CRP V4-1811 urejanje podatkov in izračun rezultatov

Code

Uroš Žibrat

Celoten postopek je sledeč

1. priprava podatkov za vsako območje oziroma statistično regijo (čiščenje in preurejanje podatkov, izračun indeksov in interpolacije)
2. izračun glavnih komponent za vsako poljino za celo leto
3. preurejanje indekov MCARI in NDVI za celo leto za vsako poljino

[Točke 1-3 opravimo ločeno za vsako območje]

4. združimo podatke vseh območij
5. Določanje košnje in oranja
6. Klasifikacije vrste posevkov
7. Presek z zbirnimi vlogami in izdelava .shp datoteke

Naložimo potrebne knjižnice.

Hide

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

Pripravimo funkciji za izračun in filtriranje po mediani ter izračun drsečega povprečja.

Hide

```
#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)
}
```

Določimo delovno mapo in v njej poiščemo vse datoteke, ki vsebujejo besedno zvezo “bands_out”. Te datoteke nato zložimo po datumu. Predpostavljamo, da so v eni mapi vsi podatki za eno območje oziroma statistično regijo. Prav tako predpostavljamo strukuro imen datotek, kot je nastavljena v Python eo-learn skripti za prevzem slik.

Sledeč postopek odpiranja datotek in urejanja podatkov je spominsko zahteven. Priporočamo, da se podatke ureja ločeno za vsako statstično regijo in se jih kasneje združi. Priporočena najmanjša količina RAM-a je 16 GB. Pri 128 GB RAM je možno urejanje večih območih naenkrat, vsako v svojem oknu. Tako lahko postopek tudi pohitrimo.

Hide

```
setwd(choose.dir())
datoteke <- list.files(pattern = 'bands_out')
datoteke <- datoteke[order(as.Date(substr(datoteke, 15,24), format = "%d-%m-%Y"))] # datoteke uredimo po dne
vih
```

V primeru, če je potrebno popraviti datume v imenih datotek; če datumi niso formata dd-mm-yyyy, ampak d-m-yyyy oziroma ustrezno prilagoditi.

Hide

```
#for (i in 1:length(datoteke)) {
#  dat <- strsplit(datoteke[i], "_")[[1]][c(6,5,4)]
#  dat <- paste(dat[1], dat[2], dat[3], sep = "-")
#  nr <- strsplit(datoteke[i], "_")[[1]][c(3)]
#  if (nchar(nr) == 1) {
#    nr <- paste("00",nr, sep = "")
#  } else if (nchar(nr) == 2) {
#    nr <- paste("0",nr, sep = "")
#  }
#  dat <- paste("bands_out_", nr, "_", dat, ".csv", sep = "")
#  file.rename(from = datoteke[i], to = dat)
#  datoteke[i] <- dat
#}
```

Tole filtrira datoteke glede na število posnetkov; če jih je v celem letu manj kot 10, jih ne upošteva. Do tega lahko pride, če je bila težava pri prevzemu posnetkov.

Hide

```
#for (i in 1:length(datoteke)) {
#  dat <- strsplit(datoteke[i], "_")[[1]][3]
#  if (i == 1) {
#    nr <- dat
#  } else {
#    nr <- c(nr,dat)
#  }
#}
#cnr <- subset(count(nr), freq >= 10)
#dat <- datoteke[sapply(strsplit(datoteke, "_"), function(x) x[3] %in% cnr$x)]
#datoteke <- dat
```

Odpremo vsako datoteko, torej vse dneve za eno območje. Iz datoteke poberemo spremenljivke, ki jih potrebujemo, in filtriramo po KMRS. Za posamezen dan za vsako poljino izračunamo in filtriramo po mediani, da dobimo povprečne spektralne podpise. Nato izračunamo vegetacijske indekse in na koncu združimo v eni tabeli. To tabelo nato shranimo.

V objektu “skupaj” so združeni podatki vseh poljin in vseh dni (ne vsi dnevi v letu, ampak dnevi za katere so bili na voljo posnetki) za eno območje.

Hide

```
# odpremo vse datoteke vseh dni za eno obmocje
podatki <- list()
for (i in 1:length(datoteke)) {
  print(paste("odpiram datum ", substr(datoteke[i], 15,24)))
  tabela <- fread(file = datoteke[i])
  tabela <- subset(tabela, select = c(DATUM, KMG_MID, GERK_PID, POLJINA_ID, Blue,Green,Red,REdge705,REdge740
,REdge783,NIR,REdge865, SIFRA_KMRS))
  tabela <-  subset(tabela, subset = SIFRA_KMRS %in% c(005,006,801,809,110,206, 207, 208, 219, 220, 222, 223
, 117, 118, 200, 201, 202, 203, 204))

  #tukaj filtriramo po mediani za vsako poljino.
  tabela.split <- split(tabela[,c(2:13)], tabela$POLJINA_ID)
  tabela.split <- lapply(tabela.split, 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)
    }
  })

  dat <- data.frame(do.call(rbind, tabela.split))
  dat <- data.frame(dat[complete.cases(dat),])
  rm(tabela.split)

  #izračun indeksov
  dat$OSAVI <- 1.5*((dat$NIR - dat$Red) / (dat$NIR + dat$Red +0.16))
  dat$NDVI <- (dat$NIR - dat$Red) / (dat$NIR + dat$Red)
  dat$MCARI <- (dat$REdge740 - dat$REdge705) - (0.2 * (dat$REdge740 - dat$Green) * (dat$REdge740 + dat$REdge
705))
  dat$GNDVI <-(dat$NIR - dat$Green) / (dat$NIR + dat$Green)
  dat$GOSAVI <- 1.5*((dat$NIR - dat$Green) / (dat$NIR + dat$Green + 0.16))
  dat$PSSR <-dat$NIR / dat$Red
  dat$TCARI <- (dat$REdge705 - dat$Red) - (0.2 * (dat$REdge705 - dat$Green) * (dat$REdge705 / dat$Red))
  dat$TVI <- 0.5 * (120 * (dat$REdge740 - dat$Green) - 200 * (dat$Red - dat$Green))
  dat$GLI <- ((2 * dat$Green) - dat$Red - dat$Blue) /((2 * dat$Green + dat$Red + dat$Blue))
  dat$MTVI1 <- 1.2 * (1.2 * (dat$NIR - dat$Green) - 2.5 * (dat$Red - dat$Green))
  dat$EVI2 <- 2.4 * (dat$NIR + dat$Red) / (dat$NIR + dat$Red + 1)

  #preoblikujemo datume
  dat$DATUM <- as.Date(as.factor(substr(datoteke[i], 15,24)), format = "%d-%m-%Y")
  dat <- dat[,c(24,1:3,12,4:11,13:23)]
  podatki[[i]] <- dat

  print(paste("koncana tabela stevilka ", i, " Ostalo je se ", length(datoteke) - i, " tabel"))
}

skupaj <- rbindlist(podatki)
rm(podatki) #počistimo delovni prostor
gc()

write.csv(skupaj, file.choose()) #shranimo podatke
```

Hide

```
head(skupaj)
```

Za interpolacije potrebujemo spremenljivko z vsemi dnevi v letu.

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(od, format = "%d-%m-%Y"),as.Date(do, format = "%d-%m-%Y"),by
= 1), "%Y-%m-%d"))
names(vsi_dnevi) <- 'DATUM'
rm(od, do)
```

Podatke zdaj razdelimo po POLJINA_ID, za vsako poljino razširimo obstoječe podatke na celo leto, s čimer ustvarimo vrednosti “NA”. Te zapolnimo z interpolacijo obstoječih vrednosti iz posnetkov. To naredimo za vsak indeks za vsako poljino za celo leto.

Na koncu vse poljine združimo v eno tabelo in jo shranimo.

Hide

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

# po.poljinah je razdeljen na posamezne poljine, torej spodaj delamo za vsako poljino posebej
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)

  # razširimo obstoječe podatke na celo leto
  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)

  # za vsak indeks v poljini izvedemo interpolacijo
  for (indeks in 4:14) {
    if (nrow(polje) <= 2) {
      zdruzeno[,indeks] <- zdruzeno[which(!is.na(zdruzeno[,indeks])),indeks][1] #tukaj za vsak slučaj, če st
a pri poljini manj kot 2 podatka za celo leto
    } else {
      inter <- na_interpolation(zdruzeno[,indeks], option = "linear")
      zdruzeno[,indeks] <- inter
    }
  }
  if (nrow(zdruzeno) > 365) {
    zdruzeno <- aggregate(.~DATUM, zdruzeno[,c(1, 4:14)], mean)
   }

  #uredimo POLJINA_ID in KMRS
  zdruzeno$POLJINA_ID <- polje$POLJINA_ID[1]
  zdruzeno$SIFRA_KMRS <- polje$SIFRA_KMRS[1]

  # dodamo spremenljivko "vrsta_poljine" iz vrednosti KMRS
  zdruzeno$vrsta_poljine <- NA
  zdruzeno$vrsta_poljine <- apply(zdruzeno, 1, function(x) {
    if (x[3] == 005 | x[3] == 006) {
      x[15] <- "koruza"
    } else if (x[3] == 801) {
      x[15] <- "ozimna_psenica"
    } else if (x[3] == 809) {
      x[15] <- "ozimni_jecmen"
    } else if (x[3] == 110 | x[3] == 206 |
               x[3] == 207 | x[3] == 208 |
               x[3] == 219 | x[3] == 220 |
               x[3] == 222 | x[3] == 223) {
      x[15] <- "metuljnice"
    } else if (x[3] == 117 | x[3] == 118 |
               x[3] == 200 | x[3] == 201 |
               x[3] == 202 | x[3] == 203) {
      x[15] <- "travnik_na_njivi"
    } else if (x[3] == 204) {
      x[15] <- "trajno_travinje"
    }
  })

  return(zdruzeno)
})

#združimo v eno tabelo; torej vse poljine, vsi dnevi za eno območje
vse.poljine <- rbindlist(po.poljinah, use.names = T, fill = T)
rm(po.poljinah)
gc()

write.csv(vse.poljine, file.choose(), row.names = F)
```

Hide

```
head(vse.poljine)
```

Izvedemo analizo glavnih komponent (principal component analysis - PCA). Osi PCA potrebujemo za klasifikacije.

PCA računamo za vse poljine, po dnevih. Zato moramo vse.poljine razdeliti po dnevih in znotraj posameznega dne izračunati PCA

Hide

```
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)

#razdelimo po dnevih
dnevi <- split(vse.poljine, vse.poljine$DATUM)

#izračunamo PCA
pca <- lapply(dnevi, function(x) {
  dat <- x
  print(as.character(dat$DATUM[1]))
  vrsta_poljine <- dat$vrsta_poljine
  dat <- subset(dat, select = -c(vrsta_poljine))
  dat[mapply(is.infinite, dat)] <- NA
  dat <- na_locf(dat) # 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, vrsta_poljine)
  names(dat.skupaj)[15] <- "vrsta_poljine"
  return(dat.skupaj)
})

#potrebujemo samo prvo PCA os, tukaj sestavimo novo tabelo "pca1" s podatki za vse dni za vse poljine enega
območja
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

# ime regije; stevilko v zadnjem oklepaju je potrebno popraviti glede na pozicijo imena regije v strukturi m
ap
# predpostavka je, da so podatki za posamezni regijo v eni mapi z imenom regije, mapa pa na 4. mestu po razd
elitvi po znaku "/"
pca1$regija <- strsplit(getwd(), "/")[[1]][4]

pca1 <- pca1[,c(1,2,370,5:369,4)]

# vrsto poljine reduciramo na tri razrede, koruza, travinje in zita.
pca1$vrsta_poljine <- pca1$vrsta_poljine %>% fct_collapse(travinje = c("trajno_travinje","metuljnice","travn
ik_na_njivi"),
                                                          zita = c("ozimni_jecmen","ozimna_psenica"))

# pca1 shranimo
write.csv(pca1, file.choose(), row.names = F)
rm(pca)
```

Hide

```
head(pca1)
```

Ostalo nam je še urejanje tabel z MCARI in NDVI, ki ju potrebujemo za določanje košnje in oranja. V seznamu “dnevi” imamo spravljene vse indekse za vsak dan. Tukaj izberemo MCARI in NDVI in ju shranimo kot ločeni tabeli.

Hide

```
for (i in 1:length(dnevi)) {
  if (i == 1) {
    mcari <- data.frame(dnevi[[i]][,c(1,2,3,15,8)]) #MCARI je na 8 mestu v tabeli
  } 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, file.chose(), row.names = F)

for (i in 1:length(dnevi)) {
  if (i == 1) {
    ndvi <- data.frame(dnevi[[i]][,c(1,2,3,15,10)]) #NDVI je na 10. mestu v tabeli
  } 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, file.choose(), row.names = F)
```

Hide

```
head(mcari)
```

S tem smo zaključili pripravo podatkov za eno območje.

Zdaj moramo združiti podatke vseh območij v eno tabelo. Predpostavljamo, da so posamezne zgoraj pripravljene datoteke vseh območij shranjene v eni mapi.

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 košnje in oranja

Hide

```
#priprava podatkov
mcari.pol <- mcari$POLJINA_ID
mcari <- mcari[,4:368]
orano <- ndvi[,4:368]
imena <- paste(substr(names(orano)[1:365], 13,14), ".", substr(names(orano)[1:365], 10, 11), ".", sep = "")
#datumi v lepši obliki za uporabo v tabeli rezultatov
```

Hide

```
head(imena)
```

```
[1] "01.01." "02.01." "03.01." "04.01." "05.01." "06.01."
```

Pripravimo tabelo in poiščemo hitre spremembe indeksa MCARI v časovni vrsti. Indeks NDVI uporabljamo za oceno ali je vrednost padla pod 0,2, kar nakazuje zemljo oziroma spremembo od rastlinja do nečesa drugega.

Hide

```
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 #ko zaznamo dogodek se premaknemo za 30 dni naprej. Predpostavka, da v enem mesecu ne
kosijo 2x
        }
      }
    }
  }
  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)
  }
}
```

Hide

```
#rezultat za košnjo
count(orano$kosnja)
```

Hide

```
#rezultat za oranje
count(orano$oranje_class)
```

Hide

```
plyr::count(orano[orano$vrsta_poljine == "travinje", 10])
```

Določanje vrste posevka

Naložimo klasifikacijski model

Hide

```
model.max <- xgb.load("E:\\CRP_V4-1811_podatki_poljine\\z_virtualke\\2019\\rezultat\\XGBoost_max_klasifikacija.model")
model.prob <- xgb.load("E:\\CRP_V4-1811_podatki_poljine\\z_virtualke\\2019\\rezultat\\XGBoost_prob_klasifikacija.model")
```

Hide

```
model.max
```

```
##### xgb.Booster
raw: 17.8 Mb
xgb.attributes:
  niter
niter: 19
```

Naložimo podatke PCA združene v eni tabeli za vsa območja in prekodiramo vrsto posevka in regijo v naravna števila. To je potrebno zaradi modela, ki prepozna samo številke.

Hide

```
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))
poljina_id <- pca$POLJINA_ID
```

Uredimo podatke in izvedemo klasifikacijo.

Hide

```
pca <- scale(cbind(regija, pca[,5:369]))
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(pca$POLJINA_ID)), vrsta_poljine, pred.max, pred.prob)
names(pred) <- c("POLJINA_ID", "vrsta_polj","pop_klas","pkoruza","ptravinje","pzita")
```

Dodamo markerje za fenofaze in uredimo klasifikacije

Hide

```
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 <- pbapply::pbapply(klasifikacije, 1, function(x) {
  dat <- x[c(369,368,3:367)] #modelna klasifikacija, iz vloge, dnevi
  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)
})
```

Hide

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

```
Confusion Matrix and Statistics

             pred.max
vrsta_poljine      0      1      2
            0  55104   3576   3756
            1   1523 304687   3609
            2   7045   3459  46960

Overall Statistics

               Accuracy : 0.947
                 95% CI : (0.946, 0.947)
    No Information Rate : 0.725
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.878

 Mcnemar's Test P-Value : <2e-16

Statistics by Class:

                     Class: 0 Class: 1 Class: 2
Sensitivity             0.865    0.977    0.864
Specificity             0.980    0.957    0.972
Pos Pred Value          0.883    0.983    0.817
Neg Pred Value          0.977    0.941    0.980
Prevalence              0.148    0.725    0.126
Detection Rate          0.128    0.709    0.109
Detection Prevalence    0.145    0.721    0.134
Balanced Accuracy       0.923    0.967    0.918
```

Dodamo binarne spremenljivke, ki označujejo pravilnost klasifikacij.

Hide

```
napake_klas_bin <- function(x, vrsta) {
  rez <- vector()
  if (vrsta == "skupno") {
    if (x[370] != x[369]) {
      rez <- 1
    } else {
      rez <- 0
    }
  } else if (vrsta == "travinje") {
    if (x[369] != x[370] & x[369] == vrsta) {
      rez <- 1
    } else {
      rez <- 0
    }
  } else if (vrsta == "koruza") {
    if (x[369] != x[370] & x[369] == vrsta) {
      rez <- 1
    } else {
      rez <- 0
    }
  } else if (vrsta == "zita") {
    if (x[369] != x[370] & x[369] == 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)
```

Hide

```
head(data.frame(klasifikacije))
```

Nazadnje še naredimo shapefile za prikaz v GIS

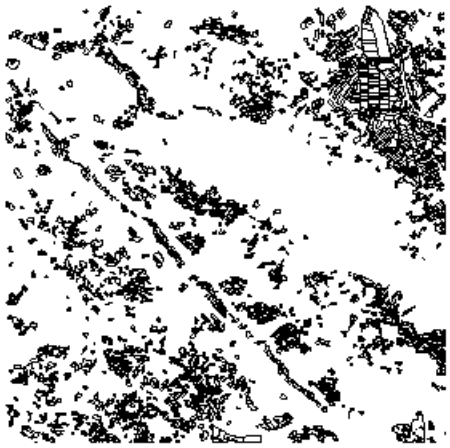
Hide

```
# iz shapefile datoteke uporabimo KMG_MID, BLOK_ID, GERK_PID, POLJINA_ID
vloge <- readOGR(dsn = file.choose()) #shapefile zbirnih vlog
shape.zdruzeno <- merge(vloge[,c(4,5,6,9)], rezultat, by = "POLJINA_ID") #združimo vloge in rezultate
shape.zdruzeno <- shape.zdruzeno[-which(is.na(shape.zdruzeno@data[,5]), arr.ind = T),] #odstranimo poljine,
ki jih nismo uporabili (torej tiste, ki smo jih filtrirali po KMRS)

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

Hide

```
plot(shape.zdruzeno)
```

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



Postopek je zaključen.

OPCIJA:

Za vsako poljino še izrišemo graf NDVI in MCARI.

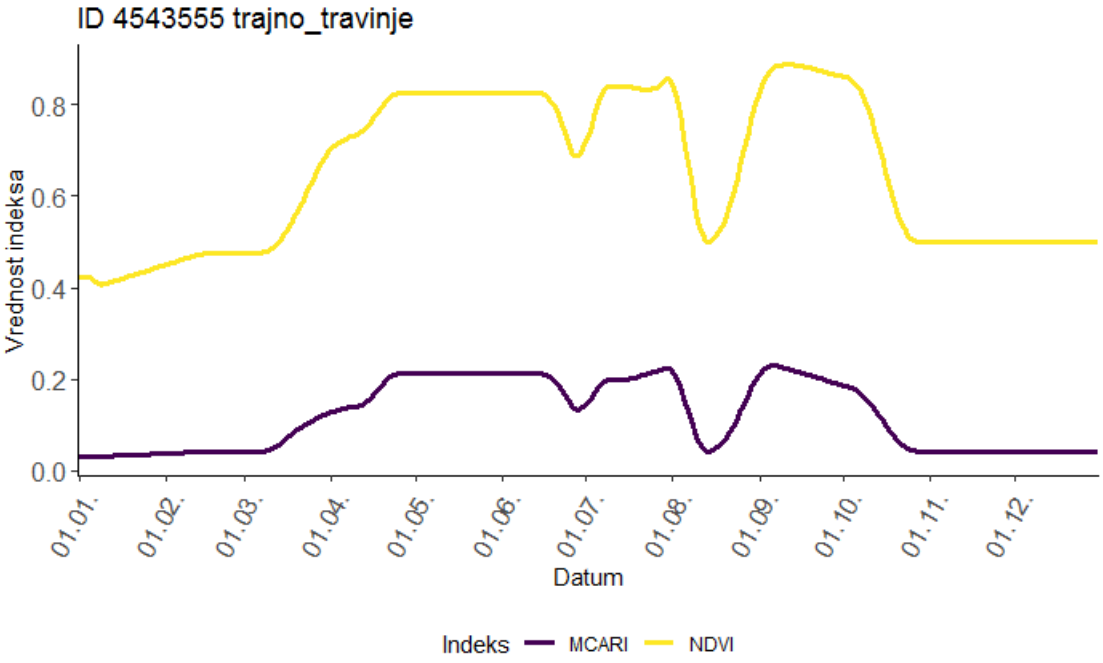
Hide

```
require(reshape2)
require(ggplot2)

# funkcija za izračun drsečega povprečja na grafih
mav_graf <- function(x,n=5){
  a <- mav(x, okno)
  a <- c(na_locf(a[1:100], option = "nocb"), a[101:200], na_locf(a[201:length(a)]))
  return(a)
}

# funkcija za izris grafov
# hkrati izriše MCARI in NDVI, po en graf za vsako poljino
grafi <- function(x,y, izris = F) {
  dat <- rbind(x,y)
  temp <- dat[,2:(ncol(dat)-1)]
  temp <- data.frame(t(apply(temp,1,function(x) mav_graf(x, okno))))
  dat[,2:366] <- temp
  dat[,(ncol(dat)+1)] <- c("MCARI","NDVI")
  names(dat)[ncol(dat)] <- "Indeks"
  names(dat)[2:366] <- imena #narejeno pri košnji in oranju
  ime.datoteke <- paste("ID_", dat$POLJINA_ID[1], ".png", sep = "")

  dat.melt <- melt(subset(dat, select = -c(vrsta_poljine)), id = c("Indeks", "POLJINA_ID"))

  graf <- ggplot(dat.melt, aes(x = variable, y = value, group = Indeks, colour = Indeks)) +  #mora biti inde
ks, ker je POLJINA_ID isti
    geom_line(size = 1.1) +
    scale_colour_viridis_d(option = "viridis") +
    labs(x = "Datum", y = 'Vrednost indeksa') +
    scale_x_discrete(breaks = c("01.01.","01.02.","01.03.","01.04.",
                                "01.05.","01.06.","01.07.","01.08.",
                                "01.09.","01.10.","01.11.","01.12.")) +
    scale_y_continuous(breaks = c(0,0.2, 0.4,0.6,0.8,1)) +
    theme_classic() +
    theme(axis.text.x = element_text(angle = 60, hjust = 1, size = 12),
          axis.text.y = element_text(size = 12),
          legend.position = 'bottom') +
    ggtitle(paste("ID", dat$POLJINA_ID[1], dat$vrsta_poljine[1]))

  #ustrezno vnesti pot do mape, kjer naj bodo slike spravljene
  ggsave(ime.datoteke, plot = last_plot(), dpi = 72, scale = 1,
```

```
        path = " ")
  if (izris == T) {
    return(graf) #če bi želeli izris grafa v konzoli; za testiranje, sicer upočasni postopek pri velikem šte
vilu grafov
  }
}

# izriše okno s progress bar, da je lažje spremljati, kako daleč je izris grafov
require(tcltk)
require(svMisc)
gui_progress <- function(value, max.value) {
  # če je treba odstraniti staro pogovorno okno
  if (value > max.value) {
    try(tkdestroy(get_temp("gui_progress_window")), silent = TRUE)
    delete_temp(c("gui_progress_state", "gui_progress_window",
                  "gui_progress_cancel"))
    return(invisible(FALSE))
  } else if (exists_temp("gui_progress_window") &&
             !inherits(try(tkwm.deiconify(tt <- get_temp("gui_progress_window")),
                           silent = TRUE), "try-error")) {
    # pogovorno okno obstaja, zato se fokusira nanj in spremeni vrednost
    tkfocus(tt)
    state <- get_temp("gui_progress_state")
    tclvalue(state) <- value
  } else {
    # pogovorno okno je treba na novo narediti
    # najprej preveri, da ni ostal še kak "cancel"
    delete_temp("gui_progress_cancel")
    # Tcl spremenljivka, v kateri je spravljena vrednost (progress value)
    state <- tclVar(value)
    assign_temp("gui_progress_state", state)
    # naredi novo pogovorno okno s progress bar-om
    tt <- tktoplevel()
    assign_temp("gui_progress_window", tt)
    tktitle(tt) <- "Waiting..."
    sc <- tkscale(tt, orient = "h", state = "disabled", to = max.value,
                  label = "Progress:", length = 200, variable = state)
    tkpack(sc)
    but <- tkbutton(tt, text = "Cancel", command = function() {
      # gumb za ustavitev postopka
      assign_temp("gui_progress_cancel", TRUE)
      tkdestroy(tt)
    })
    tkpack(but)
  }
  invisible(TRUE)
}
# registriramo kot funkcijo v progress()
change_temp(".progress", "gui_progress", gui_progress,
            replace.existing = TRUE)
rm(gui_progress) # počistimo

# če uporabljamo fread so tabele razred "data.frame data.table", s čimer so lahko tukaj težave. Zato pretvor
imov čisti data.frame
ndvi <- data.frame(ndvi)
mcari <- data.frame(mcari)

#izrišemo grafe
#postopek je počasen. Za 400 000 poljin potrebuje približno dva dneva
for (j in 1:nrow(mcari)) {
  grafi(mcari[j,3:ncol(mcari)], ndvi[j,c(3:ncol(ndvi))]) #uporabimo lahko tudi druge indekse, ampak je treba
biti pozoren na razpon vrednosti in po potrebi normalizirati -> recimo funkcija scale()
  progress(j, max.value = nrow(mcari), console = TRUE)
  if (exists_temp("gui_progress_cancel")) {
    progress(101, console = FALSE) # za vsak slučaj dodatno počistimo
    break
  }
}
```

Hide

```
grafi(mcari[23542,3:369], ndvi[23542,c(3:ncol(ndvi))])
```


ID 4543555 trajno_travinje
Vrednost indeksa
0.0
0.2
0.4
0.6
0.8
01.01.
01.02.
01.03.
01.04.
01.05.
01.06.
01.07.
01.08.
01.09.
01.10.
01.11.
01.12.
Datum
Indeks
MCARI
NDVI