R code to calculate national and regional indices of coverage by protected areas of marine, terrestrial and freshwater Key Biodiversity Areas

Developed by Maria Dias, BirdLife International

This document contains R code to overlap spatial data from the *World Database of Protected Areas* onto shapefiles of Key Biodiversity Areas to output a file with a row for each part of each KBA that is overlapped by a unique combination of protected areas, giving the earliest year of designation for the relevant protected area(s) (Annex 1).

It also provides R code to calculate trends over time in protected area coverage of KBAs for each country and region (Annex 2), and R code to plot graphs using the output data (Annex 3).

The methods followed are published in Butchart et al. 2015 (Conservation Letters 2015, 8: 329–337).

Annex 1. R code to overlap spatial data from the *World Database of Protected Areas* onto shapefiles of Key Biodiversity Areas to output a file with a row for each part of each KBA that is overlapped by a unique combination of protected areas, giving the earliest year of designation for the relevant protected area(s).

This code requires the following input files (layer in a gdb or table in CSV format):

1. Layer "KbaMapGlobal_POL": polygon layer with the boundaries of all kbas in the world. The attribute table must have the following fields: SitRecID; Country; ISO3.

2. Layer "WDPA_Apr_2016_MER_public": polygon layer with the boundaries of Protected Areas in the world. The attribute table must have the following fields: WDPAID; STATUS_YR; ISO3

3. Table "classif_kbas_FINAL2017.csv": table identifying the different types of KBA (marine, freshwater or mountain and IBA, KBA and Aze status).

SitRecID	marine	terrestrial	coastal	Freshwater	mountain	IbaStts	KbaStts	AzeStts

11	*1*	*1*	*1*	*0*	*0*	*confirmed*	*confirmed*	*-*

4. Table "Iso_countries.csv": list of ISO codes, country names and respective regions/other classifications (note 0 values – not NA – for missing values in “Region”, “Developing”,”LDC” , “LLDC_SIDS”, “IPBES_region” and “M49_Region fields”)

ordem	country_SIS	countryname	region1	ISO3	ISO_BL	ISO_SDG	uname_BL	uname_SDG	Region	Developing	LDC	LLDC_SIDS	IPBES_region	M49_Region
247	*High Seas*	*Areas Beyond National Jurisdiction*	*High Seas*	*ABNJ*	*ABNJ*	*ABNJ*	*High Seas*	*High Seas*	*0*	*0*	*0*	*0*	*0*	*0*
202	*Aruba*	*Aruba*	*Caribbean Islands*	*ABW*	*ABW*	*ABW*	*Aruba*	*Aruba*	*Latin America & the Caribbean*	*Developing*	*0*	*SIDS*	*Americas*	*Latin America and the Caribbean (MDG=M49)*

The code outputs the files listed above, all sharing the same structures (but based on different classifications for the field “region”):

1. Input data for R global KBAs.csv
2. Input data for R COUNTRY KBAs.csv
3. Input data for R Region KBAs.csv
4. Input data for R Developing KBAs.csv
5. Input data for R LDC KBAs.csv
6. Input data for R LLDC_SIDS KBAs.csv
7. Input data for R M49_Region KBAs.csv
8. Input data for R IPBES_region KBAs.csv

Siteid	region	country	perprotected	year
8002	*LLDC*	*Afghanistan*	*0.856469*	*1978*

Text that is highlighted in yellow should be amended to give the appropriate folder names and file paths. Topoly errors might prevent the script to run properly for some countries (see IMPORTANT NOTES section of the script).

```
# overlap pa-kba calculator v1.  26/06/2016
# Maria Dias/BirdLife International maria.dias@birdlife.org
# Script to estimate the overlap of PAs and KBAs (giving earliest year of designation)

### IMPORTANT NOTES
# Given the complexity of the code, it is strongly recommended to copy-paste the code into an R editor (e.g. Tinn-R)
# The minimum requirement to run the script is a 16 GB RAM machine
# WDPA layer is extremely large, can take several minutes (up to half an hour) to upload
# to facilitate the process, the code runs the script and saves a file country by country. This avoids reanalysing the countries already done in case of error
# it might occur an error preventing to calculate which kbas overlap with ibas for the entire country - error in gIntersects step. Examples: Jordan and USA. These situations
are easily identifiable in the final csv file (filter by SitRecID=9999, kba=9999). Additional lines of codes should be run if this happens (see codes from "redoiso=F" - around
line 260, to "## save files for countries" - around line 450; change redoiso=F to redoiso=T), and the results added to the overall table


### packages to instal
kpacks <- c('sp','maptools', 'rgdal', 'adehabitat','geosphere','fields','spatstat','maps','rgeos','data.table')  ### install automatically the relevant packages
new.packs <- kpacks[!(kpacks %in% installed.packages()[,"Package"])]
if(length(new.packs)) install.packages(new.packs)
lapply(kpacks, require, character.only=T)
remove(kpacks, new.packs)
#########################################################

### working folders and files to import
folder=("C:/OvlPAKba")  ## set the working directory
setwd(folder)
finfolder=" C:/OvlPAKba/filespcountry" #folder where the files per country will be saved

kbas<-readOGR(' C:/SITES/Sites.gdb',layer='KbaMapGlobal_POL')    ### loads the kba layer from the geodatabase specified; migth take several minutes to load
pas<- readOGR('C:/ wdpa/Jan_17_WDPA_Final_Data.gdb',layer='WDPA_Jan_17_Merge')  ### reads the 2017 pa layer from the geodatabase specified;  migth take several
minutes to load

# to read from shapefiles instead of geodatabases
#kbas<-readOGR(folder,layer='kbas2F')  #open shapefiles saved in the path defined above (folder)
#pas<- readOGR(' C:/WDPA/WDPA_April_2016_MER.gdb',layer='WDPA_Apr_2016_MER_public')  ### reads the pa layer from the geodatabase specified

tabmf=read.csv("classif_kbas_FINAL2017.csv")   ## file with types of kbas
```

```
isos=read.csv("Iso_countries.csv")          ## file with ISO codes; should be stored in the wkfolder specified above

if ("SitRcID"%in%names(kbas@data)) kbas$SitRecID=kbas$SitRcID
pas$STATUS_YR<-as.numeric(as.character(pas$STATUS_YR))

################################  custom functions
lu=function (x=x) length(unique(x))
#####################################

kbas$ISO3=as.character(kbas$ISO3)
unique(kbas$ISO3)
unique(kbas$Country[kbas$ISO3=="---"])
kbas$ISO3[kbas$ISO3=="---"&kbas$ Country =="High Seas"]="ABNJ"
kbas$ISO3[kbas$ISO3=="---"&kbas$ Country!="High Seas"]="RUS"

## transboundary pas
cnpa=data.frame(ISO3=unique(pas@data$ISO3))
cnpa$nchart=nchar(as.character(cnpa$ISO3))
cnpa=cnpa[cnpa$nchart>4,]
cnpa
transb=data.frame()
for (g in 1:nrow(cnpa))
{
#g=2
cnpa1=cnpa[g,]
sp=substr(cnpa1$ISO3, 4, 5)
if (sp=="; ")
{
cnpa2=data.frame(ISO3=strsplit(as.character(cnpa1$ISO3), split="; ")[[1]])
cnpa2$oISO3=as.character(cnpa1$ISO3)
}
if (sp!="; ")
{
cnpa2=data.frame(ISO3=strsplit(as.character(cnpa1$ISO3), split=";")[[1]])
cnpa2$oISO3=as.character(cnpa1$ISO3)
}
```

```
    transb=rbind(transb, cnpa2)
    }
    transb


    DgProjw <- CRS(proj4string(kbas))

    listcnts=unique(kbas$ISO3)
    listcnts=listcnts[listcnts!="---"]
    lu(listcnts)

    #listcnts=listcnts[listcnts%in%unique(pas@data$ISO3)]
    lu(listcnts)


    ##############################
    ##### OVERLAP WITH PROTECTED AREAS
    ### per country

    finaltab=data.frame()
    tt=proc.time()
    for (x in 1:length(listcnts)) #length(listcnts)
    #for (x in 1: length(listcnts)) #
    {      #starts loop for all countries
    #country="SOM"   # ISO name

    #x=2
    #x=which(listcnts==country)
    country=as.character(listcnts[x])
    print(country)

    print(x)

    kbac=kbas[kbas$ISO3==country,]
    pac=pas[pas$ISO3==country,]  ## protected areas within the country

```

```
if (country%in%transb$ISO3)
{
iso3=c(country,transb$oISO3[country==transb$ISO3])
iso3
pac=pas[pas$ISO3%in%iso3,]
}

countryn=kbac@data$Country[1]
print(countryn)
if (country=="RUS") countryn="Russia"
print(countryn)

plotit=F
if(plotit){
plot(kbac, border=3)
plot(pac, add=T)
title(main=paste(countryn, country))
box()
axis(1)
axis(2)
}

length(pac)
if (length(pac)==0) areasov=data.frame(SitRecID=kbac$SitRecID, kba=9999, ovl=0, year=0, nPAs=0, percPA=0,ISO=country,COUNTRY=countryn)

if (length(pac)>0)
{
ovkba=NULL
ovkba=tryCatch({gIntersects(pac,kbac, byid=T)}, error=function(e){})
length(ovkba)

if (length(ovkba)==0) areasov=data.frame(SitRecID=9999, kba=9999, ovl=9999, year=9999, nPAs=9999, percPA=9999,ISO=country,COUNTRY=countryn)

if (length(ovkba)>0)
{

```

```
areasov=data.frame()
#####next bit is new code which re-assigns missing years to a randomly selected year from PAs in the respective country
if (min(pac@data$STATUS_YR)==0)
{
ryears=pac@data$STATUS_YR[pac@data$STATUS_YR>0]
if (length(ryears)==0) ryears=pas@data$STATUS_YR[pas@data$STATUS_YR>1986]
if (length(ryears)==1) ryears=c(ryears, ryears)

pac@data$STATUS_YR[pac@data$STATUS_YR==0]= sample(ryears,nrow(pac@data[pac@data$STATUS_YR==0,]), replace=T) ## selects a year randomly from the pool of
possible years
}
############

for (z in 1:length(kbac))    #length(kbac)
{ ## starts loop for all kbas in the country
#z=11
#z=which(kbac$SitRecID=="23937")
#print(z)
kbaz=kbac[z,]
head(kbaz@data)
akba=9999
akba=suppressWarnings(tryCatch({gArea(kbaz, byid=FALSE)}, error=function(e){}))
akba

length(which(ovkba[z,]==T))

if (length(which(ovkba[z,]==T))>0)  ### when at least 1 pa overlaps with the kba
{
#win.graph()

pacz=pac[which(ovkba[z,]==T),]
length(pacz)
pacz@data

if (plotit){
plot(kbaz)
```

```
plot(pacz, col=rgb(0,0,.8,0.2), border=0, add=T)
}

yearspacz=pacz@data$STATUS_YR
ovf=NULL

ovf=tryCatch({gIntersection(pacz,kbaz, byid=T)}, error=function(e){}) ## spatial intersection kba and all pas overlapping
length(ovf)
class(ovf)

if (class(ovf)=="SpatialCollections")
{
ovf=ovf@polyobj
rnames=paste(row.names(pacz@data),row.names(kbaz@data))
yearspacz=yearspacz[which(rnames%in%row.names(ovf))]
}

if (class(ovf)!="SpatialPolygons")  ovf=NULL

if (class(ovf)=="SpatialPolygons"&length(yearspacz)>0)
{

ovfpol=SpatialPolygonsDataFrame(ovf, data=data.frame(id=row.names(ovf), row.names=row.names(ovf), year=yearspacz)) ## converts  ovf in a spatial polygon data frame
ovfpol@data
years=sort(unique(ovfpol$year))
years

year1=min(years)
ovf1=ovfpol[ovfpol$year==year1,]
length(ovf1)
ovf11=NULL
ovf11=tryCatch({unionSpatialPolygons(ovf1, ovf1@data$year)}, error=function(e){})
length(ovf11)
if(plotit) plot(ovf11, add=T, col=2)

ovlz=suppressWarnings(tryCatch({gArea(ovf11, byid=FALSE)}, error=function(e){}))
```

```
if (length(ovlz)==0) ovlz=9999
ovlz

areasov1<-data.frame(SitRecID=kbaz@data$SitRecID, kba=akba, ovl=ovlz, year=year1, nPAs=length(ovf1))

areasov1

if (length(years)>1)
{
for (w in 2:length(years))
{
#w=2
rema=1-(sum(areasov1$ovl[areasov1$ovl<9999])/akba)  ## to see if there is still any area left
rema
if (rema>0.02)
{
year2=years[w]
year2
ovf2=ovfpol[ovfpol$year==year2,]
length(ovf2)
ovf22=NULL
ovf22=tryCatch({unionSpatialPolygons(ovf2, ovf2@data$year)}, error=function(e){})
length(ovf22)
if(plotit) plot(ovf22, add=T, col=w+1)
ovfprev=ovfpol[ovfpol$year<year2,]
if(plotit) plot(ovfprev, add=T, col=1)

ovf23=NULL
ovf23=tryCatch({gDifference(ovf22, ovfprev)}, error=function(e){})

if(plotit) plot(ovf23, add=T, col="grey")

ovlz=suppressWarnings(tryCatch({gArea(ovf23, byid=FALSE)}, error=function(e){}))
if (length(ovlz)==0) ovlz=9999
ovlz

```

```
areasov1=rbind(areasov1,data.frame(SitRecID=kbaz@data$SitRecID, kba=akba, ovl=ovlz, year=year2, nPAs=length(ovf2)))
areasov1
}
}
}
areasov1

}  # ends loop for    class(ovf)=="SpatialPolygons"
if (length(ovf)==0|class(ovf)!="SpatialPolygons") areasov1=data.frame(SitRecID=kbaz@data$SitRecID, kba=akba, ovl=9999, year=0, nPAs=0)  ## error in spatial overlap
}  ## ends loop for PAs overlapping with the KBA
if (length(which(ovkba[z,]==T))==0) areasov1=data.frame(SitRecID=kbaz@data$SitRecID, kba=akba, ovl=0, year=0, nPAs=0)   ## no pas overlapping the kba
areasov=rbind(areasov,areasov1)
}  ## ends loop for all kbas in the country

areasov$percPA=areasov$ovl/areasov$kba
areasov
max(areasov$perc)
areasov$ISO=country
areasov$COUNTRY=countryn

} # ends loop for ovlkba>0
}  ## ends loop for length(pac)>1

finaltab=rbind(finaltab,areasov)

tname=paste(finfolder,"/",countryn, ".csv", sep="")
tname
write.csv(areasov, tname, row.names=F)

}
(proc.time()-tt)[1]/60 ## time in minutes

head(finaltab)
str(finaltab)
lu(finaltab$x)
write.csv(finaltab, "finaltab.csv", row.names=F)
```

```
### end here
#################################################################################################

###################### countries with severe geometry problems (preventing gIntersects to run)

### USA x=226     ### Jordan (JOR) x=100
redoiso=F
if (redoiso)
{
x=which(listcnts=="USA")
x
country=as.character(listcnts[x])
print(country)

kbac=kbas[kbas$ISO3==country,]
pac=pas[pas$ISO3==country,]

if (country%in%transb$ISO3)
{
iso3=c(country,transb$oISO3[country==transb$ISO3])
iso3
pac=pas[pas$ISO3%in%iso3,]
}

countryn=kbac@data$Country[1]
print(countryn)

plotit=F
if(plotit){
plot(cntc, border=2, lwd=2)
plot(kbac, border=3,add=T)
plot(pac, add=T)
title(main=paste(countryn, country))
box()
axis(1)
axis(2)
```

```
}

length(pac)

validpas=data.frame(WDPAID=pac@data$WDPAID, valid=0)
for (l in 1:length(pac))
{
valid=tryCatch({gIsValid(pac[l,])}, error=function(e){})
if (length(valid)>0) validpas$valid[l]=valid
}
#write.csv(validpas, "validpas_USA.csv", row.names=F)
#validpas=read.csv("validpas_USA.csv")
validpas1=which(validpas$valid==1)
pac=pac[validpas1,]
length(pac)
sum(validpas$valid)


###########################
ovkba=NULL
ovkba=tryCatch({gIntersects(pac,kbac, byid=T)}, error=function(e){})
length(ovkba)

if (length(ovkba)==0) areasov=data.frame(SitRecID=9999, kba=9999, ovl=9999, year=9999, nPAs=9999, percPA=9999,ISO=country,Country=countryn, x=x)

if (length(ovkba)>0)
{

areasov=data.frame()

#####next bit is new code which re-assigns missing years to a randomly selected year from PAs in the respective country
if (min(pac@data$STATUS_YR)==0)
{
ryears=pac@data$STATUS_YR[pac@data$STATUS_YR>0]
if (length(ryears)==0) ryears=pas@data$STATUS_YR[pas@data$STATUS_YR>1986]
if (length(ryears)==1) ryears=c(ryears, ryears)
```

```
pac@data$STATUS_YR[pac@data$STATUS_YR==0]= sample(ryears,nrow(pac@data[pac@data$STATUS_YR==0,]), replace=T) ## selects a year randomly from the pool of
possible years
}

for (z in 1:length(kbac))    #length(kbac)
{ ## starts loop for all kbas in the country
#z=11
#z=which(kbac$SitRecID=="23937")
#print(z)
kbaz=kbac[z,]
head(kbaz@data)
akba=9999
akba=suppressWarnings(tryCatch({gArea(kbaz, byid=FALSE)}, error=function(e){}))
akba

length(which(ovkba[z,]==T))

if (length(which(ovkba[z,]==T))>0)  ### when at least 1 pa overlaps with the kba
{
#win.graph()

pacz=pac[which(ovkba[z,]==T),]
length(pacz)
pacz@data

if (plotit){
plot(kbaz)
plot(pacz, col=rgb(0,0,.8,0.2), border=0, add=T)
}

yearspacz=pacz@data$STATUS_YR
ovf=NULL

ovf=tryCatch({gIntersection(pacz,kbaz, byid=T)}, error=function(e){}) ## spatial intersection kba and all pas overlapping
length(ovf)
class(ovf)
```

```
397
398     if (class(ovf)=="SpatialCollections")
399     {
400     ovf=ovf@polyobj
401     rnames=paste(row.names(pacz@data),row.names(kbaz@data))
402     yearspacz=yearspacz[which(rnames%in%row.names(ovf))]
403     }
404
405     if (class(ovf)!="SpatialPolygons")  ovf=NULL
406
407     if (class(ovf)=="SpatialPolygons"&length(yearspacz)>0)
408     {
409
410     ovfpol=SpatialPolygonsDataFrame(ovf, data=data.frame(id=row.names(ovf), row.names=row.names(ovf), year=yearspacz)) ## converts  ovf in a spatial polygon data frame
411     ovfpol@data
412     years=sort(unique(ovfpol$year))
413     years
414
415     year1=min(years)
416     ovf1=ovfpol[ovfpol$year==year1,]
417     length(ovf1)
418     ovf11=NULL
419     ovf11=tryCatch({unionSpatialPolygons(ovf1, ovf1@data$year)}, error=function(e){})
420     length(ovf11)
421     if(plotit) plot(ovf11, add=T, col=2)
422
423     ovlz=suppressWarnings(tryCatch({gArea(ovf11, byid=FALSE)}, error=function(e){}))
424     if (length(ovlz)==0) ovlz=9999
425     ovlz
426
427     areasov1=data.frame(SitRecID=kbaz@data$SitRecID, kba=akba, ovl=ovlz, year=year1, nPAs=length(ovf1))
428     areasov1
429
430     if (length(years)>1)
431     {
432     for (w in 2:length(years))
```

```
{
#w=2
rema=1-(sum(areasov1$ovl[areasov1$ovl<9999])/akba)  ## to see if there is still any area left
rema
if (rema>0.02)
{
year2=years[w]
year2
ovf2=ovfpol[ovfpol$year==year2,]
length(ovf2)
ovf22=NULL
ovf22=tryCatch({unionSpatialPolygons(ovf2, ovf2@data$year)}, error=function(e){})
length(ovf22)
if(plotit) plot(ovf22, add=T, col=w+1)
ovfprev=ovfpol[ovfpol$year<year2,]
if(plotit) plot(ovfprev, add=T, col=1)

ovf23=NULL
ovf23=tryCatch({gDifference(ovf22, ovfprev)}, error=function(e){})

if(plotit) plot(ovf23, add=T, col="grey")

ovlz=suppressWarnings(tryCatch({gArea(ovf23, byid=FALSE)}, error=function(e){}))
if (length(ovlz)==0) ovlz=9999
ovlz

areasov1=rbind(areasov1,data.frame(SitRecID=kbaz@data$SitRecID, kba=akba, ovl=ovlz, year=year2, nPAs=length(ovf2)))
areasov1
}
}
}
areasov1

}  # ends loop for    class(ovf)=="SpatialPolygons"
if (length(ovf)==0|class(ovf)!="SpatialPolygons") areasov1=data.frame(SitRecID=kbaz@data$SitRecID, kba=akba, ovl=9999, year=0, nPAs=0)  ## error in spatial overlap
}  ## ends loop for PAs overlapping with the KBA
```

```
469 if (length(which(ovkba[z,]==T))==0) areasov1=data.frame(SitRecID=kbaz@data$SitRecID, kba=akba, ovl=0, year=0, nPAs=0)   ## no pas overlapping the kba
470 areasov=rbind(areasov,areasov1)
471 }  ## ends loop for all kbas in the country
472
473 areasov$percPA=areasov$ovl/areasov$kba
474 areasov
475 max(areasov$perc)
476 areasov$ISO=country
477 areasov$COUNTRY=countryn
478
479 } # ends loop for ovlkba>0
480
481 tname=paste(finfolder,"/",countryn, ".csv", sep="")
482 tname
483 write.csv(areasov, tname, row.names=F)  ## save files for countries that needed to be reanalysed
484 }
485 ######################################
486 ##############################################
487
488 isofiles=F ## if we need to bind all country files in a single table
489
490 if (isofiles)
491 {
492 fls=dir(finfolder)
493 final1=data.frame()
494 for (y in 1:lu(fls)) final1=rbind(final1, read.csv(paste(finfolder, fls[y], sep= '/')))
495 }
496 #write.csv(final1, "finaltab.csv", row.names=F)
497
498
499 lu(final1$ISO)
500
501 if (isofiles==F) final1=finaltab
502 #final1=read.csv("finaltab.csv")
503 final=final1[final1$ovl!=9999,]
504 lu(final1$SitRecID)-lu(final$SitRecID) # number of KBAs with problems of geometry
```

```
505
506 str(final)
507 str(tabmf)
508
509 tabf=merge(final, tabmf, by="SitRecID")
510 str(tabf)
511 head(tabf)
512 max(tabf$percPA)
513 tabf$percPA[tabf$percPA>1]=1
514
515 max(tabf$percPA)
516 resf=with(tabf, aggregate(percPA, list(SitRecID=SitRecID), sum))
517 max(resf$x)
518 nrow(resf[resf$x>1,]) # number of kbas for which the sum of the overlap of the multiple years is more than 1 (can be due to geometry oversimplification in gDifference step
519
520 kbas2fix=unique(resf$SitRecID[resf$x>1])
521 lu(kbas2fix)
522 #rescale kbas perc prot
523 if (length(kbas2fix)>0) for (k in 1:length(kbas2fix))
524 tabf[tabf$SitRecID==kbas2fix[k],]$percPA=tabf[tabf$SitRecID==kbas2fix[k],]$percPA/sum(tabf[tabf$SitRecID==kbas2fix[k],]$percPA)
525
526 #check if was fixed (result should be 1)
527 max(with(tabf, aggregate(percPA, list(SitRecID=SitRecID), sum))$x)
528
529
530 ## link regions
531 isos$ISO<-isos$ISO3
532
533 str(tabf)
534 tabff=merge(tabf, isos, by="ISO")
535 str(tabff)
536
537 unique(tabff$year)
538 tabff$year[tabff$year==0&tabff$ovl!=0]=3000  ## PAs with no year of designation; assume 3000 for posterior randomization (NOTE that changed in 2017; random process
539 was added before the spatial overlap)
540
```

```
unique(tabff$ovl[tabff$year==3000]) ## should be NULL; only the kbas with 0 overlap with PAs have year=0, because random year was added to the codes before the overlapping step (modified in 2017)

#write.csv(tabff, "tabff.csv", row.names=F)
#tabff=read.csv("tabff.csv")

#tables to export

fields=c("COUNTRY","Region","Developing","LDC","LLDC_SIDS","IPBES_region", "M49_Region")


tfname="Input data for R global KBAs.csv"
tfname
tab2export=with(tabff, data.frame(siteid=SitRecID, country=COUNTRY, perprotected=percPA*100, year=year))
str(tab2export)
head(tab2export)
tail(tab2export)
write.csv(tab2export,tfname, row.names=F)

for (f in 1:length(fields))
{
#f=7
ff=as.character(fields[f])
ff

tfname=paste("Input data for R ", ff, "KBAs.csv")
tfname
tab2export=with(tabff, data.frame(siteid=SitRecID, region=tabff[,which(names(tabff)==ff)], country=COUNTRY, perprotected=percPA*100, year=year))
str(tab2export)
head(tab2export)
tail(tab2export)
write.csv(tab2export,tfname, row.names=F)

}
```

Annex 2. R code to calculate trends over time in protected area coverage of KBAs for each country and region using the output from running the R code in Annex 1.

This code requires an input file (CSV format) with column headings:

Siteid; region; country; perprotected; year

The input file should have multiple rows per KBA for each part of a KBA that is overlapped by a unique combination of protected areas, with the percentage of the entire KBA extent that is covered, and the earliest year of designation for any of the relevant protected areas. Note that "perprotected" needs to be a percentage not proportion. Where the year of protected area designation is unknown (not documented in the WDPA), put 3000 in the year column. Where the site is unprotected, put 0 in both the perprotected and the year columns.

The code outputs files showing trends in protected area coverage of KBAs for (a) each country listed in the input file, (b) each region listed in the input file, and (c) globally. Each file has values for

Year; 95CIlowCount; 95CImidCount; 95CIhiCount; 95CIlowPercentage; 95CImidPercentage; 95CIhiPercentage; region

i.e. giving the number and percentage of sites that are completely (i.e. >98%) covered by protected areas, for each year from 1900 to the final year set in the highlighted text in the code.

Text that is highlighted should be amended to give the appropriate folder names and file paths, number of randomisations (if this is to be adjusted) and final year to extrapolate the results to.

NOTE 2017: The code was changed in 2017 to 1) read automatically all the input files from the previous step and save the output files for each type of KBA 2) calculate the mean percentage of area covered by protected areas and 3) ignore the randomization of non-known years, as that step was included before the spatial overlap, in the previous code. Therefore, the number of randomizations needed should be set as 1

Two additional tables are needed, one with the classification of sites as marine, terrestrial, etc ("*classif_kbas_FINAL2017.csv*"), and other with the list of imput files (table "*in_out_files.csv*"):

inputfile	**global**	**outputfile1**	**code**
Input data for R global KBAs.csv	1	Output data for R global KBAs	global
Input data for R COUNTRY KBAs.csv	0	Output data for R country KBAs	country

Input data for R Region KBAs.csv	0	Output data for R Region KBAs	region
Input data for R Developing KBAs.csv	0	Output data for R Developing KBAs	developing
Input data for R LDC KBAs.csv	0	Output data for R LDC KBAs	LDC
Input data for R LLDC_SIDS KBAs.csv	0	Output data for R LLDC_SIDS KBAs	LLDC_SIDS
Input data for R IPBES_region KBAs.csv	0	Output data for R IPBES_region KBAs	IPBES
Input data for R M49_Region KBAs.csv	0	Output data for R M49_Region KBAs	M49_Region

For each input file the code will produce automatically 5 output files (for all sites, terrestrial, marine, freshwater and mountain), with the following structure:

year	95CIlowCount	95CImidCount	95CIhiCount	95CIlowPercentage	95CImidPercentage	95CIhiPercentage	CI95lowPercentage_Area	CI95midPercentage_Area	CI95hiPercentage_Area	region	code	sset
1900	0	0	0	0	0	0	0	0	0	Afghanistan	country	Freshwater

```
# IBA randomizer 0.8
# JPW Scharlemann 14/04/2014 jorn.scharlemann@unep-wcmc.org
# Script to randomly allocate year of protection and percentage protected to IBAs with missing data
# Missing years and percentage protection are allocated based on the PAs with known information from that country, or if the country has two or fewer IBAs that are protected information will be allocated at random from all IBAs
# data preparation:
#
####################
# Prepare a datafile
# extract from Excel spreadsheet and save as .csv file
# ibaname (or IBA ID)
# country
# region
# perprotected = proportion protected of IBA (i.e. percentage protected / 100), 0 if missing information
# year = year of protection, 3000 if missing information
####################
setwd ("C:/Users/KBAs/input tables ")
workd <- " C:/Users/ KBAs/input tables "

inout=read.csv("in_out_files.csv")
tabmf=read.csv("classif_kbas_FINAL2017.csv")

subsets=c("all","marine","terrestrial","Freshwater", "mountain")   #change if only some subsets are needed
yrs = seq(1900, 2017, by = 1) # years to be analysed

#ssets2run=1: length(subsets)

for (w in 1:nrow(inout)){
#w=2

# Parameters that need to be set by user
j = 1 # number of randomisations needed

infile = as.character(inout$inputfile[w])  #"Input data for R global Region KBAs.csv"
glb=F
reg=T
if (inout$global[w]==1)
{
glb=T
reg=F
}
glb
reg

if (glb) {

# NOTE: Outfiles will be overwritten, including those for regions.
####################

ibadat1 = read.csv(infile, header = TRUE, sep = ",", quote="\"", dec=".", comment.char="", as.is = TRUE) # readin data

for (k in ssets2run)
{
#k=1
sset=subsets[k]
```

```
if (k==1) {
ibadat=ibadat1
outfile=paste(inout$outputfile1[w], ".csv", sep="")
}
if (k>1) {
kbasi=tabmf$SitRecID[tabmf[,which(names(tabmf)==sset)]==1]
ibadat=ibadat1[ibadat1$siteid%in%kbasi,]
outfile=paste(inout$outputfile1[w], "_", sset, ".csv", sep="")
}

attach (ibadat)
n = length(siteid)
ibaname = unique(siteid)
m = length(unique(siteid))
print(paste("Number of IBAs being analysed ", m," with ",n, " sites.", sep = ""))
poolryears=ibadat$year
####################
# define the custom functions
# correct sampling function, as standard R function not correct
resample <- function(x, size, ...)
  if(length(x) <= 1) { if(!missing(size) && size == 0) x[FALSE] else x
  } else sample(x, size, ...)
# random allocation of year protected for those with missing years, asign from all if none found in country
# where x = year, y = country and z = %protected
ranyear <-function(x, y, z) {
out <- as.vector(c())
for (i in 1:length(y))
{
if(x[i] != 3000) {
out = append(out, x[i], after = length(out))
} else {
if(length(as.numeric(x[y == y[i] & x != 3000 & z > 0])) > 1) {
out = append(out, resample(as.numeric(x[y == y[i] & x != 3000 & z > 0]), 1, replace = TRUE), after = length(out))
} else {
out = append(out, resample(as.numeric(x[y == y & x != 3000 & z > 0]), 1, replace = TRUE), after = length(out))
}}}
return(out)
}

# cummulative number of IBAs that are totally protected
# where x = protected and y = year and id = siteid
stattotp <-function(x, y, id) {
#yrs = seq(1900, 2016, by = 1)
out <- as.vector(c())
for (i in 1:length(yrs))
{
temp = tapply(x[y <= yrs[i]], id[y <= yrs[i]], sum)
out = append(out, length(temp[temp >= 98]), after = length(out))
}
return(out)
}


# cummulative average percentage of IBAs that are protected
# where x = protected and y = year and id = siteid
statavep <-function(x, y, id) {
#yrs = seq(1900, 2016, by = 1)
```

```
out <- as.vector(c())
for (i in 1:length(yrs))
{
temp = tapply(x[y <= yrs[i]], id[y <= yrs[i]], sum)
out = append(out, sum(temp/100), after = length(out))
}
return(out)
}


################################
# start data randomization
################################
mkdata <-function(x, y, z, j) {
# where X = %protected, y=country, z=year, j=number of iterations
# results are written into an array with row, column, dimension representing samples, years from 1900-2016,
and count of IBAs with 100% protection & mean area protected
# res[,,1] will give counts of IBAs with 100% protection, and res[,,2] will give the means
res <- matrix(data=NA, length(x),j)
rownames (res) <- siteid # write the IBA ID number to rowname
for (i in 1:j) {
ry = ranyear(z, y, x)
#st = stattotp (rp, ry)
#sm = statmeanp(rp, ry)
res[,i] <- ry
print(i)
}
return(res)
}
output = mkdata(perprotected, country, year, j)

# select subsets
################################
# calculate values
################################
calcvals = function(x,y) {
res <- matrix(data=NA, j, length(yrs))
for (i in 1:j) {
res[i,] <- stattotp(x, y[,i], rownames(y))
print(i)
print(j)
}
return(res)
}

calcvals2 = function(x,y) {
res <- matrix(data=NA, j, length(yrs))
for (i in 1:j) {
res[i,] <- statavep(x, y[,i], rownames(y))
print(i)
print(j)
}
return(res)
}

# calculate the results   ## % of PAs fully covered
alldata = calcvals(perprotected, output)
```

```
res <- matrix(data=NA, ncol=7, nrow=length(yrs), dimnames = list(c(seq(1,length(yrs), by =1)),c("year",
"95CIlowCount","95CImidCount","95CIhiCount", "95CIlowPercentage", "95CImidPercentage",
"95CIhiPercentage")))
res[,1] <- yrs
for (i in 1:length(yrs))
{
res[i,2] <-quantile(alldata[,i], probs = c(0.025))
res[i,3] <-quantile(alldata[,i], probs = c(0.5))
res[i,4] <-quantile(alldata[,i], probs = c(0.975))
print(i)
}
res[,5] <- res[,2]/m *100
res[,6] <- res[,3]/m *100
res[,7] <- res[,4]/m *100

res2=data.frame(res)
names(res2)=c("year", "95CIlowCount","95CImidCount","95CIhiCount", "95CIlowPercentage",
"95CImidPercentage", "95CIhiPercentage")

#write.table(res, file = outfile, append = FALSE, quote = TRUE, sep = ",", eol = "\n", na = "NA", dec = ".",
row.names = FALSE)


# calculate the results   ## % ok KBAs covered
alldata = calcvals2(perprotected, output)

res <- matrix(data=NA, ncol=7, nrow=length(yrs), dimnames = list(c(seq(1,length(yrs), by =1)),c("year",
"95CIlowCount","95CImidCount","95CIhiCount", "95CIlowPercentage_Area", "95CImidPercentage_Area",
"95CIhiPercentage_Area")))
res[,1] <- yrs
for (i in 1:length(yrs))
{
res[i,2] <-quantile(alldata[,i], probs = c(0.025))
res[i,3] <-quantile(alldata[,i], probs = c(0.5))
res[i,4] <-quantile(alldata[,i], probs = c(0.975))
print(i)
}
res[,5] <- res[,2]/m *100
res[,6] <- res[,3]/m *100
res[,7] <- res[,4]/m *100

res2$CI95lowPercentage_Area=res[,5]
res2$CI95midPercentage_Area=res[,6]
res2$CI95hiPercentage_Area=res[,7]
res2$code=inout$code[w]
res2$sset=sset
write.csv(res2, file = outfile, row.names=F)

detach(ibadat)

}
}
# end script
###############
```

```
### by region

if (reg) {

########################################
ibadat1 = read.csv(infile, header = TRUE, sep = ",", quote="\"", dec=".", comment.char="", as.is = TRUE) #
readin data

outputfile1=as.character(inout$outputfile1[w])
#if (glb) outputfile1="Output data for R Country KBAs"

#if(!"region"%in%names(ibadat1)) ibadat1$region=ibadat1$country

for (k in ssets2run)
{
#k=1
sset=subsets[k]
if (k==1) {
ibadat=ibadat1
outfile2=paste(outputfile1, ".csv", sep="")
}
if (k>1) {
kbasi=tabmf$SitRecID[tabmf[,which(names(tabmf)==sset)]==1]
ibadat=ibadat1[ibadat1$siteid%in%kbasi,]
outfile2=paste(outputfile1, "_", sset, ".csv", sep="")
}

head(ibadat)
regs=unique(ibadat$region)
regs
length(regs)


j=1
#j=1000
#j=1000

regres=data.frame()
for (z in 1:length(regs))
{
#z=1
#z=which(regs=="Angola")
z
ibadat2=ibadat[ibadat$region==regs[z],]
attach (ibadat2)

n = length(siteid)
ibaname = unique(siteid)
m = length(unique(siteid))
print(paste("Number of IBAs being analysed ", m," with ",n, " sites.", sep = ""))
####################
# define the custom functions
# correct sampling function, as standard R function not correct
resample <- function(x, size, ...)
 if(length(x) <= 1) { if(!missing(size) && size == 0) x[FALSE] else x
  } else sample(x, size, ...)
```

```
# random allocation of year protected for those with missing years, asign from all if none found in country
# where x = year, y = country and z = %protected
ranyear <-function(x, y, z) {
out <- as.vector(c())
for (i in 1:length(y))
{
if(x[i] != 3000) {
out = append(out, x[i], after = length(out))
} else {
if(length(as.numeric(x[y == y[i] & x != 3000 & z > 0])) > 1) {
out = append(out, resample(as.numeric(x[y == y[i] & x != 3000 & z > 0]), 1, replace = TRUE), after = length(out))
} else {
out = append(out, resample(poolryears, 1, replace = TRUE), after = length(out))
}}}
return(out)
}

# cummulative number of IBAs that are totally protected
# where x = protected and y = year and id = siteid
stattotp <-function(x, y, id) {
#yrs = seq(1900, 2016, by = 1)
out <- as.vector(c())
for (i in 1:length(yrs))
{
temp = tapply(x[y <= yrs[i]], id[y <= yrs[i]], sum)
out = append(out, length(temp[temp >= 98]), after = length(out))
}
return(out)
}


# cummulative average percentage of IBAs that are protected
# where x = protected and y = year and id = siteid
statavep <-function(x, y, id) {
#yrs = seq(1900, 2016, by = 1)
out <- as.vector(c())
for (i in 1:length(yrs))
{
temp = tapply(x[y <= yrs[i]], id[y <= yrs[i]], sum)
if (length(temp)>0) out = append(out, sum(temp/100), after = length(out))
if (length(temp)==0) out = append(out, 0, after = length(out))
}
return(out)
}


################################
# start data randomization
################################
mkdata <-function(x, y, z, j) {
# where X = %protected, y=country, z=year, j=number of iterations
# results are written into an array with row, column, dimension representing samples, years from 1900-2016,
and count of IBAs with 100% protection & mean area protected
# res[,,1] will give counts of IBAs with 100% protection, and res[,,2] will give the means
res <- matrix(data=NA, length(x),j)
rownames (res) <- siteid # write the IBA ID number to rowname
for (i in 1:j) {
```

```
ry = ranyear(z, y, x)
#st = stattotp (rp, ry)
#sm = statmeanp(rp, ry)
res[,i] <- ry
print(i)
}
return(res)
}
output = mkdata(perprotected, country, year, j)

# select subsets
##############################
# calculate values
##############################
calcvals = function(x,y) {
res <- matrix(data=NA, j, length(yrs))
for (i in 1:j) {
res[i,] <- stattotp(x, y[,i], rownames(y))
print(i)
print(j)
}
return(res)
}

calcvals2 = function(x,y) {
res <- matrix(data=NA, j, length(yrs))
for (i in 1:j) {
res[i,] <- statavep(x, y[,i], rownames(y))
print(i)
print(j)
}
return(res)
}


# calculate the results   % PAs totally covered
alldata = calcvals(perprotected, output)

res <- matrix(data=NA, ncol=7, nrow=length(yrs), dimnames = list(c(seq(1,length(yrs), by =1)),c("year",
"95CIlowCount","95CImidCount","95CIhiCount", "95CIlowPercentage", "95CImidPercentage",
"95CIhiPercentage")))
res[,1] <- yrs
for (i in 1:ncol(alldata))
{
res[i,2] <-quantile(alldata[,i], probs = c(0.025))
res[i,3] <-quantile(alldata[,i], probs = c(0.5))
res[i,4] <-quantile(alldata[,i], probs = c(0.975))
print(i)
}
res[,5] <- res[,2]/m *100
res[,6] <- res[,3]/m *100
res[,7] <- res[,4]/m *100

res2=data.frame(res)
names(res2)=c("year", "95CIlowCount","95CImidCount","95CIhiCount", "95CIlowPercentage",
"95CImidPercentage", "95CIhiPercentage")
```

```
# calculate the results   % coverage of KBAs
alldata = calcvals2(perprotected, output)

res <- matrix(data=NA, ncol=7, nrow=length(yrs), dimnames = list(c(seq(1,length(yrs), by =1)),c("year",
"95CIlowCount","95CImidCount","95CIhiCount", "95CIlowPercentage_Area", "95CImidPercentage_Area",
"95CIhiPercentage_Area")))
res[,1] <- yrs
for (i in 1:length(yrs))
{
res[i,2] <-quantile(alldata[,i], probs = c(0.025))
res[i,3] <-quantile(alldata[,i], probs = c(0.5))
res[i,4] <-quantile(alldata[,i], probs = c(0.975))
print(i)
}
res[,5] <- res[,2]/m *100
res[,6] <- res[,3]/m *100
res[,7] <- res[,4]/m *100

res2$CI95lowPercentage_Area=res[,5]
res2$CI95midPercentage_Area=res[,6]
res2$CI95hiPercentage_Area=res[,7]

res2$region=regs[z]
res2$code=inout$code[w]
res2$sset=sset

detach(ibadat2)

regres=rbind(regres, res2)
}

str(regres)
unique(regres$region)
length(unique(regres$region))
tail(regres)
write.table(regres, file = outfile2, append = FALSE, quote = TRUE, sep = ",", eol = "\n", na = "NA", dec = ".",
row.names = FALSE)
} # ends the loop of subsets (marine, freswhater, etc)
} # ends loop regional

} ## ends the loop of input files
```

Annex 3. R code to plot graphs showing trends over time in protected area coverage of KBAs using the output from running the R code in Annex 2.

This code requires input files (CSV format) with column headings:

year	95CIlowCount	95CImidCount	95CIhiCount	95CIlowPercentage	95CImidPercentage	95CIhiPercentage	CI95lowPercentage_Area	CI95midPercentage_Area	CI95hiPercentage_Area	region	code	sset
1900	0	0	0	0	0	0	0	0	0	Afghanistan	country	Freshwater

These are the Output files from the previous step and give the median (and upper and lower confidence intervals) for the number and percentage of KBAs that are completely covered by protected areas, and also the mean percentage of kba area covered by protected areas.

It outputs two pdf/PNG files for each country/region and type of kbas (all, marine, terrestrial, etc), one showing trends in protected area coverage of KBAs (filename in the format "PA coverage of KBAs") and one showing trends in mean area covered by PAs (filename in the format "Mean perc area protectedKBAs"). The three letters at the end are the ISO code for the country.

I also outputs csv files with the data used to produce the plots (e.g. "PA coverage mountain KBAs by country region.csv")

Important note: the files will be saved in subfolders within the *finfolder* , structured as follows:

finfolder/*type of kba*/*region classification* (ex: final graphs/terrestrial/country, or final graphs/marine/region). This structure should be created in advance to ensure that the files will be saved correctly. Note that the names should match the ones mentioned in the table above (type of kba=sset, region classification=code)

Text that is highlighted should be amended to give the appropriate folder names and file paths, number of randomisations (if this is to be adjusted) and final year to extrapolate the results to.

```
folder="C:/KBAs/input tables"
setwd(folder)
lu=function (x=x) length(unique(x))

cname="KBAs"                              ### or KBAs

fls=dir(pattern="Output data for R") ### important note; the code assumes that all files named "Output data for R" saved in the working folder will be analysed
#fls=dir(pattern="M49_Region")
fls
length(fls)

for (w in 1:length(fls)) #length(fls)
{
#w=11
fl1=fls[w]
fl1
tab=read.csv(fl1)    ### or KBAs

#tp=substr(fl1,nchar(fl1)-7, nchar(fl1))
```

```
#tp
#tpsset=data.frame(tp=c("KBAs.csv","ater.csv","rine.csv","rial.csv","tain.csv"),
sset=c("all","Freshwater","marine","terrestrial","mountain"))
#sset=tpsset$sset[tpsset$tp==tp]
sset=tab$sset[1]

desc=tab$code[1]
desc

finfolder=paste(folder, "final graphs/", sep="/")  ### "graphs_per_country/" should be an existing subfolder in
the folder mentioned above ; change if necessary


varis=c("percKBAcov","percKBATot")

for (y in 1:length(varis))
{
#y=2
vari=varis[y]

if (vari=="percKBATot"){
tab$mid=tab$X95CImidPercentage
tab$qn05=tab$X95CIlowPercentage
tab$qn95=tab$X95CIhiPercentage
nfile="PA coverage of "
lbgr="% sites completely covered by protected areas"
}

if (vari=="percKBAcov"){
tab$mid=tab$CI95lowPercentage_Area
tab$qn05=tab$CI95midPercentage_Area
tab$qn95=tab$CI95hiPercentage_Area
nfile="Mean perc area protected"
lbgr="mean % area covered by protected areas"
}

cinz=grey(0.7)
min(tab$qn95-tab$qn05)
max(tab$qn95-tab$qn05)

head(tab)
str(tab)

cts=unique(tab$region[tab$region!="0"])
lu(cts)

#### one png per country

if (length(cts)==0)
{
ct1="global"
tabi=tab
head(tabi)
basy=0
topy=100
if (max(tabi$qn95)<20) topy=50
```

```
file1name=paste(finfolder, sset,"/", desc, "/", nfile, cname, "_", ct1, "_", sset, ".png", sep="")
file1name
png(file=file1name, width = 12, height = 10, units="in", res=600)

par(las=1,cex.axis=1.2, cex.lab=1.3, cex.main=1.5, mar=c(5,5,4,2), mgp=c(3.5, 1, 0))

with(tabi,plot(year,mid, ylim=c(basy,topy), xlim=c(min(year), max(year)+1), type="n", xlab="Year", ylab=lbgr,
yaxt="n"))
axis(2, seq(basy, topy, by=(topy-basy)/5))

with(tabi,polygon(c(year,rev(year)),c(qn05,rev(qn95)), col=cinz, border=cinz))
with(tabi,lines(year,mid, col=2, lwd=2))
dev.off()
}

if (length(cts)>0)
{
for (i in 1:length(cts))
{
#i=1
#i=which(cts=="CAF")
ct1=as.character(cts[i])
ct1
tabi=tab[tab$region==ct1,]
head(tabi)
basy=0
topy=100
if (max(tabi$qn95)<20) topy=50

file1name=paste(finfolder, sset,"/", desc, "/", nfile, cname, "_", ct1, "_", sset, ".png", sep="")
file1name
png(file=file1name, width = 12, height = 10, units="in", res=600)
par(las=1,cex.axis=1.2, cex.lab=1.3, cex.main=1.5, mar=c(5,5,4,2), mgp=c(3.5, 1, 0))

with(tabi,plot(year,mid, ylim=c(basy,topy), xlim=c(min(year), max(year)+1), type="n", xlab="Year", ylab=lbgr,
yaxt="n"))
axis(2, seq(basy, topy, by=(topy-basy)/5))

with(tabi,polygon(c(year,rev(year)),c(qn05,rev(qn95)), col=cinz, border=cinz))
with(tabi,lines(year,mid, col=2, lwd=2))
dev.off()
}
} # ends loop one png per country




###### single pdf with all countries
if (length(cts)>1)
{
pdfallname=paste(finfolder, sset,"/", desc, "/", nfile, cname, " _all plots ", desc, "_", sset, ".pdf", sep="")
pdfallname

pdf(file=pdfallname, width = 12, height = 10)
par(las=1,cex.axis=1.2, cex.lab=1.3, cex.main=1.5, mar=c(5,5,4,2), mgp=c(3.5, 1, 0))
```

```
for (i in 1:length(cts))
{
#i=1
#i=which(cts=="CAF")
ct1=as.character(cts[i])
ct1
tabi=tab[tab$region==ct1,]
head(tabi)

basy=0
topy=100
if (max(tabi$qn95)<20) topy=50

par(las=1,cex.axis=1.2, cex.lab=1.3, cex.main=1.5, mar=c(5,5,4,2), mgp=c(3.5, 1, 0))

with(tabi,plot(year,mid, ylim=c(basy,topy), xlim=c(min(year), max(year)+1), type="n", xlab="Year", ylab=lbgr,
yaxt="n", main=ct1))
axis(2, seq(basy, topy, by=(topy-basy)/5))
with(tabi,polygon(c(year,rev(year)),c(qn05,rev(qn95)), col=cinz, border=cinz))
with(tabi,lines(year,mid, col=2, lwd=2))
}
dev.off()
}

### one pdf per country
if (length(cts)==0)
{
ct1="global"
tabi=tab
head(tabi)
basy=0
topy=100
if (max(tabi$qn95)<20) topy=50

pdf1name=paste(finfolder, sset,"/", desc, "/", nfile, cname, "_", ct1, "_", sset,".pdf", sep="")
pdf1name
pdf(file=pdf1name, width = 12, height = 10)
par(las=1,cex.axis=1.2, cex.lab=1.3, cex.main=1.5, mar=c(5,5,4,2), mgp=c(3.5, 1, 0))

with(tabi,plot(year,mid, ylim=c(basy,topy), xlim=c(min(year), max(year)+1), type="n", xlab="Year", ylab=lbgr,
yaxt="n"))
axis(2, seq(basy, topy, by=(topy-basy)/5))

with(tabi,polygon(c(year,rev(year)),c(qn05,rev(qn95)), col=cinz, border=cinz))
with(tabi,lines(year,mid, col=2, lwd=2))
dev.off()

}

if (length(cts)>0)
{
for (i in 1:length(cts))
{
#i=1
#i=which(cts=="CAF")
ct1=as.character(cts[i])
ct1
```

```
tabi=tab[tab$region==ct1,]
head(tabi)
basy=0
topy=100
if (max(tabi$qn95)<20) topy=50

pdf1name=paste(finfolder, sset,"/", desc, "/", nfile, cname, "_", ct1, "_", sset, ".pdf", sep="")
pdf1name
pdf(file=pdf1name, width = 12, height = 10)
par(las=1,cex.axis=1.2, cex.lab=1.3, cex.main=1.5, mar=c(5,5,4,2), mgp=c(3.5, 1, 0))

with(tabi,plot(year,mid, ylim=c(basy,topy), xlim=c(min(year), max(year)+1), type="n", xlab="Year", ylab=lbgr,
yaxt="n"))
axis(2, seq(basy, topy, by=(topy-basy)/5))

with(tabi,polygon(c(year,rev(year)),c(qn05,rev(qn95)), col=cinz, border=cinz))
with(tabi,lines(year,mid, col=2, lwd=2))
dev.off()
}
} # ends loop one pdf per country

} # ends loop varis

} # ends loop input files



## create single file for outputs


fls=dir(pattern="Output data for R")
fls
length(fls)

fin1=data.frame()
for (w in 1:length(fls)) #length(fls)
{
#w=7
fl1=fls[w]
fl1
tab=read.csv(fl1)    ### or KBAs
head(tab)
nrow(tab)
unique(tab$region)
if (!"region"%in%names(tab))  tab$region="Global"
head(tab)
tab=tab[tab$region!="0",]
nrow(tab)
unique(tab$region)
tab2=tab[c("year","X95CIlowCount","X95CImidCount","X95CIhiCount","X95CIlowPercentage","X95CImidPerce
ntage","X95CIhiPercentage","CI95lowPercentage_Area","CI95midPercentage_Area","CI95hiPercentage_Area",
"region","sset")]
tab2$file1=fl1
fin1=rbind(fin1,tab2)
}
length(unique(fin1$file1)) # should be the number of output files to merge
```

```
subsets=unique(fin1$sset)

for (g in 1:length(subsets))
{
#g=1
sset=subsets[g]

nffile=paste("PA coverage ", sset, " KBAs by country  region.csv")
nffile
fin2=fin1[fin1$sset==sset,]
nrow(fin2)
unique(fin2$sset)
fin2=fin2[,1:11]
names(fin2)=c("year","Count of KBAs completely covered by PAs (lower CI)","Count of KBAs completely
covered by PAs (estimate)","Count of KBAs completely covered by PAs (upper CI)","% KBAs completely covered
by PAs (lower CI)","% KBAs completely covered by PAs (value)","% KBAs completely covered by PAs (upper
CI)","Mean % area of each KBA covered by PAs (lower CI)","Mean % area of each KBA covered by PAs
(value)","Mean % area of each KBA covered by PAs (upper CI)","Global/region/country")
write.csv(fin2, nffile, row.names=F)
}
```