Supplementary material to Wilson, P.D. (2001) Distance-based methods for the analysis of maps produced by species distribution models. *Methods in Ecology and Evolution*

R-scripts

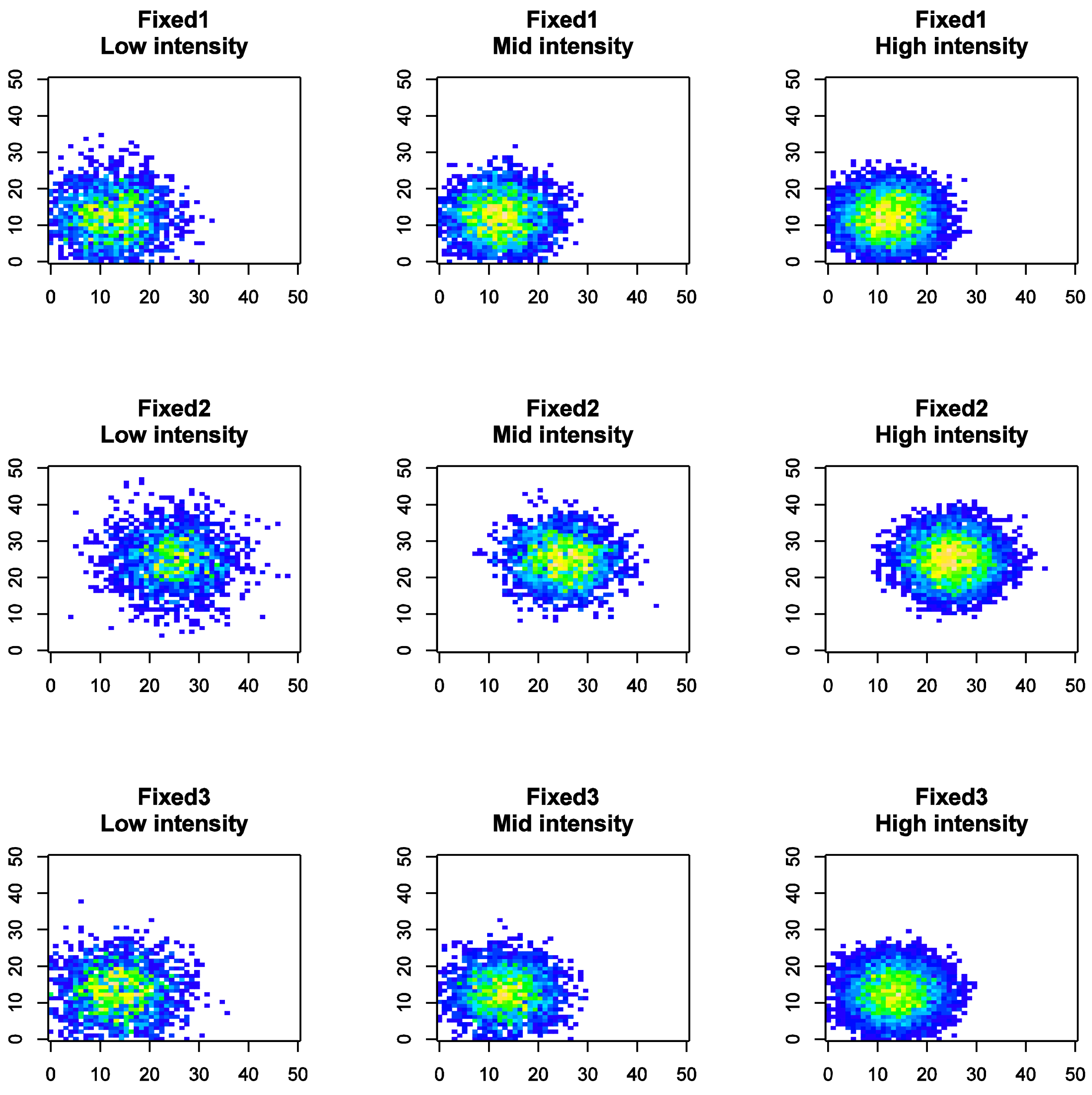
Synthetic test data

Fixed patterns

```
# Extensive testing of distance-based ordination method: Interpretability
# and sensitivity test 1
#
#  a. Generate a bivariate spatial pattern with a given intensity in a large
#     window.
#  b. Generate a similar pattern (i.e. centred on the same coordinates within
#     the same window) but with a higher intensity.
#
# Peter D. Wilson
# Department of Biological Sciences
# Maquarie University, New South Wales, Australia 2109
# email: peterdonaldwilson@gmail.com
#
# Adpated 25 Feb 2010 from Inertial Ordination test script of 1 August 2008.
# Added computation of Euclidean distances and Entropy divergences.
# Amended 13 May 2010 to create separate versions making replicated fixed and
# moving spatial patterns

library(spatstat)
library(labdsv)
#library(grDevices)

bivar <- function (x,y,mx,sx,my,sy,nmax)
 {
  return(nmax*(1/(2*pi*sx*sy))*exp(-((((x-mx)/sx)^2+((y-my)/sy)^2))/2)) # 0.25+
 }


HellingerDist <- function (mat1,mat2)
 {
  p1 <- sum(mat1,na.rm=T)
  p2 <- sum(mat2,na.rm=T)
  return(sqrt(0.5*sum((sqrt(mat1/p1) - sqrt(mat2/p2))^2,na.rm=T)))
 }


EuclideanDist <- function(mat1,mat2)
 {
  return(sqrt(sum((mat1 - mat2)^2,na.rm=T)))
 }


NormEuclideanDist <- function(mat1,mat2)
 {
  p1 <- mat1/sum(mat1,na.rm = T)
  p2 <- mat2/sum(mat2,na.rm = T)
  return(sqrt(sum((p1 - p2)^2,na.rm=T)))
 }


KullbackLeiblerDiv <- function(mat1,mat2)
```

```
 {
  s1 <- sum(mat1,na.rm=T)
  s2 <- sum(mat2.na.rm=T)
  p1 <- mat1/s1
  p2 <- mat2/s2
  ###d12 <- sum(p1*log(p1/p2),na.rm=T)
  ###d21 <- sum(p2*log(p2/p1),na.rm=T)
  return(sum(log(p1/p2)*(p1-p2),na.rm=T))   ###(d12 + d21)
 }


TsallisDiv <- function(mat1,mat2,alpha)
 {
  p1 <- mat1/sum(mat1,na.rm=T)
  p2 <- mat1/sum(mat2,na.rm=T)
  d12 <- 1 - sum((p1^alpha)/(p2^(alpha - 1)),na.rm=T)
  d21 <- 1 - sum((p2^alpha)/(p1^(alpha - 1)),na.rm=T)
  return((1/(1 - alpha))*(d12+d21))
 }


pcoPlotFunc <- function(xpts,ypts,numTests,numSteps,aTitle,PlotChars,PlotCol,
                        PlotTag)
 {
  plot(xpts,ypts,xlab="Axis 1",ylab="Axis 2",pch=PlotChars,col=PlotCol,
       main=aTitle,cex=1.9)
  mtext(PlotTag,line=1,font=2,adj = 0)
 }

numPatterns <- 3
numSteps <- 3
numReps <- 5

# Key parameters for bivariate normal intensity function "bivar"
mx <- c(0.5,1,0.55)
sx <- 0.25
my <- c(0.5,1,0.52)
sy <- 0.25

plotChr <- c(rep(16,numSteps*numReps),rep(17,numSteps*numReps),rep(3,numSteps*numReps))

c1 <- rgb(102,102,51,maxColorValue=255)
c2 <- rgb(102,102,255,maxColorValue=255)
c3 <- rgb(204,204,102,maxColorValue=255)

plotCol <- rep(c(rep(c1,numReps),rep(c2,numReps),rep(c3,numReps)),numPatterns)

# Other parameters of the test space
nx <- 50
ny <- 50
#NumCells <- nx*ny

BigWin <- owin(c(0,2),c(0,2))

d <- vector("list", numPatterns*numSteps*numReps) # numPatterns

ObjectNames <- c("Fixed1Lo","Fixed1Mid","Fixed1Hi","Fixed2Lo","Fixed2Mid",
                 "Fixed2Hi","Fixed3Lo","Fixed3Mid","Fixed3Hi") #,"Move1Lo",

maxInt <- c(2000,2500,3000)

# Note: a small value (0.05) is added to each grid to avoid a div by zero error
# when computing the Kullback-Leibler divergence

dptr <- 1

for (i in 1:numPatterns)
 {
  for (j in 1:numSteps)
   {
    for (k in 1:numReps)
     {
      PP <- rpoispp(bivar,win=BigWin,mx=mx[i],sx=sx,my=my[i],sy=sy,nmax=maxInt[j])

      d[[dptr]] <- quadratcount(PP,nx,ny) + 0.05

      dptr <- dptr + 1
     }
   }
 }

# Make distance matrices
HD <- matrix(0,numPatterns*numSteps*numReps,numPatterns*numSteps*numReps)
ED <- matrix(0,numPatterns*numSteps*numReps,numPatterns*numSteps*numReps)
KLD <- matrix(0,numPatterns*numSteps*numReps,numPatterns*numSteps*numReps)
TD <- matrix(0,numPatterns*numSteps*numReps,numPatterns*numSteps*numReps)
NED <- matrix(0,numPatterns*numSteps*numReps,numPatterns*numSteps*numReps)

for (i in 2:(numPatterns*numSteps*numReps))
 {
  for (j in 1:(i-1))
   {
    HD[i,j] <- HellingerDist(as.matrix(d[[i]]),as.matrix(d[[j]]))
    HD[j,i] <- HD[i,j]
    ED[i,j] <- EuclideanDist(as.matrix(d[[i]]),as.matrix(d[[j]]))
    ED[j,i] <- ED[i,j]
    KLD[i,j] <- KullbackLeiblerDiv(as.matrix(d[[i]]),as.matrix(d[[j]]))
    KLD[j,i] <- KLD[i,j]
    TD[i,j] <- TsallisDiv(as.matrix(d[[i]]),as.matrix(d[[j]]),0.25)
    TD[j,i] <- TD[i,j]
    NED[i,j] <- NormEuclideanDist(as.matrix(d[[i]]),as.matrix(d[[j]]))
    NED[j,i] <- NED[i,j]
   }
 }

HDpco <- pco(as.dist(HD))
EDpco <- pco(as.dist(ED))
KLDpco <- pco(euclidify(as.dist(KLD)))
TDpco <- pco(euclidify(as.dist(TD)))
NEDpco <- pco(as.dist(NED))

#postscript("F:/Distance-based map ordination/PCoordPlotsFIXED.eps",
#            width = 6.5,height=10,paper="a4",horizontal = F)

par(mfrow=c(3,2))
#PCoA of Hellinger Distance
pcoPlotFunc(HDpco$points[,1],HDpco$points[,2],numPatterns*numReps,numSteps,"",
            plotChr,plotCol,"a. Hellinger")
#PCoA of Euclidean Distance
pcoPlotFunc(EDpco$points[,1],EDpco$points[,2],numPatterns*numReps,numSteps,"",
            plotChr,plotCol,"b. Euclidean")
#PCoA of Normed Euclidean Distance
pcoPlotFunc(NEDpco$points[,1],NEDpco$points[,2],numPatterns*numReps,numSteps,"",
            plotChr,plotCol,"c. Normalised Euclidean")
#PCoA of Kullback-Leibler Divergence
pcoPlotFunc(KLDpco$points[,1],KLDpco$points[,2],numPatterns*numReps,numSteps,"",
            plotChr,plotCol,"d. Kullback-Leibler")
#PCoA of Tsallis Divergence (alpha = 0.25)
pcoPlotFunc(TDpco$points[,1],TDpco$points[,2],numPatterns*numReps,numSteps,"",
            plotChr,plotCol,"e. Tsallis")

#dev.off()
```

Supplementary Figure 1: Example plot of the fixed-position test data set. Colour gradient blue to orange represents low to high density values in each grid cell of a map.

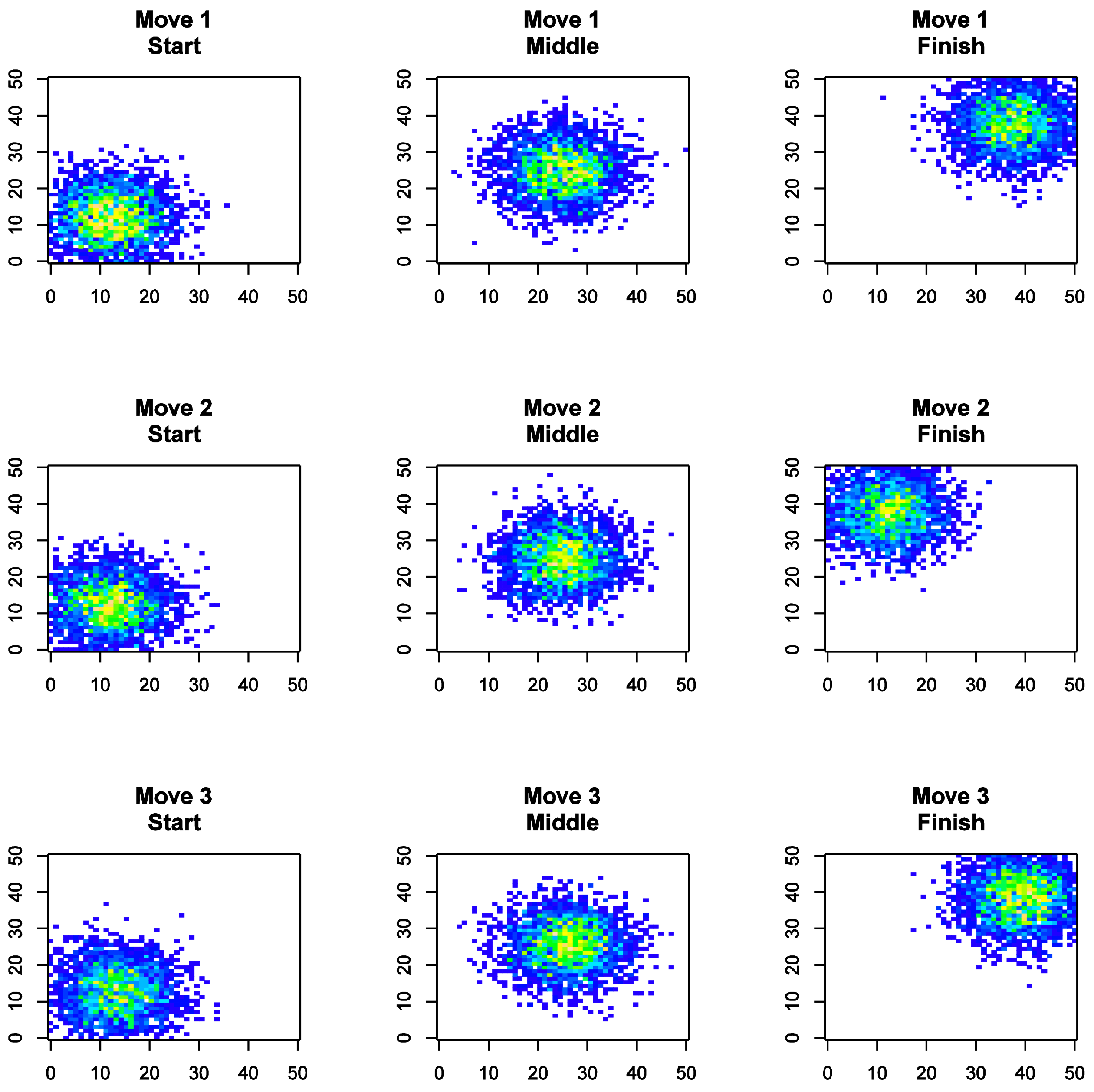


Moving patterns (sequences)

```
# Extensive testing of distance-based ordination method: Interpretability
# and sensitivity test 1
#
#  a. Generate a bivariate spatial pattern with a given intensity in a large
#     window.
#  b. Generate a sequence of similar patterns with increasing intensity.
#
# Peter D. Wilson
# Department of Biological Sciences
# Macquarie University, New South Wales, Australia 2109
# email: peterdonaldwilson@gmail.com
#
# Adapted 25 Feb 2010 from Inertial Ordination test script of 1 August 2008.
# Added computation of Euclidean and Entropy divergences. PDW 30 April 2010
# Amended 13 May 2010 to create separate versions making replicated fixed and
# moving spatial patterns: This version implements a set of changing intensity
# moving patterns
# Patched 7 June 2010 to add annotations to the plots

library(spatstat)
library(labdsv)
#library(grDevices)

bivar <- function (x,y,mx,sx,my,sy,nmax)
 {
  return(nmax*(1/(2*pi*sx*sy))*exp(-((((x-mx)/sx)^2+((y-my)/sy)^2))/2)) # 0.25+
 }


HellingerDist <- function (mat1,mat2)
 {
  p1 <- sum(mat1,na.rm=T)
  p2 <- sum(mat2,na.rm=T)
  return(sqrt(0.5*sum((sqrt(mat1/p1) - sqrt(mat2/p2))^2,na.rm=T)))
 }


EuclideanDist <- function(mat1,mat2)
 {
  return(sqrt(sum((mat1 - mat2)^2,na.rm=T)))
 }


NormEuclideanDist <- function(mat1,mat2)
 {
  p1 <- mat1/sum(mat1,na.rm = T)
  p2 <- mat2/sum(mat2,na.rm = T)
  return(sqrt(sum((p1 - p2)^2,na.rm=T)))
 }


KullbackLeiblerDiv <- function(mat1,mat2)
 {
  s1 <- sum(mat1,na.rm=T)
  s2 <- sum(mat2.na.rm=T)
  p1 <- mat1/s1
  p2 <- mat2/s2
  ###d12 <- sum(p1*log(p1/p2),na.rm=T)
  ###d21 <- sum(p2*log(p2/p1),na.rm=T)
  return(sum(log(p1/p2)*(p1-p2),na.rm=T))   ###(d12 + d21)
 }


TsallisDiv <- function(mat1,mat2,alpha)
 {
  p1 <- mat1/sum(mat1,na.rm=T)
  p2 <- mat1/sum(mat2,na.rm=T)
  d12 <- 1 - sum((p1^alpha)/(p2^(alpha - 1)),na.rm=T)
  d21 <- 1 - sum((p2^alpha)/(p1^(alpha - 1)),na.rm=T)
  return((1/(1 - alpha))*(d12)) ####+d21
 }


pcoPlotFunc <- function(xpts,ypts,numTests,numSteps,numReps,aTitle,PlotChars,PlotCol,
                        PlotTag)
 {
  plot(xpts,ypts,xlab="Axis 1",ylab="Axis 2",pch=PlotChars,col=PlotCol,main=aTitle,cex=1.5)
  mtext(PlotTag,line=1,font=2,adj = 0)
  text(xpts[1],ypts[1],"Start",pos=2,offset=1)
  text(xpts[length(xpts)],ypts[length(xpts)],"End 1,3",pos=4,offset=1)
  n <- 1 + (numSteps - 1) + (numSteps*numReps)
  text(xpts[n],ypts[n],"End 2",pos=4,offset=1)

  j <- 1:numSteps

  for (i in 1:numTests)
   {
    k <- numSteps*(i - 1) + j
    lines(xpts[k],ypts[k],lty=2)
   }
 }

# Number of moving patterns or “sequences”
numSeqs <- 3

# Number of steps along a sequence
numSteps <- 3

# Number of replicates of a sequence to be generated
numReps <- 5

# Key parameters for bivariate normal intensity function "bivar"
mx <- c(rep(c(0.5,1.0,1.5),numReps),rep(c(0.5,1,0.5),numReps),rep(c(0.55,1.05,1.55),numReps))
sx <- rep(0.25,numSeqs*numSteps*numReps)
my <- c(rep(c(0.5,1.0,1.5),numReps),rep(c(0.5,1,1.5),numReps),rep(c(0.52,1.02,1.52),numReps))
sy <- rep(0.25,numSeqs*numSteps*numReps)

plotChr <- c(rep(16,numSteps*numReps),rep(17,numSteps*numReps),rep(3,numSteps*numReps))

c1 <- rgb(102,102,51,maxColorValue=255)
c2 <- rgb(102,102,255,maxColorValue=255)
c3 <- rgb(204,204,102,maxColorValue=255)

plotCol <- rep(c(c1,c2,c3),numSeqs*numReps)

# Other parameters of the test space
nx <- 50
ny <- 50
#NumCells <- nx*ny

BigWin <- owin(c(0,2),c(0,2))

d <- vector("list", numSeqs*numSteps*numReps)

# For a constant intensity moving pattern example, set the following three variables to the
# same value, say, 2500
nmaxLo <- 2000
nmaxMid <- 2500
nmaxHi <- 3000

# Note: a small value (0.05) is added to each grid to avoid a div by zero error
# when computing the Kullback-Leibler divergence
```

```
for (i in 1:(numSteps*numReps))
 {
  j <- numSteps*(i-1) + 1
  LoPP <- rpoispp(bivar,win=BigWin,mx=mx[j],sx=sx[j],my=my[j],sy=sy[j],
                  nmax=nmaxLo)
  d[[j]] <- quadratcount(LoPP,nx,ny) + 0.05
  MidPP <- rpoispp(bivar,win=BigWin,mx=mx[j+1],sx=sx[j+1],my=my[j+1],sy=sy[j+1],
                   nmax=nmaxMid)
  d[[j+1]] <- quadratcount(MidPP,nx,ny) + 0.05
  HiPP <- rpoispp(bivar,win=BigWin,mx=mx[j+2],sx=sx[j+2],my=my[j+2],sy=sy[j+2],
                  nmax=nmaxHi)
  d[[j+2]] <- quadratcount(HiPP,nx,ny) + 0.05
 }

# Make distance matrices
HD <- matrix(0,numSeqs*numSteps*numReps,numSeqs*numSteps*numReps)
ED <- matrix(0,numSeqs*numSteps*numReps,numSeqs*numSteps*numReps)
KLD <- matrix(0,numSeqs*numSteps*numReps,numSeqs*numSteps*numReps)
TD <- matrix(0,numSeqs*numSteps*numReps,numSeqs*numSteps*numReps)
NED <- matrix(0,numSeqs*numSteps*numReps,numSeqs*numSteps*numReps)

for (i in 2:(numSeqs*numSteps*numReps))
 {
  for (j in 1:(i-1))
   {
    HD[i,j] <- HellingerDist(as.matrix(d[[i]]),as.matrix(d[[j]]))
    HD[j,i] <- HD[i,j]
    ED[i,j] <- EuclideanDist(as.matrix(d[[i]]),as.matrix(d[[j]]))
    ED[j,i] <- ED[i,j]
    KLD[i,j] <- KullbackLeiblerDiv(as.matrix(d[[i]]),as.matrix(d[[j]]))
    KLD[j,i] <- KLD[i,j]
    TD[i,j] <- TsallisDiv(as.matrix(d[[i]]),as.matrix(d[[j]]),0.25)
    TD[j,i] <- TD[i,j]
    NED[i,j] <- NormEuclideanDist(as.matrix(d[[i]]),as.matrix(d[[j]]))
    NED[j,i] <- NED[i,j]
   }
 }

HDpco <- pco(as.dist(HD))
EDpco <- pco(as.dist(ED))
KLDpco <- pco(as.dist(KLD))
TDpco <- pco(as.dist(TD))
NEDpco <- pco(as.dist(NED))

#postscript("G:/Distance-based map ordination/PCoordPlotsMOVE2.eps",
#            width = 6.5,height=10,paper="a4",horizontal = F)

par(mfrow=c(3,2))
#PCoA of Hellinger Distance
pcoPlotFunc(HDpco$points[,1],HDpco$points[,2],numSeqs*numReps,numSteps,numReps,"",
            plotChr,plotCol,"a. Hellinger")
#PCoA of Euclidean Distance
pcoPlotFunc(EDpco$points[,1],EDpco$points[,2],numSeqs*numReps,numSteps,numReps,"",
            plotChr,plotCol,"b. Euclidean")
#PCoA of Normed Euclidean Distance
pcoPlotFunc(NEDpco$points[,1],NEDpco$points[,2],numSeqs*numReps,numSteps,numReps,"",
            plotChr,plotCol,"c. Normalised Euclidean")
#PCoA of Kullback-Leibler Divergence
pcoPlotFunc(KLDpco$points[,1],KLDpco$points[,2],numSeqs*numReps,numSteps,numReps,"",
            plotChr,plotCol,"d. Kullback-Leibler")
#PCoA of Tsallis Divergence (alpha = 0.25)
pcoPlotFunc(TDpco$points[,1],TDpco$points[,2],numSeqs*numReps,numSteps,numReps,"",
            plotChr,plotCol,"e. Tsallis")

#dev.off()
```

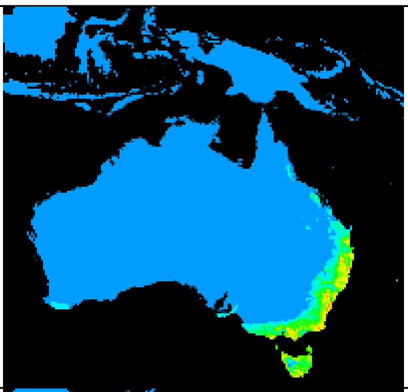
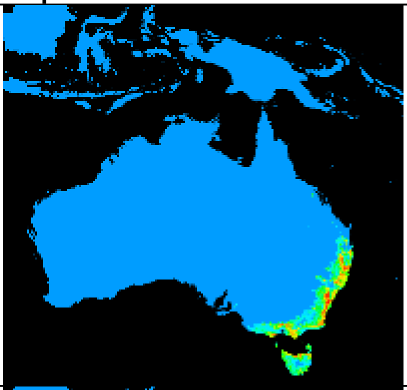
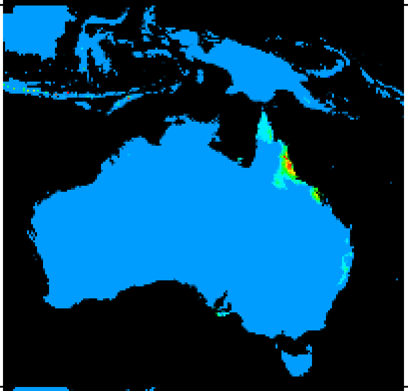
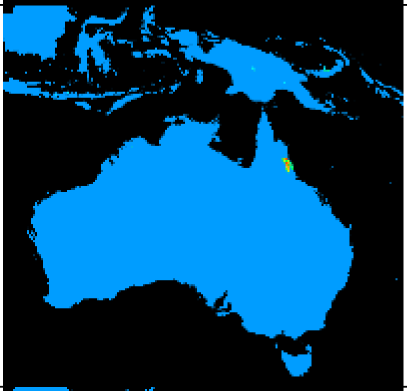
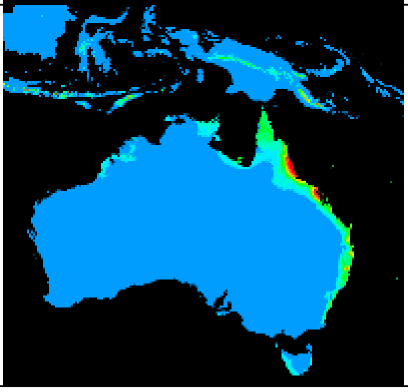
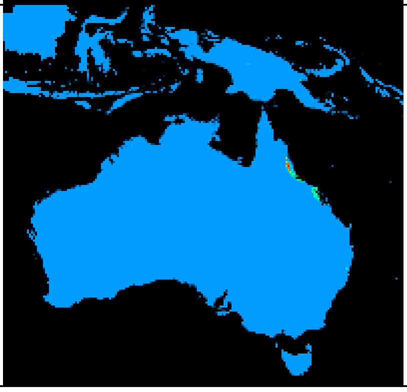
Supplementary Figure 2: Example plot of the moving patch with constant intensity test data set. Colour gradient blue to orange represents low to high density values in each grid cell of a map.

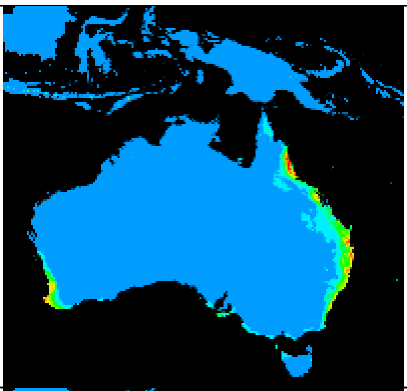
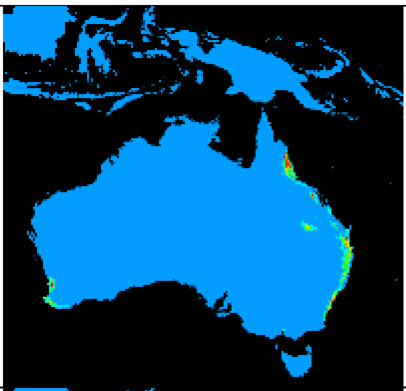
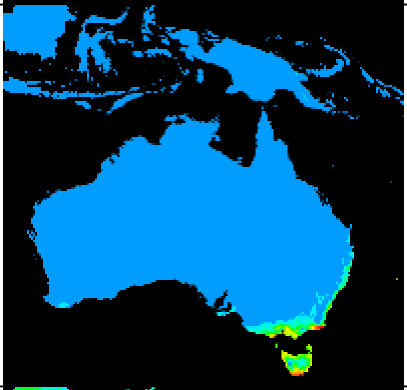
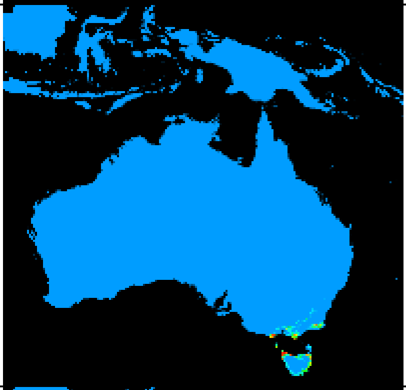
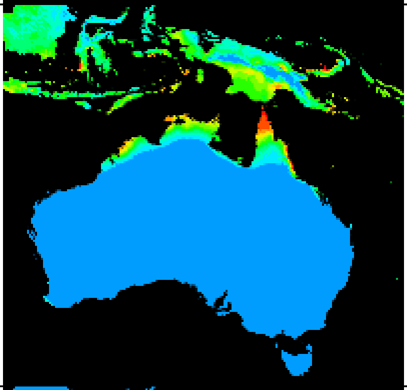
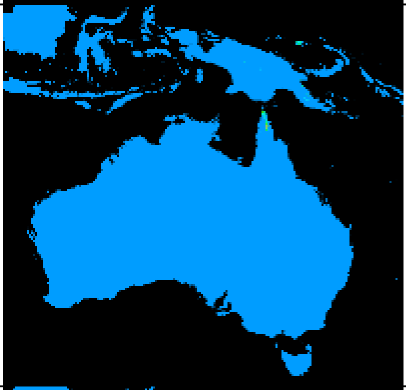
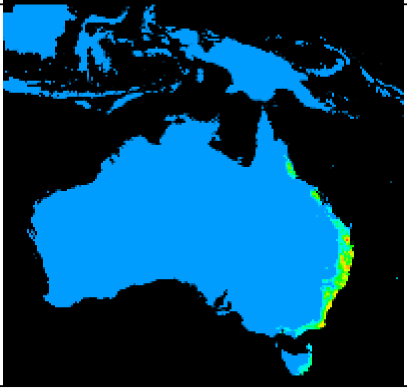
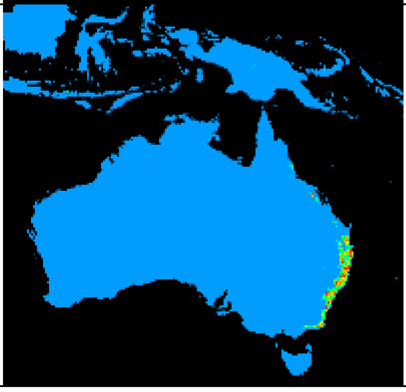


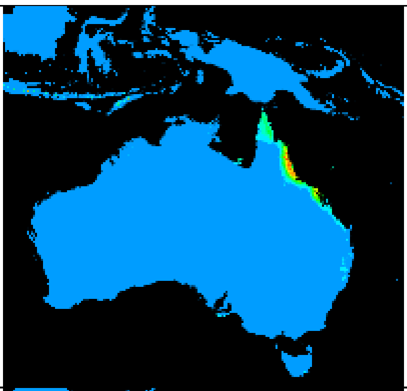
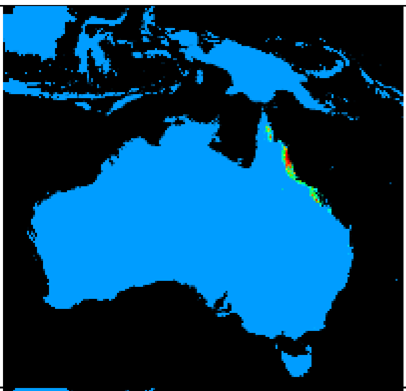
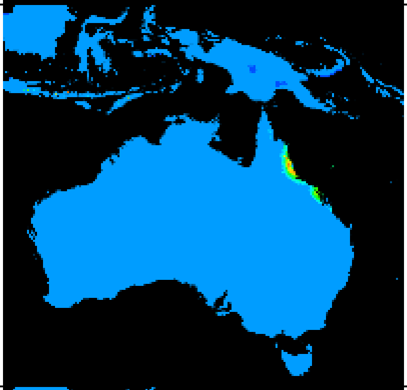
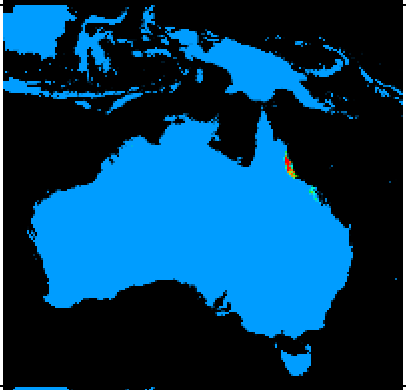
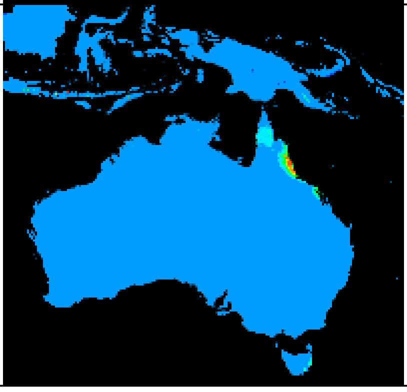
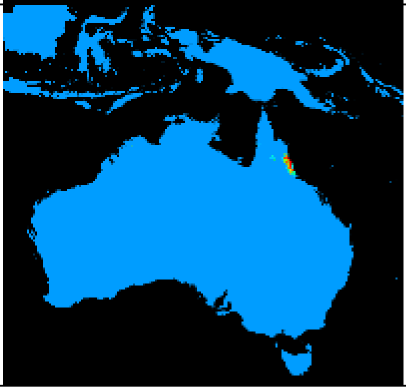
Modelling results

Cyathea *species*

Mean output maps from MaxEnt and BRT models of 10 *Cyathea* species. Data sources, climate data and modelling fitting methods are described in the main text. Colour scheme for maps represents climate suitability near zero as blue, grading through to warm colours of orange and red for climate suitability near one.

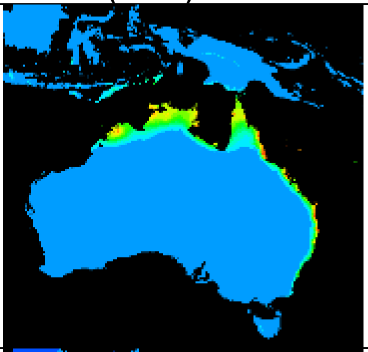
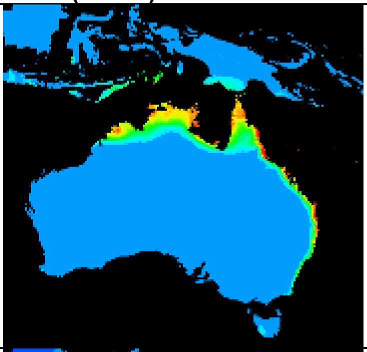
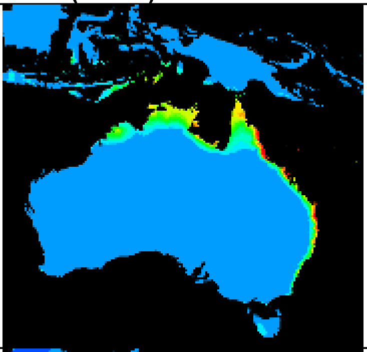
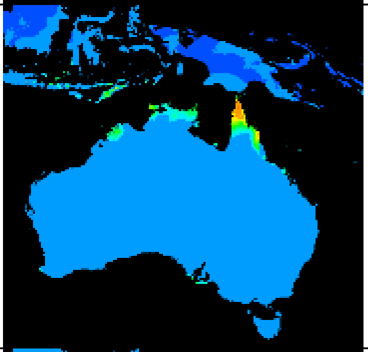
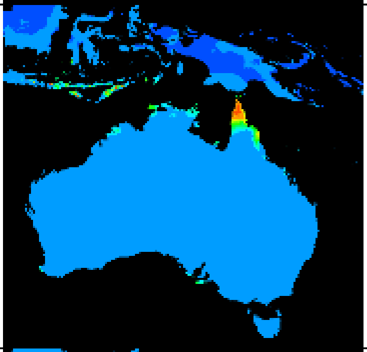
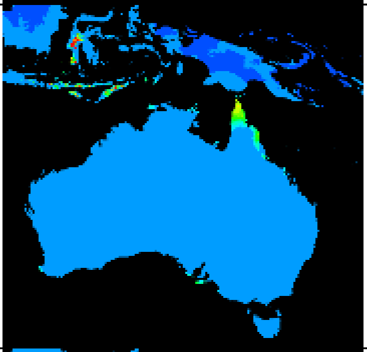
Species	Mean MaxEnt map	Mean Boosted Regression Tree Map
Cyathea australis		
C. baileyana		
C. celebica		

C. cooperi		
C. cunninghamii		
C. felina		
C. leichhardtiana		

C. rebeccae		
C. robertsiana		
C. woollsiana		

Melomys *species*

Mean maps for MaxEnt models for three Australia species of *Melomys*. Data sources, climate data and modelling method are described in the main text. Current (mean) maps are the fitted model for current climate conditions, while 2020 (mean) and 2050 (mean) are the mean maps for the fitted model projected onto decadal average climate centred on the indicated year. Colour scheme is the same described above for *Cyathea* species.

Species	Current (mean)	2020 (mean)	2050 (mean)
Melomys burtoni			
M. capensis			
M. cervinipes