen-values of multiples (3 x 3) matrices
%
% INPUT:
%   A: (3 x 3 x n) array
% OUTPUT:
%   D: (3 x n). EIG3 returns in D(:,k) three eigen-values of A(:,:,k)
%
% See also: CardanRoots, eig2, eig
%
% Author: Bruno Luong <brunoluong@yahoo.com>
% History:
%     Original 20-May-2010

if size(A,1) ~= 3 || size(A,2) ~= 3
    error('A must be [3x3xn] array');
end

A = reshape(A, 9, []).';

P3 = 1;
% Trace
P2 = -(A(:,1)+A(:,5)+A(:,9));
% Principal minors
M11 = A(:,5).*A(:,9) - A(:,8).*A(:,6);
M22 = A(:,9).*A(:,1) - A(:,3).*A(:,7);
M33 = A(:,1).*A(:,5) - A(:,4).*A(:,2);
P1 = (M11 + M22 + M33);
```

```
% Determinant
P0 = - A(:,1).*M11 ...
     + A(:,4).*(A(:,2).*A(:,9)-A(:,8).*A(:,3)) ...
     - A(:,7).*(A(:,2).*A(:,6)-A(:,5).*A(:,3));


D = CardanRoots(P3, P2, P1, P0).';


end
```

fire_gen.m

```
function [occ,intensity]=fire_gen(mo)
%% Generates stochastic time series for fire occurence and frequency
% Klamath model, Miller et al 2018, Ecography

if strcmp(mo.freq_sample_flag,'lin')
    Freq_steps=linspace(mo.f_start,mo.f_stop,mo.f_steps);
elseif strcmp(mo.freq_sample_flag,'log')
    Freq_steps=logspace(log10(mo.f_start),log10(mo.f_stop),mo.f_steps);
else
    error('freq_sample_flag not properly specified')
end
int_mean_steps=linspace(mo.i_start,mo.i_stop,mo.i_steps);
lognormal_means=repmat(log(...
    int_mean_steps.^2./sqrt(mo.i_var+(int_mean_steps.^2))),[mo.f_steps,1,...
        mo.sim_time]);
lognormal_vars=repmat(sqrt(log...
    (mo.i_var./int_mean_steps.^2 + 1)),[mo.f_steps,1,mo.sim_time]);

%  now use vars that are properly scaled to mean!
intensity=lognrnd(lognormal_means,lognormal_vars);
% this is the IID 0-1 array for fire occurence with probability=Frequency
occ=squeeze(binornd(1,repmat(Freq_steps,[1,1,mo.i_steps,mo.sim_time])));

end
```

germ_fec_gen_func.m

```
function [germs,fecs]=germ_fec_gen_func(mo,pf)
%% Generates variatino in germination and fecundity
% Klamath model, Miller et al 2018, Ecography

germs=NaN(2,mo.f_steps,mo.i_steps,mo.sim_time);

if strcmp(mo.germ_fec_flag,'same')

germ1=lognrnd(log(pf.germ_rate(1)),pf.germ_sigma(1),[1,mo.sim_time]);
germ2=lognrnd(log(pf.germ_rate(2)),pf.germ_sigma(2),[1,mo.sim_time]);

germs(1,:,:,:)=permute(squeeze(repmat(germ1,[1,1,mo.i_steps,mo.f_steps])),[3,2,1]);
germs(2,:,:,:)=permute(squeeze(repmat(germ2,[1,1,mo.i_steps,mo.f_steps])),[3,2,1]);

germs=min(germs,pf.max_germ);
% so now these go germs(PFT,f_step,i_step,t_ind)

fec1=lognrnd(log(pf.fec_base(1)),pf.fec_sigma(1),[1,mo.sim_time]);
fec2=lognrnd(log(pf.fec_base(2)),pf.fec_sigma(2),[1,mo.sim_time]);

fecs(1,:,:,:)=permute(squeeze(repmat(fec1,[1,1,mo.i_steps,mo.f_steps])),[3,2,1]);
fecs(2,:,:,:)=permute(squeeze(repmat(fec2,[1,1,mo.i_steps,mo.f_steps])),[3,2,1]);

else
    error('germ fec flag not properly specified!')

end
end
```

Lambda_assigner.m

```
function [Lambda]=Lambda_assigner(mo,pf,germs,fecs,mort_raw)
% Assign values to the primary Lambda matrix.
% Klamath model, Miller et al 2018, Ecography
LBD=NaN(2,3,3,mo.f_steps,mo.i_steps,mo.sim_time);
%% zero out all zeros
LBD(:,1,2,:,:,:)=0;
LBD(:,2,3,:,:,:)=0;
LBD(:,3,1,:,:,:)=0; % 3x2= 6 zeros

% assign all Lambda that NEVER  depend on density
LBD(1,1,3,:,:,:)=fecs(1,:,:,:);
LBD(2,1,3,:,:,:)=fecs(2,:,:,:); % 2 fecs

LBD(1,1,1,:,:,:)=squeeze((1-mort_raw(1,1,:,:,:))).*squeeze((1-germs(1,:,:,:)));
LBD(2,1,1,:,:,:)=squeeze((1-mort_raw(2,1,:,:,:))).*squeeze((1-germs(2,:,:,:))); % 2...
      seedbank persists

% assign Lambda that do NOT depend on density
%if there are no ladder or smother effects. These will have to be upated
LBD(1,3,3,:,:,:)=1-mort_raw(1,3,:,:,:);
LBD(2,3,3,:,:,:)=1-mort_raw(2,3,:,:,:); % 2 mature persists

LBD(1,2,2,:,:,:)=(1-mort_raw(1,2,:,:,:)).*(1-pf.size_transition(1));
LBD(2,2,2,:,:,:)=(1-mort_raw(2,2,:,:,:)).*(1-pf.size_transition(2)); %2 juv ...
    persists

LBD(1,3,2,:,:,:)=(1-mort_raw(1,2,:,:,:)).*(pf.size_transition(1));
LBD(2,3,2,:,:,:)=(1-mort_raw(2,2,:,:,:)).*(pf.size_transition(2)); %2 juv grow ups
```

```
Lambda=LBD;
%% NB does NOT do competition assignement, that will almost always be state ...
    dependent. This way we can use this function in other places too
```

mort_response_func.m

```
function [mort_raw]=mort_response_func(mo,pf,intensity,occ)
% generates first-pass mortality response, based only on fire intensity and
% sigmoidal PFT reaction
% Klamath model, Miller et al 2018, Ecography

morts=repmat(pf.mort_rate_base,[1,1,mo.f_steps,mo.i_steps,mo.sim_time]);

for pft=1:2
    for size=2:3

       morts(pft,size,logical(occ))=sigmf(intensity(logical(occ)),...
           [pf.mort_curve_steepness(pft,size),pf.mort_curve_ramps(pft,size)]);
    end
end
mort_raw=morts;
end
```

mortality_feedbacks_func.m

```
function [mort_fedback]=mortality_feedbacks_func(resident_densities,mort_in,occ,pf)
% Modifies mortality based on PFT size class densities, i.e. ladder fuels
% and shading/smothering of SH by CON
%% Klamath model, Miller et al 2018, Ecography
mort_fedback=mort_in;
% compute the effective "ladder density" - put all aboveground types
% together and weight them appropriately. Mostly (1,2) and (2,3) will be
% zero, but this way we can play with them too
ladder_densities...
        =squeeze(pf.ladder_wt(1,3).*resident_densities(1,3,:,:,:)...
                +pf.ladder_wt(1,2).*resident_densities(1,2,:,:,:)...
                +pf.ladder_wt(2,3).*resident_densities(2,3,:,:,:)...
                +pf.ladder_wt(2,2).*resident_densities(2,2,:,:,:));

sig1=squeeze(mort_in(2,3,logical(occ)));
sig2=squeeze(sigmf(ladder_densities(logical(occ)),[pf.ladder_steepness,...
    pf.ladder_ramp]));

%
mort_fedback(2,3,logical(occ))=1-(1-sig1)...
                               .*(1-sig2)...
                               .*(1-sig1.*sig2).^pf.interaction_strength;
% now for smothering - just do this for ALL time, makes things easier also
% no interaction strength.
smother_densities...
        =squeeze(pf.smother_wt(1,3).*resident_densities(1,3,:,:,:)...
                +pf.smother_wt(1,2).*resident_densities(1,2,:,:,:)...
                +pf.smother_wt(2,3).*resident_densities(2,3,:,:,:)...
                +pf.smother_wt(2,2).*resident_densities(2,2,:,:,:));
```

```
mort_fedback(1,3,:,:,:)=(1-squeeze(mort_in(1,3,:,:,:))).*
    .*(sigmf(smother_densities,[pf.smother_steepness,pf.smother_ramp])).*
    +squeeze(mort_in(1,3,:,:,:));


end
```

plot_stability_outcomes.m

```
%% for plotting stability outcomes
% Klamath model, Miller et al 2018, Ecography
figure
ticks=6;
%%%%%%%%%%%%%%%%%%%%
% plot
image(flipud(outcome),'CDataMapping','scaled')
% handling
set(gca,'XTick',linspace(1,mo.i_steps,ticks))
set(gca,'Xticklabel',round(100*linspace(mo.i_start,mo.i_stop,ticks))/100)

if strcmp(mo.freq_sample_flag,'lin')
    set(gca,'YTick',linspace(1,mo.f_steps,ticks))
    set(gca,'Yticklabel',round(100*linspace(mo.f_stop,mo.f_start,ticks))/100) % ...
        image has a flipped axis system...
elseif strcmp(mo.freq_sample_flag,'log')
    set(gca,'YTick',(linspace(1,mo.f_steps,ticks)))
    set(gca,'Yticklabel',round(1000*logspace(log10(mo.f_stop),log10(mo.f_start),...
        ticks))/1000) % image has a flipped axis system...

else
    display('Frequency sampling not properly specified!')
end
set(gca, 'XColor', 'k')
set(gca, 'YColor', 'k')
caxis([1 4])
colormap(viridis(4)) % this is a good colormap, good contrast, prints well in ...
    grayscale, etc.
colorbar('YTick',[1, 2,3,4 ],'YTickLabel',{'COEX','CON','SH','ASE'})
```

```
xlabel('Intensity Index','FontSize',18)
ylabel('Fire Frequency','FontSize',18)
%title(['stability outcomes-',name],'FontSize',18,'Interpreter','none')
axis square
grid on
set(gca,'fontsize',13)
%%


% save fig if desired
%export_fig(['./figures/',name,'OutcomeLambda_tilda_stability.png'],'-transparent')
```

References

Azuma, D. L., Donnegan, J., and Gedney, D. 2004. Southwest Oregon Biscuit fire: an analysis of forest resources and fire severity. Technical Report Research Paper PNW-RP-560, US Department of Agriculture, Forest Service, Portland, OR.

Donato, D. C., Fontaine, J. B., Robinson, W. D., Kauffman, J. B., and Law, B. E. 2009. Vegetation response to a short interval between high-severity wildfires in a mixed-evergreen forest. *Journal of Ecology* 97:142–154.

Halofsky, J. E., Donato, D. C., Hibbs, D. E., Campbell, J. L., Cannon, M. D., Fontaine, J. B., Thompson, J. R., Anthony, R. G., Bormann, B. T., Kayes, L. J., Law, B. E., Peterson, D. L., and Spies, T. A. 2011. Mixed-severity fire regimes: lessons and hypotheses from the Klamath-Siskiyou ecoregion. *Ecosphere* 2:1–19. art40.

Odion, D. C., Moritz, M. A., and DellaSala, D. A. 2010. Alternative community states maintained by fire in the Klamath Mountains, USA. *Journal of Ecology* 98:96–105.

Thompson, J. R. and Spies, T. A. 2009. Vegetation and weather explain variation in crown damage within a large mixed-severity wildfire. *Forest Ecology and Management* 258:1684–1694.

Thompson, J. R., Spies, T. A., and Olsen, K. A. 2011. Canopy damage to conifer plantations within a large mixed-severity wildfire varies with stand age. *Forest Ecology and Management* 262:355–360.