# TESTER for vignette

Erika Lin

2025-07-09

**Last updated 2025-07-09**

This workflow demonstrates how the living shoreline suitability meta-analysis model can be updated with new local data, from initial data importing and cleaning, to mapped figures of shoreline suitability. This meta-analysis model can also be used to predict the suitability and probability of success of living shoreline restoration from the geospatial data. The model and package were built for the 2024 Florida Statewide Coastal Restoration Guide (SCRG) project, and are based on the data and results of existing Living Shoreline Suitability Model (LSSM) studies. The package is maintained by the UCF Ecoinformatics Lab. Materials can be accessed at https://github.com/ecoinformatic/SCRG/.

## Setup

To work with spatial data for this meta-analysis model, the `sf` and `ecoinfoscrg` R packages are needed. Other packages may also be helpful to load for data handling and visualization.

```
# If the packages are not installed, run the following:
# install.packages("sf")
# install.packages("dplyr")
# install.packages("ggplot2")
# install.packages("devtools")  # if `devtools` was not previously installed
# devtools::install_github("https://github.com/ErikaYLin/ecoinfoscrgTEST/")

# Load ecoinfoscrg
library(ecoinfoscrgTEST)

# Load other requisite packages
sf::sf_use_s2(FALSE)  # for working with planar data
```

```
## Spherical geometry (s2) switched off
```

```
library(sf)  # spatial data
```

```
## Linking to GEOS 3.11.2, GDAL 3.8.2, PROJ 9.3.1; sf_use_s2() is FALSE
```

```
library(dplyr)  # data wrangling and handling
```

```
##
## 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
```

```
library(ggplot2)  # data visualization
```

## Import & Clean Shapefile (.shp) Data

**NOTE:** *For demonstration purposes, we will use four of the same studies used to build the base meta-analysis model to show how the model can be updated. Users should use new study data collected for this purpose and do not need to include these four studies, as they are already implemented in the package.*

The `cleaningWrangling()` function automates the process of cleaning and combining data. The output is a list that includes the combined response variables and study names (`state`), the corresponding table of predictor variables (`predictors`), and vectors of all numeric variables (`numerical_vars`), all categorical variables (`categorical_vars`), and all binary variables (`binary_vars`). The arguments for `wranglingCleaning()` are `data` and `response`, which should be formatted as follows:

- `data` should be a named list of shape (shp) files, 1 file per study
- `response` is a vector of shoreline suitability response variable names, 1 per study

```
## Import shape file data
# Choctawatchee Bay data (transformed 0.001deg.shp from WGS 84 to 6346 17N )
choc <- st_read("../Data/choctawatchee_bay/choctawatchee_bay_lssm_POINTS_0.001deg.shp")
```

```
## Reading layer 'choctawatchee_bay_lssm_POINTS_0.001deg' from data source
##   'C:\Users\erika\OneDrive - University of Central Florida\R\SCRG-TEST\Data\choctawatchee_bay\choctau
##   using driver 'ESRI Shapefile'
## Simple feature collection with 8639 features and 37 fields
## Geometry type: POINT
## Dimension:     XY
## Bounding box:  xmin: -86.63938 ymin: 30.36957 xmax: -86.09774 ymax: 30.52112
## Geodetic CRS:  WGS 84
```

```
choc <- choc[1:300,]

# Pensacola Bay data (transformed 0.001deg.shp from WGS 84 to 6346 17N )
pens <- st_read("../Data/santa_rosa_bay/Santa_Rosa_Bay_Living_Shoreline_POINTS_0.001deg.shp")
```

```
## Reading layer 'Santa_Rosa_Bay_Living_Shoreline_POINTS_0.001deg' from data source 'C:\Users\erika\Onel
##   using driver 'ESRI Shapefile'
## Simple feature collection with 10692 features and 32 fields
## Geometry type: POINT
## Dimension:     XY
## Bounding box:  xmin: -87.42733 ymin: 30.30139 xmax: -86.76889 ymax: 30.66474
## Geodetic CRS:  WGS 84
```

```r
pens <- pens[1:300,]

# Tampa Bay data (transformed 0.001deg.shp from WGS 84 to 6346 17N )
tampa <- st_read("../Data/tampa_bay/Tampa_Bay_Living_Shoreline_Suitability_Model_Results_POINTS_0.001deg
```

```
## Reading layer 'Tampa_Bay_Living_Shoreline_Suitability_Model_Results_POINTS_0.001deg' from data source
##   using driver 'ESRI Shapefile'
## Simple feature collection with 24623 features and 31 fields
## Geometry type: POINT
## Dimension:     XY
## Bounding box:  xmin: -82.85016 ymin: 27.44184 xmax: -82.29521 ymax: 28.03974
## Geodetic CRS:  WGS 84
```

```r
tampa <- tampa[1:300,]

# Indian River Lagoon data (transformed 0.001deg.shp from WGS 84 to 6346 17N )
IRL <- st_read("../Data/indian_river_lagoon/UCF_livingshorelinemodels_MosquitoNorthIRL_111m.shp")
```

```
## Reading layer 'UCF_livingshorelinemodels_MosquitoNorthIRL_111m' from data source 'C:\Users\erika\One
##   using driver 'ESRI Shapefile'
## Simple feature collection with 3013 features and 54 fields
## Geometry type: POINT
## Dimension:     XY
## Bounding box:  xmin: -80.92103 ymin: 28.52703 xmax: -80.64031 ymax: 29.07355
## Geodetic CRS:  WGS 84
```

```r
IRL <- IRL[1:300,]

## Note that the response variable for IRL does not follow the same evaluation
## convention as the other LSSMs
# This response will need to be converted separately to levels 1-3:
IRL <- IRL %>%
  mutate(Priority = as.numeric(case_when(
    Priority %in% c(0, 1) ~ 1,
    Priority %in% c(2, 3) ~ 2,
    Priority %in% c(4, 5) ~ 3,
    TRUE ~ Priority  # keeps other values unchanged
  )))

# Rename "Erosion" variable in IRL to avoid duplicate variable names
colnames(IRL)[10] <- "Erosion_1"

# Combine data into a named list
data <- list(choc = choc, pens = pens, tampa = tampa, IRL = IRL)

# Create vector of response variables in the order of the studies
response <- c("SMMv5Class", "SMMv5Class", "BMPallSMM", "Priority")


# Clean data using wranglingCleaning()
data <- wranglingCleaning(data = data, response = response)
```

```r
# After wranglingCleaning.R
pred <- data$predictors  # store predictor data separately for easy access
state <- data$state  # store response data separately for easy access
# numerical_vars <- data$numerical_vars  # store variable names in separate vectors
categorical_vars <- data$categorical_vars
binary_vars <- data$binary_vars

# Additional manual spell check
for (var in c(categorical_vars, binary_vars)) {
  print(unique(pred[[var]]))  # check for additional inconsistencies in data entry
}
```

```
## [1] "0-5"  "5-30" NA
## [1] "Low"      "Moderate" "High"     NA
## [1] "Shallow" "Deep"    NA
## [1] "No"    "Roads" NA
## [1] NA                    "Permanent Structure" "No"
##  [1] "Forested"        "Bare"            "Residential"     "Paved"
##  [5] "Commercial"      "Scrub-shrub"     "Extensive marsh" "Marsh"
##  [9] "Agriculture"     "Military"        NA                "Grass"
## [1] "No"               "Riprap"           "Bulkhead"         "Marina"
## [5] "Debris"           "Marina <50 slips" "Wharf"            NA
## [1] NA   "No"
## [1] "No"            "Marsh present" "Marsh island"  NA
## [1] NA            "Tidal creek" "No"
## [1] NA    "Yes"
## [1] NA    "Yes"
## [1] "Undefended" "Defended"   NA
## [1] "Low"      "Moderate" "High"     NA
## [1] NA          "Revetment" "Bulkhead"  "Debris"
## [1] NA    "Yes"
## [1] NA                           "New Smyrna Beach"
## [3] "Edgewater"                  "Canaveral National Seashore"
## [1] NA   "TR" "TS"
## [1] NA      "SM"    "BH"    "NS"    "HS"    "BH,HS"
##  [1] NA                 "CON"              "O-Metal, CON"     "RB, CON, CQ"
##  [5] "CON, RB"          "CON, O-Metal"     "RB"               "CONCRETE"
##  [9] "RB, CON"          "RR, CON, CQ"      "RR"               "O"
## [13] "RR, CQ"           "CON, O-Metal, WO" "CON, O-Metal, RB" "WO, CON"
## [17] "RR, CQ, CON"      "RR, CON"          "CON, O-Stone"     "RB, O"
## [21] "O-Metal"          "CON, RB, O-Stone" "CQ, RR, CON"      "CQ, CON, RR"
## [25] "RB, CQ, CON"
## [1] NA       "N"      "M"      "V"      "P"      "C"      "R"      "R, N"
## [9] "O-Road"
## [1] NA  2  3  1
## [1] NA    "PUB" "PRI"
## [1] NA      "V"     "R"     "S, V"  "S"     "W"     "V, OY"
##  [1] NA             "S, V"         "W"            "V"            "R"
##  [6] "S"            "R, V"         "V, OY"        "V, Shell"     "V,"
## [11] "V, Shell, S"  "S, Shell"     "V, Shell, OY" "V, R"         "OY"
## [16] "R, S"         "S, R"         "S, OY"        "Shell, V"
##  [1] NA         "S"        "OY"       "W"        "S, OY"    "S, R"
##  [7] "W, S"     "R"        "R, S"     "V"        "S, Shell" "S, W"
```

```
## [13] "S, V"     "V, Shell" "V, R"     "W, R"     "OY, S"    "Shell"
##  [1] NA             "W"            "S"            "Shell, S"     "V"
##  [6] "MF"           "V, Shell, OY" "S, OY"       "S, R"         "OY"
## [11] "OY, S"
## [1]  NA   0  99 106 111 133 112 520 101
## [1]  NA   0 112 111 122 101 231
## [1]  NA   0 112 111 210
## [1] NA         "110, 111" "210"      "112, 111"
## [1] NA  0  2
## [1] "0" "1" NA
## [1] "0" "1" NA
## [1] "0" "1" NA
## [1] "Canal" NA      "0"
## [1] NA  "0" "1"
## [1] NA  "1" "0"
## [1] "0" "1" NA
## [1] "0" "1" NA
## [1] NA  1  0
## [1] NA  "0" "1"
## [1] NA  "0" "1"
## [1] NA  "0" "1"
## [1] NA  1  0
```

```r
# Fix unique cases of misspellings
# pred$Adj_H2[44059] <- "V"  # original spelling had an extra comma

# For binary variables not entered as Yes/No or Y/N or 1/0, change to 1/0
pred$canal <- gsub("Canal", "1", pred$canal)

# Update predictors in data object for next steps
data$predictors <- pred
```

The data is now clean and all studies have been merged into a single predictor dataset. We will now impute variables with missing values and standardize numerical variables.

## Impute Missing Data & Standardize

Datasets may have missing values for certain predictor variables. For numerical and binary variables, we can use imputation techniques to fill these missing values at the site-wide scale and at the state-wide scale. Available techniques include using the median of the existing values for each variable or using the mean. Alternatively, working with spatial data offers the opportunity to use Kriging, a geostatistical method for spatial interpolation, because variables may be spatially autocorrelated. Kriging is only available at the site-wide scale, as it relies on using spatial autocorrelation for interpolation. The option to impute, and which method to use for imputation, are up to the user. However, we recommend Kriging if possible to better inform the model.

Here, we have chosen to use auto-Kriging to estimate missing values for site-wide numeric predictors where possible, and the remaining numeric predictors will be imputed with the state-wide mean values. We will also standardize our data to be centered on 0 with a standard deviation of 1, so that values are comparable across sites and variables. The mean and standard deviation used for standardization for each variable is saved to ensure that the standardization process is consistent.

```r
# Impute and standardize the data
data <- standardize(data = data, site.method = "krige", state.method = "meanImpute")
```

```
## Converting categorical data to binary dummy variables...

## Site-wide Imputation...

## Kriging selected for site-wide imputation

## Sites 1/4

## Predictor 1/3

## Warning in automap::autoKrige(stats::as.formula(formula), num.var, new_data =
## sp::SpatialPointsDataFrame(sf::st_coordinates(sf::st_as_sf(newdat)), : Removed
## 0.5 duplicate observation(s) in input_data:

## Predictor 2/3

## Predictor 3/3

## Warning in automap::autoKrige(stats::as.formula(formula), num.var, new_data =
## sp::SpatialPointsDataFrame(sf::st_coordinates(sf::st_as_sf(newdat)), : Removed
## 31 duplicate observation(s) in input_data:

## Warning in sqrt(krige_result$var1.var): NaNs produced

## Sites 2/4

## Sites 3/4

## Predictor 1/4

## Predictor 2/4

## Predictor 3/4

## Predictor 4/4

## Sites 4/4

## Predictor 1/4

## Predictor 2/4

## Predictor 3/4

## Predictor 4/4
```

```
## Mean state-wide imputation selected

## Standardizing numeric predictors
```

If Kriging (recommended), this process can take time. We can save the output as an R file (.rda), so that it can be easily accessed for subsequent tasks, without having to rerun the cleaning and standardization process each time we return to the code. Note that when RDA files are saved, the original object name will also be saved, so importing a RDA file may overwrite any same-name objects in your environment. RDS (.rds) files, though generally used to save smaller objects, can be used to reassign the object to a new name upon importing. We suggest saving such files in a separate "data" folder.

```r
# Save processed data for faster loading
save(data, file = "data/data_subset_standardize_impute.rda")  # file path
# saveRDS(data, file = "data/data_subset_standardize_impute.rds")  # RDS example

# Load saved data
load("data/data_subset_standardize_impute.rda")  # imports the named "data" object
# newdata <- readRDS("data/data_subset_standardize_impute.rda")  # importing and renaming RDS
```

## Model Selection

Model selection is done by the `BUPD.R` and BUPD_nonparallel.R‘ scripts. These scripts are parallelized (for faster computation) and non-parallel, respectively. We recommend using the non-parallel script on non-Unix-based operating systems (i.e. Windows systems), as we do here. We use the ordinal probit regression for this data, to model our ordered, categorical response for shoreline suitability (categories 1, 2, 3, with 3 being the best and 1 being the worst) as a function of the other combined predictors from the LSSM studies. *Note that existing living shorelines may be categorized as unsuitable for further restoration.* AIC-based model selection is performed in a step-wise manner using the "build-up, pair-down" (BUPD) method, that evaluates the performance of models built by running through and adding predictors one at a time and then reversing the process by removing a predictor at a time, starting from the best model. The probit link function allows us to transform the response from integers of 1, 2, and 3, to a probability ranging 0-1.

Model selection via BUPD is done for each study separately, before combining the results into a meta-analysis model that will enable comparable predictions of living shoreline suitability along the entire Florida coastline. The process returns the final selected model (lowest AIC) for each site, as well as the model formula, odds ratios, and beta coefficient estimates, and the run time for the process.

```r
# Retrieve predictor data
pred <- data$predictors
pred <- pred %>%
  mutate(across(c(OBJECTID, ID, bmpCountv5, n, distance, X, Y), as.character))
## keep non predictor data as characters to avoid being selected as predictors

# List all predictors (column names of known predictors)
numeric_pred <- pred %>%
  select_if(is.numeric)
factor_pred <- pred %>%
  select_if(is.factor)
pred <- cbind(factor_pred, numeric_pred)
predictors <- colnames(pred)
```

```r
# Model selection for all 4 studies
results <- BUPD(data = data, predictors = predictors,
                parallel = FALSE)  # TRUE for faster run time if using Linux/Mac
```

```r
# Once again, we will save the results as this process can be long
save(results, file = "data/BUPD_results_subset.rda")  # file path

# Load saved data
load("data/BUPD_results_subset.rda")  # imports the named "results" object

str(results)  # inspect the results
```

```
## List of 4
##  $ choc :List of 5
##   ..$ final_model:List of 4
##   .. ..$ est    : Named num [1:7] 0.365 3.015 -0.366 4.173 -3.979 ...
##   .. .. ..- attr(*, "names")= chr [1:7] "(Intercept)" "selectThis" "roads_1" "PermStruc_2" ...
##   .. ..$ COV    : num [1:7, 1:7] 0.476 -0.004 -0.467 0.336 -0.495 ...
##   .. .. ..- attr(*, "dimnames")=List of 2
##   .. .. .. ..$ : chr [1:7] "(Intercept)" "selectThis" "roads_1" "PermStruc_2" ...
##   .. .. .. ..$ : chr [1:7] "(Intercept)" "selectThis" "roads_1" "PermStruc_2" ...
##   .. ..$ loglike: Named num -107
##   .. .. ..- attr(*, "names")= chr "1"
##   .. ..$ AIC    : Named num 228
##   .. .. ..- attr(*, "names")= chr "1"
##   ..$ final_form :Class 'formula'  language Response ~ selectThis + roads_1 + PermStruc_2 + rd_pstruc
##   .. .. ..- attr(*, ".Environment")=<environment: 0x0000028a26c5ef30>
##   ..$ odds_ratios: Named num [1:7] 1.4399 20.3933 0.6936 64.8896 0.0187 ...
##   .. ..- attr(*, "names")= chr [1:7] "(Intercept)" "selectThis" "roads_1" "PermStruc_2" ...
##   ..$ coefs      : num [1:7, 1:2] 0.365 3.015 -0.366 4.173 -3.979 ...
##   .. ..- attr(*, "dimnames")=List of 2
##   .. .. ..$ : chr [1:7] "(Intercept)" "selectThis" "roads_1" "PermStruc_2" ...
##   .. .. ..$ : chr [1:2] "Estimate" "Std. Error"
##   ..$ run_time   : chr "173.56 sec elapsed"
##  $ pens :List of 5
##   ..$ final_model:List of 4
##   .. ..$ est    : Named num [1:3] 0.174 1.569 -0.432
##   .. .. ..- attr(*, "names")= chr [1:3] "(Intercept)" "Beach" "SAV"
##   .. ..$ COV    : num [1:3, 1:3] 0.0091 -0.00739 -0.00788 -0.00739 0.02772 ...
##   .. .. ..- attr(*, "dimnames")=List of 2
##   .. .. .. ..$ : chr [1:3] "(Intercept)" "Beach" "SAV"
##   .. .. .. ..$ : chr [1:3] "(Intercept)" "Beach" "SAV"
##   .. ..$ loglike: Named num -231
##   .. .. ..- attr(*, "names")= chr "2"
##   .. ..$ AIC    : Named num 469
##   .. .. ..- attr(*, "names")= chr "2"
##   ..$ final_form :Class 'formula'  language Response ~ Beach + SAV
##   .. .. ..- attr(*, ".Environment")=<environment: 0x0000028a26c5ef30>
##   ..$ odds_ratios: Named num [1:3] 1.19 4.803 0.649
##   .. ..- attr(*, "names")= chr [1:3] "(Intercept)" "Beach" "SAV"
##   ..$ coefs      : num [1:3, 1:2] 0.1742 1.5692 -0.4321 0.0954 0.1665 ...
##   .. ..- attr(*, "dimnames")=List of 2
##   .. .. ..$ : chr [1:3] "(Intercept)" "Beach" "SAV"
```

```
##   .. .. ..$ : chr [1:2] "Estimate" "Std. Error"
##   ..$ run_time   : chr "25.97 sec elapsed"
##  $ tampa:List of 5
##   ..$ final_model:List of 4
##   .. ..$ est    : Named num [1:2] 0.026 0.311
##   .. .. ..- attr(*, "names")= chr [1:2] "(Intercept)" "bathymetry_2"
##   .. ..$ COV    : num [1:2, 1:2] 0.00795 -0.00795 -0.00795 0.01537
##   .. .. ..- attr(*, "dimnames")=List of 2
##   .. .. .. ..$ : chr [1:2] "(Intercept)" "bathymetry_2"
##   .. .. .. ..$ : chr [1:2] "(Intercept)" "bathymetry_2"
##   .. ..$ loglike: Named num -322
##   .. .. ..- attr(*, "names")= chr "3"
##   .. ..$ AIC    : Named num 648
##   .. .. ..- attr(*, "names")= chr "3"
##   ..$ final_form :Class 'formula'  language Response ~ bathymetry_2
##   .. .. ..- attr(*, ".Environment")=<environment: 0x0000028a26c5ef30>
##   ..$ odds_ratios: Named num [1:2] 1.03 1.36
##   .. ..- attr(*, "names")= chr [1:2] "(Intercept)" "bathymetry_2"
##   ..$ coefs      : num [1:2, 1:2] 0.026 0.3107 0.0891 0.124
##   .. ..- attr(*, "dimnames")=List of 2
##   .. .. ..$ : chr [1:2] "(Intercept)" "bathymetry_2"
##   .. .. ..$ : chr [1:2] "Estimate" "Std. Error"
##   ..$ run_time   : chr "11.34 sec elapsed"
##  $ IRL  :List of 5
##   ..$ final_model:List of 4
##   .. ..$ est    : Named num [1:9] 5.837 -13.629 7.642 0.257 -15.907 ...
##   .. .. ..- attr(*, "names")= chr [1:9] "(Intercept)" "Edge_Type_31" "WTLD_VEG_3_2" "Slope_4" ...
##   .. ..$ COV    : num [1:9, 1:9] 1.47e+04 -1.56e+04 8.86e+02 3.17e-02 -1.56e+04 ...
##   .. .. ..- attr(*, "dimnames")=List of 2
##   .. .. .. ..$ : chr [1:9] "(Intercept)" "Edge_Type_31" "WTLD_VEG_3_2" "Slope_4" ...
##   .. .. .. ..$ : chr [1:9] "(Intercept)" "Edge_Type_31" "WTLD_VEG_3_2" "Slope_4" ...
##   .. ..$ loglike: Named num -35.1
##   .. .. ..- attr(*, "names")= chr "1"
##   .. ..$ AIC    : Named num 88.1
##   .. .. ..- attr(*, "names")= chr "1"
##   ..$ final_form :Class 'formula'  language Response ~ Edge_Type_3 + WTLD_VEG_3_2 + Slope_4 + Edge_Ty
##   .. .. ..- attr(*, ".Environment")=<environment: 0x0000028a26c5ef30>
##   ..$ odds_ratios: Named num [1:9] 3.43e+02 1.21e-06 2.08e+03 1.29 1.24e-07 ...
##   .. ..- attr(*, "names")= chr [1:9] "(Intercept)" "Edge_Type_31" "WTLD_VEG_3_2" "Slope_4" ...
##   ..$ coefs      : num [1:9, 1:2] 5.837 -13.629 7.642 0.257 -15.907 ...
##   .. ..- attr(*, "dimnames")=List of 2
##   .. .. ..$ : chr [1:9] "(Intercept)" "Edge_Type_31" "WTLD_VEG_3_2" "Slope_4" ...
##   .. .. ..$ : chr [1:2] "Estimate" "Std. Error"
##   ..$ run_time   : chr "3454.5 sec elapsed"
```

# Living Shoreline Suitability Meta-Analysis Model

## Preparing for the Meta-Analysis Model

The `varCov()` function is used to organize and clean the model results for studies that are to be used in the meta-analysis model. This calls the `getBetas()` function that retrieves the beta coefficient estimates and their standard deviations, combining all the information across studies. `getBetas()` can also be used

separately for coefficient retrieval from the ordinal probit regression models, however, we recommend running only `varCov()` to streamline the process.

```
# Retrieve selected models and assign to list
mods <- list(results[[1]]$final_model,  # 1st study (Choctawatchee)
             results[[2]]$final_model,  # 2nd study (Pensacola)
             results[[3]]$final_model,  # 3rd study (Tampa)
             results[[4]]$final_model)  # 4th study (IRL)

# Retrieve betas and store in list
Betas <- list(results[[1]]$coefs,  # 1st study (Choctawatchee)
              results[[2]]$coefs,  # 2nd study (Pensacola)
              results[[3]]$coefs,  # 3rd study (Tampa)
              results[[4]]$coefs)  # 4th study (IRL)

# Retrieve betas and variance-covariance matrices for betas
VARCOV <- varCov(data = data,
                 predictors = predictors,
                 mods = mods,
                 Betas = Betas)
```

```
## Identified selected models from 4 studies.

## Identified beta coefficient estimates from 4 studies.
## Identified beta coefficient estimates from 4 studies.

## Warning in FUN(X[[i]], ...): NAs introduced by coercion

## Betas estimates combined. See objects ‘combined_betas‘ and ‘combined_betas_only‘.

## Warning in FUN(X[[i]], ...): NAs introduced by coercion

## Standard error estimates combined. See objects ‘combined_se‘ and ‘combined_se_only‘.
```

## Updating the meta-analysis model with local data

The meta-analytic regression model can be updated with new data from Living Shoreline suitability assessments at local sites. This would better inform the model of the relationships between different factors and restoration suitability, which would be particularly useful for comparing suitability between different sites and studies in a standardized fashion, or for obtaining a continuous measure of the probability of success at each site (as opposed to categorized recommendations).

```
# Build meta-analysis model
meta_analysis <- meta_regression(data = data,
                                 varCov = VARCOV)
```

```
## Warning: Ratio of largest to smallest sampling variance extremely large. May
## not be able to obtain stable results.
```

## Predicting from the meta-analysis model

### Predicting after updating the model

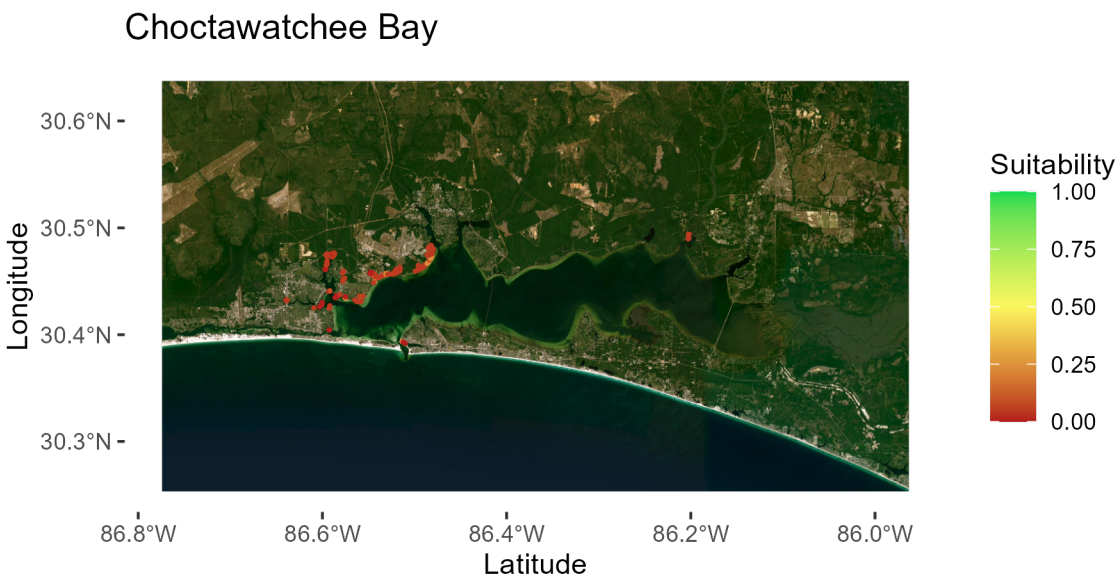
The `predict.meta()` function can be used to predict the suitability of a site for Living Shoreline restoration based on the model produced with `meta_regression()`. This requires the input of the meta-analysis model, updated with data from the sites for which suitability predictions will be made. Suitability for Living Shoreline restoration is represented as a probability for each location point, ranging 0-1 (0-100%). If no data is available, the `predict.rma()` function from the `metafor` package can help retrieve the state-wide average suitability from the model average (see "Predicting from the base state-wide model").

```
# Predict from updated model
predictions <- predict.meta_regression(data = data, meta_regression = meta_analysis)

# Add predictions to full predictor data set
pred$study <- data$state$study  # add studies back to predictors
pred$predictions <- predictions[,1]
```

Now, we can map these predictions to get a better picture of suitable locations. Remember the shapefile data we loaded at the beginning? The "geometry" column will come in handy for transforming our data into spatial features, so that we can plot our predictions on a map of the study site.

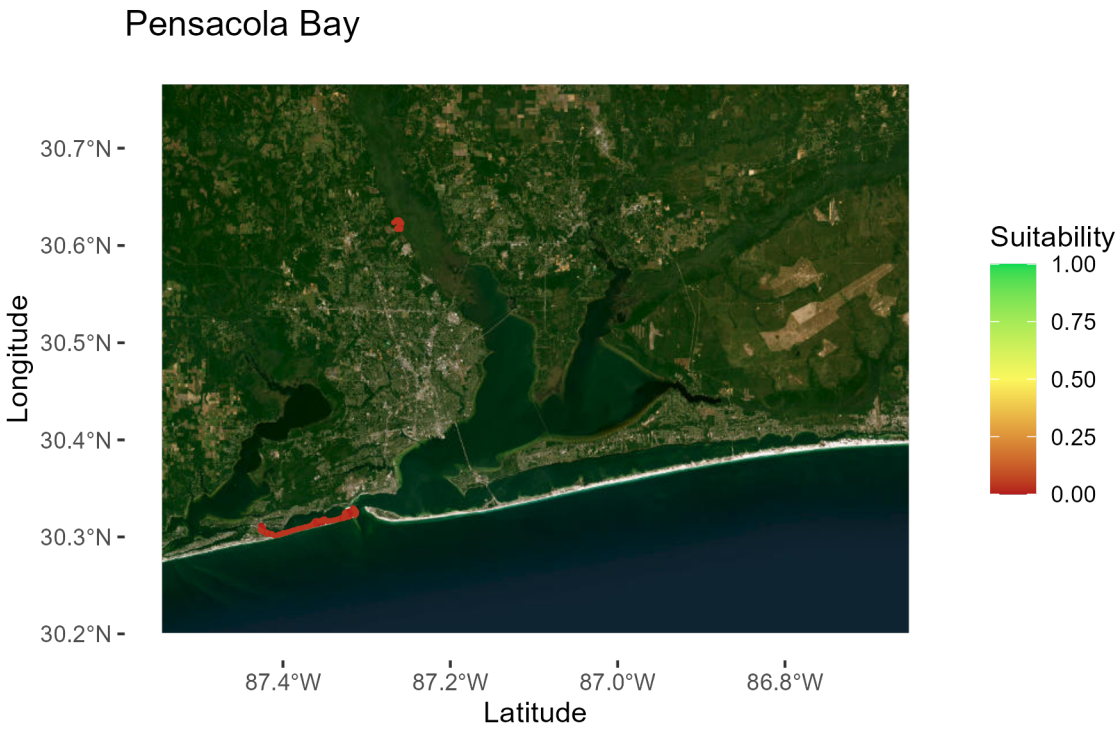
```
# Set the basemap to ESRI World Imagery
basemaps::set_defaults(map_service = "esri", map_type = "world_imagery")

## Choctawatchee Bay
# Subset full data set to Choctawatchee only
choc_pred <- pred[pred$study == "choc",]

# Obtain geometry from original shapefile
sf::st_geometry(choc_pred) <- choc$geometry  # this makes `choc_pred` a "sf" object

# Transform the coordinate reference system (CRS)
plot_choc <- sf::st_transform(choc_pred, crs = 3857)  # ensures proper projections
## The correct CRS is important for projecting properly onto the basemap

# Establish the bounding box for the map (corners of the plot)
choc_bb <- sf::st_bbox(c(xmin = -9644651-15000,
                         ymin = 3551143-15000,
                         xmax = -9584357+15000,
                         ymax = 3570713+15000),
                       crs = 3857)

# Extract the transformed coordinates
choc_coords <- as.data.frame(sf::st_coordinates(plot_choc))
plot_choc <- cbind(plot_choc, choc_coords)  # append to predictors

# Plot the predictions using `ggplot2`
choc_map <- basemaps::basemap_ggplot(ext = choc_bb) +  # extent = bounding box
  geom_sf(data = plot_choc, aes(x = X, y = Y, color = predictions), size = 0.25) +
  labs(x = "Latitude", y = "Longitude", title = "Choctawatchee Bay") +
  scale_color_gradient2(name = "Suitability", midpoint = 0.5, limits = c(0,1),
                        low = "#B31B1B", mid = "#FCF75E", high = "#0BDA51") +
  theme(plot.background = element_blank(),
```

```
        panel.background = element_blank(),
        panel.grid = element_blank(),
        legend.background = element_blank())
choc_map
```



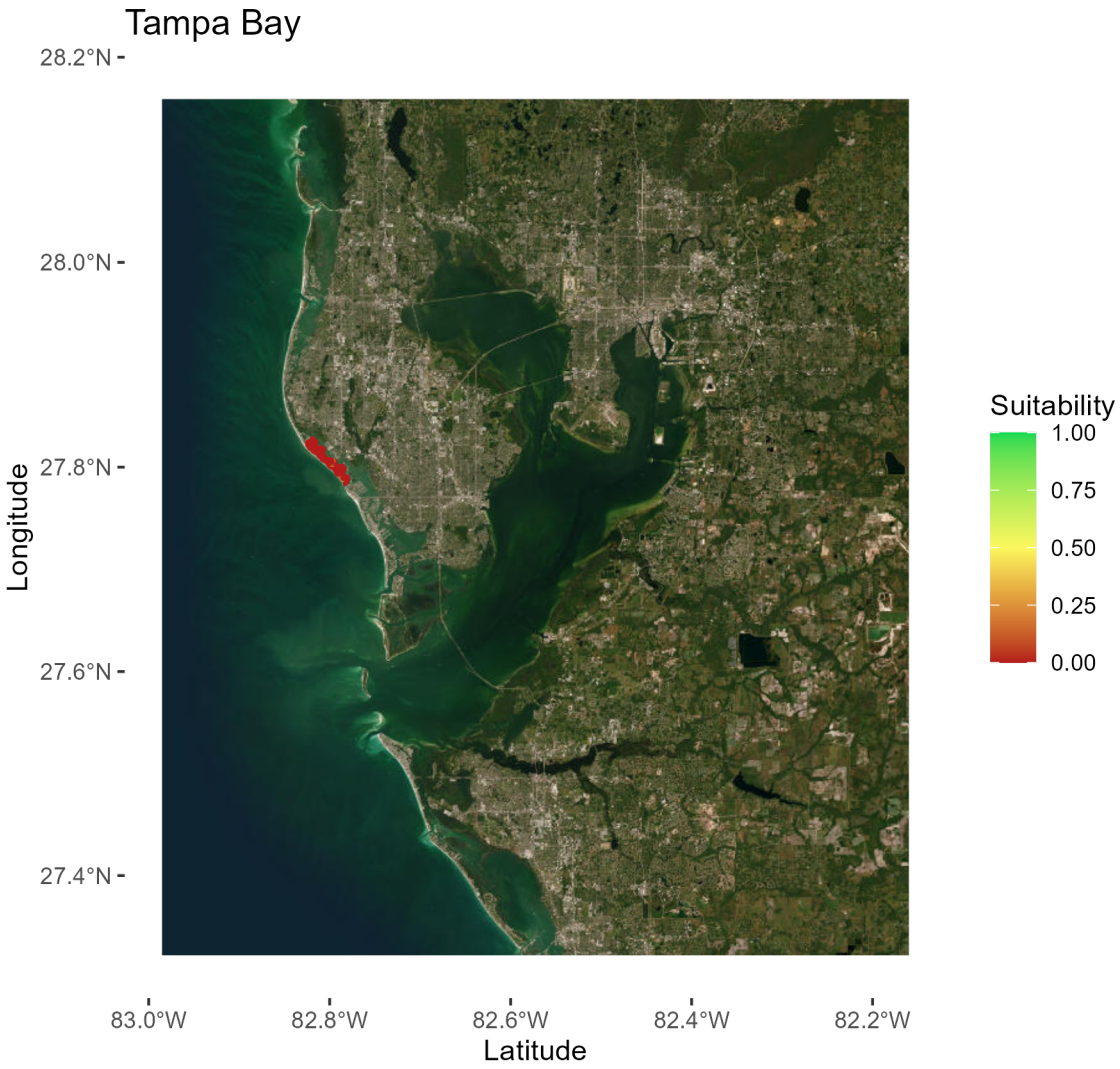

```
## Pensacola Bay
# Subset full data set to Pensacola only
pens_pred <- pred[pred$study == "pens",]

# Obtain geometry from original shapefile
sf::st_geometry(pens_pred) <- pens$geometry  # this makes `pens_pred` a "sf" object

# Transform the coordinate reference system (CRS)
plot_pens <- sf::st_transform(pens_pred, crs = 3857)  # ensures proper projections
## The correct CRS is important for projecting properly onto the basemap
```

```r
# Establish the bounding box for the map (corners of the plot)
pens_bb <- sf::st_bbox(c(xmin = -9732366-13000,
                         ymin = 3542349-13000,
                         xmax = -9659068+13000,
                         ymax = 3589286+13000),
                       crs = 3857)

# Extract the transformed coordinates
pens_coords <- as.data.frame(sf::st_coordinates(plot_pens))
plot_pens <- cbind(plot_pens, pens_coords)  # append to predictors

# Plot the predictions using `ggplot2`
pens_map <- basemaps::basemap_ggplot(ext = pens_bb) +  # extent = bounding box
  geom_sf(data = plot_pens, aes(x = X, y = Y, color = predictions), size = 0.25) +
  labs(x = "Latitude", y = "Longitude", title = "Pensacola Bay") +
  scale_color_gradient2(name = "Suitability", midpoint = 0.5, limits = c(0,1),
                        low = "#B31B1B", mid = "#FCF75E", high = "#0BDA51") +
  theme(plot.background = element_blank(),
        panel.background = element_blank(),
        panel.grid = element_blank(),
        legend.background = element_blank())
pens_map
```

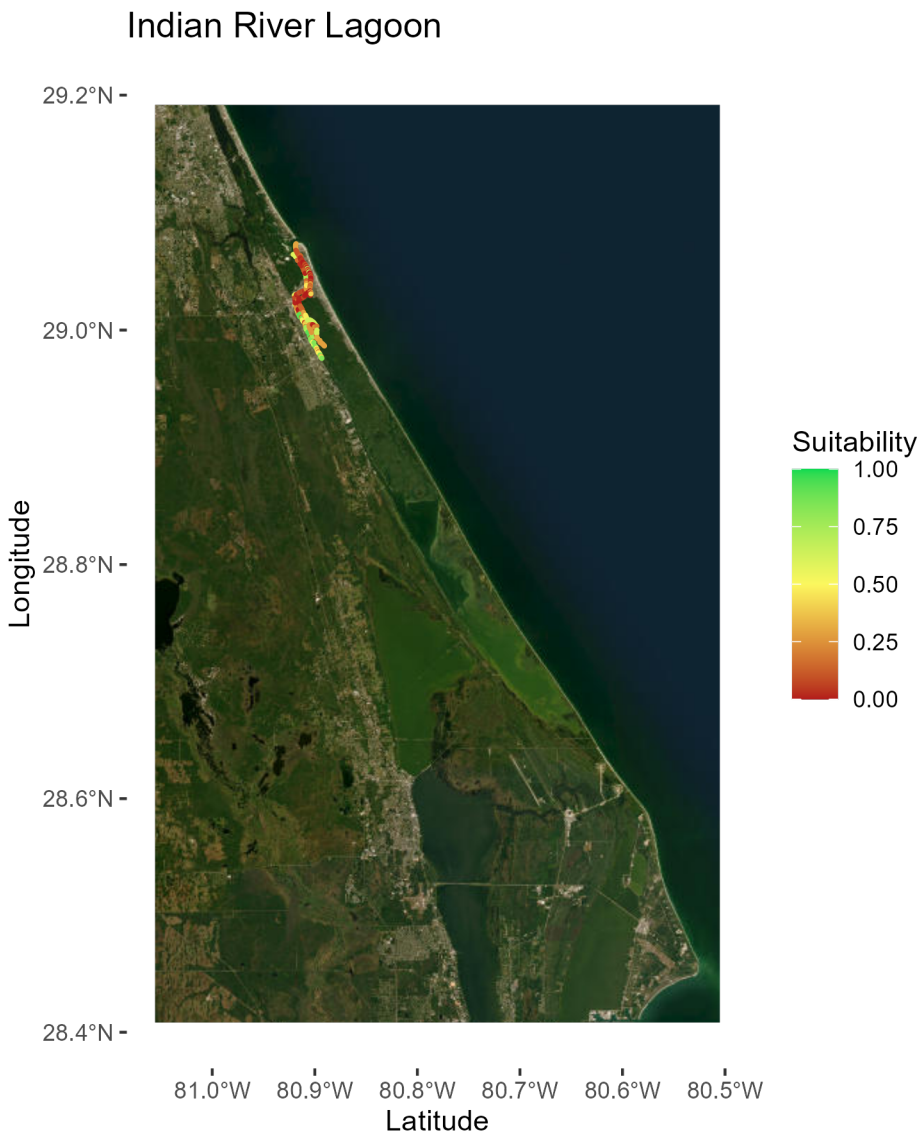



```
## Tampa Bay
# Subset full data set to Tampa only
tampa_pred <- pred[pred$study == "tampa",]

# Obtain geometry from original shapefile
sf::st_geometry(tampa_pred) <- tampa$geometry  # this makes `tampa_pred` a "sf" object

# Transform the coordinate reference system (CRS)
plot_tampa <- sf::st_transform(tampa_pred, crs = 3857)  # ensures proper projections
## The correct CRS is important for projecting properly onto the basemap

# Establish the bounding box for the map (corners of the plot)
tampa_bb <- sf::st_bbox(c(xmin = -9222837-15000,
                          ymin = 3178783-15000,
                          xmax = -9161061+15000,
                          ymax = 3253985+15000),
                        crs = 3857)
```

```r
# Extract the transformed coordinates
tampa_coords <- as.data.frame(sf::st_coordinates(plot_tampa))
plot_tampa <- cbind(plot_tampa, tampa_coords)  # append to predictors

# Plot the predictions using `ggplot2`
tampa_map <- basemaps::basemap_ggplot(ext = tampa_bb) +  # extent = bounding box
  geom_sf(data = plot_tampa, aes(x = X, y = Y, color = predictions), size = 0.25) +
  labs(x = "Latitude", y = "Longitude", title = "Tampa Bay") +
  scale_color_gradient2(name = "Suitability", midpoint = 0.5, limits = c(0,1),
                        low = "#B31B1B", mid = "#FCF75E", high = "#0BDA51") +
  theme(plot.background = element_blank(),
        panel.background = element_blank(),
        panel.grid = element_blank(),
        legend.background = element_blank())
tampa_map
```



```r
## Indian River Lagoon
# Subset full data set to IRL only
IRL_pred <- pred[pred$study == "IRL",]

# Obtain geometry from original shapefile
sf::st_geometry(IRL_pred) <- IRL$geometry  # this makes `IRL_pred` a "sf" object

# Transform the coordinate reference system (CRS)
plot_IRL <- sf::st_transform(IRL_pred, crs = 3857)  # ensures proper projections
## The correct CRS is important for projecting properly onto the basemap

# Establish the bounding box for the map (corners of the plot)
IRL_bb <- sf::st_bbox(c(xmin = -9008088-15000,
                        ymin = 3315585-15000,
                        xmax = -8976838+15000,
                        ymax = 3385010+15000),
                      crs = 3857)

# Extract the transformed coordinates
IRL_coords <- as.data.frame(sf::st_coordinates(plot_IRL))
plot_IRL <- cbind(plot_IRL, IRL_coords)  # append to predictors

# Plot the predictions using `ggplot2`
IRL_map <- basemaps::basemap_ggplot(ext = IRL_bb) +  # extent = bounding box
  geom_sf(data = plot_IRL, aes(x = X, y = Y, color = predictions), size = 0.25) +
  labs(x = "Latitude", y = "Longitude", title = "Indian River Lagoon") +
  scale_color_gradient2(name = "Suitability", midpoint = 0.5, limits = c(0,1),
                        low = "#B31B1B", mid = "#FCF75E", high = "#0BDA51") +
  theme(plot.background = element_blank(),
        panel.background = element_blank(),
        panel.grid = element_blank(),
        legend.background = element_blank())
IRL_map
```




**Predicting from the base state-wide model**

Currently, predictions at locations outside of the studies used to build the base meta-analysis model will return the model average as there is no information present for such sites. However, we intend to add state-wide geospatial data, such as bathymetry, to better inform the model at the state-wide scale. We will be updating the model, package, and predictions to reflect this.

```
# Use `predict.rma` from the `metafor` package
statewide_avg <- predict(meta_analysis)

# Back-transform to probabilities ranging 0-1
statewide_avg <- pnorm(statewide_avg$pred)
statewide_avg[1]
```

```
## [1] 0.8308891
```

```
## The statewide average of the base meta-analysis model is approx. 0.83,
## meaning that sites are on average 83% suitable for living shoreline restoration
```