T-LoCoH Script of one individual – Cheetah 1

```
#Import Cheetah1 data and check
class(Cheetah1)
head(Cheetah1)
plot(Cheetah1[,c("long","lat")],pch=20)
#need to convert to UTM coordinates, 34S
#make sure sp and rgdal are selected packages
Cheetah1.sp.latlong<-SpatialPoints(Cheetah1[,c("long","lat")],proj4string=CRS("+proj=longlat
+ellps=WGS84"))
Cheetah1.sp.utm<-spTransform(Cheetah1.sp.latlong, CRS("+proj=utm +south +zone=34
+ellps=WGS84"))
#create a matrix
Cheetah1.mat.utm<-coordinates(Cheetah1.sp.utm)
head(Cheetah1.mat.utm)
#Change column names
colnames(Cheetah1.mat.utm)<-c("x","y")
head(Cheetah1.mat.utm)
#Adjust time zone
class(Cheetah1$TimeUTC)
head(as.character(Cheetah1$TimeUTC))
Cheetah1.gmt<-as.POSIXct(Cheetah1$TimeUTC, tz="UTC")
Cheetah1.gmt[1:3]
local.tz<-"Africa/Johannesburg"
Cheetah1.localtime<-as.POSIXct(format(Cheetah1.gmt, tz=local.tz), tz=local.tz)
Cheetah1.localtime[1:3]
#the above section converts data to local time

#Create locoh object with no other variables
Cheetah1.lxy<-xyt.lxy(xy=Cheetah1.mat.utm, dt=Cheetah1.gmt, id="ID", dup.dt.check=T)
#duplicate time stamps will be detected.
#to see how many points left
summary(Cheetah1.lxy)
#look at graph
plot(Cheetah1.lxy)
#to look at distributions by date, step length, sampling interval
hist(Cheetah1.lxy)
lxy.plot.freq(Cheetah1.lxy, deltat.by.date=T)
#to check if bursts of data points, plot cumulative percentage of sampling intervals
lxy.plot.freq(Cheetah1.lxy, cp=T)
# Want to thin data set to one point per hour
lxy.thin.

# To include time as a factor in the analysis need to select an s value
#Two methods can help with this:
##1. Pick s where 40-80% of hulls are time selected
Cheetah1.lxy<-lxy.ptsh.add(Cheetah1.lxy)
#Read graph to pick value of s where proportion of time between 04.-0.8.
#2. Pick s based on set time value, based on research question. If not sure can plot natural
frequencies in data, plotting distance to centroid
lxy.plot.pt2ctr(Cheetah1.lxy)
```

```
#s is also dependent on map units, so plot s so time and space equalise over range of values
lxy.plot.sfinder(Cheetah1.lxy)
#can replot this to look at specific time points of interest. Time is in seconds
lxy.plot.sfinder(Cheetah1.lxy, delta.t=3600*c(12, 24, 48,96))
#using this plot 24hrs shows a fairly balanced time point to select s. In this case s=0.05

#Next task is to identify nearest neighbours, k, r, a methods
#start with K method
#unsure yet how many nearest neighbours give best space use model, so start with 25
Cheetah1.lxy<-lxy.nn.add(Cheetah1.lxy, s=0.05, k=25)
#if want to try various levels of s
#Cheetah1.lxy<-lxy.nn.add(Cheetah1.lxy, s=c(0.005,0.5), k=10)
#to save this
lxy.save(Cheetah1.lxy, dir=".")

#Creating Hullsets. Already identified 25nn for each point. Now try range of k values
Cheetah1.lhs<-lxy.lhs(Cheetah1.lxy,k=3*2:8, s=0.05)
summary(Cheetah1.lhs, compact=T)

#Create isopleths for each k value
Cheetah1.lhs<-lhs.iso.add(Cheetah1.lhs)
lxy.save(Cheetah1.lxy, dir=".")

#May need gpclib to show graphs
##RTools temporary install
pathRtools <- paste(c("c:\\Rtools\\bin",
                      "c:\\Rtools\\MinGW\\bin",
                      "c:\\MiKTeX\\miktex\\bin",
                      "c:\\R\\bin\\i386",
                      "c:\\windows",
                      "c:\\windows\\system32"), collapse=";")
Sys.setenv(PATH=paste(pathRtools,Sys.getenv("PATH"),sep=";"))

## get gpclib from source
if (!require(gpclib)) install.packages("gpclib", type="source")

#Creating isopleths now that GpClib is added.

#plot the isopleths for each value of k
plot(Cheetah1.lhs, iso=T, record=T, ufipt=F)

#selection of one k value with original points
plot(Cheetah1.lhs, iso=T, k=24, allpts=T, cex.allpts=0.1, col.allpts="gray50", ufipt=F)
#to help decide k can also look at isopleth area curves and edge:area curves
lhs.plot.isoarea(Cheetah1.lhs)
#need to look for jumps in area between k values. In this for 95% big jump between 18 and 21 so
want lower k value
#be good to look at hullsets for k between 15 and 21
lhs.plot.isoear(Cheetah1.lhs)
#if you decide on value of k can just use those values for rest of analysis
#k=15 is good choice here as at core area say 30% k=15 is good
```

```
Cheetah1.lhs.k15<-lhs.select(Cheetah1.lhs, k=15)

#Now trying a method
#use auto function supplying p and n. As had previously chosen k=15 can decide p as %points having
15 neighbours. Try p=98%
Cheetah1.lxy<-lxy.nn.add(Cheetah1.lxy, s=0.05, a=auto.a(nnn=15, ptp=0.98))
#auto a is nearly 40000 but may need to increase k value to get enough points to reach 98%
#Will try to identify enough neighbours for a=40000
Cheetah1.lxy<-lxy.nn.add(Cheetah1.lxy, s=0.05, a=40000)
# create new hullsets object with different a values
Cheetah1.lhs.amixed<-lxy.lhs(Cheetah1.lxy, s=0.05, a=30:40*1000, iso.add=T)
# plot again isopleth area
lhs.plot.isoarea(Cheetah1.lhs.amixed)
# plot edge:area curves
lhs.plot.isoear(Cheetah1.lhs.amixed)
#selection a value of 40000
Cheetah1.lhs.a<-lxy.lhs(Cheetah1.lxy, s=0.05, a=40000, iso.add=T)

#Now looking at time use metrics using the a method
#Choosing time value of 12hrs to separate events
Cheetah1.lhs.a<-lhs.visit.add(Cheetah1.lhs.a, ivg=3600*12)
#For visitation
hist(Cheetah1.lhs.a, metric="nsv")
plot(Cheetah1.lhs.a, hpp=T, hpp.classify="nsv", ivg=3600*12, col.ramp="rainbow")
Cheetah1.aoi<aoi()
plot(Cheetah1.lhs.a, hpp=T, hpp.classify="nsv", ivg=3600*12, col.ramp="rainbow",
aoi=Cheetah1.aoi)
#This shows areas that are highly visited
#For Duration
hist(Cheetah1.lhs.a, metric="mnlv", ivg=3600*12)
plot(Cheetah1.lhs.a, hpp=T, hpp.classify="mnlv", col.ramp="rainbow")
#long duration areas are different to highly visited areas
# Can plot visitation against duration
hsp<-lhs.plot.scatter(Cheetah1.lhs.a, x="nsv", y="mnlv", col="spiral", bg="black")
plot(Cheetah1.lhs.a, hpp=T, hsp=hsp, hpp.classify="hsp")
lhs.save(Cheetah1.lhs.a, dir=".")

#Can re-do for different time values as well as other metrics check hm.expr()

#Then export as csv files or shape files as required
```

Two-stage approach for analysing data.
Described for one variable, distance to marking trees.

```
#Start with duration at marking trees
#First going to model individuals separately
mtdata<-groupedData(durmt~markingtree|idmt, data=allmarkingtree, FUN=mean,
labels=list(x="distance to mt", y="duration"))
head(mtdata)
# Model each individual separately
mtlist<-lmList(durmt~markingtree|idmt, data=mtdata)
plot(resid(mtlist))
# Not that great try logging data
mtlist<-lmList(log(durmt)~markingtree|idmt, data=mtdata)
plot(resid(mtlist))
#Looks better, will use logged data

#Second will model data using meta-analysis
#extract coefficients, observed outcomes
mtdurcoef <- lapply(mtlist, coef)
#extract var-covar matrix
mtdurvcov <- lapply(mtlist, vcov)
#creation of dummy variables
mtdestm <- rep(c("intercept","slope"), length(mtdurcoef))
mtdsubj <- rep(names(mtdurcoef), each=2)
#create vector
mtdurcoef <- unlist(mtdurcoef)
mtdurvcov <- bldiag(mtdurvcov)
#multivariate meta-analysis with model coefs
mtdres2 <- rma.mv(mtdurcoef ~ mtdestm - 1, mtdurvcov, random = ~ mtdestm | mtdsubj,
struct="UN", method="ML")
summary(mtdres2)
#look at effect of Q
confint(mtdres2, tau2=1)
#Check model assumptions
plot(rstandard.mv(mtdres2$z))
#test of outliers rstand>3 and hatvalue>2 times average hat
rsmtd<-rstandard(mtdres2)
htmtd<-hatvalues(mtdres2)/mean(hatvalues(mtdres2))
plot(htmtd, rsmtd$z, ylim = c(-4.0,4))
#assessing output
predict(mtdres2, level=95, transf=exp, tau2=1, newmods=c(1,0), addx=T)
# repeat for required values eg.e newmods=c(1,6000)

## Now modelling visitation rates at marking tree
#As count data will use poisson distribution but first to check for overdispersion
C1<-glm(visitmt~markingtree, data=allmarkingtree$idmt$==C1, family=poisson)
dispersiontest(C1, trafo=1)
#Repeat for all individuals. In this case all overdispersed, so will use quasipoisson family

mtdata2<-groupedData(visitmt~markingtree|idmt, data=allmarkingtree, FUN=mean,
labels=list(x="distance to mt", y="visitation"))
```

```
head(mtdata2)
mtlist<-lmList(visitmt~markingtree|idmt, data=mtdata2, family=quasipoisson)
plot(resid(mtlist)

##Second will model data using meta-analysis
#extract coefficients
mtvcoef <- lapply(mtlist, coef)
#extract var-covar matrix
mtvvcov <- lapply(mtlist, vcov)
#creation of dummy variables
mtvestm <- rep(c("intercept","slope"), length(mtvcoef))
mtvsubj <- rep(names(mtvcoef), each=2)
#create vector
mtvcoef <- unlist(mtvcoef)
mtvvcov <- bldiag(mtvvcov)
#multivariate meta-analysis with model coefs
mtvres2 <- rma.mv(mtvcoef ~ mtvestm - 1, mtvvcov, random = ~ mtvestm | mtvsubj, struct="UN",
method="ML")
summary(mtvres2)
#Look at effect of Q
confint(mtvres2)

#Check model assumptions
plot(rstandard(mtvres$z)
#test of outliers rstand>3 and hatvalue>2 times average hat
rsmtv<-rstandard(mtvres2)
htmtv<-hatvalues(mtvres2)/mean(hatvalues(mtvres2))
plot(htmtv, rsmtv$z, ylim = c(-4.0,4))

#assessing output
predict(mtvres2, level=95, transf=exp, tau2=1, newmods=c(1,0), addx=T)
#Repeat for required values
```

#Example script for including all data for duration analysis

```
alldata<-groupedData(log(duration)~markingtree+homestead+centre|ID, data=allrvcdata,
FUN=mean, labels=list(x="distance", y="duration"))
head(alldata)
alldata2<-na.omit(alldata)
alllist<-lmList(log(duration)~markingtree+homestead+centre|ID, data=alldata2)

#extract coefficients, observed outcomes
alldurcoef <- lapply(alllist, coef)
#extract var-covar matrix
alldurvcov <- lapply(alllist, vcov)
#creation of dummy variables
alldestm <- rep.int(c("intercept","mtslope","hsslope","cnslope"), times=4)
alldestm
alldsubj <- rep(names(alldurcoef), each=1)
#create vector
alldurcoef <- unlist(alldurcoef)
```

```
alldurvcov <- bldiag(alldurvcov)
#multivariate meta-analysis with model coefs
alldres2 <- rma.mv(alldurcoef ~ alldestm - 1, alldurvcov, random = ~ alldestm | alldsubj, struct="UN",
method="ML")
summary(alldres2)
#look at effect of Q
confint(alldres2, tau2=1)
#check model assumptions
plot(residuals(alldres2))
plot(fitted(alldres2)~resid(alldres2))
```