CRP V4-1811 urejanje posnetkov in podatkov za druge vire satelitskih posnetkov

Code

Uroš Žibrat

Če želimo celoten postopek predobdelave posnetkov izvesti v R. Posnetke smo že prevzeli in jih tukaj samo obdelamo.

Primer je narejen za posnetke PlanetScope, ki so bili prej prevzeti z -curl. Za celo območje je en posnetek na dan. Začetek postopka je tukaj nekoliko drugačen, ker najprej naložimo posnetke in naredimo presek z zbirnimi vlogami. Nato izvlečemo potrebne podatke in jih dalje obdelamo po enakem postopku, kot za Sentinel posnetke, prevzete in predobdelane v eo-learn.

Hide

```
require(maptools)
require(raster)
require(rgdal)
require(data.table)
require(zoo)
require(imputeTS)
require(forcats)
require(plyr)
require(tidyverse)

#funkcija za mediano
mediana.filter <- function(x, n = 2) { #funkcija za filtriranje podatkov po SD okoli mediane
  x <- x - median(x)
  s <- n * mad(x)
  (x <= s) & (x > -s)  #logicnima vrednost T/F glede na to, ce je podatek znotraj pasu mediana +- n*mad
}

# funkcija za drseče povprečje
mav <- function(x, n= 5) {
  stats::filter(x, rep(1/n,n), sides = 2)
}

setwd("D:\\ESA_Planetscope\\Kras_2017\\utm\\samo_tif")
kras <- readOGR(dsn = "D:\\ESA_Planetscope\\Kras_2017\\utm\\kras_subvencije2017_UTM.shp")
datoteke <- list.files(pattern = ".tif")
dat.udm <- list.files(path = "D:\\ESA_Planetscope\\Kras_2017_Surface-reflectance\\files\\merged", pattern =
"udm")
datoteke <- datoteke[order(as.Date(substr(datoteke, 6,15), format = "%d-%m-%Y"))]
dat.udm <- dat.udm[order(as.Date(substr(datoteke, 6,15), format = "%d-%m-%Y"))]
dat.udm <- paste("D:\\ESA_Planetscope\\Kras_2017_Surface-reflectance\\files\\merged\\", dat.udm, sep = "")
```

Odpremo multispektralne posnetke, dodamo masko oblakov, naredimo presek, pretvorimo raster v točke, izračunamo in filtriramo rastrske celice po mediani in izračunamo svetlobni odboj.

Hide

```
podatki <- list()
for (i in 1:length(datoteke)) {
  print(i)
  slika <- stack(datoteke[i])
  udm <- stack(dat.udm[i])
  maska <- mask(slika, kras)
  maska <- mask(maska, udm)
  tocke <- rasterToPoints(maska, spatial = T)
  presek <- over(kras, tocke, returnList = T)

  presek.mean <- lapply(presek, function(x) {
    pod <- x
    if (nrow(pod) == 0 | nrow(pod) == 1) {
      pod <- as.numeric(as.vector(pod))
      return(pod)
    } else if (nrow(pod < 5)) {
      pod <- apply(pod,2,mean)
      return(pod)
    } else {
      pod.m <- apply(pod, 2, function(x) mediana.filter(x,2))
      pod <- pod[apply(pod.m,1,all),]
      pod <- apply(pod,2,mean)
      return(pod)
    }
  })

  for (k in 1:length(presek.mean)) {
    if (k == 1) {
      dat <- presek.mean[[k]]
    } else {
      dat <- rbind(dat, presek.mean[[k]])
    }
  }

  dat <- data.frame(dat)
  dat$POLJINA_ID <- kras$POLJINA_ID
  dat$SIFRA_KMRS <- kras$SIFRA_KMRS
  dat <- data.frame(dat[complete.cases(dat),])
  dat[,1:4] <- dat[,1:4] * 0.1
  names(dat)[1:4] <- c("Blue","Green","Red","NIR")
  dat <- dat[,c(6,5,1:4)]
  podatki[[i]] <- dat
}

podatki <- lapply(podatki, function(x) {
  x[,3:6] <- x[,3:6] / 1000
  return(x)
})
```

Izračunamo indekse

Hide

```
dnevi <- as.Date(substr(datoteke, 6,15), format = "%d-%m-%Y")

indeksi <- lapply(podatki, function(x) {
  pod <- subset(x, subset = SIFRA_KMRS %in% c(005,006,801,809,110,206, 207, 208, 219, 220, 222, 223, 117, 11
8, 200, 201, 202, 203, 204))
  pod$Rededge705 <- (pod$Red + pod$NIR) /2.056
  pod$Rededge740 <- (pod$Red + pod$NIR) /1.959

  pod$OSAVI <- 1.5*((pod$NIR - pod$Red) / (pod$NIR + pod$Red +0.16))
  pod$NDVI <- (pod$NIR - pod$Red) / (pod$NIR + pod$Red)
  pod$MCARI <- (pod$Rededge740 - pod$Rededge705) - (0.2 * (pod$Rededge740 - pod$Green) * (pod$Rededge740 + p
od$Rededge705))
  pod$GNDVI <-(pod$NIR - pod$Green) / (pod$NIR + pod$Green)
  pod$GOSAVI <- 1.5*((pod$NIR - pod$Green) / (pod$NIR + pod$Green + 0.16))
  pod$PSSR <-pod$NIR / pod$Red
  pod$TCARI <- (pod$Rededge705 - pod$Red) - (0.2 * (pod$Rededge705 - pod$Green) * (pod$Rededge705 / pod$Red)
)
  pod$TVI <- 0.5 * (120 * (pod$Rededge740 - pod$Green) - 200 * (pod$Red - pod$Green))
  pod$GLI <- ((2 * pod$Green) - pod$Red - pod$Blue) /((2 * pod$Green + pod$Red + pod$Blue))
  pod$MTVI1 <- 1.2 * (1.2 * (pod$NIR - pod$Green) - 2.5 * (pod$Red - pod$Green))
  pod$EVI2 <- 2.4 * (pod$NIR + pod$Red) / (pod$NIR + pod$Red + 1)
  return(pod)
})
names(indeksi) <- dnevi

skupaj <- rbindlist(indeksi, idcol = "DATUM")
```

Interpolacije za vse dni v letu za vsako poljino

Hide

```
leto <- strsplit(as.character(Sys.Date()), "-")[[1]][1]
od <- paste("01-01-", leto, sep = "")
do <- paste("31-12-", leto, sep = "")

vsi_dnevi <- data.frame(format(seq.Date(as.Date("01-01-2017", format = "%d-%m-%Y"),as.Date("31-12-2017", for
mat = "%d-%m-%Y"),by = 1), "%Y-%m-%d"))
names(vsi_dnevi) <- 'DATUM'
rm(od, do)


po.poljinah <- split(skupaj,skupaj$POLJINA_ID)

po.poljinah <- lapply(po.poljinah, function(x) {
  polje <- subset(x, select = c(DATUM, POLJINA_ID, SIFRA_KMRS,
                                EVI2, GLI, GNDVI, GOSAVI, MCARI, MTVI1, NDVI, OSAVI, PSSR, TCARI, TVI))
  print(polje$POLJINA_ID[[1]])
  polje$DATUM <- as.factor(polje$DATUM)
  polje$POLJINA_ID <- as.factor(polje$POLJINA_ID)

  zdruzeno <- merge(polje, vsi_dnevi, by = "DATUM", all.y = T)
  zdruzeno <- zdruzeno[order(as.Date(as.character(zdruzeno$DATUM), format = "%Y-%m-%d")),]
  zdruzeno <- data.frame(zdruzeno)

  for (indeks in 4:14) {
    inter <- na_interpolation(zdruzeno[,indeks], option = "linear")
    zdruzeno[,indeks] <- inter
  }

  if (nrow(zdruzeno) > 365) {
    zdruzeno <- aggregate(.~DATUM, zdruzeno[,c(1, 4:14)], mean)
  }

  #na eni poljini je celo leto isti posevek, isto stevilo pixlov
  zdruzeno$POLJINA_ID <- polje$POLJINA_ID[1]
  zdruzeno$SIFRA_KMRS <- polje$SIFRA_KMRS[1]
  zdruzeno$vrsta_poljine <- NA

  # dodamo spremenljivko "vrsta_poljine" iz vrednosti KMRS
  for (k in 1:nrow(zdruzeno)) {
    if (zdruzeno$SIFRA_KMRS[k] == 005 | zdruzeno$SIFRA_KMRS[k] == 006) {
      zdruzeno$vrsta_poljine[k] <- "koruza"
    } else if (zdruzeno$SIFRA_KMRS[k] == 801) {
      zdruzeno$vrsta_poljine[k] <- "ozimna_psenica"
    } else if (zdruzeno$SIFRA_KMRS[k] == 809) {
      zdruzeno$vrsta_poljine[k] <- "ozimni_jecmen"
    } else if (zdruzeno$SIFRA_KMRS[k] == 110 | zdruzeno$SIFRA_KMRS[k] == 206 |
               zdruzeno$SIFRA_KMRS[k] == 207 | zdruzeno$SIFRA_KMRS[k] == 208 |
               zdruzeno$SIFRA_KMRS[k] == 219 | zdruzeno$SIFRA_KMRS[k] == 220 |
               zdruzeno$SIFRA_KMRS[k] == 222 | zdruzeno$SIFRA_KMRS[k] == 223) {
      zdruzeno$vrsta_poljine[k] <- "metuljnice"
    } else if (zdruzeno$SIFRA_KMRS[k] == 117 | zdruzeno$SIFRA_KMRS[k] == 118 |
               zdruzeno$SIFRA_KMRS[k] == 200 | zdruzeno$SIFRA_KMRS[k] == 201 |
               zdruzeno$SIFRA_KMRS[k] == 202 | zdruzeno$SIFRA_KMRS[k] == 203) {
      zdruzeno$vrsta_poljine[k] <- "travnik_na_njivi"
    } else if (zdruzeno$SIFRA_KMRS[k] == 204) {
      zdruzeno$vrsta_poljine[k] <- "trajno_travinje"
    }
  }
  return(zdruzeno)
})

vse.poljine <- rbindlist(po.poljinah, use.names = T)
write.csv(vse.poljine, "PlanetScope_KRAS_2017_vse-poljine_vsi-dnevi.csv", row.names = F)
```

Izračun PCA

Hide

```
# urejanje po dnevih
vse.poljine$DATUM <- as.Date(vse.poljine$DATUM, format = "%Y-%m-%d")
a <- levels(as.factor(vse.poljine$DATUM))
a <- a[order(a)]
vse.poljine$DATUM <- factor(vse.poljine$DATUM, levels = a)

dnevi <- split(vse.poljine, vse.poljine$DATUM)

pca <- lapply(dnevi, function(x) {
    dat <- x
    dat[mapply(is.infinite, dat)] <- NA
    dat <- na_locf(dat) #iz zoo ali imputeTS, NA zamenja z zadnjo vrednostjo
    dat.scale <- apply(dat[,4:14], 2, scale)
    dat[,4:14] <- data.frame(dat.scale)
    dat.pca <- prcomp(dat[,4:14])
    dat.skupaj <- data.frame(dat[,1:3], dat.pca$x, dat[,15])
    names(dat.skupaj)[15] <- "vrsta_poljine"
    return(dat.skupaj)
})


for (i in 1:length(pca)) {
  if (i == 1) {
    pca1 <- data.frame(pca[[i]][,c(1:3,15, 4)])
    ime <- paste("Dne_", names(pca)[i], sep = "")
  } else {
    pca1 <- data.frame(cbind(pca1, pca[[i]][,4]))
    ime <- c(ime, paste("Dne_", names(pca)[i], sep = ""))
  }
}
names(pca1)[5:ncol(pca1)] <- ime

pca1$regija <- strsplit(getwd(), "/")[[1]][4]
pca1 <- pca1[,c(1,2,3,370,5:369,4)]
pca1$vrsta_poljine <- pca1$vrsta_poljine %>% fct_collapse(travinje = c("trajno_travinje","metuljnice","travn
ik_na_njivi"),
                                                          zita = c("ozimni_jecmen","ozimna_psenica"))
write.csv(pca1, "PCA1_Kras_3r.csv", row.names = F)
rm(pca)
```

MCARI in NDVI

Hide

```
for (i in 1:length(dnevi)) {
  if (i == 1) {
    mcari <- data.frame(dnevi[[i]][,c(1,2,3,15,8)])
  } else {
    mcari <- data.frame(cbind(mcari, dnevi[[i]][,8]))
  }
}
names(mcari)[5:ncol(mcari)] <- ime
mcari <- mcari[,c(1,2,3,5:ncol(mcari),4)]

write.csv(mcari, "MCARI_Kras.csv", row.names = F)



for (i in 1:length(dnevi)) {
  if (i == 1) {
    ndvi <- data.frame(dnevi[[i]][,c(1,2,3,15,10)])
  } else {
    ndvi <- data.frame(cbind(ndvi, dnevi[[i]][,10]))
  }
}
names(ndvi)[5:ncol(ndvi)] <- ime
ndvi <- ndvi[,c(1,2,3,5:ncol(ndvi),4)]

write.csv(ndvi, "NDVI_Kras.csv", row.names = F)
```

Zdruzevanje podatkov obmocij oziroma statisticnih regij

Hide

Hide

```
setwd(choose.dir()) # kjer so spravljeni podatki vseh območij oziroma statističnih regij

# združevanje PCA
# v eni mapi spravljene vse datoteke s podatki PCA1
datoteke <- list.files(pattern = c('PCA1'))

seznam <- list()
for (i in 1:length(datoteke)) {
  print(i)
  seznam[[i]] <- read.csv(datoteke[i])
}
pca1 <- rbindlist(seznam)
write.csv(pca1, file.choose(), row.names = F)

# MCARI in NDVI
datmcari <- list.files(pattern = 'MCARI')
seznam <- list()
for (i in 1:length(datmcari)) {
  seznam[[i]] <- read.csv(datmcari[i])
}
mcari <- rbindlist(seznam)
write.csv(mcari, file.choose(), row.names = F)

datndvi <- list.files(pattern = 'NDVI')
seznam <- list()
for (i in 1:length(datndvi)) {
  seznam[[i]] <- read.csv(datndvi[i])
}
ndvi <- rbindlist(seznam)
write.csv(ndvi, file.choose(), row.names = F)
```

Zaznavanje sprememb v časovni vrsti

Hide

```
mcari.pol <- mcari$POLJINA_ID
mcari <- mcari[,4:369]
orano <- ndvi[,4:369]
imena <- paste(substr(names(orano)[1:365], 13,14), ".", substr(names(orano)[1:365], 10, 11), ".", sep = "")

orano$kosnja <- NA
okno <- 10 # rabimo za drseče povprečje
orano$dnevi <- 0
orano$oran_klas <- NA
korak <- 15 # časovni korak za določanje sprememb. Prenizka vrednost bi zaznala tudi naravno prisotno nihanj
e vrednosti indeksov, previsoka pa zgreši dogodke
for (row in 1:nrow(orano)) {
  counter <- 0
  n <- 0
  dan <- vector()
  oran <- 0
  dano <- vector()
  m <- 0
  if (row > 1) {
    a <- mav(as.numeric(as.vector(as.character(orano[row,1:365]))), okno)
    b <- mav(as.numeric(as.vector(as.character(mcari[row,1:365]))), okno)
    a <- c(na_locf(a[1:100], option = "nocb"), a[101:200], na_locf(a[201:length(a)])) #prvo NA vrednost pote
gnemo na zacetek in zadnjo na konec
    b <- c(na_locf(b[1:100], option = "nocb"), b[101:200], na_locf(b[201:length(b)])) #tako nimamo na začetk
u in koncu leta samih ničel
    y <- 1:length(b) #tole so dnevi, torej dejanska x-os
    for (i in 1:length(a)) {
      n <- n + i
      if (n >= length(a)-okno) {
        break
      } else {
        if (lm(b[n:(n+korak)]~y[n:(n+korak)])$coefficients[2] < -0.0001) { #tule zaznamo hitro spremembo vre
dnost indeksa; je deloma neodvisno od samega indeksa
          if (n %in% 94:291) { #kosnjo gledamo od aprila do 15.10.
            counter <- counter + 1
            dan <- c(dan, imena[n])
          }
```

```
          }
          m <- n
          if (m %in% c(1:120,247:368)) { #oranje gledamo od januarja do aprila in septembra do decembra
            for (j in 0:30) { #za oranje nas zanima kako dolgo traja obdobje nizkih vrednosti
              if ((m+j) > length(a)) {
                break
              } else {
                if (a[m+j] <= 0.2 & a[m+j] >= 0) { #pod 0.2 stejemo da je zemlja, negativne vrednosti pa sne
g,megla,voda itd.
                  oran <- oran + 1
                }
              }
            }
          }
          n <- n + 30
        }
      }
    }
  }
  if (oran >= 30) { #stevilo dni z vrednostmi pod 0.2, ce je vec kot 30 damo isto vrednost; razdelimo v tri
razrede
    oran <- 2
  } else if (oran > 0 & oran < 30) {
    oran <- 1
  } else {
    oran <- 0
  }
  orano$kosnja[row] <- counter
  orano$dnevi[row] <- paste(dan, collapse = ',')
  orano$oran_klas[row] <- oran
  if (row %% 1000 == 0) { #spremljamo dogajanje. Vsakih 1000 poljin se izpiše doseženo število
    print(row)
  }
}

count(orano$kosnja)
count(orano[orano$vrsta_poljine %in% c('travnik_na_njivi','trajno_travinje','metuljnice'),367])

count(orano$oran_klas)
count(orano[orano$vrsta_poljine %in% c('travnik_na_njivi','trajno_travinje','metuljnice'),369])
```

Določanje vrste posevka

Hide

```
require(xgboost)

model.max <- xgb.load("E:\\CRP_V4-1811_podatki_poljine\\z_virtualke\\2019\\rezultat\\XGBoost_max_klasifikaci
ja.model")
model.prob <- xgb.load("E:\\CRP_V4-1811_podatki_poljine\\z_virtualke\\2019\\rezultat\\XGBoost_prob_klasifika
cija.model")

pca <- fread(file.choose())
regija <- pca$regija %>% fct_collapse("0" = "dolenjska",
                                      "1" = "gorenjska",
                                      "2" = "goriska",
                                      "3" = "koroska",
                                      "4" = "notranjska",
                                      "5" = "obala",
                                      "6" = "osSlo",
                                      "7" = "posavje",
                                      "8" = "prekmurje",
                                      "9" = "savinjska",
                                      "10" = "stajerska",
                                      "11" = "zasavje")
vrsta_poljine <- pca$vrsta_poljine %>% fct_collapse("0" = "koruza",
                                                    "1" = "travinje",
                                                    "2" = "zita")

regija <- as.numeric(as.character(regija))
vrsta_poljine <- as.numeric(as.character(vrsta_poljine))

pca <- scale(cbind(regija, pca[,4:368]))
```

```
dpred <- xgb.DMatrix(data = as.matrix(pca), label = as.matrix(vrsta_poljine))

pred.prob <- predict(model.prob, dpred, reshape = T)
pred.max <- predict(model.max, dpred)
pred <- data.frame(as.integer(as.character(pca1$POLJINA_ID)), vrsta_poljine, pred.max, pred.prob)
names(pred) <- c("POLJINA_ID", "vrsta_polj","pop_klas","pkoruza","ptravinje","pzita")


xtab <- table(vrsta_poljine, pred.max)
caret::confusionMatrix(xtab)

mcari <- read.csv('MCARI_2019.csv') #če je mcari naložen že od prej, potem to ni potrebno
klasifikacije <- merge(mcari[,c(2,3,4:369)], pred, by = 'POLJINA_ID') #združimo klasifikcije in izbrane stol
pce mcari


klasifikacije$vrsta_polj <- as.factor(klasifikacije$vrsta_polj) %>% fct_collapse(koruza = c("0"),
                                                                                  travinje = c("1"),
                                                                                  zita = c("2"))
klasifikacije$pop_klas <- as.factor(klasifikacije$pop_klas) %>% fct_collapse(koruza = c("0"),
                                                                              travinje = c("1"),
                                                                              zita = c("2"))

# fenofaze
klasifikacije$pop_klas <- apply(klasifikacije, 1, function(x) {
  dat <- x[c(369,368,3:367)]
  popravek <- vector()
  maks.vrednost <- max(dat[2:length(dat)])
  pozicija.max <- match(maks.vrednost, dat[2:length(dat)])
  if (pozicija.max %in% c(180:242) & dat[2] == 'koruza' & dat[1] == 'zita') { #1 je predicted, 2 je actual
    popravek <- 'koruza'
  } else if (pozicija.max %in% c(99:167) & dat[2] == 'zita' & dat[1] == 'koruza') {
    popravek <- 'zita'
  } else if (pozicija.max %in% c(99:167) & dat[2] == 'zita' & dat[1] == 'travinje') {
    popravek <- 'zita'
  } else if (pozicija.max %in% c(180:242) & dat[2] == 'koruza' & dat[1] == 'travinje' & dat[pozicija.max] >=
0.035) {
    popravek <- 'koruza'
  } else {
    popravek <- dat[1]
  }
  return(popravek)
})

xtab <- table(vrsta_poljine, klasifikacije$pop_klas)
caret::confusionMatrix(xtab)

napake_klas_bin <- function(x, vrsta) {
  rez <- vector()
  if (vrsta == "skupno") {
    if (x[3] != x[4]) {
      rez <- 1
    } else {
      rez <- 0
    }
  } else if (vrsta == "travinje") {
    if (x[3] != x[4] & x[3] == vrsta) {
      rez <- 1
    } else {
      rez <- 0
    }
  } else if (vrsta == "koruza") {
    if (x[3] != x[4] & x[3] == vrsta) {
      rez <- 1
    } else {
      rez <- 0
    }
  } else if (vrsta == "zita") {
    if (x[3] != x[4] & x[3] == vrsta) {
      rez <- 1
    } else {
      rez <- 0
    }
```

```
  }
  return(rez)
}

klasifikacije$err_bin <- apply(klasifikacije, 1, function(x) napake_klas_bin(x, "skupno")) #skupne napačne k
lasifikacije
klasifikacije$err_kor <- apply(klasifikacije, 1, function(x) napake_klas_bin(x, "koruza")) #napačno klasific
irane koruze
klasifikacije$err_tr <- apply(klasifikacije, 1, function(x) napake_klas_bin(x, "travinje")) #napačno klasifi
cirano travinje
klasifikacije$err_zita <- apply(klasifikacije, 1, function(x) napake_klas_bin(x, "zita")) #napačno klasifici
rana žita

#uredimo rezultate klasifikacij, združimo s košnjo in oranjem ter shranimo
klasifikacije <- klasifikacije[,c(1,368:376)]
rezultat <- data.frame(klasifikacije, orano[,c(366:368)])
write.csv(rezultat, file.choose(), row.names = F)
```

Rezultate uredimo v shapefile

Hide

```
shape.zdruzeno <- merge(kras[,c(4,5,6,9)], rezultat, by = "POLJINA_ID")
```

```
Warning message:
In readChar(file, size, TRUE) : truncating string with embedded nuls
```

Hide

```
shape.zdruzeno <- shape.zdruzeno[-which(is.na(shape.zdruzeno@data[,5]), arr.ind = T),]

writeOGR(shape.zdruzeno, ".","rezultati", drive = "ESRI Shapefile", overwrite_layer = T)
plot(shape.zdruzeno)
```