Non-linear thermal gradients shape broad-scale patterns in geographic range size and can reverse Rapoport's Rule: Appendix C

Adam Tomašových, David Jablonski, Sarah K. Berke, Andrew Z. Krug, James W. Valentine

Range-shuffling null model

We upload four input files. First, we upload one global array (`SSTgridmap`) with 360 rows (longitudes) and 180 columns (latitudes) corresponding to 1º cells that contain mean sea-surface temperature values. Geographic ranges are placed on this type of array (with the algorithm accounting for varying longitudinal extent of individual latitudes and for edge effects of the gridded map when randomized ranges pass from the Eastern to the Western Hemisphere). Second, we upload two vectors with species latitudinal (`Slatextent`) and longitudinal (`Slongextent`) ranges (946 species) observed in empirical database. These latitudinal and longitudinal ranges are approximately conserved in the null model (using range-rejection algorithm). Third, we upload two vectors with latitudinal (`shelflats`) and longitudinal (`shelflongs`) coordinates of 1º cells that contain shelf depths (< 200 m). Midpoints of geographic ranges are placed on these coordinates (i.e., not on open ocean or on continents). Fourth, we upload two vectors with 535 latitudinal (`lats`) and longitudinal (`longs`) coordinates of 1º cells actually sampled by the empirical database. Species latitudinal and thermal ranges predicted by the null model are based on these coordinates. Therefore, for example, sampled latitudinal ranges can differ from true (complete) latitudinal ranges. The example below is using the Western Atlantic data.

```
#MAP WITH SEA-SURFACE TEMPERATURE
load("SST gridded data.Rdata")
SSTgridmap=t(SSTgridmap)
SSTgridmap<-SSTgridmap[180:1,]

#COORDINATES OF SHELF CELLS LOCALITIES
load("Shelf coordinates.Rdata")
```

```
shelflats=allWAlats
shelflongs=allWAlongs

#COORDINATES OF SAMPLED LOCALITIES (OCCURRENCES)
load("Coordinates and SST of sampled cells.Rdata")

#EMPIRICAL LATITUDINAL AND LONGITUDINAL RANGES OF SPECIES
load(file="Empirical species ranges.Rdata")
Ntaxa<-length(Slatextent)

#GRID MAP WITH ROWS (LATITUDES) AND COLUMNS (LONGITUDES)
gridlat=seq(-89.5,89.5,by=1)
gridlong=seq(-179.5,179.5,by=1)
gridmap<-matrix(data=0, nrow=length(gridlat), ncol=length(gridlong))
rownames(gridmap)=gridlat
colnames(gridmap)=gridlong

#VECTORIZE ALL 1° CELLS
xcells<-numeric();ycells<-numeric();
for (xx in 1:nrow(gridmap)) {
	for (yy in 1:ncol(gridmap)) {
		xcells<-append(xcells, gridlong[yy], length(xcells))
		ycells<-append(ycells, gridlat[xx], length(ycells))
		}
}

gridmap<-matrix(data=0, nrow=length(gridlat), ncol=length(gridlong));
rownames(gridmap)=gridlat
colnames(gridmap)=gridlong

#VECTORIZE LATITUDES AND LONGITUDES OF SHELF CELLS
shelf=numeric()
for (i in 1:length(shelflats)) {
	shelf[i]<-paste(shelflats[i],shelflongs[i])
	}

#SPECIES RANGE LIMITS
Slimit<-rep(NA,Ntaxa);Nlimit<-rep(NA,Ntaxa);
Elimit<-rep(NA,Ntaxa);Wlimit<-rep(NA,Ntaxa)
#SPECIES LATITUDINAL RANGE
Slatrange<-rep(NA,Ntaxa)
#SPECIES SAMPLED LATITUDINAL RANGE
sampledSlatrange<-rep(NA,Ntaxa)
#SPECIES THERMAL IQR RANGE
SSSTrange<-rep(NA,Ntaxa)
```

```
#DISTRIBUTION OF SHELVES AND LOCALITIES
require(oce)
par(cex=1.4)
data(coastlineWorld)
plot(coastlineWorld, clongitude=-80, clatitude=0, span=20000)
points(shelflongs,shelflats,pch=21, bg="blue", xlab="Longitude",ylab="Latitude")
points(longs,lats,pch=21, bg="yellow", cex=0.65)
```

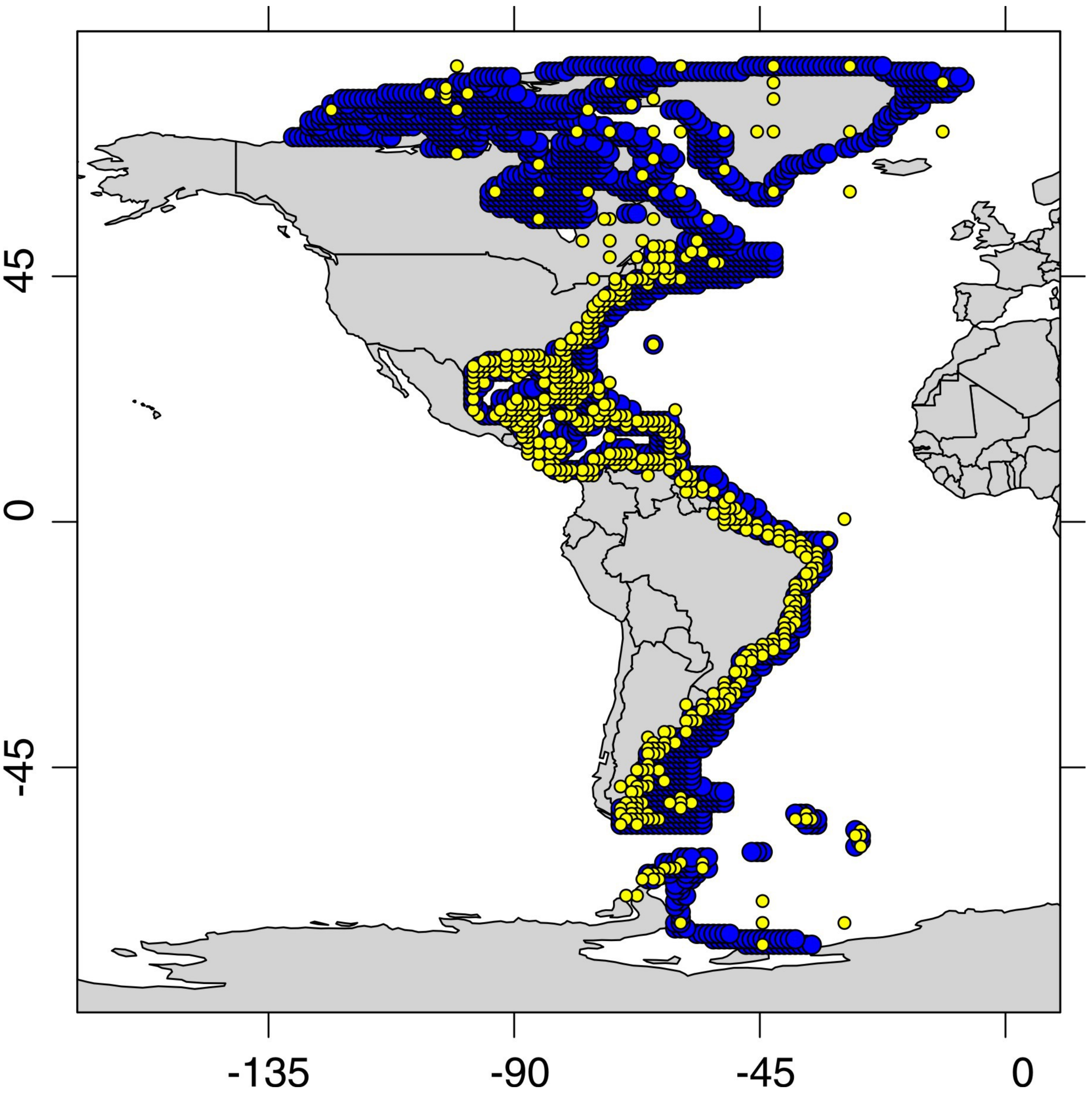

Figure C1. Distribution of 1º cells covering shelf depths in the Western Atlantic (blue), and distribution of 1º cells represented by occurrences in the empirical database (yellow). The occurrences with low spatial resolution (> 1º) were assigned to five coordinates, including their midpoint, outer latitudinal (southern and northern) coordinates, and longitudinal (eastern and western) coordinates. Therefore, some of these coordinates can fall on the continent or

open ocean. However, this type of artifact does not affect computation of modeled latitudinal ranges and thermal ranges (such coordinates do not have any sea-surface temperature).

In the simulation below, range midpoints are randomly determined by drawing coordinates of shelf cells, weighting the selection probability of each cell by its longitudinal width in km (according to the longitudinal extent in kilometers of each 1° cell). The number of range midpoints randomly placed at latitudes with a large shelf extent thus exceeds the number of range midpoints placed at latitudes with a narrow shelf extent. The vector `distance` is used for this weighting – it consists of longitudinal distances that characterize different latitudes.

```
#MODELED ARRAY WITH PRESENCE-ABSENCE IN EACH CELL FOR CELLS THAT WILL BE SAMPLED
simcomp<-array(0, dim=c(length(lats),Ntaxa))
#SOME VECTORS FOR BOOKKEEPING
secondlongint<-numeric()
addlongitude<-numeric()

#LENGTH OF ONE LONGITUDINAL DEGREE IN METERS FROM 0.5 TO 89.5 latitude
#THEN DIVIDED BY 1000 TO GET KM
longitudelats=seq(0.5,90,by=1)
distance=c(111319,111303,111252,111168,111050,110899,110714,110495,110243,109958,
109639,109288,108903,108485,108034,107550,107034,106486,105905,105292,104647,103970
,103262,102523,101752,100950,100118,99255,98362,97439,96486,95504,94493,93453,92385
,91288,90164,89012,87832,86626,85394,84135,82851,81541,80206,78847,77463,76056,7462
5,73172,71696,70198,68678,67137,65576,63994,62393,60772,59133,57475,55800,54107,
52398,50673,48932,47176,45405,43620,41822,40010,38187,36351,34504,32647,30779,
28902,27016,25121,23219,21310,19393,17471,15544,13611,11675,9735,7791,5846,3898,
1949,0)/1000

proplongitudinalweight=distance/max(distance)
finalweight<-numeric()
for (i in 1:length(shelflats)) {
       position<-which(abs(shelflats[i])==longitudelats)
       finalweight[i]<-proplongitudinalweight[position]
       }
```

For each species, we randomly select one shelf cell from the vector of shelf coordinates. This cell (`idFORxlocation`) represents a midpoint of species range, with some longitude and latitude (`xlocation` and `ylocation`). Empirical latitudinal and longitudinal ranges are halved (`xradius` and `yradius`), and then used to generate rectangles around these midpoints. First, northern, southern, eastern and western range limits are determined. Second, the portions of ranges that are placed outside the grid edge are transported into the adjacent hemisphere so that geographic ranges are not artificially truncated by grid edges. For example, if the range midpoint is placed at 80°S and at 90°E and its latitudinal radius is ~2000 km, thus approximately 20° to the north and south and crossing the pole, one equatorward range limit will be at 60°S and 90°E and another equatorward range limit will be at 80°S and 90°W. An array `gridmap` with 180 rows and 360 columns is then filled with numeric values so that the cells occupied by a given species have the column number and the cells with absences have zero values. A presence-absence array `simcomp` is then generated – it has species in columns and individual latitudinal-longitudinal 1° cells in rows. This array is then used for computations of species latitudinal and thermal ranges, and then concatenated into an array where 1° cells are concatenated into 5° latitudinal bands. The `repeat` loop is used for checking whether northern or southern range limits are located on the shelf. If not, the random sampling of midpoints is repeated and the range mid-point is re-positioned, thus approximately conserving the transect-level empirical distribution of latitudinal range sizes. An example of `gridmap` showing the rectangle with several 1° cells occupied by one randomly placed species is below. Zeros correspond to empty cells.

```
     -60.5 -59.5 -58.5 -57.5 -56.5 -55.5 -54.5 -53.5 -52.5 -51.5
50.5     0     0     0     0     0     0     0     0     0     0
51.5     0     0     0     0     0     0     0     0     0     0
52.5     0     0     0     0     0     0     0     0     0     0
53.5     0     0     0   123   124   125   126     0     0     0
54.5     0     0     0   123   124   125   126     0     0     0
55.5     0     0     0   123   124   125   126     0     0     0
56.5     0     0     0   123   124   125   126     0     0     0
57.5     0     0     0     0     0     0     0     0     0     0
```

```
58.5     0     0     0     0     0     0     0     0     0     0
59.5     0     0     0     0     0     0     0     0     0     0

################################################################################
# SIMULATION RUN FOR ALL SPECIES
################################################################################
for (i in 1:Ntaxa) {

#repeat loop for range-rejection, this can be turned off if rejection is not used
repeat {

#SAMPLE SPECIES RANGE MIDPOINTS
      idFORxlocation<-sample(c(1:length(shelflongs)),1,replace=T,prob=finalweight)
      xlocation<-shelflongs[idFORxlocation]
      ylocation<-shelflats[idFORxlocation]

#ASSIGN LATITUDINAL AND LONGITUDINAL RADIUS TO EACH SPECIES, USING EMPIRICAL
#LATITUDINAL AND LONGITUDINAL RANGES DIVIDED BY 2
yradius=Slatextent[i]/2
xradius=Slongextent[i]/2

#IDENTIFY NORTHERN AND SOUTHERN RANGE LIMIT
Nlimit[i]<-ylocation+yradius
Slimit[i]<-ylocation-yradius

#IDENTIFY EASTERN AND WESTERN RANGE LIMITS
Elimit[i]<-xlocation+xradius
Wlimit[i]<-xlocation-xradius

#IF RANGE LIMITS ARE BEYOND +-90°, TRUNCATE THEM. HOWEVER, KEEP THE #UNTRUNCATED
#LIMITS FOR LATER
Nlimit2<-Nlimit[i]
Slimit2<-Slimit[i];
if (Slimit[i]<=(-89.5)) {
      Slimit2<-Slimit[i];
      Slimit[i]<--89.5
      }
if (Nlimit[i]>=90) {
      Nlimit2<-Nlimit[i];
      Nlimit[i]<-89.5
      }
Slatrange[i]<-Nlimit[i]-Slimit[i]

################################################################################
#CORRECTING FOR TRUNCATION EFFECTS ON THE WESTERN AND EASTERN EDGES
```

```
#IF RANGE LIMITS ARE BEYOND +-180°, TRUNCATE THEM. HOWEVER, KEEP THE UNTRUNCATED
#LIMITS FOR LATER COMPUTATION
##################################################################################
if (Wlimit[i]>(-180) & Elimit[i]<180) {
      #EWmake is a vector that identifies the longitudes in the gridmap
      #longint is a vector that identifies the column locations in the gridmap
      EWmake<-c(seq(Wlimit[i], Elimit[i],by=1))
      longint<-findInterval(EWmake+0.5,gridlong)
}

#IF LONGITUDINAL RANGE LIMITS ARE TRUNCATED ON THE WESTERN SIDE
if (Wlimit[i]<=(-180)) {
      Wlimit2<-Wlimit[i]
      Wlimit[i]<--180
      remain<-180+(Wlimit2+180)
      id<-findInterval(remain+0.5, gridlong)
      if (Wlimit[i]+0.5<Elimit[i]) {
            EWmake=c(seq(gridlong[id],180,by=1),seq(Wlimit[i]+0.5, Elimit[i],by=1))
            }
      if (Wlimit[i]+0.5>Elimit[i]) {
            EWmake<-c(seq(gridlong[id],180,by=1),Wlimit[i]+0.5)
            }
      longint<-findInterval((EWmake),gridlong)
}

#IF LONGITUDINAL RANGE LIMITS ARE TRUNCATED  ON THE EASTERN SIDE
if (Elimit[i]>=180) {
      Elimit2<-Elimit[i]
      Elimit[i]<-180
      remain<--180+(Elimit2-180)
      id<-findInterval(remain+0.5, gridlong);
      if (Wlimit[i]<Elimit[i]) {
            EWmake=c(seq(Wlimit[i],Elimit[i],by=1),seq(-179.5,gridlong[id], by=1))
            }
      if (Wlimit[i]>Elimit[i]) {
            EWmake<-c(Elimit[i],  seq(-179.5,gridlong[id], by=1))
            }
      longint<-findInterval(EWmake,gridlong)
}

##################################################################################
#CORRECTING FOR TRUNCATION EFFECTS ON THE NORTHERN AND SOUTHERN EDGES
##################################################################################
first=findInterval(Slimit[i],gridlat)
second=findInterval(Nlimit[i],gridlat)
```

```
#NSmake is a vector that identifies the longitudes (columns) in the gridmap
if (first == second) {NSmake<-gridlat[first]}
if (first < second) {NSmake<-seq(gridlat[first],gridlat[second],by=1)}

#latint is a vector that identifies the latitudes (rows) in the gridmap
latint<-findInterval(NSmake,gridlat)
NSint<-latint

#NORTHERN EDGE EFFECT
#compute how much does the range extend into the other hemisphere,
#and generate the vector "secondNSint" that defines S and N limits in the #other
#hemisphere
if (Nlimit2 >= 90)  {
       secondNlimit<-90-(Nlimit2-90)
       first=findInterval(secondNlimit+0.5,gridlat)
       second=findInterval(89.5,gridlat)
       make<-seq(gridlat[first],gridlat[second],by=1)
       secondNSint<-findInterval(make, gridlat)
       firstNSinterval<-min(latint):180
       latint<-firstNSinterval
       #concatenate ranges from both hemispheres into "NSint" that
       #identifies the latitudes in gridmap
       NSint<-c(min(latint):180, rev(secondNSint))
}

#SOUTHERN EDGE EFFECT
if (Slimit2 <= -90) {
       thirdSlimit<--90-(Slimit2+90)
       first=findInterval(thirdSlimit+0.5,gridlat)
       second=findInterval(-89.5,gridlat)
       make<-seq(gridlat[second],gridlat[first],by=1)
       secondNSint<-findInterval(make, gridlat)
       NSint<-c(rev(secondNSint), 1:max(latint))
}
##################################################################################
#IDENTIFY LONGITUDINAL MIDPOINT OF RANGES - FURTHER CORRECTING FOR TRUNCATION
##################################################################################

if (length (EWmake) == 1)  {
       midlongitude<-findInterval(EWmake,gridlong)
       }
if (length (EWmake) > 1)   {
       midlongitude<-EWmake[((length(EWmake))/2)];
       midlongitude<-findInterval(midlongitude,gridlong)
       }
```

```
#IDENTIFY LATITUDINAL MIDPOINT OF SPECIES RANGE - IF THE NUMBER OF THE OCCUPIED
#CELLS ALONG LATITUDE IS EVEN OR THERE IS JUST ONE CELL OCCUPIED
if(length(NSint) == 1) {
	latitudecenter=c(NSint,NSint)
	}
if(length(NSint) > 1) {
	latitudecenter<-c(NSint[floor(length(NSint)/2)], NSint
[ceiling(length(NSint)/2)])
	}
#IDENTIFY LATITUDINAL MIDPOINT OF SPECIES RANGE - IF THE NUMBER OF THE OCCUPIED
#CELLS ALONG LATITUDE IS NOT EVEN
if (!is.integer(length(NSint)/2)) {
	latitudecenter=c(NSint [ceiling(length(NSint)/2)], NSint
[ceiling(length(NSint)/2)])
	}


##################################################################################
#FILL ALL CELLS IN A GRIDDED MAP WHERE A GIVEN SPECIES SHOULD OCCUR
##################################################################################
#GRIDDED MAP
gridmap<-matrix(data=0, nrow=length(gridlat), ncol=length(gridlong))
rownames(gridmap)=gridlat
colnames(gridmap)=gridlong
gridmap[latitudecenter[1],longint]<-longint
helpint<-longint
if (length(helpint) > 1) {
	if(latitudecenter[1] < 180)	{
		for (f in (latitudecenter[2]+1): max(latint)) {
		gridmap[f,helpint]<-helpint
		}
	}
}
helpint<-longint
#THIS LOOP ADDS CELLS TO EACH LATITUDE
if (length(helpint) > 1) {
	for (f in ((latitudecenter[1]-1)):min(latint)) {
	gridmap[f,helpint]<-helpint
	}
}

##################################################################################
#IDENTIFY OCCUPIED CELLS IN RANGES WITH NORTHERN TRUNCATION
##################################################################################
if (Nlimit2 > 90)   {
for (z in 1:length(longint)) {
	if (longint[z] <= 180) {secondlongint[z]<-longint[z]+180}
	if (longint[z] > 180) {secondlongint[z]<-longint[z]-180}
	}
	helpint<-secondlongint
	addlongitude=helpint
	if (length(helpint) == 1) {gridmap[180,helpint]<-helpint}
	if (length(helpint) > 1) {
		for (f in 180:min(secondNSint)) {
			gridmap[f,helpint]<-helpint
			}
	}
}

##################################################################################
#IDENTIFY OCCUPIED CELLS IN RANGES WITH SOUTHERN TRUNCATION
##################################################################################
if (Slimit2 < -90)  {
	for (z in 1:length(longint)) {
		if (longint[z] <= 180) {secondlongint[z]<-longint[z]+180}
		if (longint[z] > 180) {secondlongint[z]<-longint[z]-180}
		}
	helpint<-secondlongint
	addlongitude=helpint
	if (length(helpint) > 1) {
		for (f in 1:max(secondNSint)) {
			gridmap[f,helpint]<-helpint
			}
	}
}

##################################################################################
#CHECK RANGE REJECTION - HERE, ASSESSING JUST NORTHERN AND SOUTHERN COORDINATES AT
#GEOGRAPHICAL RANGE MIDPOINT
##################################################################################
#IDENTIFY LATITUDE AND LONGITUDE OF NORTHERN AND SOUTHERN LIMITS TO CHECK WHETHER
#THEY ARE LOCATED ON SHELF
if (Slimit2 < -90)  {
	Sfocal<-paste(gridlat[min(NSint)],gridlong[round(median(addlongitude),0)])
	}
if (Nlimit2 >  90) {
	Nfocal<-paste(gridlat[max(NSint)],gridlong[round(median(addlongitude),0)])
	}
if (Slimit2 >= -90) {
	Sfocal<-paste(gridlat[min(NSint)],gridlong[midlongitude])
	}
if (Nlimit2 <=  90) {
	Nfocal<-paste(gridlat[max(NSint)],gridlong[midlongitude])
	}

#IF NORTHERN OR SOUTHERN RANGE EDNPOINT IS LOCATED ON SHELF, STOP THE LOOP
if (!is.na(pmatch(Sfocal, shelf)) | !is.na(pmatch(Nfocal, shelf))) {break}
}

##################################################################################
#GENERATE SPECIES-BY-CELL TABLES WITH PRESENCE-ABSENCE OF SPECIES (simcomp)
##################################################################################
#gridmap[(min(latint)-3):(max(latint)+3), (min(longint)-3):(max(longint)+3)]

x<-numeric(); y<-numeric(); SSTtemp=numeric()
for (v in 1:length(lats)) {
	idx<-which(gridlat==lats[v])
	idy<-which(gridlong==longs[v])
		if (gridmap[idx,idy] > 0) {
			simcomp[v,i]=1
			y<-append(y,lats[v],length(y))
			x<-append(x,longs[v],length(x))
SSTtemp<-append(SSTtemp,SSTgridmap[idx,idy],length(SSTtemp))

			}
	}
##################################################################################
#COMPUTE LATITUDINAL RANGE OF THE SAMPLED SPECIES, CONDITIONED BY THE GRID CELLS
#THAT ARE ACTUALLY SAMPLED IN THE EMPIRICAL DATASET
##################################################################################
polys<-cbind(x,y)
if (nrow(polys) > 1) {sampledSlatrange[i]<-max(y)-min(y)}
if (nrow(polys)==1){sampledSlatrange[i]=0.5}

##################################################################################
#IDENTIFY THOSE GRID CELLS THAT ARE OCCUPIED BY A GIVEN SPECIES IN THE GRIDDED MAP
#WITH TEMPERATURE DATA AND COMPUTE SPECIES THERMAL RANGES
##################################################################################
SSSTrange[i]=IQR(SSTtemp, na.rm=T)
print(i)
}		#finish the loop for species i
```

Here, latitudinal and thermal ranges are assigned to cells within arrays occupied by a given species.

```
##################################################################################
#GENERATE ARRAYS WITH SPECIES IN ROWS AND 1° CELLS IN COLUMNS
```

```
##################################################################################
Slatcomp<-array(NA,dim=c(ncol(simcomp),nrow(simcomp))) #array for lat. ranges
Ssampledlatcomp<-array(NA,dim=c(ncol(simcomp),nrow(simcomp)))
SSSTcomp<-array(NA,dim=c(ncol(simcomp),nrow(simcomp))) #array for thermal ranges

for (i in 1:ncol(simcomp)) {
	for (k in 1:nrow(simcomp)) {
		if (simcomp [k,i] > 0) {
			Slatcomp [i,k]<-Slatrange[i]*111.25
			Ssampledlatcomp [i,k]<-sampledSlatrange[i]*111.25
			SSSTcomp [i,k]<-SSSTrange[i]
			}
		}
	}
Scomp<-t(simcomp)
```

We assign individual 1º cells with presence-absence data and range-size information to 5º bands, using the vector `scaledlats`.

```
##################################################################################
#CONCATENATING 1° CELLS INTO LATITUDINAL BANDS
#GENERATING COMPOSITIONAL TABLES FOR PER-BAND ANALYSES OF DIVERSITY AND RANGE SIZE
##################################################################################
latsequence<-seq(-87.5,87.5,by=5)
scaledlats=numeric()
for (i in 1:length(lats)) {
	temp=findInterval(lats[i],latsequence-2.5)
	scaledlats[i]=latsequence[temp]
	}
newlatsequence=sort(unique(scaledlats))
latsequenceIDs=numeric()
for (i in 1:length(newlatsequence)) {
	latsequenceIDs[i]=which(newlatsequence[i]==latsequence)
	}
binlatrangecomp=t(aggregate(t(Slatcomp), by=list(scaledlats), mean, na.rm=T)[,-1])
bincomp=t(aggregate(t(Scomp), by=list(scaledlats), max, na.rm=T)[,-1])
binSSSTrangecomp=t(aggregate(t(SSSTcomp), by=list(scaledlats), mean, na.rm=T)[,-1])
binsampledlatrangecomp=t(aggregate(t(Ssampledlatcomp), by=list(scaledlats), mean,
na.rm=T)[,-1])
colnames(bincomp)<-sort(unique(scaledlats))
colnames(binlatrangecomp)<-sort(unique(scaledlats))
latbins=colnames(bincomp)

#RICHNESS
```

```
BANDrichness<-apply(bincomp, MARGIN=2, sum)
BANDmediansampledlatext<-apply(t(binsampledlatrangecomp),MARGIN=1,median,na.rm=T)
BANDmedianlatext<-apply(t(binlatrangecomp),MARGIN=1,median,na.rm=T)
BANDmedianspSSTextent<-apply(t(binSSSTrangecomp),MARGIN=1,median,na.rm=T)

##################################################################################
#PLOTS
##################################################################################
par(cex=1.4)
par(mfrow=c(2,2))
plot(Slatrange, SSSTrange, pch=21, bg="gray51", cex=1.3,  xlab="Species latitudinal
range (km)", ylab="Species thermal range", cex.main=0.9)
plot(newlatsequence,BANDmediansampledlatext, pch=16, cex=1.3, type="b",
xlab="Latitude", ylab="Median latitudinal range", ylim=c(0,10000), cex.main=0.9)
```

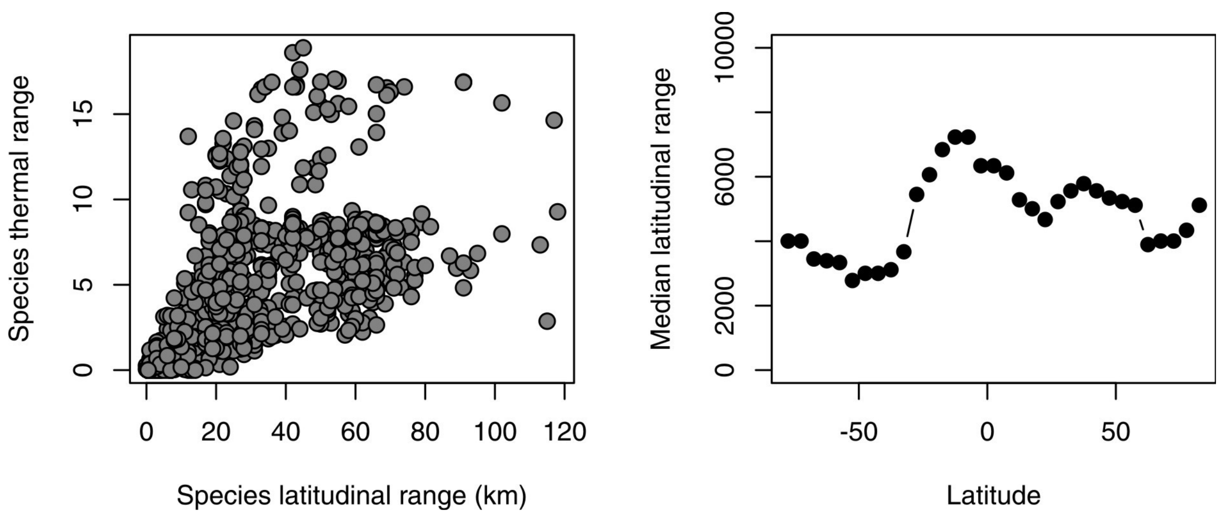

Figure C2. The relationship between species latitudinal and thermal ranges (left), and the latitudinal gradient in median per-band latitudinal range size (right), both predicted by one simulation run.

```
par(mfrow=c(2,2))
plot(newlatsequence,BANDmedianspSSTextent, pch=16, cex=1.3, type="b",
xlab="Latitude", ylab="Spatial thermal range (IQR)", ylim=c(0,20), cex.main=0.9)
plot(newlatsequence,BANDrichness, pch=16, cex=1.3, type="b", xlab="Latitude",
ylab="Species richness", ylim=c(0,500), cex.main=0.9)
```

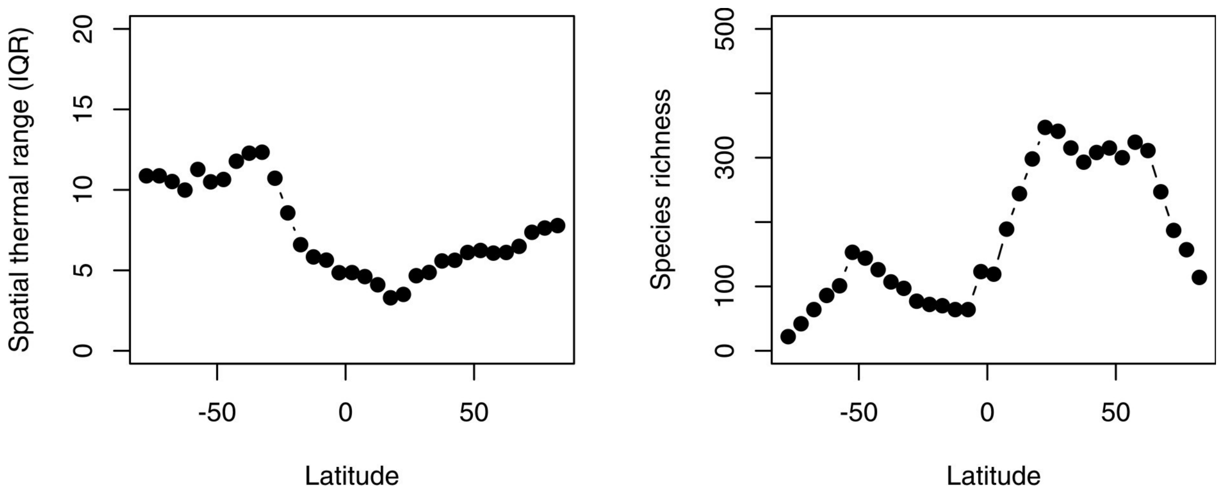

Figure C3. The latitudinal gradient in median thermal range size (right) and in per-band species richness (right) predicted by one simulation.