Boreal Surface water inventory - technical documentation

ABMI Geospatial Centre

December, 2017

Contents

1 Introduction

The ABMI Geospatial Centre aims to provide spatially comprehensive and accurate Alberta-wide datasets to support the needs of the ABMI and other stakeholders. The primary dataset used for hydrography information in Alberta is the Government of Alberta Base Features Hydrography Polygons (GoA hydropolys) (Alberta Environment and Parks, formerly ESRD, 2004). While this dataset is comprehensive, it is older and may need updates in certain areas. Large scale government projects such as the Biodiversity Management Framework need a consistent up-to-date hydrographic dataset for calculations of their biodiversity indicators. Newly available satellite data in the form of Sentinel-1 and -2 (Copernicus, 2014) provide a great option for higher temporal (average six day revisit), and higher spatial resolution (10m) mapping of waterbodies in Alberta. Additionally, Google Earth Engine (GEE) provides a cloud based platform for quick and easy access and processing of Sentinel-1 and -2 data. These data can be utilized to support and update hydrographic spatial layers such as the GoA hydropolys.

2 Surface water classification model

2.1 Methods

2.1.1 Study area

The study area primarily consists of the Boreal Natural Region of Alberta with small parts of the foothills, parkland, and Canadian Shield included. This study area makes up about 60% (397, 958 km^2) of the total area of Alberta.

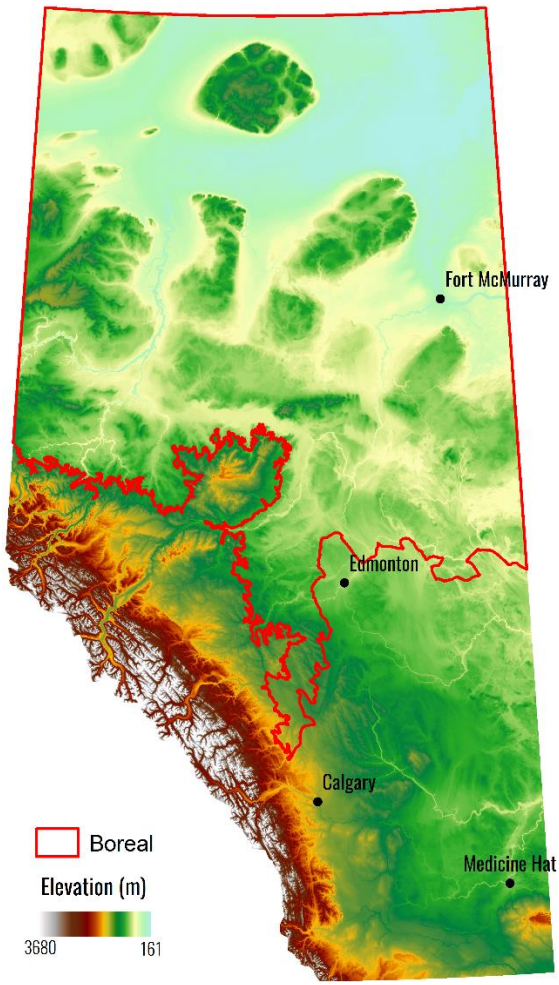

Figure 1: The spatial delineation of the boreal study region.

2.1.2 Data

Sentinel-1 and -2 (S1 and S2) (Copernicus [2016, 2017]) data was used to classify surface water in the boreal region. All data were acquired, processed, and downloaded through Google Earth Engine (GEE) (Google Earth Engine Team, 2015). GEE stores S1 (SAR imagery) ground range detected scenes which have been pre-processed with the Sentinel-1 Toolbox (Sentinel Application Platform – Sentinel-1 Toolbox). These pre-processing steps include thermal noise removal, radiometric calibration, and terrain correction (Google Earth Engine Team, 2015). S1 images were further processes in the GEE environment by performing an incidence angle correction (Gauthier *et al.,* 1998) and smoothing with a 3x3 Sigma Lee filter (Lee *et al.,* 2009) (credit to Guido Lemoine for GEE code). S2 top of atmosphere data was acquired through GEE.

All S1 images intersecting with the boreal region during the 2016 – 2017 spring/summer months (May-September) were used. Pixel vales were not used were removed from images if wind speed for the day of acquisition was over 12 km/h. Daily wind speed data was taken from the NCEP Climate Forecast System Version 2 (Saha *et al.*, 2014). Additionally, the normalized difference of polarization (NDPOL) was calculated for each image (see Table 1 for equation). The final VH and NDPOL grids were generated by

taking the mean value of the time series pixel stack. A total of 478 S1 images were used in the calculation of the VH and NDPOL variables.

The S2 (optical imagery) images intersecting with the boreal region during 2016-2017 leaf-on season (May 15 – August 31) were used to generate the Normalized Difference Water Index (NDWI) and Principal component 1 (PC1). Clouds, shadows, snow, and ice were removed with the QA60 band (a quality control band which flags snow, ice, and clouds), and further cloud masking was done using bands 1 (aerosols) and 11 (cloud identification). PC1 was generated with the 10m S2 bands (B2, B3, B4, and B8) in a principal component analysis. The merged S2 data was generated using a median composting algorithm where the median time series value for each pixel was selected as the most representative pixel. A total of 3,148 S2 images were used in the calculation of NDWI and PC1. All the input variables can be seen in Figure 2 and the equations and description can be seen in Table 1.

Training data was taken from the Alberta Biodiversity Monitoring Institute 3x7km Land Cover Photoplots (hereafter 3x7s) (ABMI, 2016). These photoplots are derived from high resolution 3D image interpretation and give detailed attribution of land cover information. They are typically very accurate with less than 1% of features possessing errors (ABMI, 2016).

Table 1: List of input variables used in the surface water classification

Variable	Data source	Equation	Description
VH	Sentinel-1	-	Vertical polarization sending horizontal polarization receiving SAR backscatter in decibels.
NDWI	Sentinel-2	$\frac{(Band\ 3 - Band\ 8)}{(Band\ 3 + Band\ 8)}$	Normalized difference Water Index from (Mcfeeters, 1996)
NDPOL	Sentinel-1	$\frac{(VH - VV)}{(VH + VV)}$	Normalized Difference of Polarization.
PC1	Sentinel-2	-	The first principal component of variation of Bands 2, 3, 4, and 8 of Sentinel-2 data

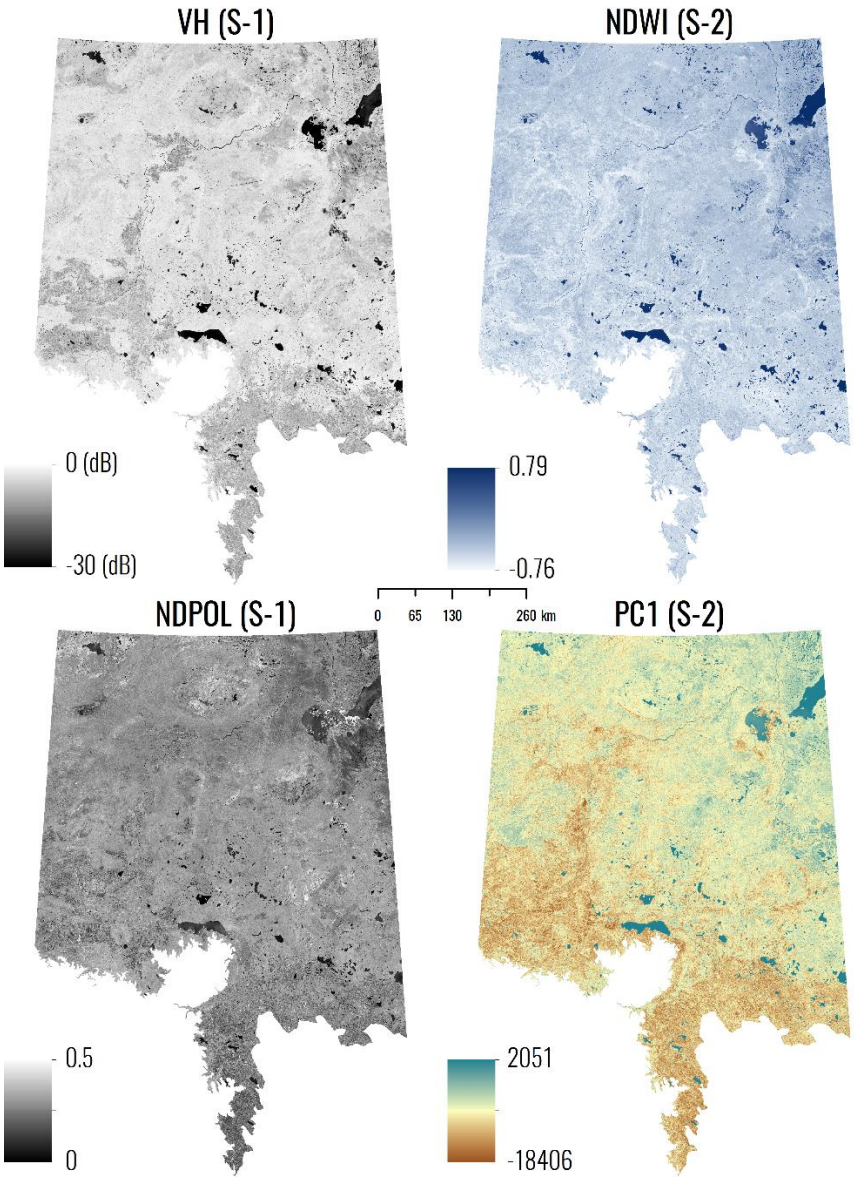

Figure 2: The four input raster layers used for the Boreal surface water classification. Two raster layers are generated from Sentinel-1 (S-1) data and two are generated from Sentinel-2 (S-2) data.

2.1.3 Data exploration

Figures 3 – 6 show the distribution of values for the four input variables for water and land classes. All variables show a strong distinction between water and land. Figure 7 shows the obvious clustering of water and land points when VH and NDWI are plotted against each other. As expected these variables are highly cross correlated and also highly correlated to water (Table 2). VH and NDWI show the highest correlation to water presence and are expected to be the most important variables in the water classification (Table 2).

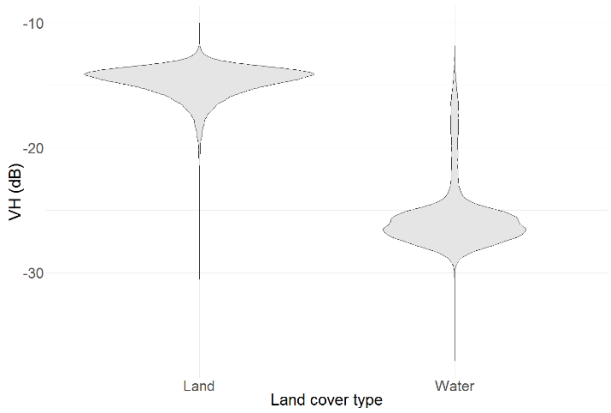

Figure 3: Distribution of VH values for water and land classes.

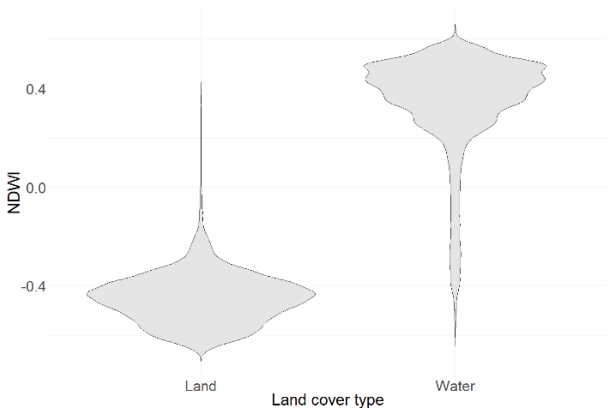

Figure 4: Distribution of NDWI values for water and land classes.

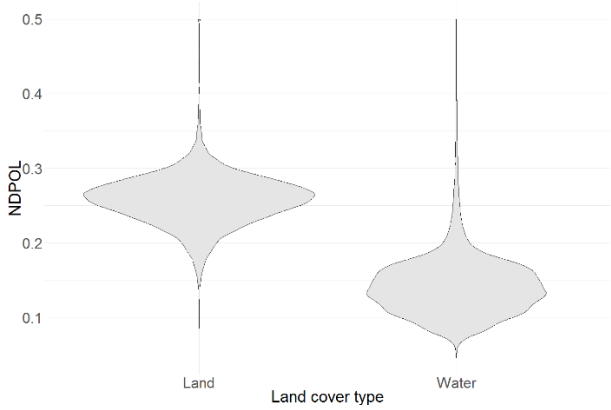

Figure 5: Distribution of NDPOL values for water and land classes.

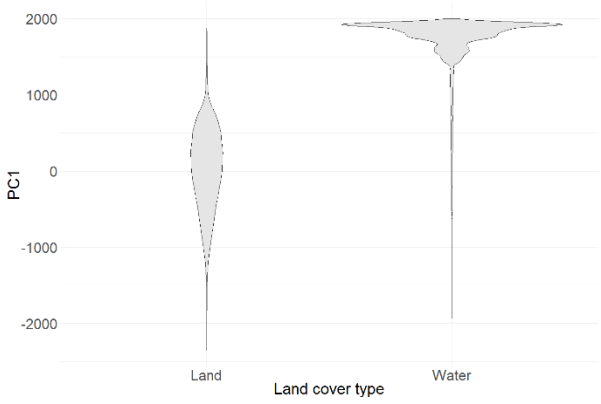

Figure 6: Distribution of PC1 values for water and land classes.

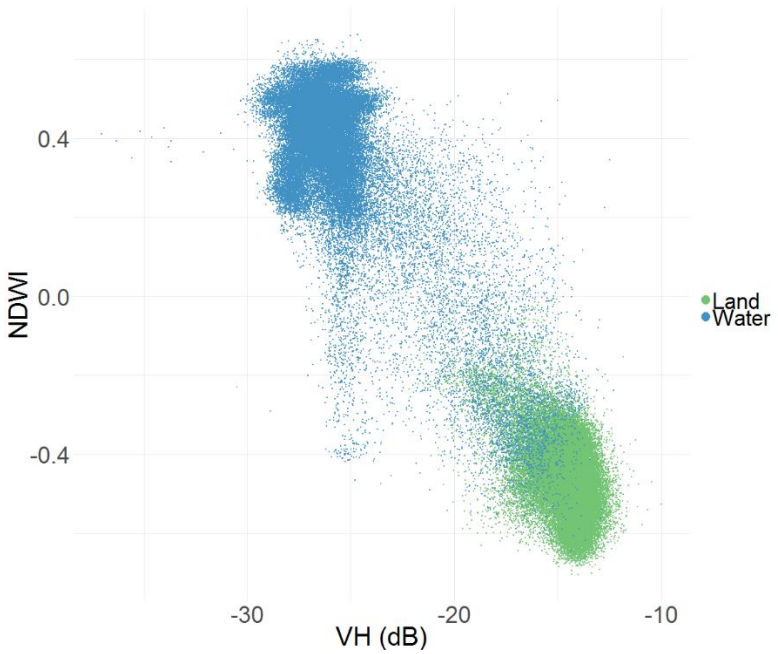

Figure 7: VH and NDWI values plotted for 100,000 random points colored by whether the point falls in water or land.

Table 2: Cross correlation between four input variables and a binary water/land grid.

	VH	NDWI	NDPOL	PC1	water
VH	1.00	-0.96	0.85	-0.89	-0.93
NDWI	-0.96	1.00	-0.84	0.95	0.92
NDPOL	0.85	-0.84	1.00	-0.79	-0.82
PC1	-0.89	0.95	-0.79	1.00	0.88
water	-0.93	0.92	-0.82	0.88	1.00

2.1.4 Water classification - machine learning algorithm

To classify water a machine learning algorithm was developed in R Statistical Software (R Core Team, 2013). This algorithm uses a boosted regression tree (BRT) modelling approach (Elith *et al.,* 2008). To build a model, 494 random points were placed at a distance of at least 1.5 kilometers apart in known land areas devoid of human footprint and 494 random points were placed 150 meters apart in known water areas. Training water/land was taken from the 3x7s. Using these 988 points a data frame was built describing the values of the four input variables and their binary water/land information. This data frame was then put into the boosted regression tree modelling function using a tree complexity of 5, learning rate of 0.005, and bag fraction of 0.5 as seen in Hird *et al.* (2017). This model output: responses for the four input variables, variables importance, and Area Under the Receiver Operating Characteristic Curve (AUROC) value. The model was then used to predict the probability of water given the four input variables. This process was repeated 40 times which generated 40 probability of water grids. This was done to reduce statistical overfitting and spatial auto-correlation (Parisien *et al.,* 2011). The mean value of these 40 grids was used to produce the final probability of water grid. Water was then classified as any value above 0.76 resulting in a binary water(1)/land(0) raster. The whole R script can be seen in Script 1.

```
#####################################################################
#---------------------------------------------------------------------
#---------------------------------------------------------------------
#Boreal Region Waterbody Classification
#Filename: "Boreal_WaterBodyClassification.R"
#Written and developed by Evan DeLancey - GIS Land Use Analyst
#Alberta Biodiversity Monitoring Institute, Sept, 7, 2017
#---------------------------------------------------------------------
#---------------------------------------------------------------------
#####################################################################

#Load libraries
library(raster)
library(rgdal)
library(ggplot2)
library(dplyr)
library(caret)
library(snow)
library(rgeos)
library(RPyGeo)
library(dismo)
library(gbm)
library(ggthemes)

tt1 <- Sys.time()

#location of input rasters
location <- "J:/LandCover/CurrentSurfaceWater/Boreal/processed"
#location of land and water shapefiles
location.WaterLand <- "J:/LandCover/CurrentSurfaceWater/Boreal/training"
#location of outputs
outputs <- "J:/LandCover/CurrentSurfaceWater/Boreal/OUTPUTS"
#version of the outputs
OutNum <- "_v1"
#enter number of iteration of subsampling
iter <- 40
#set your python location
py.loc <- "C:/Python27/ArcGIS10.4"
#Set minimum distance of sample water points
MinDist <- 1500

#set location of a temporary raster dump
#this can take up 100-300BG per run but is deleted after
rasterOptions(maxmemory = 1e+09,tmpdir = "J:/RtmpRasterDump")



#DEFINE FUNCTIONS
#####################################################################
#---------------------------------------------------------------------
#---------------------------------------------------------------------

#1) set up function to get random points give location of your land and water shapefile and the minimum distance
pts.gen <- function(directory, mindist){
   #set up ArcGIS environment
   w <- directory
   env <- rpygeo.build.env(python.path = py.loc, workspace = directory, overwrite=1)
   rpygeo.geoprocessor("CreateRandomPoints_management", c("", "ptsLand.shp", constraining_feature_class = "LandDissolve.shp",
      constraining_extent = "", number_of_points_or_field = 20000, minimum_allowed_distance = mindist), env = env, detect.required.extensions = T)
   ptsLand <- readOGR(w, "ptsLand")
   npts <- length(ptsLand)
   mdist <- mindist/10
   rpygeo.geoprocessor("CreateRandomPoints_management", c("", "ptsWater", constraining_feature_class = "LakesDissolve.shp",
      constraining_extent = "", number_of_points_or_field = npts, minimum_allowed_distance = mdist), env = env, detect.required.extensions = T)
   ptsWater <- readOGR(w, "ptsWater")
   pts.model <- rbind(ptsLand, ptsWater)
   return(pts.model)
}

#2)multiplot function
multiplot <- function(..., plotlist=NULL, file, cols=1, layout=NULL) {
  library(grid)

  # Make a list from the ... arguments and plotlist
  plots <- c(list(...), plotlist)

  numPlots = length(plots)

  # If layout is NULL, then use 'cols' to determine layout
  if (is.null(layout)) {
    # Make the panel
    # ncol: Number of columns of plots
    # nrow: Number of rows needed, calculated from # of cols
    layout <- matrix(seq(1, cols * ceiling(numPlots/cols)),
                     ncol = cols, nrow = ceiling(numPlots/cols))
  }
```

```
 if (numPlots==1) {
   print(plots[[1]])

 } else {
   # Set up the page
   grid.newpage()
   pushViewport(viewport(layout = grid.layout(nrow(layout), ncol(layout))))

   # Make each plot, in the correct location
   for (i in 1:numPlots) {
    # Get the i,j matrix positions of the regions that contain this subplot
    matchidx <- as.data.frame(which(layout == i, arr.ind = TRUE))

    print(plots[[i]], vp = viewport(layout.pos.row = matchidx$row,
                                    layout.pos.col = matchidx$col))
   }
 }
}
#------------------------------------------------------------
#------------------------------------------------------------
#################################################################



#Name vars and get min and max for response curves
#################################################################################
#------------------------------------------------------------
#------------------------------------------------------------
#set input varibles
water.tifs <- c("VH.tif", "NDWI.tif", "NDPOL.tif", "PC1.tif")
water.colnames <- c("VH", "NDWI", "NDPOL", "PC1", "water")
water.bricknames <- c("VH", "NDWI", "NDPOL", "PC1")
fls <- water.tifs

setwd(location)
#extract min and max of all input rasters for response curves
r <- raster(fls[1])
min1 <- cellStats(r, 'min')
max1 <- cellStats(r, 'max')
r <- raster(fls[2])
min2 <- cellStats(r, 'min')
max2 <- cellStats(r, 'max')
r <- raster(fls[3])
min3 <- cellStats(r, 'min')
max3 <- cellStats(r, 'max')
r <- raster(fls[4])
min4 <- cellStats(r, 'min')
max4 <- cellStats(r, 'max')
minval <- rbind(min1, min2, min3, min4)
maxval <- rbind(max1, max2, max3, max4)
mm.df <- data.frame(water.bricknames, minval, maxval)

#binary training raster
water <- raster("water.tif")
#------------------------------------------------------------
#------------------------------------------------------------
#################################################################



#BUILD MODEL 1
#################################################################################
#------------------------------------------------------------
#------------------------------------------------------------
pts.model <- pts.gen(location.WaterLand, MinDist)
dat <- data.frame(row=1:length(pts.model))
for (i in 1:length(fls)){
   r <- raster(fls[i])
   ext <- extract(r,pts.model)
   dat <- cbind(dat,ext)
   print(paste0("done ",fls[i]))
}
ext <- extract(water, pts.model)
dat <- cbind(dat,ext)
dat <- dat[,-1]
colnames(dat) <- water.colnames
dat <- na.omit(dat)
cor(dat)

#defint model list
fit <- list()
#build model
fit[[1]] <- gbm.step(dat, 1:length(fls), length(fls)+1, family = "bernoulli", tree.complexity = 5,
   learning.rate = 0.005, bag.fraction = 0.5, silent = TRUE, warnings = FALSE)
df.importance <- data.frame(summary(fit[[1]]))
df.importance <- arrange(df.importance, var)
df.importance <- df.importance[,2]
```

```
response.df <- data.frame(dummy=c(1:1001))
for (n in water.bricknames){
   d <- plot.gbm(fit[[1]], i.var = n, return.grid=TRUE, type="response")
   get.min.max <- filter(mm.df, water.bricknames == n)
   mn <- get.min.max[,2]
   mx <- get.min.max[,3]
   xout <- seq(mn, mx, (mx-mn)/1000)
   d <- approx(d[,1], d[,2], xout = xout, rule=2)
   d <- as.data.frame(d)
   response.df <- cbind(response.df,d)
}
#----------------------------------------------------------
#----------------------------------------------------------
#################################################################




#BUILD MODEL ALL MODELS AND OUTPUT STATS
#################################################################
#----------------------------------------------------------
#----------------------------------------------------------
setwd(location)

#define fit AUC and dev
AUC <- vector()
dev <- vector()
for (i in 2:iter){
   pts.model <- pts.gen(location.WaterLand, MinDist)
   dat <- data.frame(row=1:length(pts.model))
   for (j in 1:length(fls)){
      r <- raster(fls[j])
      ext <- extract(r,pts.model)
      dat <- cbind(dat,ext)
   }
   ext <- extract(water, pts.model)
   dat <- cbind(dat,ext)
   dat <- dat[,-1]
   colnames(dat) <- water.colnames
   dat <- na.omit(dat)

   #build model
   fit[[i]] <- gbm.step(dat, 1:length(fls), length(fls)+1, family = "bernoulli", tree.complexity = 5,
      learning.rate = 0.005, bag.fraction = 0.5, silent = TRUE, warnings = FALSE)
   v.importance <- as.data.frame(summary(fit[[i]]))
   v.importance <- arrange(v.importance, var)
   v.importance <- v.importance[,2]
   df.importance <- cbind(df.importance, v.importance)

   for (n in water.bricknames){
      d <- plot.gbm(fit[[i]], i.var = n, return.grid=TRUE, type="response")
      get.min.max <- filter(mm.df, water.bricknames == n)
      mn <- get.min.max[,2]
      mx <- get.min.max[,3]
      xout <- seq(mn, mx, (mx-mn)/1000)
      d <- approx(d[,1], d[,2], xout = xout, rule=2)
      d <- as.data.frame(d)
      response.df <- cbind(response.df,d)
   }

   #model stats
   AUC[i] <- fit[[i]]$cv.statistics$discrimination.mean
   dev[i] <- (fit[[i]]$self.statistics$mean.null - fit$self.statistics$mean.resid) / fit$self.statistics$mean.null

   print(paste0("done building model ", i))
}

importance <- rowMeans(df.importance)
imp.names <- c("VH", "NDWI", "NDPOL", "PC1")
importance <- data.frame(imp.names, importance)
#----------------------------------------------------------
#----------------------------------------------------------
#################################################################




#GENERATE RESPONSE CURVES AND MODEL STATS
#################################################################
#----------------------------------------------------------
#----------------------------------------------------------
response.df <- response.df[,-1]
response.df.x <- response.df[,seq(1,length(response.df),2)]
response.df.x <- response.df.x[,1:length(water.bricknames)]
response.df.y <- response.df[,seq(2,length(response.df),2)]
for (i in 1:length(water.bricknames)){
```

```
  varname <- water.bricknames[i]
  collumns <- seq(i, length(response.df.y), length(water.bricknames))
  yvals <- response.df.y[,collumns]
  xvals <- response.df.x[,i]
  yvals.mean <- rowMeans(yvals)
  yvals.std <- apply(yvals,1,sd)
  yvals.neg.std <- yvals.mean - yvals.std
  yvals.add.std <- yvals.mean + yvals.std
  yvals.df <- cbind(xvals, yvals.mean, yvals.neg.std, yvals.add.std)
  yvals.df <- as.data.frame(yvals.df)
  if(i>0){
    xlim <- c(min(xvals), max(xvals))
  } else{
    xlim <- c(min(xvals), 1200)
  }
  g <- ggplot(yvals.df, aes(x=xvals, y=yvals.mean)) +
    theme_minimal()+
    geom_ribbon(aes(ymin=yvals.neg.std, ymax=yvals.add.std), fill="gray80") +
    geom_line(colour="limegreen", size=1.6) +
    xlab(varname) + ylab("predicted probability") + xlim(xlim) +
    theme(axis.title.x = element_text(size=22), axis.title.y = element_text(size=20),axis.text = element_text(size=16))
  assign(paste0(varname,".plot"), g)
}

#output model stats
setwd(outputs)
AUC <- mean(AUC)
dev <- mean(dev)
stats <- cbind(AUC,dev)
write.csv(stats, paste0("WaterModelStats", OutNum, ".csv"), row.names=FALSE)

png(paste0("WaterResponseCurves", OutNum, ".png"), width = 1300, height = 1100)
multiplot(VH.plot, NDWI.plot, NDPOL.plot, PC1.plot, cols=2)
dev.off()

png(paste0("WaterVarImportance", OutNum, ".png"), width = 1000, height = 900)
ggplot(importance,aes(x=reorder(imp.names,-importance),y=importance, fill=importance)) + geom_bar(stat="identity") + theme_minimal() +
  theme(axis.title.x = element_text(size=24), axis.title.y = element_text(size=24),axis.text = element_text(size=22), legend.text = element_text(size=20), legend.title =
element_text(size=22)) +
  labs(x = "Variables") + labs(y = "Importance" )
dev.off()
#-----------------------------------------------------------
#-----------------------------------------------------------
#####################################################################



#####################################################################
#SECTION 2 PREDICT WATERBODIES BY NTS TILE
#####################################################################

#LOOP THROUGH PREDICTION OF RASTERS BASED ON MODEL FITS
#####################################################################
#-----------------------------------------------------------
#-----------------------------------------------------------

PUs <- readOGR("J:/LandCover/CurrentSurfaceWater/Boreal", "Boreal_PUs")
for (i in 1:length(Pus)){

  t1 <- Sys.time()

  #build raster brick
  setwd(location)
  fls <- water.tifs
  PU <- PUs[i,]
  #build raster brick
  r1 <- raster(fls[1])
  r1 <- crop(r1, PU)

  r2 <- raster(fls[2])
  r2 <- crop(r2, PU)

  r3 <- raster(fls[3])
  r3 <- crop(r3, PU)

  r4 <- raster(fls[4])
  r4 <- crop(r4, PU)


  r.b <- brick(r1,r2,r3,r4)
  names(r.b) <- water.bricknames

  #predict fit[[1]]
  beginCluster(15)
  r.b.p <- clusterR(r.b, raster::predict, args = list(model = fit[[1]], type = "response", n.trees = fit[[1]]$gbm.call$best.trees))
  endCluster()
  plot(r.b.p)

  #START PREDICTION OF RASTER STACK
```

```
    for (n in 2:length(fit)){
      beginCluster(15)
      prediction <- clusterR(r.b, raster::predict, args = list(model = fit[[n]], type = "response", n.trees = fit[[n]]$gbm.call$best.trees))
      endCluster()
      r.b.p <- stack(r.b.p, prediction)
      print(paste0("Done model prediction ", n))
    }

    wet <- calc(r.b.p, fun = mean)

    #Save water prediction rasters
    setwd(outputs)
    writeRaster(wet, paste0("WaterProbability", i, "_", OutNum, ".tif"), datatype= "FLT4S")

    t2 <- Sys.time()
    t.diff <- difftime(t2,t1, units="hours")
    print(paste0("Done predicting PU ", i, " it took ", round(t.diff, 2), " hours"))

}
#-----------------------------------------------------------------
#-----------------------------------------------------------------
######################################################################
tt2 <- Sys.time()
t.diff <- difftime(tt2,tt1, units="hours")
print(paste0("total script took ", round(t.diff, 2), " hours"))
```

Script 1: R machine learning algorithm for classifying water in the boreal region.

2.1.5 Cross validation accuracy assessment

An independent cross validation accuracy assessment of the binary water/land layer was completed by generating 200,000 points in 3x7 areas devoid of human footprint. Values from the 3x7 training data and the modeled surface water were then extracted for each point. With this data frame, a traditional accuracy was calculated along with a confusion matrix, and a kappa statistic.

2.2 Results

The results of the BRT model show that VH was the most important input variable in the model followed by PC1 and NDWI (Figure 8). NDPOL was seen to have minimal influence on the model (Figure 8). The response curves showed expected trends with low VH signal having high probabilities of being water, high NDWI and PC1 values having high probability of water (Figure 9).

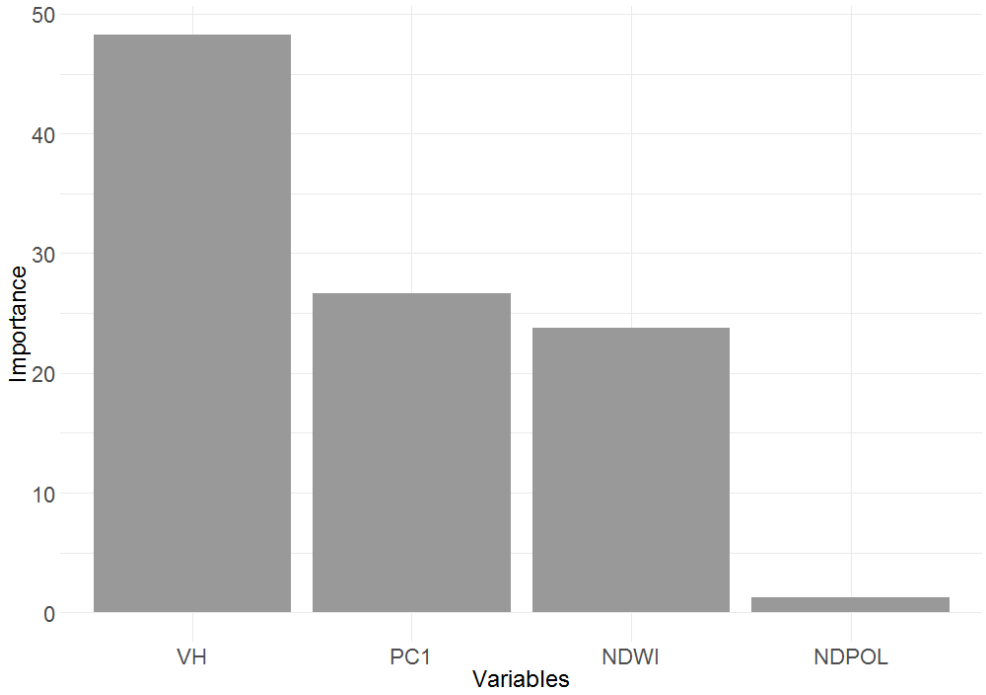

Figure 8: Importance of each of the response variables in the water classification BRT model.

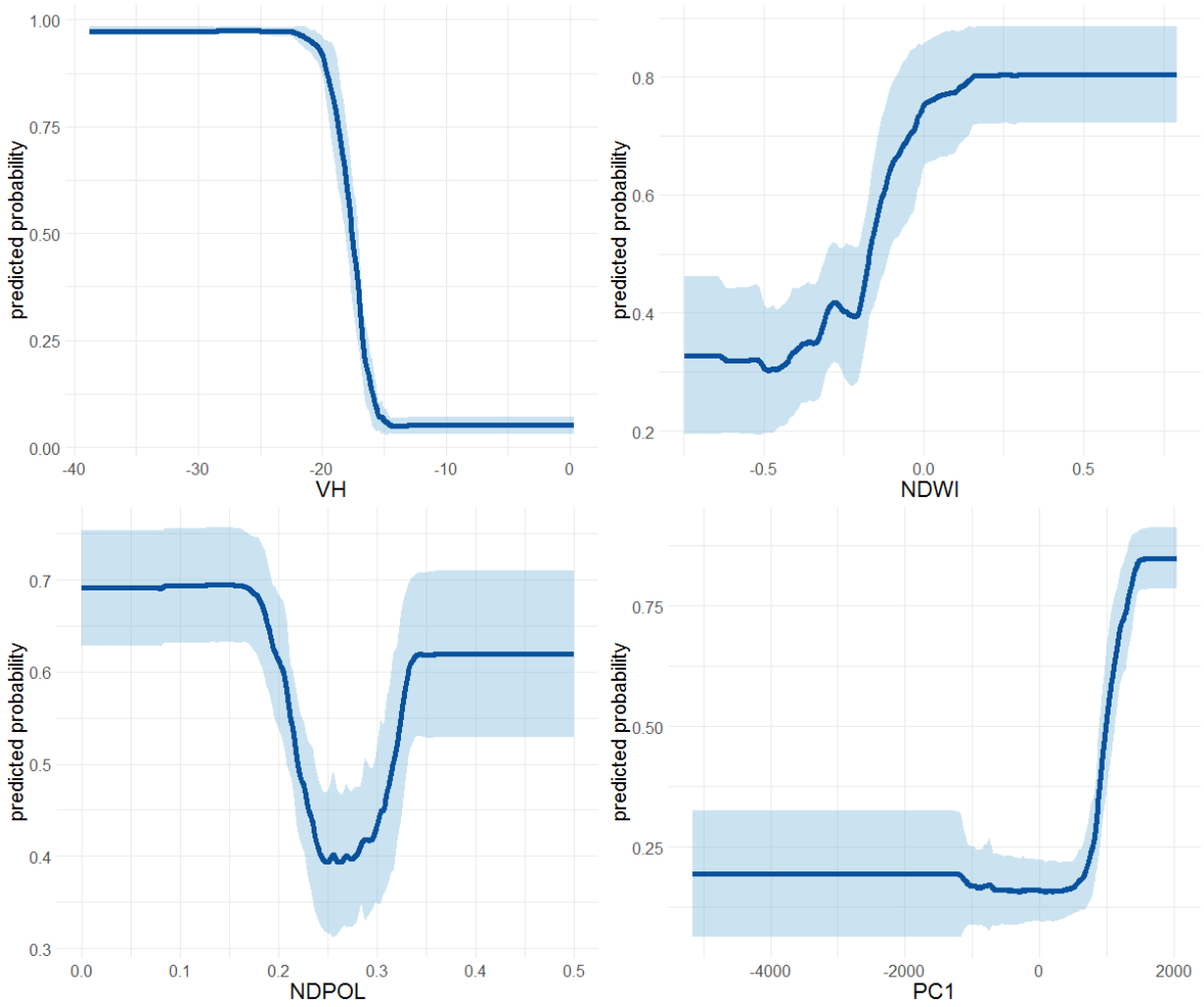

Figure 9: Average partial dependence response curves for the four input variables. Predicted probability represents the probability of a pixel being water.

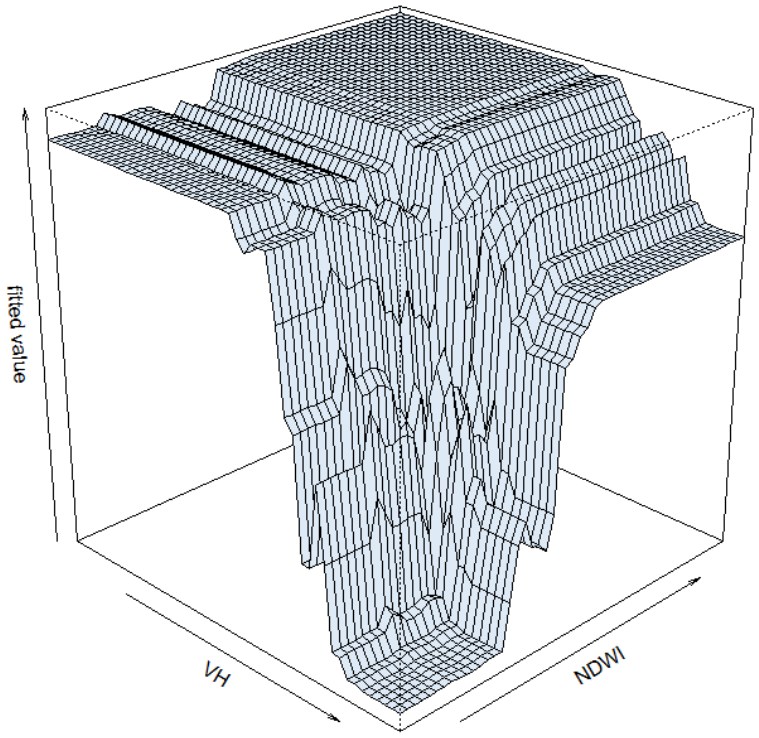

Figure 10: Two way partial dependence plots with VH and NDWI variables.

Figure 11a shows the results of the model prediction on the study area. It does a good job of distinguishing waterbodies and land. The major areas of concern are recent fires which show distinct areas of higher probability of water. Figure 11b shows the land and water classes across the landscape using a threshold of 0.76 which was deemed as the threshold with the best accuracy. This binary land/water results is seen to have a 99.5% accuracy to the training data and a 0.92 kappa statistic when compared to the 3x7 Photoplot data. Again, most false water areas occur in areas of recent high severity burn fires. The full list of results from the cross validation accuracy assessment can be seen in Table 2 and 3.

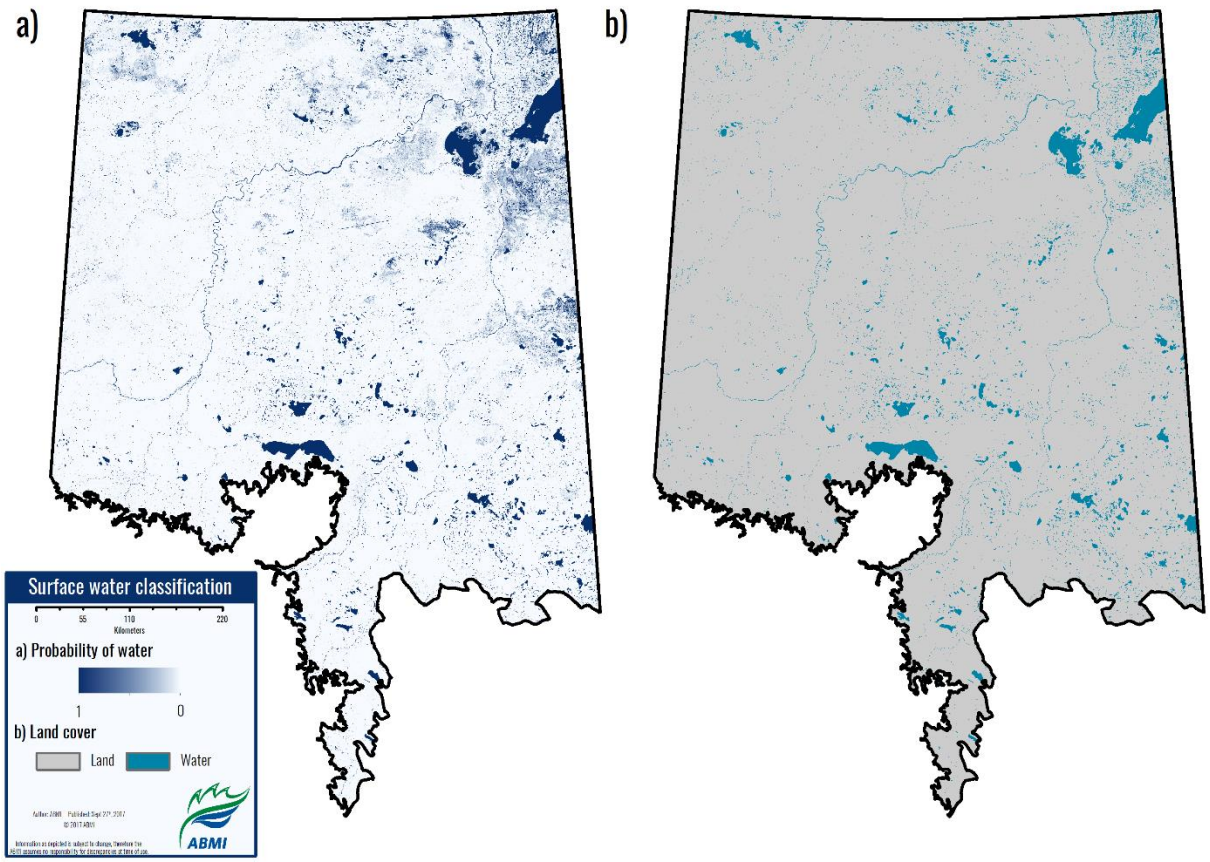

Figure 11: a) Probability of water for the boreal region with deep blue being the highest probability of water and light blue being low probability. b) the probability of water classified into water and land based on a 0.76 threshold.

Table 2: Confusion matric for the cross validation accuracy assessment to the 3x7s. Overall accuracy shown in the bottom right in bold.

	Land	Water	User accuracy
Land	81519	272	0.997
Water	198	4290	0.956
Producer accuracy	0.998	0.940	**0.995**

Table 3: Kappa statistic and AUROC of model and results

Kappa statistics cross validation accuracy assessment	0.945
AUROC of boosted regression tree model	0.98

2 Surface water quality control and cleaning

2.1 Methods

Once the binary water/land raster was generated it was turned into a polygon feature which was cleaned and quality checked (QC). The polygon layers were QCed with 1.5m resolution SPOT 2016 RBG data at a 1:30,000 scale. Polygons were marked as “CORRECT”, “INACCURATE”, or “INCORECT”. After double checking the “INCORECT” polygons, they were subsequently removed from the surface water inventory.

Some waterbodies were missed due the masking of the HFI cultivation layer. In cases where obvious waterbodies were missed we digitized these boundaries using the SPOT 2016 imagery. Additionally, some algal blooms caused holes in the middle of lakes, which were filled in with a digitized polygon if noticed in the cleaning and auditing processes. Finally, any polygon with an area less than 0.01 ha (1 10x10m pixel) was removed as this is likely a sliver polygon created by the masking of HF features.

2.2 Results

Most of the errors were either small polygons (2x2 or 1x1 10m pixels), areas of recent fires, or algal blooms. The cleaning process improved the overall accuracy by 0.13% and improved the kappa statistic by 0.007 (Table 4 and 5).

Table 4: Confusion matrix for the cross validation accuracy assessment to the 3x7 training data for the cleaned surface water polygons. Overall accuracy shown in the bottom right in bold.

	Land	Water	User accuracy
Land	93797	275	0.997
Water	136	4296	0.969
Producer accuracy	0.999	0.939	**0.996**

Table 5: Kappa statistic of cleaned polygons

Kappa statistics of cross validation accuracy assessment	0.952

About 26% of the errors were found to be in the Zama Lake region which looks to be undergoing rapid changes (e.g., possible draining). It is likely that the modeled results contain more accurate areas than the training data as the modeled results are the most up to date. It appears that Zama Lake was mostly full when the photo plots were completed. The overall accuracy is therefore likely closer to 99.7%.

3 Field attribution

3.1 Naming

The surface water polygons were spatially joined to named waterbodies from the Government of Alberta Base Features Hydrography Polygons (Alberta Environment and Parks, formerly ESRD, 2004). If a surface water polygon intersected with a named lake that polygon would take on the name. The Surface water polygons were then dissolved by the NAME field so named features such as large rivers would be one single feature.

3.2 Temporal attribution

The surface water polygons were used to extract the mean Hydro Temporal Variability (HTV) value for the polygons. The HTV dataset (DeLancey *et al.,* 2018) is a raster layer which summarizes the percent of time water is seen in a given 10x10m pixel. This can give an idea of whether the polygon is permanent or recurring. HTV field values of 0-60 will usually correspond with recurring lakes while 61-100 will often be permanent lakes. Based on the HTV field, a “PERMANENT” field was added with “YES” being HTV values from 61-100 and “NO” being HTV values of 0-60. To get an idea of the dynamic regions of lakes the HTV layer can underlay the surface water polygons as seen in Figure 12.

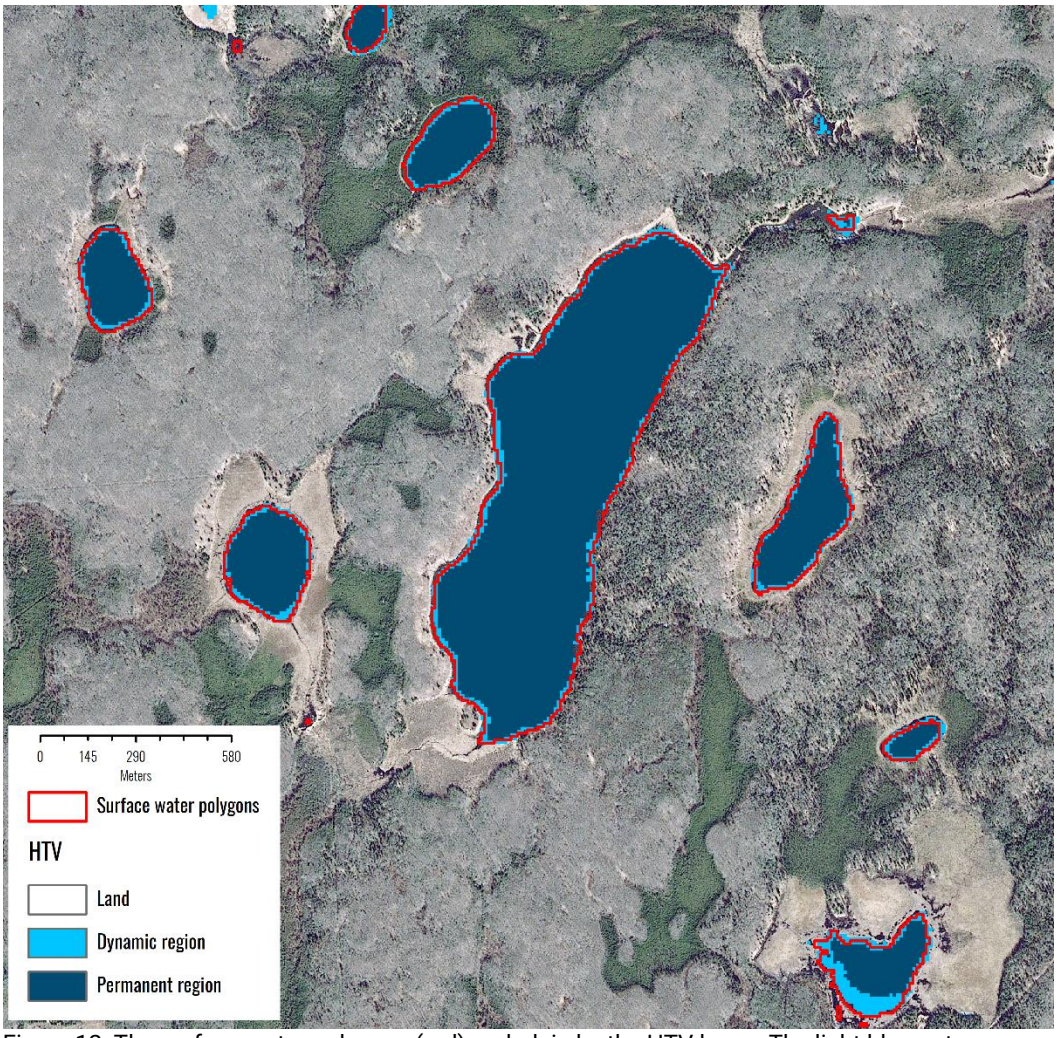

Figure 12: The surface water polygons (red) underlain by the HTV layer. The light blue outer regions of the lake represent the dynamic regions of the lake while the dark blue represents the permanent region. The light blue area can represent the normal seasonal fluctuation in lakes or the long term trend in water loss/gain.

3.3 Depth and volume attribution

For lakes over 10ha, average lake depth (field "AvgDepth") and volume (field "Volume") were taken from the WWF HydroSHEDS HydroLAKES version 1.0 layer (Messager *et al.,* 2016). The HydroLAKES polygons with volume and depth attribution were spatially joined to the Boreal surface water polygons. The results of this join can be seen in Figure 13. For more information on how lake depth and volume were calculated refer to Messager *et al.* (2016).

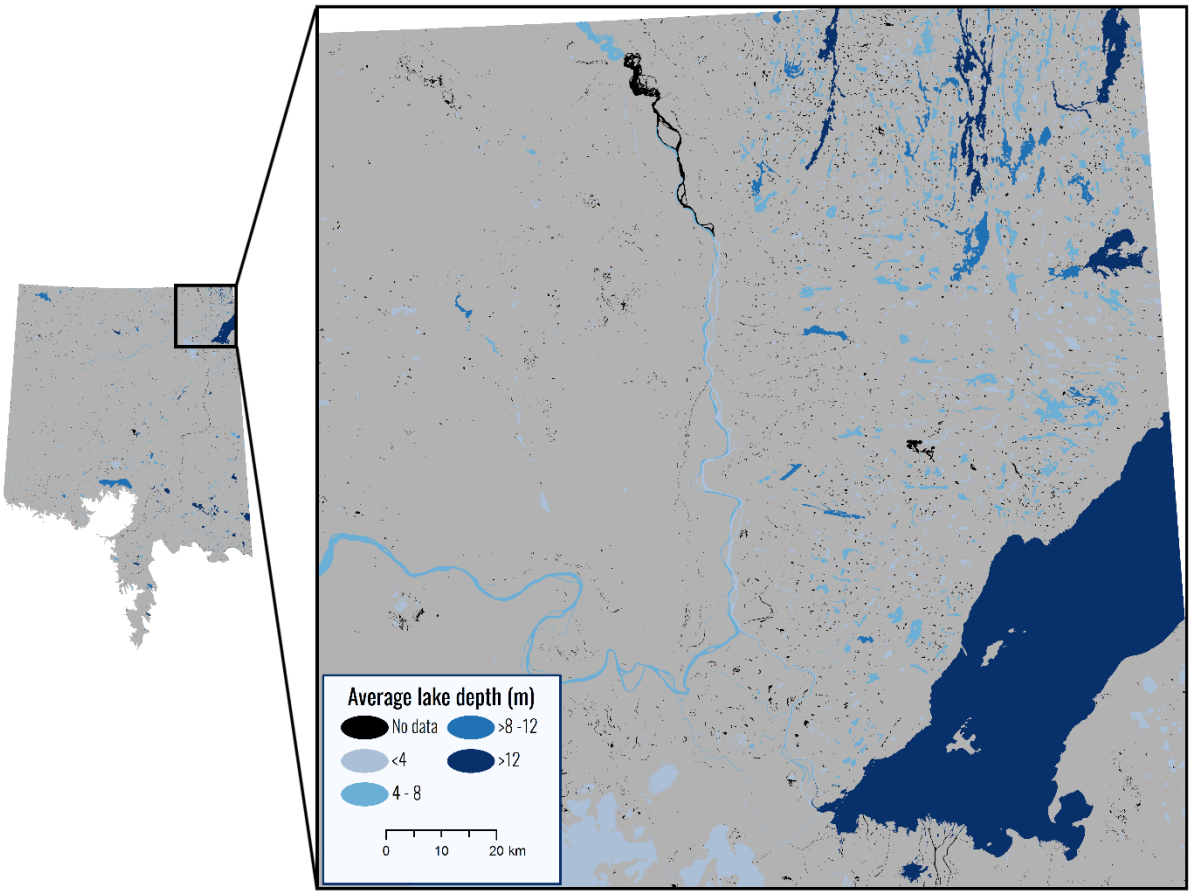

Figure 13: Average lake depth for lakes in the Canadian Shield Region of Alberta. Darker blues indicate greater average depth.

4 References

Alberta Biodiversity Monitoring Institute. 2016. “ABMI Photo-Plot Quality Control Manual.” Edmonton, Alberta.

Alberta Environment and Parks, Government of Alberta. 2004. “Base Water Polygon.” Edmonton, Alberta, Canada.

Copernicus Sentinel-1 data [2016, 2017], European Space Agency.

DeLancey, E.R., Kariyeva, J., Cranston, J., and Brisco, B. 2018. “Monitoring hydro temporal variability in Alberta, Canada with multi-temporal Sentinel-1 SAR data.” *Canadian Journal of Remote Sensing,* Vol. 44(No.1): pp.1-10.

Elith, J., Leathwick, J.R., and Hastie, T. 2008. “A working guide to boosted regression trees.” *Journal of Animal Ecology*, Vol. 77(No.4): pp. 802-813.

Gauthier, Y., Bernier, and Fortin, J-P. 1998. “Aspect and incidence angle sensitivity in ERS-1 SAR data.” *International Journal of Remote Sensing*, Vol. 19(No.10): pp. 2001-2006.

Google Earth Engine Team. 2015. “Google Earth Engine: A planetary-scale geospatial analysis platform.” https://eathengine.google.com.

Hird, J., DeLancey, E.R., McDermid, G.J., and Kariyeva, J. 2017. “Google Earth Engine, Open-Access Satellite Data, and Machine Learning in Support of Large-Area Probabilistic Wetland Mapping.” *Remote Sensing*, Vol. 9(No.12): pp. 1315.

Lee, J-S., Wen, J-H., Ainsworth, T.L., Chen, K-S., and Chen, A.J. 2008. “Improved Sigma Filter for Speckle Filtering of SAR Imagery.” *IEEE Transactions on Geosciences and Remote Sensing*, Vol. 47(No.1): pp. 202-213.

McFeeters, S.K. 1996. “The use of the Normalized Difference Water Index (NDWI) in the delineation of open water features.” *Remote Sensing Letters,* Vol. 17(No.7): pp. 1425-1432.

Messager, M.L., Lehner, B. Grill, G., Nedeva, I, and Schmitt, O. 2016. “Estimating the volume and age of water stored in global lakes using a geo-statistical approach.” *Nature Communications,* Vol. 7: pp. 1-11.

Parisien, M.A., Parks, S.A., Krawchuk, M.A., Flannigan, M.D., Bowman, L.M., Moritz, M.A. 2011. “Scale-dependent controls on the area burned in the boreal forest of Canada, 1980-2005.” *Ecological Application*, Vol. 21: pp. 789-805.

R Core Team. 2013. “R: A language and environment for statistical computing.” R Foundation for Statistical Computing, Vienna, Austria. URL http://www.R-project.org/.

Saha, S., Moorthi, S., Wu, X., Wang, J., Nagdiga, S., Tripp, P., Behringer, D., Chuang, H-Y., Iredell, M., Ek, M., Yang, R., Mendez, M.P., Dool, H.v.d., Zhang, Q., Wang, W., Chen, M., and Becker, E. 2014. “The NCEP Climate Forecast System Version 2.” *Journal of Climate*, Vol. 27: pp. 2185-2208.