Pick Your Flavor of Random Forest

Elizabeth A. Freeman, Tracey S. Frescino, Gretchen G. Moisen

September 10, 2024

Abstract

The **ModelMap** package (Freeman, 2009) for R (R Development Core Team, 2008) has added two additional variants of random forests: quantile regression forests and conditional inference forests. The **quantregForest** package (Meinshausen and Schiesser, 2015) is used for quantile regression forest (QRF) models. QRF models provide the ability to map the predicted median and individual quantiles. This makes it possible to map lower and upper bounds for the predictions without relying on the assumption that the predictions of individual trees in the model follow a normal distribution. The **party** package (Hothorn et al., 2006; Strobl et al., 2007, 2008) is used for conditional inference forest (CF) models. CF models offer two advantages over traditional RF models: they avoid RF's bias towards predictor variables with higher numbers of categories; and, they provide a conditional importance measure, allowing a better understanding of the relative importance of correlated predictor variables.

1 Introduction

The **ModelMap** package (Freeman, 2009) for R (R Development Core Team, 2008) has added two additional variants of random forests: quantile regression forests and conditional inference forests.

Quantile regression forest (QRF) models provide the ability to map the predicted median as well individual quantiles. This makes it possible to map lower and upper bounds for the predictions without relying on the assumption that the predictions of individual trees in the model follow a normal distribution. Mapping the predicted median rather than the predicted mean also may help avoid non-linearity of data leading to biased mean predictions (Anderson et al., 2014). The **quantregForest** package (Meinshausen and Schiesser, 2015) is used for QRF models.

Conditional inference forest CF models offer two advantages over traditional RF models: they avoid RF's bias towards predictor variables with higher numbers of categories; and, they provide a conditional importance measure, allowing a better understanding of the relative importance of correlated predictor variables. CF models avoid the potential bias found in RF variable importance measures when the predictors vary in their scale or their number of categories (Strobl et al., 2007). This potential bias in RF models is caused by the modeling process itself, where CART decision trees (Breiman et al., 1984) based on the Gini criterion favor predictors with more categories or larger scales (Strobl et al., 2007). In addition to the biases in the model building process, when the permutation importance is calculated, the permutation process favors correlated predictor variables (Strobl et al., 2008). CF models can avoid both these pitfalls, though one potential draw back is that CF models are very computer intensive, especially for large datasets. The **party** package (Hothorn et al., 2006; Strobl et al., 2007, 2008) is used for CF models.

Name	Type	Description
ELEV250	Continuous	90m NED elevation (ft) resampled to 250m, average of 49 points
NLCD01_250	Categorical	National Land Cover Dataset 2001 resampled to 250m - min. value of 49 points
EVI2005097	Continuous	MODIS Enhanced vegetation index
NDV2005097	Continuous	MODIS Normalized difference vegetation index
NIR2005097	Continuous	MODIS Band 2 (Near Infrared)
RED2005097	Continuous	MODIS Band 1 (Red)

Table 1: Predictor variables

2 Example dataset

The `model.explore` function can be used for both continuous and factored predictors, and for binary, categorical, and continuous responses. The out put graphics vary depending on the predictor and response types.

The data set used in this vignette is from a pilot study in Nevada launched in 2004 involving acquisition and photo-interpretation of large-scale aerial photography, the Nevada Photo-Based Inventory Pilot (NPIP) (Frescino et al., 2009). The data files for these examples are included in the **ModelMap** package installation in the R library directory. The data sets are under the 'external' then under 'vignetteexamples'.

The predictor data set consists of 6 predictor variables: 5 continuous variables, and 1 categorical variable (Table 1). The predictor layers are 250-meter resolution, pixel-based raster layers including Moderate Resolution Imaging Spectro-radiometer (MODIS) satellite imagery (Justice et al., 2002), a Landsat Thematic Mapper-based, thematic layer of predicted land cover, National Land Cover Data (NLCD) (Homer et al., 2004), and a topographic layer of elevation from the National Elevation Data (Gesch et al., 2002).

The continuous response variables are percent cover of Pinyon and Sage. The binary response variables are presence of Pinyon and Sage. The categorical response variable is the vegetation category: TREE, SHRUB, OTHERVEG, and NONVEG.

The MODIS data included 250-meter, 16-day, cloud-free, composites of MODIS imagery for April 6, 2005: visible-red (RED) and near-infrared (NIR) bands and 2 vegetation indices, normalized difference vegetation index (NDVI) and enhanced vegetation index (EVI) (Huete et al., 2002). The land cover and topographic layers were 30-meter products re-sampled to 250 meter using majority and mean summaries, respectively.

The rectangular subset of Nevada chosen for these maps was deliberately selected to lie along the diagonal edge of the study region to illustrate how **ModelMap** handles unsampled regions of a rectangle (Figure 1).

Load the **ModelMap** package.

```
R> library("ModelMap")

R> library(raster)
R> elevfn <- paste(getwd(),"/VModelMapData_dem_ELEVM_250.img",sep="")
R> mapgrid <- raster(elevfn)
```

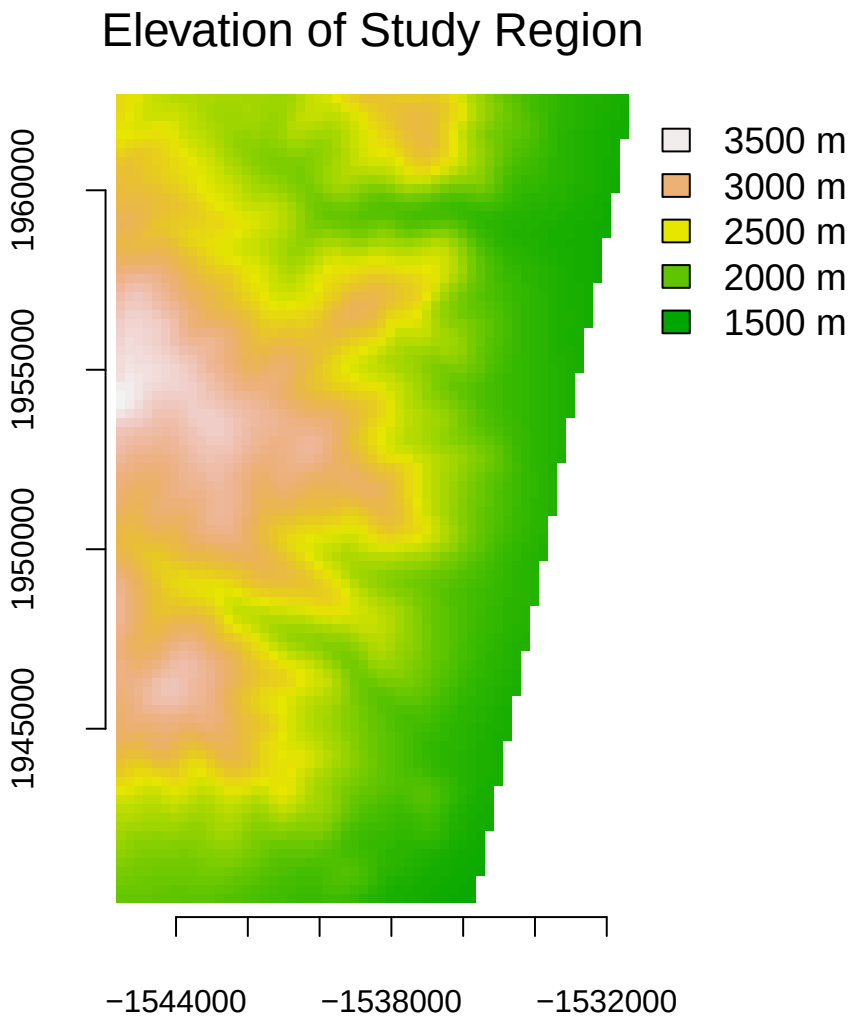

Figure 1: Elevation of the subset of the study region used for this vignette. This is a small area located along the southeast edge of nevada, containing a portion of a small mountain range. The Projection: Universal Transverse Mercator (UTM) Zone 11, Datum: NAD83

2.1 Set up

After installing the **ModelMap** package, find the sample data sets from the R installation and copy them to your working directory. The data consists of five files and is located in the vignette directory of **ModelMap**, for example, in `C:\R\R-2.15.0\library\ModelMap\vignettes`.

There are 5 files:

VModelMapData.csv
VModelMapData_LUT.csv
VModelMapData_dem_ELEVM_250.img
VModelMapData_modis_STK2005097.img
VModelMapData_nlcd_NLCD01_250.img

Next define some of the arguments.

Define training and test data file names. Note that the arguments `qdata.trainfn` and `qdata.testfn` will accept either character strings giving the file names of CSV files of data, or the data itself in the form of a data frame.

```
R> qdatafn <- "VModelMapData.csv"
R> qdata.trainfn <- "VModelMapData_TRAIN.csv"
R> qdata.testfn  <- "VModelMapData_TEST.csv"
```

Define the output folder.

```
R> folder <- getwd()
```

Split the data into training and test sets. In example 1, an independent test set is used for model validation diagnostics. The function `get.test()` randomly divides the original data into training and test sets. This function writes the training and test sets to the folder specified by `folder`, under the file names specified by `qdata.trainfn` and `qdata.testfn`. If the arguments `qdata.trainfn` and `qdata.testfn` are not included, file names will be generated by appending `"_train"` and `"_test"` to `qdatafn`.

```
R> get.test(       proportion.test=0.2,
                 qdatafn=qdatafn,
                 seed=42,
                 folder=folder,
                 qdata.trainfn=qdata.trainfn,
                 qdata.testfn=qdata.testfn)
```

Define the predictors and define which predictors are categorical. Example 1 uses five continuous predictors: the four predictor layers from the MODIS imagery plus the topographic elevation layer. As none of the chosen predictors are categorical set `predFactor` to `FALSE`.

```
R> predList <- c( "ELEV250",
                "NLCD01_250",
                "EVI2005097",
                "NDV2005097",
                "NIR2005097",
                "RED2005097")
R> predFactor <- c("NLCD01_250")
```

Define the column that contains unique identifiers for each data point. These identifiers will be used to label the output file of observed and predicted values when running model validation.

```
R> unique.rowname <- "ID"
```

Define raster look up table.

```
R> rastLUTfn     <- "VModelMapData_LUT.csv"
R> rastLUTfn     <- read.table( rastLUTfn,
                                header=FALSE,
                                sep=",",
                                stringsAsFactors=FALSE)
R> rastLUTfn[,1] <- paste(folder,rastLUTfn[,1],sep="/")
```

3 Example 1 - Quantile Regression Forest

3.1 Build Models

Also, when working with QRF models, **ModelMap** will create two models: the QRF model, and a RF model built with from the same training data and with the same model parameters. This allows simple comparison of predicted median and predicted mean. When `model.type="QRF"` the `model.build()` function returns these models as a list with two components. This list can be used as the `model.obj` argument to the other **ModelMap** functions. If you wish to use functions from outside **ModelMap** on these models you can extract the individual QRF and RF models by name. For example: `model.obj$QRF` and `model.obj$RF`.

```
R> MODELfn.pinyon    <- "VQuantile_QRF_Pinyon"
R> MODELfn.sage    <- "VQuantile_QRF_Sage"

R> response.name.pinyon <- "PINYON"
R> response.name.sage <- "SAGE"
R> response.type   <- "continuous"

R> QRF.pinyon <- model.build( model.type="QRF",
                              qdata.trainfn=qdata.trainfn,
                              folder=folder,
                              unique.rowname=unique.rowname,
                              MODELfn=MODELfn.pinyon,
                              predList=predList,
                              predFactor=predFactor,
                              response.name=response.name.pinyon,
                              response.type=response.type,

                              #importance=TRUE,
                              #quantiles=c(0.1,0.2,0.5,0.8,0.9)
 )
R> QRF.sage <- model.build( model.type="QRF",
                              qdata.trainfn=qdata.trainfn,
                              folder=folder,
                              unique.rowname=unique.rowname,
                              MODELfn=MODELfn.sage,
                              predList=predList,
                              predFactor=predFactor,
```

```
                                response.name=response.name.sage,
                                response.type=response.type,

                                #importance=TRUE,
                                #quantiles=c(0.1,0.2,0.5,0.8,0.9)
 )
```

3.2 Diagnostics

```
R> QRF.pinyon.pred <- model.diagnostics( model.obj=QRF.pinyon,
                                         qdata.testfn=qdata.testfn,
                                         folder=folder,
                                         MODELfn=MODELfn.pinyon,
                                         unique.rowname=unique.rowname,
                                         quantiles=c(0.1,0.5,0.9),
                                 # Model Validation Arguments
                                         prediction.type="TEST",
                                         device.type=c("pdf","png"),
                                         cex=1.2)
R> QRF.sage.pred <- model.diagnostics( model.obj=QRF.sage,
                                         qdata.testfn=qdata.testfn,
                                         folder=folder,
                                         MODELfn=MODELfn.sage,
                                         unique.rowname=unique.rowname,
                                         quantiles=c(0.1,0.5,0.9),
                                  # Model Validation Arguments
                                         prediction.type="TEST",
                                         device.type=c("pdf","png"),
                                         cex=1.2)
```

3.3 Importance Plots

Importance is currently unavailable for QRF models. It will be re-enabled when the quantregForest package is updated.

3.4 Interaction Plots

For QRF models, interaction plots can be generated for the median predicted value, as well as for individual quantiles.

The `model.interaction.plot()` function provides a diagnostic plot useful in visualizing two-way interactions between predictor variables. Two of the predictor variables from the model are used to produce a grid of possible combinations over the range of both variables. The remaining predictor variables are fixed at either their means (for continuous predictors) or their most common value (for categorical predictors). Model predictions are generated over this grid and plotted as the z axis. The `model.interaction.plot()` function was developed from the `gbm.perspec` function from the tutorial provided as appendix S3 in Elith et al. (2008).

The `model.interaction.plot()` function provides two graphical options: an image plot, and a 3-D perspective plot. These options are selected by setting `plot.type = "image"` or `plot.type = "persp"`. The `x` and `y` arguments are used to specify the predictor variables for

the X and Y axis. The predictors can be specified by name, or by number, with the numbers referring to the order the variables appear in `predList`.

The `pred.means` argument allows you to use a named vector to specify values for the predictors not being used as X and Y axis variables. In the example below, `pred.means` are set to values associated with high levels of Pinyon, as determined from the `model.explore()` plots. The reasoning behind this, is to see what effect the X and Y axis variables have on Pinyon cover, when the other predictors are at their ideal values (

```
R> pred.means <- c( ELEV250    = 2300,
                    NLCD01_250 = 42,
                    EVI2005097 = 1800,
                    NDV2005097 = 3100,
                    NIR2005097 = 2000,
                    RED2005097 = 1000)
R> opar <- par(mfrow=c(3,1),mar=c(2,3,0,2),oma=c(0,0,3,0))
R> for(quantile in c(0.1,0.5,0.9)){
      model.interaction.plot( QRF.pinyon$QRF,
                              x="ELEV250",
                              y="EVI2005097",
                              main="",
                              plot.type="persp",
                              device.type=c("none"),
                              MODELfn=MODELfn.pinyon,
                              folder=paste(folder,"/interaction",sep=""),
                              quantile=quantile,
                              pred.means=pred.means,
                              zlim=c(0,60))
      mtext(paste( quantile*100,"% Quantile",sep=""),side=2,line=1,font=2,cex=1)
 }
R> mtext("Pinyon Cover by Quantile",side=3,line=1,cex=1.8,outer=TRUE)
R> par(opar)
R>
```

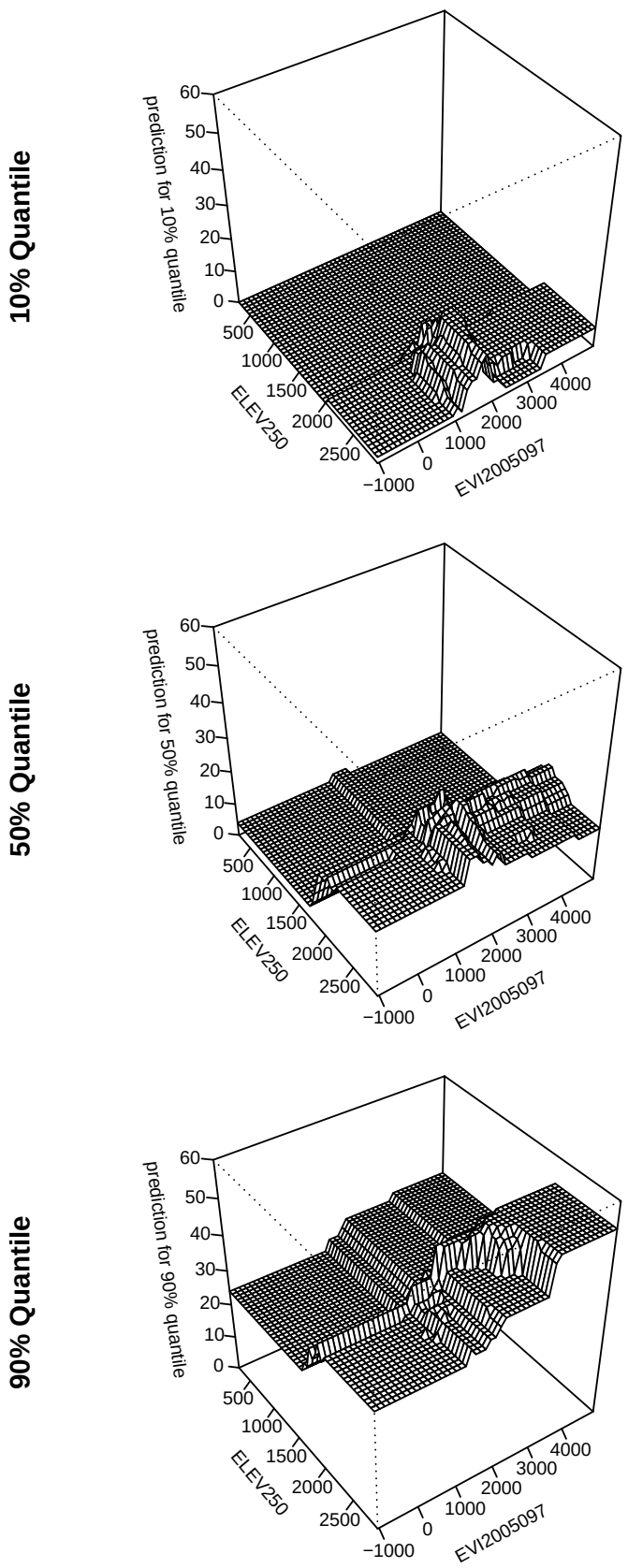

Figure 2: Pinyon Percent Cover - Interaction between elevation and EVI with other predictors fixed at optimum values for Pinyon.

3.5 Map production

3.5.1 Raster Lookup Table

The `model.mapmake()` uses a look up table to associate the predictor variables with the rasters. The function argument `rastLUTfn` will accept either a file name of the CSV file containing the table, or the data frame itself.

Although in typical user applications the raster look up table must include the full path for predictor rasters, the table provided for the examples will be incomplete when initially downloaded, as the working directory of the user is unknown and will be different on every computer. This needs to be corrected by pasting the full paths to the user's working directory to the first column, using the value from `folder` defined above.

```
R> rastLUTfn     <- "VModelMapData_LUT.csv"
R> rastLUTfn     <- read.table(  rastLUTfn,
                                header=FALSE,
                                sep=",",
                                stringsAsFactors=FALSE)
R> rastLUTfn[,1] <- paste(folder,rastLUTfn[,1],sep="/")
```

3.5.2 Predict Over the Rasters

```
R> model.mapmake(  model.obj=QRF.pinyon,
                   folder=folder,
                   MODELfn=MODELfn.pinyon,
                   rastLUTfn=rastLUTfn,
                   na.action="na.omit",
                # Mapping arguments
                   map.sd=TRUE,
                   quantiles=c(0.1,0.5,0.9))
R> model.mapmake(  model.obj=QRF.sage,
                   folder=folder,
                   MODELfn=MODELfn.sage,
                   rastLUTfn=rastLUTfn,
                   na.action="na.omit",
                # Mapping arguments
                   map.sd=TRUE,
                   quantiles=c(0.1,0.5,0.9))
```

3.5.3 RF Map

When building QRF models, **ModelMap** creates both a QRF model and an ordinary RF model. We will start by comparing the maps of mean predicted canopy cover from the RF model (Figure 3) with those of median predicted canopy cover from teh QRF model (Figure 4).

From the RF and the QRF maps, we can see that Pinyon percent cover is higher in the mountains, while Sage percent cover is higher in the foothills at the edges of the mountains.

Note that the sample map data was taken from the South Eastern edge of our study region, to illustrate how **ModelMap** deals with portions of the rectangle that fall outside of the study region. The empty wedge at lower right in the maps is the portion outside the study area. **ModelMap** uses `-9999` for unsampled data. When viewing maps in a GIS, a mask file can be used to hide unsampled regions, or other commands can be used to set the color for `-9999` values.

Since we know that percent cover can not be negative, we will set `zlim` to range from zero to the maximum value found in our map and define a color ramp such that zero values will display as white, shading to dark green for high values of cover.

```
R> l <- seq(100,0,length.out=101)
R> c <- seq(0,100,length.out=101)
R> col.ramp <- hcl(h = 120, c = c, l = l)

R> opar <- par(mfrow=c(1,2),mar=c(3,3,2,1),oma=c(0,0,3,4),xpd=NA)
R> mapgrid.pinyon <- raster(paste(MODELfn.pinyon,"_map_RF.img",sep=""))
R> mapgrid.sage <- raster(paste(MODELfn.sage,"_map_RF.img",sep=""))
R> zlim <- c(0,60)
R> legend.label<-rev(pretty(zlim,n=5))
R> legend.colors<-col.ramp[trunc((legend.label/max(legend.label))*100)+1]
R> legend.label<-paste(legend.label,"%",sep="")
R> image( mapgrid.pinyon,
        col=col.ramp,
        xlab="",ylab="",xaxt="n",yaxt="n",
        zlim=zlim,
        asp=1,bty="n",main="")
R> mtext(response.name.pinyon,side=3,line=1,cex=1.2)
R> image( mapgrid.sage,
        col=col.ramp,
        xlab="",ylab="",xaxt="n",yaxt="n",
        zlim=zlim,
        asp=1,bty="n",main="")
R> mtext(response.name.sage,side=3,line=1,cex=1.2)
R> legend( x=xmax(mapgrid.sage),y=ymax(mapgrid.sage),
                    legend=legend.label,
                    fill=legend.colors,
                    bty="n",
                    cex=1.2)
R> mtext("RF - Mean Percent Cover",side=3,line=1,cex=1.5,outer=T)
R> par(opar)


R> opar <- par(mfrow=c(1,2),mar=c(3,3,2,1),oma=c(0,0,3,4),xpd=NA)
R> mapgrid.pinyon <- brick(paste(MODELfn.pinyon,"_map_QRF.img",sep=""))
R> mapgrid.sage <- brick(paste(MODELfn.sage,"_map_QRF.img",sep=""))
R> image( mapgrid.pinyon[[2]],
        col=col.ramp,
        xlab="",ylab="",xaxt="n",yaxt="n",
        zlim=zlim,
        asp=1,bty="n",main="")
R> mtext(response.name.pinyon,side=3,line=1,cex=1.2)
R> image( mapgrid.sage[[2]],
        col=col.ramp,
        xlab="",ylab="",xaxt="n",yaxt="n",
        zlim=zlim,
        asp=1,bty="n",main="")
```

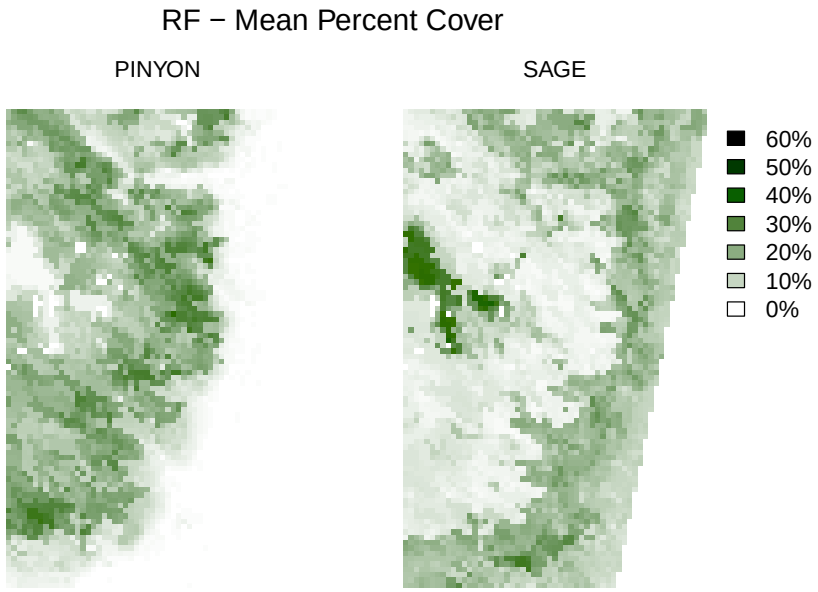

Figure 3: RF Maps of mean percent cover for Pinyon and Sage (RF models).

```
R> mtext(response.name.sage,side=3,line=1,cex=1.2)
R> legend( x=xmax(mapgrid.sage),y=ymax(mapgrid.sage),
                    legend=legend.label,
                    fill=legend.colors,
                    bty="n",
                    cex=1.2)
R> mtext("QRF - Median (50% quantile) Percent Cover",side=3,line=1,cex=1.5,outer=T)
R> par(opar)


R> opar <- par(mfrow=c(1,2),mar=c(3,3,2,1),oma=c(0,0,3,4),xpd=NA)
R> image( mapgrid.pinyon[[1]],
        col=col.ramp,
        xlab="",ylab="",xaxt="n",yaxt="n",
        zlim=zlim,
        asp=1,bty="n",main="")
R> mtext(response.name.pinyon,side=3,line=1,cex=1.2)
R> image( mapgrid.sage[[1]],
        col=col.ramp,
        xlab="",ylab="",xaxt="n",yaxt="n",
        zlim=zlim,
        asp=1,bty="n",main="")
R> mtext(response.name.sage,side=3,line=1,cex=1.2)
R> legend( x=xmax(mapgrid.sage),y=ymax(mapgrid.sage),
```

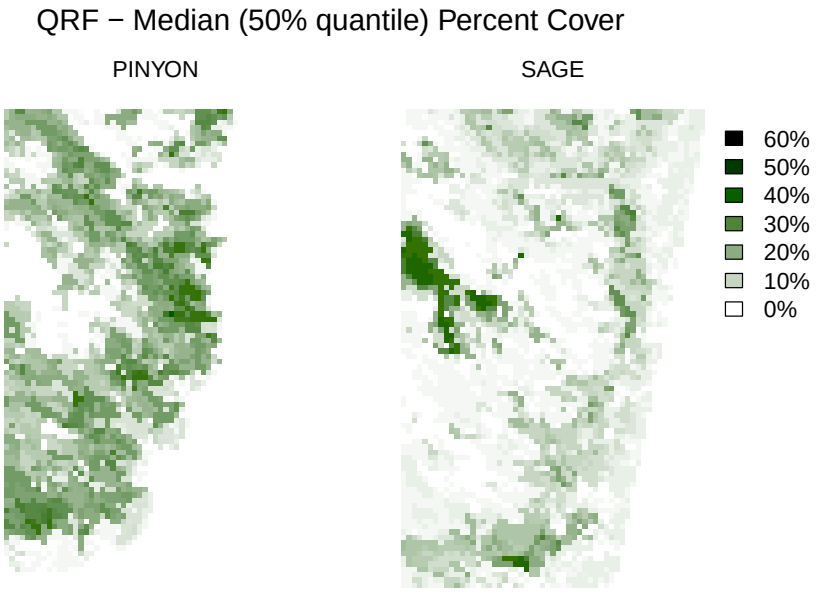

Figure 4: Maps of median percent cover (50% quantile) for Pinyon and Sage (QRF models).

```
                    legend=legend.label,
                    fill=legend.colors,
                    bty="n",
                    cex=1.2)
R> mtext("QRF - Lower bound (10% quantile)",side=3,line=1,cex=1.5,outer=T)
R> par(opar)


R> opar <- par(mfrow=c(1,2),mar=c(3,3,2,1),oma=c(0,0,3,4),xpd=NA)
R> image( mapgrid.pinyon[[3]],
        col=col.ramp,
        xlab="",ylab="",xaxt="n",yaxt="n",
        zlim=zlim,
        asp=1,bty="n",main="")
R> mtext(response.name.pinyon,side=3,line=1,cex=1.2)
R> image( mapgrid.sage[[3]],
        col=col.ramp,
        xlab="",ylab="",xaxt="n",yaxt="n",
        zlim=zlim,
        asp=1,bty="n",main="")
R> mtext(response.name.sage,side=3,line=1,cex=1.2)
R> legend( x=xmax(mapgrid.sage),y=ymax(mapgrid.sage),
                    legend=legend.label,
                    fill=legend.colors,
```

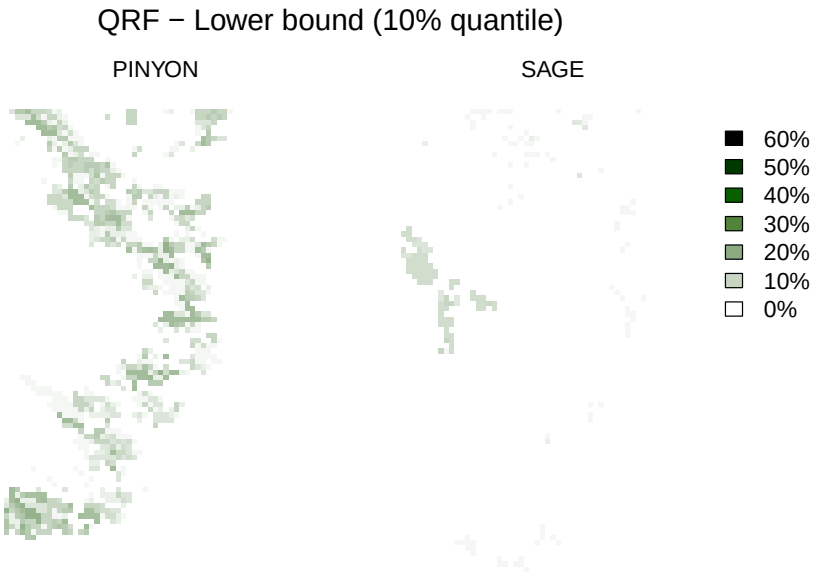

Figure 5: Maps of Lower bound (10% quantile) for Pinyon and Sage (QRF models).

```
                    bty="n",
                    cex=1.2)
R> mtext("QRF model - Upper - 90% quantile",side=3,line=1,cex=1.5,outer=T)
R> par(opar)
```

3.6

4 Conditional Inference Forest

Conditional Inference (CF) models (from the **party** package) address two possible biases in Random Forest models. First, it has been shown that variable importance in Random Forest is biased in cases where predictor variables are different is scale (for continuous predictors) or number of categories (for factored predictors) (Strobl et al., 2007). Second, when predictor variables are correlated, traditional random forest will divide the importance amongst the correlated predictors, sometimes giving the impression that none of these predictors are important to the model.

CF models do not preferentially select for predictors of larger scale or higher numbers of categories. And CF models have an option to calculate the conditional importance, to distinguish between predictors that are truly important to the model, and predictors that only appear important due to their correlation with other predictors.

A draw back with CF models is that they are much more memory intensive than RF models. And currently if you have more than just 3 or 4 predictors, it is not possible for most desktop or laptop computers to calculate the conditional importance for more than approximately 400 data points.

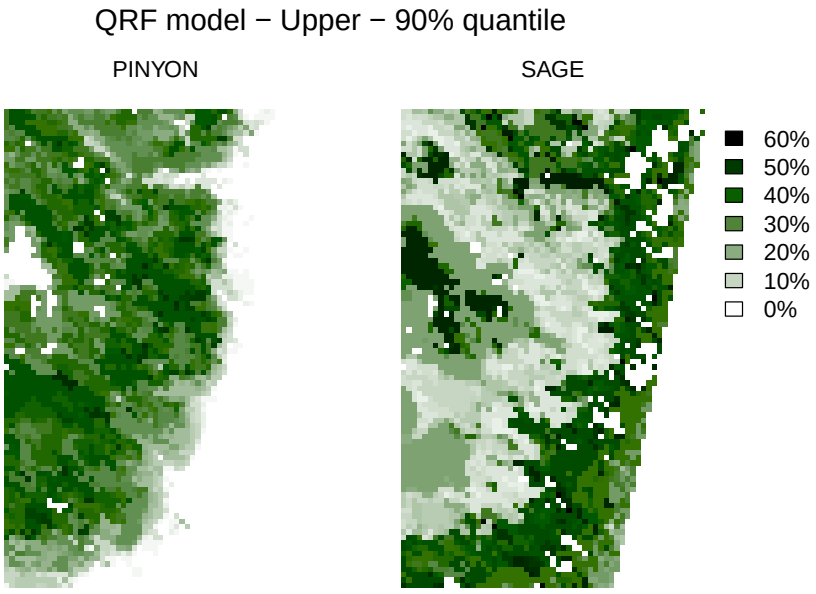

Figure 6: Maps of upper bound (90% quantile) for Pinyon and Sage (QRF models).

4.1 Build Models

```
R> MODELfn.pinyon    <- "VQuantile_CF_Pinyon"
R> MODELfn.sage    <- "VQuantile_CF_Sage"

R> response.name.pinyon <- "PINYON"
R> response.name.sage <- "SAGE"
R> response.type   <- "continuous"

R>   qdata<-read.csv(qdata.trainfn)
R>         IS.NUM<-sapply(qdata,is.numeric)
R>         qdata[,IS.NUM]<-sapply(qdata[,IS.NUM],as.numeric)
R> CF.pinyon  <- model.build( model.type="CF",
                              qdata.trainfn=qdata,
                              folder=folder,
                              unique.rowname=unique.rowname,
                              MODELfn=MODELfn.pinyon,
                              predList=predList,
                              predFactor=predFactor,
                              response.name=response.name.pinyon,
                              response.type=response.type)
R> CF.sage  <- model.build( model.type="CF",
                              qdata.trainfn=qdata,
                              folder=folder,
                              unique.rowname=unique.rowname,
```

```
                        MODELfn=MODELfn.sage,
                        predList=predList,
                        predFactor=predFactor,
                        response.name=response.name.sage,
                        response.type=response.type)
```

4.2 Importance Plots

Here we will compare the unconditional variable importances from the CF model with the unconditional importances from the ordinary RF model (Figure 7). For conditional CF importances, change `cf.conditional.1` to `TRUE`. It is also possible to use `model.importance.plot()` to compare conditional and unconditional importances for a single CF model.

Be aware that conditional importance will take much longer to calculate, and may fail completely if your training data has more than approximately 400 data points.

Note, importances are currently unavailable for QRF models, and even the RF model built as part of a QRF model only currently has `imp.type` equals `2` (the node impurity).

```
R> opar <- par(mfrow=c(2,1),mar=c(3,3,3,3),oma=c(0,0,3,0))
R> Imp.pinyon<-model.importance.plot( model.obj.1=CF.pinyon,
                        model.obj.2=QRF.pinyon$RF,
                        model.name.1="unconditional (CF)",
                        model.name.2="unconditional (RF)",
                        imp.type.1=1,
                        imp.type.2=2,
                        cf.conditional.1=FALSE,
                        sort.by="predList",
                        predList=predList,
                        scale.by="sum",
                        main="Pinyon Percent Cover",
                        device.type="none",
                        cex=0.9)
R> Imp.sage<-model.importance.plot( model.obj.1=CF.sage,
                        model.obj.2=QRF.sage$RF,
                        model.name.1="unconditional (CF)",
                        model.name.2="unconditional (RF)",
                        imp.type.1=1,
                        imp.type.2=2,
                        cf.conditional.1=FALSE,
                        sort.by="predList",
                        predList=predList,
                        scale.by="sum",
                        main="Sage Percent Cover",
                        device.type="none",
                        cex=0.9)
R> mtext("CF versus RF Variable Importance",side=3,line=0,cex=1.8,outer=TRUE)
R> par(opar)
```

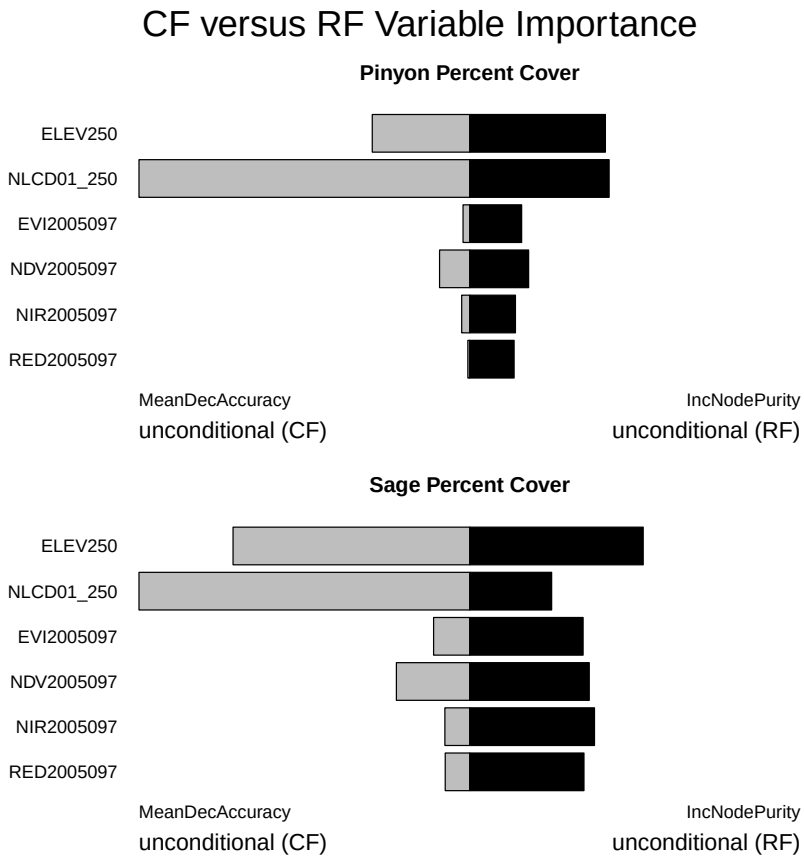

Figure 7: Comparison of conditional variable importance from the CF model with unconditional importance form the RF model .

4.3 Map production

4.3.1 Predict Over the Rasters

```
R> model.mapmake(  model.obj=CF.pinyon,
                   folder=folder,
                   MODELfn=MODELfn.pinyon,
                   rastLUTfn=rastLUTfn,
                   na.action="na.omit"
                )
R> model.mapmake(  model.obj=CF.sage,
                   folder=folder,
                   MODELfn=MODELfn.sage,
                   rastLUTfn=rastLUTfn,
                   na.action="na.omit"
                )
```

4.3.2 CF Map

Maps of CF models (Figure 8).

```
R> opar <- par(mfrow=c(1,2),mar=c(3,3,2,1),oma=c(0,0,3,4),xpd=NA)
R> mapgrid.pinyon <- raster(paste(MODELfn.pinyon,"_map.img",sep=""))
R> mapgrid.sage <- raster(paste(MODELfn.sage,"_map.img",sep=""))
R> zlim <- c(0,60)
R> image( mapgrid.pinyon,
        col=col.ramp,
        xlab="",ylab="",xaxt="n",yaxt="n",
        zlim=zlim,
        asp=1,bty="n",main="")
R> mtext(response.name.pinyon,side=3,line=1,cex=1.2)
R> image( mapgrid.sage,
        col=col.ramp,
        xlab="",ylab="",xaxt="n",yaxt="n",
        zlim=zlim,
        asp=1,bty="n",main="")
R> mtext(response.name.sage,side=3,line=1,cex=1.2)
R> legend( x=xmax(mapgrid.sage),y=ymax(mapgrid.sage),
                    legend=legend.label,
                    fill=legend.colors,
                    bty="n",
                    cex=1.2)
R> mtext("CF - Mean Percent Cover",side=3,line=1,cex=1.5,outer=T)
R> par(opar)
```

References

W. Anderson, S. Guikema, B. Zaitchik, and W. Pan. Methods for estimating population density in data-limited areas: Evaluating regression and tree-based models in peru. *PLoS one*, 9(7): e100037, 2014.

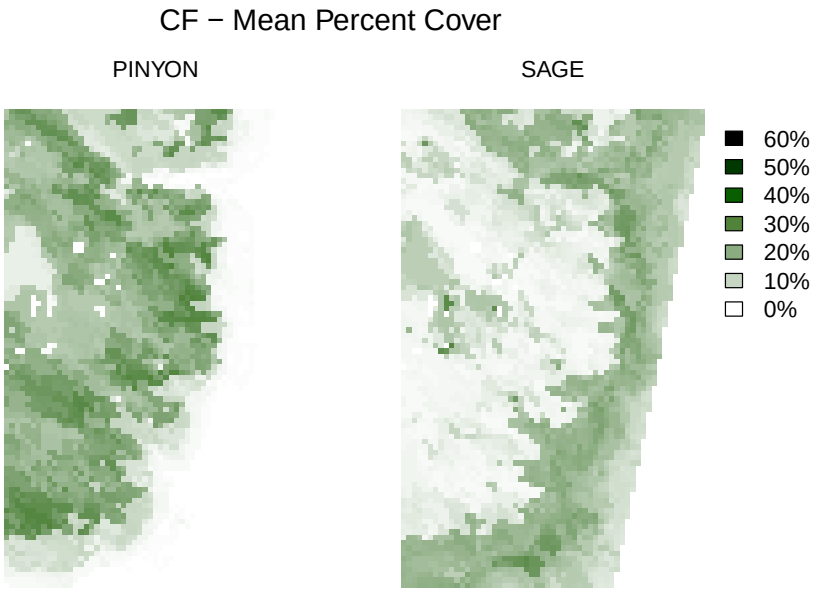

Figure 8: RF Maps of mean percent cover for Pinyon and Sage (RF models).

L. Breiman, R. A. Friedman, R. A. Olshen, and C. G. Stone. *Classification and Regression Trees*. Wadsworth, 1984.

J. Elith, J. R. Leathwick, and T. Hastie. A working guide to boosted regression trees. *Journal of Animal Ecology*, 77:802–813, 2008.

E. Freeman. ***ModelMap****: An R Package for Modeling and Map production using Random Forest and Stochastic Gradient Boosting*. USDA Forest Service, Rocky Mountain Research Station, 507 25th street, Ogden, UT, USA, 2009. URL `http://CRAN.R-project.org/`. eafreeman@fs.fed.us.

T. S. Frescino, G. G. Moisen, K. A. Megown, V. J. Nelson, Elizabeth, Freeman, P. L. Patterson, M. Finco, K. Brewer, and J. Menlove. Nevada photo-based inventory pilot(npip) photo sampling procedures. Gen. Tech. Rep. RMRSGTR-222, U.S. Departmentof Agriculture, Forest Service, Rocky Mountain Research Station., Fort Collins, CO, 2009.

D. Gesch, M. Oimoen, S. Greenlee, C. Nelson, M. Steuck, and D. Tyler. The national elevation dataset. photogrammetric engineering and remote sensing. *Photogrammetric Engineering and Remote Sensing*, 68:5–11, 2002.

C. Homer, C. Huang, L. Yang, B. Wylie, and M. Coan. Development of a 2001 national landcover database for the united states. *Photogrammetric Engineering and Remote Sensing*, 70: 829–840, 2004.

T. Hothorn, P. Buehlmann, S. Dudoit, A. Molinaro, and M. V. D. Laan. Survival ensembles. *Biostatistics*, 7(3):355–373, 2006.

A. Huete, K. Didan, T. Miura, E. P. Rodriguez, X. Gao, and L. G. Ferreira. Overview of the radiometric and biophysical performance of the modis vegetation indices. *Remote Sensing of Environment*, 83:195–213, 2002.

C. O. Justice, J. R. G. Townshend, E. F. Vermote, E. Masuoka, R. E. Wolfe, N. Saleous, D. P. Roy, and J. T. Morisette. An overview of modis land data processing and product status. *Remote Sensing of Environment*, 83:3–15, 2002.

N. Meinshausen and L. Schiesser. *quantregForest: Quantile Regression Forests*, 2015. URL `http://CRAN.R-project.org/package=quantregForest`. R package version 1.1.

R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2008. URL `http://www.R-project.org`. ISBN 3-900051-07-0.

C. Strobl, A.-L. Boulesteix, A. Zeileis, and T. Hothorn. Bias in random forest variable importance measures: Illustrations, sources and a solution. *BMC Bioinformatics*, 8(25), 2007. URL `http://www.biomedcentral.com/1471-2105/8/25`.

C. Strobl, A.-L. Boulesteix, T. Kneib, T. Augustin, and A. Zeileis. Conditional variable importance for random forests. *BMC Bioinformatics*, 9(307), 2008. URL `http://www.biomedcentral.com/1471-2105/9/307`.