Imputing crown closure

Kyle

17/02/2021

## Purpose

Natural origin stands do not have a projection of crown closure - VDYP doesn’t output crown closure over time. Thus, we will impute its value using the corresponding VRI crown closure estimate within 10 years of the reference year time stamp. To do this, a strata based imputation will be used, where only observations with the same leading species group will be used.

## Imputation parameter selection

Test some of the parameter options for nearest neighbor imputation. This includes the distance metric and number of neighbors, k.

species<-data.table(species = c('Dr', 'E', 'Ac', 'At', 'B', 'Cw','Yc', 'Fd', 'H', 'L', 'Mb', 'P', 'Sb', 'Ss','Se','Sw'), sql=c("('D','DG','DM','DR')","('E','EB','EE','ES','EW','EX','EXP','EXW','EA','EP')","('A','AC','ACB','ACT','AD','AX')","('AT')","('B','BB','BC','BM','BP','BL','BG','BA')", "('C','CW')","('Y','YC')","('F','FD','FDC','FDI')", "('H','HX','HXM', 'HM','HW')", "('L','LD','LS','LW','LA','LT')","('M','MB','ME','MN','MR','MS','MV')","('P','PM','PR','PS','PX', 'PL','PLC','PLI','PY','PW','PJ','PXJ','PF','PA')","('SB')","('SS')","('SE','SXE')","('SW', 'SXW','S','SA','SN','SX','SXB','SXL','SXS','SXX')"))  
strata<-lapply(seq\_len(nrow(species)), function(i) species[i])  
  
if(FALSE){  
   
error<-lapply(strata, function(x){  
 ref\_vri<-data.table(getTableQuery(paste0("SELECT crown\_closure, basal\_area, proj\_age\_1 as age, quad\_diam\_125 as qmd, proj\_height\_1 as height, vri\_live\_stems\_per\_ha as tph, live\_stand\_volume\_125 as tvol, bec\_zone\_code, species\_cd\_1 FROM veg\_comp\_lyr\_r1\_poly2020 where bclcs\_level\_2 = 'T' and reference\_year >= 2012 and species\_cd\_1 is not null and crown\_closure >= 0 and basal\_area >= 0 and proj\_height\_1 >= 0 and vri\_live\_stems\_per\_ha >= 0 and quad\_diam\_125 >= 0 and UPPER(species\_cd\_1) IN ", x$sql ,";")))  
 preds\_euc\_k1<-impute(yai(x=ref\_vri[,c("basal\_area", "tvol", "height", "tph", "qmd")], y=ref\_vri$crown\_closure, method="euclidean", k=1), k=1, method = 'closest')  
 preds\_mal\_k1<-impute(yai(x=ref\_vri[,c("basal\_area", "tvol","height", "tph", "qmd")], y=ref\_vri$crown\_closure, method="mahalanobis", k=1), k=1, method = 'closest')  
 #preds\_rf\_k1<-impute(yai(x=ref\_vri[,c("basal\_area", "height", "tph")], y=ref\_vri$crown\_closure, method="randomForest", k=1), k=1, method = 'closest')  
 preds\_euc\_k3<-impute(yai(x=ref\_vri[,c("basal\_area", "tvol", "height", "tph", "qmd")], y=ref\_vri$crown\_closure, method="euclidean", k=3), k=3, method = 'mean')  
 preds\_mal\_k3<-impute(yai(x=ref\_vri[,c("basal\_area", "tvol","height", "tph", "qmd")], y=ref\_vri$crown\_closure, method="mahalanobis", k=3), k=3, method = 'mean')   
 #preds\_rf\_k3<-impute(yai(x=ref\_vri[,c("basal\_area", "height", "tph")], y=ref\_vri$crown\_closure, method="randomForest", k=3), k=3, method = 'dstWeighted')  
 preds\_euc\_k5<-impute(yai(x=ref\_vri[,c("basal\_area", "tvol","height", "tph", "qmd")], y=ref\_vri$crown\_closure, method="euclidean", k=5), k=5, method = 'mean')  
 preds\_mal\_k5<-impute(yai(x=ref\_vri[,c("basal\_area", "tvol","height", "tph", "qmd")], y=ref\_vri$crown\_closure, method="mahalanobis", k=5), k=5, method = 'mean')   
 #preds\_rf\_k5<-impute(yai(x=ref\_vri[,c("basal\_area", "height", "tph")], y=ref\_vri$crown\_closure, method="randomForest", k=5), k=5, method = 'dstWeighted')  
 preds\_euc\_k10<-impute(yai(x=ref\_vri[,c("basal\_area", "tvol", "height", "tph", "qmd")], y=ref\_vri$crown\_closure, method="euclidean", k=10), k=10, method = 'mean')  
 preds\_mal\_k10<-impute(yai(x=ref\_vri[,c("basal\_area", "tvol", "height", "tph", "qmd")], y=ref\_vri$crown\_closure, method="mahalanobis", k=10), k=10, method = 'mean')   
 preds\_euc\_k15<-impute(yai(x=ref\_vri[,c("basal\_area", "tvol","height", "tph", "qmd")], y=ref\_vri$crown\_closure, method="euclidean", k=15), k=15, method = 'mean')  
 preds\_mal\_k15<-impute(yai(x=ref\_vri[,c("basal\_area", "tvol", "height", "tph", "qmd")], y=ref\_vri$crown\_closure, method="mahalanobis", k=15), k=15, method = 'mean')   
 avg.y<-mean(preds\_euc\_k1$y.o)  
 data.table(species = x$species, null = sqrt(sum((preds\_euc\_k1$y.o-avg.y)\*\*2)/nrow(preds\_euc\_k1)),rmsd\_euc\_k1 = rmsd(na.omit(preds\_euc\_k1),scale= FALSE)["y",],rmsd\_mal\_k1 = rmsd(na.omit(preds\_mal\_k1),scale= FALSE)["y",],rmsd\_euc\_k3 = rmsd(na.omit(preds\_euc\_k3),scale= FALSE)["y",],rmsd\_mal\_k3 = rmsd(na.omit(preds\_mal\_k3),scale= FALSE)["y",],rmsd\_euc\_k5 = rmsd(na.omit(preds\_euc\_k5),scale= FALSE)["y",],rmsd\_mal\_k5 = rmsd(na.omit(preds\_mal\_k5),scale= FALSE)["y",],rmsd\_euc\_k10 = rmsd(na.omit(preds\_euc\_k10),scale= FALSE)["y",],rmsd\_mal\_k10 = rmsd(na.omit(preds\_mal\_k10),scale= FALSE)["y",],rmsd\_euc\_k15 = rmsd(na.omit(preds\_euc\_k15),scale= FALSE)["y",],rmsd\_mal\_k15 = rmsd(na.omit(preds\_mal\_k15),scale= FALSE)["y",])  
})  
  
out<-rbindlist(error)  
saveRDS(out, file = "cc\_imputation\_params.rds")  
}  
  
readRDS("cc\_imputation\_params.rds")

## species null rmsd\_euc\_k1 rmsd\_mal\_k1 rmsd\_euc\_k3 rmsd\_mal\_k3  
## 1: Dr 18.35882 12.530924 12.786013 11.559508 11.512777  
## 2: E 17.44330 12.623247 12.686727 10.991980 11.141997  
## 3: Ac 16.33087 12.137283 12.263349 10.648811 10.401447  
## 4: At 17.46142 10.738157 10.857941 9.431042 9.496163  
## 5: B 15.16124 8.860321 8.593722 7.747319 7.494991  
## 6: Cw 13.34496 9.304887 9.422810 8.247731 8.263206  
## 7: Yc 14.19715 7.350276 7.332991 6.400813 6.284852  
## 8: Fd 18.32138 8.652226 8.692546 7.813642 7.849296  
## 9: H 16.93710 9.904009 9.902751 8.773867 8.739065  
## 10: L 19.09534 8.839674 8.886711 7.606039 7.619123  
## 11: Mb 17.44242 10.917345 11.014187 9.781873 9.979744  
## 12: P 17.64685 9.479028 9.448109 8.538746 8.506473  
## 13: Sb 13.92801 9.140332 9.053885 8.192416 7.997315  
## 14: Ss 17.44297 12.203802 12.179127 10.845903 10.825103  
## 15: Se 13.77057 7.770808 7.658029 6.804064 6.590150  
## 16: Sw 14.54796 8.425589 8.439284 7.316636 7.295448  
## rmsd\_euc\_k5 rmsd\_mal\_k5 rmsd\_euc\_k10 rmsd\_mal\_k10 rmsd\_euc\_k15 rmsd\_mal\_k15  
## 1: 11.453813 11.502816 11.137682 11.214111 10.993671 11.217604  
## 2: 10.882558 10.870250 10.799738 10.823921 10.859974 10.870723  
## 3: 10.300490 10.178359 10.024881 9.993436 9.985906 9.959616  
## 4: 9.184179 9.271307 9.068217 9.143033 9.051363 9.120476  
## 5: 7.577964 7.299221 7.519041 7.219456 7.542825 7.236191  
## 6: 8.078461 8.092762 7.990562 8.013516 7.999914 8.033079  
## 7: 6.289858 6.168373 6.221248 6.122856 6.257198 6.153593  
## 8: 7.707951 7.724973 7.660387 7.663480 7.658305 7.661326  
## 9: 8.579409 8.531598 8.481418 8.432357 8.495879 8.438180  
## 10: 7.436678 7.433091 7.346427 7.296335 7.336685 7.309103  
## 11: 9.372996 9.682233 9.459281 9.527428 9.464654 9.502603  
## 12: 8.441897 8.386372 8.458659 8.384151 8.501808 8.426251  
## 13: 8.053592 7.857128 8.023875 7.834537 8.045802 7.873423  
## 14: 10.629110 10.443533 10.365197 10.396092 10.495888 10.462738  
## 15: 6.620756 6.377142 6.522030 6.242964 6.508561 6.220907  
## 16: 7.102900 7.063703 6.983388 6.931268 6.965103 6.906423

## Impute

Preform the imputation into the VDYP yield curves

if(FALSE){  
 conn<-DBI::dbConnect(dbDriver("PostgreSQL"), host=keyring::key\_get('dbhost', keyring = 'postgreSQL'), dbname = keyring::key\_get('dbname', keyring = 'postgreSQL'), port='5432' ,user=keyring::key\_get('dbuser', keyring = 'postgreSQL') ,password= keyring::key\_get('dbpass', keyring = 'postgreSQL'))  
   
 dbExecute(conn, "ALTER TABLE yc\_vdyp\_2020\_adj add column crownclosure integer default 0")  
 dbDisconnect(conn)  
}  
  
###Set the list needed for running a stratified imputation  
species\_imp<-data.table(species = c("('Dr')", "('E','Ea','Ep')", "('Ac')", "('At')", "('B','Bl','Bg','Ba')", "('Cw')","('Yc')", "('Fd')", "('H', 'Hm', 'Hw')", "('L', 'Lw', 'La', 'Lt')", "('Mb')", "('P', 'Pl', 'Pj', 'Py', 'Pw', 'Pa', 'Pf')", "('Sb')", "('Ss')","('Se')","('Sw', 'S')"), sql=c("('D','DG','DM','DR')","('E','EB','EE','ES','EW','EX','EXP','EXW','EA','EP')","('A','AC','ACB','ACT','AD','AX')","('AT')","('B','BB','BC','BM','BP','BL','BG','BA')", "('C','CW')","('Y','YC')","('F','FD','FDC','FDI')", "('H','HX','HXM', 'HM','HW')", "('L','LD','LS','LW','LA','LT')","('M','MB','ME','MN','MR','MS','MV')","('P','PM','PR','PS','PX', 'PL','PLC','PLI','PY','PW','PJ','PXJ','PF','PA')","('SB')","('SS')","('SE','SXE')","('SW', 'SXW','S','SA','SN','SX','SXB','SXL','SXS','SXX')"))  
strata\_imp<-lapply(seq\_len(nrow(species\_imp)), function(i) species\_imp[i])  
  
if(FALSE){  
#Run the imputation by strata  
imp\_results<-lapply(strata\_imp, function(x){  
 ref\_vri<-data.table(getTableQuery(paste0("SELECT crown\_closure, basal\_area as basalarea, proj\_age\_1 as age, quad\_diam\_125 as qmd, proj\_height\_1 as height, vri\_live\_stems\_per\_ha as tph, live\_stand\_volume\_125 as tvol, bec\_zone\_code, species\_cd\_1 FROM veg\_comp\_lyr\_r1\_poly2020 where bclcs\_level\_2 = 'T' and reference\_year >= 2012 and species\_cd\_1 is not null and crown\_closure >= 0 and basal\_area >= 0 and proj\_height\_1 >= 0 and vri\_live\_stems\_per\_ha >= 0 and quad\_diam\_125 >= 0 and UPPER(species\_cd\_1) IN ", x$sql)))  
  
 yc\_vdyp<-data.table(getTableQuery(paste0("SELECT ycid, yc\_grp, age, tvol, height, qmd, basalarea, tph from yc\_vdyp\_2020\_adj where yc\_grp in (select distinct(yc\_grp) from vdyp\_vri2020 where species\_cd\_1 IN ", x$species,") and basalarea >=0 and tph >= 0;")))  
  
 mal <- yai(x=ref\_vri[,c("basalarea", "tvol", "height", "tph")], y=ref\_vri$crown\_closure, method="mahalanobis", k=10, noRefs=TRUE)  
  
rownames(yc\_vdyp) <- paste0("test.",rownames(yc\_vdyp))  
m10 <- newtargets(mal , newdata=yc\_vdyp, k=10)  
yc\_vdyp$cc<-as.integer(impute(m10,k=10,method = 'dstWeighted', vars=yvars(m10))$y)  
yc\_vdyp[height<5,cc:=0 ] #Donpt include crown closure of small trees.  
   
conn<-DBI::dbConnect(dbDriver("PostgreSQL"), host=keyring::key\_get('dbhost', keyring = 'postgreSQL'), dbname = keyring::key\_get('dbname', keyring = 'postgreSQL'), port='5432' ,user=keyring::key\_get('dbuser', keyring = 'postgreSQL') ,password= keyring::key\_get('dbpass', keyring = 'postgreSQL'))  
  
#sql\_string <- "Update yc\_vdyp\_all set cc = %s where ycid = %s and age = %s;"  
#dbSendStatement(conn, paste(sprintf(sql\_string, yc\_vdyp[ycid==-23919,]$cc, yc\_vdyp[ycid==-23919,]$ycid, yc\_vdyp[ycid==-23919,]$age), collapse=""))  
  
dbWriteTable(conn, "pred\_cc\_impute", yc\_vdyp[,c("ycid", "age", "cc")], append=TRUE)  
   
dbDisconnect(conn)   
})  
}

## Set the yc table

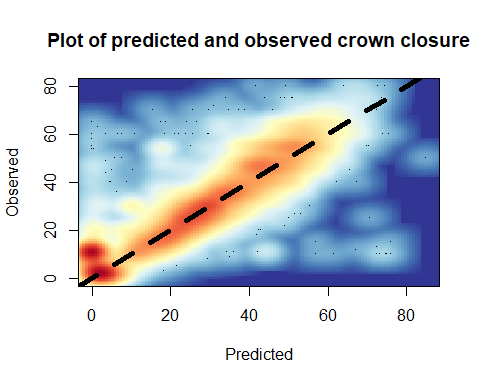
Merge the imputed cc with the yield curve table using feature\_id and age.

if(FALSE){  
conn<-DBI::dbConnect(dbDriver("PostgreSQL"), host=keyring::key\_get('dbhost', keyring = 'postgreSQL'), dbname = keyring::key\_get('dbname', keyring = 'postgreSQL'), port='5432' ,user=keyring::key\_get('dbuser', keyring = 'postgreSQL') ,password= keyring::key\_get('dbpass', keyring = 'postgreSQL'))  
dbExecute(conn, "UPDATE yc\_vdyp\_2020\_adj  
SET crownclosure = pred\_cc\_impute.cc  
FROM pred\_cc\_impute  
WHERE yc\_vdyp\_2020\_adj.ycid = pred\_cc\_impute.ycid and yc\_vdyp\_2020\_adj.age = pred\_cc\_impute.age;")  
dbDisconnect(conn)   
}

## Validation

Compare the predicted curves to the 2020 VRI.

comp\_vdyp\_vri<-data.table(getTableQuery(paste0("  
SELECT t.feature\_id, t.crown\_closure, k.crownclosure as upper, y.crownclosure as lower,  
 (((k.crownclosure - y.crownclosure \*1.0)/10)\*(t.age - CAST(t.age/10 AS INT)\*10))+ y.crownclosure as cc  
 FROM (SELECT crown\_closure, feature\_id, proj\_age\_1 as age FROM vdyp\_vri2020   
 where reference\_year >= 2019 and site\_index > 10 limit 10000) as t  
 LEFT JOIN yc\_vdyp\_2020\_adj y   
 ON t.feature\_id = (-1\*y.ycid) AND CAST(t.age/10 AS INT)\*10 = y.age  
 LEFT JOIN yc\_vdyp\_2020\_adj k   
 ON t.feature\_id = (-1\*k.ycid) AND round(t.age/10+0.5)\*10 = k.age WHERE t.age > 0 ;")))  
  
library(RColorBrewer)  
buylrd = c("#313695", "#4575B4", "#74ADD1", "#ABD9E9", "#E0F3F8", "#FFFFBF",  
 "#FEE090", "#FDAE61", "#F46D43", "#D73027", "#A50026")   
myColRamp = colorRampPalette(c(buylrd))  
  
# smoothed scatterplot  
smoothScatter(x=comp\_vdyp\_vri[!is.na(cc) & !is.na(crown\_closure),]$cc, y=comp\_vdyp\_vri[!is.na(cc) & !is.na(crown\_closure),]$crown\_closure,  
 colramp=myColRamp,  
 main="Plot of predicted and observed crown closure",  
 xlab="Predicted",  
 ylab="Observed")  
abline(0,1, lty = 2, lwd = 5, col = "black")



comp\_vdyp\_vri[, diff:=(cc-crown\_closure)\*\*2]  
sqrt(sum(na.omit(comp\_vdyp\_vri$diff))/nrow(na.omit(comp\_vdyp\_vri)))

## [1] 9.47436