AUC Analysis 11/24/2020

Code that accompanies final paper. Code to run Ech2o-SPAC is omitted, with spatial outputs of maximum LST simply imported here.
(Note: I had planned to write this in R, which I am more fluent with, and convert it to Python later. That ended up being a more time-consuming process that I thought, and I didn't have the time to do so, I apologize.)

Set up workspace

```r
library(raster)
```

```
## Loading required package: sp
```

```r
library(rgdal)
```

```
## rgdal: version: 1.5-18, (SVN revision 1082)
## Geospatial Data Abstraction Library extensions to R successfully loaded
## Loaded GDAL runtime: GDAL 3.0.4, released 2020/01/28
## Path to GDAL shared files: /usr/share/gdal
## GDAL binary built with GEOS: TRUE
## Loaded PROJ runtime: Rel. 6.3.1, February 10th, 2020, [PJ_VERSION: 631]
## Path to PROJ shared files: /usr/share/proj
## Linking to sp version:1.4-4
## To mute warnings of possible GDAL/OSR exportToProj4() degradation,
## use options("rgdal_show_exportToProj4_warnings"="none") before loading rgdal.
```

```r
library(data.table)
```

```
##
## Attaching package: 'data.table'

## The following object is masked from 'package:raster':
##
##     shift
```

```r
library(stringr)
library(ROCR)
library(ggplot2)
library(RColorBrewer)

## Define projection for all rasters
albers <- '+proj=aea +lat_1=29.5 +lat_2=45.5 +lat_0=23 +lon_0=-96 +x_0=0 +y_0=0 +ellps=GRS80 +towgs84=0

## Universal theme for my plots
```

```r
mytheme <- theme_minimal() +
           theme(axis.title = element_text(size = 15, family='Serif'),
                 strip.text = element_text(size=10, family='Serif'),
                 axis.text = element_text(family='Serif'),
                 legend.text = element_text(size=10, family='Serif'),
                 legend.title = element_text(size=10, family='Serif'),
                 legend.key = element_rect(fill='white', color='white'),
                 panel.background = element_rect(fill='white'))
```

Load VCF data

```r
## VCF Rasters
vcffiles <- list.files('/home/rankrc/Thesis/data/VCF/ech2o_Zackinputs_20200928', full.names=T)
vcf_ras <- lapply(1:length(vcffiles), function(i){
  temp <- raster(read.asciigrid(vcffiles[i]))
  crs(temp) <- albers
  return(temp)
})
names(vcf_ras) <- gsub('vcf', 'huc', str_extract(vcffiles, 'vcf_[0-9]+_[0-9]+'))

## Turn VCF values into 1 (covered) or 0 (uncovered) for different classification thresholds
vcf_ras_classified_5pct <- lapply(1:length(vcf_ras), function(i)
  reclassify(x = vcf_ras[[i]],
             rcl = matrix(c(0,   0.05, 0,
                            0.05, 1, 1)))
)
names(vcf_ras_classified_5pct) <- names(vcf_ras)

vcf_ras_classified_10pct <- lapply(1:length(vcf_ras), function(i)
  reclassify(x = vcf_ras[[i]],
             rcl = matrix(c(0,   0.1, 0,
                            0.1, 1, 1)))
)
names(vcf_ras_classified_10pct) <- names(vcf_ras)

vcf_ras_classified_15pct <- lapply(1:length(vcf_ras), function(i)
  reclassify(x = vcf_ras[[i]],
             rcl = matrix(c(0,   0.15, 0,
                            0.15, 1, 1)))
)
names(vcf_ras_classified_15pct) <- names(vcf_ras)

vcf_ras_classified_20pct <- lapply(1:length(vcf_ras), function(i)
  reclassify(x = vcf_ras[[i]],
             rcl = matrix(c(0,   0.2, 0,
                            0.2, 1, 1)))
)
names(vcf_ras_classified_20pct) <- names(vcf_ras)

vcf_ras_classified_25pct <- lapply(1:length(vcf_ras), function(i)
  reclassify(x = vcf_ras[[i]],
             rcl = matrix(c(0,   0.25, 0,
                            0.25, 1, 1)))
)
names(vcf_ras_classified_25pct) <- names(vcf_ras)
```

Load Ech2o-SPAC maximum LST estimates

```
tskin_max_files <- list.files('/home/rankrc/Thesis/data/ech2o_outputs/maps_onerunforeachyear_withspin_20
tskin_max_files <- tskin_max_files[!grepl('tskin', tskin_max_files)]
tskin_maps_maximums <- lapply(1:length(tskin_max_files), function(i) raster(tskin_max_files[i]))
names(tskin_maps_maximums) <- str_extract(tskin_max_files, 'huc_[0-9]+_[0-9]+')
```

Convert Tskin and VCF rasters to data tables, matching up pixel positions

```
model_pred_dt_5pct <- do.call('rbind', lapply(1:length(tskin_maps_maximums), function(i)
  data.table(lst_max = values(tskin_maps_maximums[[i]]),
             vcf_cover = values(vcf_ras_classified_5pct[[which(names(vcf_ras_classified_5pct) == names(
             huc = substr(names(tskin_maps_maximums)[i], 5,12),
             year = substr(names(tskin_maps_maximums)[i], 14,17),
             cell_id = 1:ncell(tskin_maps_maximums[[i]]))
  ))
model_pred_dt_5pct <- model_pred_dt_5pct[!is.na(vcf_cover)]#[, total_cells := .N, by=c('huc', 'year')]

model_pred_dt_10pct <- do.call('rbind', lapply(1:length(tskin_maps_maximums), function(i)
  data.table(lst_max = values(tskin_maps_maximums[[i]]),
             vcf_cover = values(vcf_ras_classified_10pct[[which(names(vcf_ras_classified_10pct) == names
             huc = substr(names(tskin_maps_maximums)[i], 5,12),
             year = substr(names(tskin_maps_maximums)[i], 14,17),
             cell_id = 1:ncell(tskin_maps_maximums[[i]]))
))
model_pred_dt_10pct <- model_pred_dt_10pct[!is.na(vcf_cover)]

model_pred_dt_15pct <- do.call('rbind', lapply(1:length(tskin_maps_maximums), function(i)
  data.table(lst_max = values(tskin_maps_maximums[[i]]),
             vcf_cover = values(vcf_ras_classified_15pct[[which(names(vcf_ras_classified_15pct) == names
             huc = substr(names(tskin_maps_maximums)[i], 5,12),
             year = substr(names(tskin_maps_maximums)[i], 14,17),
             cell_id = 1:ncell(tskin_maps_maximums[[i]]))
))
model_pred_dt_15pct <- model_pred_dt_15pct[!is.na(vcf_cover)]

model_pred_dt_20pct <- do.call('rbind', lapply(1:length(tskin_maps_maximums), function(i)
  data.table(lst_max = values(tskin_maps_maximums[[i]]),
             vcf_cover = values(vcf_ras_classified_20pct[[which(names(vcf_ras_classified_20pct) == names
             huc = substr(names(tskin_maps_maximums)[i], 5,12),
             year = substr(names(tskin_maps_maximums)[i], 14,17),
             cell_id = 1:ncell(tskin_maps_maximums[[i]]))
))
model_pred_dt_20pct <- model_pred_dt_20pct[!is.na(vcf_cover)]

model_pred_dt_25pct <- do.call('rbind', lapply(1:length(tskin_maps_maximums), function(i)
  data.table(lst_max = values(tskin_maps_maximums[[i]]),
             vcf_cover = values(vcf_ras_classified_25pct[[which(names(vcf_ras_classified_25pct) == names
             huc = substr(names(tskin_maps_maximums)[i], 5,12),
             year = substr(names(tskin_maps_maximums)[i], 14,17),
            cell_id = 1:ncell(tskin_maps_maximums[[i]]))
))
model_pred_dt_25pct <- model_pred_dt_25pct[!is.na(vcf_cover)]
```

Do some cleaning up: combine Ech2o/VCF tables into one and remove large rasters from my environment

```
model_pred_dt <- do.call('rbind', list(model_pred_dt_5pct[, threshold := '5%'], model_pred_dt_10pct[, th
                                       model_pred_dt_15pct[, threshold := '15%'], model_pred_dt_20pct[,
                                       model_pred_dt_25pct[, threshold := '25%']))

rm(list = c('vcf_ras', 'vcf_ras_classified_5pct', 'vcf_ras_classified_10pct', 'vcf_ras_classified_15pct
        'vcf_ras_classified_20pct', 'vcf_ras_classified_25pct', 'tskin_maps_maximums'))
```

Generate ROC curve and calculate AUC statistic for each basin and VCF threshold

```
hucs <- unique(model_pred_dt$huc)
thresholds <- unique(model_pred_dt$threshold)

## Generate list of ROC curves and AUC statistics
auc_stats_list <- lapply(1:length(hucs), function(i) # loop through each basin
  lapply(1:length(thresholds), function(j){ # loop through each VCF threshold

    x <- model_pred_dt[huc == hucs[i] & threshold == thresholds[j]]$lst_max
    y <- model_pred_dt[huc == hucs[i] & threshold == thresholds[j]]$vcf_cover

    if (all(y == y[1])){ # avoid case where all y belong to one class
      list(acc = NA,
           fracs = NA,
           auc = NA)
      }
    else {
      pred_rocr <- prediction(x, y)
      list(acc = performance(pred_rocr, 'acc'), # overall accuracy
           fracs = performance(pred_rocr, 'tpr', 'fpr'), # ROC curve
           auc = performance(pred_rocr, 'auc')) # AUC statistic
    }
  }))

## Extract AUC statistics from list and put in one data table
auc_table <- do.call('rbind', lapply(1:length(auc_stats_list), function(i)
  do.call('rbind', lapply(1:length(auc_stats_list[[i]]), function(j){

    if (all(!is.na(auc_stats_list[[i]][[j]]))){ # avoid case where all y belong to one class
      data.table(auc = as.numeric(slot(auc_stats_list[[i]][[j]]$auc, 'y.values')),
                 huc = hucs[i],
                 threshold = thresholds[j])
    }
    else {
      data.table(auc = NA,
                 huc = hucs[i],
                 threshold = thresholds[j])
    }
    }))
  ))

## Extract ROC curve values from list and put in one data table
roc_curves <- do.call('rbind', lapply(1:length(auc_stats_list), function(i)
  do.call('rbind', lapply(1:length(auc_stats_list[[i]]), function(j){

    if(all(!is.na(auc_stats_list[[i]][[j]]))){ # avoid case where all y belong to one class

      data.table(x = unlist(slot(auc_stats_list[[i]][[j]]$fracs, 'x.values')), # false positive rate
                 y = unlist(slot(auc_stats_list[[i]][[j]]$fracs, 'y.values')), # true positive rate
                 cutoff = unlist(slot(auc_stats_list[[i]][[j]]$fracs, 'alpha.values')), # LST cutoff va
                 huc = hucs[i],
                 threshold = thresholds[j])
      }
    else {
      data.table(x = NA,
                 y = NA,
                 cutoff = NA,
                 huc = hucs[i],
                 threshold = thresholds[j])
    }
    }))
  ))
```

Calculate Youden statistic for each basin and VCF threshold

```
## Calculate list of sensitivity and specificity
youden_stats_list <- lapply(1:length(hucs), function(i)
  lapply(1:length(thresholds), function(j){

    x <- model_pred_dt[huc == hucs[i] & threshold == thresholds[j]]$lst_max
    y <- model_pred_dt[huc == hucs[i] & threshold == thresholds[j]]$vcf_cover

    if (all(y == y[1])){ # avoid case where all y belong to one class
      list(youden_stats = NA)
      }
    else {
      pred_rocr <- prediction(x, y)
      list(youden_stats = performance(pred_rocr, 'sens', 'spec')) # sensitivity and specificity
    }
  }))

## Combine list of sensitivity and specificity into one data table
youden_stats_dt <- do.call('rbind', lapply(1:length(youden_stats_list), function(i)
  do.call('rbind', lapply(1:length(youden_stats_list[[i]]), function(j)

    if (all(!is.na(youden_stats_list[[i]][[j]]))){ # avoid case where all y belong to one class

      data.table(sensitivity = unlist(slot(youden_stats_list[[i]][[j]]$youden_stats, 'y.values')),
                 specificity = unlist(slot(youden_stats_list[[i]][[j]]$youden_stats, 'x.values')),
                 cutoff = unlist(slot(youden_stats_list[[i]][[j]]$youden_stats, 'alpha.values')),
                 huc = hucs[i],
                 threshold = thresholds[j])
```

```r
    }
    else {
      data.table(sensitivity = NA,
                 specificity = NA,
                 cutoff = NA,
                 huc = hucs[i],
                 threshold = thresholds[j])
    }
    ))
  ))

## Remove all cases where y values belonged to all one class
youden_stats_dt <- youden_stats_dt[!is.na(sensitivity)]

## Calculate youden index value and add to table
youden_stats_dt[, youden_index := sensitivity + specificity - 1][
  , max_youden_index := max(youden_index, na.rm=T), by=c('threshold', 'huc')]

## Find OCV (cutoff values that maximize Youden statistic)
optimal_cutoff_values <- youden_stats_dt[youden_index == max_youden_index]
```