Appendix II

Notebook MT5751 - Project 3 Occupancy Estimation

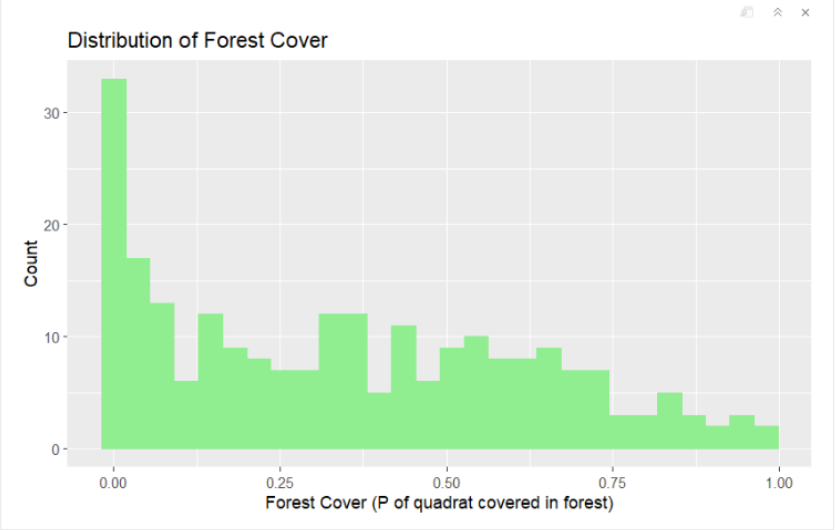
I) Notebook Setup

```r
# Load libraries
library(tidyverse)
library(corrplot)
library(ggplot2)
library(MuMIn)
library(unmarked)
library(tidyr)
library(car)
library(AICcmodavg)
library(dplyr)

# Load Data
library(statsecol)

# Save the data frame
data(willow)
willow_df <- willow
str(willow_df)

willow_df <- as.data.frame(sapply(willow_df, as.numeric))
```

II) Data exploration

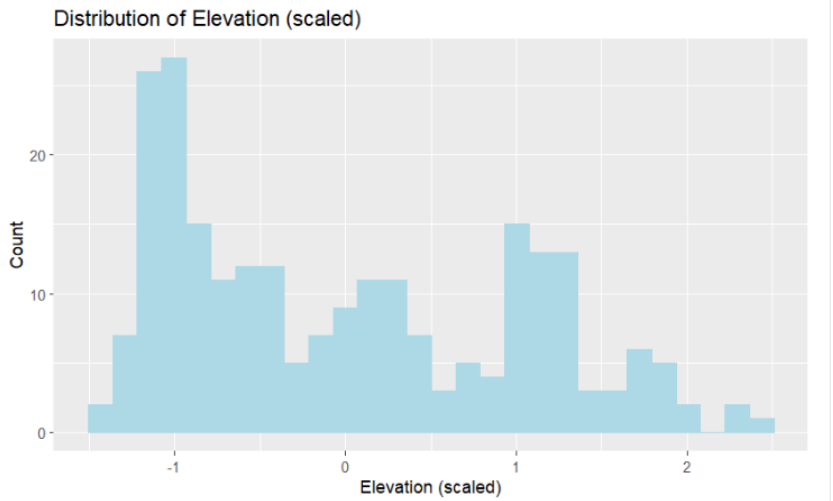
A) Distribution of forest Cover

```r
ggplot(data = willow_df, aes(x = forest)) +
  geom_histogram(bins = 28, fill = 'lightgreen', color = 'lightgreen') +
  labs(title = "Distribution of Forest Cover", x = "Forest Cover (P of quadrat covered in forest)", y = "Count")
```



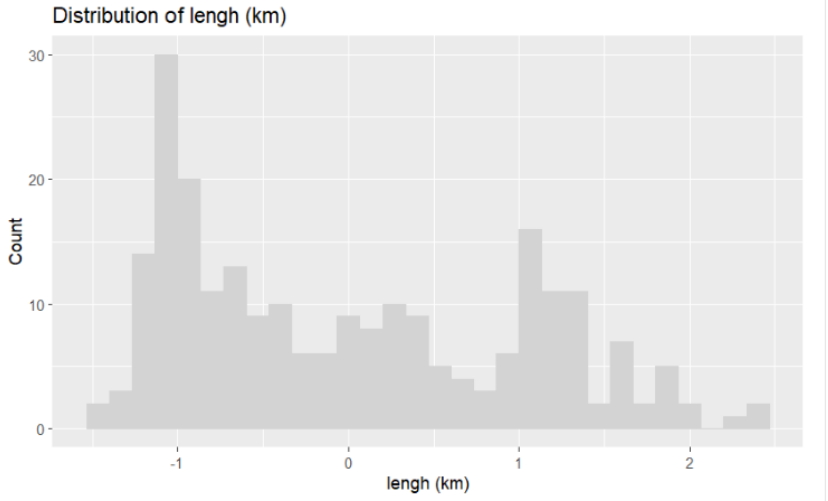
B) Distribution of elevation

```{r}
ggplot(data = willow_df, aes(x = elev)) +
  geom_histogram(bins = 28, fill = 'lightblue', color = 'lightblue') +
  labs(title = "Distribution of Elevation (scaled)", x = "Elevation (scaled)", y = "Count")
```



D) Distribution of lengh

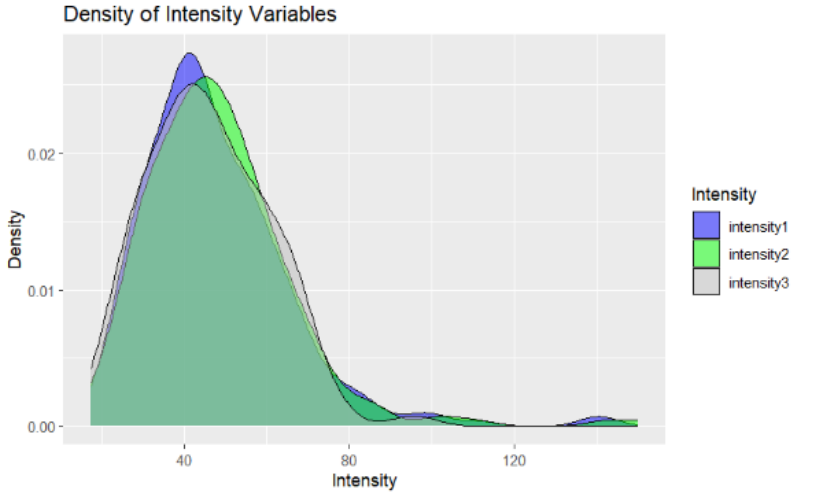
```{r}
ggplot(data = willow_df, aes(x = elev)) +
geom_histogram(bins = 30,  fill = 'lightgrey', color = 'lightgrey') +
labs(title = "Distribution of lengh (km)", x = "lengh (km)", y = "Count")#
Introduction
```



E) Distribution of intensity per visit

```{r}
# Convert intensity to numeric
willow_df$intensity2 <- as.numeric(willow_df$intensity2)
willow_df$intensity3 <- as.numeric(willow_df$intensity3)

# Reshape data
willow_df_long <- pivot_longer(willow_df, cols = starts_with("intensity"), names_to
= "variable", values_to = "value")

# Create plot
effort_willow_df_long <- ggplot(willow_df_long, aes(x = value, fill = variable)) +
  geom_density(alpha = 0.5) +
  xlab("Intensity") +
  ylab("Density") +
  ggtitle("Density of Intensity Variables") +
  scale_fill_manual(name = "Intensity", values = c("intensity1" = "blue",
"intensity2" = "green", "intensity3" = "grey"))

effort_willow_df_long
```

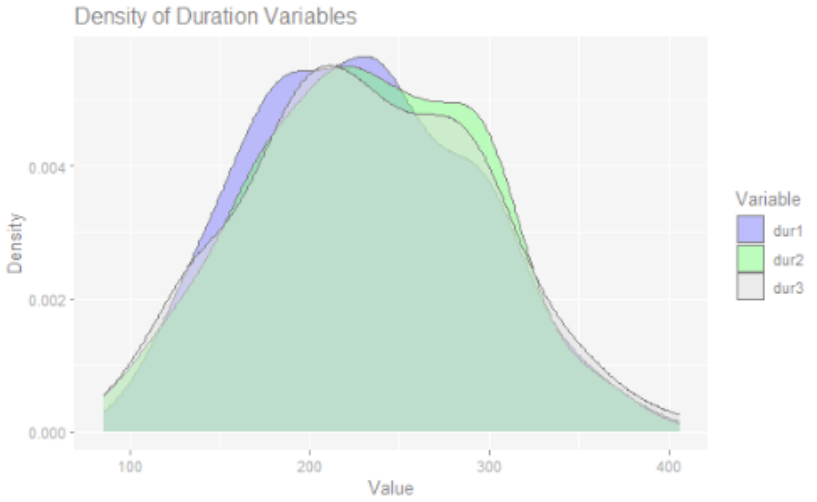
```
⚠ Warning: [38;5;232mRemoved 46 rows containing non-finite values (`stat_density()`).[39m
```



E) Distribution of Duration per visit

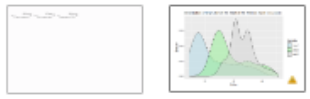
```{r}
# Convert dur to numeric
willow_df$dur2 <- as.numeric(willow_df$dur2)
willow_df$dur3 <- as.numeric(willow_df$dur3)

# Reshape data
willow_df_dur <- pivot_longer(willow_df, cols = starts_with("dur"),
                              names_to = "variable", values_to = "value")

# Create a plot
willow_plot_dur <- ggplot(willow_df_dur, aes(x = value, fill = variable)) +
  geom_density(alpha = 0.5) +
  xlab("Value") +
  ylab("Density") +
  ggtitle("Density of Duration Variables") +
  scale_fill_manual(name = "Variable",
                    values = c("dur1" = "blue", "dur2" = "green", "dur3" = "grey"))

willow_plot_dur
```

Warning: [38;5;232mRemoved 46 rows containing non-finite values (`stat_density()`).[39m

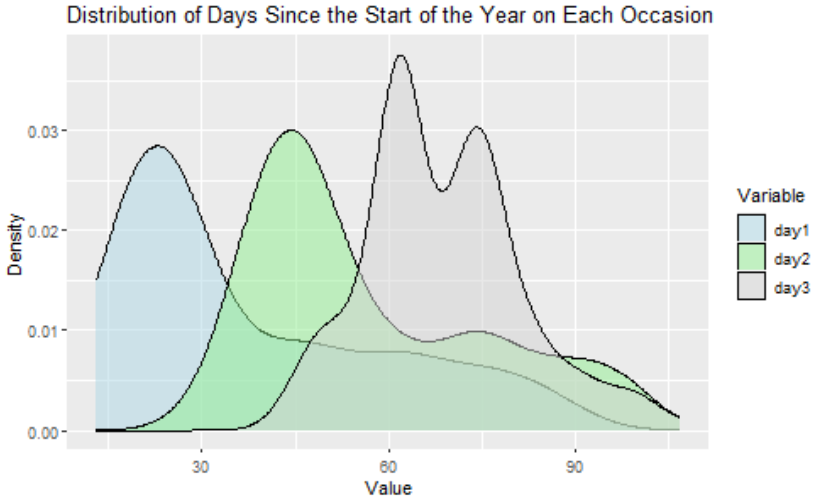


F) Distribution of days since the start of the year on each occasion

```{r}
# Convert day to numeric
willow_df$day3 <- as.numeric(willow_df$day3)
sapply(select(willow_df, starts_with("day")), class)


# Pivot the data
willow_df %>%
  pivot_longer(cols = starts_with("day"), names_to = "variable", values_to =
"value") %>%
  ggplot(aes(x = value, fill = variable)) +
  geom_density(alpha = 0.5) +
  labs(x = "Value", y = "Density", title = "Distribution of Days Since the Start of
the Year on Each Occasion") +
  scale_fill_manual(name = "Variable", values = c("lightblue", "lightgreen",
"lightgrey"))
```



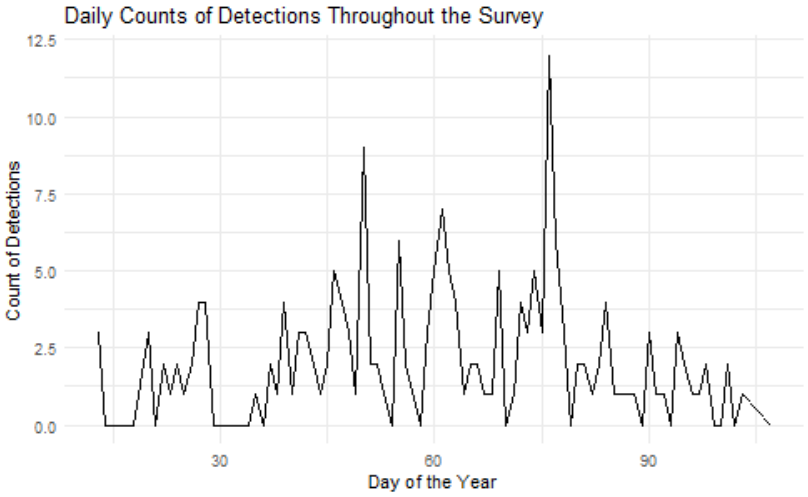
```
⚠ Warning: [38;5;232mRemoved 42 rows containing non-finite values (`stat_density()`).[39m
```



H) Sampling period

```{r}
# Reshape Data
data_range <- melt(willow_df, id.vars = c("day1", "day2", "day3"),
                   measure.vars = c("y.1", "y.2", "y.3"),
                   variable.name = "survey", value.name = "detection")

# Create a new column
data_range$day_of_survey <- ifelse(data_range$survey == "y.1", data$day1,
                                   ifelse(data_range$survey == "y.2", data$day2,
data$day3))

# Aggregate counts
Counts <- data_range %>%
  group_by(day_of_survey) %>%
  summarise(detection_count = sum(detection, na.rm = TRUE))

# Plot
ggplot(Counts, aes(x = day_of_survey, y = detection_count)) +
  geom_line() +
  labs(title = "Daily Counts of Detections Throughout the Survey",
       x = "Day of the Year", y = "Count of Detections") +
  theme_minimal()
```

```
Warning: [38;5;232mRemoved 1 row containing missing values (`geom_line()`).[39m
```



III) "unmarked" frame object

```{r}
# Create Dataframe
UM_willow_df <- unmarkedFrameOccu(
  y = willow_df[, c("y.1","y.2","y.3")],
  siteCovs = data.frame(forest = willow_df$forest,
                        elev = willow_df$elev,
                        length = willow_df$length,
                        elevsq = willow_df$elevsq),
  obsCovs = list(intensity = willow_df[, c("intensity1", "intensity2", "intensity3"
)],
                 dur = willow_df[, c("dur1","dur2","dur3")],
                 day = willow_df[, c("day1","day2","day3")])
)
summary(UM_willow_df)
```

```
unmarkedFrame Object

237 sites
Maximum number of observations per site: 3
Mean number of observations per site: 2.81
Sites with at least one detection: 79

Tabulation of y observations:
   0    1 <NA>
 483  182   46

Site-level covariates:
     forest             elev              length          elevsq
 Min.   :0.0000   Min.   :-1.4429   Min.   :1.200   Min.   :0.000016
 1st Qu.:0.0800   1st Qu.:-0.9478   1st Qu.:4.200   1st Qu.:0.219154
 Median :0.3300   Median :-0.1742   Median :5.000   Median :0.957877
 Mean   :0.3479   Mean   : 0.0000   Mean   :5.103   Mean   :0.995781
 3rd Qu.:0.5700   3rd Qu.: 0.9862   3rd Qu.:5.900   3rd Qu.:1.337285
 Max.   :0.9800   Max.   : 2.4251   Max.   :9.400   Max.   :5.881141

Observation-level covariates:
   intensity           dur              day
 Min.   : 17.31   Min.   : 85.0   Min.   : 13.00
 1st Qu.: 35.96   1st Qu.:180.0   1st Qu.: 38.00
 Median : 45.00   Median :230.0   Median : 52.00
 Mean   : 47.39   Mean   :230.7   Mean   : 53.66
 3rd Qu.: 56.82   3rd Qu.:280.0   3rd Qu.: 72.00
 Max.   :150.00   Max.   :406.0   Max.   :107.00
 NA's   :46       NA's   :46      NA's   :42
```

VI) Full Vs Null Model Fit

```{r}
# Full model with all covariates
willow_f <- occu(formula = ~ day + intensity + dur ~ length + forest + elev + elevsq
, data = UM_willow_df)

# Output
willow_fu <- summary(willow_f)
```

Description: df [4 × 4]

	Estimate <dbl>	SE <dbl>	z <dbl>	P(>|z|) <dbl>
(Intercept)	-1.177485517	0.911853607	-1.2913098	0.1965962668
day	0.004413742	0.008216376	0.5371885	0.5911374383
intensity	-0.027203567	0.014335565	-1.8976278	0.0577451263
dur	0.014312274	0.004274964	3.3479285	0.0008141802

4 rows

```{r}
# Null model with no covariates
willow_n <- occu(formula = ~ 1 ~ 1, data = UM_willow_df)

# Output
willow_nu <- summary(willow_n)
```

Description: df [1 × 4]

	Estimate <dbl>	SE <dbl>	z <dbl>	P(>|z|) <dbl>
(Intercept)	1.324917	0.1741857	7.606344	2.819586e-14

1 row

VI) Model Selection

```{r}
# Define full model with interactions
full <- occu(formula = ~ day + dur + intensity +  day:dur +  day:intensity +
intensity:dur ~ length + elev + elevsq + forest + forest:elev + forest:elevsq, data
= UM_willow_df)

# dredge
Dredge_output_with_interaction <- suppressWarnings(dredge(full, rank = "AIC"))

# Print the first 5 rows of the output
Dredge_output_with_interaction[1:5, ]
```

```
Fixed terms are "p(Int)" and "psi(Int)"
Global model call: occu(formula = ~day + dur + intensity + day:dur + day:intensity
+
    intensity:dur ~ length + elev + elevsq + forest + forest:elev +
    forest:elevsq, data = UM_willow_df)
---
Model selection table
     p(Int) psi(Int)   p(day)   p(dur)   p(int) psi(elev) psi(elvs) psi(frs)
psi(lng) psi(elev:frs) psi(elvs:frs) df
3527 -1.035  -1.1070          0.014260 -0.02560     1.152   -0.9610    2.347
3.726          2.908  9
3523 -1.196  -1.0580          0.009622               1.149   -0.9950    2.175
3.376          2.718  8
3528 -1.121  -1.0610 0.003252 0.013980 -0.02603     1.189   -1.0050    2.242
3.485          2.823 10
4039 -1.072  -0.9566          0.014560 -0.02651     1.124   -0.9365    2.419
-0.03579         3.832          2.946 10
4035 -1.158  -1.2620          0.009483               1.183   -1.0110    2.151
0.04346          3.300          2.647  9
       logLik   AIC delta weight
3527 -199.797 417.6  0.00  0.329
3523 -200.882 417.8  0.17  0.302
3528 -199.711 419.4  1.83  0.132
4039 -199.797 419.6  2.00  0.121
4035 -200.841 419.7  2.09  0.116
Models ranked by AIC(x)
```

B) Define best models

```{r}
# Best model 1
Best_1 <- occu(formula = ~ dur + intensity ~  elev + elevsq + forest + (elev:forest
) + (elevsq:forest),  data=UM_willow_df)

#  Best model 2
Best_2 <- occu(formula = ~ dur ~  elev + forest + elevsq  + (elev:forest) + (elevsq
:forest), data=UM_willow_df)

#  Best model 3
Best_3 <- occu(formula = ~ day + intensity + dur ~  elev + forest + elevsq  + (elev
:forest) + (elevsq:forest), data=UM_willow_df)

#  Best model 4
Best_4 <- occu(formula = ~ intensity + dur ~  length + elev + forest + elevsq  +
(elev:forest) + (elevsq:forest), data=UM_willow_df)

#  Best model 5
Best_5 <- occu(formula = ~ intensity + dur ~  elev + forest + elevsq  + (elev
:forest) , data=UM_willow_df)

#  Best model 6
Best_6 <- occu(formula = ~ intensity + dur ~  elev + forest + elevsq, data
=UM_willow_df)
```

C) regularized occupancy

```{r}
# Define a range of lambda values to test
lamb_vals <- 10^seq(-4, -1, length.out = 30)

# Initialize vector
AIC_vals <- numeric(length(lamb_vals))

# Loop over lambda values, fit model each time, and record AIC
for (i in seq_along(lamb_vals)) {
    model <- occuPEN(~ intensity + dur ~ elev + forest + elevsq + (elev:forest),
                     data = UM_willow_df,
                     lambda = lamb_vals[i])
    AIC_vals[i] <- AIC(model)
}

# Find lambda with lowest AIC
min_lamb <- lamb_vals[which.min(AIC_vals)]

# Fit best model
Best_5_reg <- occuPEN(~ intensity + dur ~ elev + forest + elevsq + (elevsq:forest),
                      data = UM_willow_df,
                      lambda = min_lamb)
```

VI) Model Evaluation

A) Multicolinearity

```{r}
# Best model 1
vif_results <- vif(Best_1, type = 'state')
print("VIF Results for Best Model 1:")
print(vif_results)
vif_results_1 <- vif(mod = Best_1, type = 'det')
print(vif_results_1)

# Best model 2
print("VIF Results for Best Model 2:")
vif_results_1 <- vif(mod = Best_2, type = 'state')
print(vif_results_1)

# Best model 3
vif_results <- vif(mod = Best_3, type = 'state')
print("VIF Results for Best Model 3:")
print(vif_results)
vif_results_1 <- vif(mod = Best_3, type = 'det')
print(vif_results_1)

# Best model 4
vif_results <- vif(mod = Best_4, type = 'state')
print("VIF Results for Best Model 4:")
print(vif_results)
vif_results_1 <- vif(mod = Best_4, type = 'det')
print(vif_results_1)

# Best model 5
vif_results <- vif(mod = Best_5, type = 'state')
print("VIF Results for Best Model 5:")
print(vif_results)
vif_results_1 <- vif(mod = Best_5, type = 'det')
print(vif_results_1)

# Best model 5
vif_results <- vif(mod = Best_5, type = 'state')
print("VIF Results for Best Model 5:")
print(vif_results)
vif_results_1 <- vif(mod = Best_5, type = 'det')
print(vif_results_1)

# Best model 6
vif_results <- vif(mod = Best_6, type = 'state')
print("VIF Results for Best Model 6:")
print(vif_results)
vif_results_1 <- vif(mod = Best_6, type = 'det')
print(vif_results_1)
```

B) Goodness-of-fit

I) Model with both interactions

```{r}
# Create a suppress warning wrapper
suppressWarningsWrapper <- function(...) {
  suppressWarnings({
    mb.gof.test(...)})}

GOF_WILLOW_1 <- suppressWarningsWrapper(Best_1, nsim = 5, plot.hist = FALSE)
print(GOF_WILLOW_1)
```

II) Model with one interactions

```{r}
GOF_WILLOW_2 <- suppressWarningsWrapper(Best_5, nsim = 5, plot.hist = FALSE)

print(GOF_WILLOW_2)
```

III) Model with no interaction

```{r}
GOF_WILLOW_3 <- suppressWarningsWrapper(Best_5, nsim = 50, plot.hist = FALSE)

print(GOF_WILLOW_3)
```

B) Cross validation

I) Model with both interactions

```
{r}
(kfold_1 = crossVal(Best_1, method="Kfold", folds=10))
```

Description: df [1 × 2]

	Estimate <dbl>	SD <dbl>
Mean absolute err...	0.2396447	0.036333

1 row

II) Model with one interactions

```
{r}
(kfold_2 = crossVal(Best_5, method="Kfold", folds=10))
```

Description: df [1 × 2]

	Estimate <dbl>	SD <dbl>
Mean absolute err...	0.253979	0.05142638

1 row

III) Model with no interaction

```
{r}
(kfold_3 = crossVal(Best_6, method="Kfold", folds=10))
```

Description: df [1 × 2]

	Estimate <dbl>	SD <dbl>
Mean absolute err...	0.2591313	0.03737407

1 row

IV) Model With one interaction and regularization

```
{r}
(kfold_3 = crossVal(Best_5_reg, method="Kfold", folds=10))
```

Description: df [1 × 2]

	Estimate <dbl>	SD <dbl>
Mean absolute err...	0.2489738	0.02977029

1 row

VII) Effects plots

A) Detection

a) Duration

```{r}
# Create a dummy data frame with the mean intensity
Duration_df <- data.frame(dur = seq(
  min(UM_willow_df@obsCovs$dur, na.rm=TRUE),
                                    max(UM_willow_df@obsCovs$dur, na.rm=TRUE),
                                    length = 30),
                          intensity = mean(UM_willow_df@obsCovs$intensity, na.rm=TRUE
))

# Create predictions
Duration_df <- predict(Best_5,type="det",newdata = Duration_df, append = TRUE)

# Plot the results
Effects_duration <- ggplot(data = Duration_df, aes(x = dur, y = Predicted)) +
  geom_ribbon(aes(ymin=lower, ymax=upper), fill="#808080", alpha=0.1) +
  geom_line(size=1,color="#ADD8E6") +
  ylab("Pr(Detection)") + xlab("Survey Duration (Min)") + ylim(0,1) + theme_bw()

Effects_duration
```

b) Intensity

```{r}
# Create dummy data frame
Intensity_df <- data.frame(
  dur = mean(UM_willow_df@obsCovs$dur, na.rm = TRUE),
  intensity = seq(
    min(UM_willow_df@obsCovs$intensity, na.rm = TRUE),
    max(UM_willow_df@obsCovs$intensity, na.rm = TRUE),
    length.out = 30))

# Create predictions
Intensity_df <- predict(Best_5,type="det",newdata = Intensity_df, append = TRUE)

# Plot the results
Effects_Intensity <- ggplot(data = Intensity_df, aes(x = intensity, y = Predicted))
+
  geom_ribbon(aes(ymin=lower, ymax=upper), fill="#808080", alpha=0.1) +
  geom_line(size=1,color="#228B22") +
  ylab("Pr(Detection)") + xlab("Intensity of search (duration/length) ") + ylim(0,1)
+ theme_bw()

Effects_Intensity
```

A) Occupancy

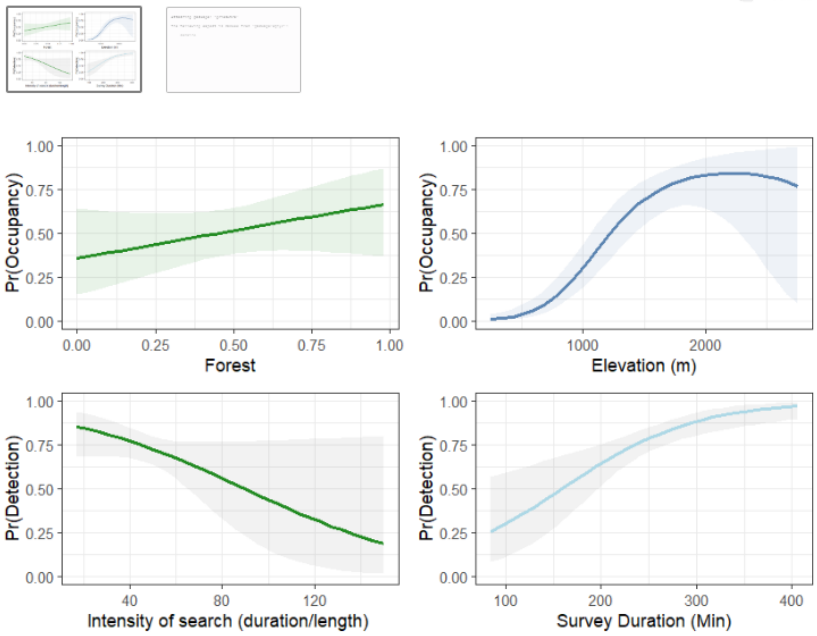
a) Elevation

```{r}
# Set mean and standard deviation of elevation
avgElevation <- 1182.574
elevationSD <- 646.333

# Create sequence so that it can be squared
elev_seq <- seq(min(UM_willow_df@siteCovs$elev, na.rm = TRUE),
                max(UM_willow_df@siteCovs$elev, na.rm = TRUE),
                length = 30)

# Create dummy data frame
elevation_df <- data.frame(
  elev = elev_seq,
  elevsq = elev_seq^2,  # Square elevation
  forest = mean(UM_willow_df@siteCovs$forest, na.rm = TRUE))

# Create predictions
elevation_df <- predict(Best_5, type = "state", newdata = pred_psi_ele, append =
TRUE)

# Plot the results
elevation_effects <- ggplot(data = elevation_df, aes(x = (elev * elevationSD +
avgElevation),, y = Predicted)) +

  geom_ribbon(aes(ymin = lower, ymax = upper), fill = "#5B84B1FF", alpha = 0.1) +
  geom_line(size = 1, color = "#5B84B1FF") +
  ylab("Pr(Occupancy)") + xlab("Elevation (m)") + ylim(0, 1) + theme_bw()

elevation_effects
```

C) Forest

```{r}
# Create prediction data frame
Forest_df <- data.frame(
  forest = seq(min(UM_willow_df@siteCovs$forest, na.rm = TRUE),
               max(UM_willow_df@siteCovs$forest, na.rm = TRUE),
               length = 30),
  elev = mean(UM_willow_df@siteCovs$elev, na.rm = TRUE),
  elevsq = (mean(UM_willow_df@siteCovs$elev, na.rm = TRUE))^2)

# Generate predictions
Forest_df <- predict(Best_5, type = "state", newdata = Forest_df, append = TRUE)

# Create plot
Plot_forest <- ggplot(data = Forest_df, aes(x = forest, y = Predicted)) +
  geom_ribbon(aes(ymin = lower, ymax = upper), fill = "#228B22", alpha = 0.1) +
  geom_line(size = 1, color = "#228B22") +
  ylab("Pr(Occupancy)") + xlab("Forest") + ylim(0, 1) + theme_bw()

Plot_forest
```

```{r}
library(gridExtra)

combined_plot <- grid.arrange(Plot_forest, elevation_effects, Effects_Intensity,
Effects_duration, ncol = 2, nrow = 2)
ggsave("combined_plot.png", combined_plot, width = 10, height = 10)
```



VII) Prediction

A) Detection prediction

```{r}
# Define quadrats
quadList <- c(25, 62, 150, 203)
# Initialize lists
detectionPred <- vector("list", length = length(quadList))
detectionCounter <- 1

# Loop for each quadrat
for (quad in quadList) {
  # Data values
  quadData <- data.frame(intensity = UM_willow_df@obsCovs$intensity[quad],
                         dur = UM_willow_df@obsCovs$dur[quad],
                         elev = UM_willow_df@siteCovs$elev[quad],
                         elevsq = UM_willow_df@siteCovs$elevsq[quad],
                         forest = UM_willow_df@siteCovs$forest[quad])

    # Prediction
  detectionPred[[detectionCounter]] <- predict(Best_5, type = "det", newdata =
quadData, append = TRUE)
  detectionCounter <- detectionCounter + 1}

# Format df
detectionDF <- do.call(rbind, lapply(detectionPred, unlist)) %>% round(3)
rownames(detectionDF) <- paste0("Quad ", quadList)
colnames(detectionDF) <- c("PredictedVal", "Std", "Lower", "Upper", "Int", "Dur",
"Elev", "EleS", "Forest")

# Adjusting elevation
detectionDF[, "Elev"] <- detectionDF[, "Elev"] * elevationSD + avgElevation

detectionDF
```

```
         PredictedVal   Std Lower Upper   Int Dur      Elev  EleS Forest
Quad 25         0.577 0.085 0.408 0.730 43.42 165  620.2643 0.758   0.03
Quad 62         0.748 0.051 0.636 0.834 36.84 210  370.1334 1.581   0.14
Quad 150        0.739 0.068 0.588 0.849 27.54 190 1660.2141 0.546   0.86
Quad 203        0.816 0.042 0.720 0.885 36.92 240 1980.1489 1.522   0.06
```

A) Occupancy prediction

```{r}
# Initialize lists
occupancyPred <- vector("list", length = length(quadList))
occupancyCounter <- 1

# Loop for each quadrat
for (quad in quadList) {
  # apend values to data frame
  quadData <- data.frame(intensity = UM_willow_df@obsCovs$intensity[quad],
                         dur = UM_willow_df@obsCovs$dur[quad],
                         elev = UM_willow_df@siteCovs$elev[quad],
                         elevsq = UM_willow_df@siteCovs$elevsq[quad],
                         forest = UM_willow_df@siteCovs$forest[quad])
  # Prediction
  occupancyPred[[occupancyCounter]] <- predict(Best_5, type = "state", newdata =
quadData, append = TRUE)
  occupancyCounter <- occupancyCounter + 1}

# Format df
occupancyDF <- do.call(rbind, lapply(occupancyPred, unlist)) %>% round(3)
rownames(occupancyDF) <- paste0("Quad ", quadList)
colnames(occupancyDF) <- c("PredictedVal", "Std", "Lower", "Upper", "Int", "Dur",
"Elev", "EleS", "Forest")


# Adjusting elevation
occupancyDF[, "Elev"] <- occupancyDF[, "Elev"] * elevationSD + avgElevation


occupancyDF
```

```
         PredictedVal   Std Lower Upper   Int Dur      Elev  EleS Forest
Quad 25         0.023 0.014 0.007 0.072 43.42 165  620.2643 0.758   0.03
Quad 62         0.005 0.004 0.001 0.021 36.84 210  370.1334 1.581   0.14
Quad 150        0.934 0.035 0.822 0.977 27.54 190 1660.2141 0.546   0.86
Quad 203        0.482 0.086 0.321 0.646 36.92 240 1980.1489 1.522   0.06
```