International Journal of Wildland Fire 2018, **28**, 46–61
doi:10.1071/WF18031_AC

Supplementary material

Multi-scale assessment of post-fire tree mortality models

Tucker J. Furniss [A,B,E], *Andrew J. Larson*[C], *Van R. Kane*[D] *and James A. Lutz* [A,B]

[A]Wildland Resources Department, Utah State University, 5230 Old Main Hill, Logan, UT 84322, USA.

[B]The Ecology Center, Utah State University, 5205 Old Main Hill, Logan, UT 84322, USA.

[C]Department of Forest Management, University of Montana, Missoula, MT 59812, USA.

[D]School of Environmental and Forest Sciences, University of Washington, Box 352100, Seattle, WA 98195, USA.

[E]Corresponding author email: tucker.furniss@usu.edu

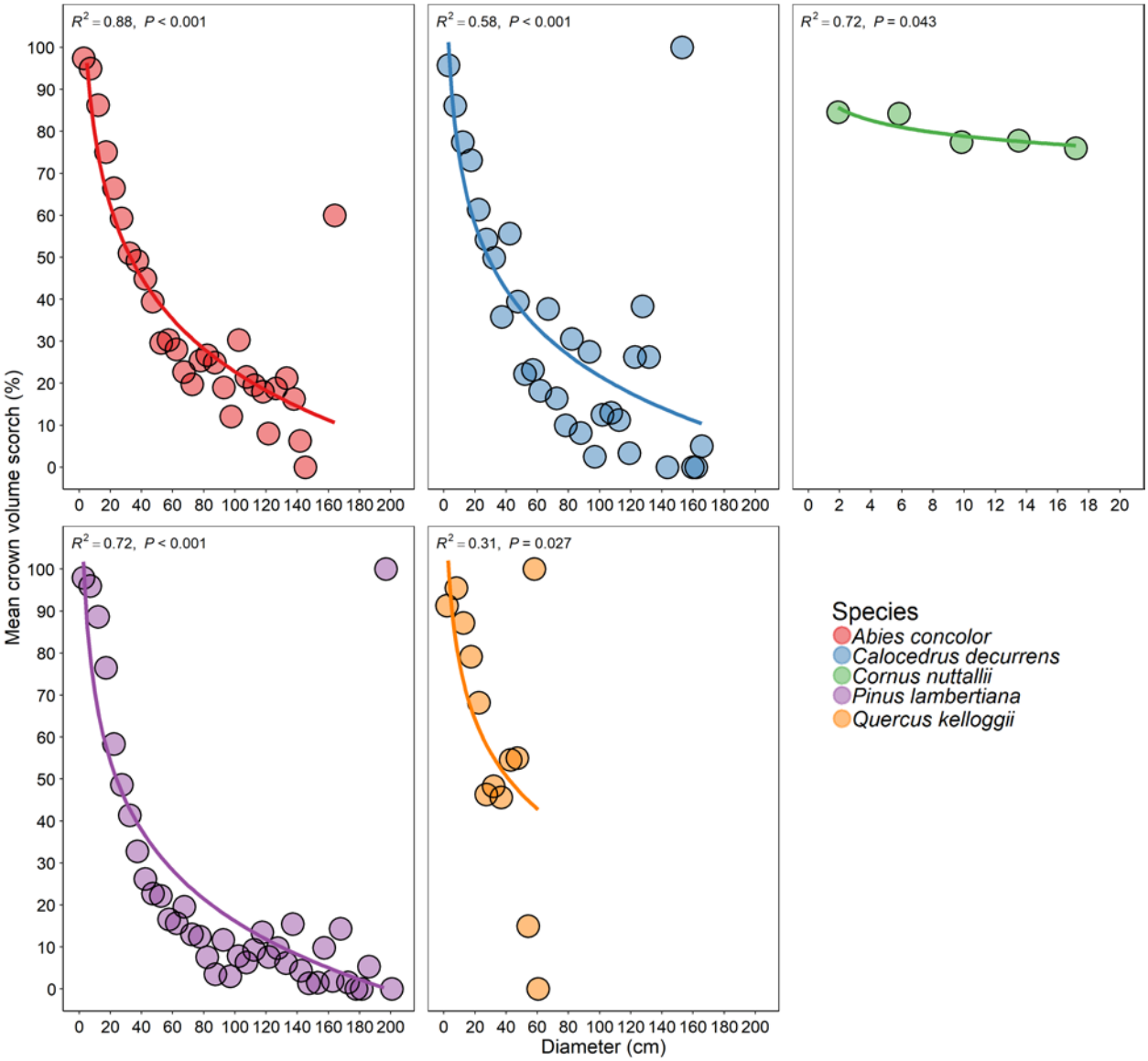

Fig. S1. The relationship between crown volume scorch and diameter at breast height for five species within the Yosemite Forest Dynamics Plot. Each dot represents a 5 cm diameter class (first dot 1 cm ≤ DBH < 5 cm).

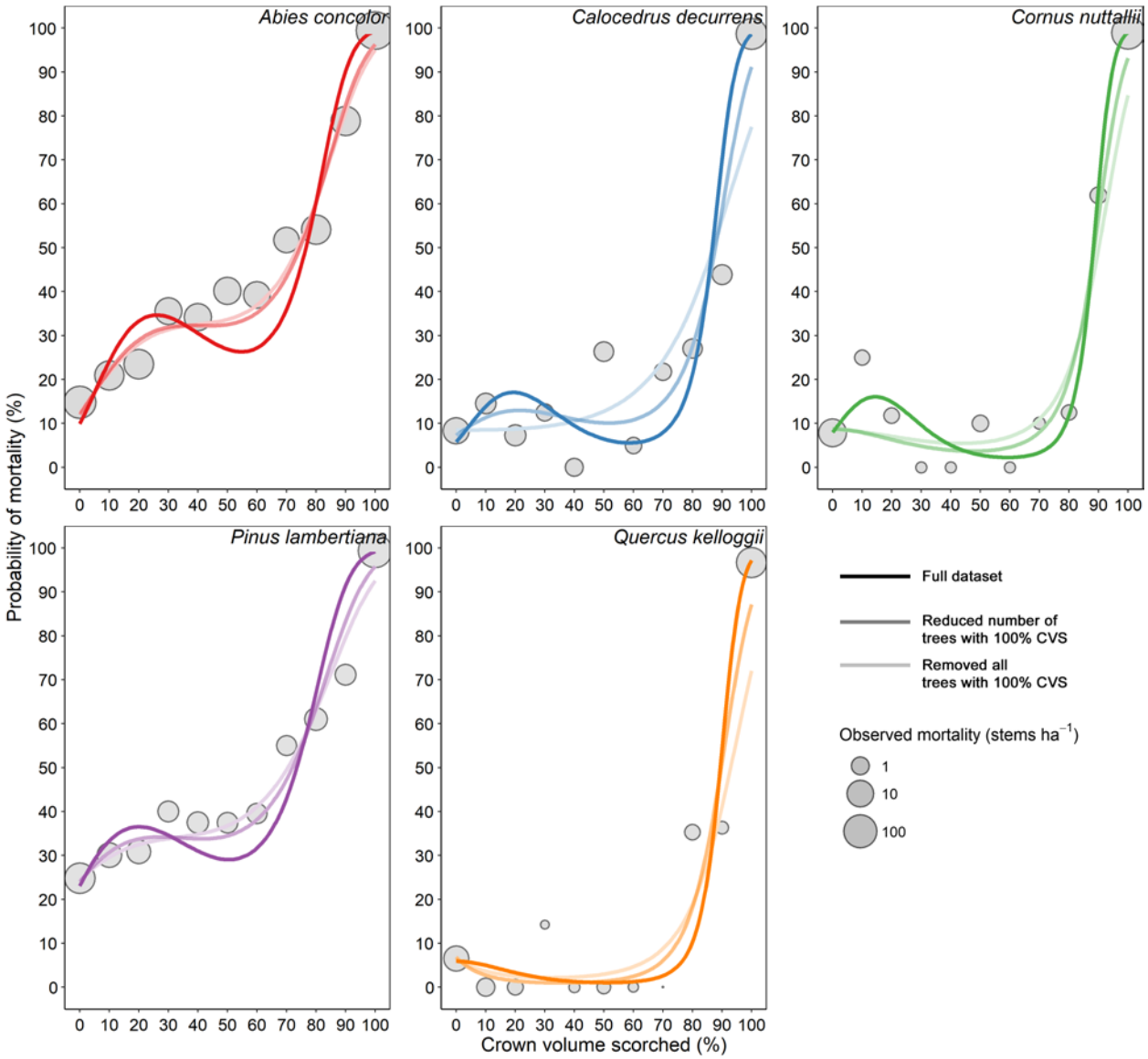

Fig. S2. Probability of mortality for trees ≥1 cm DBH as a function of crown volume scorched (CVS). Dots represent observed proportion of stems that experienced mortality in each CVS category (10% bins; there are 11 dots because there are 0% and 100% bins). Lines represent model predictions of third-order polynomial logistic regression models using subsets of the data (full dataset, removing some trees with 100% CVS, removing all trees with 100% CVS).

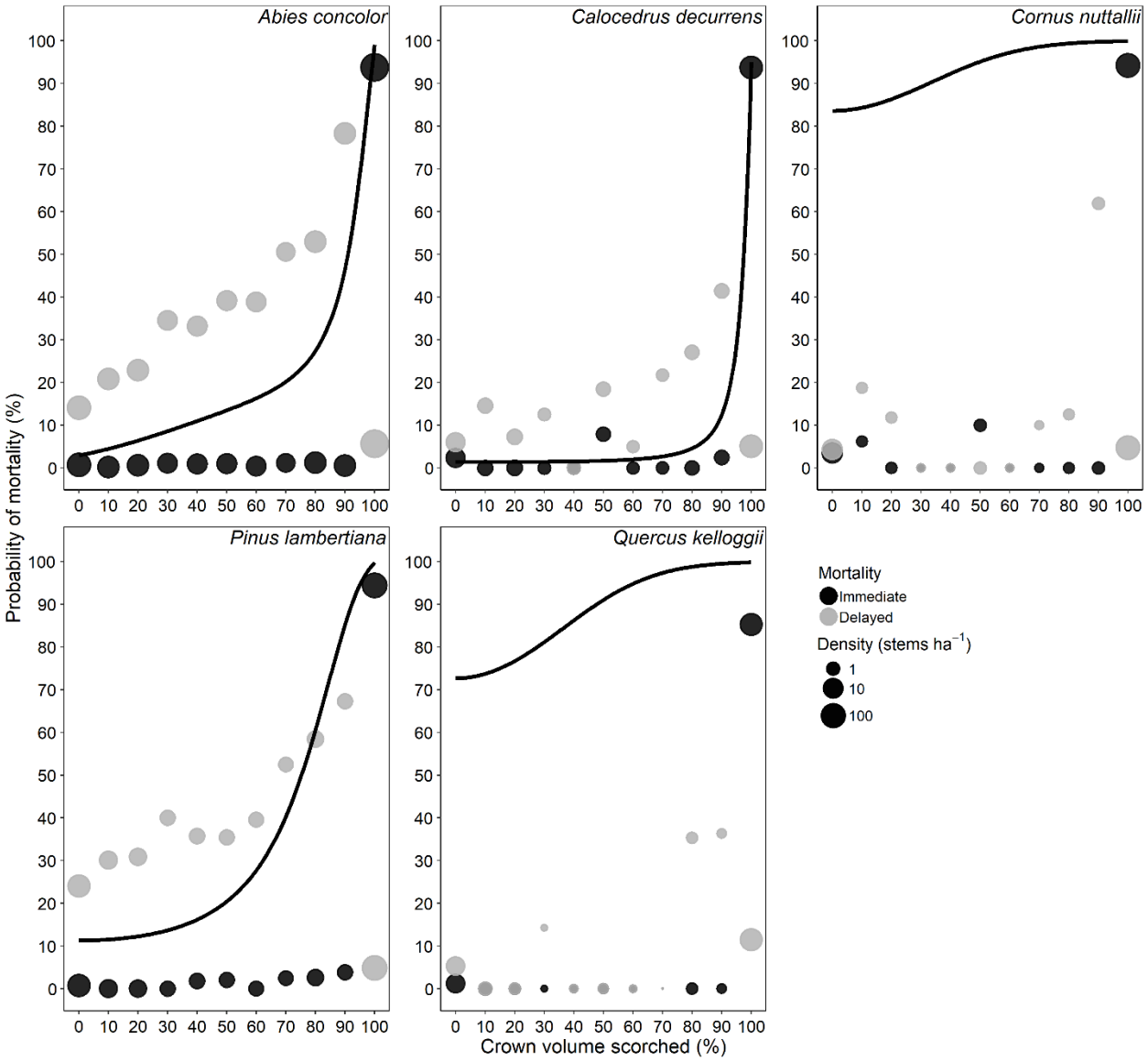

Fig. S3. Probability of mortality for trees ≥1 cm DBH as a function of crown volume scorched (CVS). Dots represent observed proportion of stems that experienced immediate (black dots) and delayed (grey dots) mortality in each CVS category (10% bins; there are 11 dots because there are 0% and 100% bins). Lines represent model predictions of the logistic mortality models within the First Order Fire Effects Model (FOFEM) software (v6.3).

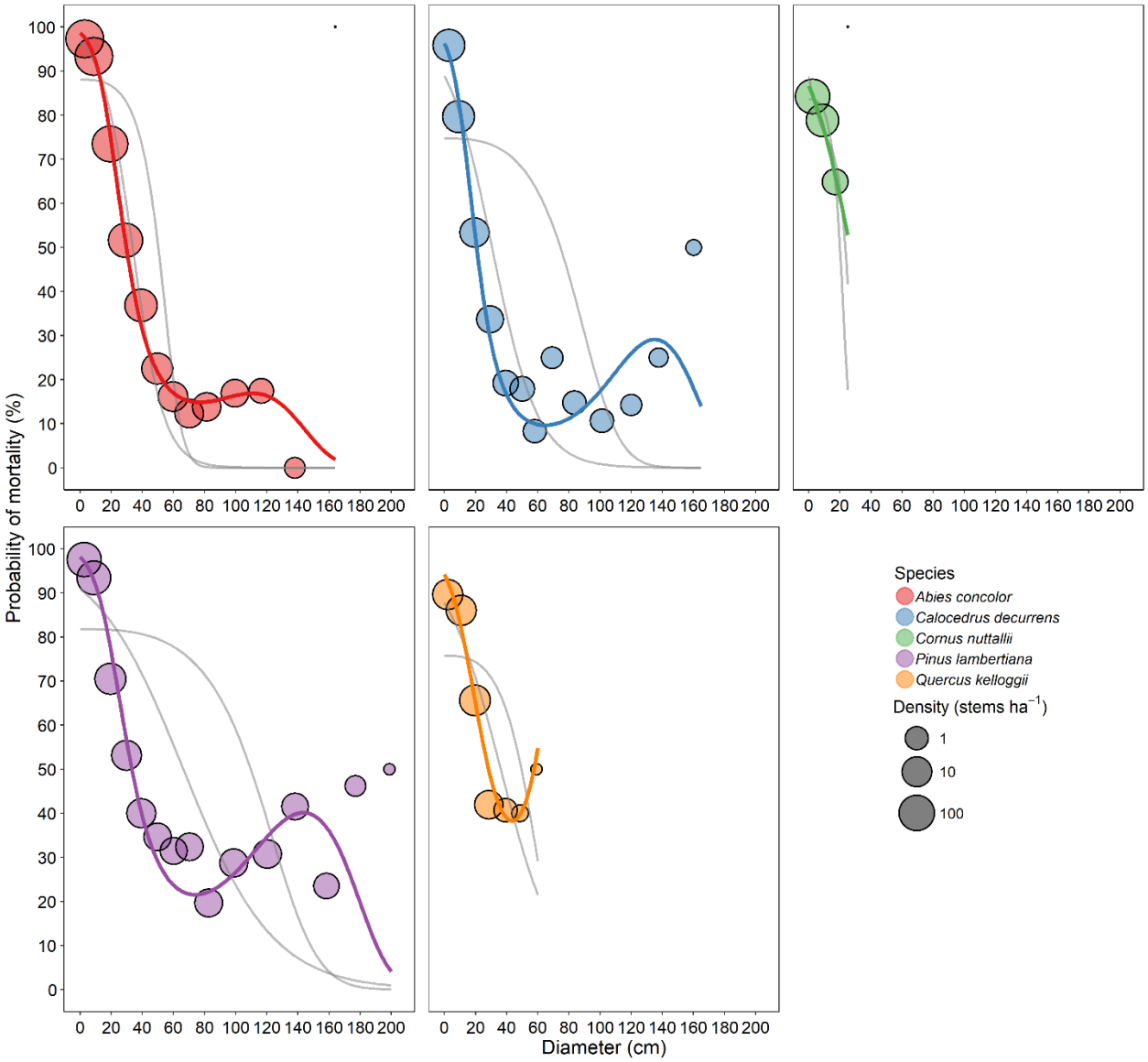

Fig. S4. Probability of mortality for trees ≥1 cm DBH as a function of tree diameter (DBH). Dots represent observed proportion of stems that were killed in each DBH category (10 cm bins, first bin 1 cm ≤ DBH < 10 cm), and lines represent species-specific logistic regression models using DBH as the independent variable and binary mortality status as the response. Grey lines represent model forms that we tested but resulted in worse fits. The dots are for graphical purposes only; the models were not parameterized on the binned data used to generate the dots.

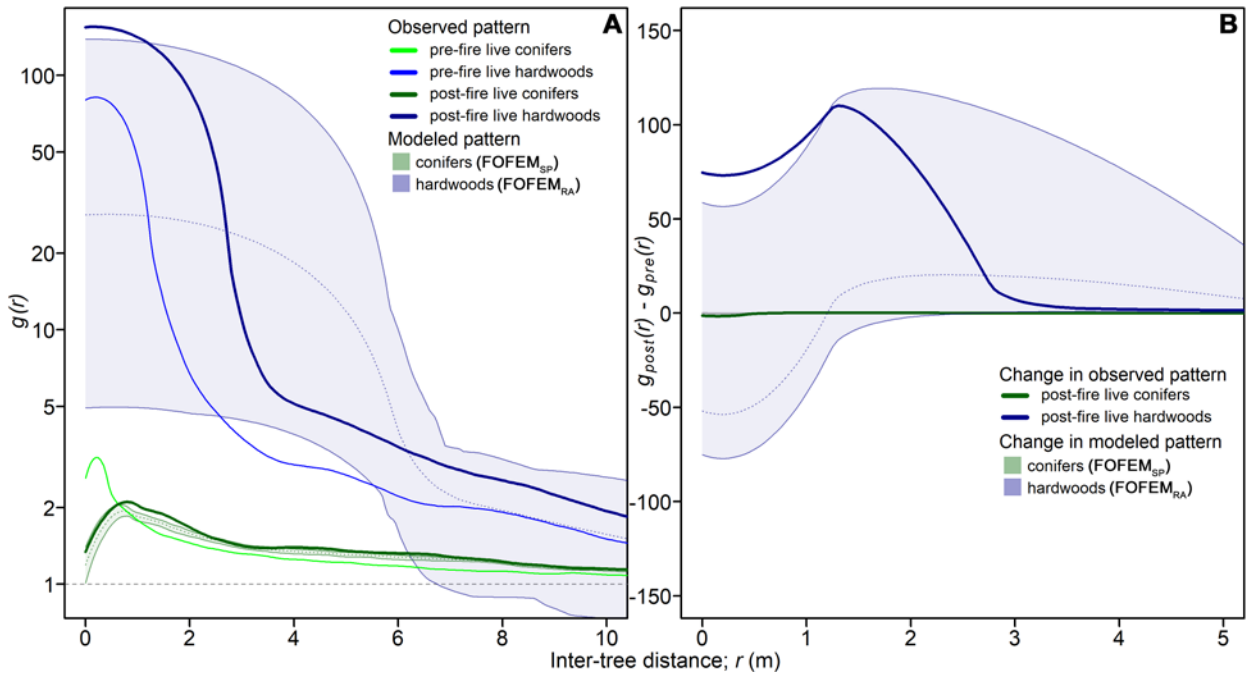

Fig. S5. Pre-fire, post-fire, and modeled post-fire spatial patterns of angiosperms and gymnosperms in the Yosemite Forest Dynamics Plot. The y-axis in A represents the value of the pair-correlation function, $g(r)$, at a range of inter-tree distances, while the y-axis in B represents change in spatial pattern calculated as $g_{post}(r) - g_{pre}(r)$. The shaded areas represent a 95% confidence envelope around predictions of mortality according to the FOFEM models, generated by 99 simulations of mortality. The grey dotted lines represent the expected value of g(r) under the null model of complete spatial randomness (A), or "no change" (B). Values above the line indicate aggregation, and values below indicate hyper-dispersion.

Table S1. Model parameters and coefficients for each of the logistic mortality models validated in this study. Model types were: a diameter at breast height (DBH) model, four locally-parameterized crown volume scorch (CVS) models (a 3rd order polynomial, "CVS", a monotonic polynomial, "CVS_{mono}", a CVS:DBH interaction model, and a model based on trees ≥10 cm), and the models used in the First Order Fire Effects Models software version 5.7+ ($FOFEM_{RA}$ and $FOFEM_{SP}$; Lutes et al. 2016, Hood and Lutes 2017). The FOFEM models use crown length scorched (CLS) rather than the closely-related CVS. The $FOFEM_{RA}$ model uses an additional bark thickness (BT) parameter; species-specific BT parameters may be found in Reinhardt and Crookston 2003.

	Model type	β_0	β_1	X_1	β_2	X_2	β_3	X_3	β_4	X_4	β_5	X_5	B_6	X_6	β_7	X_7
Abies concolor	DBH	4.2814265	-0.2036114	DBH	0.0022409	DBH^2	-7.96E-06	DBH^3	–	–	–	–	–	–	–	–
	CVS	-2.0053061	0.1083929	CVS	-0.0027835	CVS^2	2.26E-05	CVS^3	–	–	–	–	–	–	–	–
	CVS_{mono}	-1.7524821	0.0381398	CVS	-0.0002329	CVS^2	6.37E-28	CVS^{14}	–	–	–	–	–	–	–	–
	CVS+DBH	-1.6725455	0.1674234	CVS	-0.0040512	CVS^2	3.02E-05	CVS^3	-0.0081165	DBH	-0.0028328	CVS:DBH	6.26E-05	CVS^2:DBH	-3.94E-07	CVS^3:DBH
	CVS (≥10 cm)	-1.9678	0.0937255	CVS	-0.0023215	CVS^2	1.89E-05	CVS^3	–	–	–	–	–	–	–	–
	$FOFEM_{SP}$	-3.5083	0.0956	CLS	0.00184	CLS^2	0.000017	CLS^3	–	–	–	–	–	–	–	–
Calocedrus decurrens	DBH	3.2623909	-0.2020971	DBH	0.002309	DBH^2	-7.70E-06	DBH^3	–	–	–	–	–	–	–	–
	CVS	-2.6828101	0.0899026	CVS	-0.0026685	CVS^2	2.29E-05	CVS^3	–	–	–	–	–	–	–	–
	CVS_{mono}	-2.4242955	0.0150742	CVS	4.60E-28	CVS^{14}	–	–	–	–	–	–	–	–	–	–
	CVS+DBH	-2.226668	0.1238947	CVS	-0.0034661	CVS^2	2.73E-05	CVS^3	-0.0109805	DBH	-0.0018474	CVS:DBH	4.03E-05	CVS^2:DBH	-2.21E-07	CVS^3:DBH
	CVS (≥10 cm)	-3.0712897	0.068169	CVS	-0.0019057	CVS^2	1.73E-05	CVS^3	–	–	–	–	–	–	–	–
	$FOFEM_{SP}$	-4.2466	7.172E-06	CLS^3	–	–	–	–	–	–	–	–	–	–	–	–
Cornus nuttallii	DBH	1.8723618	-0.0703851	DBH	–	–	–	–	–	–	–	–	–	–	–	–
	CVS	-2.0368731	-0.0021836	CVS^2	2.67E-05	CVS^3	–	–	–	–	–	–	–	–	–	–
	CVS_{mono}	-2.4636675	0.0010754	CVS	6.37E-34	CVS^{17}	–	–	–	–	–	–	–	–	–	–
	CVS+DBH	2.2974399	-0.2371236	CVS	0.0041271	CVS^2	-1.07E-05	CVS^3	-1.8974305	DBH	0.1462882	CVS:DBH	-0.003904	CVS^2:DBH	2.58E-05	CVS^3:DBH
	CVS (≥10 cm)	-179.49551	15.588101	CVS	-0.453492	CVS^2	0.0031753	CVS^3	–	–	–	–	–	–	–	–
	$FOFEM_{RA}$	1.941	6.316	$1\text{-}e^{BT}$	-0.000535	CS^2	–	–	–	–	–	–	–	–	–	–
Pinus lambertiana	DBH	3.9485856	-0.1708985	DBH	0.0017489	DBH^2	-5.36E-06	DBH^3	–	–	–	–	–	–	–	–
	CVS	-0.9999774	0.043427	CVS	-0.0013633	CVS^2	1.34E-05	CVS^3	–	–	–	–	–	–	–	–
	CVS_{mono}	-0.9592796	0.0135337	CVS	4.44E-36	CVS^{18}	–	–	–	–	–	–	–	–	–	–
	CVS+DBH	-2.0871179	0.1459437	CVS	-0.0036413	CVS^2	2.81E-05	CVS^3	0.0136830	DBH	-0.0017048	CVS:DBH	4.46E-05	CVS^2:DBH	-3.29E-07	CVS^3:DBH
	CVS (≥10 cm)	-0.9083862	0.0259862	CVS	-0.0007339	CVS^2	8.13E-06	CVS^3	–	–	–	–	–	–	–	–
	$FOFEM_{SP}$	-2.0588	0.000814	CLS^2	–	–	–	–	–	–	–	–	–	–	–	–
Quercus kelloggii	DBH	2.7884431	-0.1268596	DBH	0.0006544	DBH^2	1.23E-05	DBH^3	–	–	–	–	–	–	–	–
	CVS	-2.2090094	-0.0581934	CVS	1.04E-05	CVS^2	–	–	–	–	–	–	–	–	–	–
	CVS_{mono}	-2.6987383	0.0007039	CVS	6.00E-20	CVS^{10}	–	–	–	–	–	–	–	–	–	–
	CVS+DBH	-4.896564	0.0995758	CVS	-0.0024104	CVS^2	2.30E-05	CVS^3	0.1172909	DBH	-0.0075775	CVS:DBH	0.0001281	CVS^2:DBH	-7.36E-07	CVS^3:DBH
	CVS (≥10 cm)	-2.0415432	-0.0766217	CVS	0.0006201	CVS^2	5.29E-06	CVS^3	–	–	–	–	–	–	–	–
	$FOFEM_{RA}$	1.941	6.316	$1\text{-}e^{BT}$	-0.000535	CS^2	–	–	–	–	–	–	–	–	–	–

Monotonic polynomial model fitting procedure (R code)

```
##################################################################################################################-
##################################################################################################################-
### R code for fitting monotonic polynomial logistic regression ###
### Author: Tucker J Furniss                                     ###
### Date: January 31, 2018                                       ###
### R version: 3.4.1                                             ###
##################################################################################################################-
#-----------------------------------------------------------------------------------------------------------------#
### NOTE: Each time you run this script, the dummy data is slightly different. To see how this procedure works for slightly
###       different data distributions, source the script multiple times.
### First, load dummy data:
dat<-data.frame('x'=c(rep(0,20),1:100,rep(100,20)),
                'y'=c(sample(c(0,1),20,prob=c(1,0),replace=T),
                      sample(c(0,1),20,prob=c(0.9,0.1),replace=T),
                      sample(c(0,1),60,prob=c(0.8,0.2),replace=T),
                      sample(c(0,1),20,prob=c(0.1,0.9),replace=T),
                      sample(c(0,1),20,prob=c(0,1),replace=T)))

# View data:
plot(dat,pch=19)

# Generate logistic model:
glm<-glm(data=dat,factor(y)~poly(x,3,raw=T),family='binomial')

# View model:
lines(x=dat$x,y=predict(glm,newdata=dat,type='response'),col='red',lwd=2.5)

### NOTE: If this curve generated by the standard glm() function is already monotonic, re-load the dummy data.
###       Sometimes the random sampling procedure used to generate the data results in a distribution of response values that
###       permit the glm() function to generate a monotonic third-order polynomial.
#-----------------------------------------------------------------------------------------------------------------#
#-----------------------------------------------------------------------------------------------------------------#
### Next, load function to determine the minimum power at whihc a curve becomes monotonic:
fit.mono.glm<-function(dat,x,y,power=1,mono.range=c(0,100),mono.res=0,max.pow=100){
  x<-dat[,x]
  y<-dat[,y]
  mono=F
  pow=power+1
  options(warn=-1)
  while(mono==F){
    if(power==1) glm<-glm(data=dat,y~I(x^1)+I(x^pow),family='binomial')
    if(power==2) glm<-glm(data=dat,y~I(x^1)+I(x^2)+I(x^pow),family='binomial')

    pred<-predict(glm,newdata=data.frame('x'=mono.range[1]:mono.range[2]),type='response')
    pred.neg<-pred[2:length(pred)]-pred[1:(length(pred)-1)]
    pred.tf<-abs(sum(pred.neg[pred.neg<0]))>mono.res
    if(length(pred.tf[pred.tf==T])>0) pow=pow+1 else mono=T
    if(pow>max.pow) stop('Error: reached max power without generating monotonic curve.
                         Try evening out independent variables input data, or increasing mono.res.')
  }
  options(warn=0)
  glm$power<-pow
  # plot(x=x,y=predict(glm,newdata=data.frame('x'=x),type='response'),ylim=c(0,1),pch=19,col='red',lwd=2.5,ylab='response')
  cat(paste('Curve became monotonic at x^',pow,'\n',sep=''))
  return(glm)
}
#-----------------------------------------------------------------------------------------------------------------#
#-----------------------------------------------------------------------------------------------------------------#
### Now run the fit.mono.glm() function. Output is a monotonic polynomial model:
mono.glm<-fit.mono.glm(dat,'x','y',power=1,mono.range=c(0,100),mono.res=0.01,max.pow=150)
# You can adjust the parameters for this function:
#   mono.range = the range of x values over which the curve must be monotonic
#   mono.res = the amount of decrease that is allowed while still considering the curve "monotonic"
#             (i.e., if you want a true monotonic curve, set mono.res=0. If you will accept a plateau, set
#             mono.res=0.001, if you will allow a slight dip, set mono.res=0.01 or 0.1)
#   max.pow = maximum power to which your independent variable will be raised.
#             If you get the error: "NA/NaN/Inf in 'x'", reduce this number.

# View the power of the high-order polynomial used to get the monotonic fit:
mono.glm$power

# Plot new model:
plot(dat,pch=19,ylab='response')
lines(x=dat$x,y=predict(glm,newdata=dat,type='response'),col='red',lwd=2.5)
lines(x=dat$x,y=predict(mono.glm,newdata=data.frame('x'=dat),type='response'),col='blue',lwd=2.5)
#-----------------------------------------------------------------------------------------------------------------#
### Now, we can optimize the fit by raising the power of the high-order polynomial until minimum AIC is reached:
pow<-mono.glm$power
new.AIC<-Inf
old.AIC<-Inf
while(new.AIC<=old.AIC&pow<=150){
  old.AIC<-new.AIC
  better.mono.glm<-glm(data=dat,y~I(x^1)+I(x^pow),family='binomial')
  new.AIC<-AIC(better.mono.glm)
  lines(x=dat$x,y=predict(better.mono.glm,newdata=dat,type='response'),ylim=c(0,1),col='grey30')
  pow<-pow+1
}
# The 'while' loop stops as soon as AIC starts to increase. Create final model one step back before AIC began to increase:
better.mono.glm<-glm(data=dat,y~I(x^1)+I(x^pow-1),family='binomial')

#View final model:
lines(x=dat$x,y=predict(better.mono.glm,newdata=dat,type='response'),col='green',lwd=2.5,ylab='response')
#-----------------------------------------------------------------------------------------------------------------#
```

code continues on next page

```
#-----------------------------------------------------------------------------------------------------------------#
### Some data may be better fit by adding in an x^2 term. Change power argument to power=2:
mono.glm2<-fit.mono.glm(dat,'x','y',power=2,mono.range=c(0,100),mono.res=0.01,max.pow=150)
# Plot model:
plot(dat,pch=19,ylab='response')
lines(x=dat$x,y=predict(glm,newdata=dat,type='response'),col='red',lwd=2.5)
lines(x=dat$x,y=predict(mono.glm2,newdata=data.frame('x'=dat),type='response'),col='blue',lwd=2.5)
#-----------------------------------------------------------------------------------------------------------------#
# The optimization loop will also run for this model if we modify the glm() call within the loop:
pow<-mono.glm2$power
new.AIC<-Inf
old.AIC<-Inf
while(new.AIC<=old.AIC&pow<=150){
  old.AIC<-new.AIC
  better.mono.glm<-glm(data=dat,y~I(x^1)+I(x^2)+I(x^pow),family='binomial')
  new.AIC<-AIC(better.mono.glm)
  lines(x=dat$x,y=predict(better.mono.glm,newdata=dat,type='response'),ylim=c(0,1),col='grey30')
  pow<-pow+1
}
better.mono.glm2<-glm(data=dat,y~I(x^1)+I(x^2)+I(x^pow-1),family='binomial')

#View final model:
lines(x=dat$x,y=predict(better.mono.glm2,newdata=dat,type='response'),col='green',lwd=2.5,ylab='response')
#-----------------------------------------------------------------------------------------------------------------#
##################################################################################################################-
##################################################################################################################-
```