Danielle Fradet

2023-12-04

```
###LOADING PACKAGES###
#library(seewave)
#library(tuneR)

###SET WORKING DIRECTORY AND READ IN FILES###

#setwd("/Volumes/LaCie/Original Audio Files/APP_Good_Original_Audio_Files")

#file_names <- list.files(path = ".", pattern = "*.wav", full.names = TRUE,
recursive = TRUE) ###was originally .WAV #recursive=look at all sub folders
in this folder #"." neutralizes folder path?


###RANDOMLY SAMPLE FILES IF NECESSARY###

#file_names <- sample(x = file_names, size = 1500, replace = FALSE) #replace
= False means that it will not replace that audio file name in the pool so it
cannot be chosen again in the sample


###CLEARING OBJECTS TO MAKE VECTOR MEMORY AVAILABLE AND CLEAR UNUSED
MEMORY###
#rm()
#gc()

###DEFINING VARAIBLES###
#segment_duration <- 10 #seconds
#sampling_rate <- 24000 #hertz
#percent_clip <- .95 #what percentage of the max/min clip you want as your
defining bound for clipping (i.e. >= .99 percent of max/min); only get very
few values that actually record as true max/min ([32767, -32768]) despite
many samples appearing as clipped in clipped audio files- 99% captures those
ones
#max_point <- 1 #16 bit file in r ranges from in r is [32767, -32768];
floating point format max is 1
#min_point <- -1 #16 bit file in r ranges from in r is [32767, -32768];
floating point format maz is -1


##########READING IN EACH FILE FOR SUBSEQUENT TASKS##########################

###READING IN FILE LOOP AND FILTERING IF NECESSARY###
#for (file_name in file_names)

#{
```

```
  # Read each wav file
  #wav <- readWave(file_name)

  # Chopping off file name portion so you dont get .wav.wav in the new file
  #short_name <- tools::file_path_sans_ext(file_name) ##the double colon says
pull specifically from this package

  # Extract file names after the last "/"
  #short_name <- basename(short_name) #need to have this code or else it will
preserve file path and will not allow you to change your working directory in
writing the csv file and will just store it to that specific audio file's
file path

  # Apply high pass filter, 250 is lower limit here. A low pass filter can
also be made using a second function
  #wav <- bwfilter(wav, f=24000, to =250, bandpass = NULL) ###if want a low
pass filter make bandpass TRUE


  # Write new filtered wav file
  #writeWave(wav, filename = paste0(short_name,"_filtered250.wav"))

  ###FINDING CLIPPING###

  # Convert from 16 bit format to floating -1 to 1
  #wav <- (wav/32768) #16 bit file in r ranges from in r is [32767, -32768]

  # Defining maximum clipped threshold
  #max_clip <- max_point * percent_clip

  # Create a new object based on the max threshold
  #max_column <- ifelse(wav@left >= max_clip, 1, 0) #assign 1 if clipped at
max, 0 if not clipped at max, for each sample

  # Create a data frame with the original vector and max object as new column
  #df <- data.frame(vector = wav@left, label = max_column)

  # Defining minimum clipped threshold
  #min_clip <- min_point * percent_clip

  # Create a new object based on min threshold
  #min_column <- ifelse(wav@left <= min_clip, 1, 0) #assign 1 if clipped at
min, 0 if not clipped at min, for each sample #replace wav with filtered.wav
if filtering

  # Create a data frame with the original vector and the new columns
  #df.max <- data.frame(vector = wav@left, label = max_column)
  #df.min <- data.frame(vector = wav@left, label = min_column)
```

```
  #df <- cbind(df.max, df.min)

  # Renaming column names in df data frame
  #colnames(df) <- c("sample_value", "max_clipped", "sample_value_2",
"min_clipped")
  #df <- subset(df, select = -sample_value_2) #columns sample_value and
sample_value_2 are the same, removing sample_value_2 from data.frame
  #max_min_clipped <- ifelse(df$max_clipped == 1 | df$min_clipped == 1, 1, 0)
#assign 1 if clipped at min, 0 if not clipped at min, for each sample; (|
means "or")
  #df$clipped <- max_min_clipped #adding clipped column to data frame

  ###CALCULATING PROPORTION OF SAMPLES CLIPPED IN A SEGMENT###

  #clipped <- as.vector(df$clipped) #need to be vector instead of data.frame,
or else get errors

  # Calculate the number of groups to get proportions for (should equal the
number of segments generated for the RMS loop) by dividing by desired number
of frames per group
  #num_groups <- (length(clipped) / (sampling_rate*segment_duration)) #groups
defined by 10 second segment duration at 24000 samples a second #length is to
get the number of rows, or number of values in the vector

  # Create empty vector to store the results
  #results <- numeric(num_groups)

  # Loop through the groups to calculate proportion of samples clipped in
each group
  #prop_segment_clipped <- c()
  #for (i in 1:num_groups) {

    # Calculate starting and ending position of each group along length
    #start_position <- (i - 1) * (sampling_rate*segment_duration) + 1 #moving
up and down the groups, so +1 moves you to the next group, -1 just pulls you
to the very beginning of that group
    #end_position <- min(i * (sampling_rate*segment_duration),
length(clipped))

    # Subset the vector for each group and sum the sample values coded as
clipped (1) or not clipped (0)
    #clipped_sum <-results[i] <- sum(clipped[start_position:end_position])
##Error in max_min_clipped[start_row:end_row, ] : incorrect number of
dimensions, attempted to fix by removing "," and that worked

    #Calculate proportion of the segment that is clipped
    #segment_prop <- clipped_sum/(sampling_rate*segment_duration) #number of
samples clipped/total number samples in segment
```

```
    # Print results
    #prop_segment_clipped <- c(prop_segment_clipped, segment_prop)

  #} #end of proportion segment clipped loop


  ###RMS Amplitude Extraction####

  # Create empty vector for RMS Amplitude calculation loop
  #rmspower <- c()

  # Calculate number of segments for each audio file, based on file duration
and time window of segments
  #num_segments <- floor(duration(wav) / segment_duration)

  # Loop through each segment

  #for (i in 1:num_segments)
  #{
    # Start and end time of each segment
    #start_time <- (i - 1)*segment_duration
    #end_time <- i * segment_duration

    # Calculate the segment length and location within audio file
    #segment <-
wav[round(start_time*sampling_rate):round(end_time*sampling_rate)]

    # Take rms measurement of converted floating point segments
    #rms_power <- rms(segment@left)

    # Convert to decibel scale (10*log) make relative to loudest possible
signal (1)
    #rel_rmspower <- 10*log((rms_power/1),base=10)

    # Save RMSpower relative measurements
    #rmspower <- c(rmspower, rel_rmspower)

  #}  #end of RMS loop

  ###COMBINING RELATIVE_RMS, CLIPPED DATA, & FILE NAME INTO DATAFRAME ###
  #rms_clip <- cbind(short_name, prop_segment_clipped, rmspower)

  ###WRITE CSV CONTAINING ORIGINAL FILE NAME###
  #write.csv(rms_clip, file = paste0("/Volumes/LaCie/Original Audio
Files/Output/",short_name,"_floatRMS95clip10s.csv")) #what measurements and
what specification

#} #end of file loop
```

```
###if you get error saying could not read RIFF files, remove faulty
recordings from time windows outside of 11/27 to 2/27--those are the ones
causing problems and not allowing code to run correctly

#set working directory
#WD <- "/Volumes/LaCie/Danielle's Folder/Palmer_wind_data"
#setwd(WD)

#designate folder path and collect all wind data files
#folder_path <- WD
#file_names_wind <- list.files(path = folder_path, pattern = "*.txt",
full.names = FALSE)

# Initialize an empty list to store individual data frames
#wind_list <- list()

# Loop through each file and read data into a list
#for (file_name in file_names_wind)

#{

  #wind <- read.table(file_name, header = TRUE, sep = "\t")
  #wind_list[[file_name]] <- wind

#}

#combine all wind files into one object
#combined_wind <- do.call(rbind, wind_list) ###rbind will stack dataframes on
top of one another


#formatting a new column do be the date of the sample for joining purposes
#Date <- as.Date(combined_wind$Sample.DateTime)
#combined_wind$Form.Date <- Date ###adding date column to dataframe


# Convert "datetime" column to POSIXct format
#Time <- as.POSIXct(combined_wind$Sample.DateTime, format = "%Y/%m/%d
%H:%M:%S") ###had to change from %Y-%m-%d because my data was in /// form

# Extract only the time component
#combined_wind$Time <- format(Time, "%H:%M:%S")

#split time column so I have formatted hour and minute columns for joining
purposes
#split_columns <- strsplit(as.character(combined_wind$Time), ":") ###split by
the space, when you view the split column object, it splits by what is in the
"", if I did by : there would be multiple "" around all of the appropraite
```

```
segments
#combined_wind$Form.Hour <- sapply(split_columns, '[', 1) #the 1 is the
information from the first split
#combined_wind$Form.Minute <- sapply(split_columns, '[', 2) #the 2 is the
information from the second split

#set working directory to acoustic data
#WD <-("/Volumes/LaCie/Original Audio Files/APP_Output_floatingRMS95clip10s")
#setwd(WD)

# load the required package
#library(dplyr)
#library(tidyverse)

# Set the path to the folder containing CSV files
#folder_path_penguins <- ("/Volumes/LaCie/Original Audio
Files/APP_Output_floatingRMS95clip10s")

# Get a list of file names in the folder
#file_names_penguins <- list.files(path = folder_path_penguins, pattern =
"\\.csv", full.names = TRUE)

# Read and combine the CSV files into one object
#combined_penguins <- bind_rows(lapply(file_names_penguins, read.csv))

#split by _ in short_name column to get recorder, date, and hour columns
#combined_penguins <- combined_penguins %>%
  #separate(short_name, into = c("Recorder", "Date", "Hour"), sep = "_")

#format the date column to match the date format from the wind data for
joining purposes
#combined_penguins$Form.Date <- as.Date(as.character(combined_penguins$Date),
format = "%Y%m%d") #formatted date

#Use substr to extract the first two digits and create formatted hour column
for joining purposes
#combined_penguins$Form.Hour <- substr(combined_penguins$Hour, 1, 2) #in
order to use substr argument it needs to be in character format

#create a minute column for joining wind data--based off of the interval
column in the acoustic data--each interval is 10s long in a 5 minute
recording, so each set of 6 intervals belongs to a different minute; have to
add a minute column because wind data only goes to minute level
#combined_penguins$Form.Minute <- ifelse(combined_penguins$X >= 1 &
combined_penguins$X <= 6, 1, combined_penguins$X )
#combined_penguins$Form.Minute <- ifelse(combined_penguins$X >= 7 &
combined_penguins$X <= 12, 2, combined_penguins$Form.Minute) #need to do
Minute.test here now so it keeps the saved info from the minute 1 interval
conversion
```

```
#combined_penguins$Form.Minute <- ifelse(combined_penguins$X >= 13 &
combined_penguins$X <= 18, 3, combined_penguins$Form.Minute) #keep doing
minute.test because it will write over itself everytime with the newest info
#combined_penguins$Form.Minute <- ifelse(combined_penguins$X >= 19 &
combined_penguins$X <= 24, 4, combined_penguins$Form.Minute)
#combined_penguins$Form.Minute <- ifelse(combined_penguins$X >= 25 &
combined_penguins$X <= 30, 5, combined_penguins$Form.Minute)
#combined_penguins$Form.Minute <- sprintf("%02d",
combined_penguins$Form.Minute) #needed to add a 0 to the front to get it to
match the Minute format in the combined wind spreadsheet


####Joining Penguin and Wind Data

#pd_wind <- left_join(combined_penguins, combined_wind) #console tells me it
is joining by form.date, form.hour, and form.minute

#export wind and penguin data
#write.csv(pd_wind, file = paste0("/Volumes/LaCie/Danielle's
Folder/","pd_wind_MASTER_floatRMS95clip10s.csv"))

#######GRAPHS####

#load library
library(dplyr)

##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union

library(ggplot2)

#set working directory
setwd("/Volumes/LaCie/Danielle's Folder/")

#read in data
pd_wind <- read.csv("APP_20231128_pd_wind_MASTER_floatRMS95clip10s.csv")

#graph of rmspower by 2 min avg wind speed (m/s) with prop clipped fill
ggplot(pd_wind,
       aes(x = as.numeric(WS.Avg.2min), y = rmspower, color =
prop_segment_clipped, fill = prop_segment_clipped)) +
  geom_point() +
```

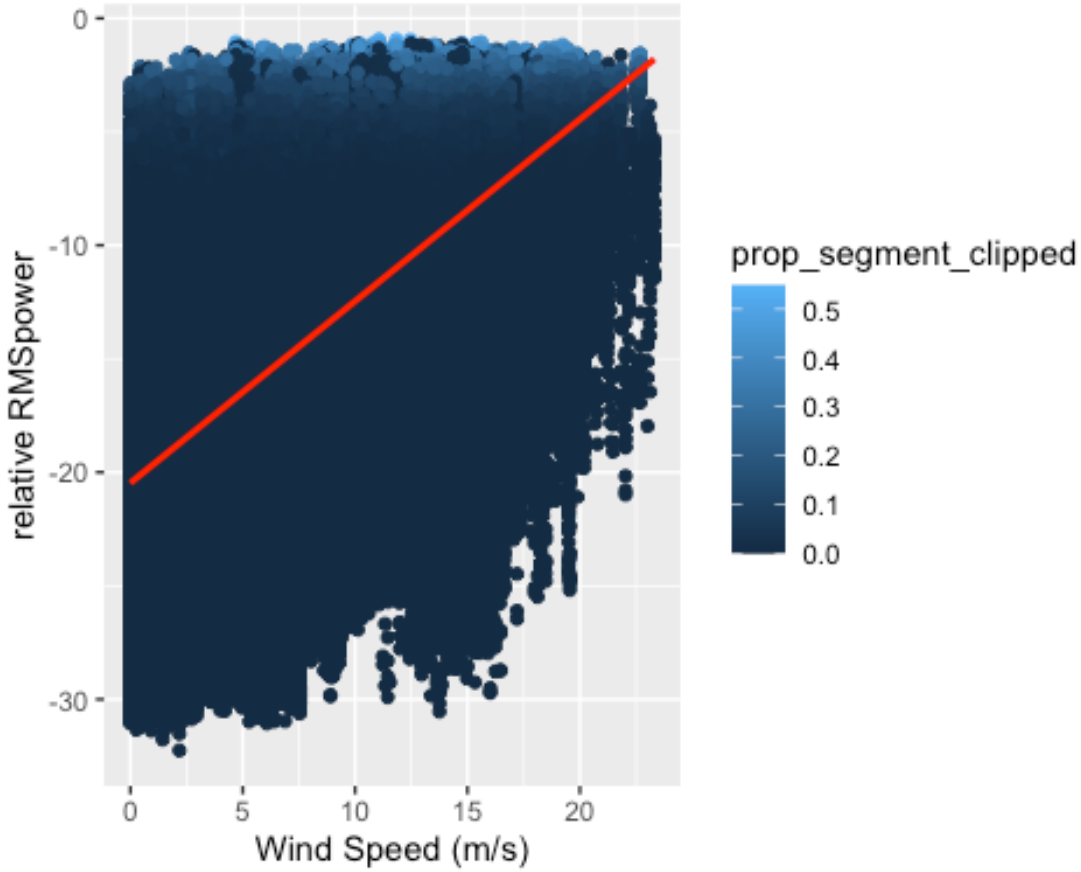
```
  geom_smooth(method = "lm", color = "red") + # used to add a regression line
  labs(x = "Wind Speed (m/s)", y = "relative RMSpower")
```

```
## `geom_smooth()` using formula = 'y ~ x'
```

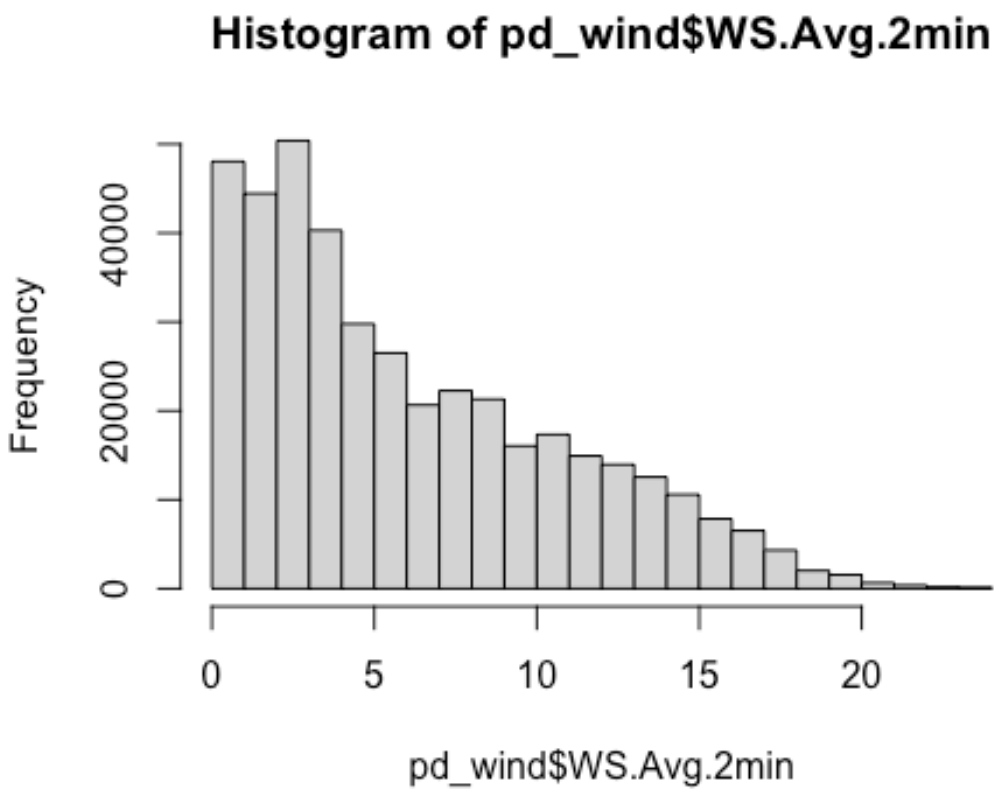
```
## Warning: Removed 5670 rows containing non-finite values (`stat_smooth()`).
```

```
## Warning: The following aesthetics were dropped during statistical
transformation: fill
## ℹ This can happen when ggplot fails to infer the correct grouping
structure in
##   the data.
## ℹ Did you forget to specify a `group` aesthetic or to convert a numerical
##   variable into a factor?
```

```
## Warning: Removed 5670 rows containing missing values (`geom_point()`).
```



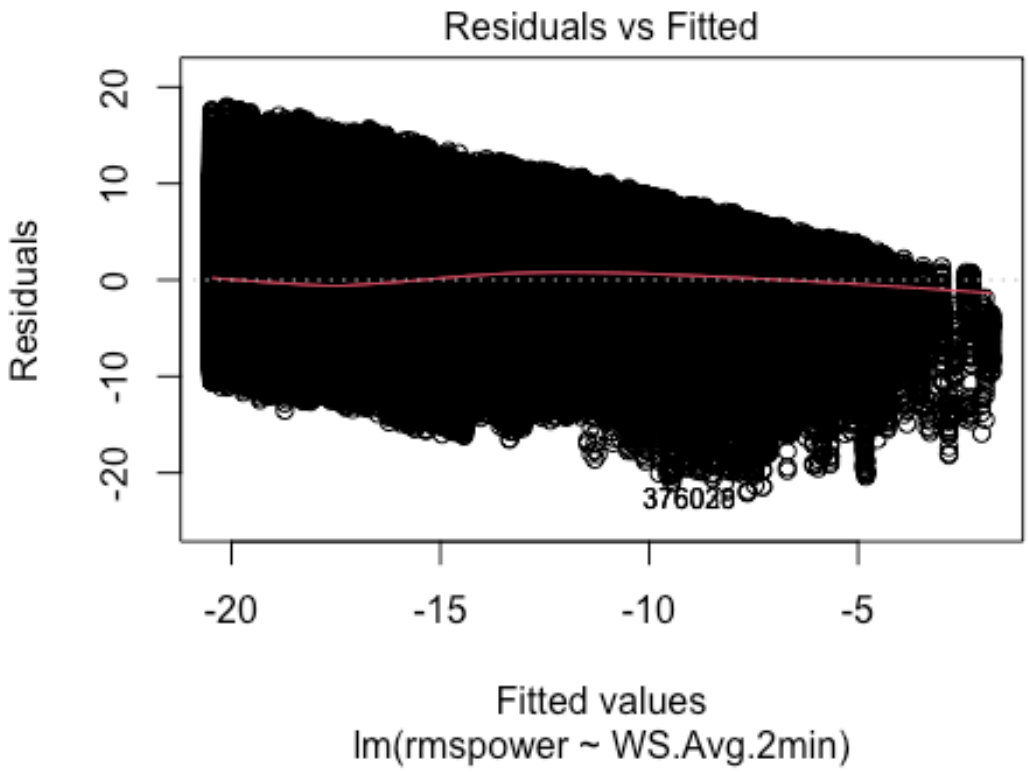

```
#linear regression model for the above plot
hist(pd_wind$WS.Avg.2min)
```

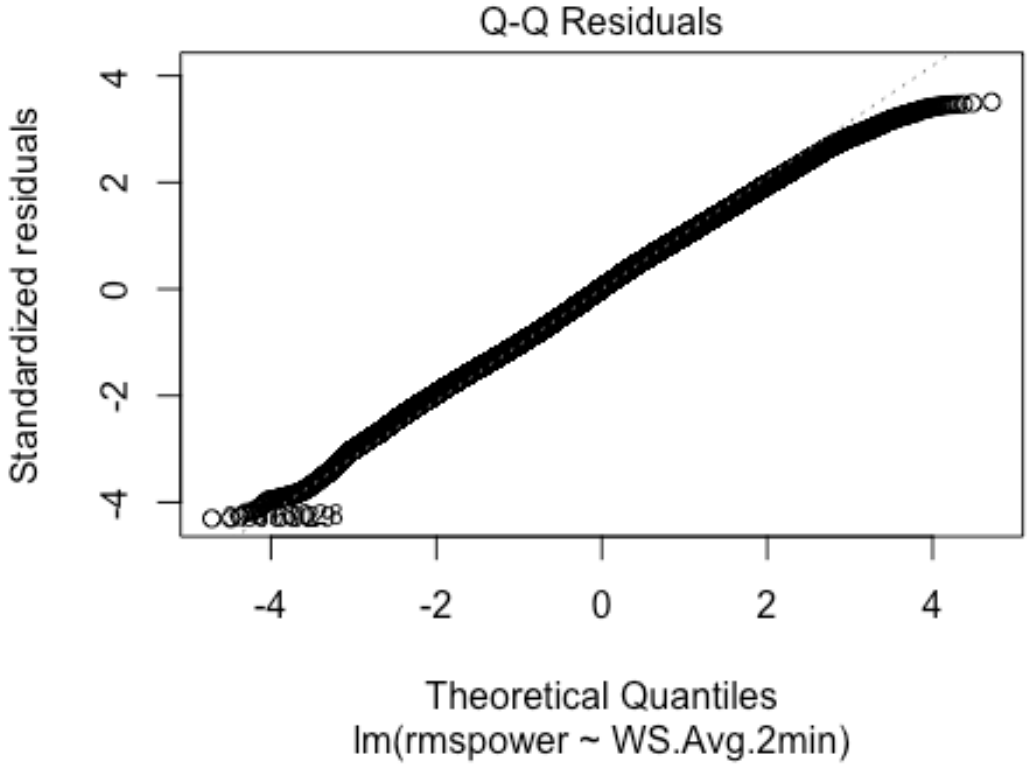


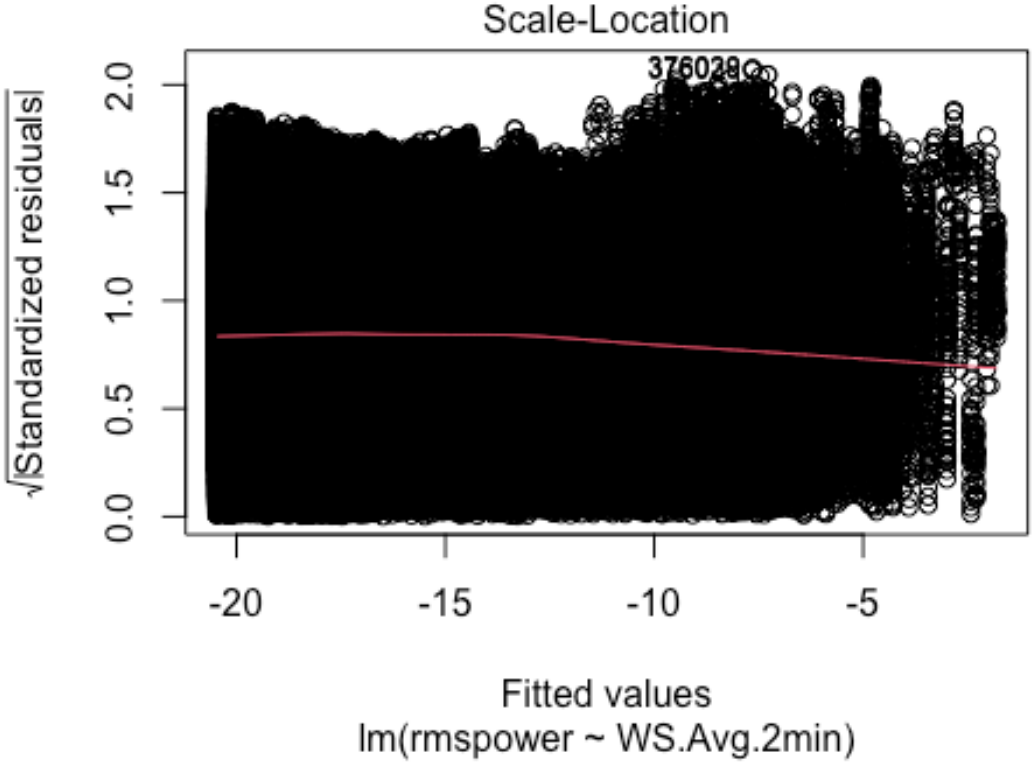

```
model_regression_rms_wind <- lm(rmspower~WS.Avg.2min, data = pd_wind)
summary(model_regression_rms_wind)
```

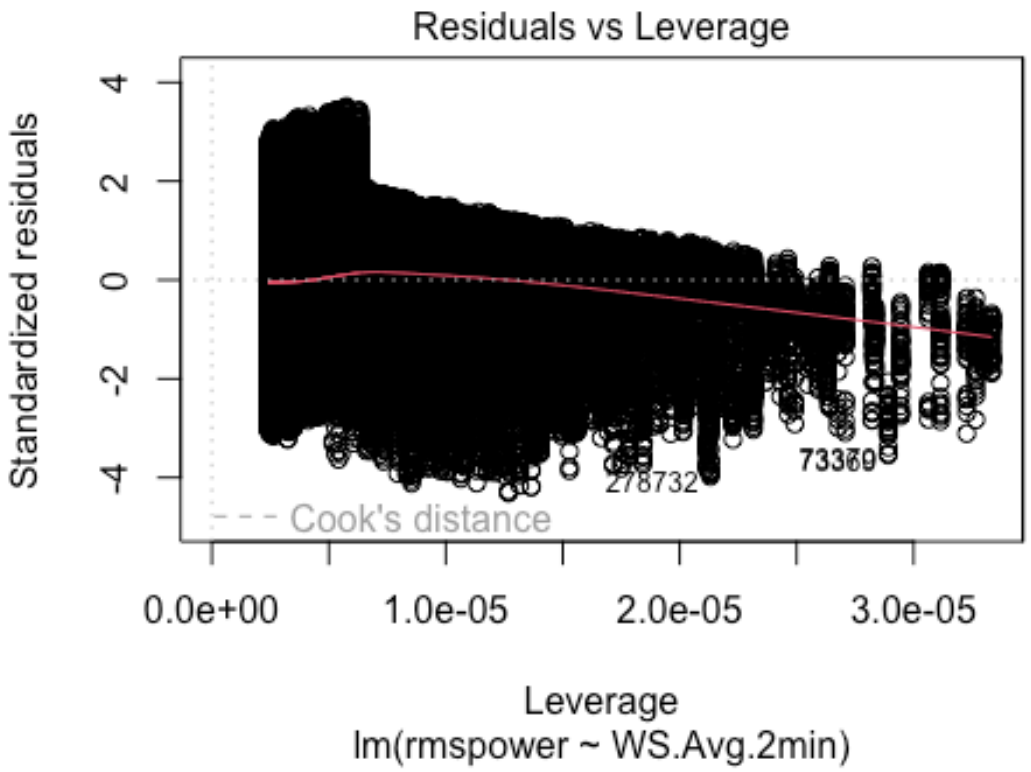
```
##
## Call:
## lm(formula = rmspower ~ WS.Avg.2min, data = pd_wind)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -22.0570  -3.6364   0.0825   3.6073  17.9899
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -20.470185   0.012853 -1592.7   <2e-16 ***
## WS.Avg.2min   0.800864   0.001654   484.1   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.123 on 412769 degrees of freedom
##   (5670 observations deleted due to missingness)
## Multiple R-squared:  0.3621, Adjusted R-squared:  0.3621
## F-statistic: 2.343e+05 on 1 and 412769 DF,  p-value: < 2.2e-16
```

```
plot(model_regression_rms_wind)
```


Residuals vs Fitted
Residuals
20
10
0
-10
-20
376026
-20
-15
-10
-5
Fitted values
lm(rmspower ~ WS.Avg.2min)


Q-Q Residuals
Standardized residuals
4
2
0
-2
-4
-4
-2
0
2
4
Theoretical Quantiles
lm(rmspower ~ WS.Avg.2min)


Scale-Location
376029
√|Standardized residuals|
0.0
0.5
1.0
1.5
2.0
-20
-15
-10
-5
Fitted values
lm(rmspower ~ WS.Avg.2min)

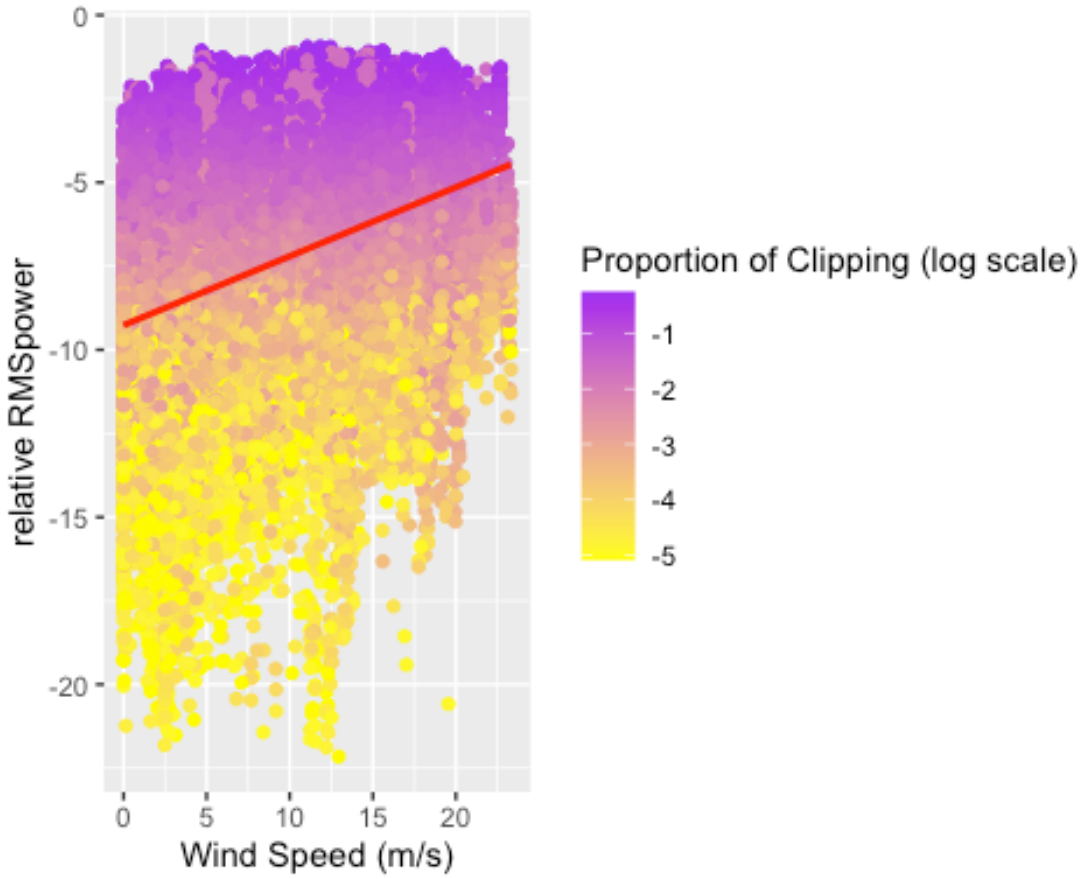



```
#graph of rmspower by 2 min avg wind speed (m/s) with fill of log 10
transformation of prop clipped with datapoints with no clipping filtered out
ggplot((pd_wind %>% filter(prop_segment_clipped >0.000005)),
       aes(x = as.numeric(WS.Avg.2min), y = rmspower, color =
log(prop_segment_clipped,10))) +
  scale_color_gradient(name = "Proportion of Clipping (log scale)", low =
"yellow", high = "purple") +
  geom_point() +
  geom_smooth(method = "lm", color = "red") + # used to add a regression line
  labs(x = "Wind Speed (m/s)", y = "relative RMSpower")
```

```
## `geom_smooth()` using formula = 'y ~ x'
```

```
## Warning: Removed 1042 rows containing non-finite values (`stat_smooth()`).
```

```
## Warning: Removed 1042 rows containing missing values (`geom_point()`).
```

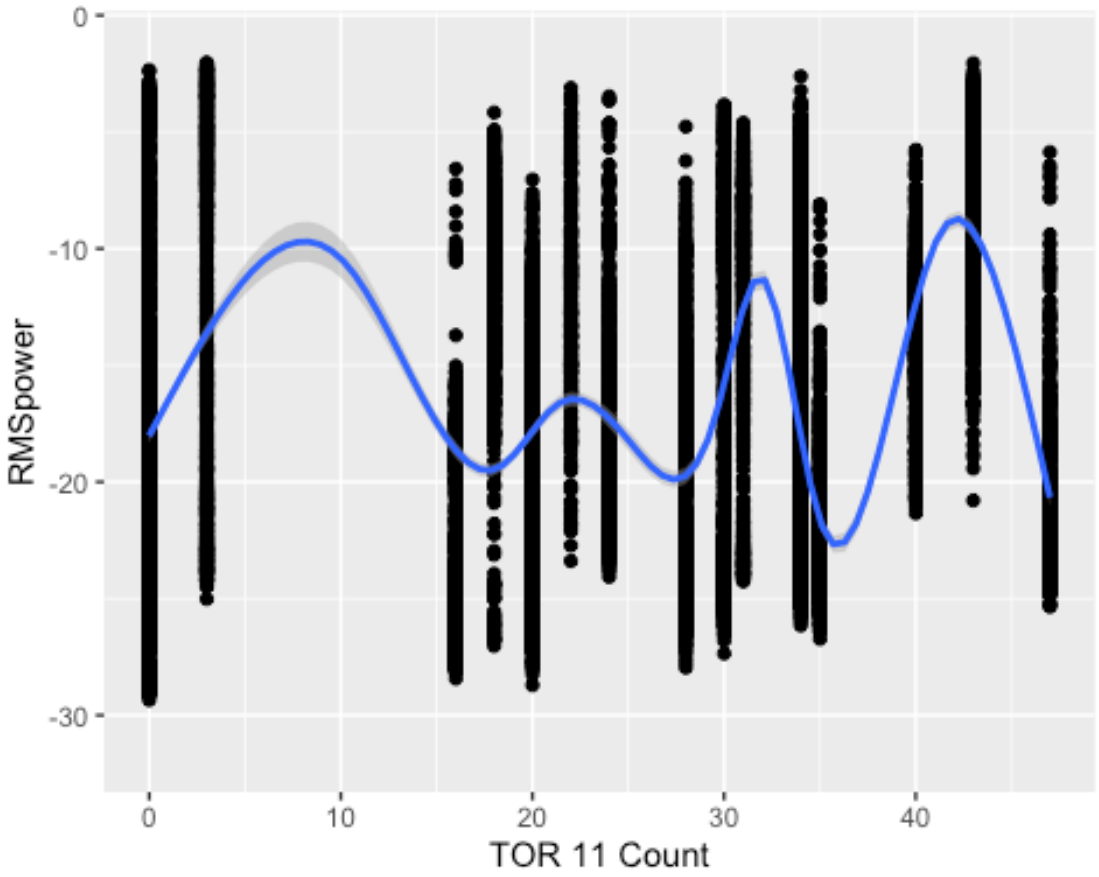



```
#wanted to log transform the prop segment clipped because many of the values
are very small and hard to distinguish from 0

###Joining pd_wind master file to biological count data###

#set working directory
#setwd("/Volumes/LaCie/Danielle's Folder/")

#read in data
#pd_wind <- read.csv("APP_20231128_pd_wind_MASTER_floatRMS95clip10s.csv")

#adding island column to dataset for joining purposes
#pd_wind$ISL <- ifelse(pd_wind$Recorder == "SMA08607" | pd_wind$Recorder ==
"SMA08411" | pd_wind$Recorder == "SMA08613" | pd_wind$Recorder == "SMA08601"
| pd_wind$Recorder =="SMA08612" | pd_wind$Recorder == "SMA08611" |
pd_wind$Recorder == "SMA08410" | pd_wind$Recorder == "SMA08604" |
pd_wind$Recorder == "SMA08614", "HUM", pd_wind$Recorder)
#pd_wind$ISL <- ifelse(pd_wind$Recorder == "SMA08609" | pd_wind$Recorder ==
"SMA08610" | pd_wind$Recorder == "SMA08608", "TOR", pd_wind$ISL)

#adding location column to dataset for joining purposes
#pd_wind$LOC <- ifelse(pd_wind$Recorder == "SMA08607" | pd_wind$Recorder ==
"SMA08411" | pd_wind$Recorder == "SMA08613" | pd_wind$Recorder == "SMA08601"
```

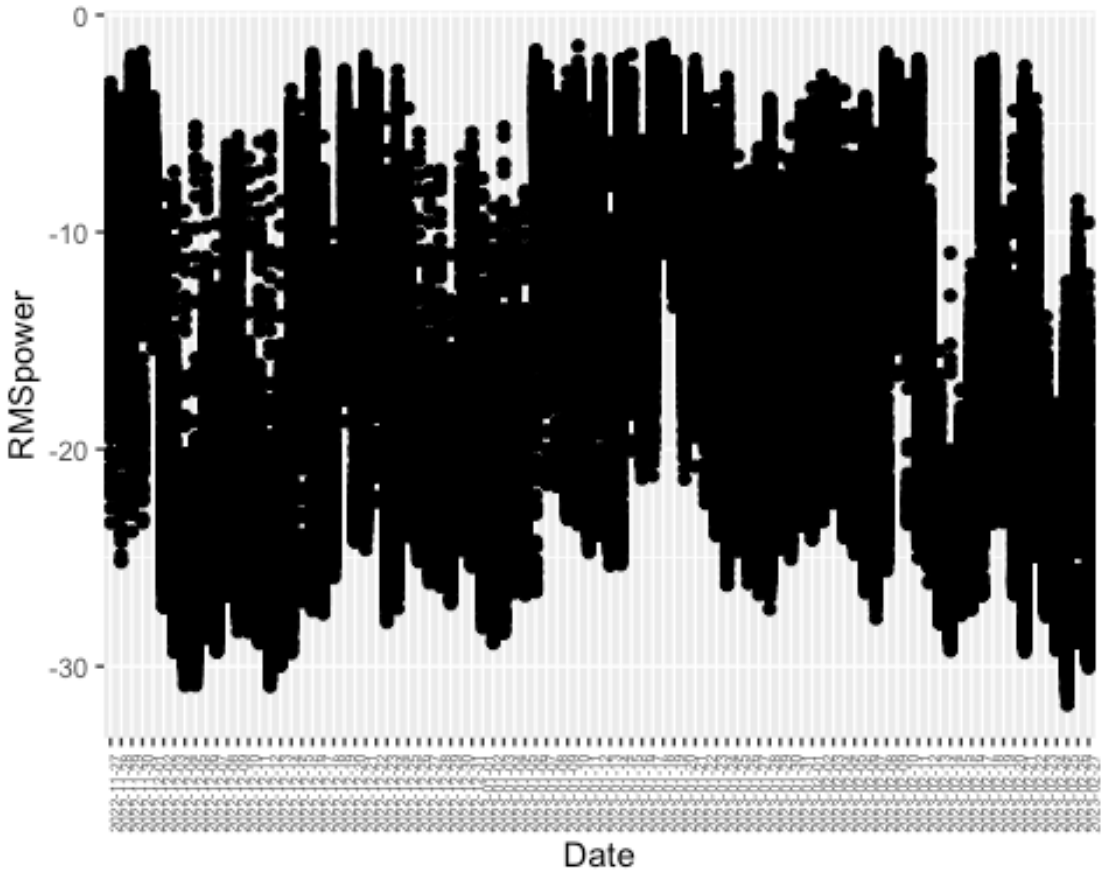
```
| pd_wind$Recorder =="SMA08612" | pd_wind$Recorder == "SMA08611" |
pd_wind$Recorder == "SMA08410" | pd_wind$Recorder == "SMA08604", "2.0" ,
pd_wind$Recorder)
#pd_wind$LOC <- ifelse(pd_wind$Recorder == "SMA08614", 3.0, pd_wind$LOC)
#pd_wind$LOC <- ifelse(pd_wind$Recorder == "SMA08609", 16.0, pd_wind$LOC)
#pd_wind$LOC <- ifelse(pd_wind$Recorder == "SMA08610", 11.0, pd_wind$LOC)
#pd_wind$LOC <- ifelse(pd_wind$Recorder == "SMA08608", 7.2, pd_wind$LOC)

#setting work directory to pull in count data
#setwd("/Volumes/LaCie/Field Datasheets")

#reading in indicator and census data to be joined
#indicator <- read.csv("Adelie_indicators_TOR_HUM.csv")
#census <- read.csv("Adelie_census_TOR_HUM.csv")

#formatting dates to match pd_wind for joining purposes
#Form.Date.I <- as.Date(indicator$DATE, format = "%m/%d/%Y")
#indicator$Form.Date <- format(Form.Date.I, "%Y-%m-%d")

#Form.Date.C <- as.Date(census$DATE, format = "%m/%d/%Y")
#census$Form.Date <- format(Form.Date.C, "%Y-%m-%d")

#removing TOR 2.0 and 3.0 from census data because has 0 penguins during
breeding season and no recorders placed there
#census <- subset(census, X != 45) #removing TOR 2.0 peak egg
#census <- subset(census, X != 46) #removing TOR 3.0 peak egg
#census <- subset(census, X != 118) #removing TOR 2.0 peak chick
#census <- subset(census, X != 119) #removing TOR 3.0 peak chick

#remove x's from indicator and census for joining purposes
#indicator <- subset(indicator, select = -X)
#census <- subset(census, select = -X)
#pd_wind <- subset(pd_wind, select = -X.1)

#remove date and season from indicator and census for joining purposes (date
is in pd_wind and want to match by form.date only, don't need season column)
#indicator <- subset(indicator, select = -SEASON)
#census<- subset(census, select = -SEASON)
#indicator <- subset(indicator, select = -DATE)
#census <- subset(census, select = -DATE)

#renaming specific column names for joining purposes
#library(dplyr)
#indicator <- indicator %>% rename(Adults_I = TOTADULTS, Chicks_I =
TOTCHICKS, Nests_I = ACTTERR, Notes_I = NOTES)
#census <- census %>% rename(Adults_C = ADULTS, Chicks_C = CHICKS, Nests_C =
NESTS, Notes_C = NOTES)
#pd_wind <- pd_wind %>% rename(segment = X)
```

```r
#adding penguin total columns
#indicator$Total_I <- indicator$Adults_I + indicator$Chicks_I
#census$Total_C <- census$Adults_C + census$Chicks_C

#filtering for just TOR 11 and converting everything to a numeric
#pd_wind_TOR11 <- filter(pd_wind, LOC == 11)
#indicator_TOR11 <- filter(indicator, LOC == 11.0)
#census_TOR11 <- filter(census, LOC == 11.0)
#numeric <- as.numeric(pd_wind_TOR11$LOC) #converting character to numeric
#pd_wind_TOR11$LOC <- numeric

#joining count data for TOR11
#TOR11_indicator_wind <-left_join(pd_wind_TOR11, indicator_TOR11) #indicator data (includes census numbers in the indicator data inherently) #can join by census later if I want, it will just only keep the data from the two census days

#export wind and penguin data
#write.csv(TOR11_indicator_wind, file = paste0("/Volumes/LaCie/Danielle's Folder/","TOR11_indicator_wind_MASTER_floatRMS95clip10s.csv"))

 #just keeps data with both census values present
#df_filled <- df %>%
  #fill(Value, .direction = "down") #direction down means it is taking the previous value and pulling it down---use this one --need the period in front of direction

#set working directory
setwd("/Volumes/LaCie/Danielle's Folder/")

#load packages for filtering and plotting
library(dplyr)
library(ggplot2)

TOR11_indicator_wind <-
read.csv("TOR11_indicator_wind_MASTER_floatRMS95clip10s.csv")

#scatterplot of rmspower by count
ggplot(TOR11_indicator_wind, aes(x = Total_I, y = rmspower)) +
  geom_point() +
  geom_smooth() +
  labs(x = "TOR 11 Count", y = "RMSpower")
```

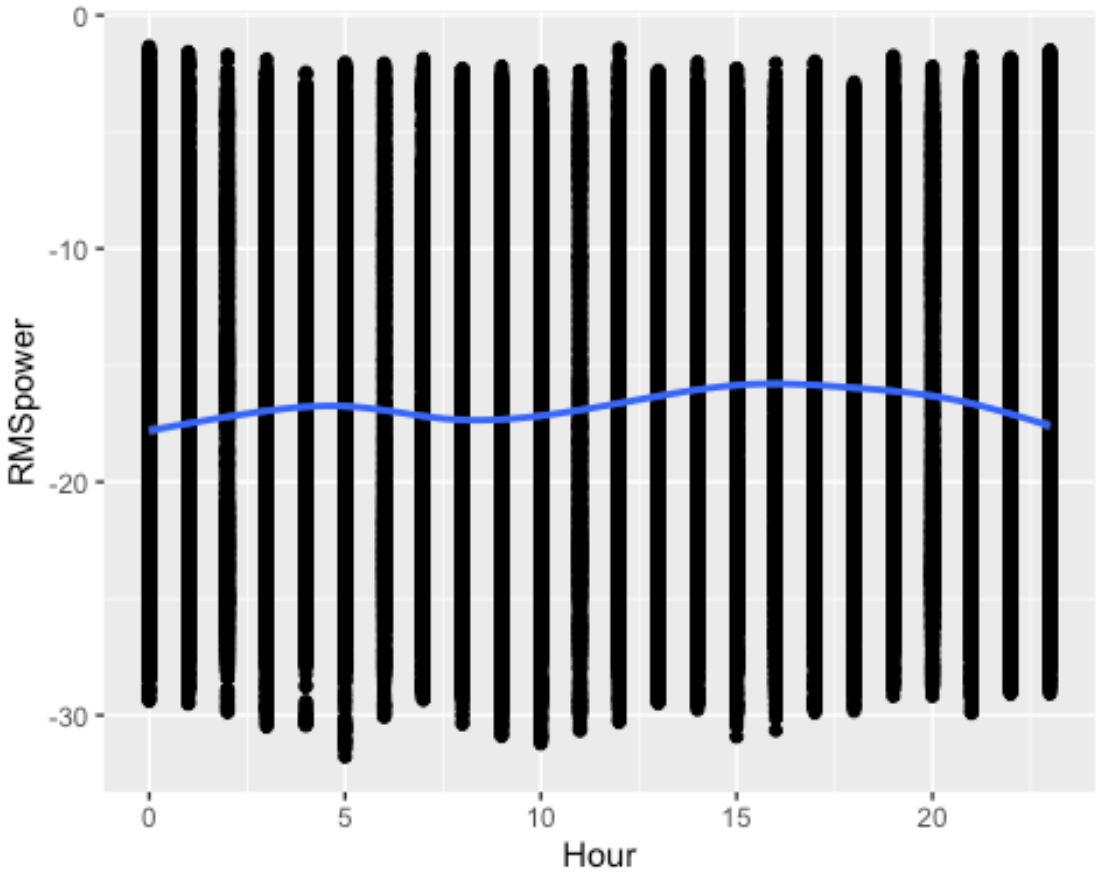
```
## `geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'

## Warning: Removed 53127 rows containing non-finite values
(`stat_smooth()`).

## Warning: Removed 53127 rows containing missing values (`geom_point()`).
```




```
#scatterplot of rmspower by date
ggplot(TOR11_indicator_wind, aes(x = Form.Date, y = rmspower)) +
  geom_point() +
  geom_smooth() +
  labs(x = "Date", y = "RMSpower") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1, size = 5))

## `geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```



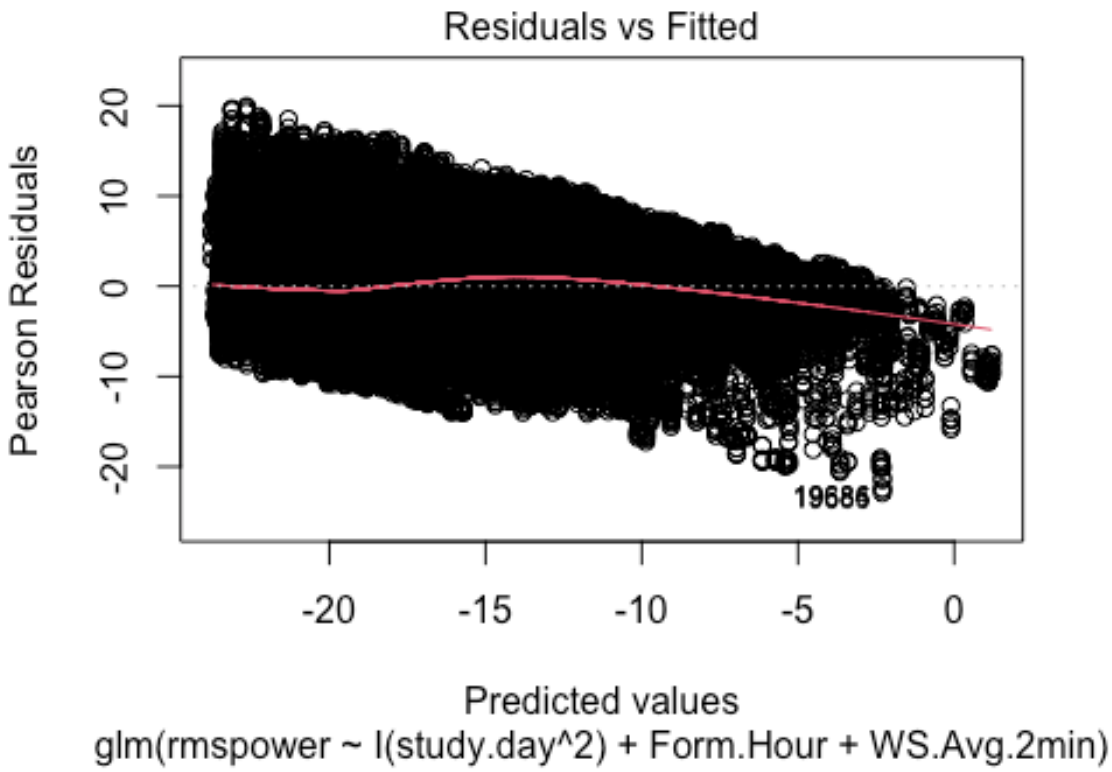

```
#scatterplot of rmspower by hour
ggplot(TOR11_indicator_wind, aes(x = Form.Hour, y = rmspower)) +
  geom_point() +
  geom_smooth() +
  labs(x = "Hour", y = "RMSpower")

## `geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
```

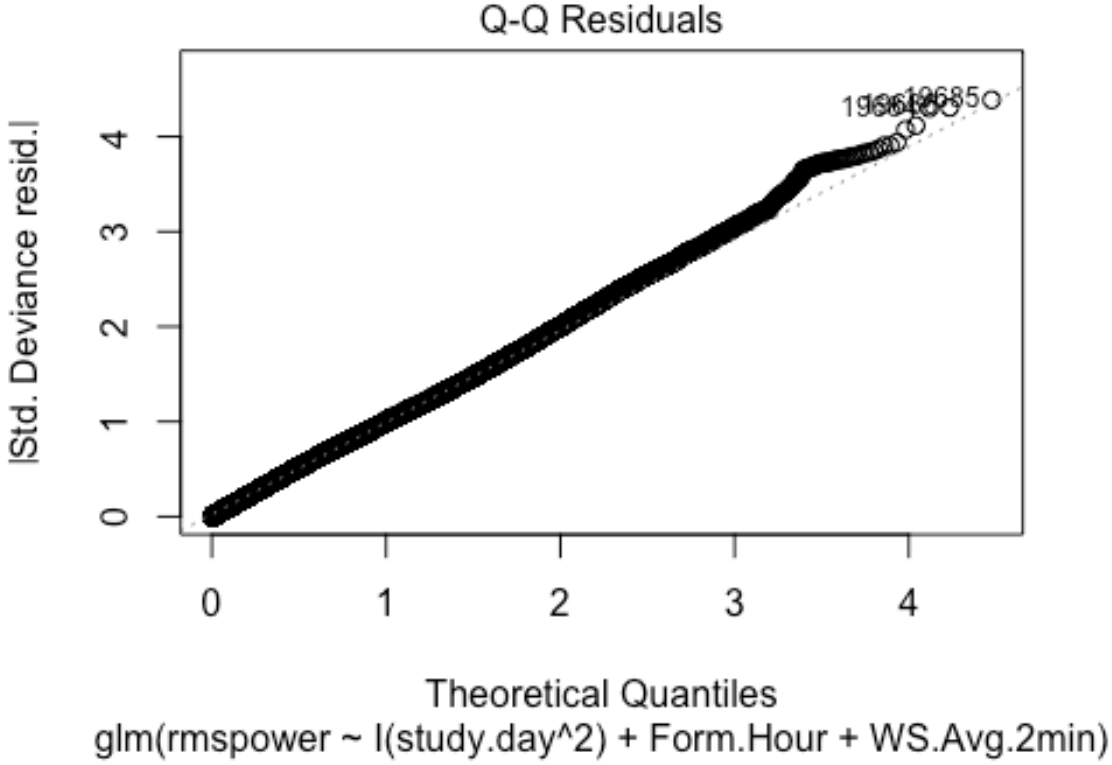


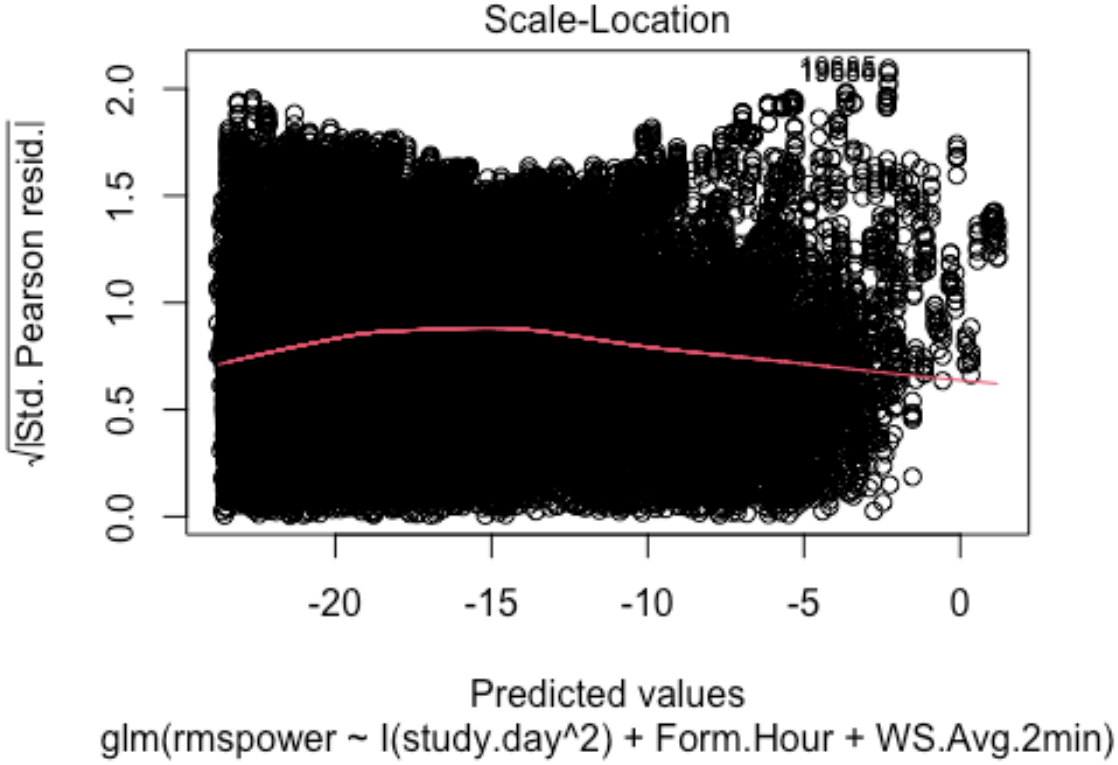

```
#transforming Form.Date from a character to a numeric
TOR11_indicator_wind$study.day <-
as.numeric(as.factor(TOR11_indicator_wind$Form.Date))

#GLM
super.model <- glm(data = TOR11_indicator_wind, rmspower~I(study.day^2) +
Form.Hour + WS.Avg.2min)
summary(super.model)

##
## Call:
## glm(formula = rmspower ~ I(study.day^2) + Form.Hour + WS.Avg.2min,
##     data = TOR11_indicator_wind)
##
## Coefficients:
##                   Estimate Std. Error  t value Pr(>|t|)
## (Intercept)    -2.308e+01  5.246e-02 -439.859  < 2e-16 ***
## I(study.day^2) -7.888e-05  7.841e-06  -10.059  < 2e-16 ***
## Form.Hour       2.317e-02  2.942e-03    7.877  3.4e-15 ***
## WS.Avg.2min     1.040e+00  4.228e-03  245.960  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

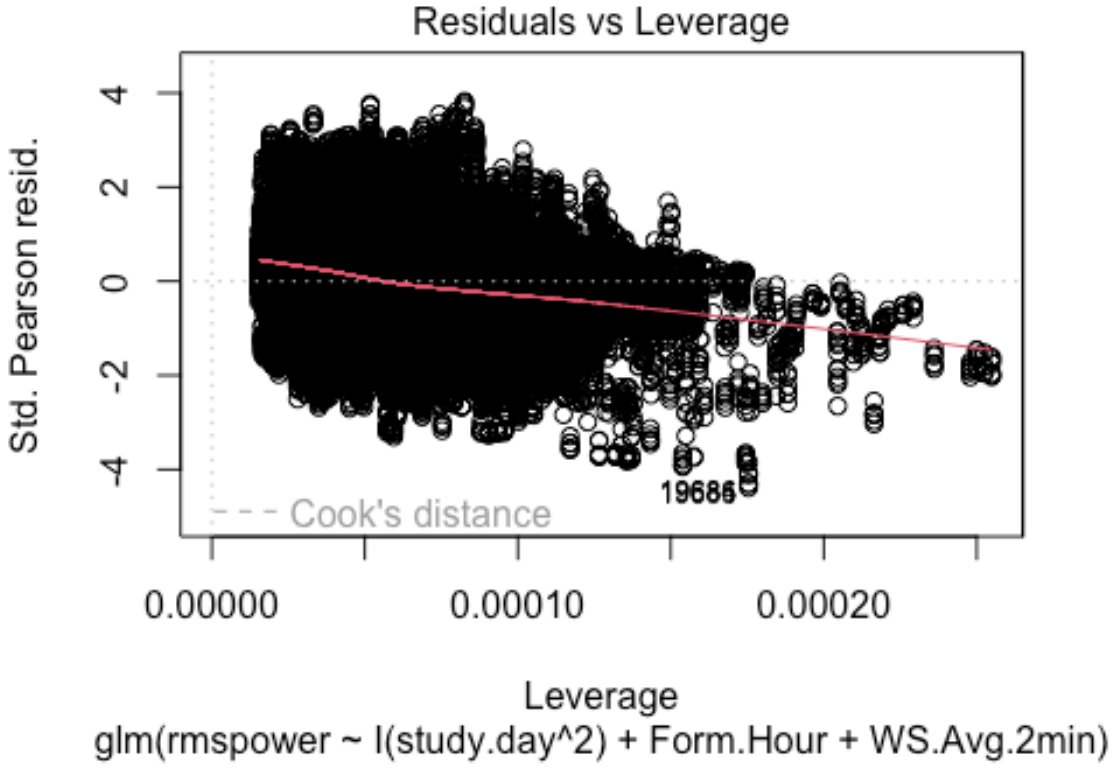
```
## (Dispersion parameter for gaussian family taken to be 27.20509)
##
##     Null deviance: 3443313  on 65636  degrees of freedom
## Residual deviance: 1785552  on 65633  degrees of freedom
##   (750 observations deleted due to missingness)
## AIC: 403101
##
## Number of Fisher Scoring iterations: 2
```

```
plot(super.model)
```


Residuals vs Fitted
Pearson Residuals
20
10
0
-10
-20
19686
-20
-15
-10
-5
0
Predicted values
glm(rmspower ~ I(study.day^2) + Form.Hour + WS.Avg.2min)


Q-Q Residuals
|Std. Deviance resid.|
0
1
2
3
4
Theoretical Quantiles
glm(rmspower ~ I(study.day^2) + Form.Hour + WS.Avg.2min)


Scale-Location
√|Std. Pearson resid.|
0.0
0.5
1.0
1.5
2.0
-20
-15
-10
-5
0
Predicted values
glm(rmspower ~ I(study.day^2) + Form.Hour + WS.Avg.2min)


Residuals vs Leverage
Std. Pearson resid.
4
2
0
-2
-4
19686
Cook's distance
0.00000
0.00010
0.00020
Leverage
glm(rmspower ~ I(study.day^2) + Form.Hour + WS.Avg.2min)