Modeling of whale calling behavior

2025-05-13

This code combines univariate and multivariate analyses of whale calls.

In RStudio, use File -> Compile Report to generate PDF from this R file.

```
# Clean the workspace
rm(list = ls())

# Packages ----

# data manipulation
library(chron)
library(dplyr)
library(tidyr)
library(readxl)
library(psych)
library(lubridate)
library(suncalc)

# plots
library(ggplot2)
theme_set(theme_light())
library(patchwork)
library(GGally)
library(gridExtra)

# models
library(flexmix)
library(car)
library(nlme)
library(randomForestSRC)
library(olsrr)
# library(rpart)
# library(rpart.plot)
# library(mgcv)
library(randomForest)
# library(ranger)

NTREE <- 500
MTRY <- 2
NSIZE <- 5

get_decimal_hour <- function(dt) {
    lubridate::hour(dt) + lubridate::minute(dt) / 60 + lubridate::second(dt) / 3600
}
```

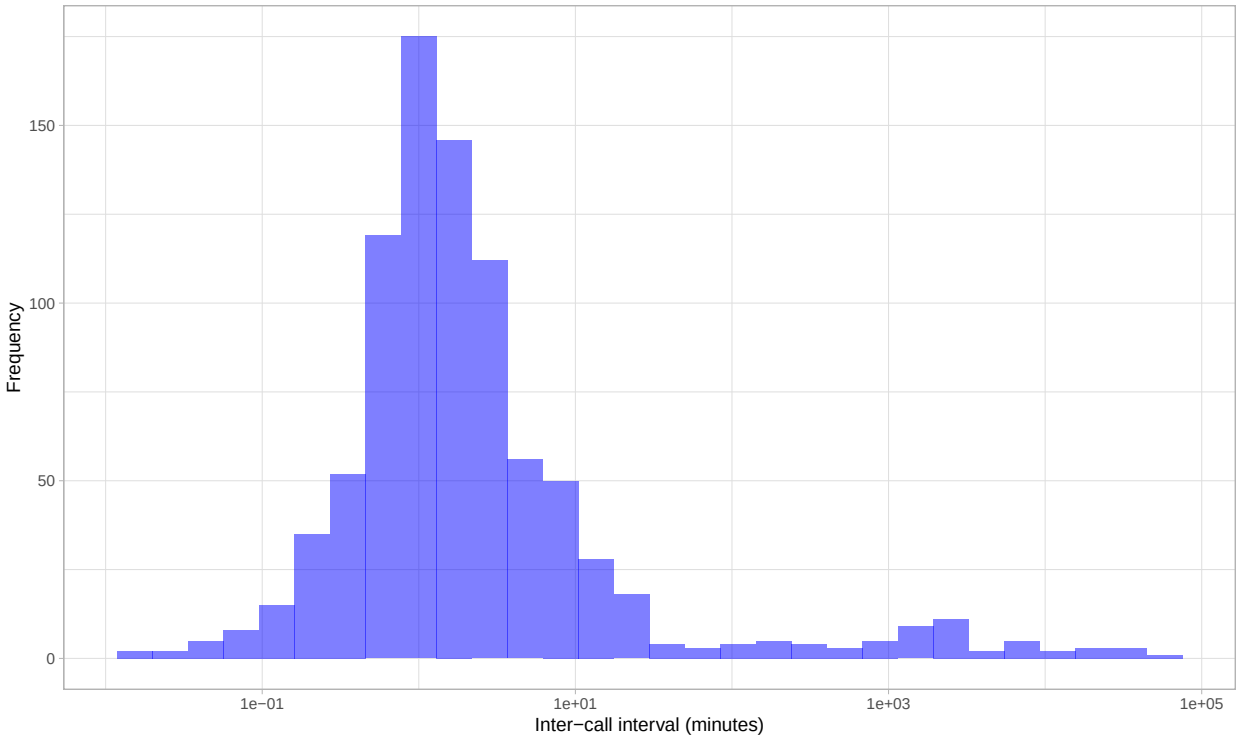
```r
# Get partial dependence data for all predictors
get_pdp_data <- function(model, data, predictors, ...) {
    require(pdp)
    pdp_list <- lapply(predictors, function(pred) {
        pd <- pdp::partial(model,
                           pred.var = pred,
                           train = data,
                           plot = FALSE,
                           ...)
        pd$predictor <- pred
        names(pd)[1] <- "x"
        pd
    })
    do.call(rbind, pdp_list)
}

# Inverse of log10, such as for inv.link = invlog10 in get_pdp_data()
invlog10 <- function(x) {
    10^x
}

# Opposite of in
`%!in%` <- Negate(`%in%`)

# Data ----

BuoyReplacement <- as.Date(c("2022-07-20"))

# Load and format the data
D <- readxl::read_xlsx("./data/NARWData_2024-07-29.xlsx",
                       sheet = 1,
                       col_types = c(rep("guess", 3),
                                     rep("numeric", 3),
                                     rep("guess", 9))) %>%
    # format dates and times
    mutate(Date = as.Date(Date),
           Time_NYC = chron::times(Time_NYC),
           TimeLastVP_NYC = chron::times(TimeLastVP_NYC)) %>%
    mutate(IsVPYesterday = TimeLastVP_NYC > Time_NYC) %>%
    mutate(Time_NYC = as.POSIXct(paste(Date, Time_NYC), tz = "America/New_York"),
           TimeLastVP_NYC = as.POSIXct(paste(Date - IsVPYesterday, TimeLastVP_NYC),
                                       tz = "America/New_York")) %>%
    # calculate time variables
    mutate(TimeSinceLastVP = as.numeric(difftime(Time_NYC, TimeLastVP_NYC, units = "hours")),
           Year = as.numeric(format(Time_NYC, "%Y")),
           Month = as.numeric(format(Time_NYC, "%m")),
           Hour = as.numeric(format(Time_NYC, "%H")),
           HourDecimal = get_decimal_hour(Time_NYC),
           DoY = as.numeric(format(Time_NYC, "%j"))) %>%
    mutate(DoYsin = sin(2 * pi * DoY / 365.25),
           DoYcos = cos(2 * pi * DoY / 365.25),
           Hoursin = sin(2 * pi * HourDecimal / 24),
           Hourcos = cos(2 * pi * HourDecimal / 24)) %>%
```

```
    # define seasons based on months
    mutate(Season = case_when(
        Month %in% 9:11 ~ "Fall",
        Month %in% c(12, 1, 2) ~ "Winter",
        Month %in% 3:5 ~ "Spring",
        TRUE ~ "Summer")) %>%
    # order seasons chronologically
    mutate(Season = factor(Season, levels = c("Fall", "Winter", "Spring", "Summer"))) %>%
    # drop unused factor levels
    mutate(Season = droplevels(Season)) %>%
    # rename some variables
    rename(Duration = DeltaTime,
           Duration90 = Dur90,
           SPLvessel = SPL_VesselPassage,
           SPLcalling = SPL_CallingPeriod,
           TimeSinceLVP = TimeSinceLastVP) %>%
    # sort chronologically
    arrange(Time_NYC) %>%
    # calculate inter-call intervals
    mutate(ICI = c(NA, as.numeric(difftime(Time_NYC[-1], Time_NYC[-length(Time_NYC)], units = "secs"))))
    # year of deployment
    mutate(YearDeployment = case_when(
        Date < BuoyReplacement ~ "Year 1",
        Date > BuoyReplacement ~ "Year 2"))

# Set the ICI between years of observations as missing (NA).
# Note that the end of the previous season is March and start of the season is November.
tmp_PreviousMonth <- c(NA, D$Month[-nrow(D)])
YearSwitch <- which(!is.na(tmp_PreviousMonth) & (tmp_PreviousMonth == 3 & D$Month == 11))
D$ICI[YearSwitch] <- NA

# Show summary
summary(D)
```

```
##  SelectionNumber      Date                Year          Month
##  Min.   : 1.00   Min.   :2021-11-11   Min.   :2021   Min.   : 1.000
##  1st Qu.: 5.00   1st Qu.:2022-12-20   1st Qu.:2022   1st Qu.: 2.000
##  Median :13.00   Median :2023-02-02   Median :2023   Median : 2.000
##  Mean   :17.52   Mean   :2022-11-17   Mean   :2023   Mean   : 4.377
##  3rd Qu.:25.00   3rd Qu.:2023-02-23   3rd Qu.:2023   3rd Qu.: 3.000
##  Max.   :78.00   Max.   :2023-03-13   Max.   :2023   Max.   :12.000
##
##     Time_NYC                      TimeLastVP_NYC
##  Min.   :2021-11-11 06:31:14.0   Min.   :2021-11-11 06:17:40.00
##  1st Qu.:2022-12-20 20:27:30.0   1st Qu.:2022-12-20 17:28:05.00
##  Median :2023-02-02 18:39:32.0   Median :2023-02-02 15:38:25.00
##  Mean   :2022-11-18 06:22:41.5   Mean   :2022-11-18 03:53:04.42
##  3rd Qu.:2023-02-23 22:55:25.0   3rd Qu.:2023-02-23 22:27:32.00
##  Max.   :2023-03-13 10:37:43.0   Max.   :2023-03-13 06:14:45.00
##
##   TimeSinceLVP           Hour         SPLcalling      SPLvessel
##  Min.   : 0.01361   Min.   : 0.00   Min.   : 94.5   Min.   :109.1
##  1st Qu.: 0.90861   1st Qu.: 8.00   1st Qu.:104.2   1st Qu.:118.6
```

```
##  Median : 1.91056   Median :18.00   Median :105.8   Median :120.1
##  Mean   : 2.49366   Mean   :14.48   Mean   :106.9   Mean   :121.1
##  3rd Qu.: 3.48417   3rd Qu.:20.00   3rd Qu.:110.0   3rd Qu.:124.7
##  Max.   :11.18444   Max.   :23.00   Max.   :118.9   Max.   :131.0
##
##     FreqMin          FreqMax        FreqDelta         Duration
##  Min.   : 26.70   Min.   :131.7   Min.   : 48.33   Min.   :0.4315
##  1st Qu.: 73.30   1st Qu.:197.3   1st Qu.:109.33   1st Qu.:0.9001
##  Median : 88.00   Median :218.7   Median :130.67   Median :1.0723
##  Mean   : 86.42   Mean   :223.6   Mean   :137.17   Mean   :1.0776
##  3rd Qu.:101.30   3rd Qu.:245.3   3rd Qu.:160.00   3rd Qu.:1.2309
##  Max.   :146.70   Max.   :405.7   Max.   :307.83   Max.   :2.2006
##
##    Duration90     IsVPYesterday    HourDecimal          DoY
##  Min.   :0.2560   Mode :logical   Min.   : 0.035   Min.   :  1.0
##  1st Qu.:0.6400   FALSE:848       1st Qu.: 8.357   1st Qu.: 36.0
##  Median :0.7040   TRUE :41        Median :18.425   Median : 54.0
##  Mean   :0.7503                   Mean   :15.012   Mean   :117.2
##  3rd Qu.:0.8320                   3rd Qu.:20.676   3rd Qu.: 72.0
##  Max.   :1.6640                   Max.   :23.984   Max.   :365.0
##
##      DoYsin            DoYcos          Hoursin           Hourcos
##  Min.   :-0.7607   Min.   :0.2604   Min.   :-1.0000   Min.   :-1.0000
##  1st Qu.: 0.0172   1st Qu.:0.5848   1st Qu.:-0.8593   1st Qu.:-0.1326
##  Median : 0.5805   Median :0.7938   Median :-0.6519   Median : 0.4772
##  Mean   : 0.4306   Mean   :0.7473   Mean   :-0.2872   Mean   : 0.2716
##  3rd Qu.: 0.8112   3rd Qu.:0.9813   3rd Qu.: 0.3643   3rd Qu.: 0.7207
##  Max.   : 0.9655   Max.   :1.0000   Max.   : 1.0000   Max.   : 1.0000
##
##     Season          ICI          YearDeployment
##  Fall  : 27   Min.   :      1   Length:889
##  Winter:668   1st Qu.:     43   Class :character
##  Spring:194   Median :     86   Mode  :character
##               Mean   :  23692
##               3rd Qu.:    214
##               Max.   :3770917
##               NA's   :2
```

```
# Bouts ----

## Define bouts ----

# Plot ICI distribution
D %>%
    ggplot(aes(x = ICI / 60)) +
    geom_histogram(aes(y = ..count..), fill = "blue", alpha = 0.5, color = NA) +
    scale_x_log10() +
    labs(x = "Inter-call interval (minutes)",
         y = "Frequency")
```



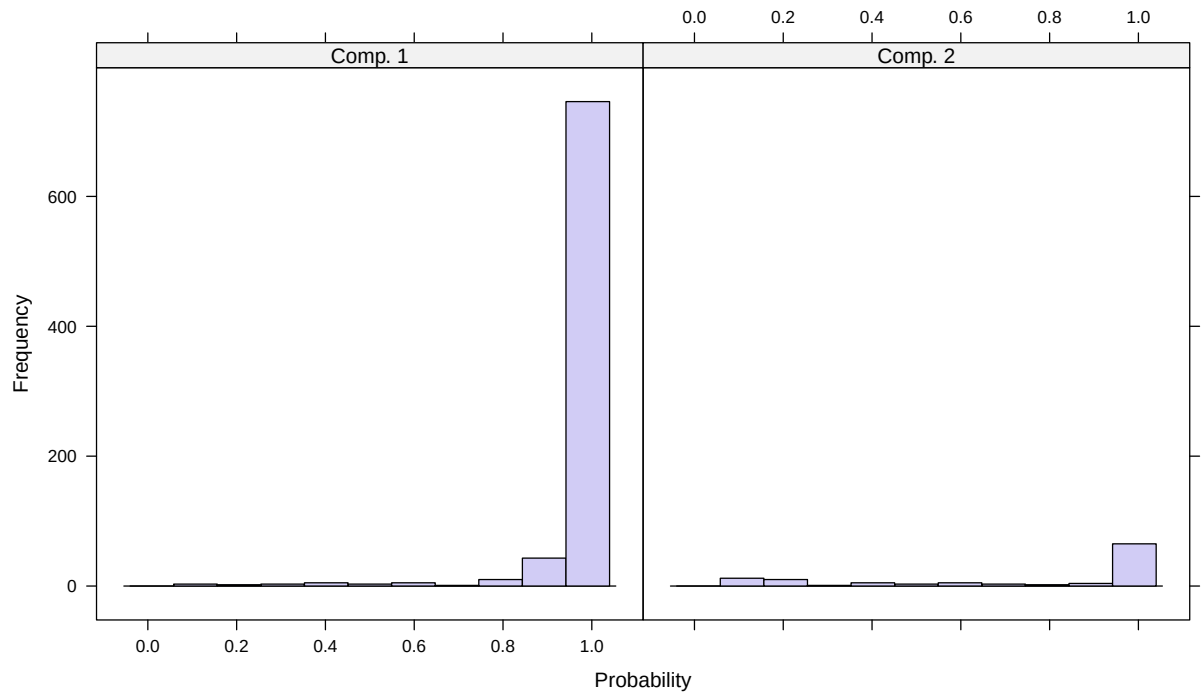

```
ggsave("images/ICI_hist.png", width = 6, height = 4, dpi = 600)

# Fit a mixture distribution (2 gamma distributions)
set.seed(123)
ICIs <- na.omit(D$ICI)
fit <- flexmix(ICIs ~ 1,
               model = FLXMRglm(ICIs ~ ., family = "Gamma"),
               k = 2)
summary(fit)
```

```
##
## Call:
## flexmix(formula = ICIs ~ 1, k = 2, model = FLXMRglm(ICIs ~ .,
##     family = "Gamma"))
##
##        prior size post>0  ratio
## Comp.1 0.887  807    827 0.9758
## Comp.2 0.113   80    887 0.0902
##
## 'log Lik.' -6088.831 (df=5)
## AIC: 12187.66   BIC: 12211.6
```

```
parameters(fit)
```

```
##                       Comp.1       Comp.2
## coef.(Intercept) 0.006996097 4.783319e-06
## shape            0.823787078 1.606166e-01
```

```r
# Plot histogram of posterior probabilities
plot(fit, root = FALSE, main = "", eps = 0.1,
     xlab = "Probability", ylab = "Frequency")
```




```r
png("images/ICI_flexmix.png", width = 6, height = 4, units = "in", res = 600)
plot(fit, root = FALSE, main = "", eps = 0.1,
     xlab = "Probability", ylab = "Frequency")
dev.off()
```

```
## pdf
##   2
```

```r
# In fit@cluster, 1s indicate that the current ICI belongs to the bout,
# and 2s indicate that there's a break between bouts (start of a new bout).
# The ICIs miss the 1st value in each year of deployment (which are NA), so it is n + 2.
# Sometimes the definition of 1 and 2 switches.

# Summarize ICIs (minutes) by cluster
psych::describeBy(ICIs/60, fit@cluster)
```

```
##
##  Descriptive statistics by group
## group: 1
##    vars   n mean   sd median trimmed  mad  min   max range skew kurtosis   se
## X1    1 807 2.38 3.04   1.27    1.69 1.11 0.02 18.28 18.27 2.58     7.17 0.11
## ------------------------------------------------------------
## group: 2
##    vars  n   mean       sd median trimmed     mad   min      max    range skew
```

```
## X1    1 80 4354.1 10148.96 762.69 1647.35 1098.79 18.78 62848.62 62829.83 3.64
##    kurtosis      se
## X1     14.7 1134.69
```

```r
# Define bouts
D$Bout <- NA
# Make sure that calls at the YearSwitch are also the breaks
ICIclusters <- fit@cluster
for (i in 1:length(YearSwitch)) {
    ii <- YearSwitch[i]
    ICIclusters <- c(ICIclusters[1:(ii - 2)], 2, ICIclusters[-c(1:(ii - 2))])
}

bi <- 1
D$Bout[1] <- bi
for (i in 1:length(ICIclusters)) {
    # If a break, start a new bout (increase bout index bi)
    if (ICIclusters[i] == 2) {
        bi <- bi + 1
    }
    D$Bout[i + 1] <- bi
}

# Summarize data for bout analysis
DBO <- D %>%
    mutate(Bout = as.factor(Bout)) %>%
    group_by(Bout) %>%
    summarise(Date = Date[1],
              Year = Year[1],
              Month = Month[1],
              DoY = DoY[1],
              Hour = Hour[1],
              Season = Season[1],
              DoYcos = DoYcos[1],
              DoYsin = DoYsin[1],
              Hoursin = Hoursin[1],
              Hourcos = Hourcos[1],
              HourDecimal = HourDecimal[1],

              ICI = mean(ICI, na.rm = TRUE),
              n_calls = n(),
              BoutDur_s = as.numeric(difftime(tail(Time_NYC, 1), Time_NYC[1], units = "secs")),

              Time_NYC = Time_NYC[1],
              TimeSinceLVP = TimeSinceLVP[1],
              SPLcalling = mean(SPLcalling),
              SPLvessel = mean(SPLvessel),

              FreqMin_first = FreqMin[1],
              FreqMin_last = tail(FreqMin, 1),
              FreqMin = mean(FreqMin),

              FreqMax_first = FreqMax[1],
              FreqMax_last = tail(FreqMax, 1),
```

```
            FreqMax = mean(FreqMax),

            FreqDelta_first = FreqDelta[1],
            FreqDelta_last = tail(FreqDelta, 1),
            FreqDelta = mean(FreqDelta),

            Duration_first = Duration[1],
            Duration_last = tail(Duration, 1),
            Duration = mean(Duration),

            Duration90_first = Duration90[1],
            Duration90_last = tail(Duration90, 1),
            Duration90 = mean(Duration90)
    ) %>%
    # convert duration to minutes
    mutate(BoutDur_mins = BoutDur_s / 60) %>%
    mutate(BoutDur_mins_log10 = log10(BoutDur_mins))

# Total number of bouts
max(D$Bout)
```

```
## [1] 82
```

```
# Summary of bout stats
DB0 %>%
    dplyr::select(BoutDur_mins, n_calls) %>%
    psych::describe()
```

```
##              vars  n  mean    sd median trimmed   mad min    max  range skew
## BoutDur_mins    1 82 23.45 24.24  15.57   19.28 16.93   0 125.82 125.82 1.91
## n_calls         2 82 10.84 11.73   7.50    8.58  6.67   1  78.00  77.00 2.92
##              kurtosis   se
## BoutDur_mins     4.14 2.68
## n_calls         11.85 1.30
```

```
# Count bouts with one call only
db1 <- DB0 %>%
    filter(n_calls == 1)
nrow(db1)
```

```
## [1] 4
```

```
# Remove the short bouts from analysis
DB <- DB0 %>%
    filter(n_calls > 1)

write.csv(DB, file = "dataderived/data_bouts.csv", row.names = FALSE)


## Bout summary ----

# Save numeric summary of the data as a table
```

```r
summary_DB <- psych::describe(DB) %>%
    as.data.frame() %>%
    dplyr::select(n, min, mean, median, max, sd, range)

# Show summary to match with the table in the paper
summary_DB
```

```
##                        n           min          mean       median          max
## Bout*                 78    1.00000000    41.6794872   41.5000000 8.200000e+01
## Date                  78           Inf           NaN           NA         -Inf
## Year                  78 2021.00000000  2022.5256410 2023.0000000 2.023000e+03
## Month                 78    1.00000000     4.3846154    2.0000000 1.200000e+01
## DoY                   78    1.00000000   118.2179487   55.5000000 3.650000e+02
## Hour                  78    0.00000000    13.2564103   16.0000000 2.300000e+01
## Season*               78    1.00000000     2.1923077    2.0000000 3.000000e+00
## DoYcos                78    0.26037648     0.7472356    0.8142924 9.999908e-01
## DoYsin                78   -0.76072009     0.4246142    0.5376774 9.655072e-01
## Hoursin               78   -0.99891033    -0.1842925   -0.3873657 9.997852e-01
## Hourcos               78   -0.99943076     0.1796920    0.3397293 9.999736e-01
## HourDecimal           78    0.88444444    13.7053241   16.3327778 2.397222e+01
## ICI                   78   27.50000000 51706.2571715 5385.3500000 1.236674e+06
## n_calls               78    2.00000000    11.3461538    8.0000000 7.800000e+01
## BoutDur_s             78   30.00000000  1479.0897436  962.0000000 7.549000e+03
## Time_NYC              78           Inf           NaN           NA         -Inf
## TimeSinceLVP          78    0.01361111     2.2629630    1.7429167 1.102028e+01
## SPLcalling            78   94.50000000   107.8358974  107.5500000 1.189000e+02
## SPLvessel             78  109.10000000   122.2294872  123.1500000 1.310000e+02
## FreqMin_first         78   26.70000000    86.8487179   85.3000000 1.467000e+02
## FreqMin_last          78   32.00000000    84.2820513   84.2000000 1.253000e+02
## FreqMin               78   48.00000000    85.1437623   86.1331551 1.104111e+02
## FreqMax_first         78  141.70000000   216.7410256  209.3500000 3.515000e+02
## FreqMax_last          78  144.00000000   215.0538462  209.5000000 3.467000e+02
## FreqMax               78  161.30000000   215.4879562  212.4761905 2.835667e+02
## FreqDelta_first       78   64.35600000   129.8927436  119.4060000 2.586670e+02
## FreqDelta_last        78   77.33300000   130.7780897  124.4580000 2.880000e+02
## FreqDelta             78   90.66700000   130.3464290  127.3036063 1.978548e+02
## Duration_first        78    0.51640000     1.0666333    1.0791500 1.702500e+00
## Duration_last         78    0.57360000     1.0615013    1.0547500 1.937500e+00
## Duration              78    0.62893750     1.0638393    1.0504250 1.568050e+00
## Duration90_first      78    0.38400000     0.7745641    0.7680000 1.472000e+00
## Duration90_last       78    0.38400000     0.7483077    0.7040000 1.280000e+00
## Duration90            78    0.49600000     0.7640415    0.7520000 1.344000e+00
## BoutDur_mins          78    0.50000000    24.6514957   16.0333333 1.258167e+02
## BoutDur_mins_log10    78   -0.30103000     1.1853688    1.2050180 2.099738e+00
##                               sd        range
## Bout*               2.391299e+01 8.100000e+01
## Date                          NA         -Inf
## Year                6.590895e-01 2.000000e+00
## Month               4.319183e+00 1.100000e+01
## DoY                 1.352428e+02 3.640000e+02
## Hour                7.229873e+00 2.300000e+01
## Season*             4.850410e-01 2.000000e+00
## DoYcos              2.335973e-01 7.396143e-01
```

```
## DoYsin             4.584409e-01 1.726227e+00
## Hoursin            7.559556e-01 1.998696e+00
## Hourcos            6.118862e-01 1.999404e+00
## HourDecimal        7.245945e+00 2.308778e+01
## ICI                1.649072e+05 1.236646e+06
## n_calls            1.181110e+01 7.600000e+01
## BoutDur_s          1.455014e+03 7.519000e+03
## Time_NYC                     NA         -Inf
## TimeSinceLVP       2.184091e+00 1.100667e+01
## SPLcalling         5.043215e+00 2.440000e+01
## SPLvessel          4.704077e+00 2.190000e+01
## FreqMin_first      2.182739e+01 1.200000e+02
## FreqMin_last       2.030673e+01 9.330000e+01
## FreqMin            1.363946e+01 6.241111e+01
## FreqMax_first      4.094089e+01 2.098000e+02
## FreqMax_last       3.916066e+01 2.027000e+02
## FreqMax            2.550202e+01 1.222667e+02
## FreqDelta_first    4.151236e+01 1.943110e+02
## FreqDelta_last     3.935943e+01 2.106670e+02
## FreqDelta          2.430327e+01 1.071878e+02
## Duration_first     2.534622e-01 1.186100e+00
## Duration_last      2.618602e-01 1.363900e+00
## Duration           1.731778e-01 9.391125e-01
## Duration90_first   2.137847e-01 1.088000e+00
## Duration90_last    2.029914e-01 8.960000e-01
## Duration90         1.342630e-01 8.480000e-01
## BoutDur_mins       2.425023e+01 1.253167e+02
## BoutDur_mins_log10 4.690455e-01 2.400768e+00
```

```
write.csv(summary_DB, 'dataderived/summary_DB.csv')


# Summarize like Table 3 in Davis et al. (2023)
# https://doi.org/10.1093/icesjms/fsad174
# but by Year and Month and add columns (Bout_dur_mins_sd)

dtmp <- DB %>%
    group_by(Year, Month) %>%
    summarize(n_bouts = n(),
              BoutDur_mins_mean = mean(BoutDur_mins),
              BoutDur_mins_sd = sd(BoutDur_mins),
              BoutDur_mins_95 = quantile(BoutDur_mins, probs = 0.95)
    )

SumTable <- D %>%
    # remove bouts formed by a single upcall
    filter(!is.element(Bout, db1$Bout)) %>%
    group_by(Year, Month) %>%
    summarize(n_calls = n(),
              ICI_95_min = quantile(ICI[-1], probs = 0.95, na.rm = TRUE) / 60
    ) %>%
    left_join(dtmp)

knitr::kable(SumTable, digits = 2)
```

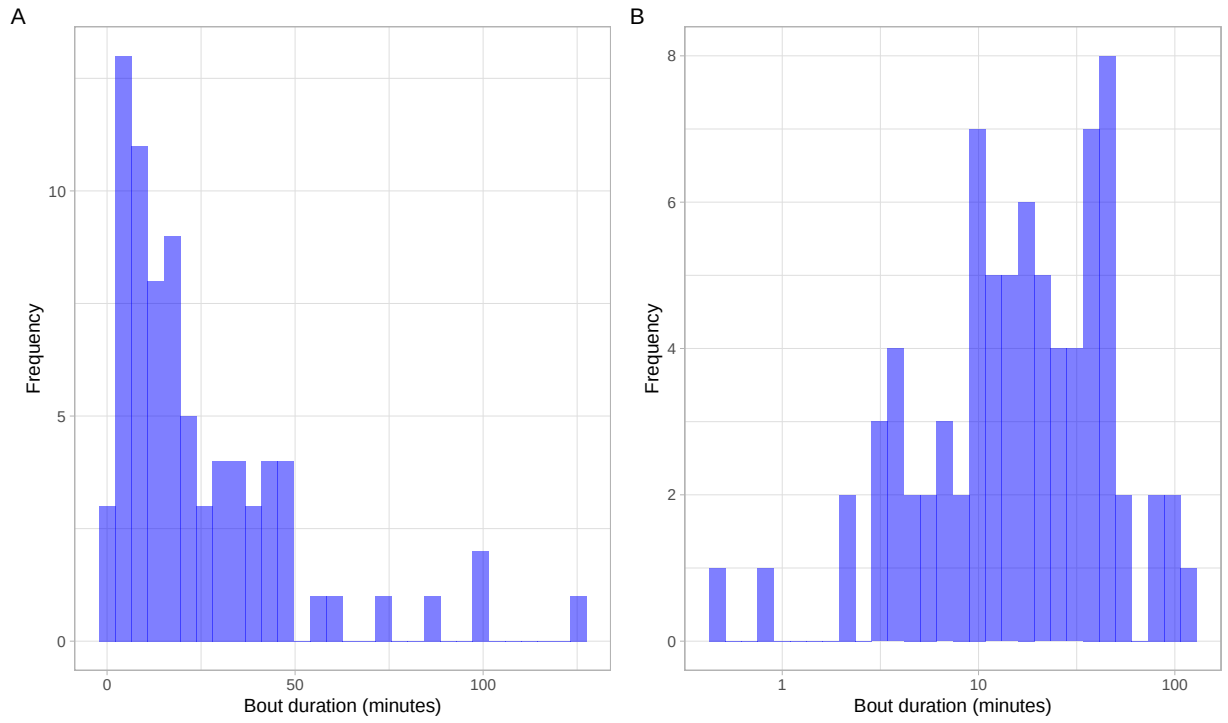
Year	Month	n_calls	ICI_95_min	n_bouts	BoutDur_mins_mea	BoutDur_mins_sd	BoutDur_mins_95
2021	11	18	10.84	2	14.92	1.34	15.77
2021	12	18	10873.03	5	7.73	5.07	13.54
2022	1	117	69.50	8	27.99	17.25	47.55
2022	2	17	275.48	2	16.17	6.88	20.55
2022	3	7	17.39	1	44.07	NA	44.07
2022	11	9	0.84	1	3.67	NA	3.67
2022	12	166	20.45	11	25.80	35.01	82.50
2023	1	67	1193.85	11	15.26	10.93	30.71
2023	2	279	49.56	20	35.56	27.56	88.05
2023	3	187	178.32	17	22.79	23.81	56.72

```
# Alternatively, summary by year

dtmp <- DB %>%
    group_by(Year) %>%
    summarize(n_bouts = n(),
              BoutDur_mins_mean = mean(BoutDur_mins),
              BoutDur_mins_sd = sd(BoutDur_mins),
              BoutDur_mins_95 = quantile(BoutDur_mins, probs = 0.95)
    )

SumTable <- D %>%
    # Remove bouts formed by a single upcall
    filter(!is.element(Bout, db1$Bout)) %>%
    group_by(Year) %>%
    summarize(n_calls = n(),
              ICI_95_min = quantile(ICI[-1], probs = 0.95, na.rm = TRUE) / 60
    ) %>%
    left_join(dtmp)

knitr::kable(SumTable, digits = 2)
```

Year	n_calls	ICI_95_min	n_bouts	BoutDur_mins_mean	BoutDur_mins_sd	BoutDur_mins_95
2021	36	12218.84	7	9.79	5.45	15.30
2022	316	24.34	23	25.56	26.49	48.19
2023	533	331.71	48	26.39	24.39	83.10

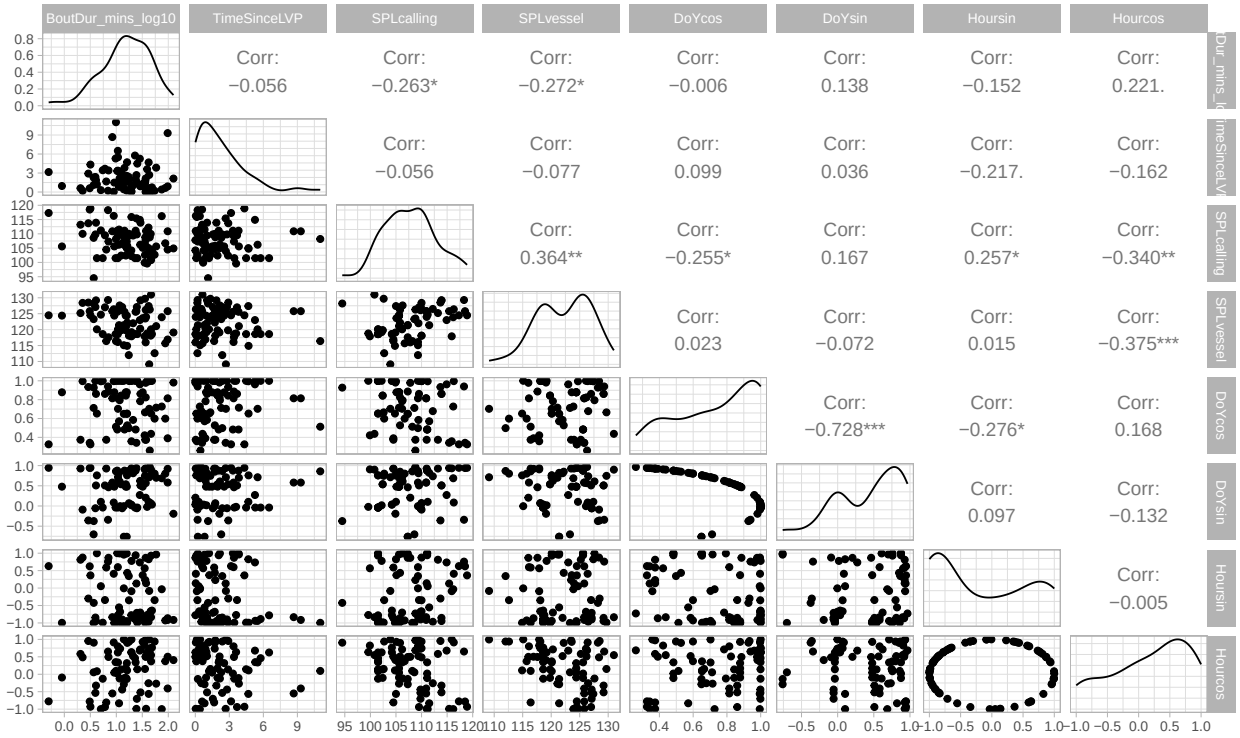
```
## Bout plots ----

# Plot bout duration distributions
p1 <- DB %>%
    ggplot(aes(x = BoutDur_mins)) +
    geom_histogram(fill = "blue", alpha = 0.5, color = NA) +
    labs(x = "Bout duration (minutes)",
         y = "Frequency")
p2 <- DB %>%
    ggplot(aes(x = BoutDur_mins)) +
    geom_histogram(fill = "blue", alpha = 0.5, color = NA) +
    scale_x_log10() +
```

```
    labs(x = "Bout duration (minutes)",
         y = "Frequency")
p1 + p2 +
    plot_annotation(tag_levels = "A")
```




```
ggsave("images/bout_durs.png", width = 8, height = 4, dpi = 600)

# Set the response variables and predictors
RESPONSES <- c("BoutDur_mins_log10")

# PREDICTORS <- c("TimeSinceLVP"
#                 ,"SPLcalling"
#                 ,"SPLvessel"
#                 ,"DoY"
#                 ,"HourDecimal"
# )

PREDICTORS_cycle <- c("TimeSinceLVP"
                      ,"SPLcalling"
                      ,"SPLvessel"
                      ,"DoYcos", "DoYsin"
                      ,"Hoursin", "Hourcos"
)

# Response variables are the first rows/columns in the matrix
DB %>%
    dplyr::select(all_of(c(RESPONSES, PREDICTORS_cycle))) %>%
    GGally::ggpairs() +
    ggplot2::theme(
```

```
        axis.text = ggplot2::element_text(size = 8),
        strip.text = ggplot2::element_text(size = 8)
    )
```




```
ggsave("images/bout_scatter.png", width = 10, height = 8, dpi = 600)

# Show correlations and p-values
DB %>%
    dplyr::select(all_of(c(RESPONSES, PREDICTORS_cycle))) %>%
    as.matrix() %>%
    Hmisc::rcorr()
```

```
##                    BoutDur_mins_log10 TimeSinceLVP SPLcalling SPLvessel DoYcos
## BoutDur_mins_log10               1.00        -0.06      -0.26     -0.27  -0.01
## TimeSinceLVP                    -0.06         1.00      -0.06     -0.08   0.10
## SPLcalling                      -0.26        -0.06       1.00      0.36  -0.25
## SPLvessel                       -0.27        -0.08       0.36      1.00   0.02
## DoYcos                          -0.01         0.10      -0.25      0.02   1.00
## DoYsin                           0.14         0.04       0.17     -0.07  -0.73
## Hoursin                         -0.15        -0.22       0.26      0.01  -0.28
## Hourcos                          0.22        -0.16      -0.34     -0.37   0.17
##                    DoYsin Hoursin Hourcos
## BoutDur_mins_log10   0.14   -0.15    0.22
## TimeSinceLVP         0.04   -0.22   -0.16
## SPLcalling           0.17    0.26   -0.34
## SPLvessel           -0.07    0.01   -0.37
## DoYcos              -0.73   -0.28    0.17
## DoYsin               1.00    0.10   -0.13
```

```
## Hoursin              0.10    1.00    0.00
## Hourcos             -0.13    0.00    1.00
##
## n= 78
##
##
## P
##                    BoutDur_mins_log10 TimeSinceLVP SPLcalling SPLvessel DoYcos
## BoutDur_mins_log10                    0.6275       0.0202     0.0159    0.9554
## TimeSinceLVP       0.6275                          0.6234     0.5053    0.3865
## SPLcalling         0.0202             0.6234                  0.0010    0.0243
## SPLvessel          0.0159             0.5053       0.0010               0.8410
## DoYcos             0.9554             0.3865       0.0243     0.8410
## DoYsin             0.2283             0.7546       0.1444     0.5300    0.0000
## Hoursin            0.1833             0.0567       0.0231     0.8981    0.0144
## Hourcos            0.0513             0.1562       0.0024     0.0007    0.1408
##                    DoYsin Hoursin Hourcos
## BoutDur_mins_log10 0.2283 0.1833  0.0513
## TimeSinceLVP       0.7546 0.0567  0.1562
## SPLcalling         0.1444 0.0231  0.0024
## SPLvessel          0.5300 0.8981  0.0007
## DoYcos             0.0000 0.0144  0.1408
## DoYsin                    0.4001  0.2506
## Hoursin            0.4001         0.9673
## Hourcos            0.2506 0.9673
```

```
# Show 95% confidence intervals for correlations of Response with Predictors
sapply(PREDICTORS_cycle, function(x)
    cor.test(unlist(DB[,RESPONSES]), unlist(DB[,x]), conf.level = 0.95)$conf.int
)
```

```
##      TimeSinceLVP  SPLcalling   SPLvessel     DoYcos      DoYsin     Hoursin
## [1,]   -0.2749164 -0.45835319 -0.46657637 -0.2286371 -0.08722708 -0.36248430
## [2,]    0.1688278 -0.04256446 -0.05300102  0.2164069  0.34977290  0.07276242
##           Hourcos
## [1,] -0.001123739
## [2,]  0.423139891
```

```
## Bout models ----

# Formula relating the response(s) to explanatory variables
fml <- as.formula(paste(RESPONSES,
                        paste(PREDICTORS_cycle, collapse = " + "),
                        sep = " ~ "))

### Linear regression ----

# Full model (response and all predictors)
M_LR_BoutDur <- lm(fml, data = DB)

# Check collinearity of predictors
ols_coll_diag(M_LR_BoutDur)
```

```
## Tolerance and Variance Inflation Factor
```

```
## ----------------------------------------
##      Variables Tolerance      VIF
## 1 TimeSinceLVP 0.8848089 1.130188
## 2   SPLcalling 0.7291019 1.371550
## 3    SPLvessel 0.7527814 1.328407
## 4       DoYcos 0.4072177 2.455689
## 5       DoYsin 0.4471551 2.236360
## 6      Hoursin 0.8297227 1.205222
## 7      Hourcos 0.7506821 1.332122
##
##
## Eigenvalue and Condition Index
## ------------------------------
##     Eigenvalue Condition Index    intercept TimeSinceLVP   SPLcalling
## 1 5.1490712067        1.000000 3.151133e-05 0.0100934571 5.747486e-05
## 2 0.9756688795        2.297276 3.124078e-06 0.0293354676 5.410315e-06
## 3 0.9691655837        2.304971 2.978187e-06 0.0009686071 9.982341e-06
## 4 0.5130534777        3.167985 1.353676e-05 0.0270785056 2.509869e-05
## 5 0.3666543089        3.747452 7.751451e-05 0.8947631598 1.529204e-04
## 6 0.0247485278       14.424135 2.723915e-03 0.0145074904 6.970731e-03
## 7 0.0010734424       69.258818 4.580157e-02 0.0013097327 9.000383e-01
## 8 0.0005645733       95.500200 9.513458e-01 0.0219435798 9.274012e-02
##      SPLvessel       DoYcos       DoYsin     Hoursin    Hourcos
## 1 4.026044e-05 1.213148e-03 4.571267e-03 0.003009569 0.00249031
## 2 2.936197e-06 2.360526e-05 3.098792e-04 0.373380265 0.32234515
## 3 5.081533e-06 3.505813e-04 3.329177e-02 0.327727632 0.30885550
## 4 2.094915e-05 8.423209e-03 3.264121e-01 0.119517280 0.05679069
## 5 1.155918e-04 3.839689e-03 1.408456e-03 0.059858667 0.09246019
## 6 3.974528e-03 9.477951e-01 6.119788e-01 0.076104915 0.02277546
## 7 3.345899e-01 2.521917e-02 6.700147e-05 0.038688592 0.00469296
## 8 6.612507e-01 1.313553e-02 2.196071e-02 0.001713081 0.18958974
```

```r
# Model selected
m_LR_BoutDur <- ols_step_both_p(M_LR_BoutDur, p_remove = 0.05)
summary(m_LR_BoutDur$model)
```

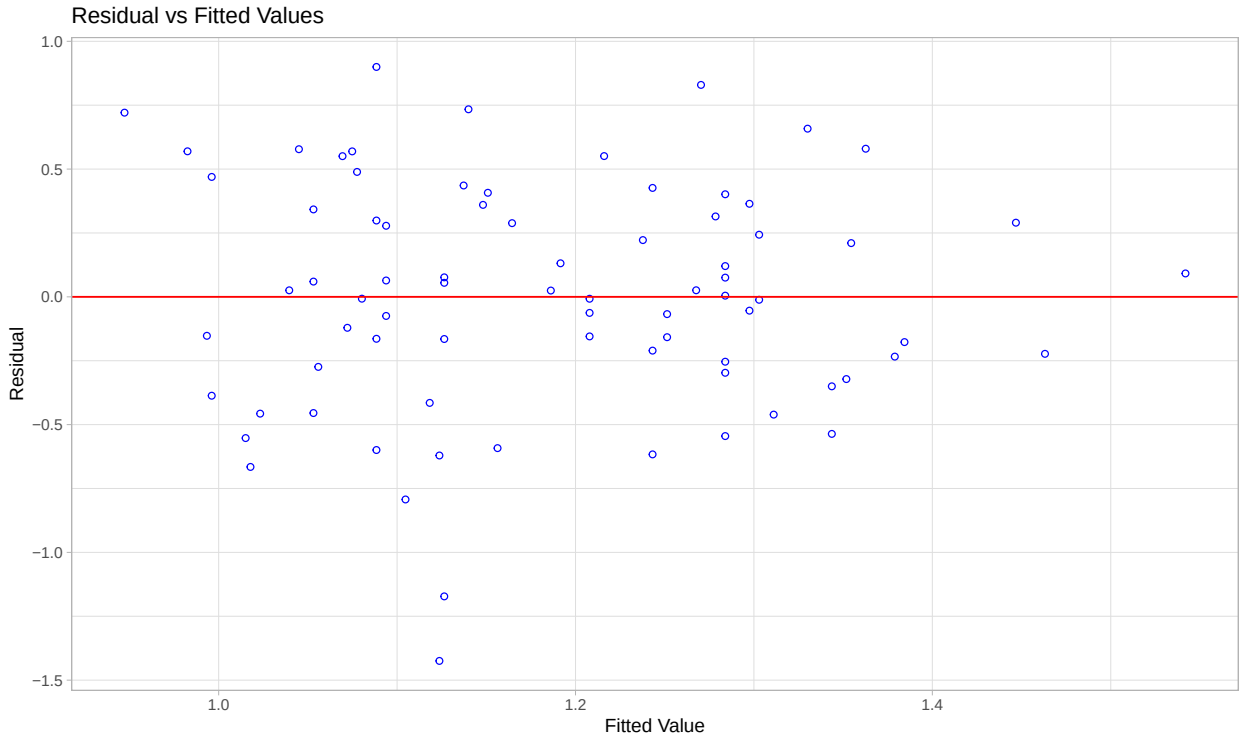
```
##
## Call:
## lm(formula = paste(response, "~", paste(preds, collapse = " + ")),
##     data = l)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.42475 -0.26902  0.01466  0.33531  0.89969
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  4.50428    1.34616   3.346  0.00128 **
## SPLvessel   -0.02715    0.01101  -2.467  0.01586 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4543 on 76 degrees of freedom
## Multiple R-squared:  0.07416,    Adjusted R-squared:  0.06198
```

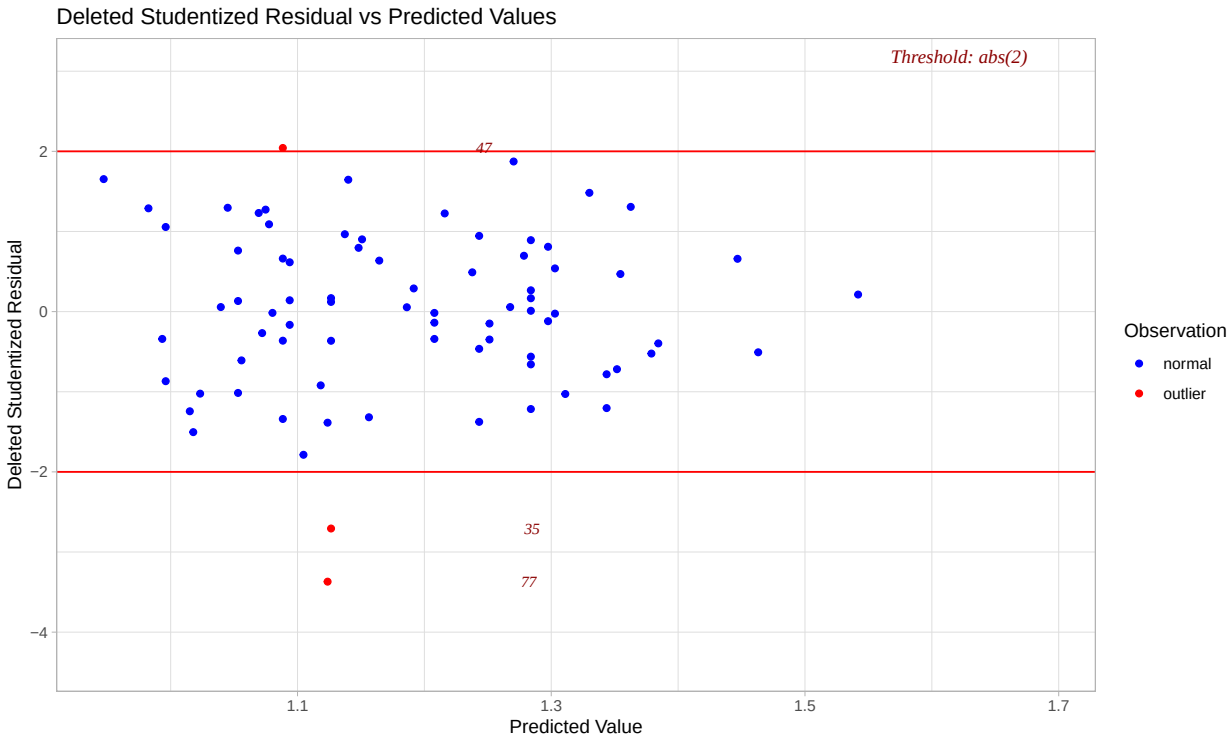
```
## F-statistic: 6.087 on 1 and 76 DF,  p-value: 0.01586
```

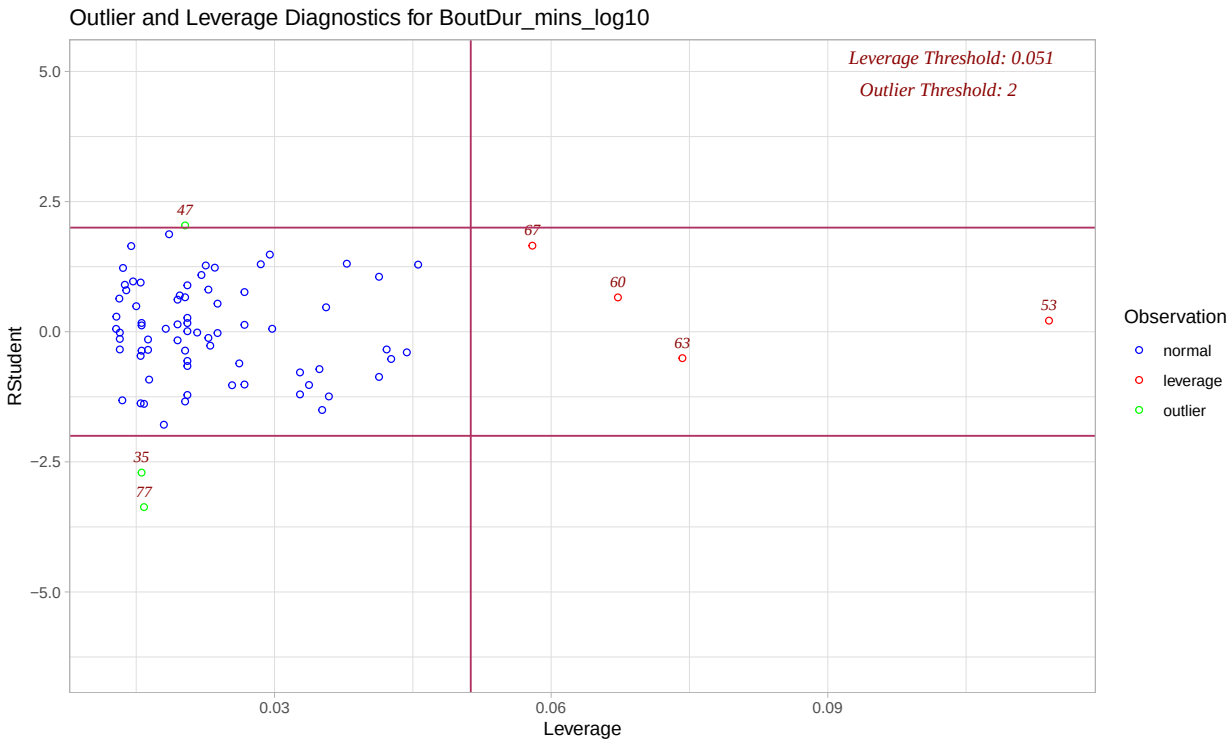
```
# Format the output
tmp <- summary(m_LR_BoutDur$model)
knitr::kable(tmp$coefficients, digits = 3)
```

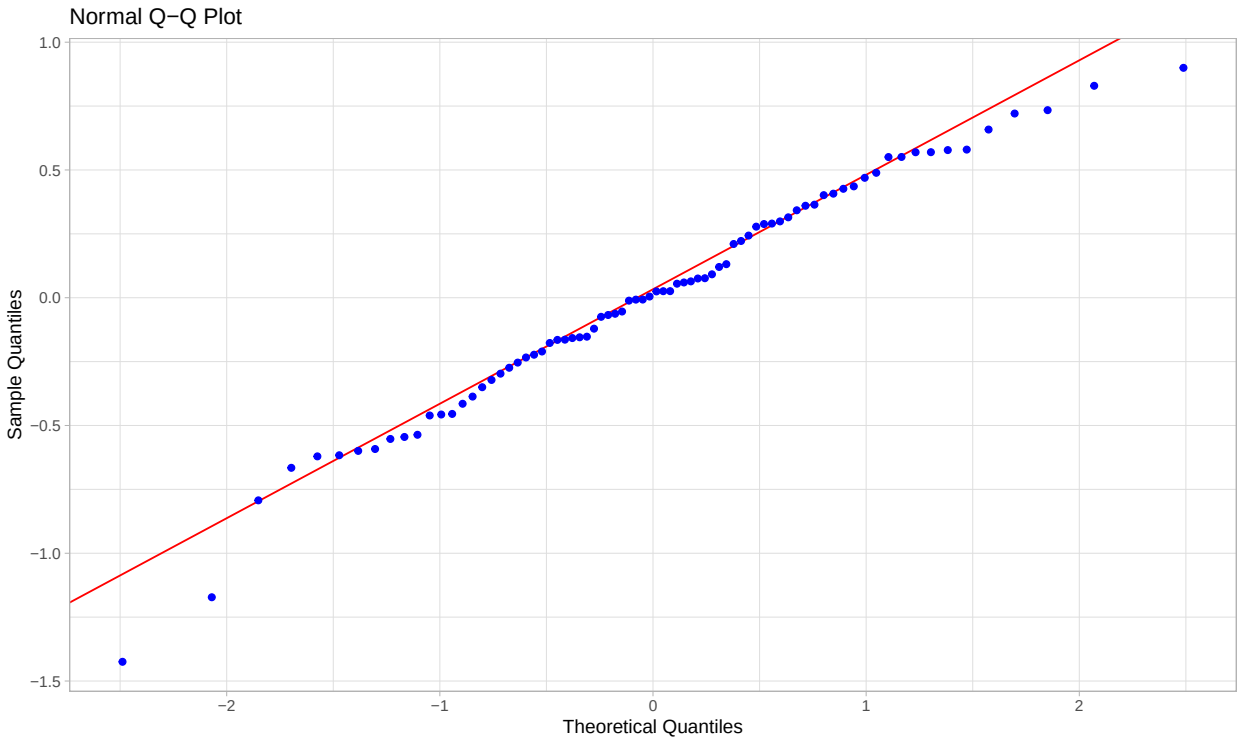
	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	4.504	1.346	3.346	0.001
SPLvessel	-0.027	0.011	-2.467	0.016

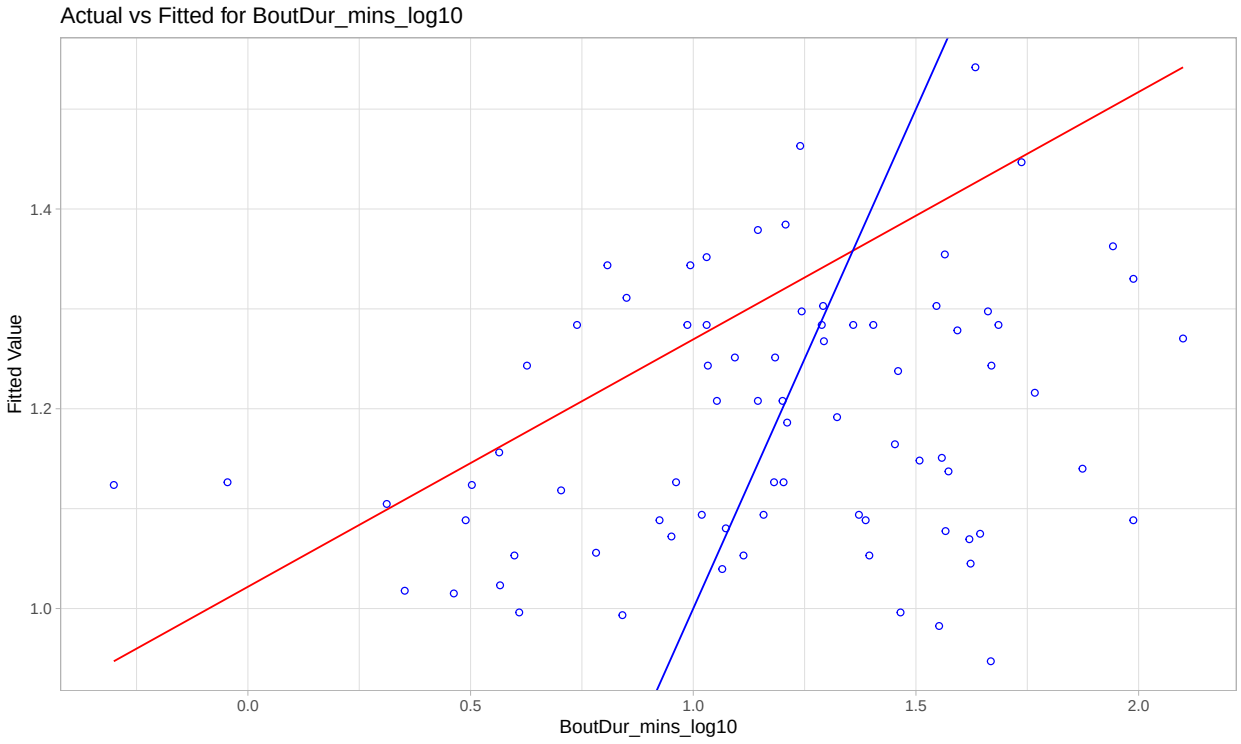
```
# Diagnostics
ols_plot_diagnostics(m_LR_BoutDur$model)
```

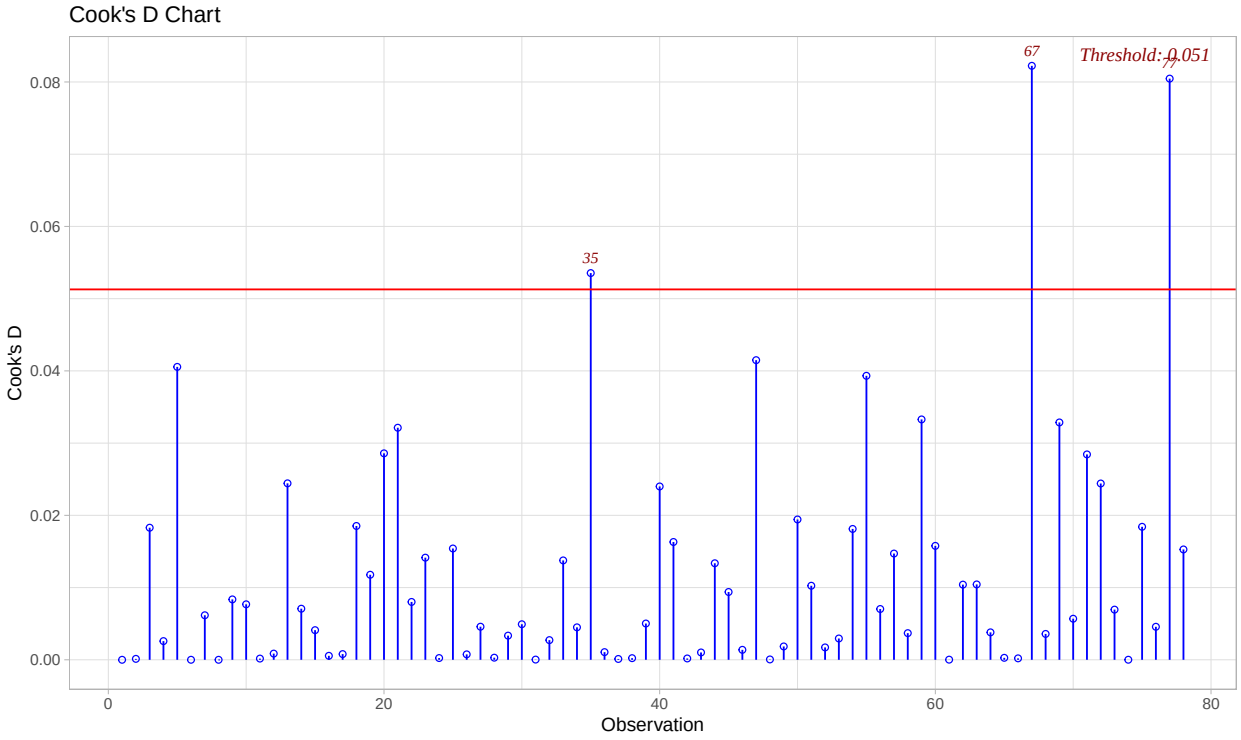


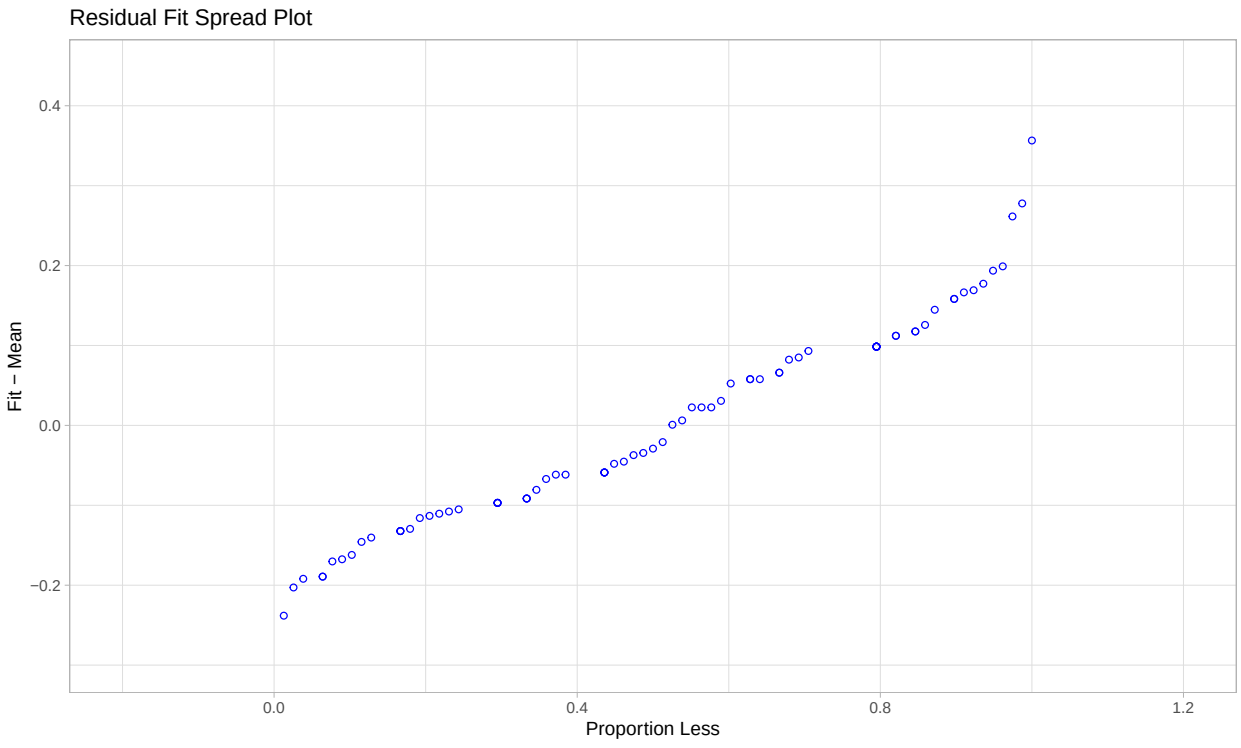

Deleted Studentized Residual vs Predicted Values
Threshold: abs(2)
47
35
77
2
0
−2
−4
Deleted Studentized Residual
1.1
1.3
1.5
1.7
Predicted Value
Observation
normal
outlier

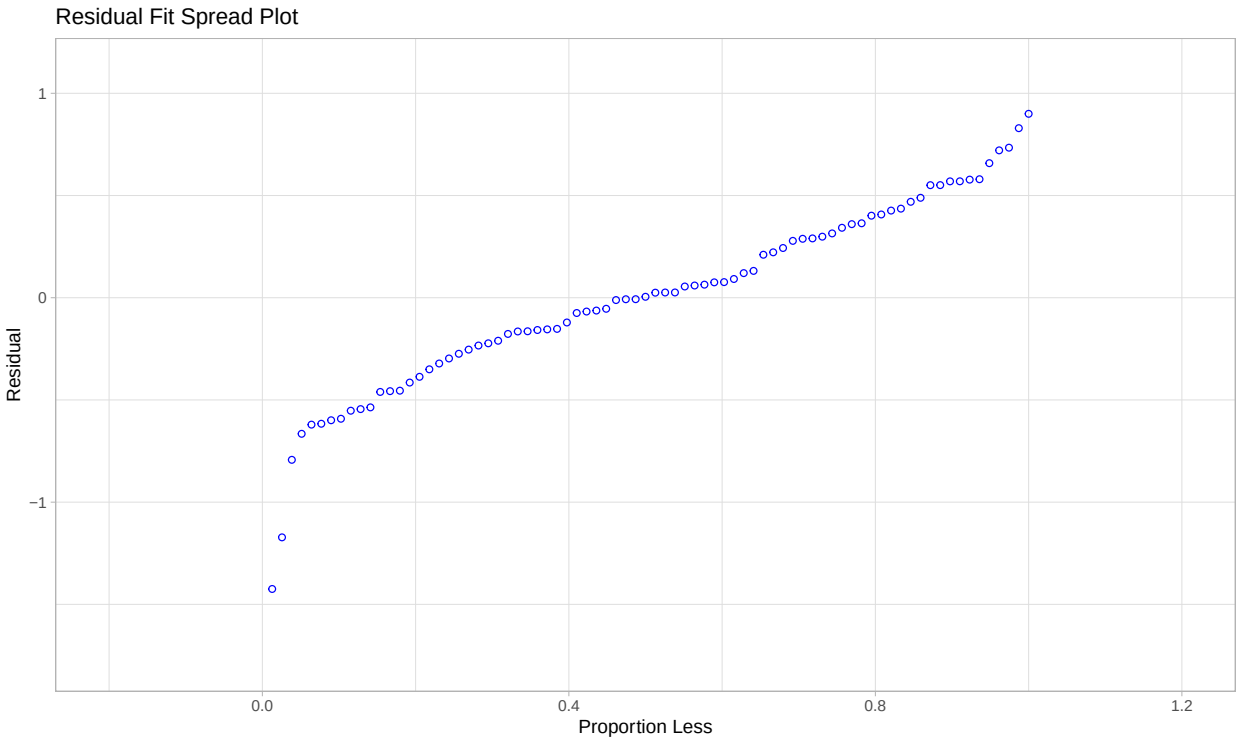

Outlier and Leverage Diagnostics for BoutDur_mins_log10
Leverage Threshold: 0.051
Outlier Threshold: 2
47
67
60
53
63
35
77
5.0
2.5
0.0
−2.5
−5.0
RStudent
0.03
0.06
0.09
Leverage
Observation
normal
leverage
outlier

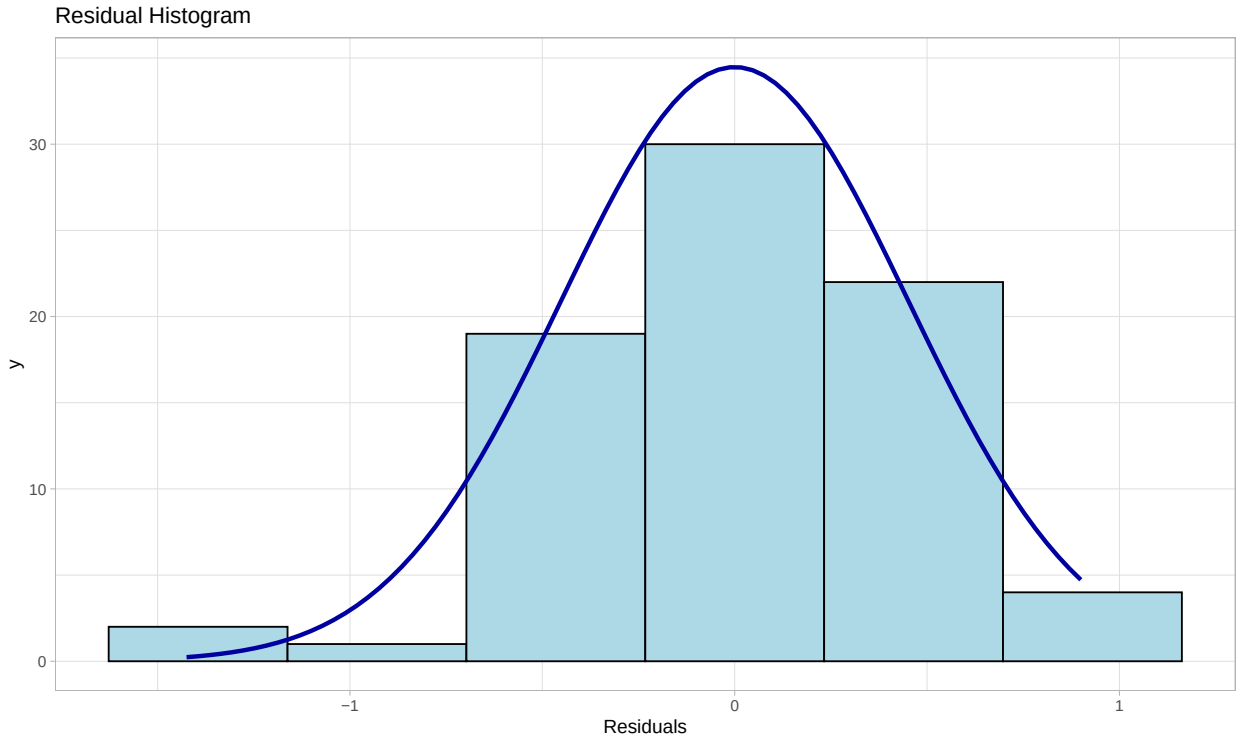

Normal Q−Q Plot
Sample Quantiles
Theoretical Quantiles
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2
−1
0
1
2

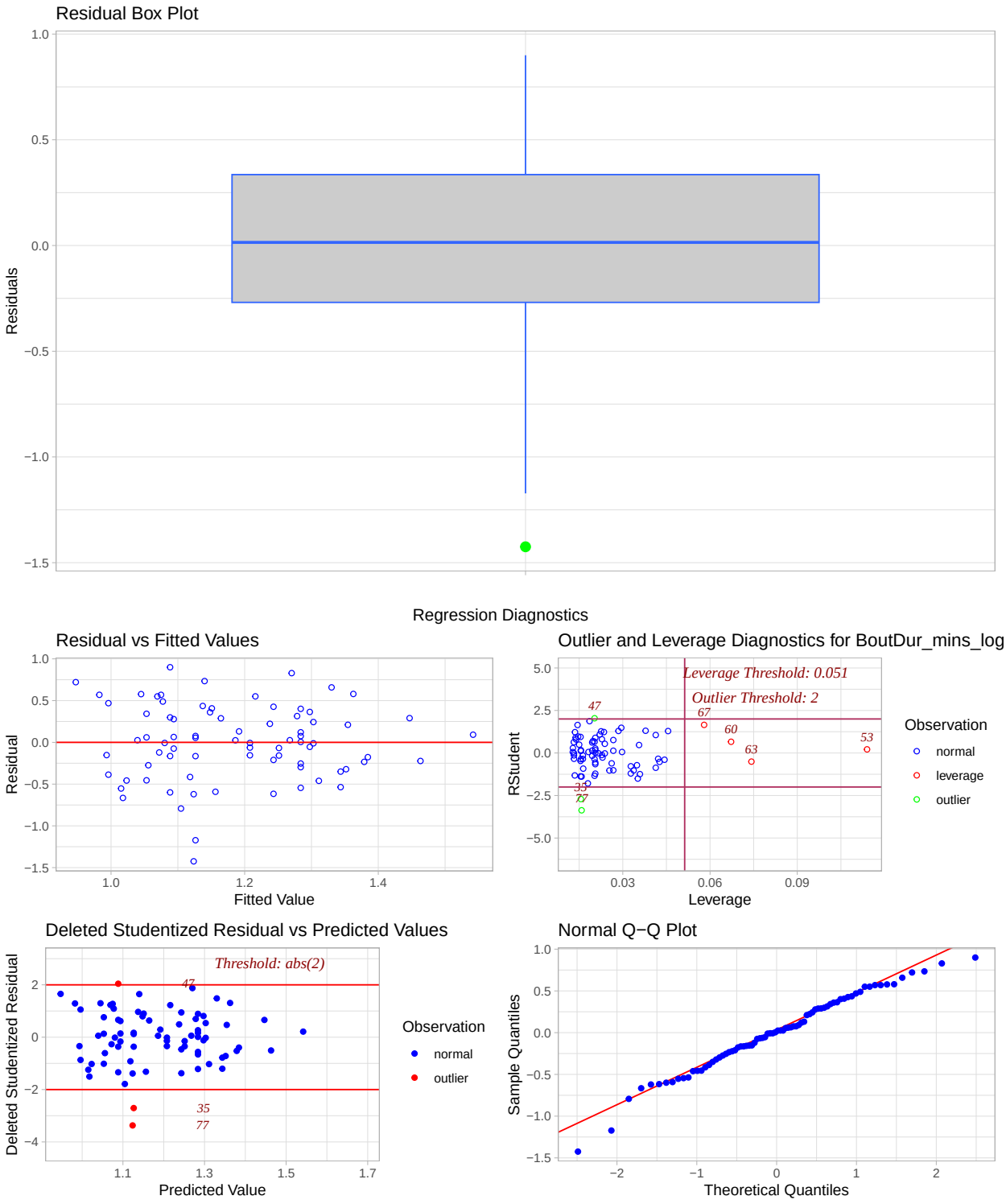

Actual vs Fitted for BoutDur_mins_log10
Fitted Value
BoutDur_mins_log10
1.4
1.2
1.0
0.0
0.5
1.0
1.5
2.0

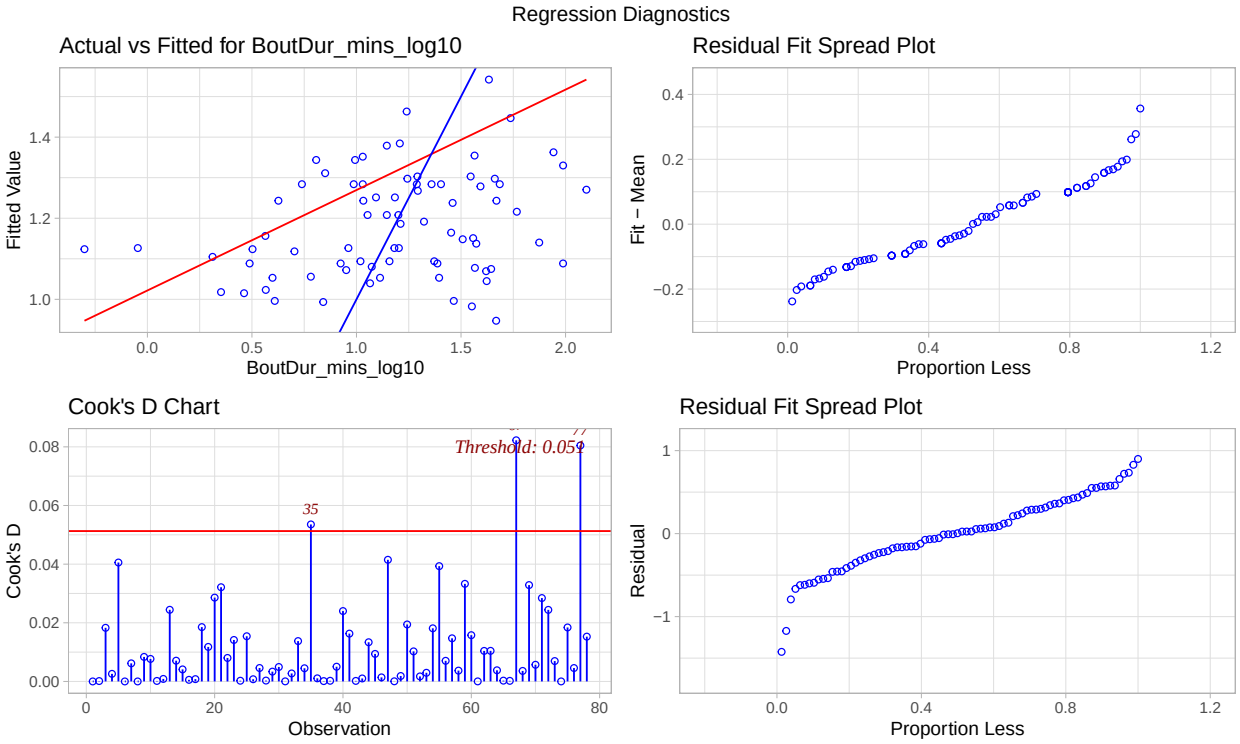

Cook's D Chart
Threshold: 0.051
67
77
35
0.08
0.06
0.04
0.02
0.00
Cook's D
0
20
40
60
80
Observation

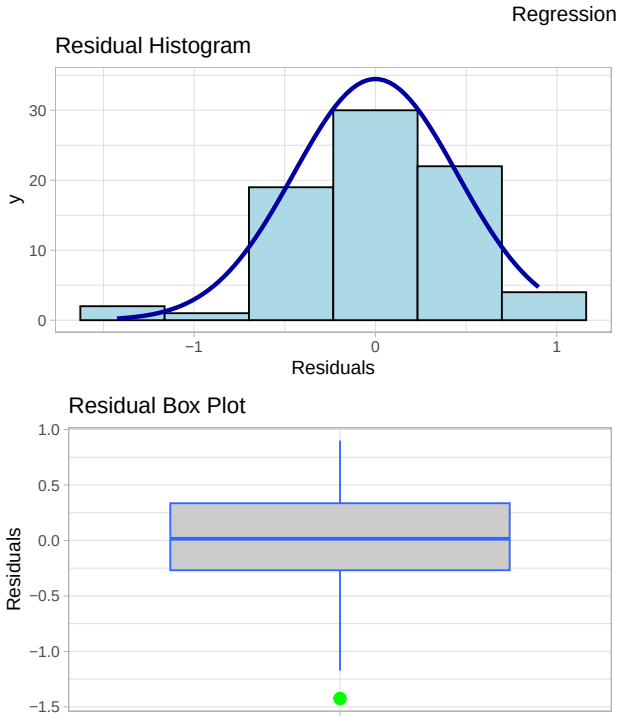

Residual Fit Spread Plot
0.4
0.2
0.0
−0.2
Fit − Mean
0.0
0.4
0.8
1.2
Proportion Less


Residual Fit Spread Plot
Residual
1
0
−1
0.0
0.4
0.8
1.2
Proportion Less

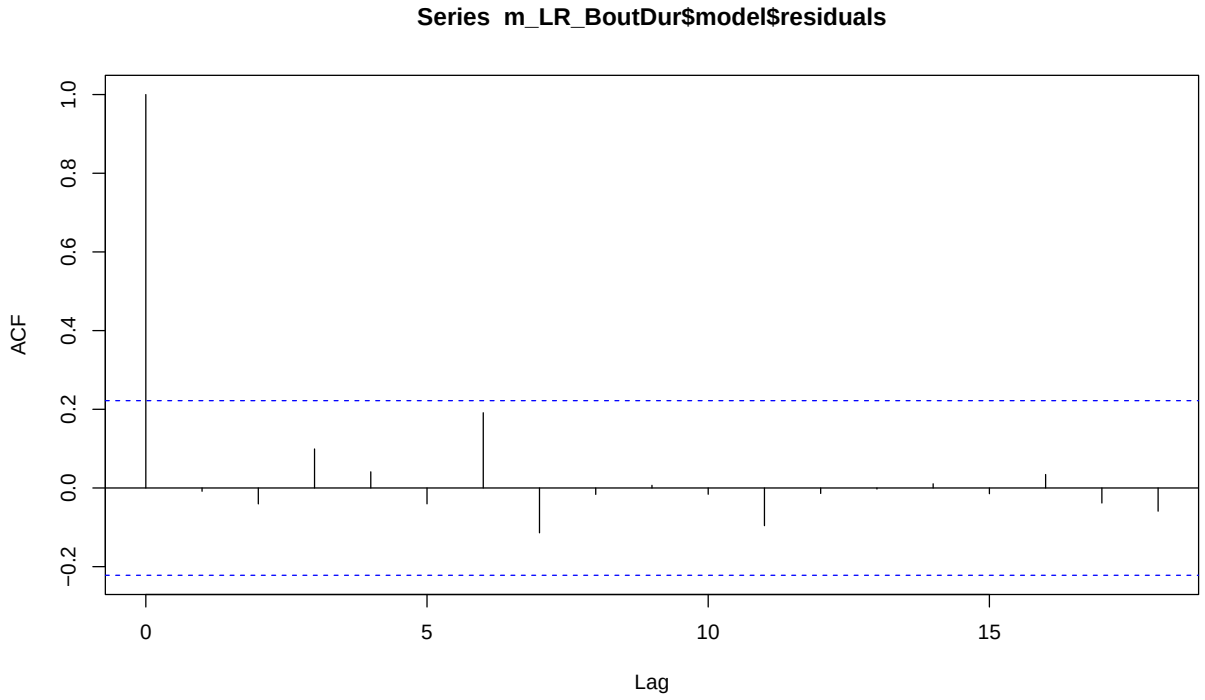

Residual Histogram
y
30
20
10
0
−1
0
1
Residuals


Residual Box Plot
Residuals
1.0
0.5
0.0
−0.5
−1.0
−1.5
Regression Diagnostics
Residual vs Fitted Values
Residual
Fitted Value
1.0
1.2
1.4
Outlier and Leverage Diagnostics for BoutDur_mins_log
RStudent
Leverage
Leverage Threshold: 0.051
Outlier Threshold: 2
47
67
60
63
53
35
77
0.03
0.06
0.09
5.0
2.5
0.0
−2.5
−5.0
Observation
normal
leverage
outlier
Deleted Studentized Residual vs Predicted Values
Deleted Studentized Residual
Predicted Value
Threshold: abs(2)
47
35
77
1.1
1.3
1.5
1.7
2
0
−2
−4
Observation
normal
outlier
Normal Q−Q Plot
Sample Quantiles
Theoretical Quantiles
−2
−1
0
1
2

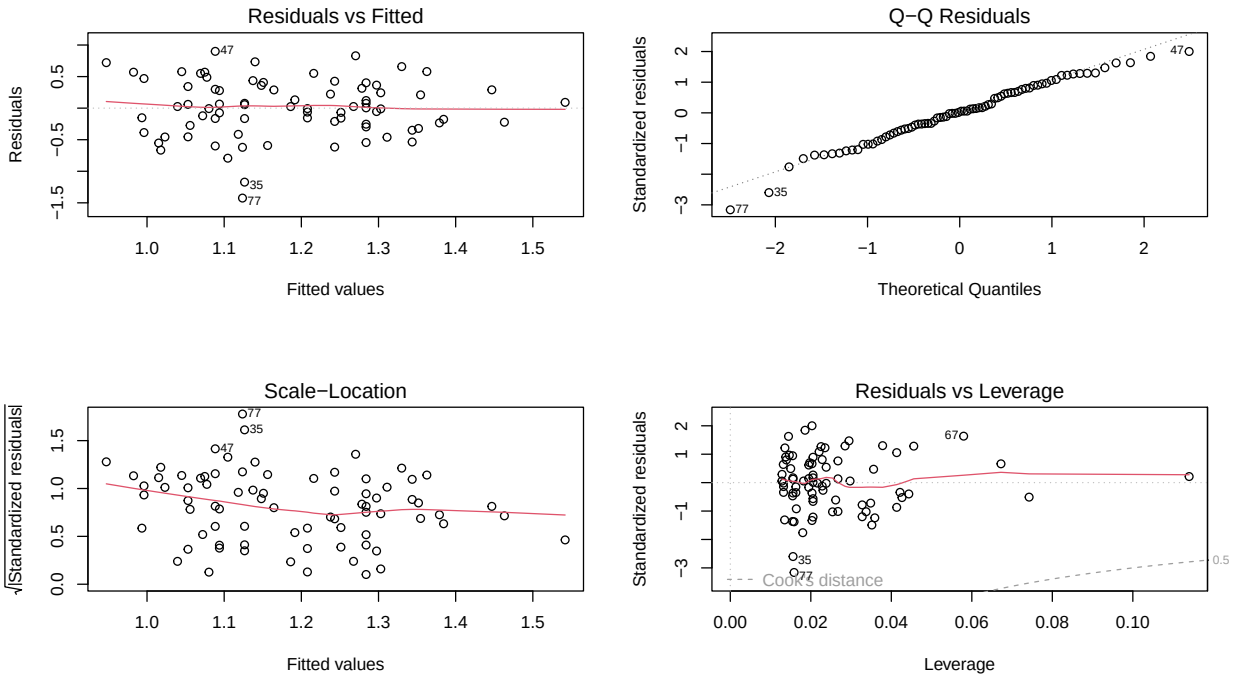





```
acf(m_LR_BoutDur$model$residuals)
```




```
par(mfrow = c(2, 2))
plot(m_LR_BoutDur$model)
```



```r
with(DB,
     cor.test(SPLvessel, SPLcalling)
     )
```

```
##
##  Pearson's product-moment correlation
##
## data:  SPLvessel and SPLcalling
## t = 3.4123, df = 76, p-value = 0.001035
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.1544870 0.5429774
## sample estimates:
##       cor
## 0.3644874
```

```r
### Random forest ----

set.seed(123)
M_RF_BoutDur <- randomForest(fml
                             ,ntree = NTREE, mtry = MTRY, nodesize = NSIZE
                             ,sampsize = ceiling(nrow(DB) * 2/3)
                             ,importance = TRUE
                             ,data = DB)
M_RF_BoutDur
```

```
##
## Call:
##  randomForest(formula = fml, data = DB, ntree = NTREE, mtry = MTRY,      nodesize = NSIZE, sampsize =
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##           Mean of squared residuals: 0.2091423
##                     % Var explained: 3.7
```

```r
plot(M_RF_BoutDur)

importance_values <- M_RF_BoutDur$importance[,1]
importance_df <- tibble(Variable = names(importance_values),
                        Importance = importance_values) %>%
    arrange(Importance)

ggplot(importance_df, aes(x = reorder(Variable, Importance), y = Importance)) +
    geom_bar(stat = "identity") +
    coord_flip() +
    labs(x = "",
         y = "Importance")
ggsave("images/importance_BoutDur.png", width = 4, height = 3, dpi = 600)

# Get PDP data
pdp_BoutDur <- get_pdp_data(M_RF_BoutDur, DB, PREDICTORS_cycle)
```

```r
# Convert the predictor column to factor for faceting
pdp_BoutDur$predictor <- factor(pdp_BoutDur$predictor, levels = PREDICTORS_cycle)

# Plot PDPs with fixed y-axes
ggplot(pdp_BoutDur, aes(x = x, y = yhat)) +
    geom_line() +
    facet_wrap(~predictor, scales = "free_x", nrow = 1) +
    labs(x = "Predictor values",
         y = expression(log[10] * "(Bout duration [minutes])"))
ggsave("images/pdp_BoutDur.png", width = 9, height = 2.5, dpi = 600)



# Occurrence analysis ----

# Save numeric summary of the data as a table
summary_D <- psych::describe(D) %>%
    as.data.frame() %>%
    dplyr::select(n, min, mean, median, max, sd, range)

# Show summary to match with the table in the paper
summary_D
```

```
##                    n           min          mean       median          max
## SelectionNumber  889    1.00000000    17.5208099   13.0000000 7.800000e+01
## Date             889           Inf           NaN           NA         -Inf
## Year             889 2021.00000000  2022.5601800 2023.0000000 2.023000e+03
## Month            889    1.00000000     4.3768279    2.0000000 1.200000e+01
## Time_NYC         889           Inf           NaN           NA         -Inf
## TimeLastVP_NYC   889           Inf           NaN           NA         -Inf
## TimeSinceLVP     889    0.01361111     2.4936630    1.9105556 1.118444e+01
## Hour             889    0.00000000    14.4836895   18.0000000 2.300000e+01
## SPLcalling       889   94.50000000   106.9066367  105.8000000 1.189000e+02
## SPLvessel        889  109.10000000   121.0942632  120.1000000 1.310000e+02
## FreqMin          889   26.70000000    86.4203600   88.0000000 1.467000e+02
## FreqMax          889  131.70000000   223.5869516  218.7000000 4.057000e+02
## FreqDelta        889   48.33300000   137.1678009  130.6670000 3.078290e+02
## Duration         889    0.43150000     1.0775864    1.0723000 2.200600e+00
## Duration90       889    0.25600000     0.7502902    0.7040000 1.664000e+00
## IsVPYesterday    889           Inf           NaN           NA         -Inf
## HourDecimal      889    0.03500000    15.0115192   18.4252778 2.398361e+01
## DoY              889    1.00000000   117.1754781   54.0000000 3.650000e+02
## DoYsin           889   -0.76072009     0.4306122    0.5804549 9.655072e-01
## DoYcos           889    0.26037648     0.7472800    0.7938440 9.999908e-01
## Hoursin          889   -0.99999848    -0.2872195   -0.6519330 9.999982e-01
## Hourcos          889   -0.99999712     0.2715805    0.4771588 9.999908e-01
## Season*          889    1.00000000     2.1878515    2.0000000 3.000000e+00
## ICI              887    1.00000000 23692.2829763   86.0000000 3.770917e+06
## YearDeployment*  889    1.00000000     1.7997750    2.0000000 2.000000e+00
## Bout             889    1.00000000    42.9538808   45.0000000 8.200000e+01
##                            sd        range
## SelectionNumber 1.606728e+01 7.700000e+01
## Date                      NA         -Inf
```

```
## Year            5.744468e-01 2.000000e+00
## Month           4.263659e+00 1.100000e+01
## Time_NYC                  NA         -Inf
## TimeLastVP_NYC            NA         -Inf
## TimeSinceLVP    2.243010e+00 1.117083e+01
## Hour            7.088089e+00 2.300000e+01
## SPLcalling      4.387383e+00 2.440000e+01
## SPLvessel       4.550777e+00 2.190000e+01
## FreqMin         1.962163e+01 1.200000e+02
## FreqMax         3.869449e+01 2.740000e+02
## FreqDelta       3.950972e+01 2.594960e+02
## Duration        2.513158e-01 1.769100e+00
## Duration90      1.999083e-01 1.408000e+00
## IsVPYesterday             NA         -Inf
## HourDecimal     7.083073e+00 2.394861e+01
## DoY             1.340355e+02 3.640000e+02
## DoYsin          4.520861e-01 1.726227e+00
## DoYcos          2.281498e-01 7.396143e-01
## Hoursin         7.079888e-01 1.999997e+00
## Hourcos         5.860469e-01 1.999988e+00
## Season*         4.621103e-01 2.000000e+00
## ICI             1.966299e+05 3.770916e+06
## YearDeployment* 4.003939e-01 1.000000e+00
## Bout            2.305026e+01 8.100000e+01
```

```
write.csv(summary_D, 'dataderived/summary_D.csv')

## Year-season counts ----

# Count number of upcalls from the year of deployment
table(D$YearDeployment)
```

```
##
## Year 1 Year 2
##    178    711
```

```
# Counts per season
table(D$YearDeployment, D$Season)
```

```
##
##          Fall Winter Spring
##   Year 1   18    153      7
##   Year 2    9    515    187
```

```
# % per season
round(table(D$YearDeployment, D$Season) * 100 / as.vector(table(D$YearDeployment)), 0)
```

```
##
##          Fall Winter Spring
##   Year 1   10     86      4
##   Year 2    1     72     26
```

```r
## Seasonal plots ----

# See
# https://stackoverflow.com/questions/15575713/modifying-timezone-of-a-posixct-object-without-changing-
# for help changing the time zone without changing the displayed time
DateMonitoring <- read_xlsx("data/Datetime_Local_NARWMonitoring.xlsx") %>%
    arrange(datetime_local) %>%
    # edit timezone
    mutate(datetime_local = as.POSIXct(as.character(datetime_local),
                                       origin = as.POSIXct("1970-01-01"),
                                       tz = "America/New_York")) %>%
    # round to hours
    mutate(datetime_local = lubridate::floor_date(datetime_local, unit = "hour")) %>%
    # keep only unique
    dplyr::distinct(datetime_local, .keep_all = TRUE)

attr(DateMonitoring$datetime_local, "tzone") <- "UTC"

DateMonitoring_day <- DateMonitoring %>%
    # subtract 5 hours to convert to "UTC - 5"
    mutate(datetime_local = datetime_local - 5*3600) %>%
    mutate(datetime_local = as.Date(datetime_local)) %>%
    dplyr::distinct(datetime_local, .keep_all = TRUE)

AllDays <- tibble(date = seq(from = as.Date(min(DateMonitoring$datetime_local)),
                             to = as.Date(max(DateMonitoring$datetime_local)),
                             by = "1 day"),
                  lat = 38.303,
                  lon = -74.645)

sun <- getSunlightTimes(data = AllDays,
                        tz = "UTC", #"America/New_York",
                        keep = c("sunrise", "sunset")) %>%
    # subtract 5 hours to convert to "UTC - 5"
    mutate(sunrise = sunrise - 5*3600,
           sunset = sunset - 5*3600) %>%
    mutate(sunrise_h = get_decimal_hour(sunrise),
           sunset_h = get_decimal_hour(sunset)) %>%
    arrange(date)

# Create a copy of the main data but switch times to UTC - 5
D_utc <- D
attr(D_utc$Time_NYC, "tzone") <- "UTC"
D_utc <- D_utc %>%
    mutate(Time_NYC = Time_NYC - 5*3600) %>%
    mutate(HourDecimal = get_decimal_hour(Time_NYC),
           Date = as.Date(Time_NYC),
           Hour = as.numeric(format(Time_NYC, "%H")))

# Year 1
DateMonitoring_day_0 <- DateMonitoring_day %>%
    dplyr::filter(datetime_local < BuoyReplacement) %>%
    dplyr::filter(datetime_local < as.Date("2022-04-01")) %>%
```

```
    dplyr::filter(datetime_local > as.Date("2021-10-20"))
sun_0 <- sun %>%
    dplyr::filter(date < BuoyReplacement) %>%
    dplyr::filter(date < as.Date("2022-04-01")) %>%
    dplyr::filter(date > as.Date("2021-10-20"))

d_0 <- D_utc %>%
    dplyr::filter(Date < BuoyReplacement)

p1 <- ggplot() +
    geom_rect(data = DateMonitoring_day_0,
              aes(ymin = datetime_local, ymax = datetime_local + days(1),
                  xmin = -Inf, xmax = Inf),
              fill = "gray", alpha = 0.5) +
    geom_point(data = sun_0, pch = 16,
               aes(x = sunrise_h, y = date), color = "orange", size = 1) +
    geom_point(data = sun_0, pch = 16,
               aes(x = sunset_h, y = date), color = "orange", size = 1) +
    geom_point(data = d_0,
               aes(x = HourDecimal, y = Date),
               color = "black", size = 1, pch = 15) +
    labs(x = "Hour of day (UTC - 5)",
         y = "") +
    scale_x_continuous(limits = c(0, 24),
                       breaks = seq(0, 24, by = 6),
                       minor_breaks = seq(0, 24, by = 2),
                       expand = c(0.01, 0.01)) +
    scale_y_date(breaks = as.Date(c("2021-11-01", "2022-01-01",
                                    "2022-04-01",
                                    "2023-01-01", "2023-04-01")),
                 date_labels = "%Y %b",
                 # date_minor_breaks = "1 month"
                 minor_breaks = seq(from = as.Date("2021-11-01"),
                                    to = as.Date("2023-04-01"),
                                    by = "1 month"),
                 expand = c(0.01, 0.01)
    )

# Year 2
DateMonitoring_day_0 <- DateMonitoring_day %>%
    dplyr::filter(datetime_local > BuoyReplacement) %>%
    dplyr::filter(datetime_local > as.Date("2022-10-20"))
sun_0 <- sun %>%
    dplyr::filter(date > BuoyReplacement) %>%
    dplyr::filter(date > as.Date("2022-10-20"))
d_0 <- D_utc %>%
    dplyr::filter(Date > BuoyReplacement)

p2 <- ggplot() +
    geom_rect(data = DateMonitoring_day_0,
              aes(ymin = datetime_local, ymax = datetime_local + days(1),
                  xmin = -Inf, xmax = Inf),
              fill = "gray", alpha = 0.5) +
```

```
    geom_point(data = sun_0, pch = 16,
               aes(x = sunrise_h, y = date), color = "orange", size = 1) +
    geom_point(data = sun_0, pch = 16,
               aes(x = sunset_h, y = date), color = "orange", size = 1) +
    geom_point(data = d_0,
               aes(x = HourDecimal, y = Date),
               color = "black", size = 1, pch = 15) +
    labs(x = "Hour of day (UTC - 5)",
         y = "") +
    scale_x_continuous(limits = c(0, 24),
                       breaks = seq(0, 24, by = 6),
                       minor_breaks = seq(0, 24, by = 2),
                       expand = c(0.01, 0.01)) +
    scale_y_date(breaks = as.Date(c("2021-11-01", "2022-01-01",
                                    "2022-11-01",
                                    "2023-01-01", "2023-04-01")),
                 date_labels = "%Y %b",
                 # date_minor_breaks = "1 month"
                 minor_breaks = seq(from = as.Date("2021-11-01"),
                                    to = as.Date("2023-04-01"),
                                    by = "1 month"),
                 expand = c(0.01, 0.01)
    )

p1 + p2 +
    plot_annotation(tag_levels = "A")

ggsave("images/occurrence_diel.png", width = 8, height = 4, dpi = 600)


## Diel analysis ----

# Count calls per date and hour, then combine with all monitoring days
D_dh <- D_utc %>%
    dplyr::group_by(Date, Hour) %>%
    dplyr::summarise(Ncalls = n())
# Check
sum(D_dh$Ncalls) == nrow(D)
```

```
## [1] TRUE
```

```
Ncalls0 <- expand.grid(Date = DateMonitoring_day$datetime_local,
                       Hour = 0:23) %>%
    as_tibble() %>%
    left_join(D_dh, by = c("Date", "Hour")) %>%
    mutate(HourFac = factor(Hour, levels = 0:23),
           Year = "year 1",
           HourSin = sin(2 * pi * Hour / 24),
           HourCos = cos(2 * pi * Hour / 24)) %>%
    arrange(Date, Hour)
# Those counts of calls that are missing during the monitoring are 0
Ncalls0$Ncalls[is.na(Ncalls0$Ncalls)] <- 0
# Check
```

```r
sum(Ncalls0$Ncalls) == nrow(D)
```

```
## [1] TRUE
```

```r
# Plot number of calls per hour
# Aggregate calls by hour
Ncalls0_agg <- Ncalls0 %>%
    group_by(Hour) %>%
    summarise(Ncalls = sum(Ncalls))
# Check
sum(Ncalls0_agg$Ncalls) == nrow(D)
```

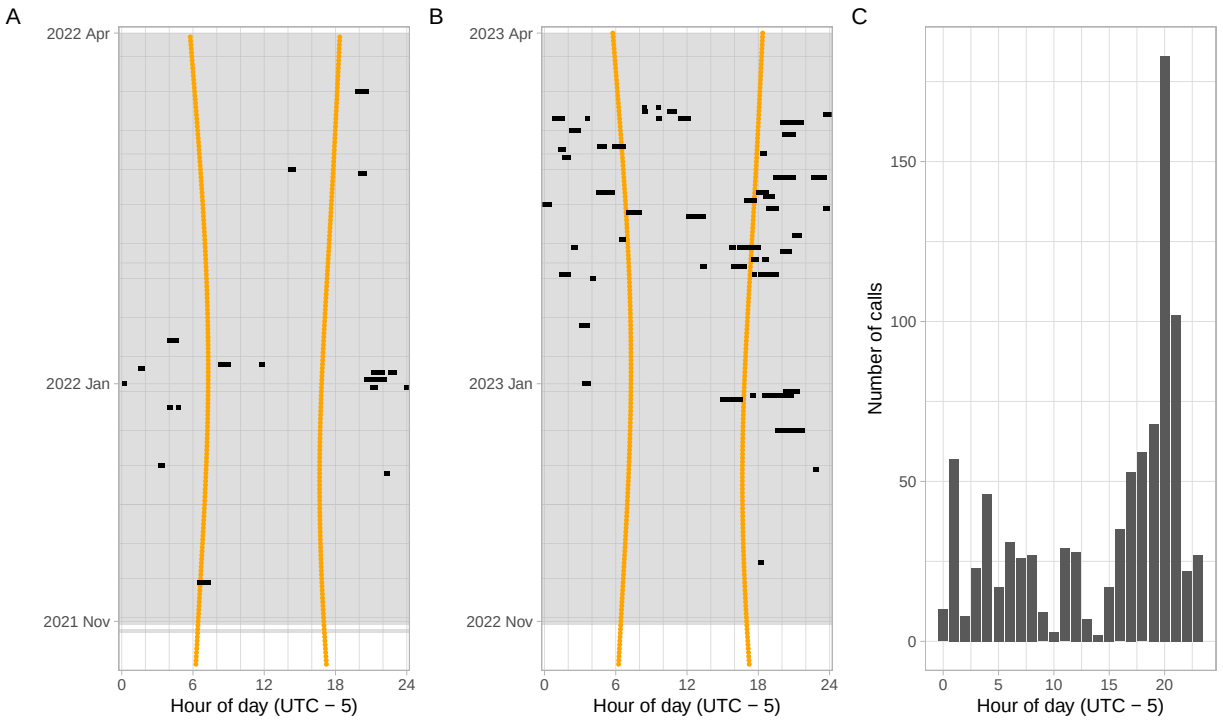
```
## [1] TRUE
```

```r
p3 <- Ncalls0_agg %>%
    ggplot(aes(x = Hour, y = Ncalls)) +
    geom_bar(stat = "identity") +
    labs(x = "Hour of day (UTC - 5)", y = "Number of calls")

p1 + p2 + p3 +
    plot_annotation(tag_levels = "A")

ggsave("images/occurrence_dielNcalls.png", width = 8, height = 3, dpi = 600)


# Use a model to test the diel pattern
# Model 1: based on categorical hours -- has higher AIC and BIC than model 2 below.
# m_ncalls_gls <- nlme::gls(log(Ncalls + 1) ~ HourFac
#           ,correlation = nlme::corAR1()
#           ,data = Ncalls0)

# Model 2: based on sin+cos transformation of hours
m_ncalls_gls <- nlme::gls(log(Ncalls + 1) ~ HourSin + HourCos
                          ,correlation = nlme::corAR1()
                          ,method = "REML"
                          ,data = Ncalls0)

summary(m_ncalls_gls)
```

```
## Generalized least squares fit by REML
##   Model: log(Ncalls + 1) ~ HourSin + HourCos
##   Data: Ncalls0
##         AIC       BIC   logLik
##   -588.4278 -553.9216 299.2139
##
## Correlation Structure: AR(1)
##  Formula: ~1
##  Parameter estimate(s):
##       Phi
## 0.3670744
##
## Coefficients:
```

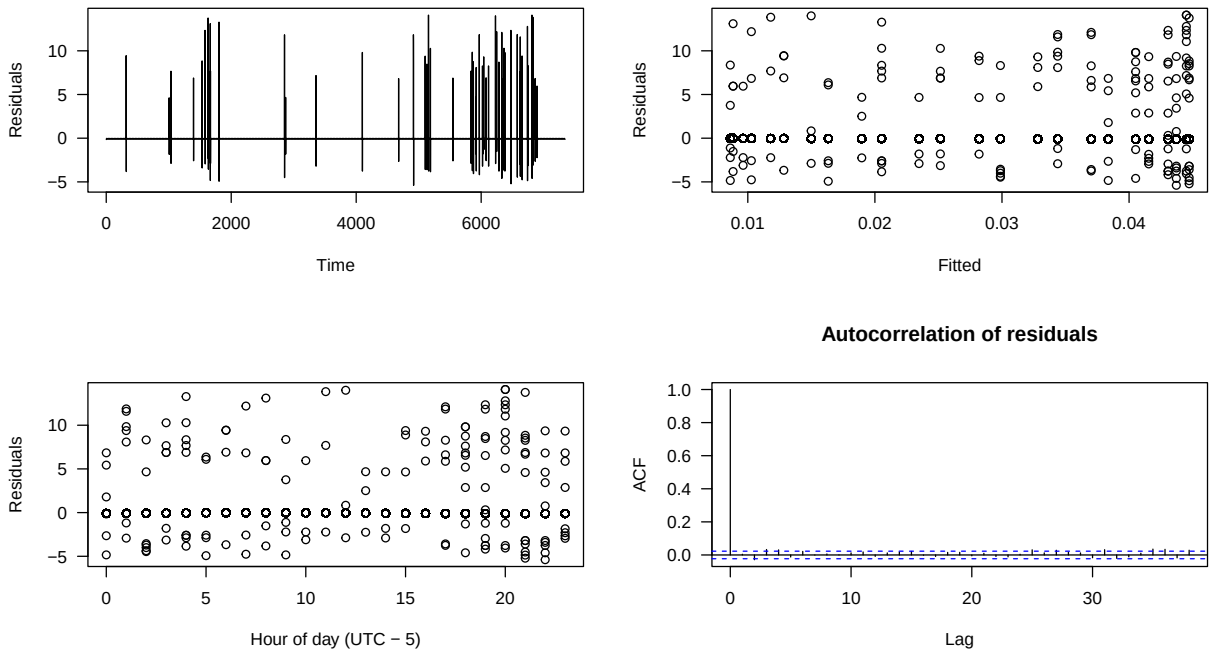
```
##                   Value   Std.Error   t-value p-value
## (Intercept)  0.02667577 0.004275948  6.238562  0.0000
## HourSin     -0.01384155 0.005867181 -2.359148  0.0183
## HourCos      0.01168837 0.005866322  1.992453  0.0464
##
##  Correlation:
##         (Intr) HourSn
## HourSin 0
## HourCos 0      0
##
## Standardized residuals:
##         Min          Q1         Med          Q3         Max
## -0.17937737 -0.15385521 -0.10087434 -0.05147023 14.30270264
##
## Residual standard error: 0.2493522
## Degrees of freedom: 7344 total; 7341 residual
```

```r
# ?nlme:::residuals.gls
# e <- residuals(m_ncalls_gls, type = "pearson")
e <- residuals(m_ncalls_gls, type = "normalized")

# Residual diagnostics plots
par(mfrow = c(2, 2))
```




```r
plot.ts(e,
        las = 1,
        ylab = "Residuals")
plot(x = fitted(m_ncalls_gls), y = e,
     las = 1,
```

```
     ylab = "Residuals", xlab = "Fitted")
plot(x = Ncalls0$Hour, y = e,
     las = 1,
     ylab = "Residuals", xlab = "Hour of day (UTC - 5)")
acf(e,
    las = 1,
    main = "Autocorrelation of residuals")
```




```
# Test significance of diel patterns, using ML since the fixed effects are different
m_ncalls_gls <- nlme::gls(log(Ncalls + 1) ~ HourSin + HourCos
                          ,correlation = nlme::corAR1()
                          ,method = "ML"
                          ,data = Ncalls0)

# Reduced model (without predictors)
m_ncalls_gls_reduced <- gls(log(Ncalls + 1) ~ 1
                            ,correlation = corAR1()
                            ,method = "ML"
                            ,data = Ncalls0)

# Likelihood ratio test
# Use the p-value to determine if the predictors significantly improve the model.
anova(m_ncalls_gls, m_ncalls_gls_reduced)
```

```
##                       Model df       AIC       BIC   logLik   Test  L.Ratio
## m_ncalls_gls              1  5 -614.3803 -579.8721 312.1902
## m_ncalls_gls_reduced      2  3 -608.8541 -588.1492 307.4271 1 vs 2 9.526203
##                       p-value
```

```
## m_ncalls_gls
## m_ncalls_gls_reduced  0.0085
```

```
# 95% confidence interval for chi-squared distribution under the null hypothesis
df <- 2 # df.full - df.reduced
c(qchisq(0.025, df), qchisq(0.975, df))
```

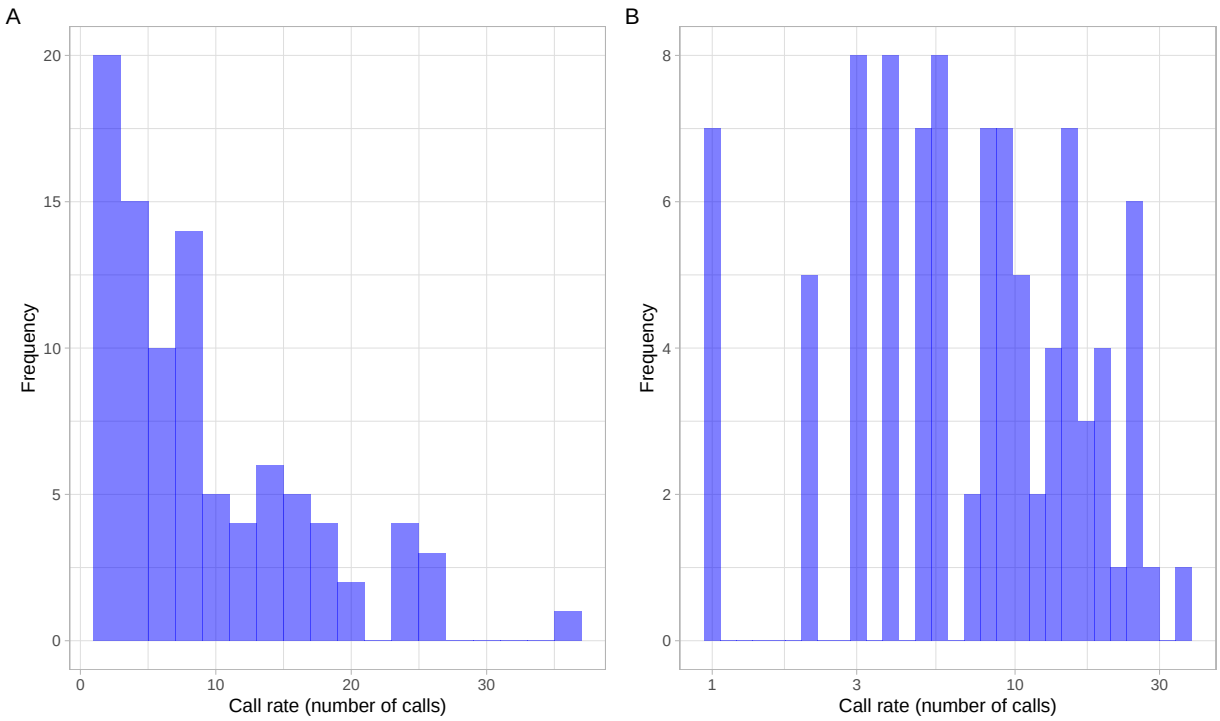
```
## [1] 0.05063562 7.37775891
```

```
# Call rates ----

## Calculate call rates ----

# Aggregate to calculate call rates
DC <- D %>%
    group_by(Date, Month, DoY, Hour, Season) %>%
    summarise(CallRate = n(),
              HourDecimal = mean(HourDecimal, na.rm = TRUE),
              TimeSinceLVP = mean(TimeSinceLVP),
              SPLcalling = mean(SPLcalling),
              SPLvessel = mean(SPLvessel),
              DoYcos = mean(DoYcos),
              DoYsin = mean(DoYsin),
              Hoursin = mean(Hoursin),
              Hourcos = mean(Hourcos)) %>%
    ungroup() %>%
    mutate(CallRate_log10 = log10(CallRate))

# Save numeric summary of the data as a table
summary_DC <- psych::describe(DC) %>%
    as.data.frame() %>%
    dplyr::select(n, min, mean, median, max, sd, range)

# Show summary to match with the table in the paper
summary_DC
```

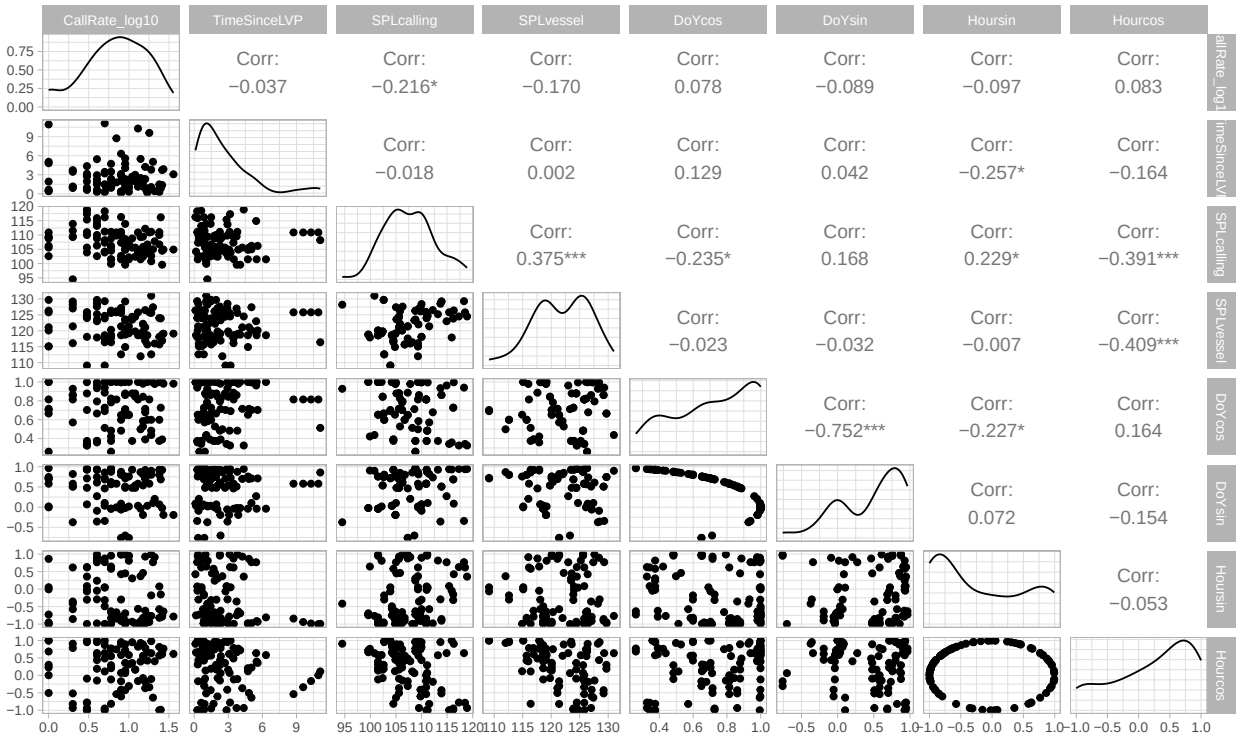
```
##                 n         min        mean      median         max          sd
## Date           93         Inf         NaN          NA        -Inf          NA
## Month          93   1.0000000   4.1397849   2.0000000  12.0000000   4.1167510
## DoY            93   1.0000000 110.0537634  54.0000000 365.0000000 128.9833462
## Hour           93   0.0000000  13.5161290  17.0000000  23.0000000   7.4346616
## Season*        93   1.0000000   2.2043011   2.0000000   3.0000000   0.4791101
## CallRate       93   1.0000000   9.5591398   8.0000000  36.0000000   7.4373651
## HourDecimal    93   0.1932778  14.0329457  17.2927257  23.9722222   7.4475570
## TimeSinceLVP   93   0.1127083   2.5389105   1.9230556  11.0916111   2.3619948
## SPLcalling     93  94.5000000 107.6989247 107.3000000 118.9000000   4.8879731
## SPLvessel      93 109.1000000 121.8172043 122.0000000 131.0000000   4.8780704
## DoYcos         93   0.2603765   0.7322226   0.7832659   0.9999908   0.2342086
## DoYsin         93  -0.7607201   0.4564720   0.5804549   0.9655072   0.4510064
## Hoursin        93  -0.9999870  -0.2025692  -0.4173580   0.9929312   0.7216207
## Hourcos        93  -0.9997351   0.2309184   0.3973491   0.9999736   0.6269601
## CallRate_log10 93   0.0000000   0.8339571   0.9030900   1.5563025   0.3880536
##                      range
```

```
## Date                 -Inf
## Month          11.0000000
## DoY           364.0000000
## Hour           23.0000000
## Season*         2.0000000
## CallRate       35.0000000
## HourDecimal    23.7789444
## TimeSinceLVP   10.9789028
## SPLcalling     24.4000000
## SPLvessel      21.9000000
## DoYcos          0.7396143
## DoYsin          1.7262273
## Hoursin         1.9929182
## Hourcos         1.9997087
## CallRate_log10  1.5563025
```

```r
write.csv(summary_DC, 'dataderived/summary_DC.csv')


## Call rates plots ----

# Plot bout duration distributions
p1 <- DC %>%
    ggplot(aes(x = CallRate)) +
    geom_histogram(fill = "blue", alpha = 0.5, color = NA, binwidth = 2) +
    labs(x = "Call rate (number of calls)",
         y = "Frequency")
p2 <- DC %>%
    ggplot(aes(x = CallRate)) +
    geom_histogram(fill = "blue", alpha = 0.5, color = NA) +
    scale_x_log10() +
    labs(x = "Call rate (number of calls)",
         y = "Frequency")
p1 + p2 +
    plot_annotation(tag_levels = "A")
```




```
ggsave("images/CallRate.png", width = 8, height = 4, dpi = 600)


# Set the response variables and predictors
RESPONSES <- c("CallRate_log10")

# Response variables are the first rows/columns in the matrix
DC %>%
    dplyr::select(all_of(c(RESPONSES, PREDICTORS_cycle))) %>%
    GGally::ggpairs() +
    ggplot2::theme(
        axis.text = ggplot2::element_text(size = 8),
        strip.text = ggplot2::element_text(size = 8)
    )
```




```
ggsave("images/CallRate_scatter.png", width = 10, height = 8, dpi = 600)

# Show correlations and p-values
DC %>%
    dplyr::select(all_of(c(RESPONSES, PREDICTORS_cycle))) %>%
    as.matrix() %>%
    Hmisc::rcorr()
```

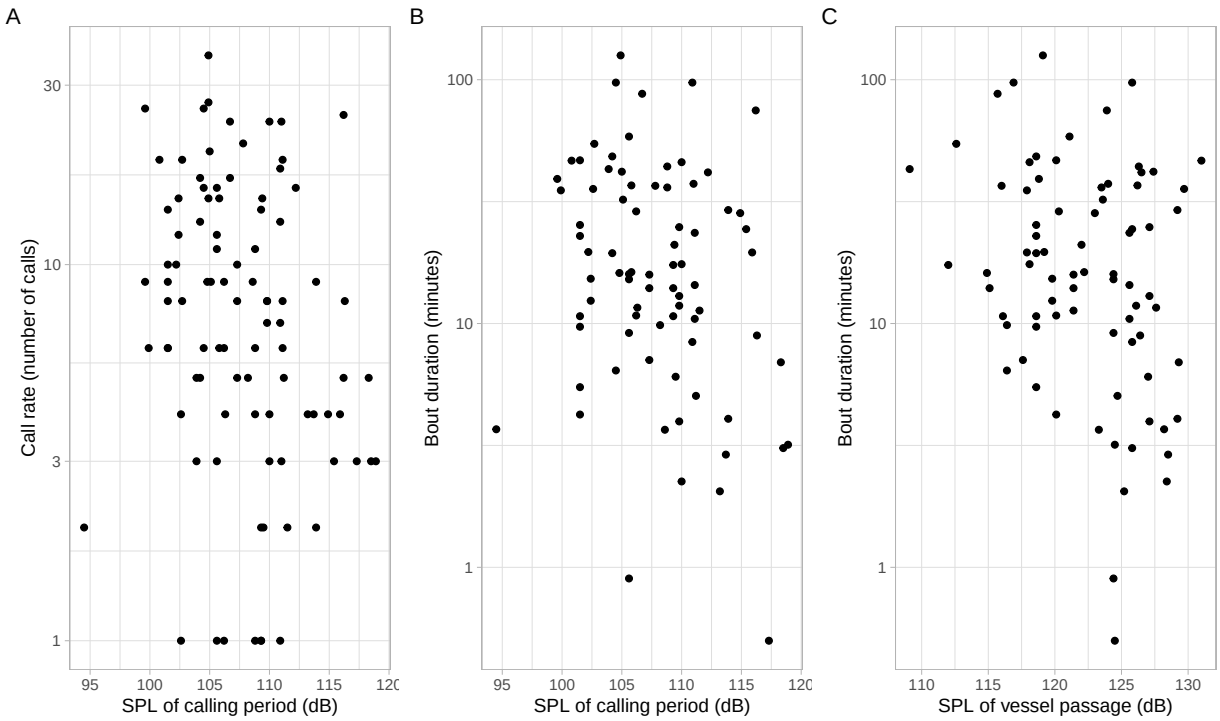
```
##                CallRate_log10 TimeSinceLVP SPLcalling SPLvessel DoYcos DoYsin
## CallRate_log10           1.00        -0.04      -0.22     -0.17   0.08  -0.09
## TimeSinceLVP            -0.04         1.00      -0.02      0.00   0.13   0.04
## SPLcalling              -0.22        -0.02       1.00      0.37  -0.24   0.17
## SPLvessel               -0.17         0.00       0.37      1.00  -0.02  -0.03
## DoYcos                   0.08         0.13      -0.24     -0.02   1.00  -0.75
## DoYsin                  -0.09         0.04       0.17     -0.03  -0.75   1.00
## Hoursin                 -0.10        -0.26       0.23     -0.01  -0.23   0.07
## Hourcos                  0.08        -0.16      -0.39     -0.41   0.16  -0.15
##                Hoursin Hourcos
## CallRate_log10   -0.10    0.08
## TimeSinceLVP     -0.26   -0.16
## SPLcalling        0.23   -0.39
## SPLvessel        -0.01   -0.41
## DoYcos           -0.23    0.16
## DoYsin            0.07   -0.15
## Hoursin           1.00   -0.05
## Hourcos          -0.05    1.00
##
## n= 93
##
```

```
##
## P
##                CallRate_log10 TimeSinceLVP SPLcalling SPLvessel DoYcos DoYsin
## CallRate_log10                     0.7236     0.0371     0.1036  0.4595 0.3946
## TimeSinceLVP   0.7236                        0.8604     0.9830  0.2195 0.6908
## SPLcalling     0.0371          0.8604                   0.0002  0.0233 0.1069
## SPLvessel      0.1036          0.9830       0.0002              0.8302 0.7639
## DoYcos         0.4595          0.2195       0.0233     0.8302          0.0000
## DoYsin         0.3946          0.6908       0.1069     0.7639  0.0000
## Hoursin        0.3563          0.0128       0.0274     0.9503  0.0289 0.4907
## Hourcos        0.4287          0.1164       0.0001     0.0000  0.1157 0.1409
##                Hoursin Hourcos
## CallRate_log10 0.3563  0.4287
## TimeSinceLVP   0.0128  0.1164
## SPLcalling     0.0274  0.0001
## SPLvessel      0.9503  0.0000
## DoYcos         0.0289  0.1157
## DoYsin         0.4907  0.1409
## Hoursin                0.6134
## Hourcos        0.6134
```

```
# Show 95% confidence intervals for correlations of Response with Predictors
sapply(PREDICTORS_cycle, function(x)
    cor.test(unlist(DC[,RESPONSES]), unlist(DC[,x]), conf.level = 0.95)$conf.int
)
```

```
##      TimeSinceLVP  SPLcalling   SPLvessel     DoYcos     DoYsin    Hoursin
## [1,]   -0.2390598 -0.40244252 -0.36107263 -0.1280974 -0.2877817 -0.2946384
## [2,]    0.1678174 -0.01336261  0.03506351  0.2769647  0.1165191  0.1091224
##         Hourcos
## [1,] -0.1227322
## [2,]  0.2819899
```

```
p1 <- DC %>%
    ggplot(aes(x = SPLcalling, y = CallRate)) +
    geom_point() +
    scale_y_log10() +
    labs(y = "Call rate (number of calls)",
         x = "SPL of calling period (dB)")
p2 <- DB %>%
    ggplot(aes(x = SPLcalling, y = BoutDur_mins)) +
    geom_point() +
    scale_y_log10() +
    labs(y = "Bout duration (minutes)",
         x = "SPL of calling period (dB)")
p3 <- DB %>%
    ggplot(aes(x = SPLvessel, y = BoutDur_mins)) +
    geom_point() +
    scale_y_log10() +
    labs(y = "Bout duration (minutes)",
         x = "SPL of vessel passage (dB)")
p1 + p2 + p3 +
    plot_annotation(tag_levels = "A")
```




```
ggsave("images/CallRateBout.png", width = 8, height = 3, dpi = 600)


## Call rate models ----

# Formula relating the response(s) to explanatory variables
fml <- as.formula(paste(RESPONSES,
                        paste(PREDICTORS_cycle, collapse = " + "),
                        sep = " ~ "))

### Linear regression ----

# Full model (response and all predictors)
M_LR_CallRate <- lm(fml, data = DC)

# Check collinearity of predictors
ols_coll_diag(M_LR_CallRate)
```

```
## Tolerance and Variance Inflation Factor
## ---------------------------------------
##      Variables Tolerance      VIF
## 1 TimeSinceLVP 0.8595796 1.163359
## 2   SPLcalling 0.7248721 1.379554
## 3    SPLvessel 0.7523850 1.329107
## 4       DoYcos 0.3826715 2.613207
## 5       DoYsin 0.4059506 2.463354
## 6      Hoursin 0.8426041 1.186797
## 7      Hourcos 0.7231538 1.382832
##
```

```
##
## Eigenvalue and Condition Index
## ------------------------------
##     Eigenvalue Condition Index    intercept TimeSinceLVP   SPLcalling
## 1 5.2747981014        1.000000 2.953017e-05 9.254659e-03 5.129621e-05
## 2 0.9620656388        2.341535 6.309981e-06 3.256112e-05 1.670558e-05
## 3 0.9169878675        2.398397 3.371422e-06 3.853852e-02 4.614910e-06
## 4 0.4871264965        3.290653 1.141185e-05 3.472853e-02 1.999544e-05
## 5 0.3335755027        3.976544 8.744210e-05 8.834172e-01 1.618075e-04
## 6 0.0238165284       14.882082 2.733203e-03 2.923872e-02 6.645242e-03
## 7 0.0010566798       70.653098 1.726499e-02 1.297317e-03 8.107090e-01
## 8 0.0005731847       95.930260 9.798637e-01 3.492480e-03 1.823914e-01
##      SPLvessel       DoYcos       DoYsin      Hoursin     Hourcos
## 1 4.152520e-05 0.0011297038 0.0038891429 0.0037216454 0.003332397
## 2 1.085700e-05 0.0001300931 0.0245172408 0.4195725404 0.220792607
## 3 3.020417e-06 0.0001242416 0.0004864174 0.3054527102 0.372863297
## 4 1.942246e-05 0.0105790019 0.2844187388 0.0845018308 0.055130784
## 5 1.400679e-04 0.0042246531 0.0023563288 0.1044569197 0.110089516
## 6 4.891900e-03 0.9397044563 0.6521852724 0.0421368119 0.013968058
## 7 4.475987e-01 0.0088137886 0.0014019200 0.0398221476 0.001632718
## 8 5.472945e-01 0.0352940615 0.0307449388 0.0003353942 0.222190624
```

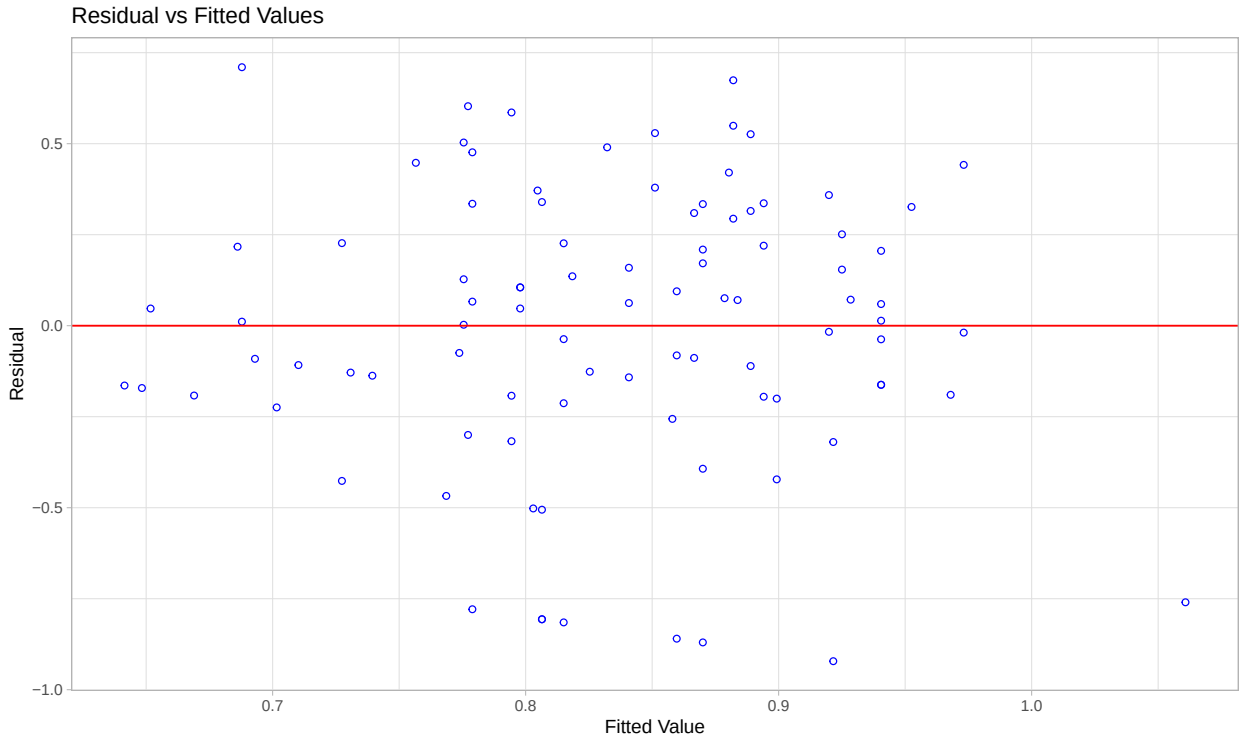
```
# Model selected
m_LR_CallRate <- ols_step_both_p(M_LR_CallRate, p_remove = 0.05)
summary(m_LR_CallRate$model)
```

```
##
## Call:
## lm(formula = paste(response, "~", paste(preds, collapse = " + ")),
##     data = l)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.92159 -0.19183  0.04721  0.29403  0.71009
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.684909   0.875937   3.065  0.00286 **
## SPLcalling  -0.017186   0.008125  -2.115  0.03714 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3809 on 91 degrees of freedom
## Multiple R-squared:  0.04686,    Adjusted R-squared:  0.03639
## F-statistic: 4.474 on 1 and 91 DF,  p-value: 0.03714
```

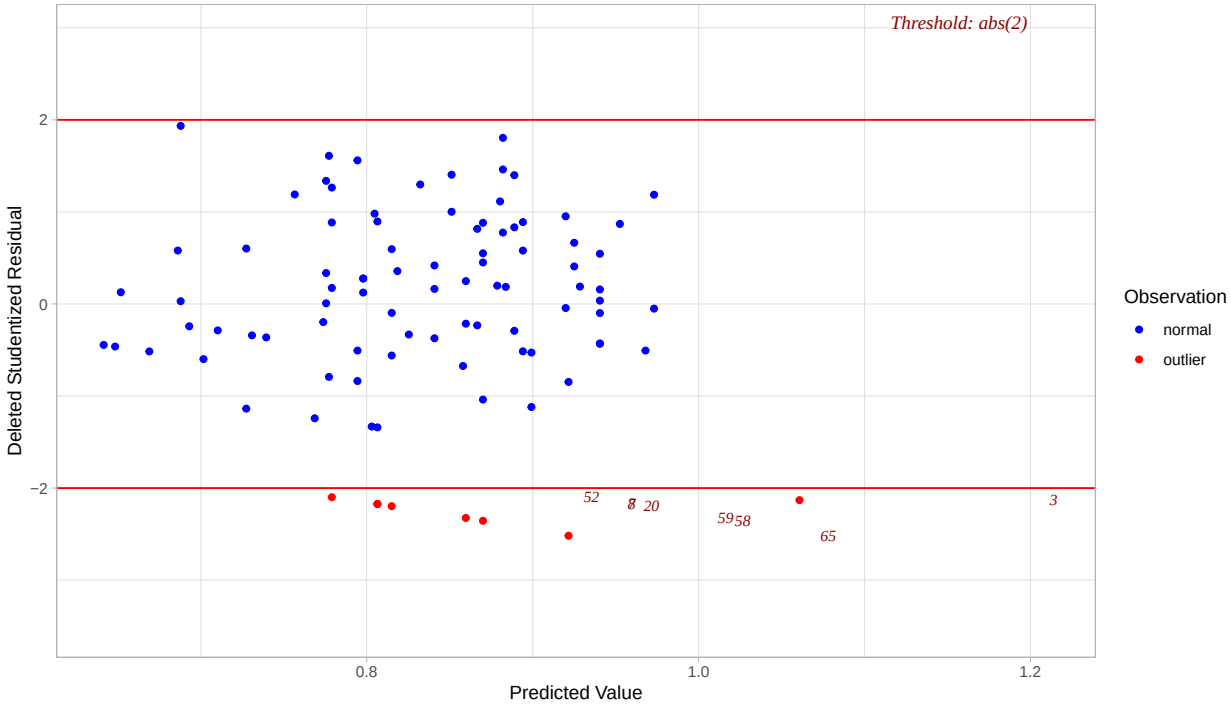
```
# Format the output
tmp <- summary(m_LR_CallRate$model)
knitr::kable(tmp$coefficients, digits = 3)
```

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	2.685	0.876	3.065	0.003
SPLcalling	-0.017	0.008	-2.115	0.037

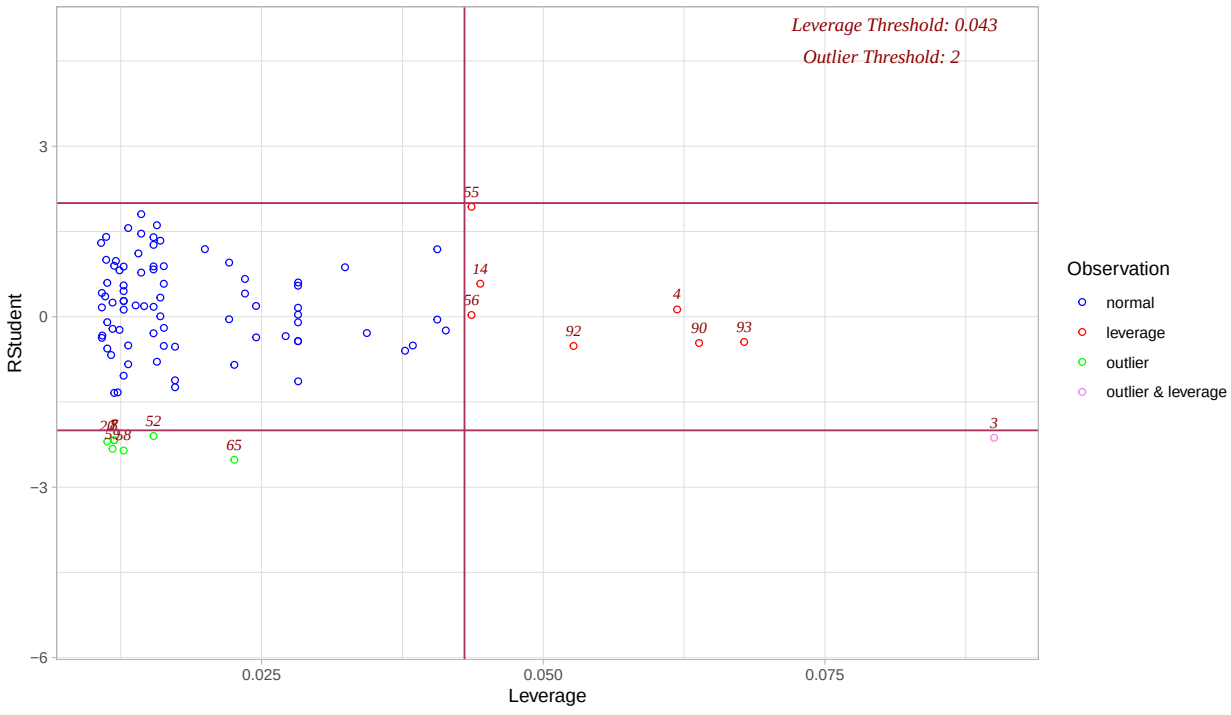
```
# Diagnostics
ols_plot_diagnostics(m_LR_CallRate$model)
```

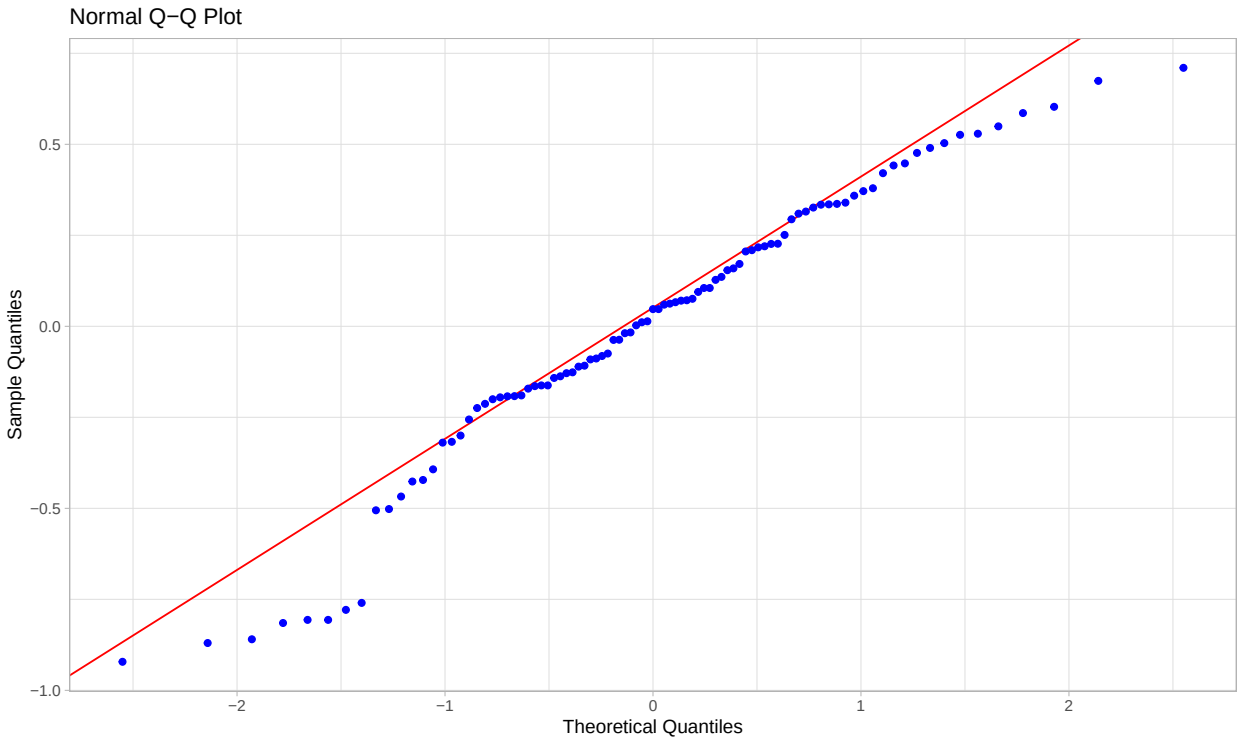


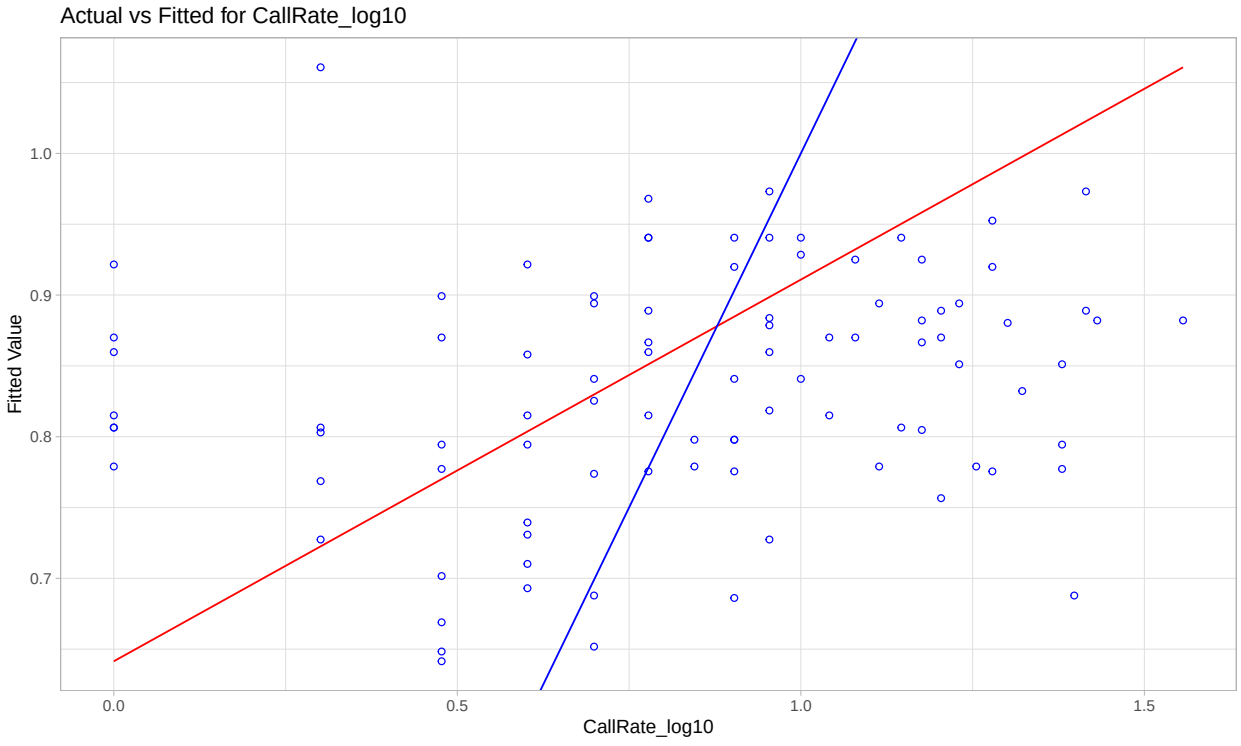
Deleted Studentized Residual vs Predicted Values

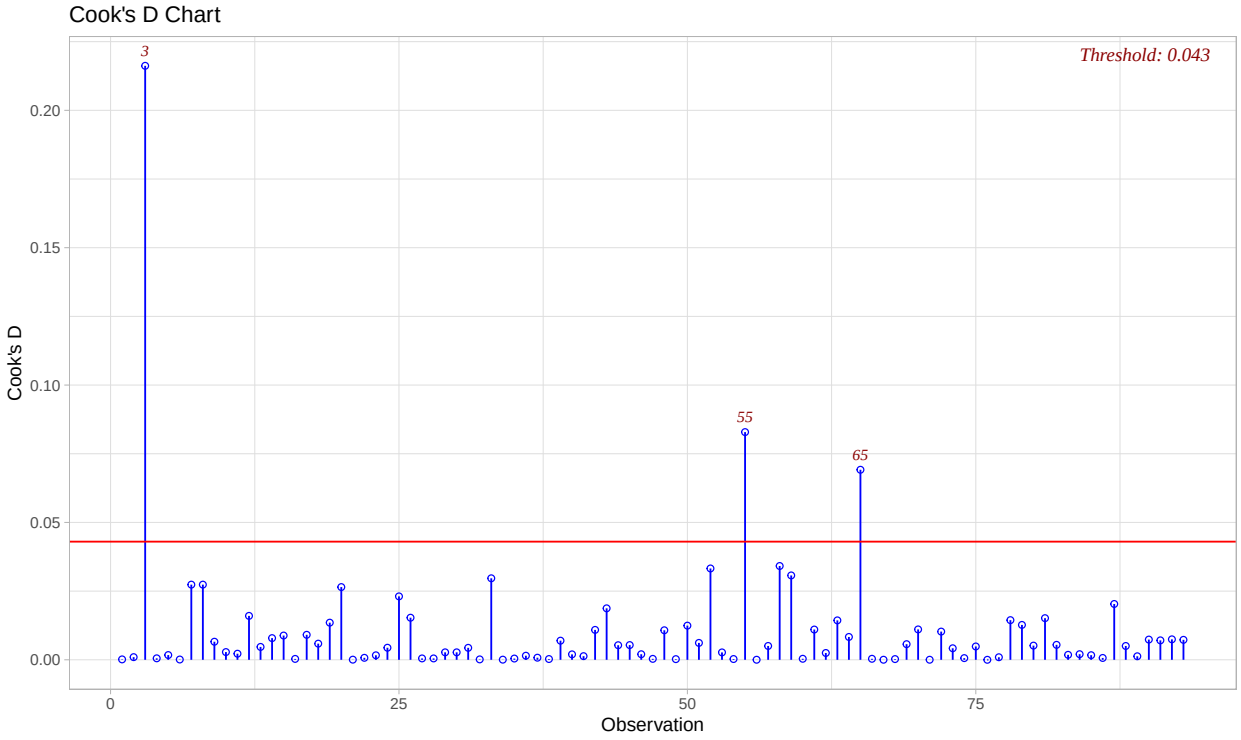


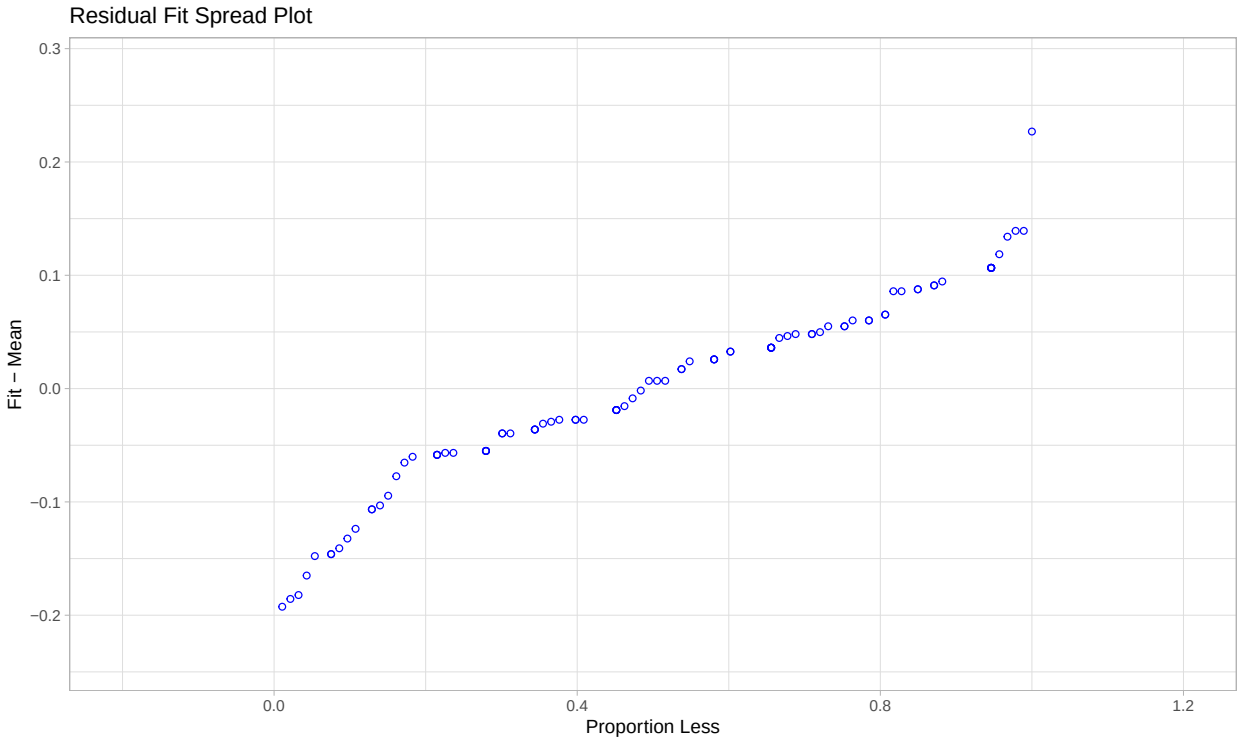
Outlier and Leverage Diagnostics for CallRate_log10

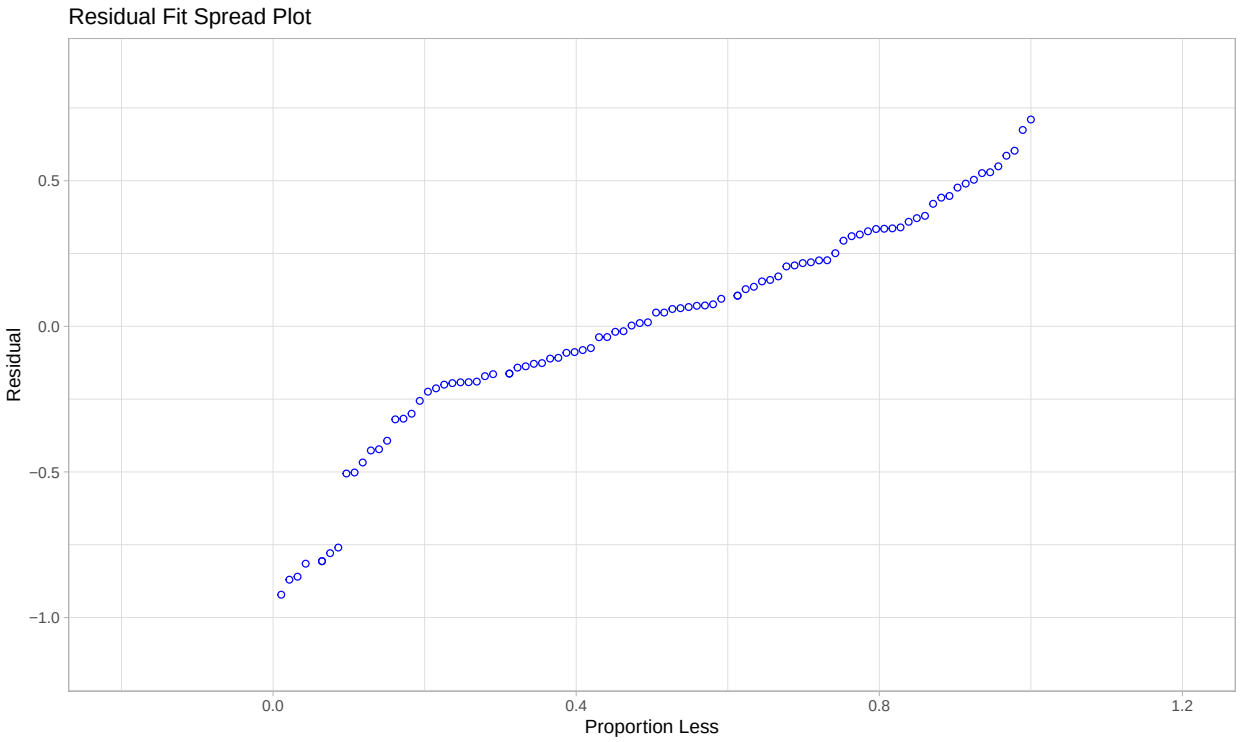


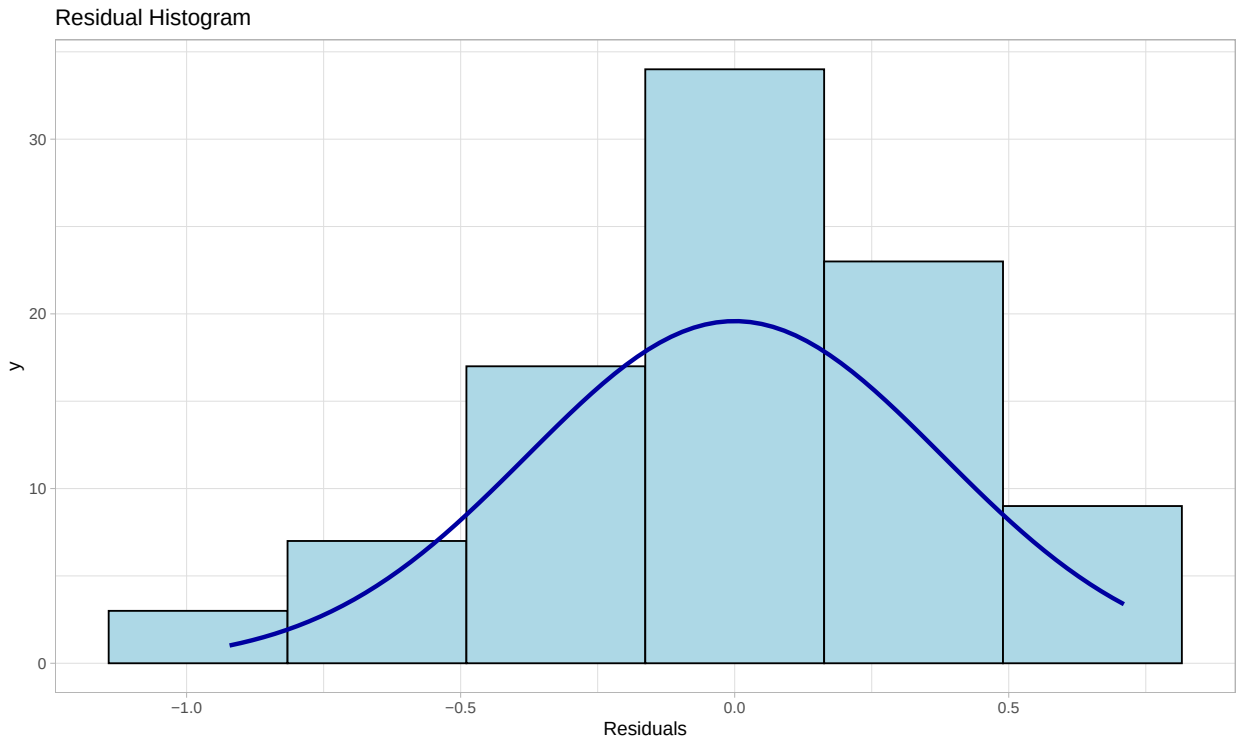

Normal Q−Q Plot
Sample Quantiles
Theoretical Quantiles
0.5
0.0
−0.5
−1.0
−2
−1
0
1
2

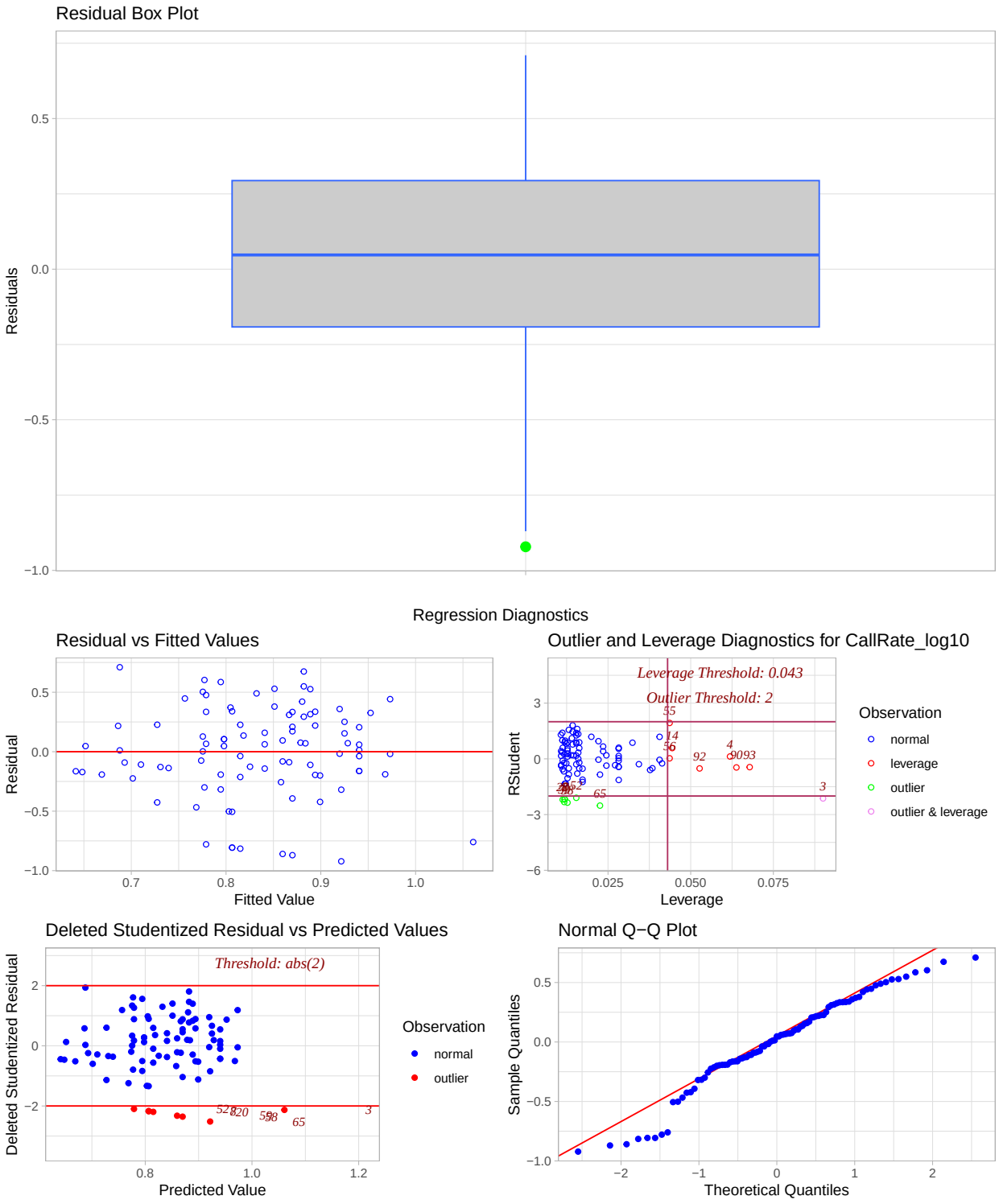

Actual vs Fitted for CallRate_log10
Fitted Value
CallRate_log10
1.0
0.9
0.8
0.7
0.0
0.5
1.0
1.5

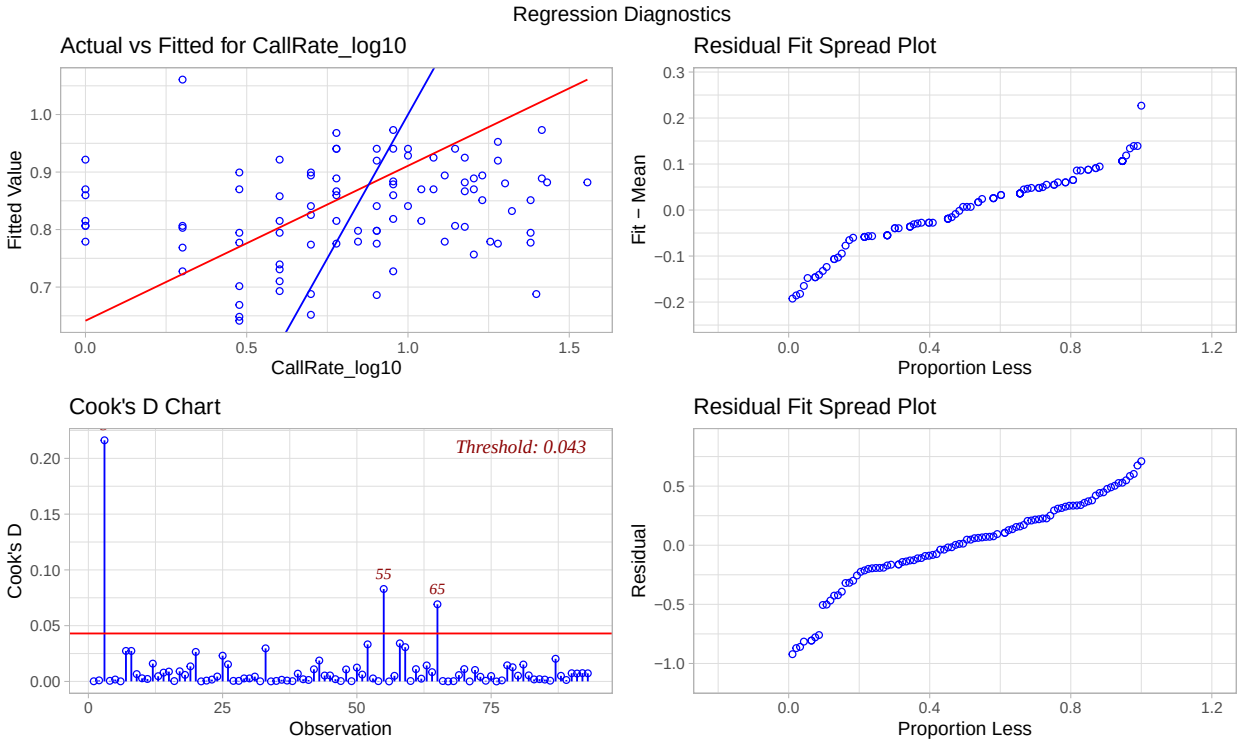

Cook's D Chart
Threshold: 0.043
3
55
65
0.20
0.15
0.10
0.05
0.00
Cook's D
0
25
50
75
Observation

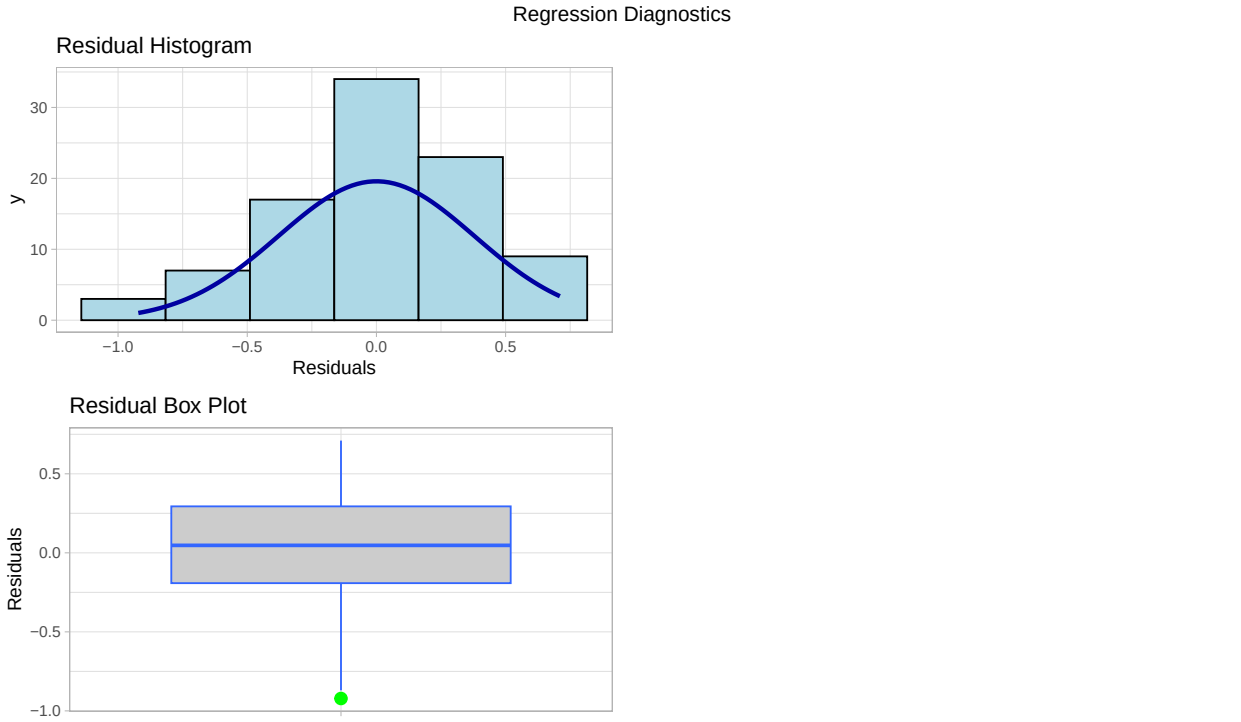

Residual Fit Spread Plot
0.3
0.2
0.1
0.0
−0.1
−0.2
Fit − Mean
0.0
0.4
0.8
1.2
Proportion Less


Residual Fit Spread Plot
Residual
0.5
0.0
−0.5
−1.0
0.0
0.4
0.8
1.2
Proportion Less

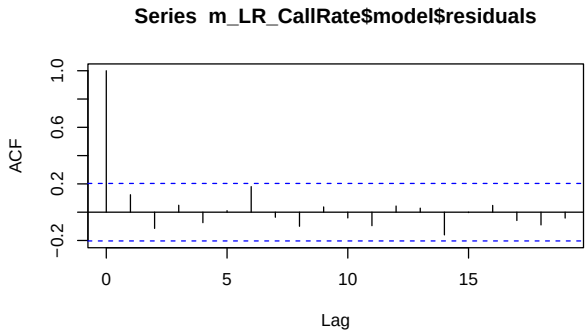

Residual Histogram
y
30
20
10
0
−1.0
−0.5
0.0
0.5
Residuals


Residual Box Plot
Residuals
0.5
0.0
−0.5
−1.0
Regression Diagnostics
Residual vs Fitted Values
Residual
Fitted Value
0.7
0.8
0.9
1.0
Outlier and Leverage Diagnostics for CallRate_log10
Leverage Threshold: 0.043
Outlier Threshold: 2
RStudent
Leverage
0.025
0.050
0.075
Observation
normal
leverage
outlier
outlier & leverage
Deleted Studentized Residual vs Predicted Values
Threshold: abs(2)
Deleted Studentized Residual
Predicted Value
Observation
normal
outlier
Normal Q−Q Plot
Sample Quantiles
Theoretical Quantiles

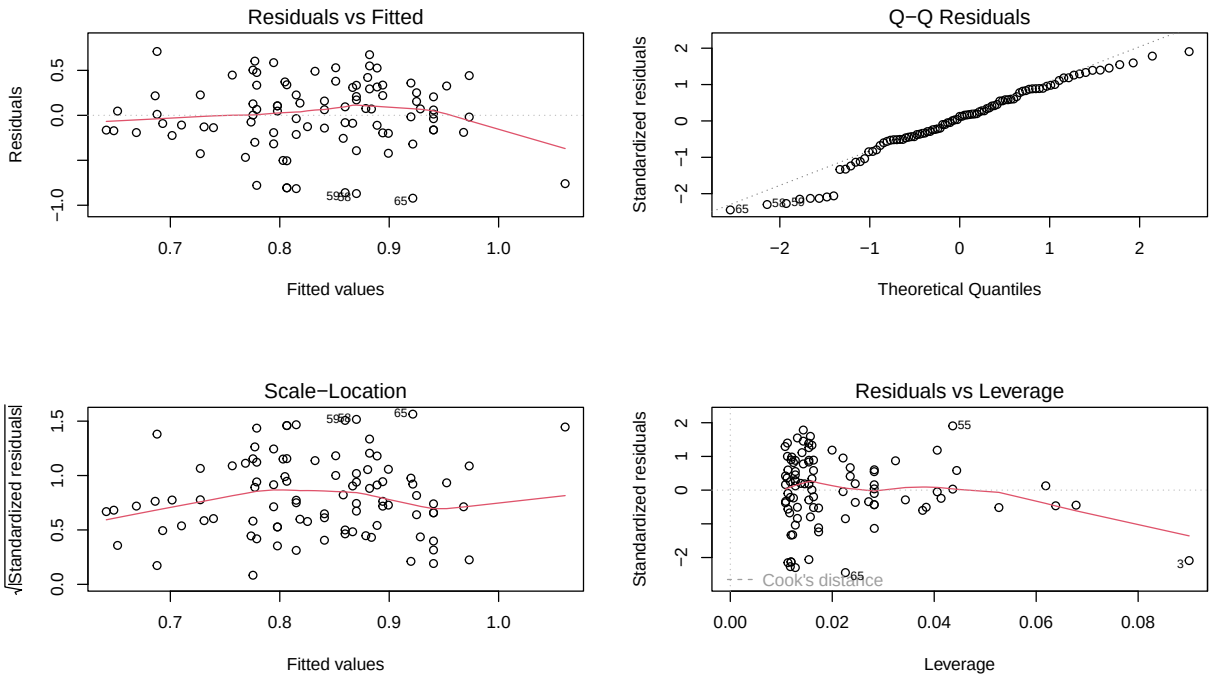





```
acf(m_LR_CallRate$model$residuals)

par(mfrow = c(2, 2))
```




```
plot(m_LR_CallRate$model)
```




```
### Random forest ----
```

```r
set.seed(123)
M_RF_CallRate <- randomForest(fml
                              ,ntree = NTREE, mtry = MTRY, nodesize = NSIZE
                              ,sampsize = ceiling(nrow(DC) * 2/3)
                              ,importance = TRUE
                              ,data = DC)
M_RF_CallRate
```

```
##
## Call:
##  randomForest(formula = fml, data = DC, ntree = NTREE, mtry = MTRY,      nodesize = NSIZE, sampsize =
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##           Mean of squared residuals: 0.1483505
##                     % Var explained: 0.41
```

```r
plot(M_RF_CallRate)

importance_values <- M_RF_CallRate$importance[,1]
importance_df <- tibble(Variable = names(importance_values),
                        Importance = importance_values) %>%
    arrange(Importance)

ggplot(importance_df, aes(x = reorder(Variable, Importance), y = Importance)) +
    geom_bar(stat = "identity") +
    coord_flip() +
    labs(x = "",
         y = "Importance")
ggsave("images/importance_CallRate.png", width = 4, height = 3, dpi = 600)

# Get PDP data
pdp_CallRate <- get_pdp_data(M_RF_CallRate, DC, PREDICTORS_cycle)

# Convert the predictor column to factor for faceting
pdp_CallRate$predictor <- factor(pdp_CallRate$predictor, levels = PREDICTORS_cycle)

# Plot PDPs with fixed y-axes
ggplot(pdp_CallRate, aes(x = x, y = yhat)) +
    geom_line() +
    facet_wrap(~predictor, scales = "free_x", nrow = 1) +
    labs(x = "Predictor values",
         y = expression(log[10] * "(Number of calls per hour)"))
ggsave("images/pdp_CallRate.png", width = 9, height = 2.5, dpi = 600)



# Call characteristics ----

## Plots ----

# Set the response variables and predictors
```

```r
RESPONSES <- c("FreqMin", "FreqMax", "FreqDelta", "Duration", "Duration90")

# Response variables are the first rows/columns in the matrix
D %>%
    dplyr::select(all_of(c(RESPONSES, PREDICTORS_cycle))) %>%
    GGally::ggpairs() +
    ggplot2::theme(
        axis.text = ggplot2::element_text(size = 8),
        strip.text = ggplot2::element_text(size = 8)
    )

ggsave("images/CallChar_scatter.png", width = 10, height = 8, dpi = 600)

# Show correlations and p-values
D %>%
    dplyr::select(all_of(c(RESPONSES, PREDICTORS_cycle))) %>%
    as.matrix() %>%
    Hmisc::rcorr()
```

```
##              FreqMin FreqMax FreqDelta Duration Duration90 TimeSinceLVP
## FreqMin         1.00    0.21     -0.29    -0.09      -0.19         0.09
## FreqMax         0.21    1.00      0.87     0.17       0.02        -0.02
## FreqDelta      -0.29    0.87      1.00     0.21       0.11        -0.06
## Duration       -0.09    0.17      0.21     1.00       0.80         0.03
## Duration90     -0.19    0.02      0.11     0.80       1.00         0.06
## TimeSinceLVP    0.09   -0.02     -0.06     0.03       0.06         1.00
## SPLcalling      0.07    0.07      0.04    -0.10      -0.03        -0.04
## SPLvessel      -0.05   -0.06     -0.03    -0.11      -0.01         0.11
## DoYcos          0.09    0.24      0.19     0.00       0.06         0.20
## DoYsin         -0.15   -0.22     -0.14    -0.01       0.01        -0.07
## Hoursin        -0.04   -0.10     -0.08     0.06       0.07        -0.22
## Hourcos         0.08   -0.04     -0.08     0.05       0.00        -0.15
##              SPLcalling SPLvessel DoYcos DoYsin Hoursin Hourcos
## FreqMin            0.07     -0.05   0.09  -0.15   -0.04    0.08
## FreqMax            0.07     -0.06   0.24  -0.22   -0.10   -0.04
## FreqDelta          0.04     -0.03   0.19  -0.14   -0.08   -0.08
## Duration          -0.10     -0.11   0.00  -0.01    0.06    0.05
## Duration90        -0.03     -0.01   0.06   0.01    0.07    0.00
## TimeSinceLVP      -0.04      0.11   0.20  -0.07   -0.22   -0.15
## SPLcalling         1.00      0.41  -0.20   0.18    0.20   -0.46
## SPLvessel          0.41      1.00  -0.02   0.01    0.08   -0.48
## DoYcos            -0.20     -0.02   1.00  -0.78   -0.21    0.11
## DoYsin             0.18      0.01  -0.78   1.00    0.10   -0.12
## Hoursin            0.20      0.08  -0.21   0.10    1.00   -0.15
## Hourcos           -0.46     -0.48   0.11  -0.12   -0.15    1.00
##
## n= 889
##
##
## P
##              FreqMin FreqMax FreqDelta Duration Duration90 TimeSinceLVP
## FreqMin              0.0000  0.0000    0.0076   0.0000     0.0052
## FreqMax      0.0000          0.0000    0.0000   0.6501     0.5751
```

```
## FreqDelta    0.0000  0.0000           0.0000   0.0011     0.0529
## Duration     0.0076  0.0000  0.0000           0.0000     0.3137
## Duration90   0.0000  0.6501  0.0011    0.0000             0.0851
## TimeSinceLVP 0.0052  0.5751  0.0529    0.3137   0.0851
## SPLcalling   0.0473  0.0289  0.2491    0.0032   0.3485     0.2054
## SPLvessel    0.1620  0.0856  0.3241    0.0011   0.8644     0.0006
## DoYcos       0.0069  0.0000  0.0000    0.9794   0.0658     0.0000
## DoYsin       0.0000  0.0000  0.0000    0.8220   0.6892     0.0432
## Hoursin      0.2368  0.0024  0.0170    0.0993   0.0295     0.0000
## Hourcos      0.0157  0.2680  0.0223    0.1609   0.9215     0.0000
##              SPLcalling SPLvessel DoYcos DoYsin Hoursin Hourcos
## FreqMin      0.0473     0.1620    0.0069 0.0000 0.2368  0.0157
## FreqMax      0.0289     0.0856    0.0000 0.0000 0.0024  0.2680
## FreqDelta    0.2491     0.3241    0.0000 0.0000 0.0170  0.0223
## Duration     0.0032     0.0011    0.9794 0.8220 0.0993  0.1609
## Duration90   0.3485     0.8644    0.0658 0.6892 0.0295  0.9215
## TimeSinceLVP 0.2054     0.0006    0.0000 0.0432 0.0000  0.0000
## SPLcalling               0.0000    0.0000 0.0000 0.0000  0.0000
## SPLvessel    0.0000                0.6237 0.8045 0.0248  0.0000
## DoYcos       0.0000     0.6237           0.0000 0.0000  0.0009
## DoYsin       0.0000     0.8045    0.0000         0.0037  0.0003
## Hoursin      0.0000     0.0248    0.0000 0.0037          0.0000
## Hourcos      0.0000     0.0000    0.0009 0.0003 0.0000
```

```
# Show 95% confidence intervals for correlations of Response with Predictors
tmp <- lapply(RESPONSES, function(r)
    sapply(PREDICTORS_cycle, function(x)
        cor.test(unlist(D[,r]), unlist(D[,x]), conf.level = 0.95)$conf.int
    )
)
names(tmp) <- RESPONSES
tmp
```

```
## $FreqMin
##      TimeSinceLVP   SPLcalling   SPLvessel     DoYcos      DoYsin     Hoursin
## [1,]   0.02812884 0.0007889881 -0.11234821 0.02497765 -0.21438020 -0.10519754
## [2,]   0.15848163 0.1317118699  0.01886581 0.15540588 -0.08585364  0.02609844
##         Hourcos
## [1,] 0.01530199
## [2,] 0.14594606
##
## $FreqMax
##      TimeSinceLVP  SPLcalling    SPLvessel    DoYcos     DoYsin     Hoursin
## [1,]  -0.08447106 0.007547449 -0.122966346 0.1757339 -0.2771590 -0.16652658
## [2,]   0.04698510 0.138347218  0.008100568 0.2997726 -0.1517288 -0.03638341
##           Hourcos
## [1,] -0.10269257
## [2,]  0.02862886
##
## $FreqDelta
##       TimeSinceLVP  SPLcalling   SPLvessel    DoYcos     DoYsin     Hoursin
## [1,] -0.1301335770 -0.02712778 -0.09864571 0.1247415 -0.2001073 -0.14505078
## [2,]  0.0008168298  0.10417876  0.03271331 0.2515691 -0.0710319 -0.01438754
##           Hourcos
```

```
## [1,] -0.14170228
## [2,] -0.01096931
##
## $Duration
##      TimeSinceLVP  SPLcalling   SPLvessel      DoYcos      DoYsin     Hoursin
## [1,]  -0.03199773 -0.16332496 -0.17403815 -0.06488981 -0.07327285 -0.01048065
## [2,]   0.09935503 -0.03309624 -0.04410676  0.06661265  0.05822223  0.12062137
##          Hourcos
## [1,] -0.01874799
## [2,]  0.11246458
##
## $Duration90
##      TimeSinceLVP  SPLcalling   SPLvessel       DoYcos      DoYsin    Hoursin
## [1,] -0.008006864 -0.09702835 -0.07146095 -0.004042329 -0.05236783 0.00731137
## [2,]  0.123058638  0.03434447  0.06003730  0.126961358  0.07911111 0.13811564
##          Hourcos
## [1,] -0.06245431
## [2,]  0.06904681
```

```
# Plot boxplots of ALL responses per season


ps <- lapply(RESPONSES, function(response) {
    ggplot(D, aes(x = .data$Season, y = .data[[response]])) +
        geom_boxplot(fill = "bisque") +
        labs(x = "Season", y = response)
})

# List sample sizes on one plot
ps[[1]] <- ps[[1]] +
    stat_summary(
        fun.data = function(x) {
            return(data.frame(
                y = min(x) - 0.1,
                label = paste0("n = ", length(x))
            ))
        },
        geom = "text",
        size = 4,
        color = "blue"
    )

p2 <- ggplot(DC, aes(x = Season, y = CallRate)) +
    geom_boxplot(fill = "darkolivegreen3") +
    labs(x = "Season", y = "Call rate") +
    stat_summary(
        fun.data = function(x) {
            return(data.frame(
                y = min(x) - 1,
                label = paste0("n = ", length(x))
            ))
        },
        geom = "text",
        size = 4,
```

```
        color = "blue"
    )

p3 <- ggplot(DB, aes(x = Season, y = BoutDur_mins)) +
    geom_boxplot(fill = "cornflowerblue") +
    labs(x = "Season", y = "Bout duration") +
    stat_summary(
        fun.data = function(x) {
            return(data.frame(
                y = min(x) - 1,
                label = paste0("n = ", length(x))
            ))
        },
        geom = "text",
        size = 4,
        color = "blue"
    )

# Combine all ggplot objects using patchwork
combined_plot <- wrap_plots(ps) + plot_spacer() + p2 + p3

# Print the combined plot
print(combined_plot) +
    plot_layout(axes = "collect") +
    plot_annotation(tag_levels = "A")

ggsave("images/Response_boxplots.png", width = 9, height = 7, dpi = 600)


## Multivariate linear regression ----

# Set formula object
mult_form_lin <- as.formula(
    paste("cbind(",
          paste0(RESPONSES, collapse = ", "),
          ") ~",
          paste0(PREDICTORS_cycle, collapse = " + ")
    )
)

# Multivariate model of linear parametric form
mult_mod_lin <- lm(mult_form_lin
                   ,data = D)

# The function summary() gives a separate summary for each response variable
summary(mult_mod_lin)
```

```
## Response FreqMin :
##
## Call:
## lm(formula = FreqMin ~ TimeSinceLVP + SPLcalling + SPLvessel +
##     DoYcos + DoYsin + Hoursin + Hourcos, data = D)
##
```

```
## Residuals:
##     Min      1Q  Median      3Q     Max
## -58.641 -12.630   1.422  14.345  62.566
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   41.2876    24.3270   1.697 0.090014 .
## TimeSinceLVP   1.1475     0.3063   3.746 0.000191 ***
## SPLcalling     0.8537     0.1741   4.904 1.12e-06 ***
## SPLvessel     -0.3204     0.1661  -1.929 0.054027 .
## DoYcos        -9.4424     4.7042  -2.007 0.045033 *
## DoYsin       -10.5536     2.3054  -4.578 5.38e-06 ***
## Hoursin       -0.6922     0.9631  -0.719 0.472532
## Hourcos        4.4306     1.3439   3.297 0.001017 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 19.03 on 881 degrees of freedom
## Multiple R-squared:  0.0669, Adjusted R-squared:  0.05949
## F-statistic: 9.024 on 7 and 881 DF,  p-value: 8.791e-11
##
##
## Response FreqMax :
##
## Call:
## lm(formula = FreqMax ~ TimeSinceLVP + SPLcalling + SPLvessel +
##     DoYcos + DoYsin + Hoursin + Hourcos, data = D)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -81.39 -26.31  -5.11  20.37 172.03
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  203.3833    47.0035   4.327 1.68e-05 ***
## TimeSinceLVP  -1.3557     0.5918  -2.291 0.022217 *
## SPLcalling     1.4590     0.3363   4.338 1.60e-05 ***
## SPLvessel     -1.2713     0.3209  -3.962 8.04e-05 ***
## DoYcos        33.6416     9.0893   3.701 0.000228 ***
## DoYsin        -8.2002     4.4545  -1.841 0.065973 .
## Hoursin       -5.6617     1.8609  -3.043 0.002416 **
## Hourcos       -6.2076     2.5966  -2.391 0.017029 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 36.77 on 881 degrees of freedom
## Multiple R-squared:  0.1043, Adjusted R-squared:  0.09714
## F-statistic: 14.65 on 7 and 881 DF,  p-value: < 2.2e-16
##
##
## Response FreqDelta :
##
## Call:
## lm(formula = FreqDelta ~ TimeSinceLVP + SPLcalling + SPLvessel +
```

```
##     DoYcos + DoYsin + Hoursin + Hourcos, data = D)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -81.862 -26.141  -6.024  22.016 167.864
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  162.0439    48.6910   3.328 0.000911 ***
## TimeSinceLVP  -2.5030     0.6131  -4.083 4.86e-05 ***
## SPLcalling     0.6051     0.3484   1.737 0.082794 .
## SPLvessel     -0.9503     0.3324  -2.859 0.004353 **
## DoYcos        43.0967     9.4156   4.577 5.39e-06 ***
## DoYsin         2.3474     4.6144   0.509 0.611087
## Hoursin       -4.9684     1.9277  -2.577 0.010116 *
## Hourcos      -10.6394     2.6899  -3.955 8.26e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 38.09 on 881 degrees of freedom
## Multiple R-squared:  0.07804,    Adjusted R-squared:  0.07072
## F-statistic: 10.65 on 7 and 881 DF,  p-value: 6.424e-13
##
##
## Response Duration :
##
## Call:
## lm(formula = Duration ~ TimeSinceLVP + SPLcalling + SPLvessel +
##     DoYcos + DoYsin + Hoursin + Hourcos, data = D)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.63014 -0.17641 -0.00864  0.15377  1.17161
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)   2.241590   0.318491   7.038 3.92e-12 ***
## TimeSinceLVP  0.006876   0.004010   1.715   0.0868 .
## SPLcalling   -0.004732   0.002279  -2.077   0.0381 *
## SPLvessel    -0.005333   0.002174  -2.453   0.0144 *
## DoYcos       -0.021043   0.061588  -0.342   0.7327
## DoYsin       -0.007128   0.030183  -0.236   0.8134
## Hoursin       0.031192   0.012609   2.474   0.0136 *
## Hourcos      -0.006154   0.017595  -0.350   0.7266
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2491 on 881 degrees of freedom
## Multiple R-squared:  0.02507,    Adjusted R-squared:  0.01732
## F-statistic: 3.236 on 7 and 881 DF,  p-value: 0.002141
##
##
## Response Duration90 :
##
```

```
## Call:
## lm(formula = Duration90 ~ TimeSinceLVP + SPLcalling + SPLvessel +
##     DoYcos + DoYsin + Hoursin + Hourcos, data = D)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.48886 -0.13610 -0.01742  0.11342  0.94291
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)   0.7732759  0.2529610   3.057 0.002304 **
## TimeSinceLVP  0.0048259  0.0031851   1.515 0.130089
## SPLcalling   -0.0019105  0.0018100  -1.055 0.291487
## SPLvessel     0.0001164  0.0017269   0.067 0.946297
## DoYcos        0.1759692  0.0489163   3.597 0.000339 ***
## DoYsin        0.0755808  0.0239728   3.153 0.001672 **
## Hoursin       0.0341926  0.0100147   3.414 0.000669 ***
## Hourcos       0.0033461  0.0139745   0.239 0.810816
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1979 on 881 degrees of freedom
## Multiple R-squared:  0.028,  Adjusted R-squared:  0.02028
## F-statistic: 3.626 on 7 and 881 DF,  p-value: 0.0007311
```

```
# Run multivariate tests and check if thy agree
car::Anova(mult_mod_lin, test.statistic = "Wilks")
```

```
##
## Type II MANOVA Tests: Wilks test statistic
##              Df test stat approx F num Df den Df    Pr(>F)
## TimeSinceLVP  1   0.96529   6.3080      5    877 9.204e-06 ***
## SPLcalling    1   0.94586  10.0395      5    877 2.330e-09 ***
## SPLvessel     1   0.96767   5.8606      5    877 2.456e-05 ***
## DoYcos        1   0.91871  15.5195      5    877 1.194e-14 ***
## DoYsin        1   0.94843   9.5375      5    877 7.132e-09 ***
## Hoursin       1   0.97538   4.4268      5    877 0.0005478 ***
## Hourcos       1   0.97609   4.2970      5    877 0.0007226 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
car::Anova(mult_mod_lin, test.statistic = "Pillai")
```

```
##
## Type II MANOVA Tests: Pillai test statistic
##              Df test stat approx F num Df den Df    Pr(>F)
## TimeSinceLVP  1  0.034715   6.3080      5    877 9.204e-06 ***
## SPLcalling    1  0.054139  10.0395      5    877 2.330e-09 ***
## SPLvessel     1  0.032332   5.8606      5    877 2.456e-05 ***
## DoYcos        1  0.081288  15.5195      5    877 1.194e-14 ***
## DoYsin        1  0.051572   9.5375      5    877 7.132e-09 ***
## Hoursin       1  0.024617   4.4268      5    877 0.0005478 ***
## Hourcos       1  0.023913   4.2970      5    877 0.0007226 ***
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
car::Anova(mult_mod_lin, test.statistic = "Hotelling-Lawley")
```

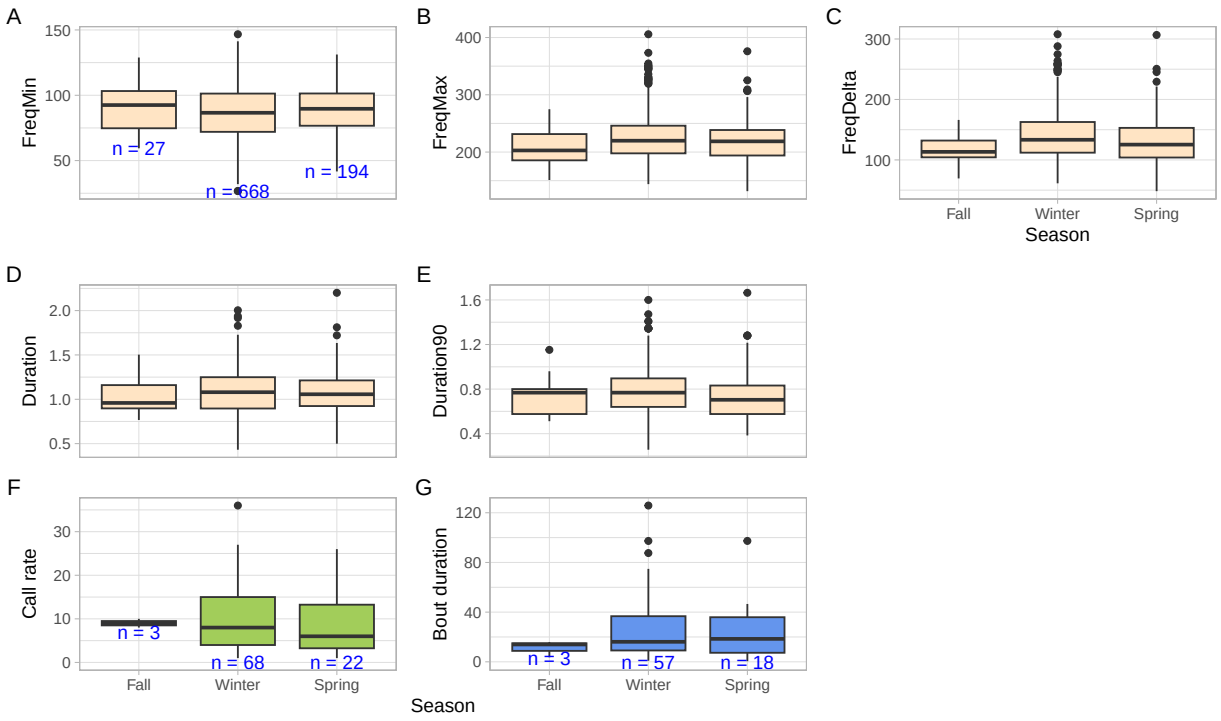
```
##
## Type II MANOVA Tests: Hotelling-Lawley test statistic
##              Df test stat approx F num Df den Df    Pr(>F)
## TimeSinceLVP  1  0.035963   6.3080      5    877 9.204e-06 ***
## SPLcalling    1  0.057238  10.0395      5    877 2.330e-09 ***
## SPLvessel     1  0.033413   5.8606      5    877 2.456e-05 ***
## DoYcos        1  0.088480  15.5195      5    877 1.194e-14 ***
## DoYsin        1  0.054376   9.5375      5    877 7.132e-09 ***
## Hoursin       1  0.025239   4.4268      5    877 0.0005478 ***
## Hourcos       1  0.024498   4.2970      5    877 0.0007226 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
car::Anova(mult_mod_lin, test.statistic = "Roy")
```

```
##
## Type II MANOVA Tests: Roy test statistic
##              Df test stat approx F num Df den Df    Pr(>F)
## TimeSinceLVP  1  0.035963   6.3080      5    877 9.204e-06 ***
## SPLcalling    1  0.057238  10.0395      5    877 2.330e-09 ***
## SPLvessel     1  0.033413   5.8606      5    877 2.456e-05 ***
## DoYcos        1  0.088480  15.5195      5    877 1.194e-14 ***
## DoYsin        1  0.054376   9.5375      5    877 7.132e-09 ***
## Hoursin       1  0.025239   4.4268      5    877 0.0005478 ***
## Hourcos       1  0.024498   4.2970      5    877 0.0007226 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

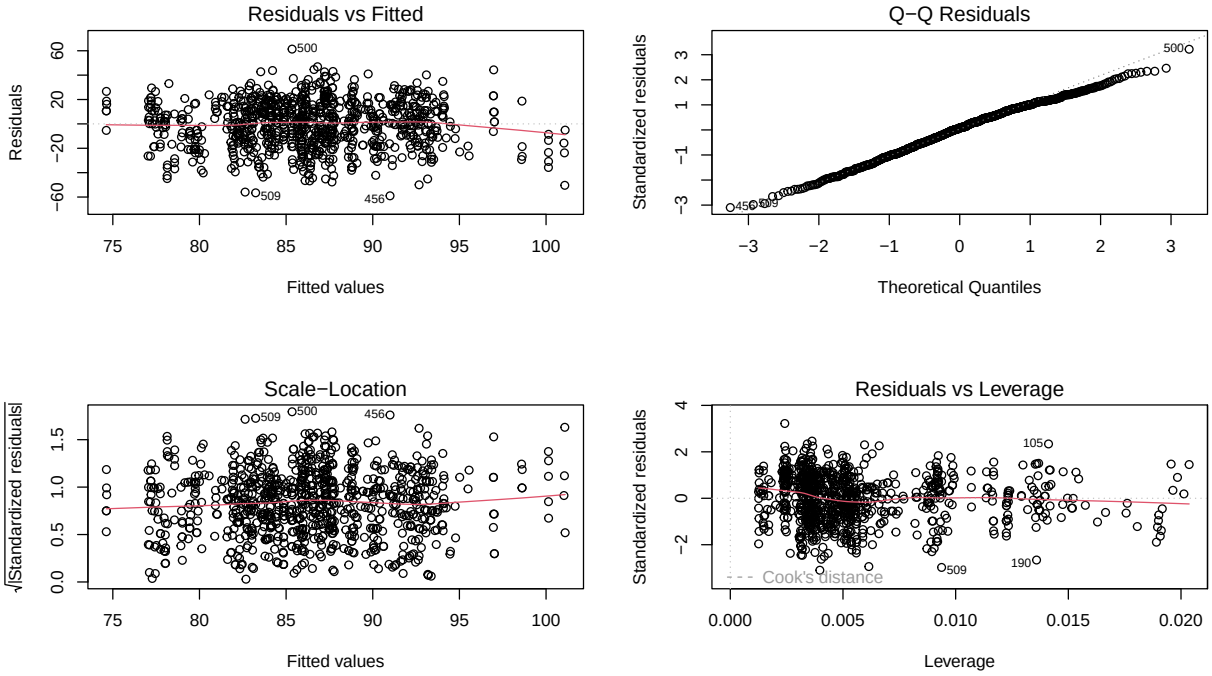
```
## Multiple (individual) linear regressions ----

# Run model selection and save results in a list
invisible(
    lapply(RESPONSES, function(y) { # y = RESPONSES[1]
        print(y)
        fml <- as.formula(paste(y,
                                paste(PREDICTORS_cycle, collapse = " + "),
                                sep = " ~ "))

        # Full model (response and all predictors)
        Mfull <- lm(fml, data = D)

        # Model selected
        Mreduced <- ols_step_both_p(Mfull, p_remove = 0.05)
        print(summary(Mreduced$model))

        # Format the output
        tmp <- summary(Mreduced$model)
```

```
        print(knitr::kable(tmp$coefficients, digits = 3))

        # Diagnostics
        par(mfrow = c(2, 2))
        plot(Mreduced$model)
    })
)
```



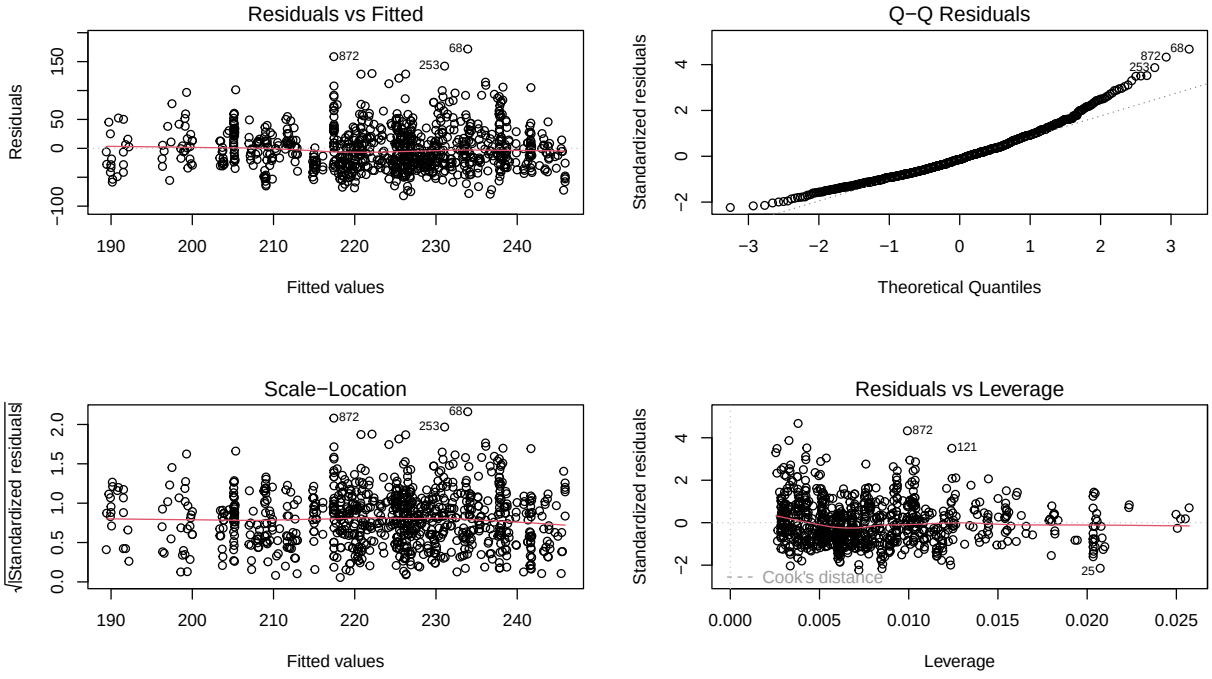

```
## [1] "FreqMin"
##
## Call:
## lm(formula = paste(response, "~", paste(preds, collapse = " + ")),
##     data = l)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -58.996 -13.056   1.426  14.398  61.356
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)    2.7178    18.0833   0.150 0.880566
## DoYsin        -6.7278     1.4453  -4.655 3.74e-06 ***
## SPLcalling     0.7730     0.1672   4.622 4.36e-06 ***
## Hourcos        5.3360     1.2566   4.246 2.40e-05 ***
## TimeSinceLVP   1.0062     0.2921   3.445 0.000598 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 19.08 on 884 degrees of freedom
```

```
## Multiple R-squared:  0.05855,    Adjusted R-squared:  0.05429
## F-statistic: 13.74 on 4 and 884 DF,  p-value: 7.038e-11
##
##
##
## |             | Estimate| Std. Error| t value| Pr(>&#124;t&#124;)|
## |:------------|--------:|----------:|-------:|------------------:|
## |(Intercept)  |    2.718|     18.083|   0.150|              0.881|
## |DoYsin       |   -6.728|      1.445|  -4.655|              0.000|
## |SPLcalling   |    0.773|      0.167|   4.622|              0.000|
## |Hourcos      |    5.336|      1.257|   4.246|              0.000|
## |TimeSinceLVP |    1.006|      0.292|   3.445|              0.001|
```



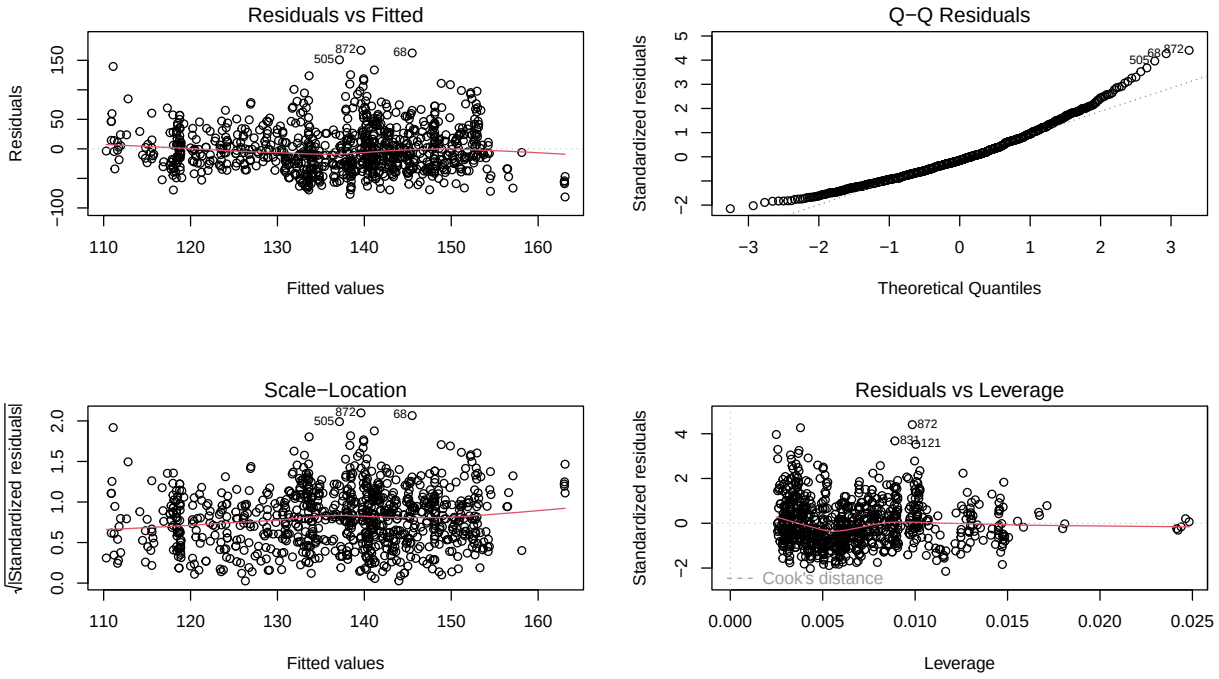

```
## [1] "FreqMax"
##
## Call:
## lm(formula = paste(response, "~", paste(preds, collapse = " + ")),
##     data = l)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -82.003 -26.308  -5.063  19.545 171.793
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  190.4795    46.5408   4.093 4.65e-05 ***
## DoYcos        46.6585     5.7188   8.159 1.16e-15 ***
## SPLcalling     1.4215     0.3362   4.229 2.60e-05 ***
## SPLvessel     -1.2378     0.3208  -3.858 0.000122 ***
```

```
## Hoursin       -5.3136     1.8537  -2.866 0.004250 **
## TimeSinceLVP  -1.4938     0.5879  -2.541 0.011220 *
## Hourcos       -6.0300     2.5984  -2.321 0.020530 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 36.82 on 882 degrees of freedom
## Multiple R-squared:  0.1008, Adjusted R-squared:  0.0947
## F-statistic: 16.48 on 6 and 882 DF,  p-value: < 2.2e-16
##
##
##
## |             | Estimate| Std. Error| t value| Pr(>&#124;t&#124;)|
## |:------------|--------:|----------:|-------:|------------------:|
## |(Intercept)  |  190.480|     46.541|   4.093|              0.000|
## |DoYcos       |   46.659|      5.719|   8.159|              0.000|
## |SPLcalling   |    1.421|      0.336|   4.229|              0.000|
## |SPLvessel    |   -1.238|      0.321|  -3.858|              0.000|
## |Hoursin      |   -5.314|      1.854|  -2.866|              0.004|
## |TimeSinceLVP |   -1.494|      0.588|  -2.541|              0.011|
## |Hourcos      |   -6.030|      2.598|  -2.321|              0.021|
```



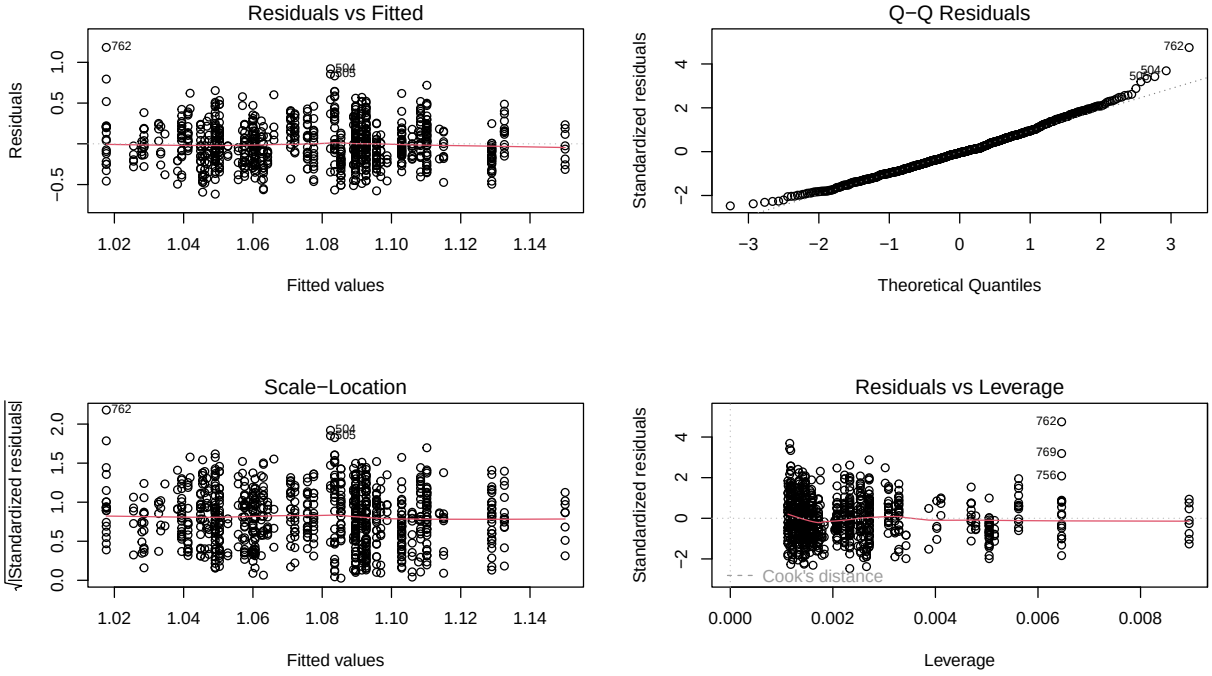

```
## [1] "FreqDelta"
##
## Call:
## lm(formula = paste(response, "~", paste(preds, collapse = " + ")),
##     data = l)
##
## Residuals:
```

```
##     Min      1Q  Median      3Q     Max
## -81.419 -26.654  -5.588  22.808 167.055
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  215.1921    39.2500   5.483 5.47e-08 ***
## DoYcos        37.9357     5.8650   6.468 1.64e-10 ***
## TimeSinceLVP  -2.5545     0.6064  -4.212 2.79e-05 ***
## Hourcos      -12.1766     2.5556  -4.765 2.21e-06 ***
## SPLvessel     -0.8097     0.3211  -2.522   0.0119 *
## Hoursin       -4.7034     1.9081  -2.465   0.0139 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 38.12 on 883 degrees of freedom
## Multiple R-squared:  0.07449,    Adjusted R-squared:  0.06925
## F-statistic: 14.21 on 5 and 883 DF,  p-value: 2.134e-13
##
##
##
## |             | Estimate| Std. Error| t value| Pr(>&#124;t&#124;)|
## |:------------|--------:|----------:|-------:|------------------:|
## |(Intercept)  |  215.192|     39.250|   5.483|              0.000|
## |DoYcos       |   37.936|      5.865|   6.468|              0.000|
## |TimeSinceLVP |   -2.555|      0.606|  -4.212|              0.000|
## |Hourcos      |  -12.177|      2.556|  -4.765|              0.000|
## |SPLvessel    |   -0.810|      0.321|  -2.522|              0.012|
## |Hoursin      |   -4.703|      1.908|  -2.465|              0.014|
```



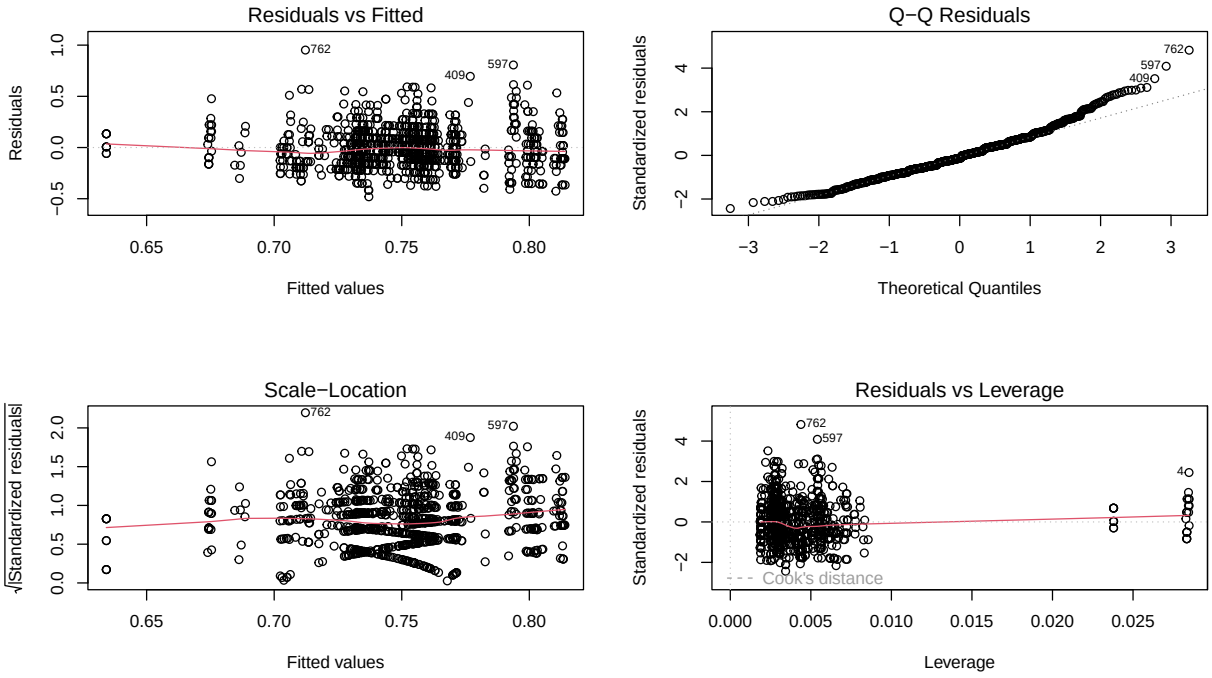

```
## [1] "Duration"
```

```
##
## Call:
## lm(formula = paste(response, "~", paste(preds, collapse = " + ")),
##     data = l)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.61762 -0.16790 -0.01172  0.15756  1.18294
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.810127   0.223347   8.105 1.75e-15 ***
## SPLvessel   -0.006049   0.001843  -3.282  0.00107 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2499 on 887 degrees of freedom
## Multiple R-squared:  0.012,  Adjusted R-squared:  0.01089
## F-statistic: 10.77 on 1 and 887 DF,  p-value: 0.00107
##
##
##
## |            | Estimate| Std. Error| t value| Pr(>&#124;t&#124;)|
## |:-----------|--------:|----------:|-------:|------------------:|
## |(Intercept) |    1.810|      0.223|   8.105|              0.000|
## |SPLvessel   |   -0.006|      0.002|  -3.282|              0.001|
```




```
## [1] "Duration90"
##
```

```
## Call:
## lm(formula = paste(response, "~", paste(preds, collapse = " + ")),
##     data = l)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.4811 -0.1303 -0.0252  0.1061  0.9518
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 0.579769   0.044164  13.128  < 2e-16 ***
## Hoursin     0.029207   0.009667   3.021  0.00259 **
## DoYcos      0.194407   0.047802   4.067 5.19e-05 ***
## DoYsin      0.078106   0.023674   3.299  0.00101 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1979 on 885 degrees of freedom
## Multiple R-squared:  0.02362,    Adjusted R-squared:  0.02031
## F-statistic: 7.137 on 3 and 885 DF,  p-value: 9.718e-05
##
##
##
## |            | Estimate| Std. Error| t value| Pr(>&#124;t&#124;)|
## |:-----------|--------:|----------:|-------:|------------------:|
## |(Intercept) |    0.580|      0.044|  13.128|              0.000|
## |Hoursin     |    0.029|      0.010|   3.021|              0.003|
## |DoYcos      |    0.194|      0.048|   4.067|              0.000|
## |DoYsin      |    0.078|      0.024|   3.299|              0.001|
```



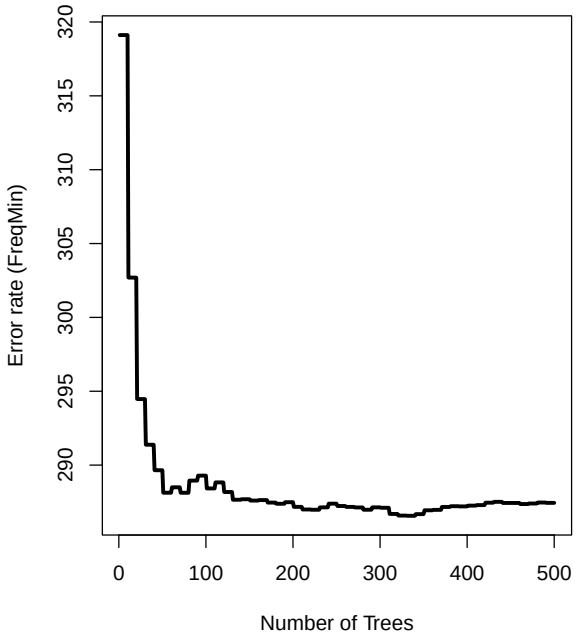
```r
## Random forest ----

# Set formula object
mult_form_rf <- as.formula(
    paste("Multivar(",
          paste0(RESPONSES, collapse = ", "),
          ") ~",
          paste0(PREDICTORS_cycle, collapse = " + ")
    )
)

# Multivariate model of nonlinear nonparametric form
set.seed(123)
mult_mod_rf <- rfsrc(mult_form_rf,
                     ntree = NTREE,
                     nodesize = NSIZE,
                     importance = "permute",
                     splitrule = "mahalanobis",
                     data = D)

# Print out RF results for each response
invisible(
    sapply(RESPONSES, function(i) print(mult_mod_rf, outcome.target = i))
)
```

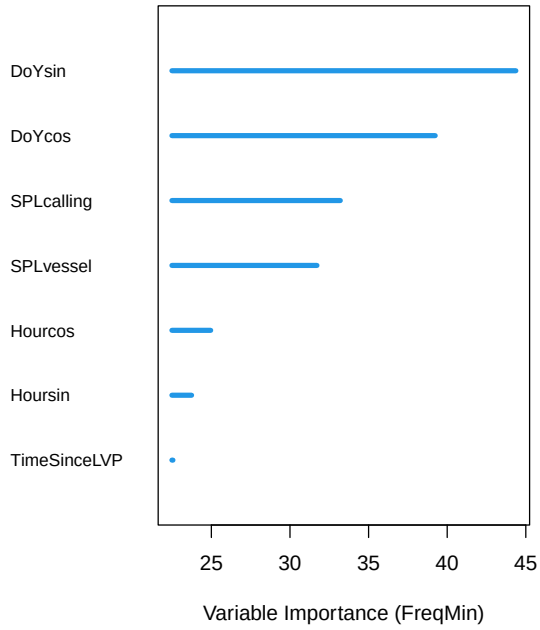
```
##                          Sample size: 889
##                      Number of trees: 500
##            Forest terminal node size: 5
##        Average no. of terminal nodes: 84.306
## No. of variables tried at each split: 3
##               Total no. of variables: 7
##               Total no. of responses: 5
##          User has requested response: FreqMin
##        Resampling used to grow trees: swor
##     Resample size used to grow trees: 562
##                             Analysis: mRF-R
##                               Family: regr+
##                       Splitting rule: mahalanobis *random*
##        Number of random split points: 10
##                      (OOB) R squared: 0.25341223
##    (OOB) Requested performance error: 287.44249765
##
##                          Sample size: 889
##                      Number of trees: 500
##            Forest terminal node size: 5
##        Average no. of terminal nodes: 84.306
## No. of variables tried at each split: 3
##               Total no. of variables: 7
##               Total no. of responses: 5
##          User has requested response: FreqMax
##        Resampling used to grow trees: swor
##     Resample size used to grow trees: 562
##                             Analysis: mRF-R
```

```
##                                Family: regr+
##                        Splitting rule: mahalanobis *random*
##        Number of random split points: 10
##                          (OOB) R squared: 0.34658439
##    (OOB) Requested performance error: 978.3355571
##
##                             Sample size: 889
##                         Number of trees: 500
##              Forest terminal node size: 5
##          Average no. of terminal nodes: 84.306
## No. of variables tried at each split: 3
##                 Total no. of variables: 7
##                 Total no. of responses: 5
##            User has requested response: FreqDelta
##          Resampling used to grow trees: swor
##       Resample size used to grow trees: 562
##                               Analysis: mRF-R
##                                 Family: regr+
##                        Splitting rule: mahalanobis *random*
##        Number of random split points: 10
##                          (OOB) R squared: 0.3181261
##    (OOB) Requested performance error: 1064.41749048
##
##                             Sample size: 889
##                         Number of trees: 500
##              Forest terminal node size: 5
##          Average no. of terminal nodes: 84.306
## No. of variables tried at each split: 3
##                 Total no. of variables: 7
##                 Total no. of responses: 5
##            User has requested response: Duration
##          Resampling used to grow trees: swor
##       Resample size used to grow trees: 562
##                               Analysis: mRF-R
##                                 Family: regr+
##                        Splitting rule: mahalanobis *random*
##        Number of random split points: 10
##                          (OOB) R squared: 0.20216369
##    (OOB) Requested performance error: 0.05039104
##
##                             Sample size: 889
##                         Number of trees: 500
##              Forest terminal node size: 5
##          Average no. of terminal nodes: 84.306
## No. of variables tried at each split: 3
##                 Total no. of variables: 7
##                 Total no. of responses: 5
##            User has requested response: Duration90
##          Resampling used to grow trees: swor
##       Resample size used to grow trees: 562
##                               Analysis: mRF-R
##                                 Family: regr+
##                        Splitting rule: mahalanobis *random*
##        Number of random split points: 10
```

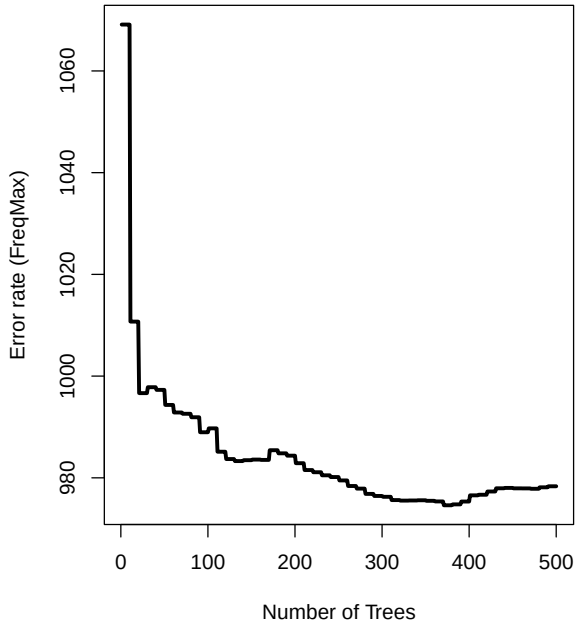
```
##                        (OOB) R squared: 0.16777385
##     (OOB) Requested performance error: 0.03325854
```

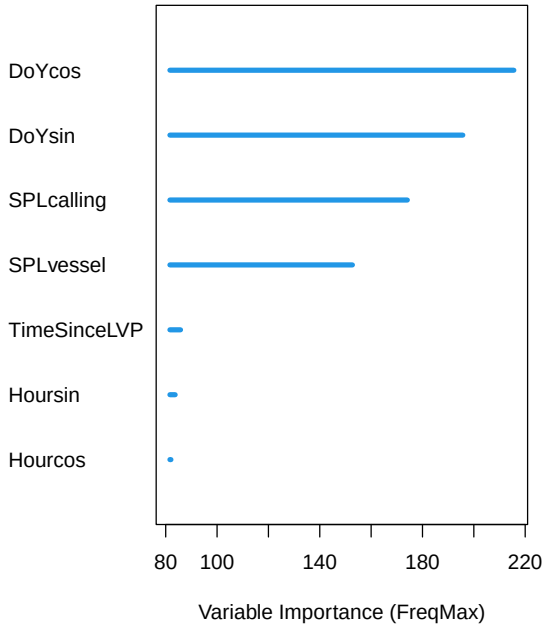
```r
# Plot RF results for each response
invisible(
    sapply(RESPONSES, function(i) plot(mult_mod_rf, m.target = i))
)
```



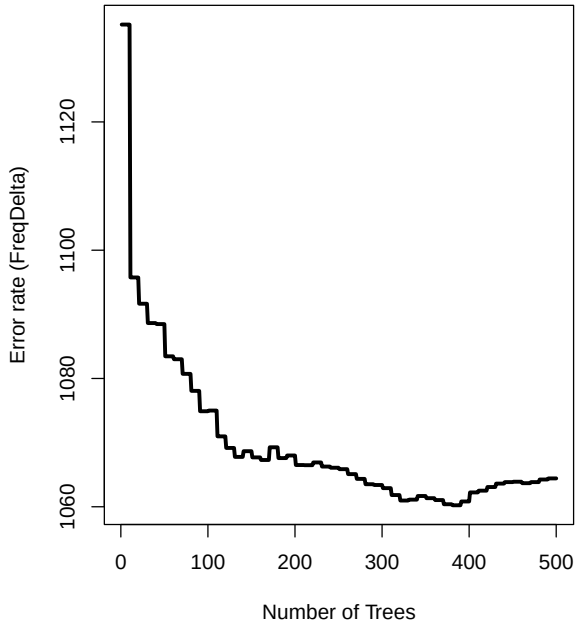


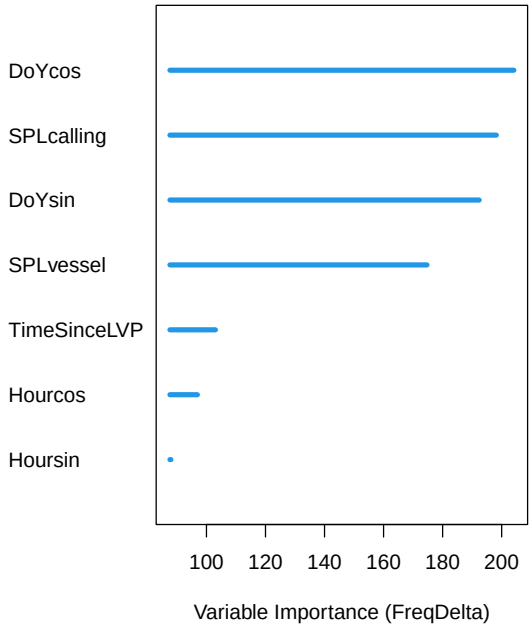

```
##
##                Importance   Relative Imp
## DoYsin            44.3670         1.0000
## DoYcos            39.2365         0.8844
## SPLcalling        33.1999         0.7483
## SPLvessel         31.7147         0.7148
## Hourcos           24.9569         0.5625
## Hoursin           23.7417         0.5351
## TimeSinceLVP      22.4915         0.5069
```



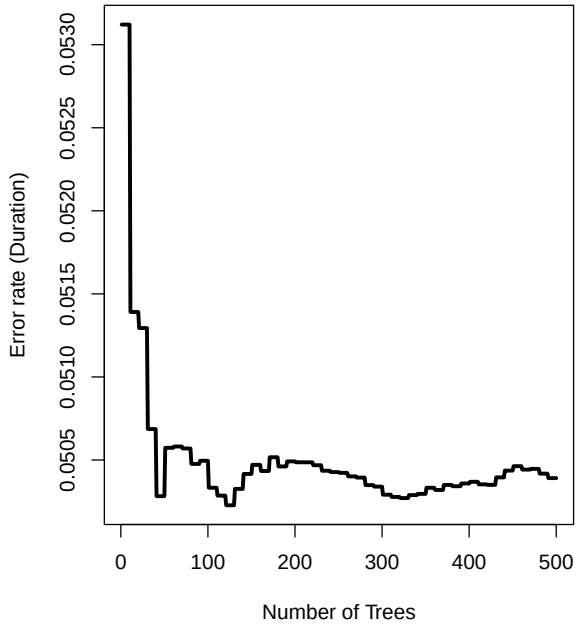


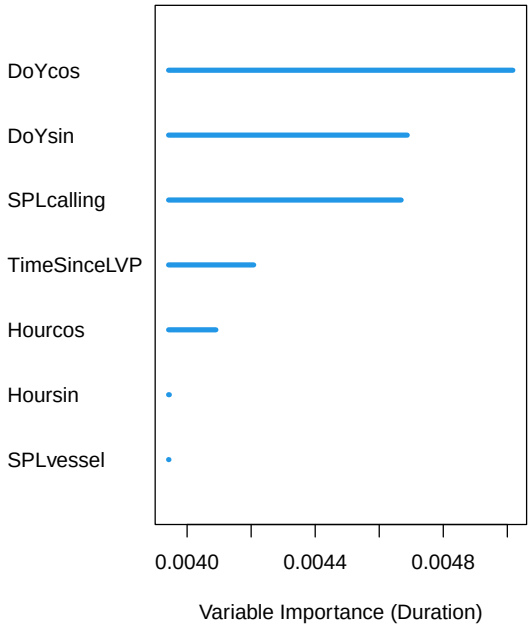

```
##
##                Importance   Relative Imp
## DoYcos           215.5817         1.0000
## DoYsin           195.7016         0.9078
## SPLcalling       174.0966         0.8076
## SPLvessel        152.7003         0.7083
## TimeSinceLVP      85.7646         0.3978
## Hoursin           83.6725         0.3881
## Hourcos           81.6397         0.3787
```



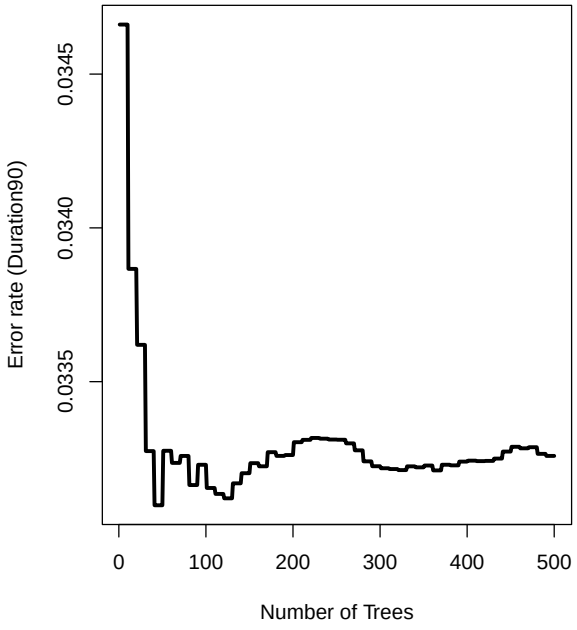


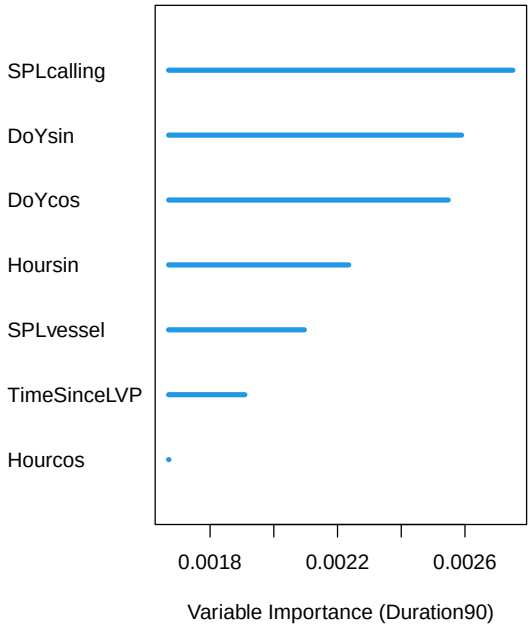

```
##
##                Importance   Relative Imp
## DoYcos           204.1240         1.0000
## SPLcalling       198.1901         0.9709
## DoYsin           192.3826         0.9425
## SPLvessel        174.6952         0.8558
## TimeSinceLVP     103.0384         0.5048
## Hourcos           96.9306         0.4749
## Hoursin           87.6267         0.4293
```






```
##
##                Importance   Relative Imp
## DoYcos             0.0050         1.0000
## DoYsin             0.0047         0.9343
## SPLcalling         0.0047         0.9305
## TimeSinceLVP       0.0042         0.8387
## Hourcos            0.0041         0.8152
## Hoursin            0.0039         0.7862
## SPLvessel          0.0039         0.7859
```

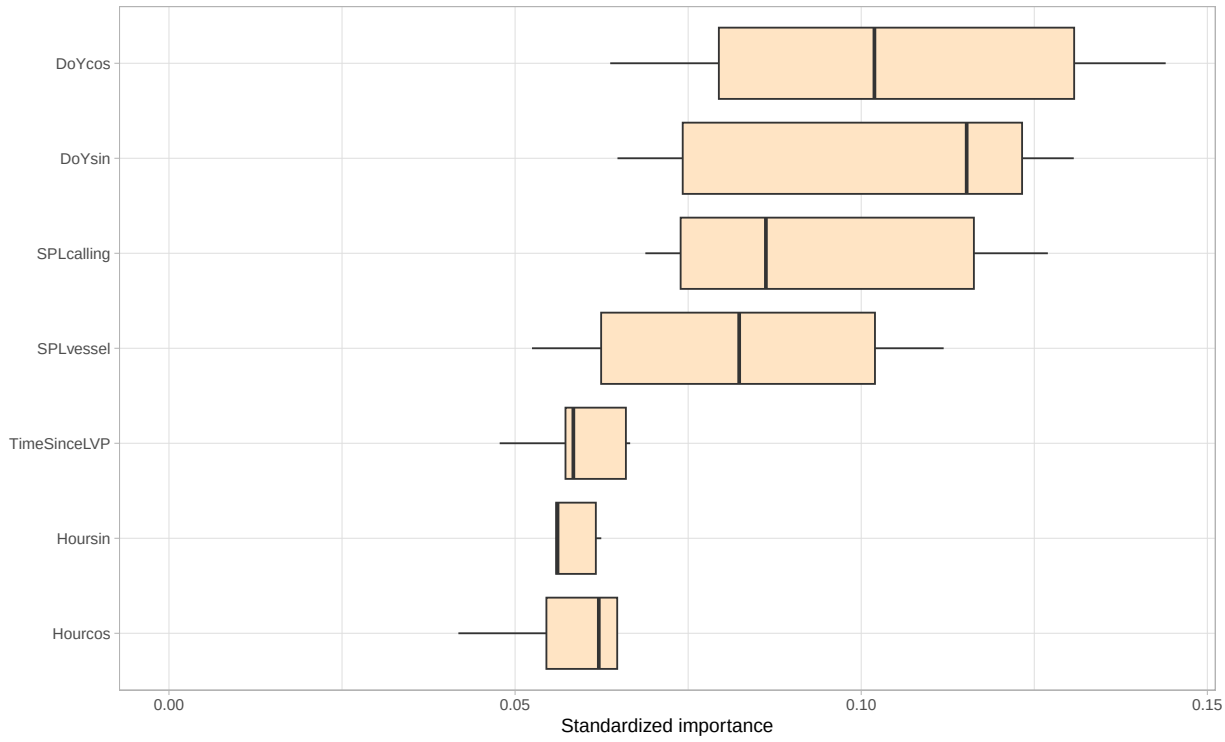





```
##
##                Importance   Relative Imp
## SPLcalling         0.0028         1.0000
## DoYsin             0.0026         0.9415
## DoYcos             0.0025         0.9262
## Hoursin            0.0022         0.8128
## SPLvessel          0.0021         0.7621
## TimeSinceLVP       0.0019         0.6942
## Hourcos            0.0017         0.6075
```

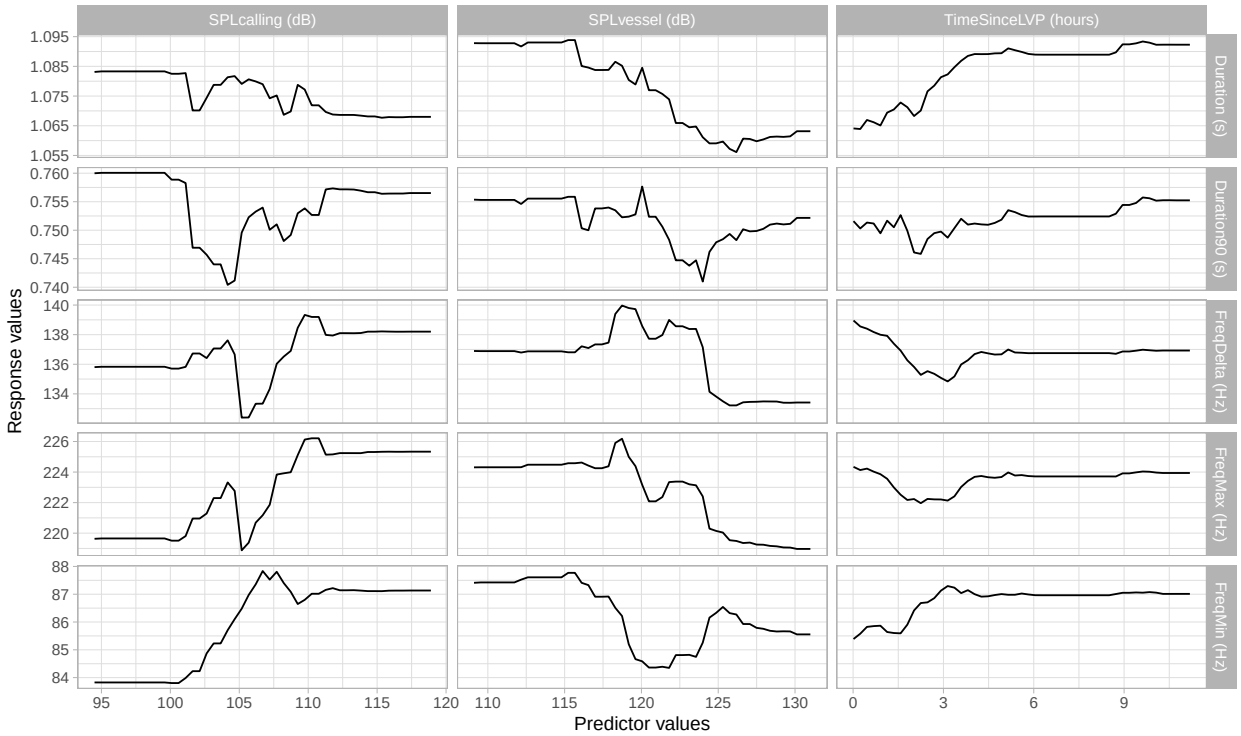
```r
# Extract standardized variable importance
svimp <- get.mv.vimp(mult_mod_rf, standardize = TRUE) %>%
    t() %>%
    as_tibble()

# Calculate average importance for sorting of the variables
svimp_avg <- apply(svimp, 2, mean) %>%
    sort()
svimp_long <- svimp %>%
    pivot_longer(cols = PREDICTORS_cycle,
                 names_to = "Variable",
                 values_to = "Value") %>%
    mutate(Variable = factor(Variable, levels = names(svimp_avg)))

# Plot importances
ggplot(svimp_long, aes(x = Value, y = Variable)) +
    xlim(0, max(svimp)) +
    geom_boxplot(fill = "bisque") +
    xlab("Standardized importance") +
```

```
    ylab("")
```



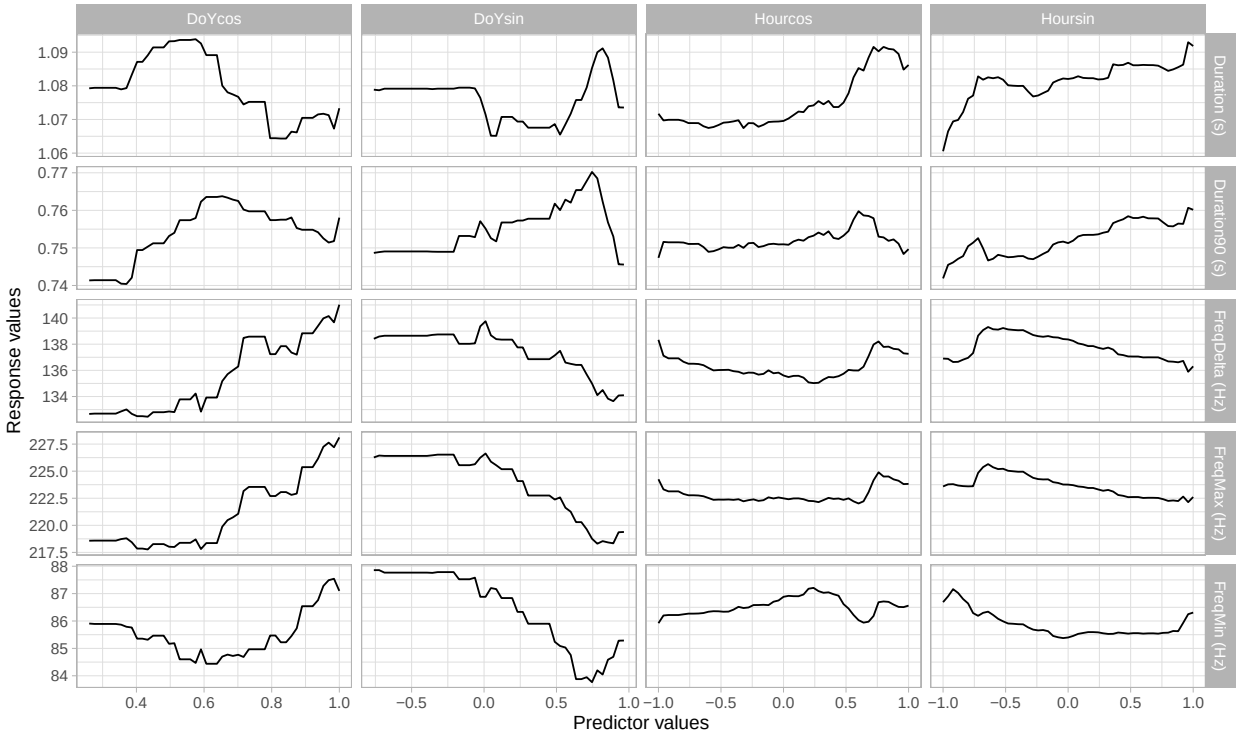

```
ggsave("images/importance_CallChar.png", width = 8, height = 5, dpi = 600)


# Get PDP data
pdp_CallChar <- lapply(PREDICTORS_cycle, function(i) { # i = "TimeSinceLVP"

    # Setup x-grid similar to pdp::partial and pdp:::pred_grid
    grid.resolution <- min(length(unique(mult_mod_rf$xvar[, i, drop = TRUE])), 51)
    xx <- seq(from = min(mult_mod_rf$xvar[, i, drop = TRUE], na.rm = TRUE),
              to = max(mult_mod_rf$xvar[, i, drop = TRUE], na.rm = TRUE),
              length = grid.resolution)

    # Use new partial.values xx instead of the mult_mod_rf$xvar[, i]
    partial.obj <- randomForestSRC::partial(mult_mod_rf,
                                            partial.xvar = i,
                                            partial.values = xx)

    pdta <- lapply(RESPONSES, function(j)
        randomForestSRC::get.partial.plot.data(partial.obj, m.target = j))

    lapply(1:length(RESPONSES), function(j)
        tibble(x = pdta[[j]]$x,
               yhat = pdta[[j]]$yhat,
               predictor = i,
               response = RESPONSES[j])) %>%
        bind_rows()
}) %>%
```

```
    bind_rows()

# Rename stuff to include units
pdp_CallChar <- pdp_CallChar %>%
    mutate(predictor = gsub("TimeSinceLVP", "TimeSinceLVP (hours)", predictor)) %>%
    mutate(predictor = gsub("SPLcalling", "SPLcalling (dB)", predictor)) %>%
    mutate(predictor = gsub("SPLvessel", "SPLvessel (dB)", predictor)) %>%
    mutate(response = if_else(grepl("Duration", response),
                              paste0(response, " (s)"),
                              response)) %>%
    mutate(response = gsub("FreqDelta", "FreqDelta (Hz)", response)) %>%
    mutate(response = gsub("FreqMax", "FreqMax (Hz)", response)) %>%
    mutate(response = gsub("FreqMin", "FreqMin (Hz)", response))


# Plot PDPs with fixed y-axes
pdp_CallChar %>%
    dplyr::filter(predictor %in% c("TimeSinceLVP (hours)", "SPLcalling (dB)", "SPLvessel (dB)")) %>%
    ggplot(aes(x = x, y = yhat)) +
    geom_line() +
    facet_grid(response ~ predictor, scales = "free") +
    labs(x = "Predictor values",
         y = "Response values")
```




```
ggsave("images/pdp_CallChar1.png", width = 9, height = 8, dpi = 600)


pdp_CallChar %>%
    dplyr::filter(predictor %!in% c("TimeSinceLVP (hours)", "SPLcalling (dB)", "SPLvessel (dB)")) %>%
```

```
    ggplot(aes(x = x, y = yhat)) +
    geom_line() +
    facet_grid(response ~ predictor, scales = "free") +
    labs(x = "Predictor values",
         y = "Response values")
```




```
ggsave("images/pdp_CallChar2.png", width = 9, height = 8, dpi = 600)
```

```
save.image("dataderived/DataImage_modeling_combined.RData")
```

```
# Citations ----

citation(package = "car", auto = TRUE)
```

```
## To cite package 'car' in publications use:
##
##   Fox J, Weisberg S, Price B (2023). _car: Companion to Applied
##   Regression_. R package version 3.1-2,
##   <https://CRAN.R-project.org/package=car>.
##
## A BibTeX entry for LaTeX users is
##
##   @Manual{,
##     title = {car: Companion to Applied Regression},
##     author = {John Fox and Sanford Weisberg and Brad Price},
```

```
##     year = {2023},
##     note = {R package version 3.1-2},
##     url = {https://CRAN.R-project.org/package=car},
##   }
```

```
citation(package = "flexmix")
```

```
## To cite package 'flexmix' in publications use:
##
##   Gruen B, Leisch F (2023). _flexmix: Flexible Mixture Modeling_. R
##   package version 2.3-19, <https://CRAN.R-project.org/package=flexmix>.
##
##   Leisch F (2004). "FlexMix: A General Framework for Finite Mixture
##   Models and Latent Class Regression in R." _Journal of Statistical
##   Software_, *11*(8), 1-18. doi:10.18637/jss.v011.i08
##   <https://doi.org/10.18637/jss.v011.i08>.
##
##   Grün B, Leisch F (2007). "Fitting Finite Mixtures of Generalized
##   Linear Regressions in R." _Computational Statistics & Data Analysis_,
##   *51*(11), 5247-5252. doi:10.1016/j.csda.2006.08.014
##   <https://doi.org/10.1016/j.csda.2006.08.014>.
##
##   Grün B, Leisch F (2008). "FlexMix Version 2: Finite Mixtures with
##   Concomitant Variables and Varying and Constant Parameters." _Journal
##   of Statistical Software_, *28*(4), 1-35. doi:10.18637/jss.v028.i04
##   <https://doi.org/10.18637/jss.v028.i04>.
##
## To see these entries in BibTeX format, use 'print(<citation>,
## bibtex=TRUE)', 'toBibtex(.)', or set
## 'options(citation.bibtex.max=999)'.
```

```
citation(package = "nlme")
```

```
## To cite package 'nlme' in publications use:
##
##   Pinheiro J, Bates D, R Core Team (2023). _nlme: Linear and Nonlinear
##   Mixed Effects Models_. R package version 3.1-164,
##   <https://CRAN.R-project.org/package=nlme>.
##
##   Pinheiro JC, Bates DM (2000). _Mixed-Effects Models in S and S-PLUS_.
##   Springer, New York. doi:10.1007/b98882
##   <https://doi.org/10.1007/b98882>.
##
## To see these entries in BibTeX format, use 'print(<citation>,
## bibtex=TRUE)', 'toBibtex(.)', or set
## 'options(citation.bibtex.max=999)'.
```

```
citation(package = "randomForestSRC", auto = TRUE)
```

```
## To cite package 'randomForestSRC' in publications use:
##
##   Ishwaran H, Kogalur UB (2024). _randomForestSRC: Fast Unified Random
```

```
##   Forests for Survival, Regression, and Classification (RF-SRC)_. R
##   package version 3.3.0,
##   <https://CRAN.R-project.org/package=randomForestSRC>.
##
## A BibTeX entry for LaTeX users is
##
##   @Manual{,
##     title = {randomForestSRC: Fast Unified Random Forests for Survival, Regression, and
## Classification (RF-SRC)},
##     author = {Hemant Ishwaran and Udaya B. Kogalur},
##     year = {2024},
##     note = {R package version 3.3.0},
##     url = {https://CRAN.R-project.org/package=randomForestSRC},
##   }
##
## ATTENTION: This citation information has been auto-generated from the
## package DESCRIPTION file and may need manual editing, see
## 'help("citation")'.
```

```
citation(package = "randomForest", auto = TRUE)
```

```
## To cite package 'randomForest' in publications use:
##
##   Breiman FobL, Cutler A, Liaw RpbA, Wiener. M (2022). _randomForest:
##   Breiman and Cutler's Random Forests for Classification and
##   Regression_. R package version 4.7-1.1,
##   <https://CRAN.R-project.org/package=randomForest>.
##
## A BibTeX entry for LaTeX users is
##
##   @Manual{,
##     title = {randomForest: Breiman and Cutler's Random Forests for Classification and
## Regression},
##     author = {Fortran original by Leo Breiman and Adele Cutler and R port by Andy Liaw and Matthew W
##     year = {2022},
##     note = {R package version 4.7-1.1},
##     url = {https://CRAN.R-project.org/package=randomForest},
##   }
##
## ATTENTION: This citation information has been auto-generated from the
## package DESCRIPTION file and may need manual editing, see
## 'help("citation")'.
```