Package `fsvm`: Fine Scale Vegetation Modeling

Idaho Department of Fish and Game

Robert Ritson, WMI Research Associate

2021-05-26

Licensing Agreement: the `fsvm` R package is proprietary and belongs to the Idaho Department of Fish and Game (IDFG) with partial funding from the Wildlife Management Institute (WMI). As such, use of this package or any of its functions or data is only authorized for IDFG personnel, partners, and cooperators for the purposes of IDFG funded research including 'Ecology of Everything' (EoE) and 'Fine Scale Vegetation Modeling' (FSVM).

`fsvm` Workflow: From raw field data to machine learning species distribution predictions

The purpose of this vignette is to illustrate the workflow for creating and updating Idaho Department of Fish and Game's Fine Scale Vegetation Model (FSVM) and guide IDFG personnel involved with the project in using the `fsvm` package to format data and get model predictions. The aim of this package is to streamline the data analysis process for this project and ensure its repeatability. Please report any bugs or suggestions to Robert Ritson at robert.ritson@idfg.idaho.gov using the subject line "FSVM".

The current package release includes a variety functions for formatting raw field data including rectifying taxonomic names, spatially associating survey data, associating data with University of Idaho produced eCognition quadpolygons of covariate data and ecological regions, as well as formatting data for machine learning models. In addition to data preparation, there are also functions for creating machine learning models, predicting from them, summarizing outputs, and producing centroid maps of the output. The majority of these functions are adapted or modified from Erin Roche's templates and functions located on the network drive *HQWILDSTAT:/Fine scale vegetation analysis/understory_veg_model*.

This package also incorporates Python scripts in some functions using the R package 'reticulate' in order to centralize FSVM workflow into a single analytical environment. This requires a valid installation of either ArcMap or ArcPro on your machine. Please note that functions using Python and ArcGIS default to ArcMap and will need to be modified by ArcPro users.

Getting Started

The current release of the `fsvm` package will always be located on the *HQWILDSTAT* network drive and will be updated anytime bugs are fixed or functions, which should be less frequent with time. To install on your local machine, you first need access to the *HQWILDSTAT* network drive then run the following function. Note that the *HQWILDSTAT* network drive will hereafter be referred to as *A:* drive.

```
install.packages("A:/Fine scale vegetation analysis/fsvm_package/fsvm_0.0.3.tar.gz",
                 repos = NULL, type = "source")
> Installing package into 'C:/Users/rritson/Documents/R/win-library/4.0'
> (as 'lib' is unspecified)
library(fsvm)
```

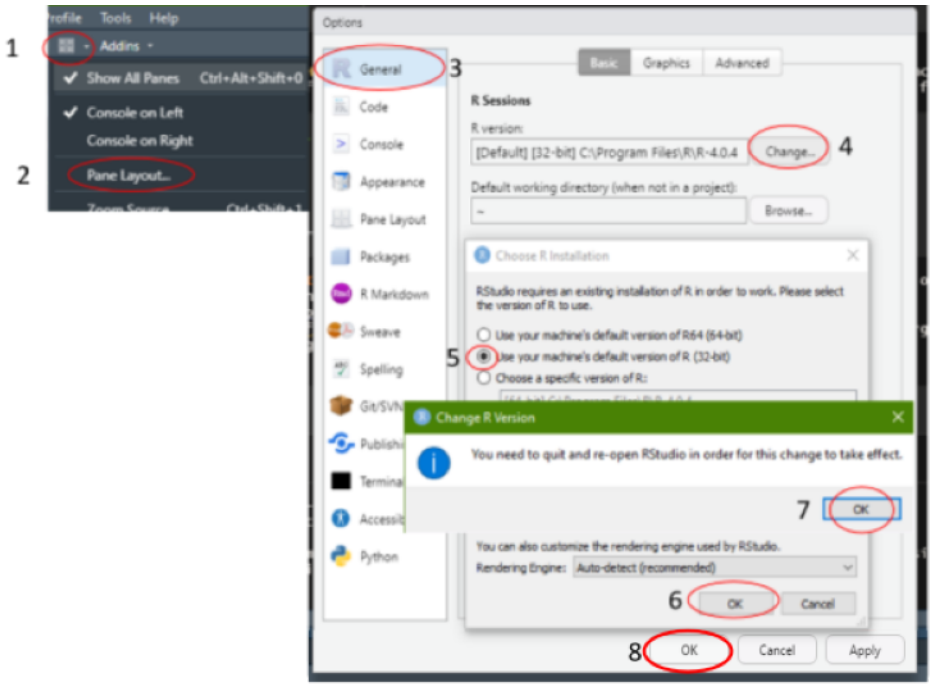
```
> Loading required package: dplyr
>
> Attaching package: 'dplyr'
> The following objects are masked from 'package:stats':
>
>     filter, lag
> The following objects are masked from 'package:base':
>
>     intersect, setdiff, setequal, union
> Loading required package: sp
> Loading required package: sf
> Linking to GEOS 3.9.0, GDAL 3.2.1, PROJ 7.2.1
> Loading required package: ggplot2
> Loading required package: data.table
>
> Attaching package: 'data.table'
> The following objects are masked from 'package:dplyr':
>
>     between, first, last
> Loading required package: foreach
> [[1]]
> [1] TRUE
>
> [[2]]
> [1] TRUE
>
> [[3]]
> [1] TRUE
>
> [[4]]
> [1] TRUE
>
> [[5]]
> [1] TRUE
>
> [[6]]
> [1] TRUE
```

Currently, any function requiring Python and ArcGIS (`py_extract_quadpolyID` and `getSVS`) need to be executed within a 32-bit architecture of R in order to match ESRI's ArcMap 32-bit architecture. Unless you have ESRI's 64-bit Background Geoprocessor (https://desktop.arcgis.com/en/arcmap/10.3/analyze/executin g-tools/64bit-background.htm) installed, then you will need switch between architectures. For the purposes of this vignette, we will leave the machine in 32-bit for the entire workflow, but certain functions may run better in a 64-bit environment with larger datasets. Future versions of this package will include a download for a 64-bit background geoprocessor, but have not been available as of this release.

Let's check the current architecture of your machine.

```r
Sys.getenv("R_ARCH")
> [1] "/i386"
```

If the command returned "/i386", then you are currently running R in a 32-bit architecture and are ready to proceed. If the command returned "/x64", then your current R session is running in a 64-bit architecture. To switch to 32-bit, follow these steps:



1. Open 'Workspace Panes'
2. Go to 'Pane Layout'
3. Click 'General'
4. Click 'Change' under R version
5. In the 'Choose R Installation' window, click 'Use your machine's default version of R (32-bit)'
6. Click 'OK' in the 'Choose R Installation' window
7. In the pop-up window 'Change R Version', click 'OK'
8. Click 'OK' in the 'Options' window
9. Quit RStudio session (Ctrl+Q), then re-open RStudio

When you re-run the command `Sys.getenv("R_ARCH")` it should now return "/i386".

Dummy Dataset

To illustrate the workflow, we will be working with the dummy dataset embedded in the package (`fsvm_dummy`).

```r
#Load data
head(fsvm::fsvm_dummy)
>           TranKey       PlotKey         Source DataType Smpl_Yr Easting Northing
> 1: J_Aster_Plot4 J_Aster_Plot4 Jessica Aster    COVER    2014 2327731  1711541
> 2: J_Aster_Plot4 J_Aster_Plot4 Jessica Aster    COVER    2014 2327731  1711541
> 3: J_Aster_Plot4 J_Aster_Plot4 Jessica Aster    COVER    2014 2327731  1711541
> 4: J_Aster_Plot4 J_Aster_Plot4 Jessica Aster    COVER    2014 2327731  1711541
> 5: J_Aster_Plot4 J_Aster_Plot4 Jessica Aster    COVER    2014 2327731  1711541
> 6: J_Aster_Plot4 J_Aster_Plot4 Jessica Aster    COVER    2014 2327731  1711541
>    Scientific_Name_Complete Prcnt_C Count
> 1:          Pinus ponderosa       1     1
```

```
> 2:    Pseudotsuga menziesii      40     1
> 3:    Amelanchier alnifolia      10     1
> 4:      Holodiscus discolor      10     1
> 5:         Lonicera ciliosa       1     1
> 6:           Mahonia repens       1     1
```

Four commonly used data types are included in this dataset: Herbarium records, IDFG Line Point Intercept, Macro Cover Plot, and IDFG Simple Vegetation Survey (Survey123). Each of these needs to be dealt with slightly different.

Herbarium records are individual locations corresponding with presence of a plant species. While these can be updated each year with additional observations and collections, it is not anticipated to be a very dynamic data stream. Since observations are collected opportunistically, these do not need to be spatially associated by a survey methodology.

```
head(fsvm_dummy[grep("Herbarium", fsvm_dummy$Source)])
>          TranKey       PlotKey               Source DataType Smpl_Yr Easting
> 1:  CPNWH_904891  CPNWH_904891  CPNWH_ID_Herbarium Presence    1988 2519183
> 2:  CPNWH_988344  CPNWH_988344  CPNWH_ID_Herbarium Presence    1912 2295254
> 3: CPNWH_3110071 CPNWH_3110071 CPNWH_IDS_Herbarium Presence    2016 2431078
> 4:  CPNWH_143135  CPNWH_143135 CPNWH_WTU_Herbarium Presence    1944 2523689
> 5: CPNWH_1000808 CPNWH_1000808  CPNWH_ID_Herbarium Presence    2014 2294908
> 6: CPNWH_1813246 CPNWH_1813246  CPNWH_RM_Herbarium Presence    2007 2373525
>    Northing Scientific_Name_Complete Prcnt_C Count
> 1:  1434209            Poa leptocoma      NA     1
> 2:  1833933           Nepeta cataria      NA     1
> 3:  1447476        Elymus virginicus      NA     1
> 4:  1439594         Antennaria rosea      NA     1
> 5:  1537363           Polygonum majus      NA     1
> 6:  1626339           Trillium ovatum      NA     1
```

IDFG Line Point Intercept surveys are generally 50 meter transects with point survey every 0.5 meter where the species presence is recorded. Raw LPI surveys will need to be prepped, including translating USDA plant codes to scientific names (`usda_translate`) and interpolating survey points along the survey transect (`assign_Lpi`). New surveys will likely be processed each year as IDFG staff will be conducting these surveys in regions of the state lacking data.

```
head(fsvm_dummy[grep("LPI", fsvm_dummy$Source)])
>                     TranKey                      PlotKey   Source DataType
> 1: IDFG_2019_q42113c1_3152 IDFG_2019_q42113c1_3152_0.5 IDFG_LPI      LPI
> 2: IDFG_2019_q42113c1_3152   IDFG_2019_q42113c1_3152_1 IDFG_LPI      LPI
> 3: IDFG_2019_q42113c1_3152 IDFG_2019_q42113c1_3152_1.5 IDFG_LPI      LPI
> 4: IDFG_2019_q42113c1_3152   IDFG_2019_q42113c1_3152_2 IDFG_LPI      LPI
> 5: IDFG_2019_q42113c1_3152 IDFG_2019_q42113c1_3152_2.5 IDFG_LPI      LPI
> 6: IDFG_2019_q42113c1_3152   IDFG_2019_q42113c1_3152_3 IDFG_LPI      LPI
>    Smpl_Yr Easting Northing Scientific_Name_Complete Prcnt_C Count
> 1:    2019 2578976  1239915    Eriogonum microthecum      NA     1
> 2:    2019 2578977  1239916          Bromus tectorum      NA     1
> 3:    2019 2578977  1239916               Poa secunda      NA     1
> 4:    2019 2578978  1239916  Pseudoroegneria spicata      NA     1
> 5:    2019 2578978  1239916  Pseudoroegneria spicata      NA     1
```

```
> 6:    2019 2578978  1239916                     <NA>      NA     1
```

Macro Cover Plot surveys vary by source, but generally comprise a 60 meter baseline with 3-4 30 meter LPI-style transects perpendicular to the baseline. Given this protocol, presence data for this survey can be easily converted to percent cover. These surveys are essentially 60x30 meter rectangles and need to be spatially represented as such (`assign_Plot`). While very useful, these intensive surveys are not currently included in IDFG's annual data collection plans for the fine scale vegetation model. Additional macro cover plots may occasionally be incorporated opportunistically.

```
head(fsvm_dummy[grep("COVER", fsvm_dummy$DataType)])
>           TranKey       PlotKey        Source DataType Smpl_Yr Easting Northing
> 1: J_Aster_Plot4 J_Aster_Plot4 Jessica Aster    COVER    2014 2327731  1711541
> 2: J_Aster_Plot4 J_Aster_Plot4 Jessica Aster    COVER    2014 2327731  1711541
> 3: J_Aster_Plot4 J_Aster_Plot4 Jessica Aster    COVER    2014 2327731  1711541
> 4: J_Aster_Plot4 J_Aster_Plot4 Jessica Aster    COVER    2014 2327731  1711541
> 5: J_Aster_Plot4 J_Aster_Plot4 Jessica Aster    COVER    2014 2327731  1711541
> 6: J_Aster_Plot4 J_Aster_Plot4 Jessica Aster    COVER    2014 2327731  1711541
>    Scientific_Name_Complete Prcnt_C Count
> 1:          Pinus ponderosa       1     1
> 2:    Pseudotsuga menziesii      40     1
> 3:    Amelanchier alnifolia      10     1
> 4:      Holodiscus discolor      10     1
> 5:         Lonicera ciliosa       1     1
> 6:           Mahonia repens       1     1
```

IDFG Simple Vegetation Survey via Survey123 is will mainly be used for ground-truthing the vegetation classification map and potentially informative for the FSVM model. These survey are generally conducted from a central coordinate with a 5 meter radius where the collector identifies the most common species present and visually estimates categorical cover. The main data preparation is to transform the coordinates to IDTM projection and spatially represent the survey radius around the coordinate (`assign_Plot`). An additional function for downloading Survey123 data directly from ArcGIS online (`getSVS`) is also included in this package, however, a valid ArcGIS Online account is required and the function has not yet been fully tested. These surveys are planned to be conducted annually by IDFG staff.

```
head(fsvm_dummy[grep("SVS", fsvm_dummy$Source)])
>                                       TranKey
> 1: IDFG_SVS_75f3a731-ee56-4993-99fe-7e10b8749dff
> 2: IDFG_SVS_75f3a731-ee56-4993-99fe-7e10b8749dff
> 3: IDFG_SVS_75f3a731-ee56-4993-99fe-7e10b8749dff
> 4: IDFG_SVS_75f3a731-ee56-4993-99fe-7e10b8749dff
> 5: IDFG_SVS_75f3a731-ee56-4993-99fe-7e10b8749dff
> 6: IDFG_SVS_7ee2a62b-c8d6-4cc3-a41b-2a474b8bf6de
>                                       PlotKey   Source DataType Smpl_Yr
> 1: IDFG_SVS_75f3a731-ee56-4993-99fe-7e10b8749dff IDFG_SVS Presence    2020
> 2: IDFG_SVS_75f3a731-ee56-4993-99fe-7e10b8749dff IDFG_SVS Presence    2020
> 3: IDFG_SVS_75f3a731-ee56-4993-99fe-7e10b8749dff IDFG_SVS Presence    2020
> 4: IDFG_SVS_75f3a731-ee56-4993-99fe-7e10b8749dff IDFG_SVS Presence    2020
> 5: IDFG_SVS_75f3a731-ee56-4993-99fe-7e10b8749dff IDFG_SVS Presence    2020
> 6: IDFG_SVS_7ee2a62b-c8d6-4cc3-a41b-2a474b8bf6de IDFG_SVS Presence    2020
>    Easting Northing Scientific_Name_Complete Prcnt_C Count
> 1: 2650461  1470926                    Salix      NA     1
> 2: 2650461  1470926           Elymus glaucus      NA     1
```

```
> 3: 2650461  1470926  Pseudoroegneria spicata      NA     1
> 4: 2650461  1470926   Rudbeckia occidentalis      NA     1
> 5: 2650461  1470926                     <NA>      NA     1
> 6: 2281146  1274409      Artemisia arbuscula      NA     1
```

In addition to spatial and taxonomic concerns, all data needs to be associated with the eCognition polygons of covariates (`py_extract_quadPolyID` and `getCovariates`) and Bailey's Eco-Regions (`assign_ecoregions`) as well as formatted for modeling (`as_fsvm`). Following data preparation, machine learning models are built ('Run Models'), predicted from ('Predict Models'), and summarized ('Summary and Maps').

Data Preparation

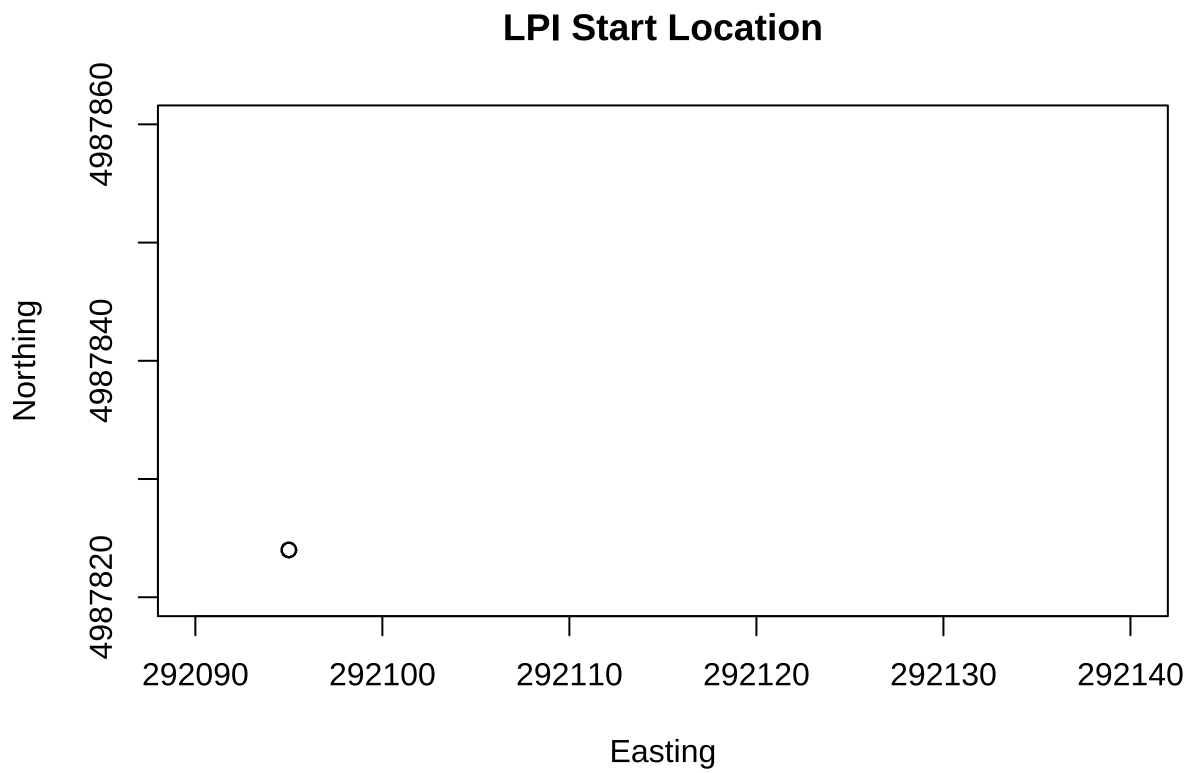
The data preparation suite of functions include taxonomic formatting (`usda_translate` and `rectify_taxa`), spatial interpolation (`assign_Lpi` and `assign_Plot`), covariate association (`py_extract_quadpolyID`, `select_quads`, `set_extent_manual`, `getCovariates` and `assign_ecoregions`), and final formatting for modeling (`as_fsvm`). From this point, it is assumed that field data has been compiled and that metadata (including relevant protocols for each source) are available.

Survey Point/Plot Interpolation

In order to accurately model fine scale vegetation distributions, we need accurate locations for each species observation in a survey. How this is accomplished depends on the protocol used to collect the data: Line Point Intercept, Macro Cover Plot, or Simple Vegetation Survey. Given the current modeling methodology, we are primarily concerned with interpolating the point intercepts of LPI surveys as generally only the starting point of the transect is recorded (`assign_Lpi`). Future methods may incorporate polygons containing observations, which can be created with the function `assign_Plot`. Point interpolations should be performed on raw data prior before merging with other data. Plot interpolation can be performed at any point as long as relevant metadata of the data sources is available (plot dimensions, azimuths, etc.).

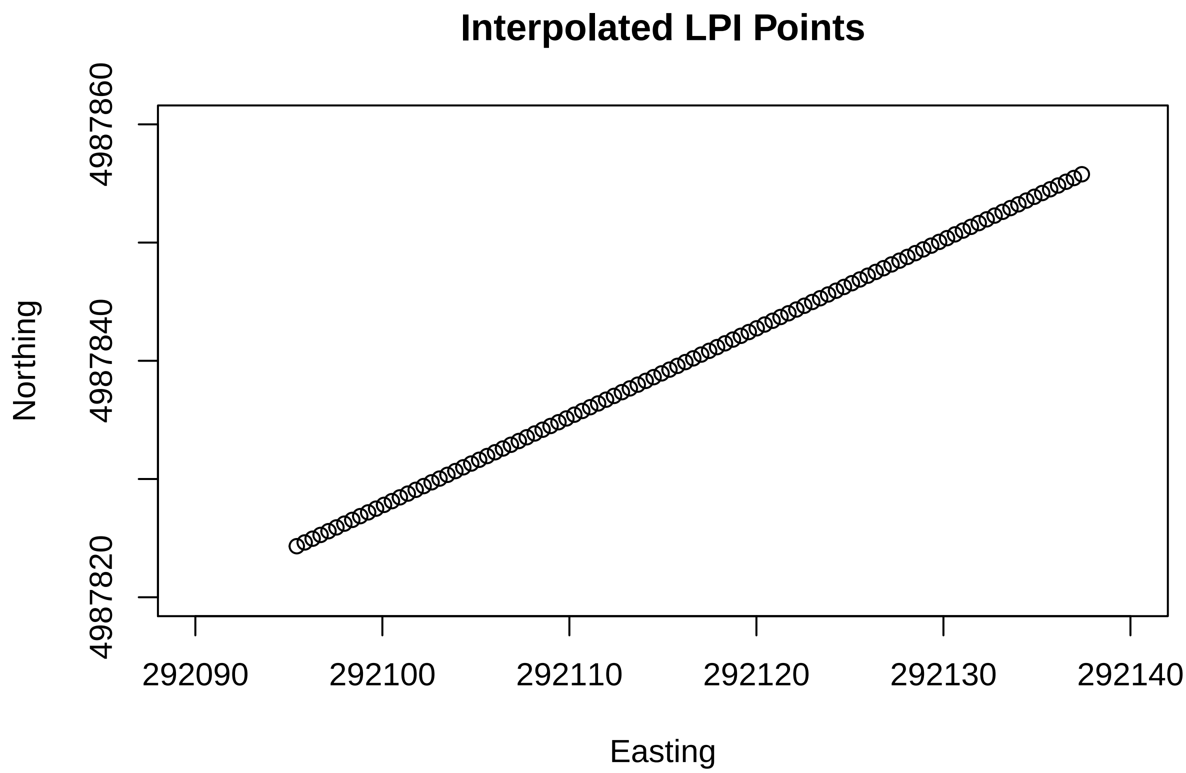
LPI Survey Interpolation For Line Point Intercept (LPI) surveys, coordinates of each point intercept along a survey line transect need to be interpolated given one of the following: a starting coordinate paired with an azimuth and survey interval, an ending coordinate paired with an azimuth and survey interval, or a starting, middle, and ending coordinates paired with a survey interval. The following example uses raw data from an LPI survey with an azimuth and a start coordinate for each survey. The line is 50 meters with intercepts every 0.5 meters.

```
## Line Point Intercept Interpolation Example
#Load Data
lpi <- data.table::fread(
"A:/Fine scale vegetation analysis/understory_veg_model/data/FieldData/DataFormatting/original_data/dima
)
lpi1 <- lpi[c(1:100),c(3,5,6,12,13,15,16)]
head(lpi1)
>             LineKey PointLoc TopCanopy Easting Northing azimuth distance
> 1: 1402241645214620      0.5      none  292095  4987824       0      100
> 2: 1402241645214620      1.0    artrw8  292095  4987824       0      100
> 3: 1402241645214620      1.5     PHLO2  292095  4987824       0      100
> 4: 1402241645214620     10.0     pssp6  292095  4987824       0      100
> 5: 1402241645214620     10.5      pose  292095  4987824       0      100
> 6: 1402241645214620     11.0     PHLO2  292095  4987824       0      100
plot(lpi1[,c("Easting","Northing")],xlim=c(292090,292140),ylim=c(4987820,4987860),
     main = "LPI Start Location")
```



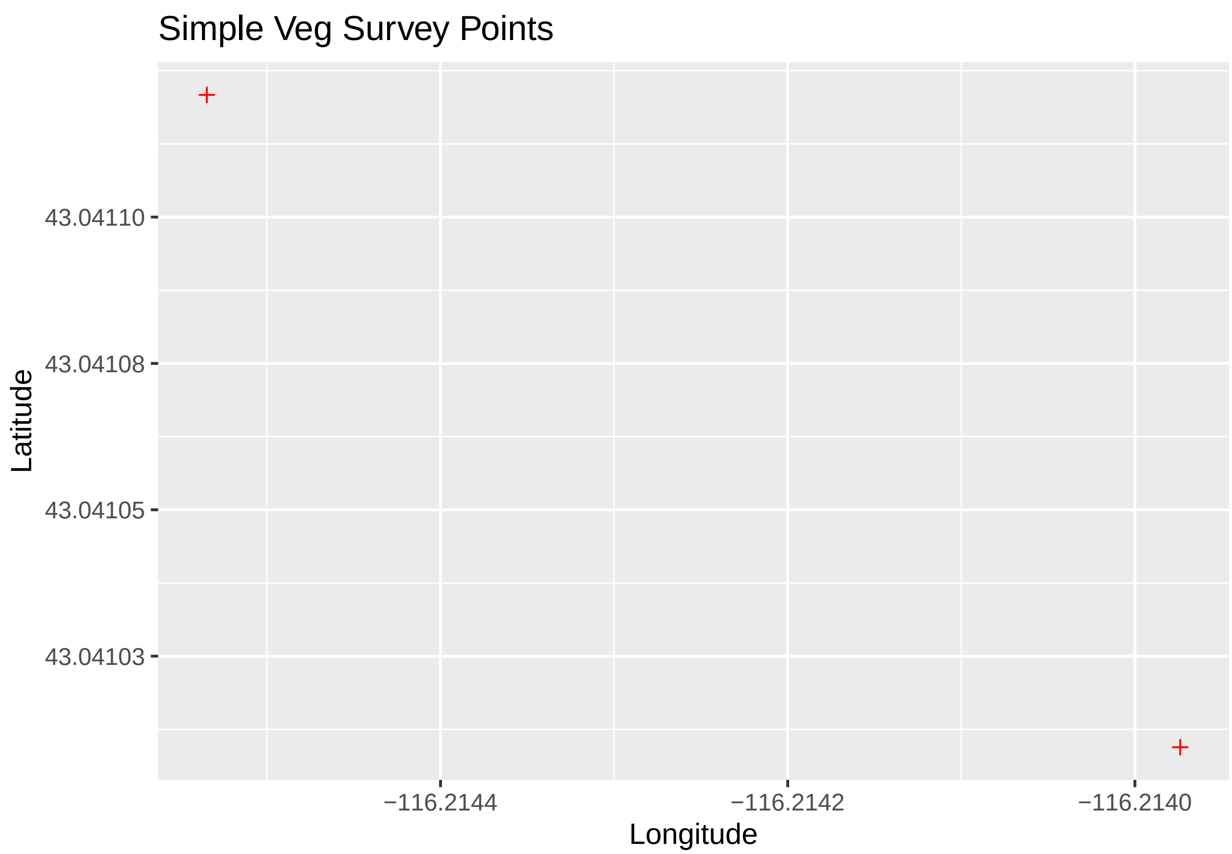

```
#Assign locations to points in LPI survey
lpi.locs <- fsvm::assign_Lpi(lpi = lpi1, x = "Easting", y = "Northing", ID = "LineKey",
                              interval = "PointLoc", n = "distance", units = "m",
                              azimuth = "azimuth", coord.type = "Start", datum = "IDTM")
head(lpi.locs)
> # A tibble: 6 x 4
>   ParentGlobalID MeterMrkr Easting Northing
>          <int64>     <dbl>   <dbl>    <dbl>
> 1        1.00e15       0.5 292095. 4987824.
> 2        1.00e15       1   292096. 4987825.
> 3        1.00e15       1.5 292096. 4987825.
> 4        1.00e15       2   292097. 4987825.
> 5        1.00e15       2.5 292097. 4987826.
> 6        1.00e15       3   292098. 4987826.

#Plot
plot(lpi.locs[,c("Easting","Northing")],xlim=c(292090,292140),ylim=c(4987820,4987860),
     main = "Interpolated LPI Points")
```



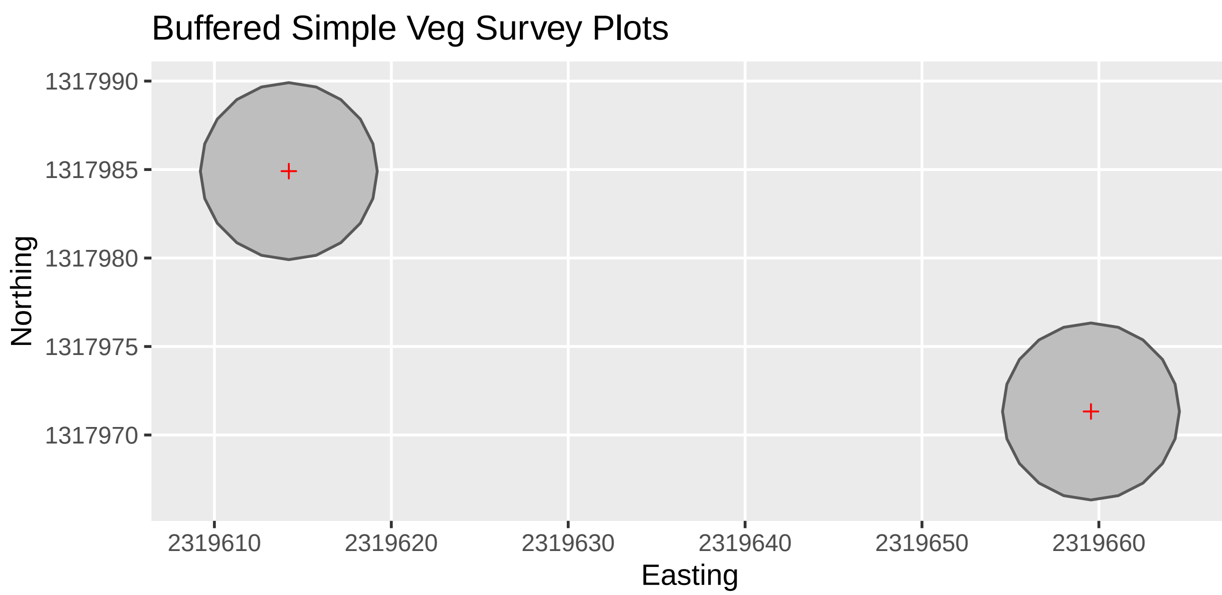
Create Plot Polygon The next function creates plot polygons around survey locations and calculates plot area. This includes drawing a specified circular buffer around a survey location like an IDFG Simple Veg Survey (Survey123 app), or a rectangular polygon like with Macro-cover plots. The following example uses Simple Veg Survey type data point.

```
## Survey 123 Example
#Load Data
svs <- data.table::fread(
"A:/Fine scale vegetation analysis/understory_veg_model/data/FieldData/DataFormatting/original_data/Sim
)
svs <- svs[c(100:101),c(2,4,9,10,11)]
head(svs)
>                                 GlobalID            Date Latitude Longitude
> 1: 61c91854-4ae6-4ad4-8523-a5fe048822dd 7/13/2020 18:00 43.04101 -116.2140
> 2: ba38fd6d-43f8-4387-9398-994beb099b6d 7/13/2020 18:00 43.04112 -116.2145
>    Radius
> 1:      5
> 2:      5
svs.pts <- ggplot(data = svs) +
  geom_point(aes(x = `Longitude`, y = `Latitude`), shape=3, color = "red") +
  ggtitle("Simple Veg Survey Points")
plot(svs.pts)
```




```
#Buffer Simple Veg Survey location by survey radius
svs.locs <- fsvm::assign_Plot(dat = svs, x = "Longitude", y = "Latitude",
                              ID = "GlobalID", size = "Radius", units = "m",
                              type = "SVS", proj = "WGS84")
head(svs.locs)
> Simple feature collection with 2 features and 6 fields
> Geometry type: POLYGON
> Dimension:     XY
> Bounding box:  xmin: 2319609 ymin: 1317966 xmax: 2319665 ymax: 1317990
> CRS:           +proj=tmerc +lat_0=42 +lon_0=-114 +k=0.9996 +x_0=2500000 +y_0=1200000 +datum=NAD83 +uni
>                                 GlobalID            Date Radius
> 1 61c91854-4ae6-4ad4-8523-a5fe048822dd 7/13/2020 18:00      5
> 2 ba38fd6d-43f8-4387-9398-994beb099b6d 7/13/2020 18:00      5
>                           geometry Easting Northing  Plot_Ar
> 1 POLYGON ((2319665 1317971, ... 2319660  1317971 78.53982
> 2 POLYGON ((2319619 1317985, ... 2319614  1317985 78.53982

#Plot
svs.plot <- ggplot(data = svs.locs) + geom_sf(aes(geometry = geometry), fill = "gray") +
  coord_sf(datum = sf::st_crs(
    "+proj=tmerc +lat_0=42 +lon_0=-114 +k=0.9996 +x_0=2500000 +y_0=1200000 +ellps=GRS80 +units=m +no_de
  geom_point(aes(x = `Easting`, y = `Northing`), shape=3, color = "red") +
  ggtitle("Buffered Simple Veg Survey Plots")
plot(svs.plot)
```



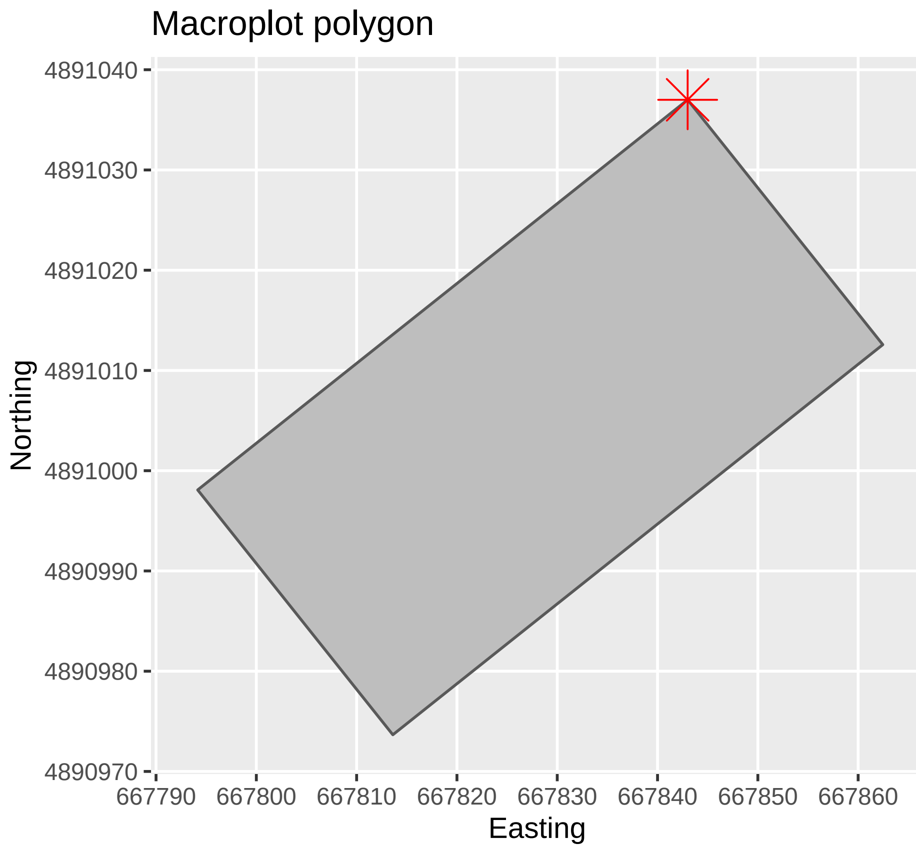
Note that `assign_Plot` automatically projects coordinates to Idaho Transverse Mercator from the original supplied datum (typically WGS84, which is a default). Other source datums can be supplied to the 'proj' argument using a valid proj4string (https://spatialreference.org/ref/).

The following example draws survey polygon for a Macro Cover Plot given the starting location of the baseline and first transect, azimuth of the baseline and first transect, and lengths of the baseline and transects (typically 60 meter baseline with 30 meter transects, defaults).

```
## Cover Plot Example
#Load Data
macro <- data.table::fread(
"A:/Fine scale vegetation analysis/understory_veg_model/data/FieldData/DataFormatting/original_data/sie
)
macro <- macro[,c(1:4,8:11)]
macro$ID <- paste0(macro$Cluster,"_",macro$Macroplot)
macro <- macro[1, c(1,2,9,5:8)]
head(macro)
>        Zone      Date  ID BaseHeadin TransectHe X_COORD Y_COORD
> 1: Sawtooth 6/23/2017 1_A        185         95  667843 4891037
macro.pts <- ggplot(data = macro) +
  geom_point(aes(`X_COORD`,`Y_COORD`), shape=8, color="red", size=7) +
  ggtitle("Macroplot locations") + xlim(667793,667863) + ylim(4890973,4891038)
plot(macro.pts)
```


```
#Draw macroplot polygon based on azimuths, distances, and starting location
macro.poly <- fsvm::assign_Plot(dat = macro, x = "X_COORD", y = "Y_COORD", ID = "ID",
                                units = "m", type = "Macro", proj = "IDTM",
                                base_dir = "BaseHeadin", tran_dir = "TransectHe",
                                base_len = 60, tran_len = 30)
head(macro.poly)
> Simple feature collection with 1 feature and 8 fields
> Geometry type: POLYGON
> Dimension:     XY
> Bounding box:  xmin: 667794.2 ymin: 4890974 xmax: 667862.5 ymax: 4891037
> CRS:           +proj=tmerc +lat_0=42 +lon_0=-114 +k=0.9996 +x_0=2500000 +y_0=1200000 +datum=NAD83 +uni
>       Zone      Date  ID BaseHeadin TransectHe                       geometry
> 1 Sawtooth 6/23/2017 1_A        185         95 POLYGON ((667843 4891037, 6...
>   Easting Northing Plot_Ar
> 1  667843  4891037    1800

#Plot
macro.plot <- ggplot(data = macro.poly) + geom_sf(aes(geometry = geometry), fill = "gray") +
  coord_sf(datum = sf::st_crs(
    "+proj=tmerc +lat_0=42 +lon_0=-114 +k=0.9996 +x_0=2500000 +y_0=1200000 +ellps=GRS80 +units=m +no_de
  geom_point(aes(`Easting`,`Northing`), shape=8, color="red", size=7) +
  ggtitle("Macroplot polygon") + xlim(667793,667863) + ylim(4890973,4891038)
plot(macro.plot)
```



Taxonomy

One of the biggest hurdles in creating fine scale vegetation models from multiple data streams is dealing with plant names. I wrote two functions which help with this by translating USDA Plant Codes to scientific names (and vice versa) as well as rectifying taxonomic names by assigning the most up-to-date scientific names to synonymous species. This important for ensuring modeling groups include all valid members of a genus, species, or subspecies. This function assigns Integrated Taxonomic Information System (itis.gov) Taxonomic Serial Numbers (TSN) as the taxonomic identifier as well as the current accepted scientific name for the species. As you will see, four modeling groups are assigned ("G1","G2","G3","G4") which correspond with genus only, species, subspecies, and variety (of the accepted scientific name). If IFWIS Taxonomic IDs are required, it is recommended to first use the `rectify_taxa` function (after `usda_resolve` if a USDA code) in order to deal with any misspelled names, then run `ifwis_resolve`. For the purposes of modeling, it is recommended to use the Taxonomic Serial Number (TSN) produced by `rectify_taxa` as this is better at grouping taxonomic synonyms together.

Resolving USDA Plant Codes The majority of field data typically records plant species observations using codes developed by the United States Department of Agriculture. The first step to rectifying taxonomic names for the models is to translate the codes to their Latin binomials using `usda_resolve`.

```
## Resolve USDA Plant Codes to Scientific Name
plantcodes <- data.frame(Code = c("ARTRW8","LEPU","PHLO2","ABCO","PIEN"))
plantcodes.resolved <- fsvm::usda_resolve(dat = plantcodes, target = "Code",
                                          resolve = "c2n")
plantcodes.resolved
>     Code                           Scientific.Name
```

```
> 1 ARTRW8 Artemisia tridentata ssp. wyomingensis
> 2   LEPU                 Leptodactylon pungens
> 3  PHLO2                      Phlox longifolia
> 4   ABCO                        Abies concolor
> 5   PIEN                     Picea engelmannii
```

Note how the `usda_resolve()` function assigns a new column to the data frame 'Scientific.Name'.

This function can also be used in reverse, returning the USDA Plant Code for a given scientific name. While not frequently necessary, it can be useful for generating a species codes list for future survey protocols or applications.

```
## Resolve Taxonomic Name to USDA Plant Code
taxanames <- data.frame(Taxa = c("Artemisia tridentata","Leptodactylon pungens","Picea",
                        "Phlox longifolia","Populus tremuloides"))
taxanames.usda <- fsvm::usda_resolve(dat = taxanames, target = "Taxa", resolve = "n2c")
head(taxanames.usda)
>                    Taxa Symbol
> 1  Artemisia tridentata  ARTR2
> 2 Leptodactylon pungens LIPU11
> 3                 Picea  PICEA
> 4      Phlox longifolia  PHLO2
> 5   Populus tremuloides  POTR5
```

Note how the new column this time is 'Symbol'. Keep in mind that the function accepts synonym symbols when resolving "c2n" but will only return the most up-to-date symbol when resolving "n2c" (see 'Leptodactylon pungens').

Rectifying Taxonomic Names (ITIS - TSN) Next, we examine how taxonomic names are rectified and assigned to modeling groups. "G1" corresponds with taxonomic genus, "G2" is species binomial, "G3" is subspecies trinomial, and "G4" is variety trinomial. As previously stated, the unique taxonomic serial number (TSN) via the Integrated Taxonomic Information System (ITIS.gov) is provided for each modeling group.

```
## Rectify taxonomic names
fsvm_dummy_rectified <- fsvm::rectify_taxa(dat = fsvm_dummy,
                                           scientific.name = "Scientific_Name_Complete")
> [1] "Rectifying Taxonomic Names..."
> [1] "Assigning Model Groups by Accepted Name..."
> [1] "Getting G1 Taxonomic Serial Number..."
> [1] "Getting G2 Taxonomic Serial Number..."
> [1] "Getting G3 Taxonomic Serial Number..."
> [1] "Getting G4 Taxonomic Serial Number..."
> [1] "Joining Taxonomic Serial Numbers and Accepted Names to Data..."
head(fsvm_dummy_rectified)
>          TranKey       PlotKey        Source DataType Smpl_Yr Easting Northing
> 1: J_Aster_Plot4 J_Aster_Plot4 Jessica Aster    COVER    2014 2327731  1711541
> 2: J_Aster_Plot4 J_Aster_Plot4 Jessica Aster    COVER    2014 2327731  1711541
> 3: J_Aster_Plot4 J_Aster_Plot4 Jessica Aster    COVER    2014 2327731  1711541
> 4: J_Aster_Plot4 J_Aster_Plot4 Jessica Aster    COVER    2014 2327731  1711541
> 5: J_Aster_Plot4 J_Aster_Plot4 Jessica Aster    COVER    2014 2327731  1711541
> 6: J_Aster_Plot4 J_Aster_Plot4 Jessica Aster    COVER    2014 2327731  1711541
>    Scientific_Name_Complete Prcnt_C Count         Accepted.Name          G1
> 1:          Pinus ponderosa       1     1       Pinus ponderosa       Pinus
> 2:    Pseudotsuga menziesii      40     1 Pseudotsuga menziesii Pseudotsuga
```

```
> 3:    Amelanchier alnifolia      10     1 Amelanchier alnifolia Amelanchier
> 4:      Holodiscus discolor      10     1   Holodiscus discolor  Holodiscus
> 5:         Lonicera ciliosa       1     1      Lonicera ciliosa    Lonicera
> 6:           Mahonia repens       1     1       Berberis repens    Berberis
>                       G2 G3 G4 G1.TSN G2.TSN G3.TSN G4.TSN
> 1:       Pinus ponderosa NA NA  18035 183365     NA   <NA>
> 2: Pseudotsuga menziesii NA NA 183418 183424     NA   <NA>
> 3: Amelanchier alnifolia NA NA  25108  25109     NA   <NA>
> 4:   Holodiscus discolor NA NA  25175  25177     NA   <NA>
> 5:      Lonicera ciliosa NA NA  35281  35288     NA   <NA>
> 6:       Berberis repens NA NA  18814  18832     NA   <NA>
```

Note the differences between the provided scientific name and the 'Accepted.Name' output. The 'Accepted.Name' should be used as the input for resolving IFWIS Taxon ID.

Resolving IFWIS Taxonomic ID This function will return IFWIS Taxon ID for a given plant species. It is recommended to use the species name provided by the `rectify_taxa` function ('Accepted.Name'). If your dataset includes fungus species in addition to vascular plants, the function also accepts "c('Plantae','Fungi')" as an argument for 'kingdom'.

```
## Resolve IFWIS Taxon IDs
fsvm_dummy_ifwis <- fsvm::ifwis_resolve(dat = fsvm_dummy_rectified,
                                        sci.name = "Accepted.Name",
                                        kingdom = "Plantae")
> [1] "Connecting to IFWIS Taxonomic Database..."
> [1] "Matching Scientific Names..."
head(fsvm_dummy_ifwis)[,c(1,8,11)]
>          TranKey Scientific_Name_Complete         Accepted.Name
> 1: J_Aster_Plot4          Pinus ponderosa       Pinus ponderosa
> 2: J_Aster_Plot4    Pseudotsuga menziesii Pseudotsuga menziesii
> 3: J_Aster_Plot4    Amelanchier alnifolia Amelanchier alnifolia
> 4: J_Aster_Plot4      Holodiscus discolor   Holodiscus discolor
> 5: J_Aster_Plot4         Lonicera ciliosa      Lonicera ciliosa
> 6: J_Aster_Plot4    Physocarpus malvaceus Physocarpus malvaceus
```

Download Survey123 data (beta)

Beta version of a potentially useful function for downloading Simple Veg Survey data directly from Survey123 ArcGIS Online. However, a valid AGOL account is required. This is difficult since IFWIS only has a limited number of account keys. Still in development, but it will rely on a Python script called by 'reticulate' (similar to `py_extract_quadpolyID`).

```
## Beta version of `getSVS` function
# Need a valid AGOL account (IFWIS)
#getSVS(py.path = "C:/Python27/ArcGIS10.6/python.exe",
#       featureService_ID = ,
#       output_format = "CSV",
#       download_folder = "C:/Temp/",
#       agol.username = ,
#       agol.password = ,
#       download_url = ,
#       filename = ,
#       token = ,
```

```
#        folder = )
```

[forthcoming] ### Associate eCognition polygons with field data Associating the covariate data stored within the eCognition polygons with the field collected species presence/percent cover data is an essential component of the fine scale vegetation model. As there are ~40 million individual eCognition polygons (>1Tb) stored within 75 100k USGS quadrangles, this is an extremely data-intensive process. In order to streamline this process, the function `py_extract_quadpolyID` leverages ArcGIS functions within a Python script which is called by the Python-to-R translator `reticulate`. However, ArcGIS uses a 32-bit architecture which means this function needs to be run in a 32-bit architecture of R. While not difficult, it is an essential step (see 'Getting Started' for instructions on switching between architectures). This step can be skipped only if you have a 64-bit Geoprocessor installed for your version of ArcMap or ArcPro, but these are not widely available. Given the need to switch between architectures on top of the already intensive process, this function can be finicky. The size of the polygon database also makes this a slow process, even with the assistance of Python and ArcMap, with speed decreasing as the the number of 100k quads with data increases. The dummy data set will take approximately 1 hour to complete (the full field data set will take at least 8 hours).

The following parameters are required arguments for associating quadpolyID with field data:

- A file path leading to a shapefile or geodatabase of field data (must convert if in a csv).
- A file path to a valid geodatabase. If you need output stored in a new geodatabase, you must provide a name to the 'newgdb.name' parameter.
  - *Troubleshooting Tip*: Errors often stem from issues with closing or opening the geodatabase. If the function fails at first (due to misspellings, etc.), it is a best practice to save your session and completely restart R. You may also need to delete the new geodatabase or switch your 'output.gdb.path' to navigate to the geodatabase you created.
- A file path to an output folder to store .dbfs of your intermediate output. The folder does not need to exist, the function will create the folder you name in the file path.
  - *Troubleshooting Tip*: While the Python script powering this function is supposed to overwrite files in the output folder and geodatabase, it may be neccessary to delete the function created folder and geodatabase and allow them to be recreated from scratch.
- A name for your merged output .dbf (Must contain .dbf in the name).

There are also several optional arguments for dealing with different types of field data formats:

- A name of for the output .RData file. The default of 'None' automatically populates the name based on whether the quad path is 'ID Only' or 'Covariates', but it also accepts a customized file name as long as it contains '.RData'.
- The name of an intercept feature. This is only neccessary if the field data is stored in a file geodatabase and corresponds with the target shapefile containing the field data. If the field data file path does not lead to a geodatabase (a valid shapefile), then leave argument of 'None'.
  - *Note*: File Geodatabases have not yet been tested for this function. If encountering troubles, it may be best to store the target field data as a regular shapefile.
- An output geodatabase name. This is only necessary if the 'output.gdb.path' does not lead to a valid geodatabase. A new geodatabase name must contain '.gdb'. If an output geodatabase already exists and is included in the output path, then this parameter can be left at 'None'.
- Selection of quadpolygons. The default of "None" will include **ALL** 75 100k quadpolygons in the loop. In order to select a valid subset of quadpolygons, this argument must use an object created from `set_extent_manual` in order to properly correspond with the names.

  - *Troubleshooting Tip*: UID of a 100k quadpolygon alone will not work for this parameter, it must contain the 'q' in front of the UID with the '_100blk'. The name of the quadpolygons in 'Covariates' varies slightly from this and may not work correctly yet (bug fixes for this are forthcoming). It may be neccessary to manually alter the character strings output from `select_quads` to properly correspond with 'Covariates' quadpolygons.

- Whether or not to export the merged dbf file. This should generally be 'TRUE', but 'FALSE' may help if function keeps failing.
  - *Troubleshooting Tip*: One of the major downsides of needing to use 32-bit architecture for this function is the reduced system memory for merging output dbfs. It may be neccessary in certain circumstances to leave the 'export' parameter as 'FALSE', complete the function, switch back to 64-bit and restart, then merge the dfs in 64-bit R. (Bug fixes and imporved workflow are forthcoming).

*In order to speed up this process, it helps to restrict the which 100k quads are looped through in the function. Two functions are included in this package to help with this: `select_quads` and `set_extent_manual`. `select_quads` is used to identify which eCognition polygons intersect with the field data and `set_extent_manual` returns the 100k quads in a format usable in the `quad.sel` parameter of `py_extract_quadpolyID`. `set_extent_manual` can also identify 100k quads corresponding with IDFG Regions, Game Management Units, or 24k quads.

```
## Select QuadPolygons (100k)
#UIDs of eCognition polygons
quads.uid <- fsvm::select_quads(dat = fsvm_dummy, x = "Easting", y = "Northing")
> OGR data source with driver: ESRI Shapefile
> Source: "C:\Users\rritson\Documents\R\win-library\4.0\fsvm\esri\100kquads", layer: "quad100k_proj"
> with 75 features
> It has 2 fields
> Warning: attribute variables are assumed to be spatially constant throughout all
> geometries
unique(quads.uid)
>  [1] "48116a1" "47116e1" "47115a1" "46116e1" "46115a1" "45115e1" "45116a1"
>  [8] "45115a1" "45114a1" "44114a1" "44113a1" "44112a1" "43114e1" "43113e1"
> [15] "43116a1" "43115a1" "43113a1" "42116e1" "43111a1" "42116a1" "42113a1"

#Set extent (get names of 100k quadpolygons)
quad.sel <- fsvm::set_extent_manual(extent = "UID_100k", selections = quads.uid)
quad.sel
>  [1] "q48116a1_100blk" "q47116e1_100blk" "q47115a1_100blk" "q46116e1_100blk"
>  [5] "q46115a1_100blk" "q45115e1_100blk" "q45116a1_100blk" "q45115a1_100blk"
>  [9] "q45114a1_100blk" "q44114a1_100blk" "q44113a1_100blk" "q44112a1_100blk"
> [13] "q43114e1_100blk" "q43116a1_100blk" "q43115a1_100blk" "q42116e1_100blk"
> [17] "q42116a1_100blk" "q43113e1_100blk" "q43113a1_100blk" "q42113a1_100blk"
> [21] "q43111a1_100blk"

## Additional examples
#IDFG Game management unit
GMU_10A_quads <- fsvm::set_extent_manual(extent = "GMU", selections = "10A")
GMU_10A_quads
> [1] "q47116a1_100blk" "q47115a1_100blk" "q46116e1_100blk" "q46115e1_100blk"
> [5] "q46116a1_100blk" "q46115a1_100blk"

#IDFG Region
R3_quads <- fsvm::set_extent_manual(extent = "Region", selections = "3")
```

```
R3_quads
>  [1] "q46116a1_100blk" "q45116e1_100blk" "q45115e1_100blk" "q45116a1_100blk"
>  [5] "q45115a1_100blk" "q45114e1_100blk" "q45114a1_100blk" "q45113e1_100blk"
>  [9] "q45113a1_100blk" "q44116e1_100blk" "q44115e1_100blk" "q44116a1_100blk"
> [13] "q44115a1_100blk" "q44114e1_100blk" "q44114a1_100blk" "q44117e1_100blk"
> [17] "q44117a1_100blk" "q43116e1_100blk" "q43115e1_100blk" "q43114e1_100blk"
> [21] "q43117e1_100blk" "q43116a1_100blk" "q43115a1_100blk" "q42116e1_100blk"
> [25] "q42115e1_100blk" "q43117a1_100blk" "q42117e1_100blk" "q42116a1_100blk"
> [29] "q42115a1_100blk" "q41116e1_100blk" "q41115e1_100blk" "q42117a1_100blk"
> [33] "q41117e1_100blk" "q43114a1_100blk" "q42114e1_100blk" "q42114a1_100blk"
> [37] "q41114e1_100blk"
```

Take notice of how many 100k quadpolygons are included in each list. For the fsvm_dummy data, observations are only located in 21 of the 75 100k quadpolygons. This removes nearly two-thirds of the quads to loop through, which will speed up the quadpolyID extraction process. Smaller extents like GMUs intersect fewer quads (6 in the case of GMU 10A) while larger extents like IDFG Region encompass more quads (37 in the case of R3). As previously mentioned, larger extents with larger datasets will greatly increase processing time. For example, a testing dataset of ~200,000 observations (Consortium of Pacific Northwest Herbaria observations) required approximately 8 hours to complete to cycle through all 75 100k quadpolygons.

```
## Extract QuadPolyID of eCognition polygons for field data
#Step 1: Select quad polygons (`select_quads` nested within `set_extent_manual`)
quad.sel <- fsvm::set_extent_manual(extent = "UID_100k",
            selections = fsvm::select_quads(dat = fsvm_dummy, x = "Easting", y = "Northing"))
> OGR data source with driver: ESRI Shapefile
> Source: "C:\Users\rritson\Documents\R\win-library\4.0\fsvm\esri\100kquads", layer: "quad100k_proj"
> with 75 features
> It has 2 fields
> Warning: attribute variables are assumed to be spatially constant throughout all
> geometries

#Step 2: Write Field data shapefile
fsvm_dummy_spatial <- sp::SpatialPointsDataFrame(data = fsvm_dummy,
coords = fsvm_dummy[,c("Easting","Northing")],
proj4string = sp::CRS("+proj=tmerc +lat_0=42 +lon_0=-114 +k=0.9996 +x_0=2500000 +y_0=1200000 +ellps=GRS8
> Warning in showSRID(uprojargs, format = "PROJ", multiline = "NO", prefer_proj
> = prefer_proj): Discarded datum Unknown based on GRS80 ellipsoid in Proj4
> definition
rgdal::writeOGR(fsvm_dummy_spatial,
                dsn = "A:/Fine scale vegetation analysis/fsvm_package",
                layer = "fsvm_dummy",
                driver = "ESRI Shapefile", overwrite_layer = T)
> Warning in rgdal::writeOGR(fsvm_dummy_spatial, dsn = "A:/Fine scale vegetation
> analysis/fsvm_package", : Field names abbreviated for ESRI Shapefile driver

#Step 3: Extract Quadpoly IDs
dummy_quadpolyID <- py_extract_quadpolyID(py.path = "C:/Python27/ArcGIS10.6/python.exe",
                    fielddata.path = "A:/Fine scale vegetation analysis/fsvm_package/fsvm_dummy.shp", #
                    output.gdb.path = here::here(),
                    output.folder.path = here::here("Output"),
                    output.dbf = "Dummy_Merge.dbf",
                    output.RData = "None",
                    intercept.feature = "None",
                    newgdb.name = "Output.gdb",
```

```
                    quad.sel = quad.sel,
                    export = T)
> [1] "FieldDataPoints_QuadPolyID.RData will be stored in C:/Users/rritson/Documents/fsvm/Output"
> [1] "Initializing Python and Loading Function"
> [1] "Reticulating QuadPolyID Extraction..."
> begin script on 2021-05-26 at 15:57:20
> listing selected quadpolygons
> loading target shapefile
> loading feature q48116a1_100blk
> selecting ecog polygons that intersect with target shapefile
> exporting
> q48116a1_100blk_select_pts exported at 16:11:08
> q48116a1_100blk joined with target shapefile at 16:11:13
> q48116a1_100blk dbf file exported at 16:11:15
> loading feature q47116e1_100blk
> selecting ecog polygons that intersect with target shapefile
> exporting
> q47116e1_100blk_select_pts exported at 16:12:21
> q47116e1_100blk joined with target shapefile at 16:12:23
> q47116e1_100blk dbf file exported at 16:12:24
> loading feature q47115a1_100blk
> selecting ecog polygons that intersect with target shapefile
> exporting
> q47115a1_100blk_select_pts exported at 16:12:47
> q47115a1_100blk joined with target shapefile at 16:12:49
> q47115a1_100blk dbf file exported at 16:12:49
> loading feature q46116e1_100blk
> selecting ecog polygons that intersect with target shapefile
> exporting
> q46116e1_100blk_select_pts exported at 16:12:56
> q46116e1_100blk joined with target shapefile at 16:12:57
> q46116e1_100blk dbf file exported at 16:12:58
> loading feature q46115a1_100blk
> selecting ecog polygons that intersect with target shapefile
> exporting
> q46115a1_100blk_select_pts exported at 16:13:04
> q46115a1_100blk joined with target shapefile at 16:13:06
> q46115a1_100blk dbf file exported at 16:13:07
> loading feature q45115e1_100blk
> selecting ecog polygons that intersect with target shapefile
> exporting
> q45115e1_100blk_select_pts exported at 16:13:13
> q45115e1_100blk joined with target shapefile at 16:13:15
> q45115e1_100blk dbf file exported at 16:13:17
> loading feature q45116a1_100blk
> selecting ecog polygons that intersect with target shapefile
> exporting
> q45116a1_100blk_select_pts exported at 16:13:23
> q45116a1_100blk joined with target shapefile at 16:13:25
> q45116a1_100blk dbf file exported at 16:13:25
> loading feature q45115a1_100blk
> selecting ecog polygons that intersect with target shapefile
> exporting
```

```
> q45115a1_100blk_select_pts exported at 16:15:00
> q45115a1_100blk joined with target shapefile at 16:15:02
> q45115a1_100blk dbf file exported at 16:15:03
> loading feature q45114a1_100blk
> selecting ecog polygons that intersect with target shapefile
> exporting
> q45114a1_100blk_select_pts exported at 16:15:09
> q45114a1_100blk joined with target shapefile at 16:15:11
> q45114a1_100blk dbf file exported at 16:15:11
> loading feature q44114a1_100blk
> selecting ecog polygons that intersect with target shapefile
> exporting
> q44114a1_100blk_select_pts exported at 16:15:18
> q44114a1_100blk joined with target shapefile at 16:15:21
> q44114a1_100blk dbf file exported at 16:15:21
> loading feature q44113a1_100blk
> selecting ecog polygons that intersect with target shapefile
> exporting
> q44113a1_100blk_select_pts exported at 16:15:38
> q44113a1_100blk joined with target shapefile at 16:15:41
> q44113a1_100blk dbf file exported at 16:15:41
> loading feature q44112a1_100blk
> selecting ecog polygons that intersect with target shapefile
> exporting
> q44112a1_100blk_select_pts exported at 16:18:00
> q44112a1_100blk joined with target shapefile at 16:18:02
> q44112a1_100blk dbf file exported at 16:18:03
> loading feature q43114e1_100blk
> selecting ecog polygons that intersect with target shapefile
> exporting
> q43114e1_100blk_select_pts exported at 16:18:10
> q43114e1_100blk joined with target shapefile at 16:18:12
> q43114e1_100blk dbf file exported at 16:18:13
> loading feature q43116a1_100blk
> selecting ecog polygons that intersect with target shapefile
> exporting
> q43116a1_100blk_select_pts exported at 16:18:19
> q43116a1_100blk joined with target shapefile at 16:18:21
> q43116a1_100blk dbf file exported at 16:18:21
> loading feature q43115a1_100blk
> selecting ecog polygons that intersect with target shapefile
> exporting
> q43115a1_100blk_select_pts exported at 16:18:28
> q43115a1_100blk joined with target shapefile at 16:18:30
> q43115a1_100blk dbf file exported at 16:18:31
> loading feature q42116e1_100blk
> selecting ecog polygons that intersect with target shapefile
> exporting
> q42116e1_100blk_select_pts exported at 16:20:17
> q42116e1_100blk joined with target shapefile at 16:20:19
> q42116e1_100blk dbf file exported at 16:20:19
> loading feature q42116a1_100blk
> selecting ecog polygons that intersect with target shapefile
```

```
> exporting
> q42116a1_100blk_select_pts exported at 16:20:27
> q42116a1_100blk joined with target shapefile at 16:20:29
> q42116a1_100blk dbf file exported at 16:20:29
> loading feature q43113e1_100blk
> selecting ecog polygons that intersect with target shapefile
> exporting
> q43113e1_100blk_select_pts exported at 16:20:35
> q43113e1_100blk joined with target shapefile at 16:20:36
> q43113e1_100blk dbf file exported at 16:20:37
> loading feature q43113a1_100blk
> selecting ecog polygons that intersect with target shapefile
> exporting
> q43113a1_100blk_select_pts exported at 16:21:01
> q43113a1_100blk joined with target shapefile at 16:21:03
> q43113a1_100blk dbf file exported at 16:21:04
> loading feature q42113a1_100blk
> selecting ecog polygons that intersect with target shapefile
> exporting
> q42113a1_100blk_select_pts exported at 16:21:22
> q42113a1_100blk joined with target shapefile at 16:21:23
> q42113a1_100blk dbf file exported at 16:21:24
> loading feature q43111a1_100blk
> selecting ecog polygons that intersect with target shapefile
> exporting
> q43111a1_100blk_select_pts exported at 16:21:30
> q43111a1_100blk joined with target shapefile at 16:21:32
> q43111a1_100blk dbf file exported at 16:21:33
> [1] "Formatting and Exporting Output"

head(dummy_quadpolyID)
>                  TranKey                    PlotKey   Source DataTyp Smpl_Yr
> 1 IDFG_2019_q42113c1_3152 IDFG_2019_q42113c1_3152_0.5 IDFG_LPI     LPI    2019
> 2 IDFG_2019_q42113c1_3152   IDFG_2019_q42113c1_3152_1 IDFG_LPI     LPI    2019
> 3 IDFG_2019_q42113c1_3152 IDFG_2019_q42113c1_3152_1.5 IDFG_LPI     LPI    2019
> 4 IDFG_2019_q42113c1_3152   IDFG_2019_q42113c1_3152_2 IDFG_LPI     LPI    2019
> 5 IDFG_2019_q42113c1_3152 IDFG_2019_q42113c1_3152_2.5 IDFG_LPI     LPI    2019
> 6 IDFG_2019_q42113c1_3152   IDFG_2019_q42113c1_3152_3 IDFG_LPI     LPI    2019
>   Easting                Scn_N_C Prcnt_C Count Shape_Leng Shape_Area
> 1 2578976   Eriogonum microthecum       0     1        556       3321
> 2 2578977         Bromus tectorum       0     1        556       3321
> 3 2578977             Poa secunda       0     1        556       3321
> 4 2578978 Pseudoroegneria spicata       0     1        556       3321
> 5 2578978 Pseudoroegneria spicata       0     1        556       3321
> 6 2578978                    <NA>       0     1        556       3321
>     QuadPoly_ID Northing     quad
> 1 q42113c1_3152  1239915 q42113c1
> 2 q42113c1_3152  1239916 q42113c1
> 3 q42113c1_3152  1239916 q42113c1
> 4 q42113c1_3152  1239916 q42113c1
> 5 q42113c1_3152  1239916 q42113c1
> 6 q42113c1_3152  1239916 q42113c1
```

Classify Eco-Region of field data

In order to improve machine learning training and predictions, the ecological region of a survey location can be a helpful additional covariate. This is easily done by assigning Bailey's Eco-Regions to field survey locations using a shapefile stored withing the package which is called by the following function.

```
## Assign Bailey's Eco-Regions to field data
#dummy_quadpolyID <- dummy_quadpolyID[c(14,15,2:7,16,9:13)]
dummy_eco <- fsvm::assign_ecoregions(dummy_quadpolyID)
> Warning in OGRSpatialRef(dsn, layer, morphFromESRI = morphFromESRI, dumpSRS =
> dumpSRS, : Discarded datum Not_specified_based_on_Clarke_1866_ellipsoid in Proj4
> definition: +proj=aea +lat_0=41 +lon_0=-117 +lat_1=43 +lat_2=48 +x_0=700000
> +y_0=0 +ellps=clrk66 +units=m +no_defs
> OGR data source with driver: ESRI Shapefile
> Source: "C:\Users\rritson\Documents\R\win-library\4.0\fsvm\esri\ecoregions", layer: "Baileys_ecoregio
> with 330 features
> It has 7 fields
> Warning: attribute variables are assumed to be spatially constant throughout all
> geometries
head(dummy_eco)
>           TranKey       PlotKey              Source  DataTyp Smpl_Yr Easting
> 124  CPNWH_864236  CPNWH_864236  CPNWH_ID_Herbarium Presence    1932 2289608
> 125  CPNWH_128442  CPNWH_128442 CPNWH_WTU_Herbarium Presence    1957 2302932
> 126   CPNWH_82824   CPNWH_82824 CPNWH_WTU_Herbarium Presence    1994 2279426
> 89   CPNWH_983134  CPNWH_983134  CPNWH_ID_Herbarium Presence    1938 2361936
> 90  J_Aster_Plot4 J_Aster_Plot4       Jessica Aster    COVER    2014 2327731
> 91  J_Aster_Plot4 J_Aster_Plot4       Jessica Aster    COVER    2014 2327731
>                                      Scn_N_C Prcnt_C Count Shape_Leng
> 124                          Carex exsiccata       0     1        440
> 125                        Prunella vulgaris       0     1       1890
> 126 Heuchera grossulariifolia var. tenuifolia      0     1        368
> 89                      Penstemon attenuatus       0     1       2060
> 90                           Pinus ponderosa       1     1       1070
> 91                      Pseudotsuga menziesii      40     1       1070
>     Shape_Area    QuadPoly_ID Northing     quad ECOCODE
> 124        683 q47116f7_29244  1835189 q47116f7   M333A
> 125       5760 q48116a6_39784  1870544 q48116a6   M333A
> 126       1269 q48116b8_28330  1889310 q48116b8   M333A
> 89        6594  q46115d7_3645  1700402 q46115d7   M333D
> 90        3822 q46116e2_13759  1711541 q46116e2   M333D
> 91        3822 q46116e2_13759  1711541 q46116e2   M333D
```

Get Covariate data

Associating eCognition covariate data with the field data is essential for modeling. This function queries the folder 'Covariates.RDS' on the network drive. This contains *.rds files for each 100k USGS quad in order to speed up the process of accessing covariate data. This function queries the files to extract covariate data for each unique QuadPoly_ID from the field data (output of `py_extract_quadpolyID`). The function formats the the covariates for modeling and drops 'NA' values (total data points may decrease; looking into getting data for these eCognition polygons).

```
## Extract eCognition QuadPolygon Covariates for field data
dummy_covs <- fsvm::getCovariates(dat = dummy_eco, rm.na = F)
> [1] "Processing: 1 % complete"
```

```
> [1] "Processing: 3 % complete"
> [1] "Processing: 4 % complete"
> [1] "Processing: 5 % complete"
> [1] "Processing: 7 % complete"
> [1] "Processing: 8 % complete"
> [1] "Processing: 9 % complete"
> [1] "Processing: 11 % complete"
> [1] "Processing: 12 % complete"
> [1] "Processing: 13 % complete"
> [1] "Processing: 15 % complete"
> [1] "Processing: 16 % complete"
> [1] "Processing: 17 % complete"
> [1] "Processing: 19 % complete"
> [1] "Processing: 20 % complete"
> [1] "Processing: 21 % complete"
> [1] "Processing: 23 % complete"
> [1] "Processing: 24 % complete"
> [1] "Processing: 25 % complete"
> [1] "Processing: 27 % complete"
> [1] "Processing: 28 % complete"
> [1] "Processing: 29 % complete"
> [1] "Processing: 31 % complete"
> [1] "Processing: 32 % complete"
> [1] "Processing: 33 % complete"
> [1] "Processing: 35 % complete"
> [1] "Processing: 36 % complete"
> [1] "Processing: 37 % complete"
> [1] "Processing: 39 % complete"
> [1] "Processing: 40 % complete"
> [1] "Processing: 41 % complete"
> [1] "Processing: 43 % complete"
> [1] "Processing: 44 % complete"
> [1] "Processing: 45 % complete"
> [1] "Processing: 47 % complete"
> [1] "Processing: 48 % complete"
> [1] "Processing: 49 % complete"
> [1] "Processing: 51 % complete"
> [1] "Processing: 52 % complete"
> [1] "Processing: 53 % complete"
> [1] "Processing: 55 % complete"
> [1] "Processing: 56 % complete"
> [1] "Processing: 57 % complete"
> [1] "Processing: 59 % complete"
> [1] "Processing: 60 % complete"
> [1] "Processing: 61 % complete"
> [1] "Processing: 63 % complete"
> [1] "Processing: 64 % complete"
> [1] "Processing: 65 % complete"
> [1] "Processing: 67 % complete"
> [1] "Processing: 68 % complete"
> [1] "Processing: 69 % complete"
> [1] "Processing: 71 % complete"
> [1] "Processing: 72 % complete"
```

```
> [1] "Processing: 73 % complete"
> [1] "Processing: 75 % complete"
> [1] "Processing: 76 % complete"
> [1] "Processing: 77 % complete"
> [1] "Processing: 79 % complete"
> [1] "Processing: 80 % complete"
> [1] "Processing: 81 % complete"
> [1] "Processing: 83 % complete"
> [1] "Processing: 84 % complete"
> [1] "Processing: 85 % complete"
> [1] "Processing: 87 % complete"
> [1] "Processing: 88 % complete"
> [1] "Processing: 89 % complete"
> [1] "Processing: 91 % complete"
> [1] "Processing: 92 % complete"
> [1] "Processing: 93 % complete"
> [1] "Processing: 95 % complete"
> [1] "Processing: 96 % complete"
> [1] "Processing: 97 % complete"
> [1] "Processing: 99 % complete"
> [1] "Processing: 100 % complete"
head(dummy_covs)
>        quad       ele       slp casp sasp twi lcv      sri tpi     minpr
> 1: q42113c1 1702.6847 15.790772   -9    1   4   0 697605.2   0 22.263939
> 2: q42113c5 2215.5897 11.603037   -9    0   7   0 738105.1   0 17.860085
> 3: q42116b5 1500.1282 27.443766   -5   -9   4   0 600102.9   0  6.442032
> 4: q42116f6 1681.2645  5.388156    0   -9   5   0 675785.1   0 11.394581
> 5: q42116h2  888.3927  2.312325    9    9   7   0 629943.8   0  6.714651
> 6: q43111d2 1865.7155 37.202021    1   -9   5   0 527537.5   0 37.127990
>        maxpr     tapr      mintp    maxtp      aws     clay     sand    silt
> 1:  60.78102 458.7241 -10.172675 29.15298 3.420000 15.20000 26.70000 58.0000
> 2: 121.05163 826.0625  -6.674450 24.79799 3.680000 17.80000 27.70000 54.5000
> 3:  37.52936 274.5854  -7.123909 29.05396 3.150000 21.80000 37.60000 40.6000
> 4:  72.31452 475.6999  -9.492475 27.95991 3.310000 23.50000 39.50000 36.9000
> 5:  27.27079 189.4750  -5.254529 33.33613 3.069796 11.54168 69.31473 19.1436
> 6:  92.90858 794.5859  -9.839847 25.92891 4.070000 12.20000 44.10000 43.7000
>          cec       d2r ph         om    caco3  tsf ff         tc sc nass dev
> 1: 13.400000  67.00000  7  2.2900000 9.000000    0  0  0.9430894  1  152  52
> 2: 16.400000  94.00000  6  2.7000000 1.000000 1996  1 27.9548564  0  152  52
> 3: 15.900000   0.00000  7  1.2000000 0.000000    0  0  0.0000000  3  152  52
> 4: 20.400000  56.00000  6  1.8900000 0.000000    0  0  0.0000000  2  152  52
> 5:  8.466103  52.67783  7  0.6351139 3.764228    0  0  0.0000000  0  176  71
> 6: 14.500000 127.00000  6 13.6000004 0.000000    0  0 44.2475426  0  142  42
>       QuadPoly_ID     water_m2    shadow_m2 bareground_m2    mgrass_m2
> 1:  q42113c1_3152 0.0003373819 0.0000000000   0.098515520 0.0084345479
> 2:  q42113c5_6741 0.0000000000 0.0004827653   0.006855267 0.0026069325
> 3:  q42116b5_2041 0.0000000000 0.0683698650   0.269850486 0.1875453622
> 4: q42116f6_36527 0.0000000000 0.0127459367   0.066809239 0.0005988024
> 5:  q42116h2_6042 0.0000000000 0.0000000000   0.019307273 0.4701592983
> 6:  q43111d2_2802 0.0046156934 0.3032309854   0.090909091 0.0000000000
>       xgrass_m2    mshrub_m2   xshrub_m2 conifer_m2     decid_m2 agriculture_m2
> 1: 0.0084345479 1.373144e-01 0.354925776 0.38765182 0.0043859649     0.00000000
> 2: 0.0026069325 2.878247e-01 0.013324322 0.42550932 0.2607898040     0.00000000
```

```
> 3: 0.1875453622 2.268834e-01 0.005080563 0.05443461 0.0002903179     0.00000000
> 4: 0.0005988024 5.223268e-01 0.369118905 0.02514970 0.0026518392     0.00000000
> 5: 0.4701592983 2.315854e-06 0.007580484 0.00000000 0.0000000000     0.03279133
> 6: 0.0000000000 1.605459e-01 0.004415011 0.42002810 0.0162552679     0.00000000
>    developed_m2 ytsf      ele2        slp2 casp2 twi2         sri2     minpr2
> 1:            0 2021 2899135.0  249.348494    81   16 486652996051  495.68298
> 2:            0   25 4908837.7  134.630456    81   49 544799138107  318.98262
> 3:            0 2021 2250384.5  753.160274    25   16 360123491019   41.49978
> 4:            0 2021 2826650.4   29.032227     0   25 456685518529  129.83647
> 5:            0 2021  789241.5    5.346845    81   49 396829174671   45.08653
> 6:            0 2021 3480894.2 1383.990389     1   25 278295823728 1378.48763
>       maxtp2      aws2    clay2     sand2     silt2      cec2      d2r2 ph2
> 1:  849.8965 11.696401 231.0400  712.890 3364.0000 179.5600  4489.000  49
> 2:  614.9403 13.542400 316.8400  767.290 2970.2500 268.9600  8836.000  36
> 3:  844.1327  9.922501 475.2400 1413.760 1648.3599 252.8100     0.000  49
> 4:  781.7565 10.956100 552.2500 1560.250 1361.6101 416.1600  3136.000  36
> 5: 1111.2974  9.423647 133.2103 4804.531  366.4773  71.6749  2774.954  49
> 6:  672.3084 16.564901 148.8400 1944.810 1909.6901 210.2500 16129.000  36
>            om2   caco32   ytsf2 Real_Shape_Area
> 1:   5.2440998 81.00000 4084441            2964
> 2:   7.2900003  1.00000     625           10357
> 3:   1.4400001  0.00000 4084441            6889
> 4:   3.5720999  0.00000 4084441           11690
> 5:   0.4033697 14.16941 4084441         4318062
> 6: 184.9600104  0.00000 4084441            4983
```

Final Formatting of Field Data for Machine Learning

Formats field data so it can be used for training and machine learning models. Field data needs to be run through `rectify_taxa`, then `py_extract_quadpolyID`, then `assign_ecoregion` before passing the data to `as_fsvm`

```
###Format Data
#fsvm_field <- dummy_eco
#fsvm_field$Include <- 1
#fsvm_field$Plot_Ar <- "NA"
#fsvm_field$DataType <- ifelse(fsvm_field$DatTyp == "Presence","LPI",fsvm_field$DatTyp)
#df.fsvm <- fsvm::as_fsvm(fielddata = fsvm_field, covariates = dummy_covs, DataType = "DataType",
#                         Prcnt_C = "Prcnt_C", ModelCode = "Scn_N_C", Plot_Ar = "Plot_Ar",
#                         genus = "G1")
#head(df.fsvm)
```

Data is now ready for machine learning modeling.

[next steps forthcoming]

Creating Machine Learning Models

Functions required for making the models ('Run_Models_Template' suite)

```
#fsvm_train
#get_forage_models
```

Predicting from Models

Functions required for getting predictions from models ('Predict_Models_Template' suite)

```
#prep_pred_covs
#fsum_predict
#getPredictions
```

Summarizing Results

Summarize and map model predictions ('Summary' and 'Centroid_Maps' Templates suite)

```
#getSummary
#getCentroidMap
```