tutorial_Deadtrees

AUTHOR
Teja Kattenborn (teja.kattenborn@geosense.uni-freiburg.de); Joachim Maack

PUBLISHED
January 14, 2025

Tutorial on dead trees in Germany

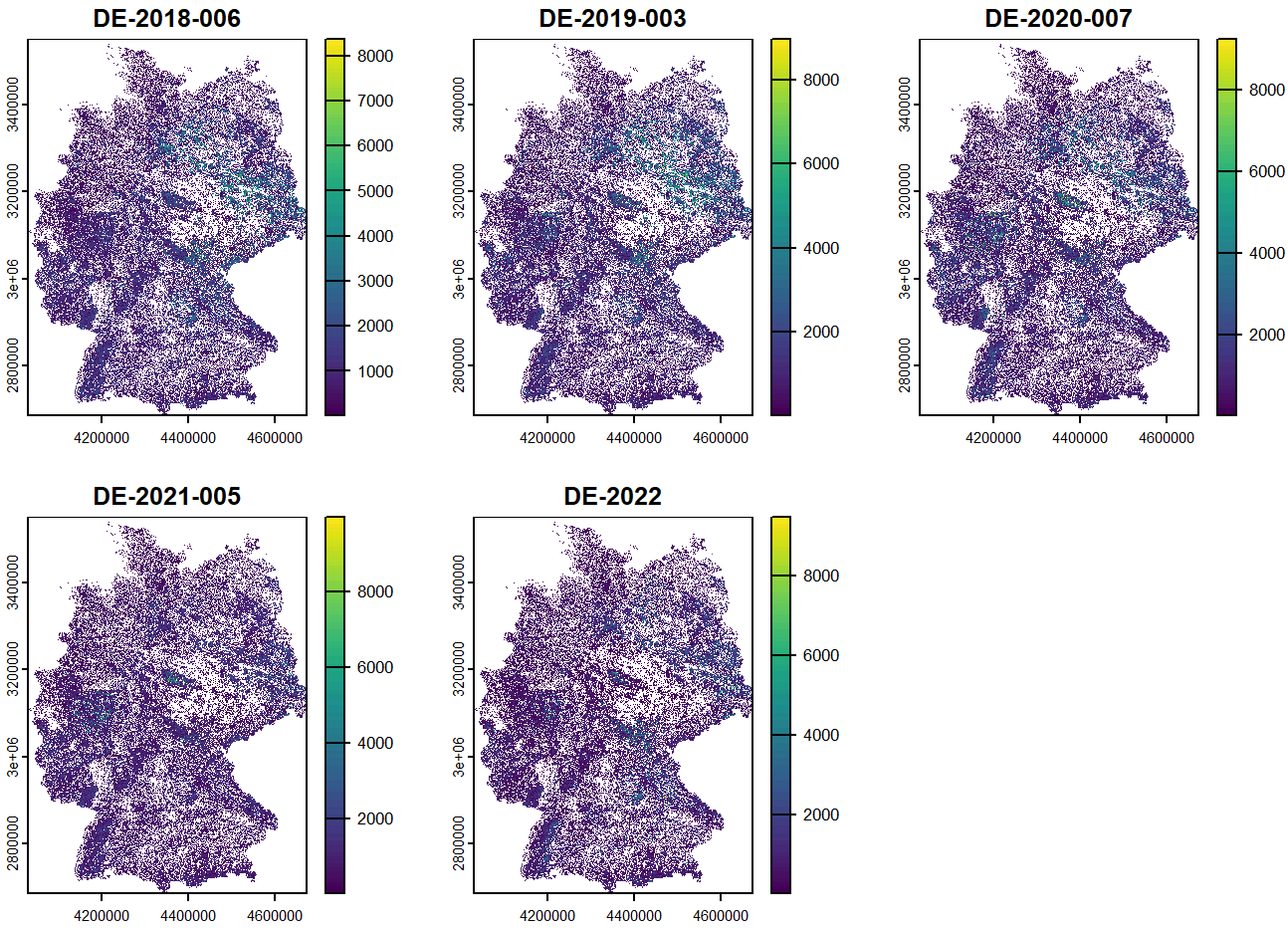
We will analyze the distribution of standing deadwood in Germany.

Standing dead trees between 2018 and 2022

We have prepared a raster stack of the dead trees product. Each band represents one year from 2018 to 2020. The resolution is down-sampled to 100 meters. The extent of the data set is Germany.

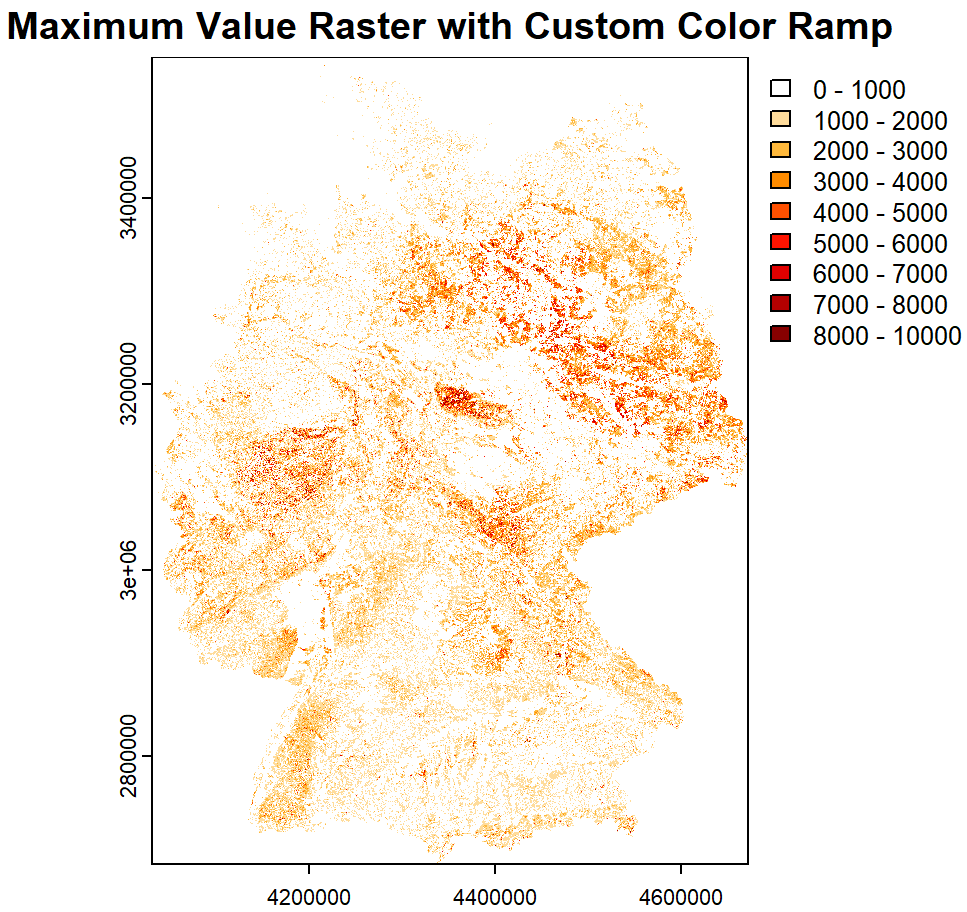
Set working directory

Loading the raster files and start exploring the data with some plotting



So where did the most trees die in this 5-year period? Lets find the hot-spots for tree mortality. Feel free to optimize the map, e.g. change the color ramp and breaks or make a whole new map.

```
# Calculate the maximum value for each pixel
max_raster = app(raster_stack, fun = max, na.rm = TRUE)


# Define a custom color ramp
color_ramp = colorRampPalette(c("white", "orange", "red", "darkred"))

# Create breaks to emphasize values above 5000
breaks = c(0, 1000, 2000, 3000, 4000, 5000, 6000, 7000, 8000, 10000)

plot(max_raster,
     col = color_ramp(length(breaks) - 1),
     breaks = breaks,
     main = "Maximum Value Raster with Custom Color Ramp")

# Customize the legend manually if needed
legend("topright",
       legend = c("Low", "5000", "High"),
       fill = color_ramp(3),
       title = "Elevation (m)")
```



The values are in % dead trees per pixel, code as integer values from 1-1000 for better data handling and faster calculations. However, for the human mind %-values are much easier to understand. So please make a plot with %-Values (1 = 0.01% and 10000 = 100%).

```
#nice plotting with %-values
```

Now we have and indication where most of the trees died.

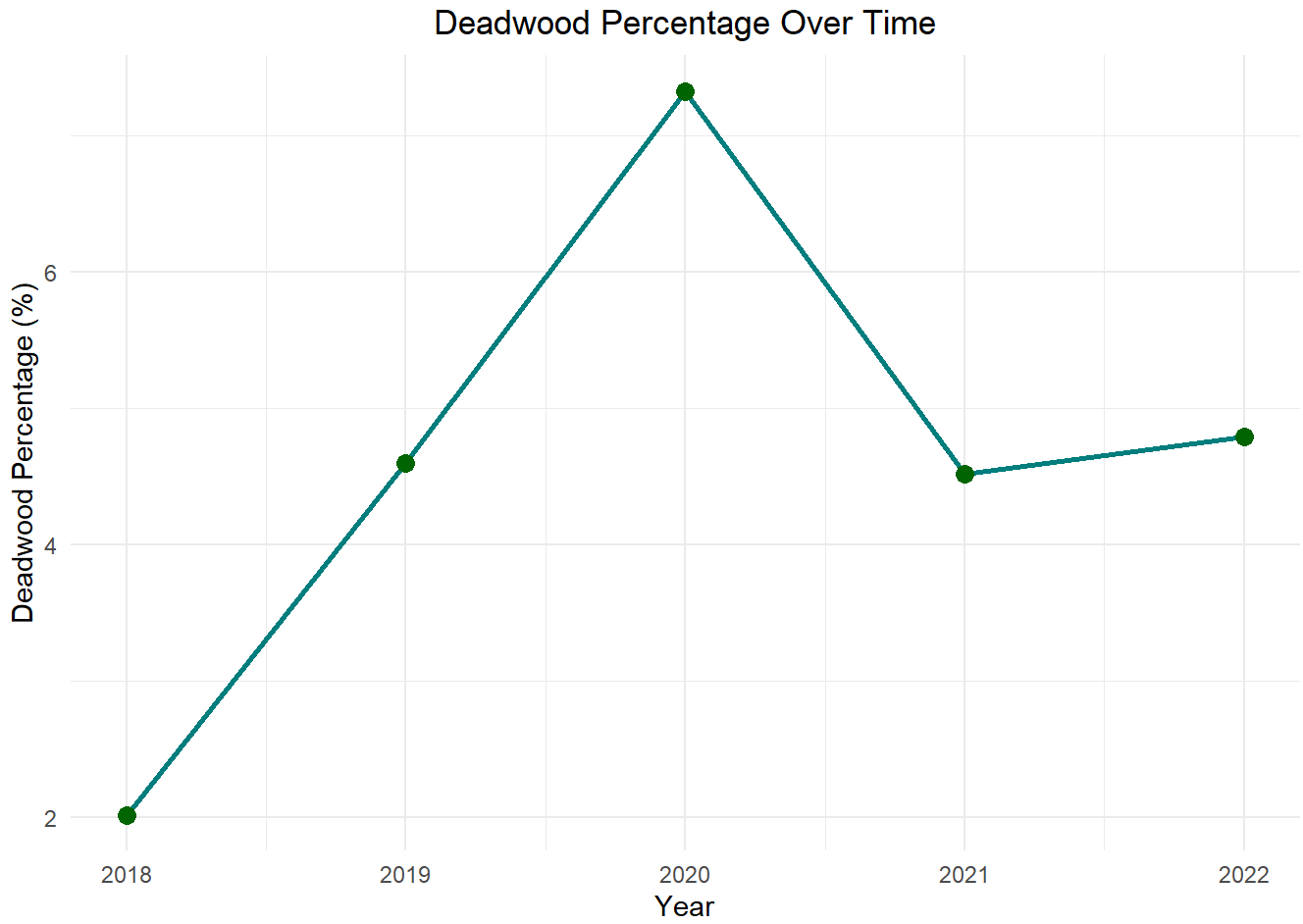
Mortality of tree species

Next we can analyze which tree species were most affected. To do so we need another map from the Thünen Institute, representing the most common tree species per pixel. We also down sampled to 100 meters. However, to process all trees species with varying thresholds at once would take too much time. So we are doing this for one species after another: Pick a species and run the calculations below with different thresholds.

	Year	TreeSpecies	AreaDeadTrees	AreaTreesSpecies	share
1	2018	8	58323	2892309	2.016486
2	2019	8	132858	2892309	4.593493
3	2020	8	211918	2892309	7.326949
4	2021	8	130644	2892309	4.516945
5	2022	8	138586	2892309	4.791535

Plotting deadwood percentage over time

```
library(ggplot2)

ggplot(results, aes(x = Year, y = share)) +
  geom_line(color = "#008080", size = 1) +    # Line style
  geom_point(color = "darkgreen", size = 3) +    # Points on the line
  theme_minimal() +
  labs(title = "Deadwood Percentage Over Time",
       x = "Year",
       y = "Deadwood Percentage (%)") +
  theme(plot.title = element_text(hjust = 0.5))
```



Connecting tree mortality with some climatic and topographic data

Lets check if we can model the relationship between environmental data and tree-mortality. We have selected environemntal data, consisting of

- SRTM (NASA space shuttle topography mission)
- Worldclim
- SoilGrids

What will we do?

1. Prepare the data for model training (merge data into a common raster stack).
2. Extract sample data from both the tree mortality map (response variable) and the climate data (predictors).
3. Train a Random Forest model and evaluate the importance of the predictors.

1. Prepare Your Data

```
library(terra)

# Maximum mortality map (response variable)
max_mortality_map = max_raster  # Maximum mortality map from the dead trees stack

# Specify the folder containing the raster files of the predictors
folder_preds = "Predictors"

# List all raster files of the predictors in the folder (e.g., .tif files)
raster_files_preds = list.files(folder_preds, pattern = "\\.tif$", full.names = TRUE)

# Load and stack the rasters
preds_raster_stack = rast(raster_files_preds)
```

```
# Check extent
ext(preds_raster_stack)
```

```
SpatExtent : 4031316.36304165, 4672516.36304165, 2684079.6079648, 3551279.6079648 (xmin, xmax, ymin, ymax)
```

```
ext(max_raster)
```

```
SpatExtent : 4031316.36304165, 4672516.36304165, 2684079.6079648, 3551279.6079648 (xmin, xmax, ymin, ymax)
```

```
# Check resolution
res(preds_raster_stack)
```

```
[1] 100 100
```

```
res(max_raster)
```

```
[1] 100 100
```

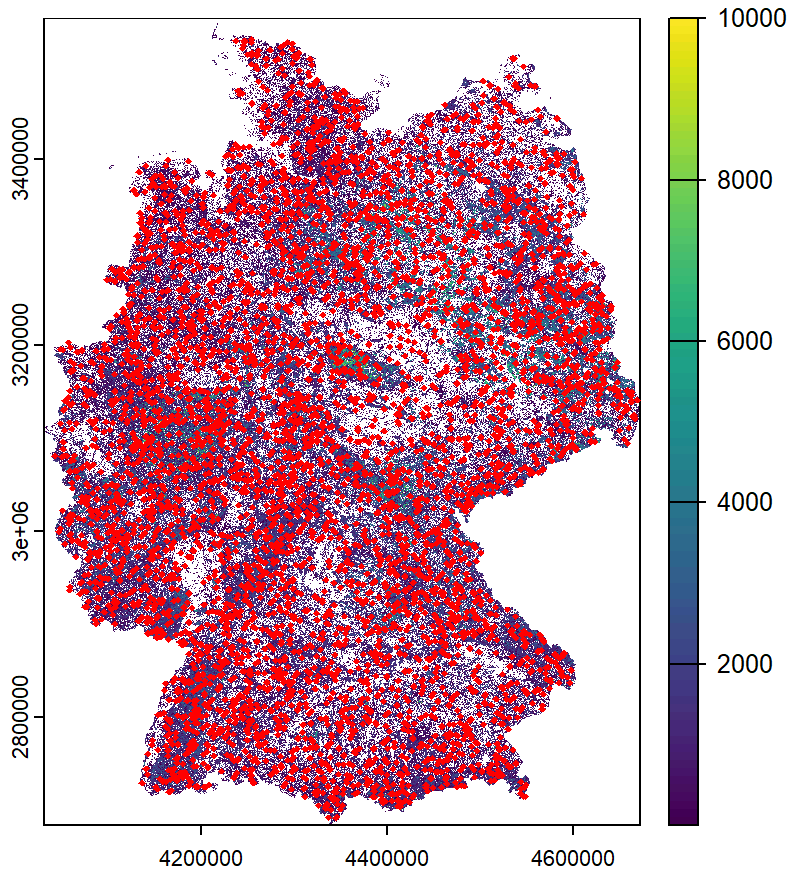
2. Sample predictor and response values

```
# combine predictors (environmental variables) and response (tree mortality)
stack_updated = c(preds_raster_stack, max_raster)

# Extract values for the response (tree mortality)
sampled_values = spatSample(stack_updated, size = 5000, method = "random", xy = TRUE, na.rm=TRUE)

# Check the sampled values
head(sampled_values)
```

```
        x       y wc2.1_30s_bio_1 wc2.1_30s_bio_12 clay_60-100cm_mean
1 4259966 2777330        6.902720         893.6746           309.8669
```

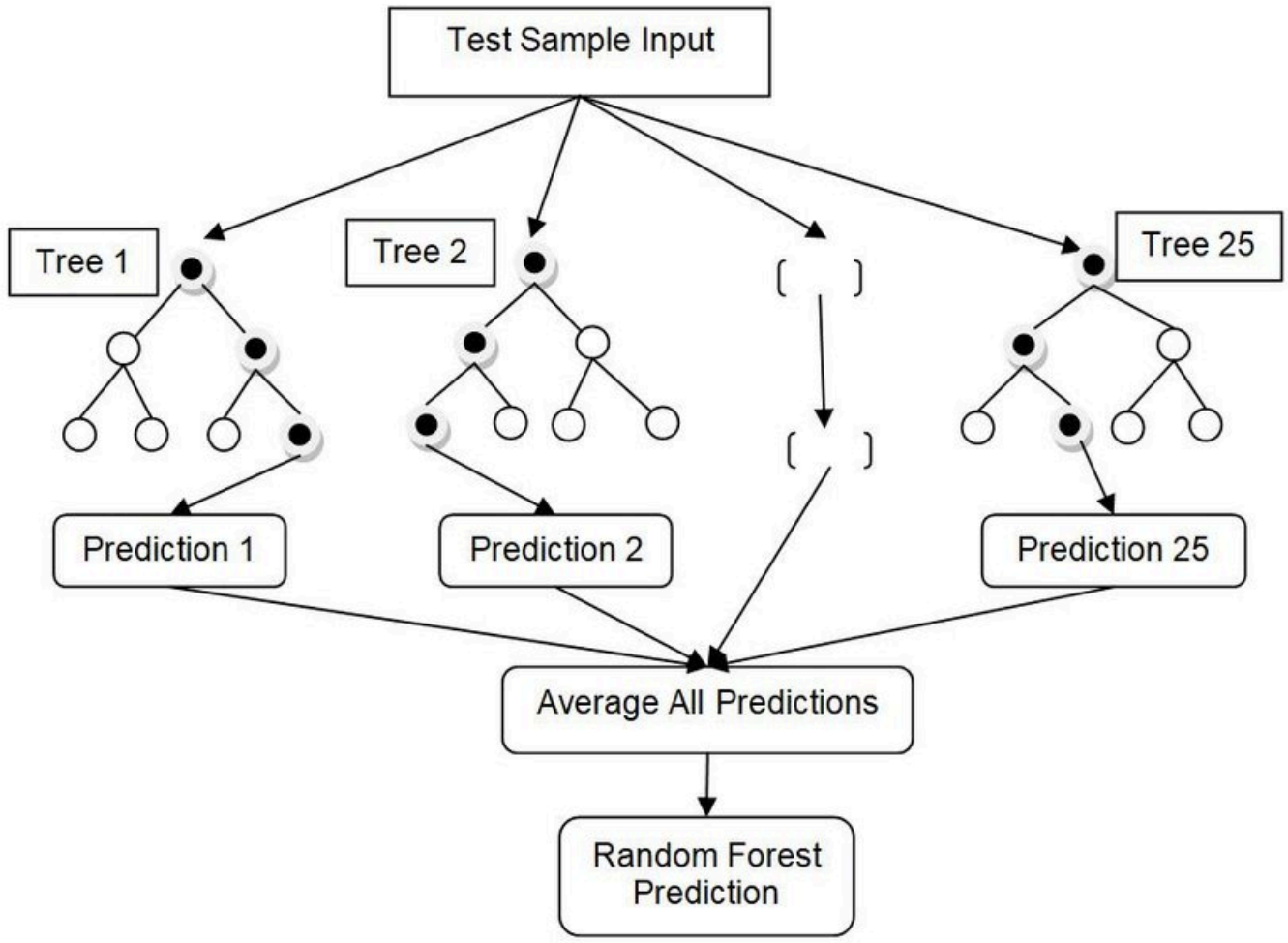
```
  2 4266866 3376930        8.838516         772.4887           130.1104
  3 4578566 3256430        8.916586         553.8244           117.7475
  4 4362166 3243830        8.993169         598.8713           267.9208
  5 4121566 3073330       10.013135         778.6621           255.9386
  6 4371666 2925830        8.697261         735.6548           267.6354
    wc2.1_30s_bio_15 wc2.1_30s_bio_17 wc2.1_30s_bio_7 sand_100-200cm_mean
1         24.18284         168.0940        26.18521            255.5174
2         16.98723         148.5362        23.35051            671.3736
3         23.14861         105.1399        25.93913            799.3964
4         16.78536         124.8285        24.00076            373.1443
5         14.36551         164.6992        24.46029            225.8372
6         15.55156         155.4508        27.16820            327.6772
      slope  max
1 2.6932673 1078
2 1.2944520 1703
3 0.9250370 2079
4 0.3976918  665
5 4.3718348 3361
6 2.5445435  260
```

```
plot(max_raster)
points(sampled_values$x, sampled_values$y, col = "red", pch = 18, cex = 0.5)
```



3. Train a Random Forest Model

What is a RandomForest? RandomForest is a machine learning model. Its underlying principle is simple; it builds on decision trees. Each branch of a tree is a decision based on a predictor (e.g. a temperature value from WorldClim or Slope from SRTM). After one branch follows another branch with a subsequent decision. The special thing about randomForest is that it creates hundreds of such trees, selects the best ones and averages there result. This "democratic" majority vote makes this method so robust and its simplicity makes it so efficient. This is why for years, randomForest is one of the most effective machine learning methods for tabular data.



Lets train a randomForest. We will train the model a few times and check if the results (explained variance, importance of the predictors) change. Why may this happen? What influence has the sample size? You could also check the correlation between the predictors and remove one if the it exceeds 0.7.

```
#library(randomForest)

# Define the response variable (tree mortality) and predictors (climate data)
response = sampled_values$max

# remove uncessesary variables (e.g. the x and y location)
head(sampled_values[, -c(1,2,11)])
```

```
  wc2.1_30s_bio_1 wc2.1_30s_bio_12 clay_60-100cm_mean wc2.1_30s_bio_15
1        6.902720         893.6746           309.8669         24.18284
2        8.838516         772.4887           130.1104         16.98723
3        8.916586         553.8244           117.7475         23.14861
4        8.993169         598.8713           267.9208         16.78536
5       10.013135         778.6621           255.9386         14.36551
6        8.697261         735.6548           267.6354         15.55156
  wc2.1_30s_bio_17 wc2.1_30s_bio_7 sand_100-200cm_mean     slope
1         168.0940        26.18521            255.5174 2.6932673
2         148.5362        23.35051            671.3736 1.2944520
3         105.1399        25.93913            799.3964 0.9250370
4         124.8285        24.00076            373.1443 0.3976918
5         164.6992        24.46029            225.8372 4.3718348
6         155.4508        27.16820            327.6772 2.5445435
```

```
predictors = sampled_values[, -c(1,2,11)]  # All columns except the first one (response variable)

# Train the Random Forest model
# What does the argument ntree? Read the help.
rf_model = randomForest(x = predictors, y = response, importance = TRUE, ntree = 500)

# Check the model results
print(rf_model)
```

```

Call:
 randomForest(x = predictors, y = response, ntree = 500, importance = TRUE)
               Type of random forest: regression
                     Number of trees: 500
No. of variables tried at each split: 2

          Mean of squared residuals: 1248864
                    % Var explained: 20.2
```

You could also check the correlation between the predictors and remove one if the it exceeds +-0.7.

Further details on the WorldClim data (bio_.*) is available here.

```
#library(ggcorrplot)

# Visualize the correlation matrix

head(sampled_values[, -c(1,2,11)])
```

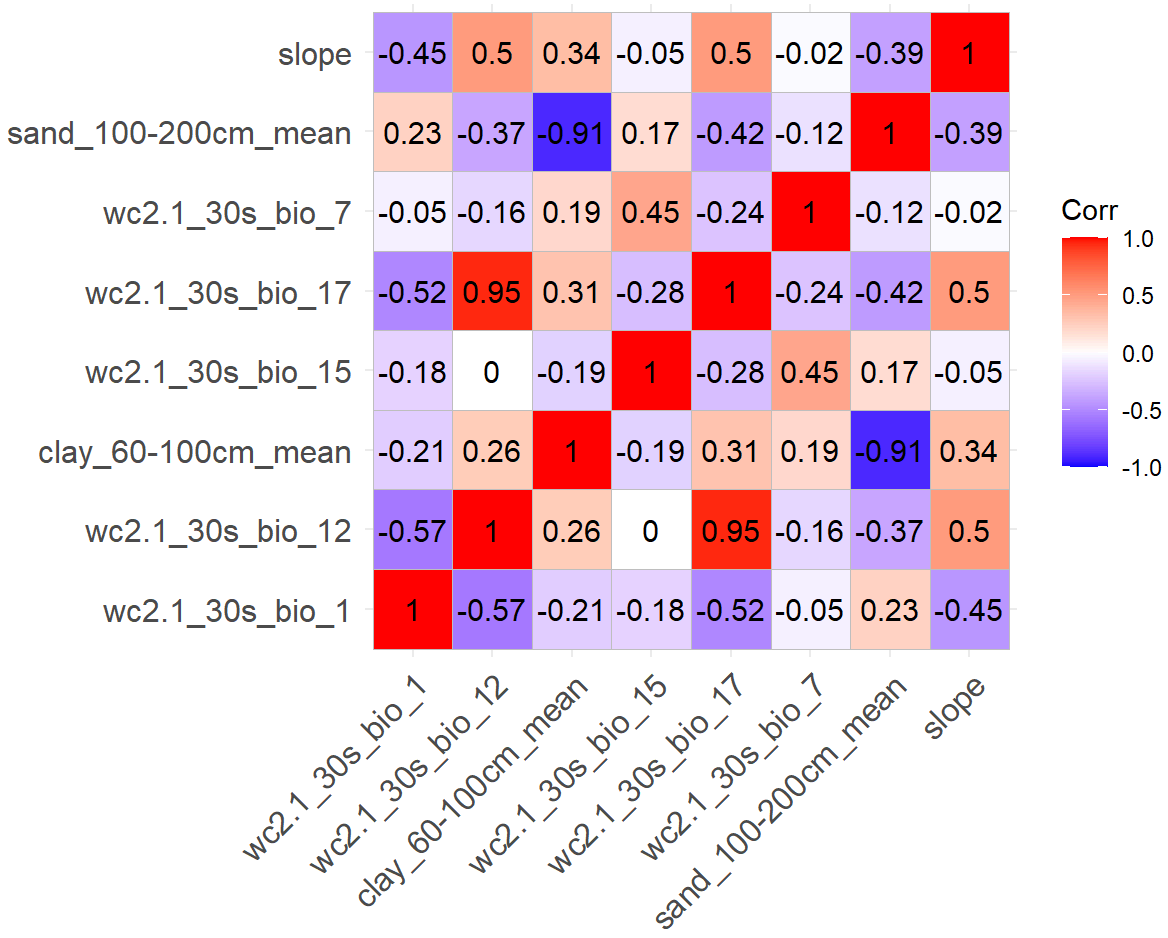
```
  wc2.1_30s_bio_1 wc2.1_30s_bio_12 clay_60-100cm_mean wc2.1_30s_bio_15
1        6.902720         893.6746           309.8669         24.18284
2        8.838516         772.4887           130.1104         16.98723
3        8.916586         553.8244           117.7475         23.14861
4        8.993169         598.8713           267.9208         16.78536
5       10.013135         778.6621           255.9386         14.36551
```

```
6        8.697261         735.6548           267.6354         15.55156
  wc2.1_30s_bio_17 wc2.1_30s_bio_7 sand_100-200cm_mean     slope
1         168.0940        26.18521            255.5174 2.6932673
2         148.5362        23.35051            671.3736 1.2944520
3         105.1399        25.93913            799.3964 0.9250370
4         124.8285        24.00076            373.1443 0.3976918
5         164.6992        24.46029            225.8372 4.3718348
6         155.4508        27.16820            327.6772 2.5445435
```

```
predictors = sampled_values[, -c(1,2,11)]  # All columns except the first one (response variable)

cor_matrix = cor(predictors, use = "complete.obs")
cor_matrix
```

```
                  wc2.1_30s_bio_1 wc2.1_30s_bio_12 clay_60-100cm_mean
wc2.1_30s_bio_1        1.00000000      -0.57496135         -0.2063083
wc2.1_30s_bio_12      -0.57496135       1.00000000          0.2554727
clay_60-100cm_mean    -0.20630827       0.25547266          1.0000000
wc2.1_30s_bio_15      -0.18203760      -0.00326302         -0.1858896
wc2.1_30s_bio_17      -0.51648072       0.95275164          0.3137254
wc2.1_30s_bio_7       -0.05037748      -0.16234697          0.1858783
sand_100-200cm_mean    0.23472866      -0.37093684         -0.9072550
slope                 -0.45107045       0.49619708          0.3421639
                  wc2.1_30s_bio_15 wc2.1_30s_bio_17 wc2.1_30s_bio_7
wc2.1_30s_bio_1         -0.18203760       -0.5164807     -0.05037748
wc2.1_30s_bio_12        -0.00326302        0.9527516     -0.16234697
clay_60-100cm_mean      -0.18588958        0.3137254      0.18587833
wc2.1_30s_bio_15         1.00000000       -0.2759053      0.45317330
wc2.1_30s_bio_17        -0.27590535        1.0000000     -0.23662734
wc2.1_30s_bio_7          0.45317330       -0.2366273      1.00000000
sand_100-200cm_mean      0.17048780       -0.4210966     -0.12275492
slope                   -0.04703542        0.5018069     -0.02084559
                  sand_100-200cm_mean       slope
wc2.1_30s_bio_1             0.2347287 -0.45107045
wc2.1_30s_bio_12           -0.3709368  0.49619708
clay_60-100cm_mean         -0.9072550  0.34216390
wc2.1_30s_bio_15            0.1704878 -0.04703542
wc2.1_30s_bio_17           -0.4210966  0.50180686
wc2.1_30s_bio_7            -0.1227549 -0.02084559
sand_100-200cm_mean         1.0000000 -0.39212127
slope                      -0.3921213  1.00000000
```

```
ggcorrplot(cor_matrix, lab = TRUE)
```



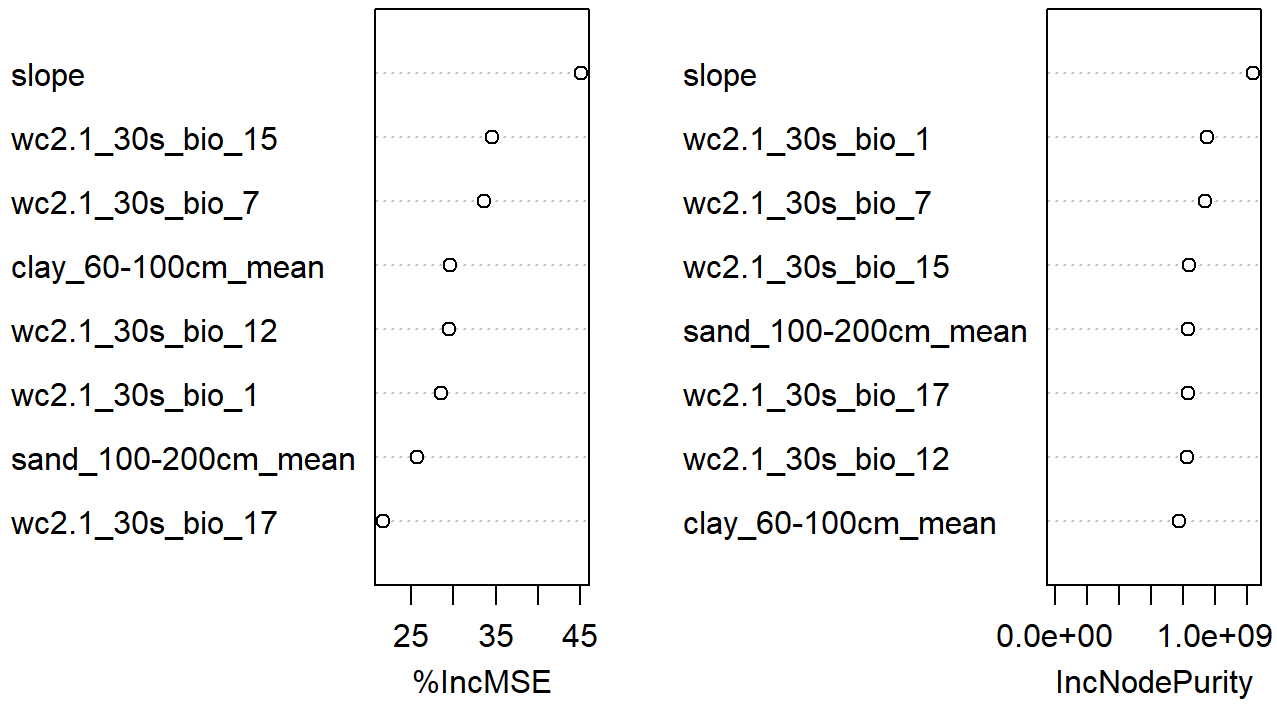
Permutation Test

In a Random Forest importance plot, %IncMSE (Percentage Increase in Mean Squared Error) is a measure of variable importance. It quantifies how much worse the model's predictions become (in terms of Mean Squared Error, MSE) when the values of a specific predictor variable are randomly permuted.

During the calculation of variable importance, the values of a predictor variable are permuted (shuffled randomly) across all observations, breaking the relationship between the predictor and the response.

IncNodePurity (Increase in Node Purity) is another measure of variable importance used in Random Forest models. It is based on the improvement in the model's ability to correctly classify or predict data (purity) that results from splitting a decision tree using a specific variable.

```
varImpPlot(rf_model , sort = TRUE  , main = "Predictor Importance" )
```



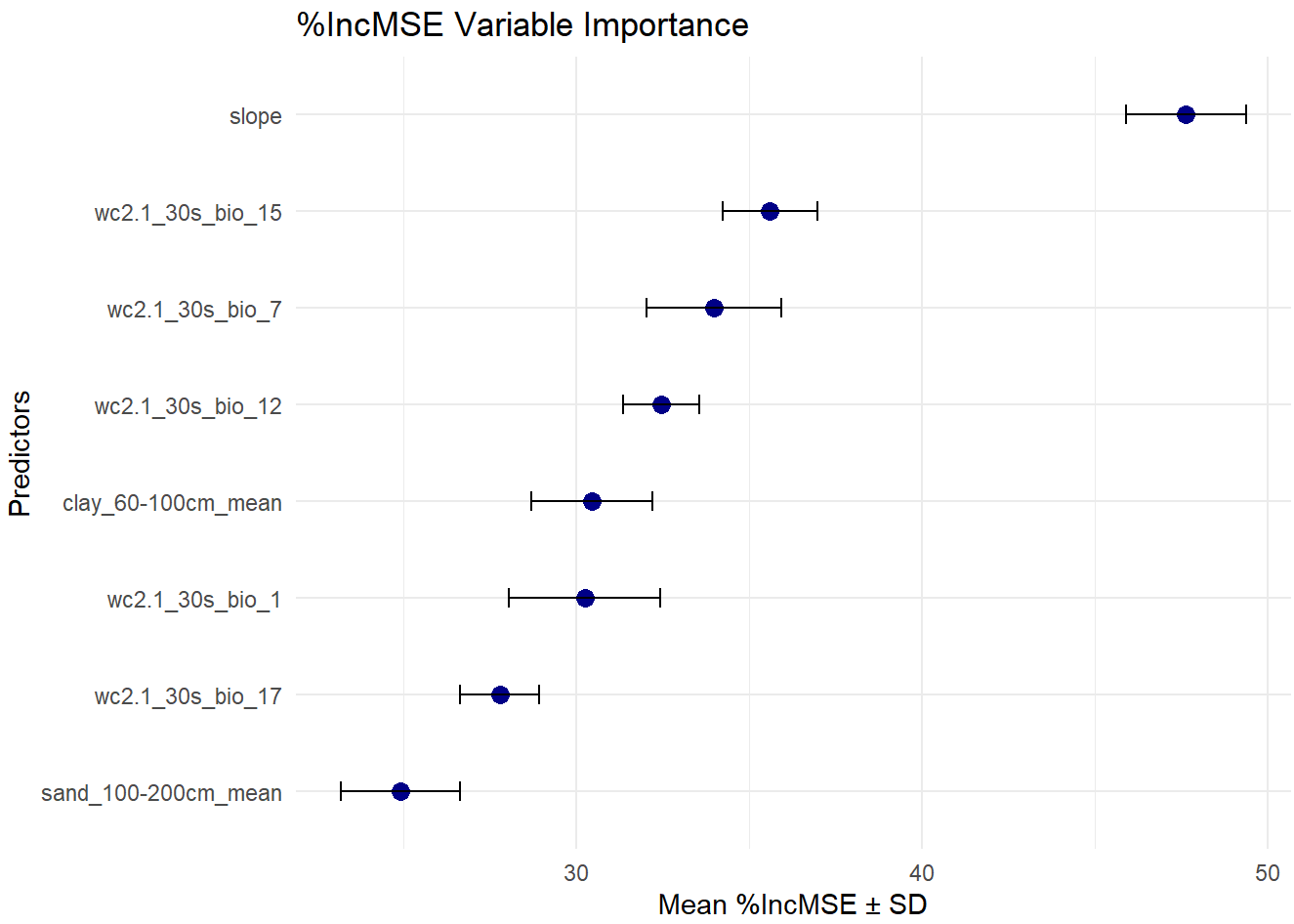
Permutation test advanced

The following procedure applies multiple permutation tests, to assess the stability of the predictor-response relationship. In other words: Are the results stable if we run them multiple times?
Warning: This code will need several minutes (~1 min * interation).

If the code needs too long on your computer, you can load reprocessed data below the loop-section.

```
# Parameters
n_iterations = 5 # Number of random samples
sample_size = 1000

# Store importance values
importance_results <- list()

# Loop to calculate variable importance
set.seed(42) # For reproducibility
for (i in 1:n_iterations) {
  # Create a random sample of the data
  sampled_data = spatSample(stack_updated, size = sample_size, method = "random", xy = TRUE, na.r

  response = sampled_values$max
```

```
  predictors = sampled_values[, -c(1,2,11)]  # All columns except the first one (response variable
  # Train Random Forest
  rf_model_it = randomForest(x = predictors, y = response, importance = TRUE)
  # Extract %IncMSE importance values
  importance_df = as.data.frame(importance(rf_model_it, type = 1)) # Type 1 for %IncMSE
  importance_df$Predictor = rownames(importance_df)
  importance_df$Iteration = i
  # Store the results
  importance_results[[i]] = importance_df
}
# Combine all importance values into a single data frame
all_importance = do.call(rbind, importance_results)
# Aggregate the importance values (mean and standard deviation)
importance_summary = all_importance %>%
  group_by(Predictor) %>%
  summarise(
    Mean_IncMSE = mean(`%IncMSE`),
    SD_IncMSE = sd(`%IncMSE`)
  )
# Load pre-processed data
#importance_summary = readRDS("results_importance.rds")
# Plot the results
ggplot(importance_summary, aes(x = reorder(Predictor, Mean_IncMSE), y = Mean_IncMSE)) +
  geom_point(size = 3, color = "darkblue") +
  geom_errorbar(aes(ymin = Mean_IncMSE - SD_IncMSE, ymax = Mean_IncMSE + SD_IncMSE), width = 0.2)
  coord_flip() +
  labs(
    title = "%IncMSE Variable Importance",
    x = "Predictors",
    y = "Mean %IncMSE ± SD"
  ) +
  theme_minimal()
```



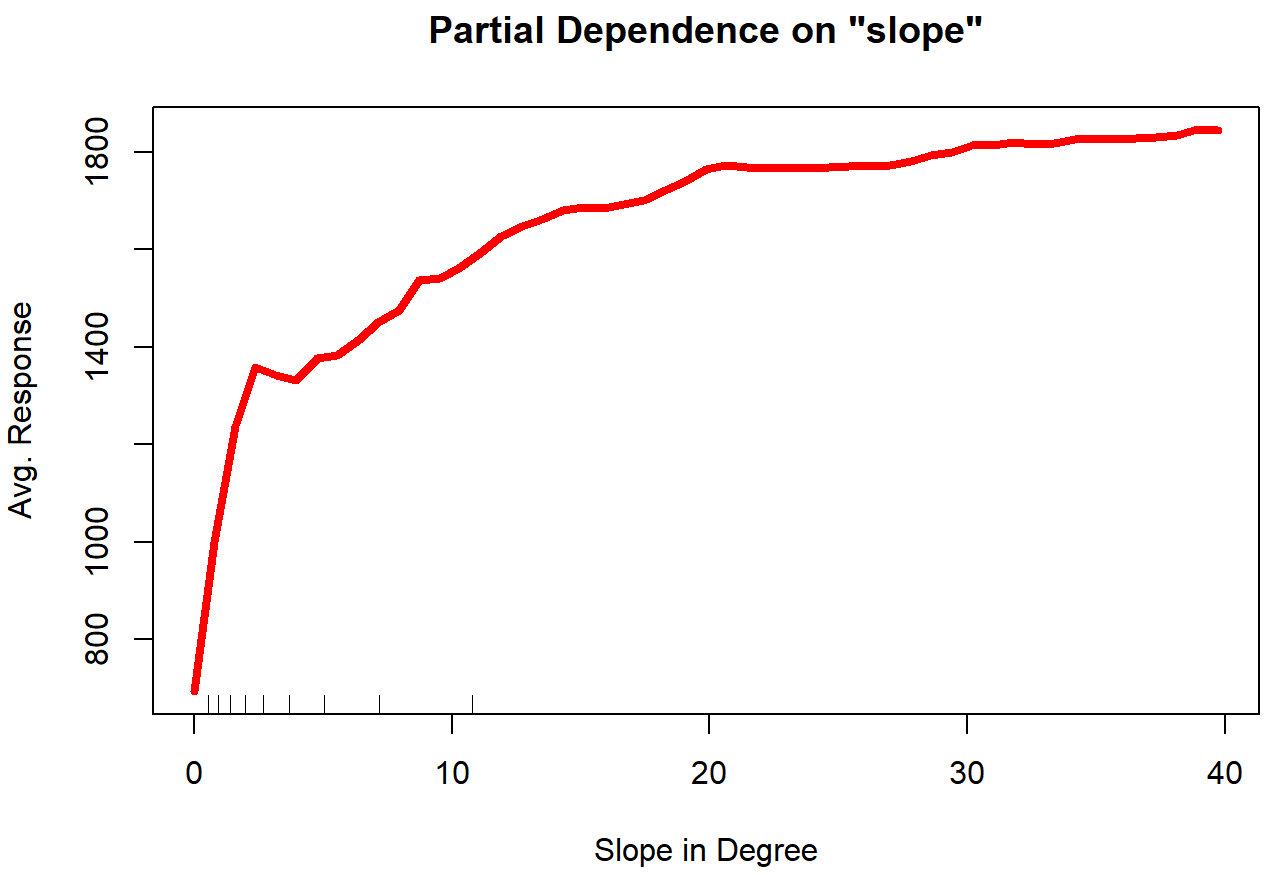
Evaluate the dependencies of individual predictors to response

Next, we can calculate the effect of each predictor on the mortality. In easy words, the effect of a single variable is calculated based on its effect on the prediction.

Slope

The response is given in the original values (1-10000). Divide by 100 for %-values

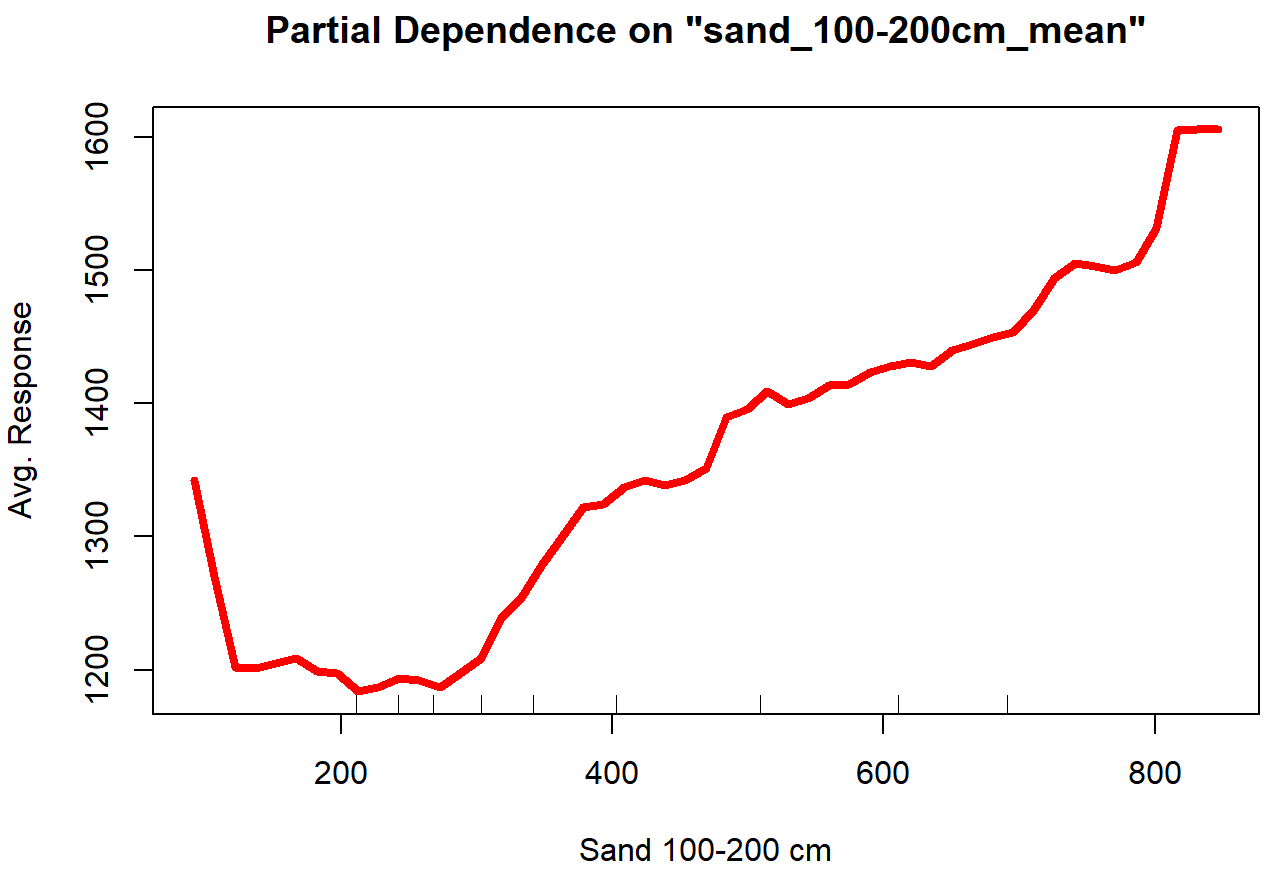
```
# Evaluate the predictors effects
partialPlot(rf_model, predictors, "slope", "1",xlab="Slope in Degree", ylab="Avg. Response",lwd=4
```



Soil: Sand 100-200 cm

The response is given in the original values (1-10000). Divide by 100 for %-values

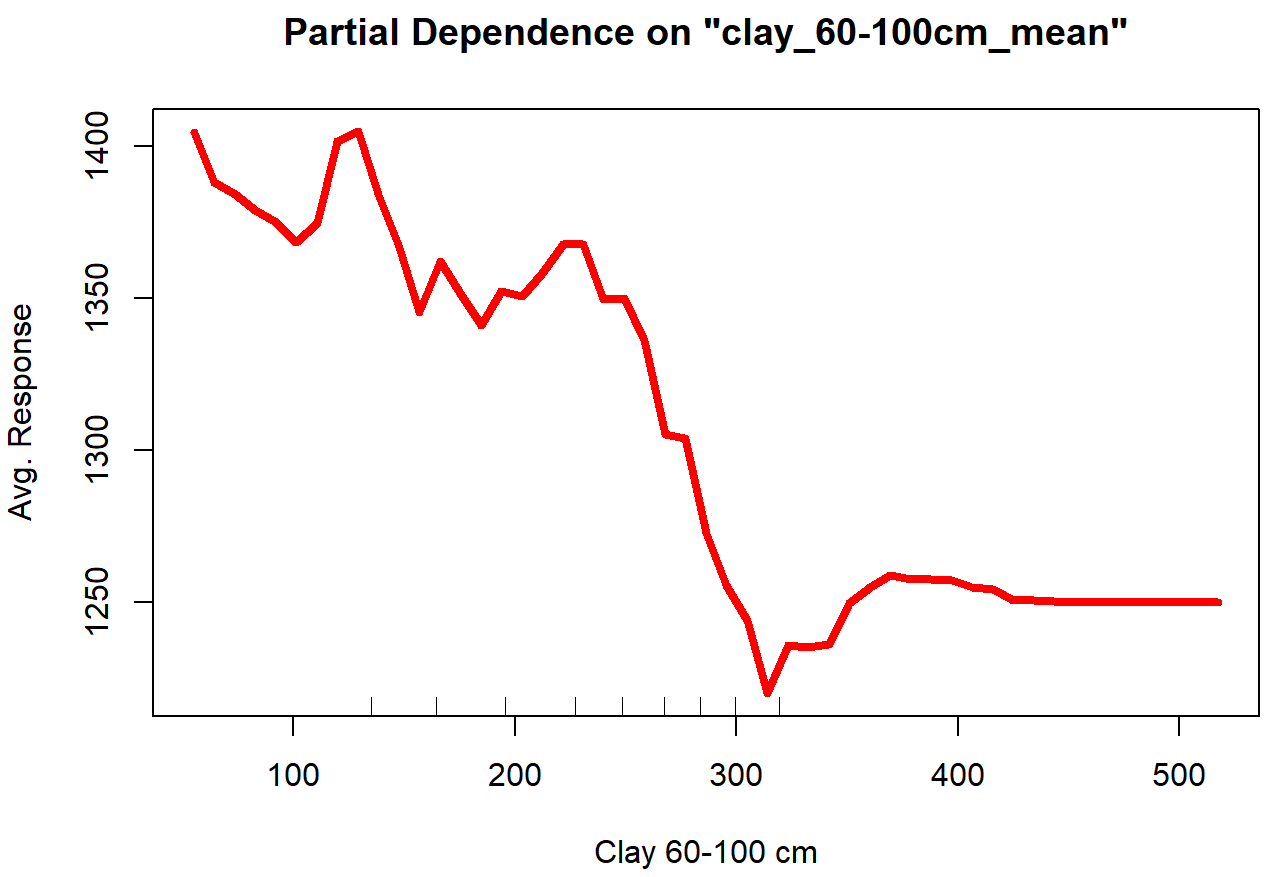
```
# Evaluate the predictors effects
partialPlot(rf_model, predictors, "sand_100-200cm_mean", "1",xlab="Sand 100-200 cm", ylab="Avg. R
```



Soil: Clay 60-100 cm

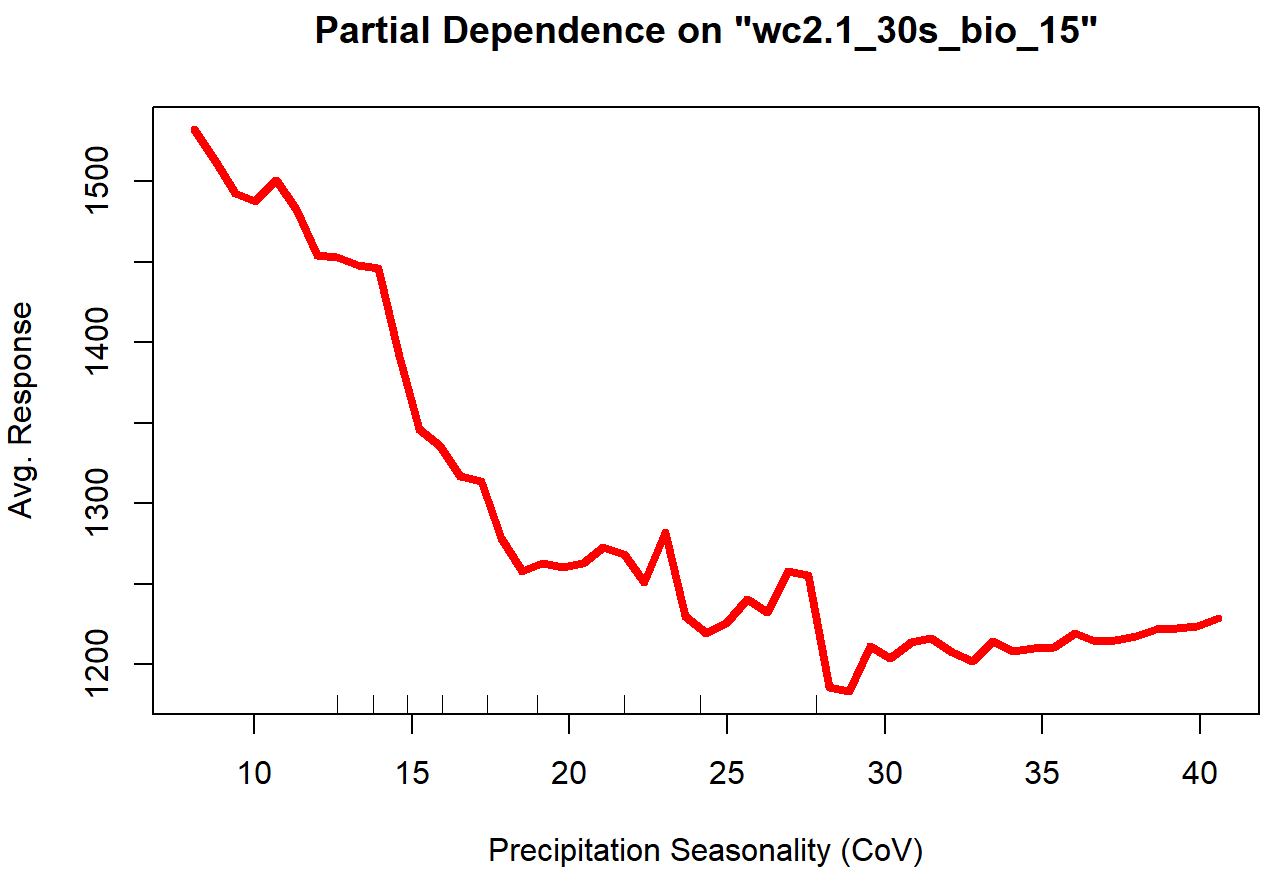
The response is given in the original values (1-10000). Divide by 100 for %-values

```
# Evaluate the predictors effects
partialPlot(rf_model, predictors, "clay_60-100cm_mean", "1",xlab="Clay 60-100 cm", ylab="Avg. Res
```



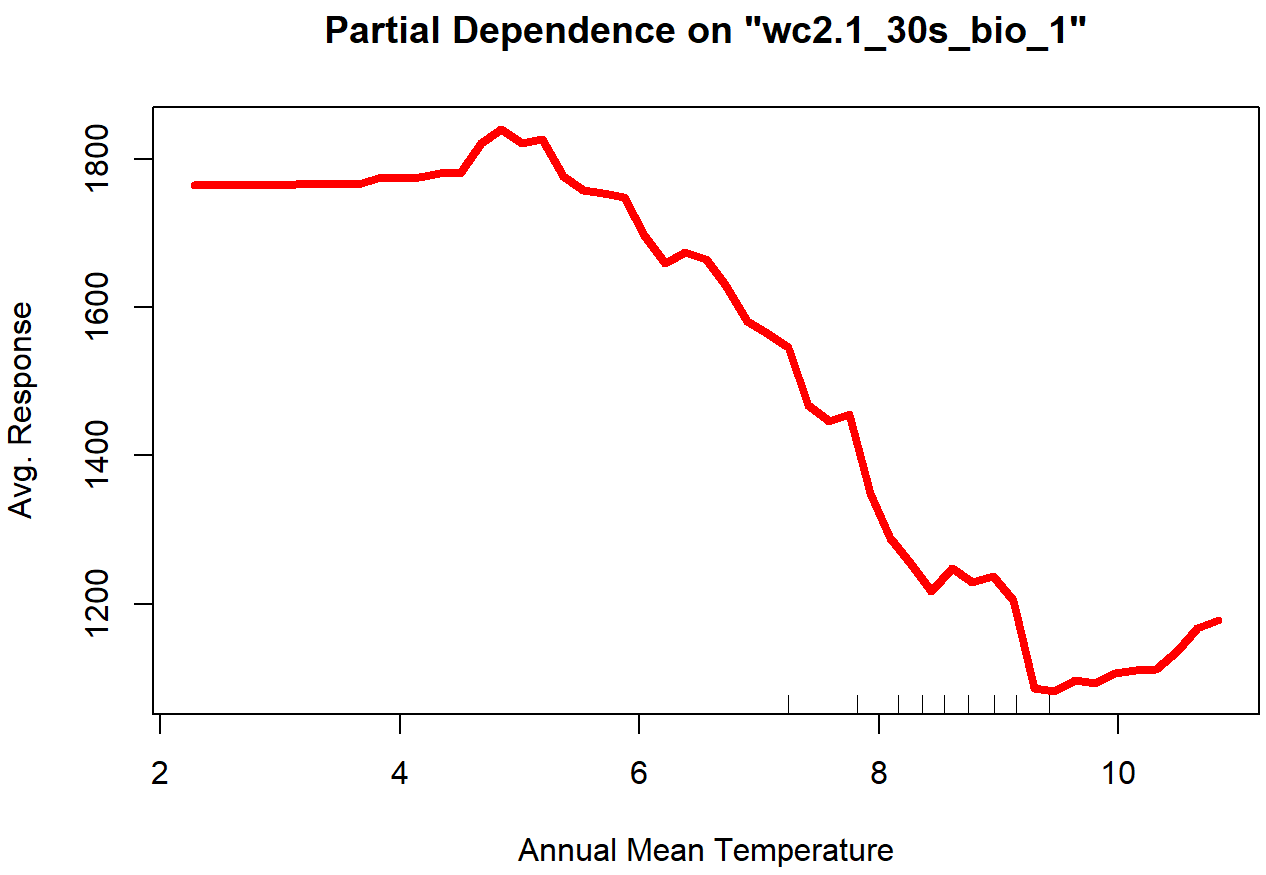
Precipitation Seasonality (wc2.1_30s_bio_15)

```
# Evaluate the predictors effects
partialPlot(rf_model, predictors, "wc2.1_30s_bio_15" , "1",xlab="Precipitation Seasonality (CoV)"
```



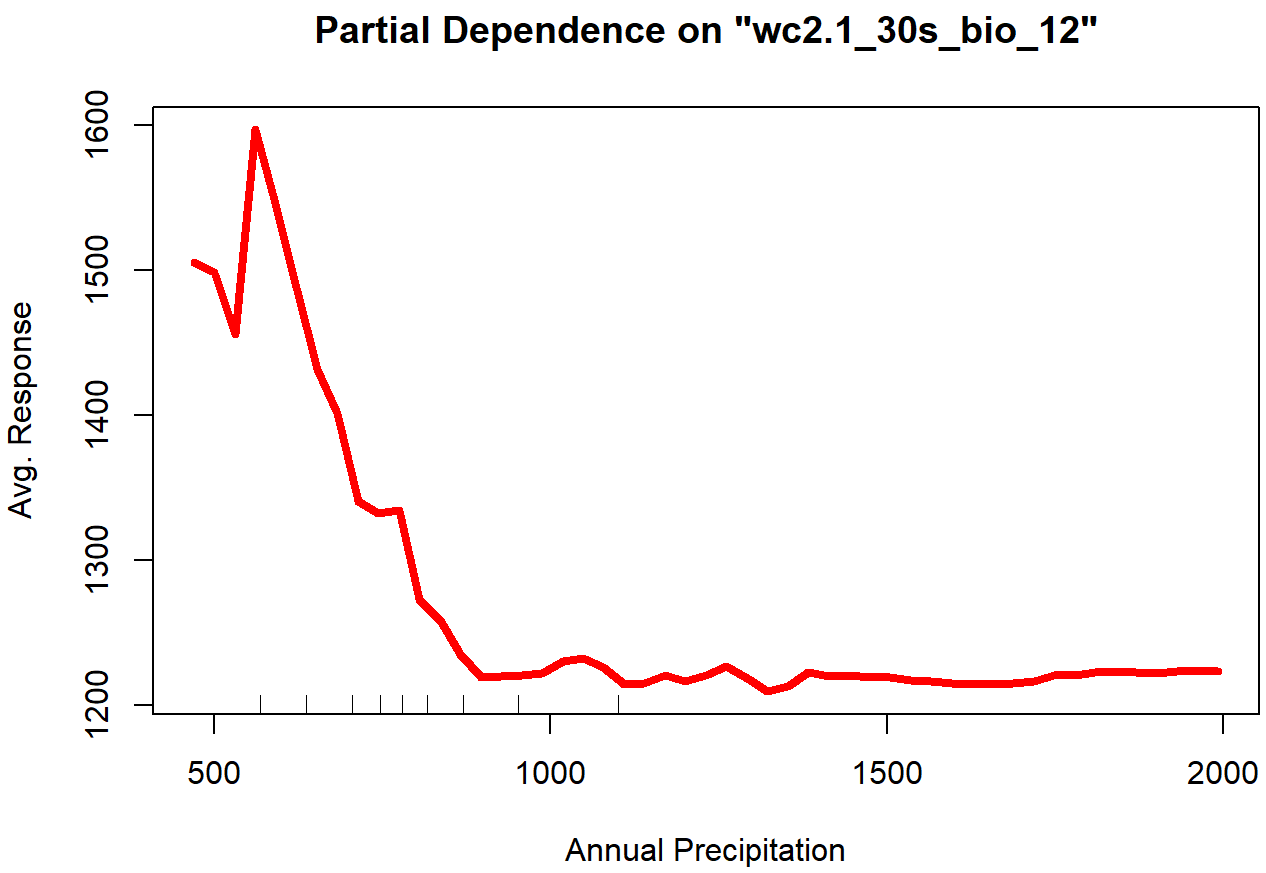
Annual Mean Temperature (wc2.1_30s_bio_1)

```
# Evaluate the predictors effects
partialPlot(rf_model, predictors, "wc2.1_30s_bio_1", "1",xlab="Annual Mean Temperature", ylab="Av
```



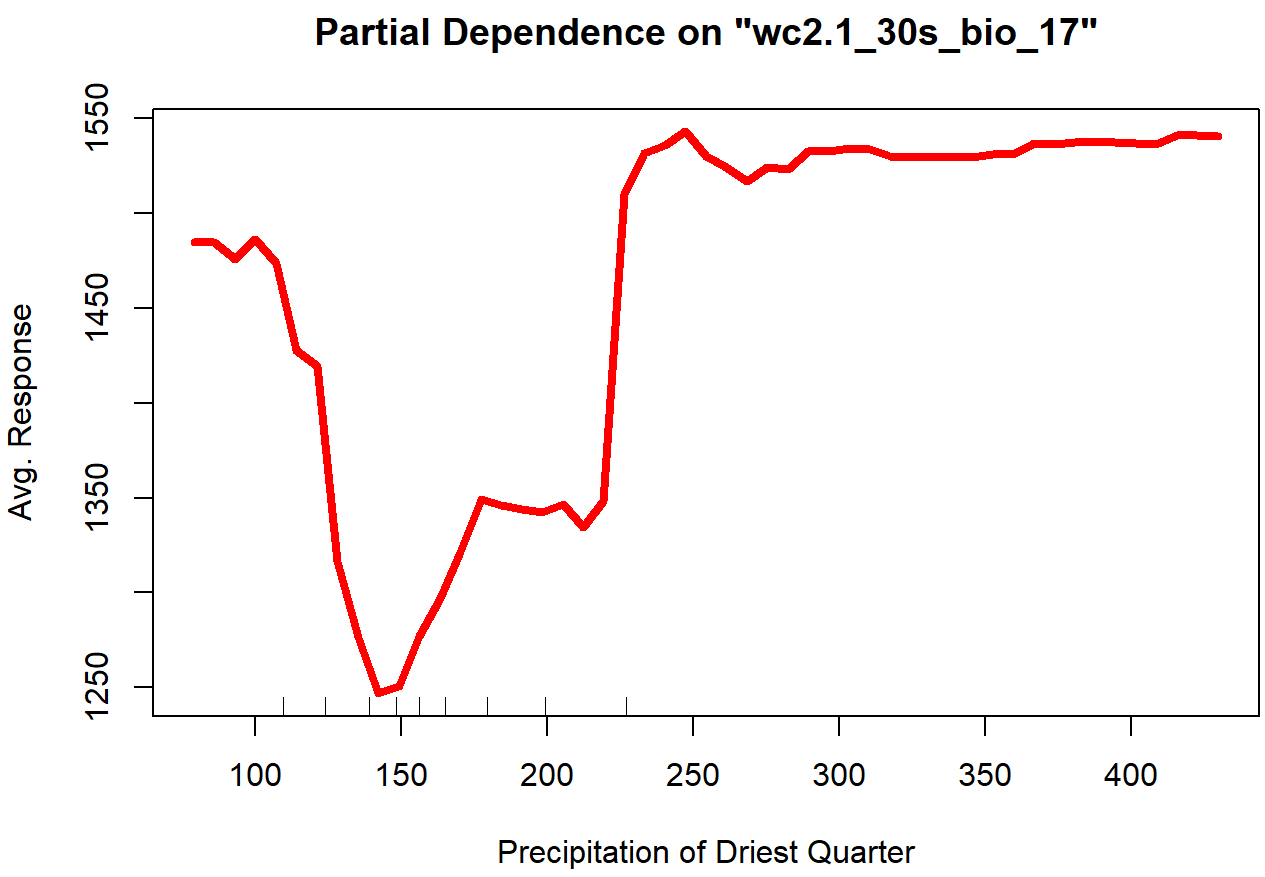
Annual Precipitation (wc2.1_30s_bio_12)

```
# Evaluate the predictors effects
partialPlot(rf_model, predictors, "wc2.1_30s_bio_12" , "1",xlab="Annual Precipitation", ylab="Avg
```



Precipitation of Driest Quarter (wc2.1_30s_bio_17)

```
# Evaluate the predictors effects
partialPlot(rf_model, predictors, "wc2.1_30s_bio_17" , "1",xlab="Precipitation of Driest Quarter"
```


Partial Dependence on "wc2.1_30s_bio_17"
Avg. Response
1250
1350
1450
1550
100
150
200
250
300
350
400
Precipitation of Driest Quarter