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

#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 drsece povprecje
mav <- function(x, n= 5) {
  stats::filter(x, rep(1/n,n), sides = 2)
}

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 dnevih

#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
#}

#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 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)

  #izracun 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$REdge705))
  dat$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) #pocistimo delovni prostor
gc()

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

head(skupaj)

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)

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)

  # razsirimo obstojece 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
slucaj, ce sta 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)
})

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

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

head(vse.poljine)

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)

#izracunamo 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 obmocja
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 map
# predpostavka je, da so podatki za posamezni regijo v eni mapi z imenom
regije, mapa pa na 4. mestu po razdelitvi 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","travnik_na_njivi"),
                                                          zita =
c("ozimni_jecmen","ozimna_psenica"))

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

head(pca1)

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)

head(mcari)

setwd(choose.dir()) # kjer so spravljeni podatki vseh obmocij oziroma
statisticnih regij

# zdruzevanje 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)
```

```
#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 lepsi
obliki za uporabo v tabeli rezultatov
head(imena)

orano$kosnja <- NA
okno <- 10 # rabimo za drsece povprecje
orano$dnevi <- 0
orano$oran_klas <- NA
korak <- 15 # casovni korak za dolocanje sprememb. Prenizka vrednost bi
zaznala tudi naravno prisotno nihanje vrednosti indeksov, previsoka pa
zgresi dogodke
for (row in 1:nrow(orano)) {
  counter <- 0
  n <- 0
  dan <- vector()
  oran <- 0
  dano <- vector()
  m <- 0
  if (row > 1) {
    a <- mav(as.numeric(as.vector(as.character(orano[row,1:365]))), okno)
    b <- mav(as.numeric(as.vector(as.character(mcari[row,1:365]))), okno)
    a <- c(na_locf(a[1:100], option = "nocb"), a[101:200],
na_locf(a[201:length(a)])) #prvo NA vrednost potegnemo na zacetek in
zadnjo na konec
    b <- c(na_locf(b[1:100], option = "nocb"), b[101:200],
na_locf(b[201:length(b)])) #tako nimamo na zacetku in koncu leta samih
nicel
    y <- 1:length(b) #tole so dnevi, torej dejanska x-os
    for (i in 1:length(a)) {
      n <- n + i
      if (n >= length(a)-okno) {
        break
      } else {
        if (lm(b[n:(n+korak)]~y[n:(n+korak)])$coefficients[2] < -0.0001)
{ #tule zaznamo hitro spremembo vrednost 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 sneg,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
izpise dosezeno stevilo
    print(row)
  }
}

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

# klasifikacije
model.max <- xgb.load("E:\\CRP_V4-
1811_podatki_poljine\\z_virtualke\\2019\\rezultat\\XGBoost_max_klasifikac
ija.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$vrsta_poljine %>% fct_collapse("0" = "koruza",
                                                    "1" = "travinje",
                                                    "2" = "zita")
regija <- as.numeric(as.character(regija))
vrsta_poljine <- as.numeric(as.character(vrsta_poljine))
poljina_id <- pca$POLJINA_ID

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")

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


mcari <- read.csv('MCARI_2019.csv') #ce je mcari nalozen ze od prej,
potem to ni potrebno
klasifikacije <- merge(mcari[,c(2,3,4:369)], pred, by = 'POLJINA_ID')
#zdruzimo klasifikcije in izbrane stolpce 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)
})

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

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$err_bin <- apply(klasifikacije, 1, function(x)
napake_klas_bin(x, "skupno")) #skupne napacne klasifikacije
klasifikacije$err_kor <- apply(klasifikacije, 1, function(x)
napake_klas_bin(x, "koruza")) #napacno klasificirane koruze
klasifikacije$err_tr <- apply(klasifikacije, 1, function(x)
napake_klas_bin(x, "travinje")) #napacno klasificirano travinje
klasifikacije$err_zita <- apply(klasifikacije, 1, function(x)
napake_klas_bin(x, "zita")) #napacno klasificirana zita
```

```
#uredimo rezultate klasifikacij, zdruzimo s kosnjo 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)

head(klasifikacije)

# 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")
#zdruzimo 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


#
##
### izris grafov
##
#

require(reshape2)
require(ggplot2)

# funkcija za izracun drsecega povprecja 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 izrise MCARI in NDVI, po en graf za vsako poljino
grafi <- function(x,y) {
  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] <- datumi
  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 indeks, 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) #ce bi zeleli izris grafa v konzoli; za testiranje,
sicer upocasni postopek pri velikem stevilu grafov
  }
}

# izrise okno s progress bar, da je lazje spremljati, kako dalec je izris
grafov
require(tcltk)
require(svMisc)
gui_progress <- function(value, max.value) {
  # ce 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 se 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) # pocistimo

# ce uporabljamo fread so tabele razred "data.frame data.table", s cimer
so lahko tukaj tezave. Zato pretvorimov cisti data.frame
ndvi <- data.frame(ndvi)
mcari <- data.frame(mcari)

#izrisemo grafe
#postopek je pocasen. Za 400 000 poljin potrebuje priblizno dva dneva
for (j in 1:nrow(mcari)) {
  grafi(mcari[j,3:ncol(mcari)], ndvi[j,c(3:ncol(ndvi))])
  progress(j, max.value = nrow(mcari), console = TRUE)
  if (exists_temp("gui_progress_cancel")) {
    progress(101, console = FALSE) # za vsak slucaj dodatno pocistimo
    break
  }
}
```