GLUE-urbanQuant

Q2: Boosted Regression Trees

David Murray-Stoker

Contents

Load Data

```
## Load city information data
city.information.data <- read_csv(
  file = "data/full_city_information.csv",
  col_types = c("fffnnninii"),
  show_col_types = FALSE
)

## Load urbanization metric LMMs
# ISC
ISC.LMM <- read_rds(
  file = "data/analysis_data/metric_by_distance_LMMs/ISC_distance_sampling_design_LMM.rds"
)
# HII
HII.LMM <- read_rds(
  file = "data/analysis_data/metric_by_distance_LMMs/HII_distance_sampling_design_LMM.rds"
)
# Mean NDVI
mean.NDVI.LMM <- read_rds(
  file = "data/analysis_data/metric_by_distance_LMMs/mean_NDVI_distance_sampling_design_LMM.rds"
)
# Min MDVI
min.NDVI.LMM <- read_rds(
  file = "data/analysis_data/metric_by_distance_LMMs/min_NDVI_distance_sampling_design_LMM.rds"
)
# Max NDVI
max.NDVI.LMM <- read_rds(
  file = "data/analysis_data/metric_by_distance_LMMs/max_NDVI_distance_sampling_design_LMM.rds"
)
# Mean annual temperature
mean.annual.temperature.LMM <- read_rds(
  file = "data/analysis_data/metric_by_distance_LMMs/mean_annual_temperature_distance_sampling_design_L
)
# Temperature seasonality
temperature.seasonality.LMM <- read_rds(
  file = "data/analysis_data/metric_by_distance_LMMs/temperature_seasonality_distance_sampling_design_L
)
# Range annual temperature
range.annual.temperature.LMM <- read_rds(
  file = "data/analysis_data/metric_by_distance_LMMs/range_annual_temperature_distance_sampling_design_
)
# Annual precipitation
annual.precipitation.LMM <- read_rds(
  file = "data/analysis_data/metric_by_distance_LMMs/annual_precipitation_distance_sampling_design_LMM.
)
# Aridity index
aridity.index.LMM <- read_rds(
  file = "data/analysis_data/metric_by_distance_LMMs/aridity_index_distance_sampling_design_LMM.rds"
)
# GDP 2005
GDP.205.LMM <- read_rds(
  file = "data/analysis_data/metric_by_distance_LMMs/GDP_2005_distance_sampling_design_LMM.rds"
)
```

```r
# SSP1 2030
SSP1.2030.LMM <- read_rds(
  file = "data/analysis_data/metric_by_distance_LMMs/SSP1_2030_distance_sampling_design_LMM.rds"
)
# SSP1 2100
SSP1.2100.LMM <- read_rds(
  file = "data/analysis_data/metric_by_distance_LMMs/SSP1_2100_distance_sampling_design_LMM.rds"
)
# SSP2 2030
SSP2.2030.LMM <- read_rds(
  file = "data/analysis_data/metric_by_distance_LMMs/SSP2_2030_distance_sampling_design_LMM.rds"
)
# SSP2 2100
SSP2.2100.LMM <- read_rds(
  file = "data/analysis_data/metric_by_distance_LMMs/SSP2_2100_distance_sampling_design_LMM.rds"
)
# SSP5 2030
SSP5.2030.LMM <- read_rds(
  file = "data/analysis_data/metric_by_distance_LMMs/SSP5_2030_distance_sampling_design_LMM.rds"
)
# SSP5 2100
SSP5.2100.LMM <- read_rds(
  file = "data/analysis_data/metric_by_distance_LMMs/SSP5_2100_distance_sampling_design_LMM.rds"
)
```

Deviation Data

```r
#' Extract slopes from fitted linear mixed-effects models, and then calculates
#' the deviation (i.e., difference in slopes) between transect and point-sampling designs.

## Set the function
extract_deviation_data <- function(fitted_model, city_data) {
  ## Get predictions from the linear mixed-effects model (LMM)
  LMM.predictions <- ggpredict(
    model = fitted_model,
    terms = c("Standardized_Distance", "Sampling_Design", "City"),
    type = "random"
  ) %>%
    as_tibble() %>%
    rename(
      Standardized_Distance = x, Predicted = predicted,
      Sampling_Design = group, City = facet
    )

  ## Get the full slope along the gradient (i.e., not binned predictions)
  # Predicted value at distance = 0
  urban.predictions <- LMM.predictions %>%
    group_by(City, Sampling_Design) %>%
    filter(Standardized_Distance == 0) %>%
    select(Predicted) %>%
    rename(Urban_Prediction = Predicted) %>%
    ungroup()
  # Predicted value at distance = 1
  rural.predictions <- LMM.predictions %>%
    group_by(City, Sampling_Design) %>%
    filter(Standardized_Distance == 1) %>%
    select(Predicted) %>%
    rename(Rural_Prediction = Predicted) %>%
    ungroup()
  # Calculate the rate of change along the urbanization gradient (slope)
  predicted.changes <- urban.predictions %>%
    full_join(rural.predictions, by = c("City", "Sampling_Design")) %>%
    mutate(Predicted_Change = Urban_Prediction - Rural_Prediction) %>%
    pull(Predicted_Change)

  ## Combine the slope data with sample type and city for analysis
  raw.deviation.data <- tibble(
    Slope = predicted.changes,
    City = rep(levels(city_data$City), times = 4),
    Sampling_Design = rep(c("GLUE", "Random_Points", "Systematic_Points", "Random_Transect"), each = 13
  )

  ## Regroup sample type by "transects" and "points"
  regrouped.deviation.data <- transform(
    raw.deviation.data,
    Sampling_Design = if_else(raw.deviation.data$Sampling_Design == "GLUE" |
      raw.deviation.data$Sampling_Design == "Random_Transect",
    "Transects",
    "Points"
```

```r
    )
  )

  ## Calculate difference in slopes between "transects" and "points" for each city
  deviation.data <- regrouped.deviation.data %>%
    group_by(City, Sampling_Design) %>%
    summarise(Mean_Slope = mean(Slope), .groups = "keep") %>%
    ungroup() %>%
    pivot_wider(names_from = Sampling_Design, values_from = Mean_Slope) %>%
    rename(Points_Slope = Points, Transects_Slope = Transects) %>%
    mutate(Slope_Absolute_Difference = abs(Transects_Slope - Points_Slope)) %>%
    full_join(city_data, by = "City") %>%
    select(Continent, Country, City, Slope_Absolute_Difference, City_Area:Number_Nearby_Cities)

  return(deviation.data)
}
```

```r
## Set the model list
LMM.list <- list(
  ISC.LMM, HII.LMM, mean.NDVI.LMM, min.NDVI.LMM, max.NDVI.LMM,
  mean.annual.temperature.LMM, temperature.seasonality.LMM,
  range.annual.temperature.LMM, annual.precipitation.LMM, aridity.index.LMM,
  GDP.205.LMM, SSP1.2030.LMM, SSP1.2100.LMM,
  SSP2.2030.LMM, SSP2.2100.LMM, SSP5.2030.LMM, SSP5.2100.LMM
)

## Get a list of deviation data in separate dataframes
deviation.data <- map_dfr(
  LMM.list,
  extract_deviation_data,
  city_data = city.information.data
)

## Add predictor variables identifier to the data
deviation.data$Urbanization_Metric <- rep(
  c(
    "ISC", "HII", "Mean_NDVI", "Min_NDVI", "Max_NDVI",
    "Mean_Annual_Temperature", "Temperature_Seasonality",
    "Range_Annual_Temperature", "Annual_Precipitation", "Aridity_Index",
    "GDP_2005", "SSP_1_2030", "SSP_1_2100", "SSP_2_2030", "SSP_2_2100",
    "SSP_5_2030", "SSP_5_2100"
  ),
  each = 136
)

## Split deviation data into list by predictor variable
deviation.data.list <- deviation.data %>%
  group_split(Urbanization_Metric)
## Set names for each dataframe in the deviation data list (alphabetical)
names(deviation.data.list) <- c(
  "Annual_Precipitation", "Aridity_Index", "GDP_2005", "ISC", "HII",
  "Max_NDVI", "Mean_Annual_Temperature", "Mean_NDVI", "Min_NDVI",
  "Range_Annual_Temperature", "SSP_1_2030", "SSP_1_2100",
```

```
  "SSP_2_2030", "SSP_2_2100", "SSP_5_2030", "SSP_5_2100", "Temperature_Seasonality"
)
```

BRT Training

```r
# Determines optimal boosted regression trees parameters for each response variable.

## Start cluster
cluster <- makeCluster(n.cores)

## Run the analysis for the environmental metric
BRT.training.list <- parLapply(
  cluster,
  deviation.data.list,
  fun = function(df) {
      require(caret)
      require(gbm)
      require(tidyverse)

      ## Format the data for the boosted regression tree analysis
      BRT.data <- df %>%
        dplyr::select(Slope_Absolute_Difference, City_Area:Number_Nearby_Cities)

      ## BRT model training across a suite of model parameters
      BRT.training <- train(
        Slope_Absolute_Difference ~ City_Area + Human_Population_Size
          + Human_Population_Density + City_Age + Number_Nearby_Cities,
        data = df,
        method = "gbm",
        verbose = FALSE,
        tuneGrid = expand.grid(
          n.trees = c(
            1000, 2500, 5000, 10000, 15000, 20000
          ),
          interaction.depth = c(2:3),
          shrinkage = c(
            0.0001, 0.0005, 0.001
          ),
          n.minobsinnode = c(5, 10, 15, 20)
        )
      )

      ## Set dataframe for best tune of the BRT
      best.tune <- BRT.training$bestTune
    }
)

## Stop cluster
stopCluster(cluster)

## Rename each dataframe within the list
names(BRT.training.list) <- c(
  "Annual.Precipitation.Training", "Aridity.Index.Training", "GDP.2005.Training",
  "ISC.Training", "HII.Training", "Max.NDVI.Training", "Mean.Annual.Temperature.Training",
  "Mean.NDVI.Training", "Min.NDVI.Training", "Range.Annual.Temperature.Training",
  "SSP.1.2030.Training", "SSP.1.2100.Training", "SSP.2.2030.Training", "SSP.2.2100.Training",
  "SSP.5.2030.Training", "SSP.5.2100.Training", "Temperature.Seasonality.Training"
```

```r
)

## Export volume deviation BRT training results
list2env(BRT.training.list, envir = .GlobalEnv)

## Final dataframe with all BRT training data
BRT.training.output <- bind_rows(
  Annual.Precipitation.Training, Aridity.Index.Training, GDP.2005.Training,
  ISC.Training, HII.Training, Max.NDVI.Training, Mean.Annual.Temperature.Training,
  Mean.NDVI.Training, Min.NDVI.Training, Range.Annual.Temperature.Training,
  SSP.1.2030.Training, SSP.1.2100.Training, SSP.2.2030.Training, SSP.2.2100.Training,
  SSP.5.2030.Training, SSP.5.2100.Training, Temperature.Seasonality.Training
)
## Add a column for urbanization metric
BRT.training.output$Urbanization_Metric <- c(
  "Annual_Precipitation", "Aridity_Index", "GDP_2005", "ISC", "HII",
  "Max_NDVI", "Mean_Annual_Temperature", "Mean_NDVI", "Min_NDVI",
  "Range_Annual_Temperature", "SSP_1_2030", "SSP_1_2100",
  "SSP_2_2030", "SSP_2_2100", "SSP_5_2030", "SSP_5_2100", "Temperature_Seasonality"
)

## Export the BRT training output
write_rds(
  BRT.training.output,
  file = "data/analysis_data/deviation_BRTs/BRT_training_output.rds"
)
```

```
## <environment: R_GlobalEnv>
```

Predictors of Deviations

Set Functions

```r
#' Fits boosted regressions tree to identify the best predictors of deviations
#' between transect and point-sampling designs.

## Set the function
fit_deviation_BRT <- function(df, BRT_paramater_df) {
  ## Get the name of the urbanization metric
  metric.name <- df %>%
    pull(Urbanization_Metric) %>%
    unique()

  ## Format the data for the boosted regression tree analysis
  BRT.data <- df %>%
    select(Slope_Absolute_Difference, City_Area:Number_Nearby_Cities)

  ## BRT parameter data
  BRT.parameters <- BRT_paramater_df %>%
    filter(Urbanization_Metric == metric.name)

  ## Fit the boosted regression tree model
  deviation.BRT.model <- gbm(
    Slope_Absolute_Difference ~ City_Area + Human_Population_Size
      + Human_Population_Density + City_Age + Number_Nearby_Cities,
    distribution = "gaussian",
    data = BRT.data,
    n.trees = BRT.parameters$n.trees,
    interaction.depth = BRT.parameters$interaction.depth,
    n.minobsinnode = BRT.parameters$n.minobsinnode,
    shrinkage = BRT.parameters$shrinkage,
    bag.fraction = 0.5,
    cv.folds = 10,
    n.cores = 4
  )
}
```

```r
#' Extracts the relative influence values from a boosted regresion tree.

## Set the function
extract_deviation_BRT_results <- function(BRT_model) {
  # Get variables and relative influence from the BRT
  BRT.summary <- as_tibble(summary.gbm(BRT_model, plotit = FALSE)) %>%
    rename(
      "Predictor_Variable" = "var",
      "Relative_Influence" = "rel.inf"
    )

  return(BRT.summary)
}
```

Run the Boosted Regression Trees

```r
## Fit the BRTs
deviation.predictor.BRT.list <- map(
  deviation.data.list,
  fit_deviation_BRT,
  BRT_paramater_df = BRT.training.output
)

## Get the relative influence for each BRT
deviation.predictor.BRT.summaries <- map_dfr(
  deviation.predictor.BRT.list,
  extract_deviation_BRT_results
) %>%
  add_column(
    Urbanization_Metric = rep(names(deviation.data.list), each = 5)
  )
```

Export Data

```
## Deviation data
write_rds(
  deviation.data,
  file = "data/analysis_data/deviation_BRTs/deviation_data.rds"
)

## BRT model list
write_rds(
  deviation.predictor.BRT.list,
  file = "data/analysis_data/deviation_BRTs/deviation_predictor_BRTs.rds"
)

## BRT summaries
write_rds(
  deviation.predictor.BRT.summaries,
  file = "data/analysis_data/deviation_BRTs/deviation_predictor_relative_influence_summary.rds"
)
```

Workspace Information

Table 1: Packages required for data management and analyses.

Package	Loaded Version	Date
caret	6.0-94	2023-03-21
dplyr	1.1.4	2023-11-17
forcats	1.0.0	2023-01-29
gbm	2.1.9	2024-01-10
ggeffects	1.4.0	2024-02-05
ggplot2	3.4.4	2023-10-12
kableExtra	1.4.0	2024-01-24
knitr	1.45	2023-10-30
lattice	0.22-5	2023-10-24
lme4	1.1-35.1	2023-11-05
lmerTest	3.1-3	2020-10-23
lubridate	1.9.3	2023-09-27
Matrix	1.6-5	2024-01-11
purrr	1.0.2	2023-08-10
readr	2.1.5	2024-01-10
snow	0.4-4	2021-10-27
stringr	1.5.1	2023-11-14
tibble	3.2.1	2023-03-20
tidyr	1.3.1	2024-01-24
tidyverse	2.0.0	2023-02-22