source_file.R

alexb

Thu Nov 15 15:11:05 2018

```
# Sourcing libraries, data, and functions

##
## Source Libraries
##

libr=c("stats","coda","spBayes","geoR","fields","maptools","graticule",
       "spatstat","raster","leaps","MPV","MASS")
options(warn=-1)
suppressPackageStartupMessages(lapply(libr, require,character.only = TRUE))
```

```
## [[1]]
## [1] TRUE
##
## [[2]]
## [1] TRUE
##
## [[3]]
## [1] TRUE
##
## [[4]]
## [1] TRUE
##
## [[5]]
## [1] TRUE
##
## [[6]]
## [1] TRUE
##
## [[7]]
## [1] TRUE
##
## [[8]]
## [1] TRUE
##
## [[9]]
## [1] TRUE
##
## [[10]]
## [1] TRUE
##
## [[11]]
## [1] TRUE
##
## [[12]]
## [1] TRUE
```

```
##
## Source files
##

### Exercise 1 (precipitation data Colorado)
jan=read.table("http://cires1.colorado.edu/~aslater/CVEN_6833/colo_monthly_precip_01.dat")
colnames(jan)<-c("Lat","Long","Elev","Prec")
jan<-jan[c("Prec",colnames(jan)[!colnames(jan) %in% "Prec"])] #Predictor first
jan$Prec<-ifelse(jan$Prec<=0,NA,jan$Prec) #Eliminate zero-neg values
jan<-jan[complete.cases(jan),] # Clean data

dem=read.table("http://cires1.colorado.edu/~aslater/CVEN_6833/colo_dem.dat")
colnames(dem)<-c("Lat","Long","Elev")

### Exercise 2 (daily precipitation)
daily=read.table("http://cires1.colorado.edu/~aslater/CVEN_6833/colo_pcp_daily_1997_01_11.dat")
colnames(daily)<-c("Lat","Long","Elev","Prec")          # temp. Headings
daily<-daily[c("Prec",colnames(daily)[!colnames(daily) %in% "Prec"])] #Predictor first
daily$Prec<-ifelse(daily$Prec>0,1,0) #Binary data
daily$Elev<-ifelse(daily$Elev<=0,NA,daily$Elev) #Remove points without Elev
daily<-daily[complete.cases(daily), ] # Clean data
daily<-daily[!rownames(daily) %in% dup.coords(daily[,2:3])[2,],] # Remove duplicates
### par(mar=c(1,1,1,1)) # Set margins to minimum
### default was par(mar=c(5.1,4.1,4.1,2.1))

##
## Source functions
##

### Spatial plotting
colo_pol<-map('county', 'colorado', fill = TRUE, col = palette(),plot=FALSE) # Colorado map
IDs <- sapply(strsplit(colo_pol$names, ":"), function(x) x[1]) # County names

colo_frame<-map('state', region = c('colorado', 'utah', 'wyoming','kansas','nebraska','oklahoma',
                                    'new mexico','arizona'),xlim=c(-109.3,-101.7),ylim=c(36.7,41.2),
                plot=FALSE,fill = TRUE,lforce = "e") # Sorrounding states
ID2s <- sapply(strsplit(colo_frame$names, ":"), function(x) x[1]) # Names

colo_pol <- map2SpatialPolygons(colo_pol, IDs=IDs, proj4string=CRS("+proj=longlat +datum=WGS84"))
colo_frame<- map2SpatialPolygons(colo_frame,IDs=ID2s,proj4string=CRS("+proj=longlat +datum=WGS84"))

cit <- data.frame(Name=c('Denver','Ft Collins','Steamboat Sp','Winter Park','Colo Sprgs', # Col cities
                         'Grand Jct','Aspen','Pueblo','Lamar','Alamosa ','Trinidad'),
                  Lat=c(39.739235,40.58897,40.48549,39.89106,38.833881,39.065369,39.19067,
                        38.247685,38.081406,37.460484,37.167772),
                  Long=c(-104.99025,-105.082458,-106.83356,-105.76072,-104.821365,-108.569527,
                         -106.819199,-104.605081,-102.614833,-105.867995,-104.494041))
grat_sp <- graticule(lons = seq(-109,-102,by=1), lats = seq(37,41,by=1),
                     proj = "+init=epsg:3501")                                  #add grid with lat and lon

#### Transform data to PointSpatial Data
getSpatialDataFrame<-function(mydata){
  projdata<-SpatialPoints(data.frame(mydata$Long,mydata$Lat),
```

```
                          proj4string=CRS("+proj=longlat +datum=WGS84"))
  return(SpatialPointsDataFrame(projdata,data = mydata))
}
idata<-list(dem,jan,cit)
fdata<-list("dem_sp","jan_sp","cit_sp")
z<-lapply(idata,getSpatialDataFrame)
for(i in 1:length(z)){assign(as.character(fdata[i]),z[[i]])}

#### Adopt NAD83 CRS
NAD83<-function(mapdata){
  return(spTransform(mapdata,CRS("+init=epsg:3501")))
}
idata<-list(dem_sp,jan_sp,cit_sp,colo_pol,colo_frame)
fdata<-list("dem_sp","jan_sp","cit_sp",'colo_pol','colo_frame')
z<-lapply(idata,NAD83)
for(i in 1:length(z)){assign(as.character(fdata[i]),z[[i]])}

#### Coerce attibutes to {spatstat} from point data to point pattern (shapefile)
col_owin <- as.owin.SpatialPolygons(colo_pol) #smaller window
colf_owin<- as.owin.SpatialPolygons(colo_frame) #Colorado with other states
dem_ppp<-as.ppp(data.frame(dem_sp@coords,dem_sp@data$Elev), W=colf_owin)
jan_ppp<-as.ppp(data.frame(jan_sp@coords,jan_sp@data$Prec), W=colf_owin)
cit_ppp<-as.ppp(data.frame(cit_sp@coords,cit_sp@data$Name), W=colf_owin)


#### plot function to include a new ppp with the Smooth.ppp function
generateplot<-function(ppp){
  plot(Smooth.ppp(ppp,eps=1000),col=rainbow(16),main = NULL)
  raster::lines(colo_pol,col="lightgray")
  points(X_sp,col="red",pch=".")
  points(cit_sp,pch=21)
  text(x = cit_sp@coords[,1], y = cit_sp@coords[,2], cit_sp@data$Name, pos = 4)
  plot(grat_sp,lty=2,add=TRUE)
  labs <- graticule_labels(lons = seq(-109,-102,by=1), lats = seq(37,41,by=1),
                           xline = -109, yline = 36.9,  proj = "+init=epsg:3501")
  text(labs, lab = parse(text = labs$lab), pos = c(2.7, 1.5)[labs$islon + 1], adj = 1.2)
}

### GLM fit (HW#1)
GLM_fit = function(data, family) {

    datanames<-colnames(data)          # temp. Headings
  if (family == "Gamma") {
    links = c("log", "inverse", "identity")
  } else if (family == "gaussian"){
    links = c("identity", "log", "inverse")
  } else if (family == "binomial"){
    links = c("logit", "probit", "cauchit","log","cloglog")
  } else {stop()
    # Possible families: 'Gamma' and 'gaussian'")
  }
  N = dim(data)[1]
```

```
#Adding interactions, by 2s and 3s
inter2<-combn(datanames[!datanames %in% "Prec"],2)
inter3<-combn(datanames[!datanames %in% "Prec"],3)

for(i in 1){
  for(j in 1:3){
    data[paste(inter2[i,j],inter2[i+1,j],sep="")]<-
      eval(parse(text=paste("data$",inter2[i,j],sep="")))*
      eval(parse(text=paste("data$",inter2[i+1,j],sep="")))
  }
}

data["All"]<-eval(parse(text=paste("data$",inter3[1,1],sep="")))*
  eval(parse(text=paste("data$",inter3[2,1],sep="")))*
  eval(parse(text=paste("data$",inter3[3,1],sep="")))

Y<-data$Prec
X<-data[colnames(data)[!colnames(data) %in% "Prec"]]
combs = leaps(X,Y, nbest=40,method="adjr2")      #  Get upto 40 combinations
# number of predictors
combos = combs$which
ncombos = length(combos[,1])

glm_xpress<-glm_xmse<-glm_aic <- rep(NA,length(links))
xpress_index<-xmse_index<-aic_index<-rep(NA,length(links))
for(j in 1:length(links)) {
  xpress<-xmse<-aic<- rep(NA,ncombos)
  for(i in 1:ncombos) {
    xx = X[,combos[i,]]
    xx=as.data.frame(xx)
    if (family == "Gamma") {
      zz=try(glm(Y ~ ., data=xx, family = Gamma(link=links[j]), maxit=500),silent=TRUE)
      #        print(c("Gamma",i,j)) #debugging
    } else if (family == "gaussian"){
      zz=try(glm(Y ~ ., data=xx, family = gaussian(link=links[j]), maxit=500),silent=TRUE)
      #        print(c("binomial",i,j)) #debugging
    } else if (family == "binomial"){
      zz=try(glm(Y ~ ., data=xx, family = binomial(link=links[j]), maxit=500),silent=TRUE)
      #        print(c("binomial",i,j)) #debugging
    }
    if (class(zz)[1]=="try-error"){
    } else{
      xpress[i]=PRESS(zz)
      xmse[i] = sum((zz$res)^2) / (N - length(zz$coef))
      aux=try(stepAIC(zz, scope=list(upper = ~., lower = ~1), trace=FALSE)$aic,
              silent=TRUE)
      aic[i]=ifelse(class(aux)=="try-error",NA,aux)
    }
  }
  # Test using AIC objective function
  if (class(zz)[1]=="try-error"){
  } else{
    glm_xpress[j]= min(xpress,na.rm=TRUE)
```

```
      glm_xmse[j]= min(xmse,na.rm=TRUE)
      glm_aic[j] = min(aic,na.rm=TRUE)
      xpress_index[j]=which.min(xpress)
      xmse_index[j]= which.min(xmse)
      aic_index[j] = which.min(aic)
    }
  }
  best_df = data.frame(rbind(glm_xpress,glm_xmse,glm_aic,
                             xpress_index,xmse_index,aic_index))
  colnames(best_df) = links[1:length(links)]

  print("Results of AIC for bestfit GLM")
  show(best_df)
  i=which.min(glm_aic)
  print(sprintf("Choosing the GLM which minimizes AIC for %s family: %s link function.",
                family, links[i]))

  if (family == "Gamma") {
    bestmod = try(glm(Y ~ ., data = X[,combos[aic_index[i],]], family = Gamma(link=links[i])))
  } else if (family == "gaussian") {
    bestmod = glm(Y ~ ., X[,combos[aic_index[i],]], family = gaussian(link=links[i]))
  } else if (family == "binomial") {
    bestmod = glm(Y ~ ., X[,combos[aic_index[i],]], family = binomial(link=links[i]))
  } else {
    print("Error!")
  }
  print(bestmod)
  return(bestmod)
}
```