From Low Marsh to High Marsh: Understanding Carbon Flux in Coos Bay's Kunz Marsh

Kristine Bernabe

2025-12-10

Introduction

Salt marshes are dynamic coastal wetlands that play a crucial role in the global carbon cycle, both storing carbon and releasing greenhouse gases such as carbon dioxide (CO_2) and methane (CH_4). However, wetlands currently face two intersecting challenges. First, the concentration of atmospheric greenhouse gases has risen dramatically in recent history, leading to increasing temperatures, changing moisture regimes, and rising sea levels (Shahan et al., 2022). Of particular concern for wetlands is CH_4, which has a global warming potential roughly 30 times greater than CO_2 over a 100-year timescale (Muñoz et al., 2024). Second, despite their role as CH_4 sources, wetlands are highly effective carbon sinks and provide numerous other ecological benefits, like filtering water, serving as critical habitat for birds, and serving as nurseries for culturally and commercially relevant fish (Salimi et al., 2021). Yet wetlands continue to disappear due to urbanization, agriculture, and climate-driven sea-level rise (Salimi et al., 2021), increasing the urgency to understand how wetlands behave under current and future climate conditions.

Kunz Marsh, a restored tidal marsh within the South Slough National Estuarine Research Reserve (SSNERR) in Coos Bay, Oregon, is an ideal setting for investigating these dynamics. Before restoration in 2003, the area had been converted to pasture (Cornu & Sadro, 2002). During restoration, land managers intentionally graded the marsh into distinct elevation zones (low, mid, high) to emulate natural elevation gradients and evaluate how ecosystem recovery varies across elevations (Cornu & Sadro, 2002). Prior work in other marshes has demonstrated that greenhouse gas (GHG) fluxes differ based on vegetation, hydrology, and land-use history, but relatively few studies have examined these patterns in Pacific Northwest marshes or within restored systems across elevation gradients.

Previous research has explored GHG emissions from coastal wetlands. For example, Shahan et al. (2022) combined co-located chamber and eddy-covariance measurements in a restored marsh and found it to be a strong net CO_2 sink and a small CH_4 source, with vegetated zones removing more CO_2 than non-vegetated mudflats. In Oregon and Washington wetlands, Williams et al. (2025) identified elevation, water table depth, and salinity as key predictors of CH_4 flux. Additionally, Schultz et al. (2023) concluded that combining broad spatial chamber measurements with continuous environmental data can effectively model multiple greenhouse gases across wetlands with varying land-use regimes. Building on this research, my study contributes new chamber-based measurements of CO_2 and CH_4 from nine sites across all three elevation zones at Kunz Marsh, collected throughout the summer and early fall of 2025. Additional hourly chamber fluxes were collected over 24 hours at several high-marsh sites to study diurnal patterns. These fluxes are paired with environmental data from one of SSNERR's meteorological towers near the site, including photosynthetically active radiation (PAR).

Despite the current literature, gaps remain in our understanding of how GHG fluxes vary within individual marshes, across elevation gradients, and over different timescales, particularly in the Pacific Northwest. Wetlands are notoriously complex systems: their fluxes are influenced by interacting hydrologic, microbial, and plant processes that can be difficult to model accurately. Moreover, West Coast flux datasets are primarily from California and Washington, with limited representation from Oregon. Expanding site-specific measurements is especially important due to this high variability. By generating new flux data from Kunz Marsh, this project helps address regional data gaps. It contributes to ongoing efforts to characterize the behavior of greenhouse gases in Pacific Northwest coastal wetlands.

This study asks three primary questions: (1) How do CO_2 and CH_4 fluxes vary across the low, mid, and high elevation zones within Kunz Marsh? (2) How do CO_2 and CH_4 fluxes, across all elevations, change seasonally from mid-summer to early fall? and (3) How do CO_2 and CH_4 fluxes vary over 24 hours, particularly in the high marsh? I predict that higher elevations would generally exhibit greater emissions of both CO_2 and CH_4. Over a 24-hour cycle, I expect CH_4 emissions to increase through the afternoon and decline overnight, while CO_2 alternates between daytime uptake and nighttime release. Seasonally, I predict greater CH_4 emissions during mid-summer with a decline into early fall, and the most substantial CO_2 uptake occurring during mid-summer. Overall, this study aims to improve our understanding of GHG flux dynamics in a restored Pacific Northwest salt marsh.

Data

Each sampling day resulted in a raw data file (.dat) retrieved from the LI-COR 7810 gas analyzer. Measurements are recorded anytime the gas analyzer is on. Each measurement is logged every second. Each raw data file contains the following components:

File header

- Model: The model of the instrument
- SN: The serial number of the instrument
- Software Version: The software version on the instrument
- Timestamp: The date and time of the beginning of the requested data (according to the instrument clock; yyyy-mm-dd hh:mm:ss)

- Timezone: The timezone setting on the instrument when the data is requested

Data header

- SECONDS: (secs) Seconds past the universal epoch (Unix time)
- NANOSECONDS: (nsecs) Nanoseconds of the seconds
- NDX: (index) A count of scans. At four scans per second, the value increases by four counts per second
- DIAG: (diag) Diagnostic code (see Status codes)
- REMARK: (-) The remark entered in the Remark field
- DATE: (date) Date of the record in yyyy-mm-dd
- TIME: (time) Time of the record in HH:MM:SS (according to the instrument clock)
- H2O: (ppm) Water vapor concentration
- CO2: (ppm) Carbon dioxide mole fraction in dry air
- CH4: (ppb) Methane mole fraction in dry air
- CAVITY_P: (kPa) Optical cavity pressure (typically near 39)
- CAVITY_T: (°C) Optical cavity temperature (typically near 55)
- LASER_PHASE_P: (kPa) Laser phase pressure
- LASER_T: (°C) Laser temperature
- RESIDUAL: (n/a) Difference between raw and best fit spectra
- RING_DOWN_TIME: (µsecs) Indicator of cavity resonance
- THERMAL_ENCLOSURE_T: (°C) Optical enclosure temperature
- PHASE_ERROR: (counts) Dimensionless indicator of mode lock state
- LASER_T_SHIFT: (°C) Shift in laser center wavelength from factory calibration
- INPUT_VOLTAGE: (V) Power supply voltage
- CHK: (CHK) Checksum; ensures that the receiving software obtained the data without error and rejects corrupted data lines

As mentioned above, meteorological data, including PAR will be used in my analyses. The PAR data I will be using is logged every 15 minutes, meaning that measurements are averaged at 15 minute intervals. This meteorological data is from the South Slough National Estuarine Research Reserve located in Coos Bay, Oregon. More specifically, this station is referred to as Tom's Creek (Station Code: SOSTCMET) and has been active since 2016. The station is located at 43.27913, 124.31837. The data can be accessed at: Centralized Data Management Office | Data Export System.

Weather data files from SSNERR contain the following data:

- Date/time stamp (local standard time)
- Historical/provisional plus column (status of QAQC)
- Frequency column (identifies records as 15-minute, hourly, or daily averages)
- Parameter value column (consult metadata for station and parameter codes)
- Parameter flag column (parameter header preceded by `F_`; see QA Flags section)
- Parameter error code column (parameter header preceded by `EC_`; may be present depending on export type; see QA Flags section)

Though I will only be using the PAR data from the tower, the following parameters are recorded:

- ATemp: average air temperature (°C)
- RH: average relative humidity (/
- BP: average barometric pressure (mb)
- WSpd: average wind speed (m/s)
- MaxWSpd: maximum wind speed (m/s)
- MaxWSpdT: time of maximum wind speed measurement (hh:mm)
- Wdir: average wind direction (degrees true North; prior to April 1, 2008, no correction for magnetic declination)
- SDWDir: wind direction standard deviation
- TotPAR: total photosynthetically active radiation (mmol/m^2 per 15-minute interval)
- TotPrcp: total precipitation (mm)
- CumPrcp: cumulative precipitation (mm)

Data wrangling of the seasonal elevation-gradient flux dataset to compute fluxes

I began by wrangling the seasonal flux data, starting with loading in the raw files, where each measurement was identified by its unique ID label. To remove potential chamber effects during the initial placement of the chamber on top of the collar, I trimmed the first minute from every measurement. I then created a metadata file to match with the raw data so I could identify the correct measurement periods and attach the corresponding temperature and chamber volume. This information is required for calculating fluxes. I converted the default units used by *fluxfinder* to match those commonly used in literature. This allows for easier comparison between CO_2 and CH_4. All of these components were fed into ff_compute_fluxes, and I repeated the full workflow for each sampling date. Although multiple flux calculation schemes were generated, I kept only the linear regression–based estimates, following guidance from the literature. Finally, using consistent file name patterns and shared column structures across sampling days, I combined each processed file into a single master dataframe. I also quality-checked the calculated fluxes using R^2 values and by comparing the calculated fluxes to values reported in the literature to ensure they were reasonable. Note, that the code below shows the example workflow described for one sampling day only, specifically, August 4, 2025. This workflow was repeated for each of the seasonal sampling days to ultimately combine each separate file containing the calculated gas flux from each day into a master file.

```r
# Setting up the workspace
rm(list=ls())

library(fluxfinder)
library(ggplot2)
library(readr)
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v stringr   1.6.0
## v forcats   1.0.1     v tibble    3.3.0
## v lubridate 1.9.4     v tidyr     1.3.1
## v purrr     1.1.0
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts
```

```r
library(lubridate)
library(hms)
```

```
##
## Attaching package: 'hms'
##
## The following object is masked from 'package:lubridate':
##
##     hms
```

```r
library(viridis)
```

```
## Loading required package: viridisLite
```

```r
library(cowplot)
```

```
##
## Attaching package: 'cowplot'
##
## The following object is masked from 'package:lubridate':
##
##     stamp
```

```r
library(nlme)
```

```
##
## Attaching package: 'nlme'
##
## The following object is masked from 'package:dplyr':
##
##     collapse
```

```r
library(emmeans)
```

```
## Welcome to emmeans.
## Caution: You lose important information if you filter this package's results.
## See '? untidy'
```

```r
library(car)
```

```
## Loading required package: carData
##
## Attaching package: 'car'
##
```

```
## The following object is masked from 'package:dplyr':
##
##     recode
##
## The following object is masked from 'package:purrr':
##
##     some
```

```
# Load and clean data
dat <- read.csv("../Kristine-GHG-Marsh-Work/Data/8.4.2025/TG10-02179-2025-08-04T143100_m
                stringsAsFactors = FALSE)

# Keep only measurement periods (non-empty REMARK)
dat <- subset(dat, REMARK != "")

# Replace all "LM1retake" into "LM1" (8/4/2025 issue)
dat$REMARK[dat$REMARK == "LM1retake"] <- "LM1"

# Keep only relevant columns, specifically (SECONDS, REMARK, DATE, TIME)
dat <- dat[, c(2, 6:8)]

# Add ts column for obs_length column time subtraction later on
dat$ts <- paste(dat$DATE,dat$TIME,sep = ' ')
str(dat$ts)
```

```
##  chr [1:3249] "8/4/2025 15:43:28" "8/4/2025 15:43:29" "8/4/2025 15:43:30" ...
```

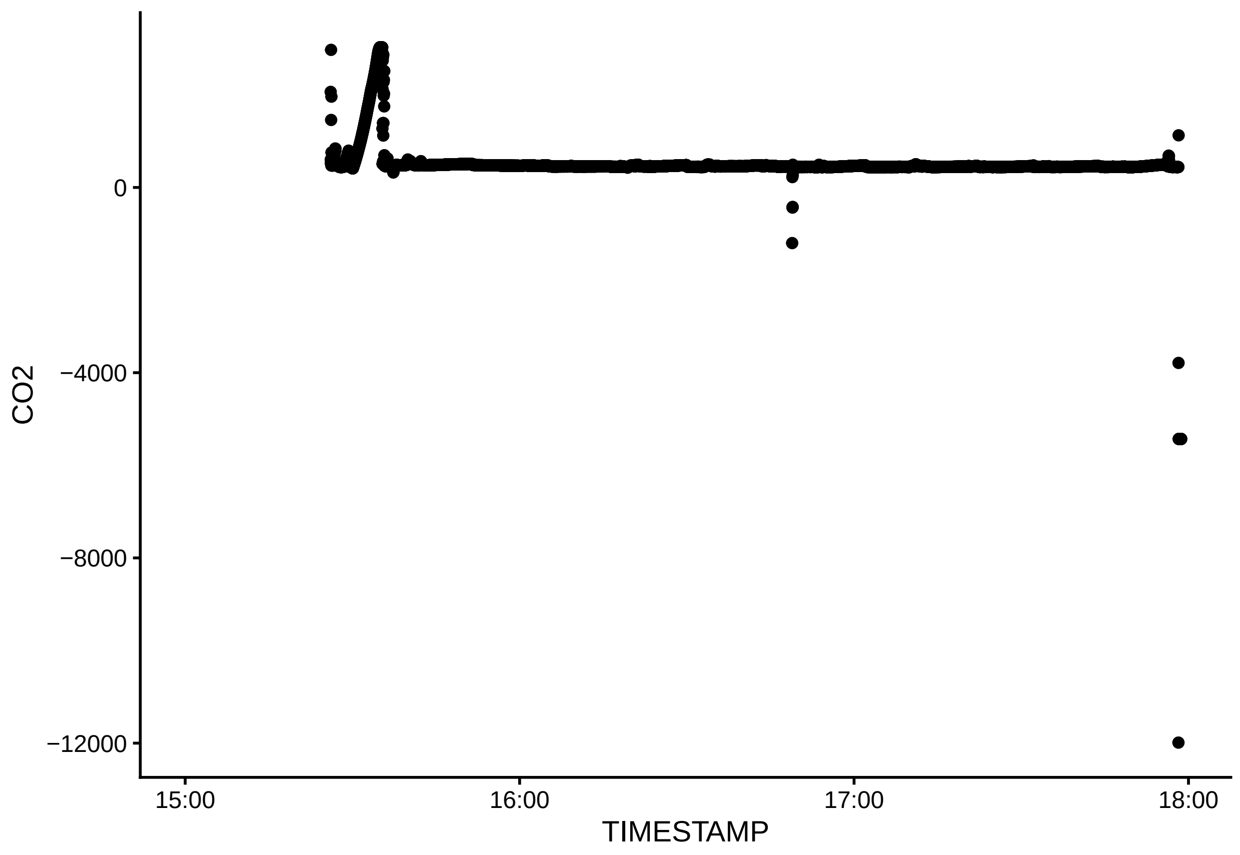
```
dat$ts <- as.POSIXct(dat$ts, format="%m/%d/%Y %H:%M:%S", tz = "America/Los_Angeles")

# Trim first minute off
dat_trimmed <- dat %>%
  group_by(REMARK) %>%
  mutate(start_time = min(ts, na.rm = TRUE)) %>%
  filter(ts >= start_time + 60) %>%
  ungroup()

# Fix date column in "raw" data to match format required by ffi_metadata_match (M/DD/Y
dat_trimmed$DATE <- as.Date(dat_trimmed$DATE, format = "%m/%d/%Y")
dat_trimmed$DATE <- as.character(dat_trimmed$DATE)

# Define summary function per site
get_site_metadata <- function(df) {

  data.frame(
    date         = unique(df$DATE),
    site         = unique(df$REMARK),
    start        = min(df$ts, na.rm = TRUE),
    end          = max(df$ts, na.rm = TRUE),
    obs_length  = as.numeric(difftime(max(df$ts, na.rm = TRUE),
                                       min(df$ts, na.rm = TRUE),
                                       units = "secs"))
  )
}

# Apply function to each site and combine
by_site <- split(dat_trimmed, dat_trimmed$REMARK)
metadata <- do.call(rbind, lapply(by_site, get_site_metadata))
rownames(metadata) <- NULL

# Adding changing volume and temperature for each measurement (in order according to m

# 8/04/2025:
metadata$volume <- c(749.5, 749.2, 749.5, 748.9, 749.2, 749.0, 749.2, 749.4, 749.2) # ad
metadata$temperature <- c(26.2, 22.2, 22.7, 24.6, 23.7, 16.3, 18.4, 17.5, 15.7) # adding

# 8/11/2025:
# metadata$volume <- c(749.5, 749.2, 749.5, 748.9, 749.2, 749.0, 749.2, 749.4, 749.2)
# metadata$temperature <- c(29, 25.5, 25.2, 25.7, 24.5, 24.6, 22.6, 24.8, 22.9) # addi
```

```
# Reading in raw data from LI 7810
f <- "../Kristine-GHG-Marsh-Work/Data/8.4.2025/TG10-02179-2025-08-04T143100 (1).data" #
df <- ffi_read_LI7810(f)
```

```
## TG10-02179-2025-08-04T143100 (1).data: read 10674 rows of TG10-02179 data, 2025-08-04
```

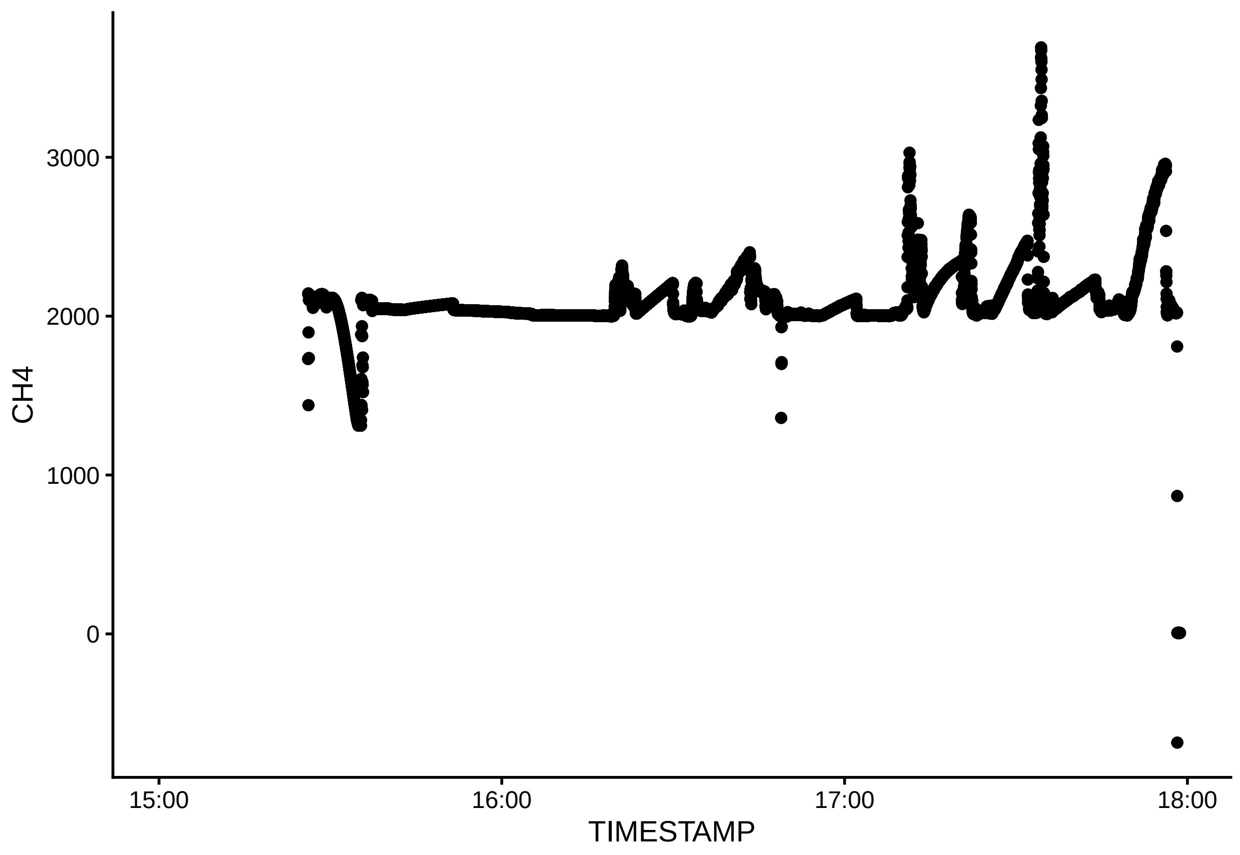
```
# Plotting raw data
ggplot(df, aes(TIMESTAMP, CO2)) +
  geom_point() +
  theme_classic()
```

```
## Warning: Removed 1516 rows containing missing values or values outside the scale rang
## (‘geom_point()‘).
```



```
ggplot(df, aes(TIMESTAMP, CH4)) +
  geom_point() +
  theme_classic()
```

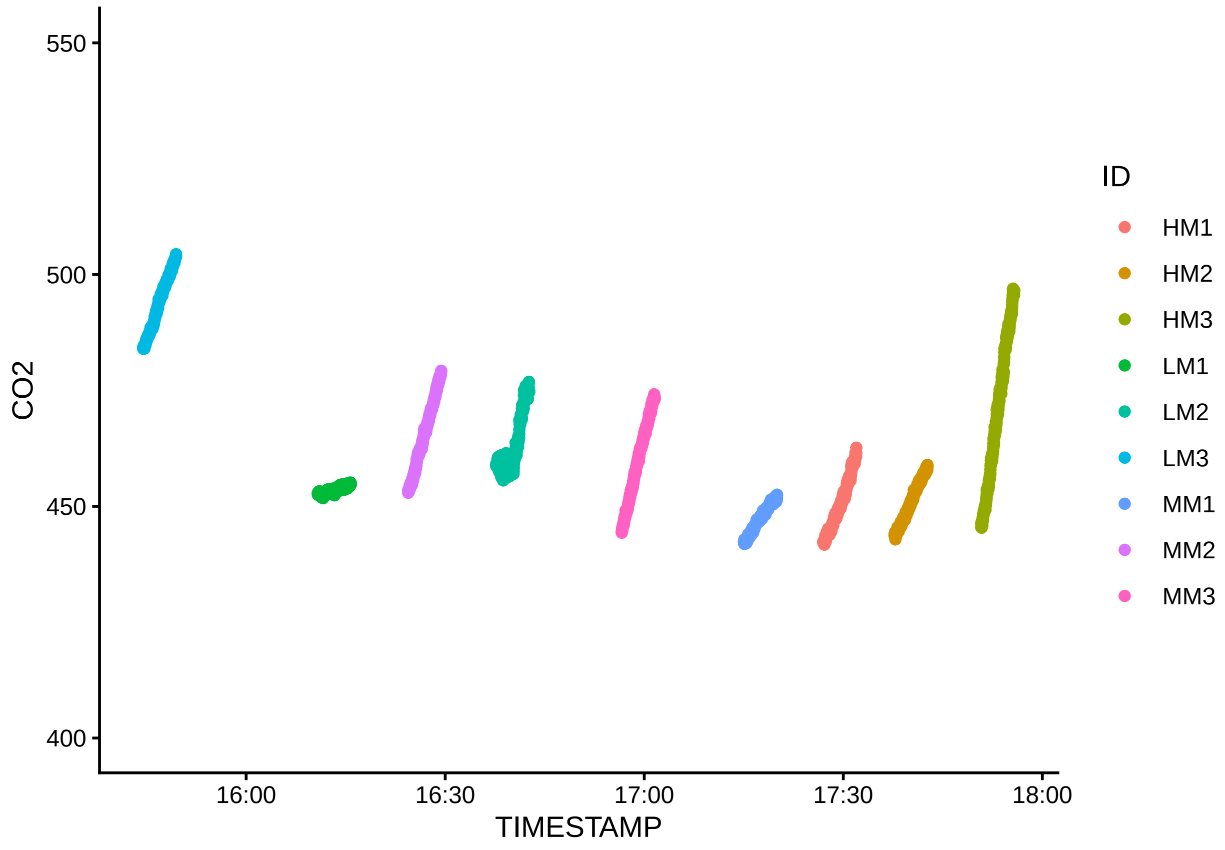
```
## Warning: Removed 1516 rows containing missing values or values outside the scale rang
## (‘geom_point()‘).
```



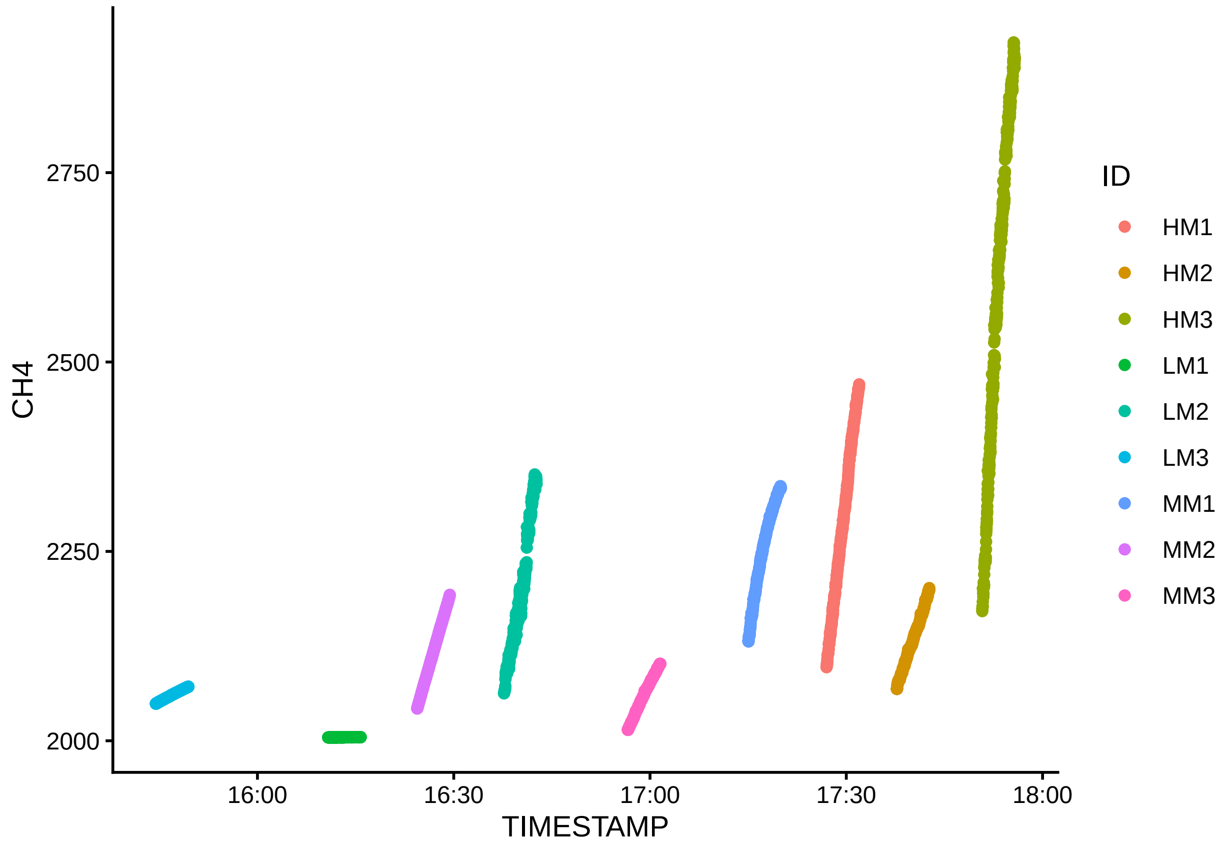
```r
# Making sure everything is the right format to enter our matching function
## data_timestamps: either character (YYYY-MM-DD HH:MM:SS) or POSIXct
## start_dates: either character (YYYY-MM-DD) or POSIXct
## start_times: either character (HH:MM:SS) or period
## obs_lengths: Observation lengths in seconds, numeric; must be same length as start_

metadata$start <- format(metadata$start, "%H:%M:%S")

# Matching metadata with raw file data set
df$metadata_row <- ffi_metadata_match(
  data_timestamps = df$TIMESTAMP,
  start_dates = metadata$date,
  start_times = metadata$start,
  obs_length = metadata$obs_length
)
#View(df)
```

```r
# Based on the row match information, add an "ID" column to the data
df$ID <- metadata$site[df$metadata_row]
metadata$metadata_row <- seq_len(nrow(metadata))
```

```r
# Remove rows that failed to match (NA ID)
df <- df %>%
  filter(!is.na(ID))

# Plot CO2 and CH4 (measurement periods now identified)
pCO2 <- ggplot(df, aes(TIMESTAMP, CO2, color = ID)) +
  geom_point() +
  ylim(400, 550) +
  theme_classic()
print(pCO2)
```



```r
pCH4 <- ggplot(df, aes(TIMESTAMP, CH4, color = ID)) +
  geom_point() +
  theme_classic()
print(pCH4)
```



```r
# Units and taking into account changing volumes

# At this point, use ID for cleaned plot labels

# Merge the data and metadata
dat_changing_vol <- df %>%
  left_join(
    metadata %>%
      select(metadata_row, volume, temperature),
    by = "metadata_row"
  )

# Unit conversion CO2 using the changing volume information:
dat_changing_vol$CO2_umol <- ffi_ppm_to_umol(dat_changing_vol$CO2,
                                             volume = dat_changing_vol$volume,
                                             temp = dat_changing_vol$temperature)
```

```
## Assuming atm = 101325 Pa
```

```
## Using R = 8.31446261815324 m3 Pa K-1 mol-1
```

```r
# Unit conversion CH4 using the changing volume information:
dat_changing_vol$CH4_nmol <- ffi_ppb_to_nmol(dat_changing_vol$CH4,
                                             volume = dat_changing_vol$volume,
                                             temp = dat_changing_vol$temperature)
```

```
## Assuming atm = 101325 Pa
## Using R = 8.31446261815324 m3 Pa K-1 mol-1
```

```r
# Using constant area
dat_changing_vol$CO2_umol_m2 <- dat_changing_vol$CO2_umol / 12.3

dat_changing_vol$CH4_nmol_m2 <- dat_changing_vol$CH4_nmol / 12.3

aggregate(CO2_umol_m2 ~ ID, data = dat_changing_vol, FUN = mean)
```

```
##    ID CO2_umol_m2
## 1 HM1     1117789
## 2 HM2     1133341
## 3 HM3     1181103
## 4 LM1     1130079
## 5 LM2     1159120
## 6 LM3     1266431
## 7 MM1     1138842
## 8 MM2     1189691
## 9 MM3     1181298
```

```r
aggregate(CH4_nmol_m2 ~ ID, data = dat_changing_vol, FUN = mean)
```

```
##    ID CH4_nmol_m2
## 1 HM1     5674808
## 2 HM2     5363156
## 3 HM3     6495259
## 4 LM1     4995469
## 5 LM2     5496686
## 6 LM3     5282810
## 7 MM1     5739554
## 8 MM2     5410039
## 9 MM3     5293871
```

```r
# Compute fluxes
fluxesCO2 <- ffi_compute_fluxes(dat_changing_vol,
                                group_column = "ID",
                                time_column = "TIMESTAMP",
                                gas_column = "CO2_umol_m2",
                                dead_band = 0)
```

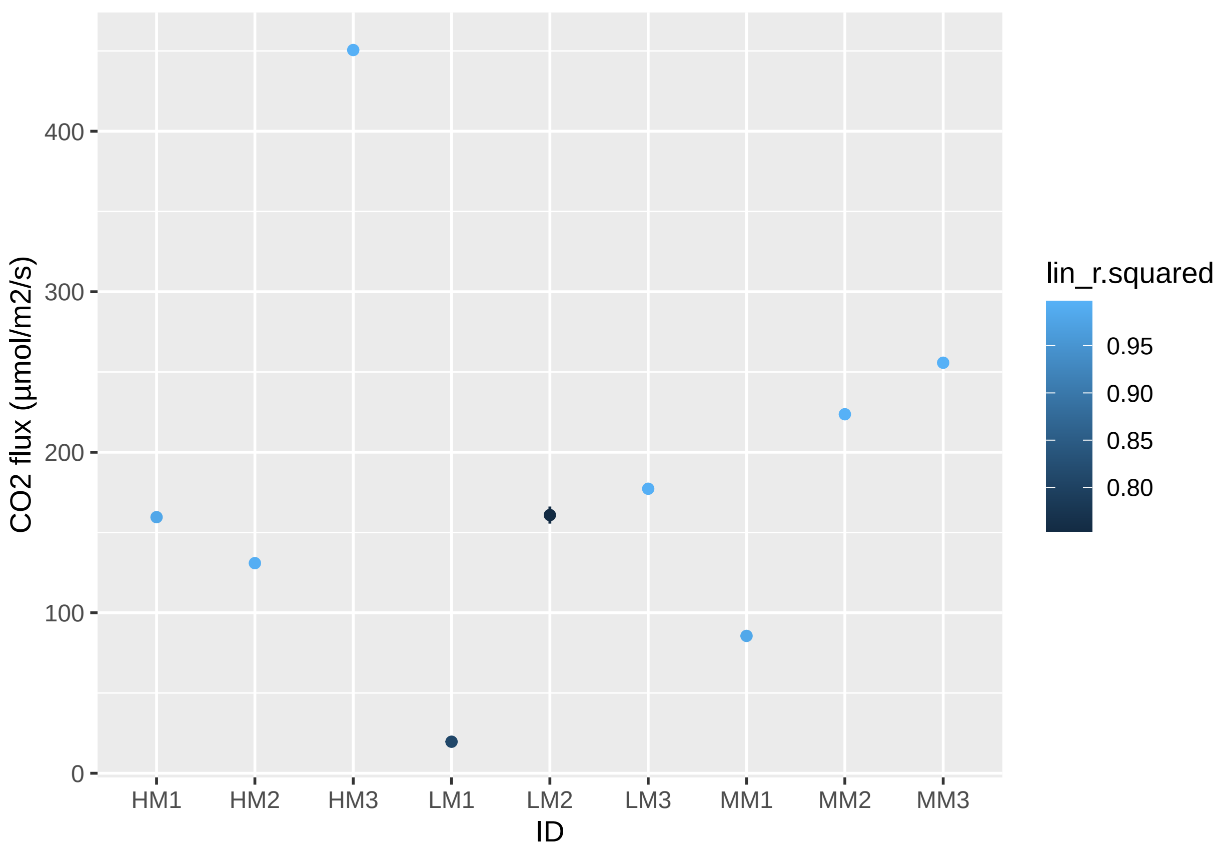
```
## NOTE: HM81_flux.estimate is not NA, implying nonlinear data
## NOTE: HM81_flux.estimate is not NA, implying nonlinear data
## NOTE: HM81_flux.estimate is not NA, implying nonlinear data
## NOTE: HM81_flux.estimate is not NA, implying nonlinear data
## NOTE: HM81_flux.estimate is not NA, implying nonlinear data
```

```
fluxesCH4 <- ffi_compute_fluxes(dat_changing_vol,
                             group_column = "ID",
                             time_column = "TIMESTAMP",
                             gas_column = "CH4_nmol_m2",
                             dead_band = 0)
```

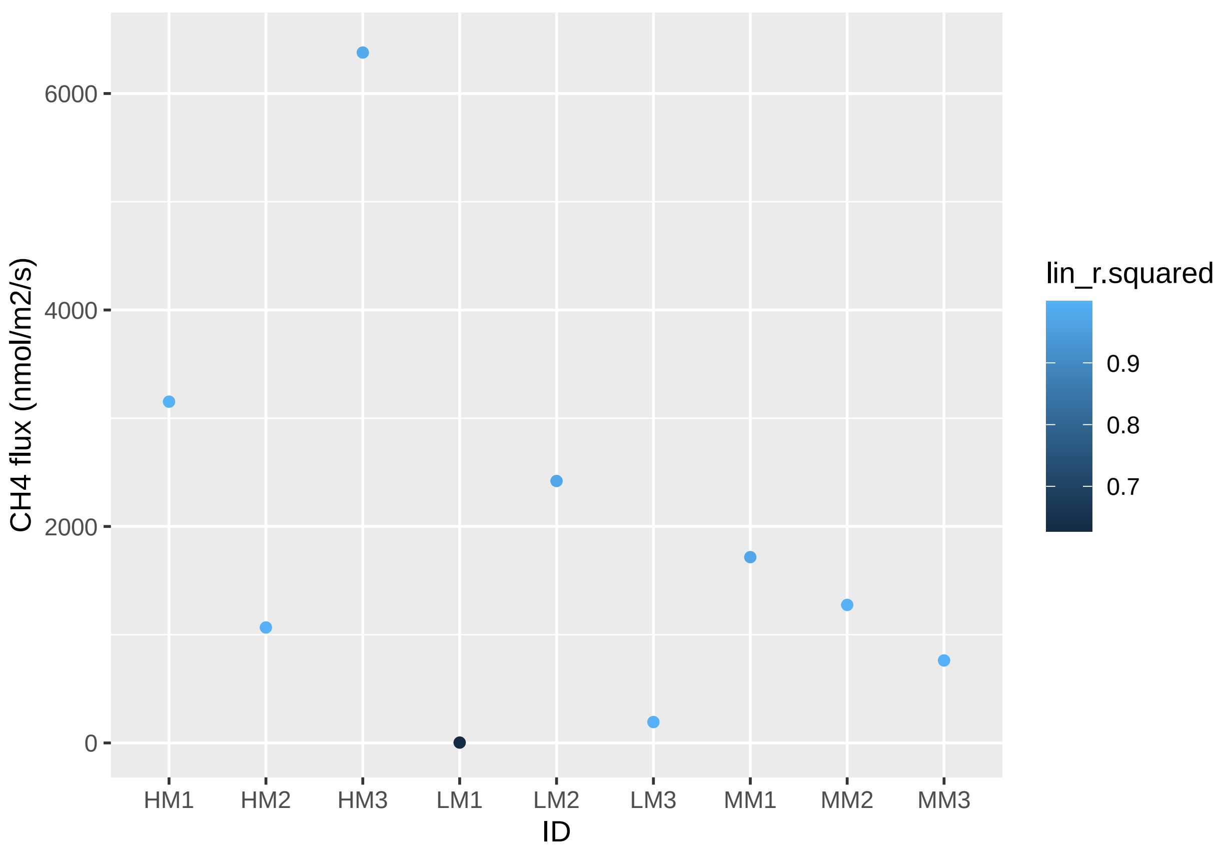
```
## NOTE: HM81_flux.estimate is not NA, implying nonlinear data
## NOTE: HM81_flux.estimate is not NA, implying nonlinear data
## NOTE: HM81_flux.estimate is not NA, implying nonlinear data
## NOTE: HM81_flux.estimate is not NA, implying nonlinear data
## NOTE: HM81_flux.estimate is not NA, implying nonlinear data
## NOTE: HM81_flux.estimate is not NA, implying nonlinear data
## NOTE: HM81_flux.estimate is not NA, implying nonlinear data
```

```
# Save fluxes to a csv file
fluxes_all <- fluxesCO2 %>%
  left_join(fluxesCH4, by = c("ID", "TIMESTAMP", "TIMESTAMP_min", "TIMESTAMP_max"), suff

# write.csv(fluxes_all, file = "../Data/Script Outputs/8.4.2025/08_04_2025_Fluxes_Calc
```

```
# Checking calculated fluxes visually
ggplot(fluxesCO2, aes(ID, lin_flux.estimate, color = lin_r.squared)) +
  geom_point() +
  geom_linerange(aes(ymin = lin_flux.estimate - lin_flux.std.error,
                     ymax = lin_flux.estimate + lin_flux.std.error)) +
  ylab("CO2 flux (µmol/m2/s)")
```



```
ggplot(fluxesCH4, aes(ID, lin_flux.estimate, color = lin_r.squared)) +
  geom_point() +
  geom_linerange(aes(ymin = lin_flux.estimate - lin_flux.std.error,
                     ymax = lin_flux.estimate + lin_flux.std.error)) +
  ylab("CH4 flux (nmol/m2/s)")
```



```
# Creating a masterfile of calculated seasonal fluxes and more flux data exploration
all_csvs <- list.files(
  path = "../Kristine-GHG-Marsh-Work/Data/Script Outputs/",
  pattern = "Trimmed\\.csv$",        # only CSV files
  recursive = TRUE,           # look into subfolders
  full.names = TRUE           # include full paths
)
all_csvs <- all_csvs[-3] # Remove 8/21/2025 since only interested in seasonal fluxes, 

combined_fluxes <- all_csvs %>%
  map_dfr(read_csv, .id = "source_file")
```

```
## New names:
## Rows: 9 Columns: 47
## -- Column specification
## -------------------------------------------------------- Delimiter: "," chr
## (1): site dbl (41): ...1, HM81_AIC_CO2, HM81_flux.estimate_CO2,
## HM81_p.value_CO2, HM8... lgl (2): rob_converged_CO2, rob_converged_CH4 dttm
## (3): TIMESTAMP, TIMESTAMP_min, TIMESTAMP_max
## i Use `spec()` to retrieve the full column specification for this data. i
## Specify the column types or set `show_col_types = FALSE` to quiet this message.
## New names:
## Rows: 9 Columns: 47
## -- Column specification
## -------------------------------------------------------- Delimiter: "," chr
## (1): site dbl (41): ...1, HM81_AIC_CO2, HM81_flux.estimate_CO2,
## HM81_p.value_CO2, HM8... lgl (2): rob_converged_CO2, rob_converged_CH4 dttm
## (3): TIMESTAMP, TIMESTAMP_min, TIMESTAMP_max
## i Use ‘spec()‘ to retrieve the full column specification for this data. i
## Specify the column types or set ‘show_col_types = FALSE‘ to quiet this message.
## New names:
## Rows: 9 Columns: 47
## -- Column specification
## -------------------------------------------------------- Delimiter: "," chr
## (1): site dbl (41): ...1, HM81_AIC_CO2, HM81_flux.estimate_CO2,
## HM81_p.value_CO2, HM8... lgl (2): rob_converged_CO2, rob_converged_CH4 dttm
## (3): TIMESTAMP, TIMESTAMP_min, TIMESTAMP_max
## i Use ‘spec()‘ to retrieve the full column specification for this data. i
## Specify the column types or set ‘show_col_types = FALSE‘ to quiet this message.
## New names:
## Rows: 9 Columns: 47
## -- Column specification
## -------------------------------------------------------- Delimiter: "," chr
## (1): site dbl (41): ...1, HM81_AIC_CO2, HM81_flux.estimate_CO2,
## HM81_p.value_CO2, HM8... lgl (2): rob_converged_CO2, rob_converged_CH4 dttm
## (3): TIMESTAMP, TIMESTAMP_min, TIMESTAMP_max
## i Use ‘spec()‘ to retrieve the full column specification for this data. i
## Specify the column types or set ‘show_col_types = FALSE‘ to quiet this message.
## New names:
## Rows: 6 Columns: 47
## -- Column specification
## -------------------------------------------------------- Delimiter: "," chr
## (1): site dbl (41): ...1, HM81_AIC_CO2, HM81_flux.estimate_CO2,
## HM81_p.value_CO2, HM8... lgl (2): rob_converged_CO2, rob_converged_CH4 dttm
## (3): TIMESTAMP, TIMESTAMP_min, TIMESTAMP_max
## i Use ‘spec()‘ to retrieve the full column specification for this data. i
## Specify the column types or set ‘show_col_types = FALSE‘ to quiet this message.
## * ‘‘ -> ‘...1‘
```

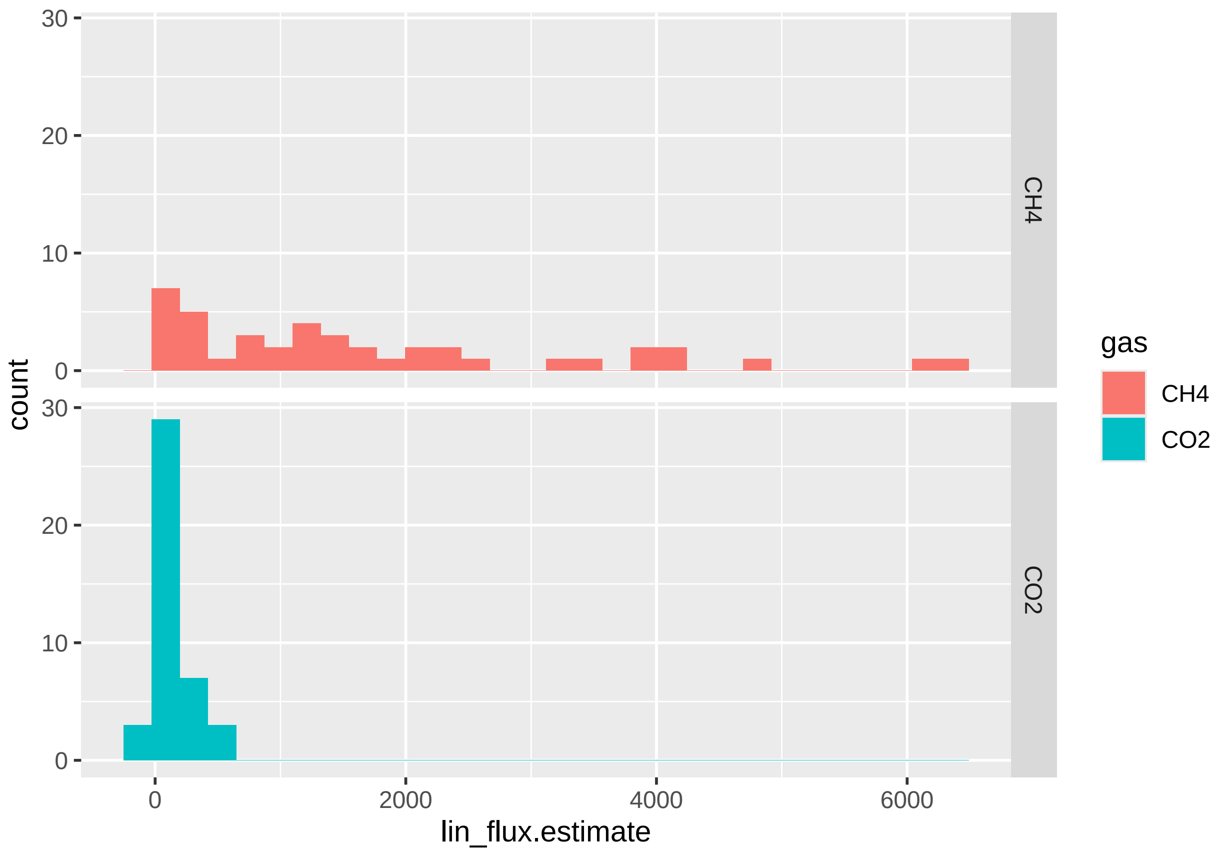
```r
combined_fluxes <- combined_fluxes %>%
  rename("sample_num" = "...1")

combined_fluxes_long <- combined_fluxes %>%
  # Gather all CO2 and CH4 columns into long form
  pivot_longer(
    cols = matches("_(CO2|CH4)$"),               # select columns that end in _CO2 or
    names_to = c("model_metric", "gas"),           # split into model_metric and gas
    names_pattern = "(.*)_(CO2|CH4)$",
    values_to = "value"
  ) %>%
  # Spread them back out so each model_metric becomes its own column again
  pivot_wider(
    names_from = model_metric,
    values_from = value
  )

df <- combined_fluxes_long

# Adding elevation column
df <- df %>%
     mutate(elevation = case_when(
     grepl("^HM", site) ~ "H",
     grepl("^MM", site) ~ "M",
     grepl("^LM", site) ~ "L"
   ))

df$elevation <- as.factor(df$elevation)
```

Seasonal flux data exploration

I created basic histograms to explore the distribution of calculated fluxes and to gauge whether their magnitudes were comparable to values reported in the literature for similar wetland systems. I also applied a simple filtering criteria, keeping only fluxes with $R^2 \geq 0.33$ and $p \leq 0.05$, following a similar quality-control approach used in previous studies.

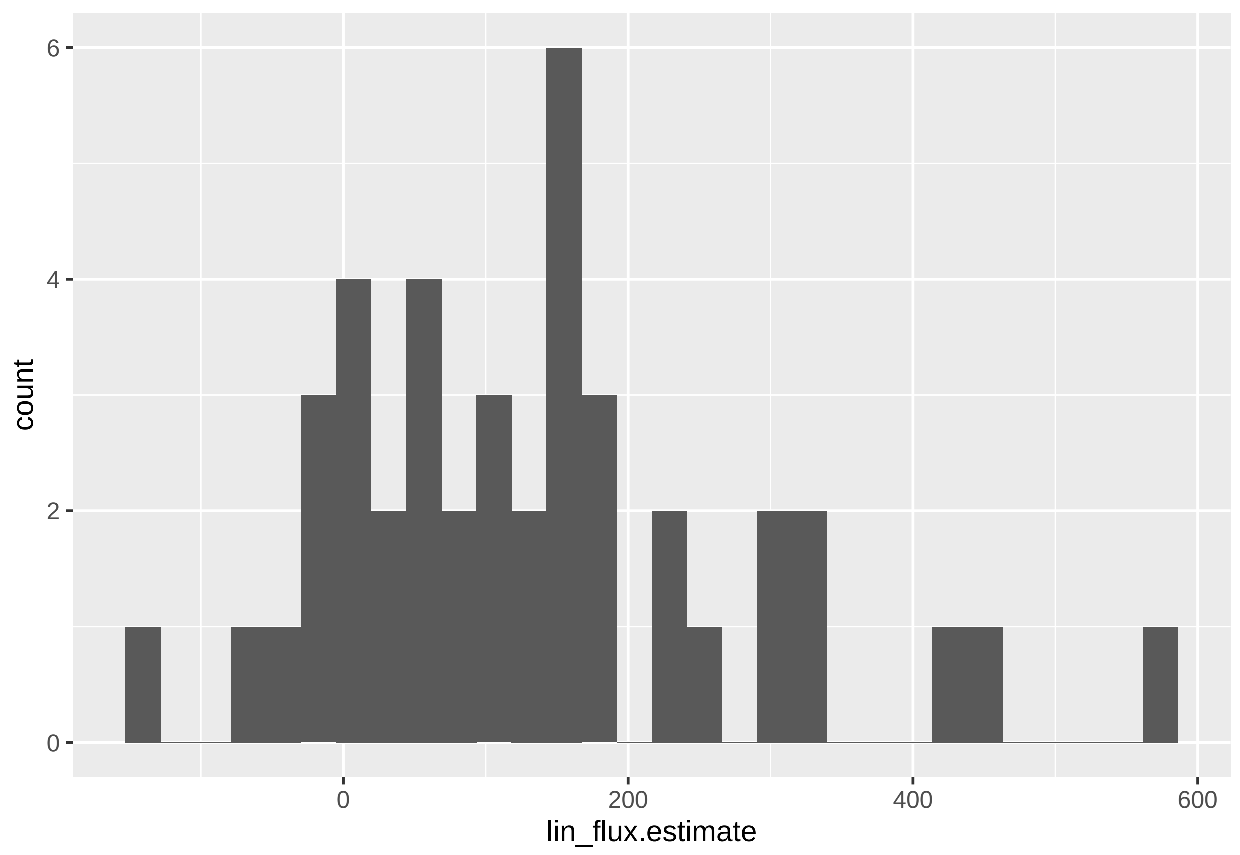
```
# Histograms before filtering

df %>% ggplot(aes(x = lin_flux.estimate, fill = gas)) +
  geom_histogram() +
  facet_grid(gas ~ .)
```

```
## `stat_bin()` using `bins = 30`. Pick better value `binwidth`.
```



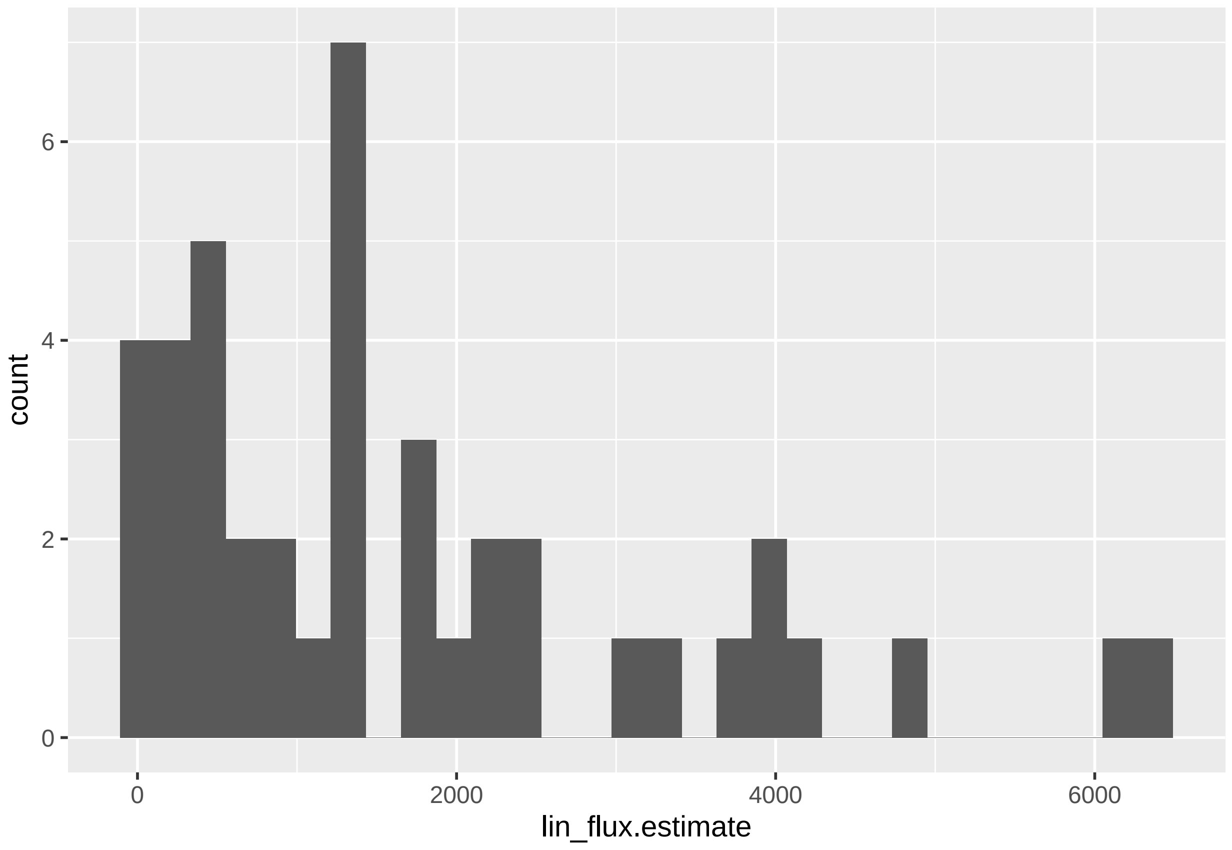
```
df %>%
  filter(gas == "CO2") %>%
  ggplot(aes(x = lin_flux.estimate)) +
  geom_histogram()
```

```
## `stat_bin()` using `bins = 30`. Pick better value `binwidth`.
```



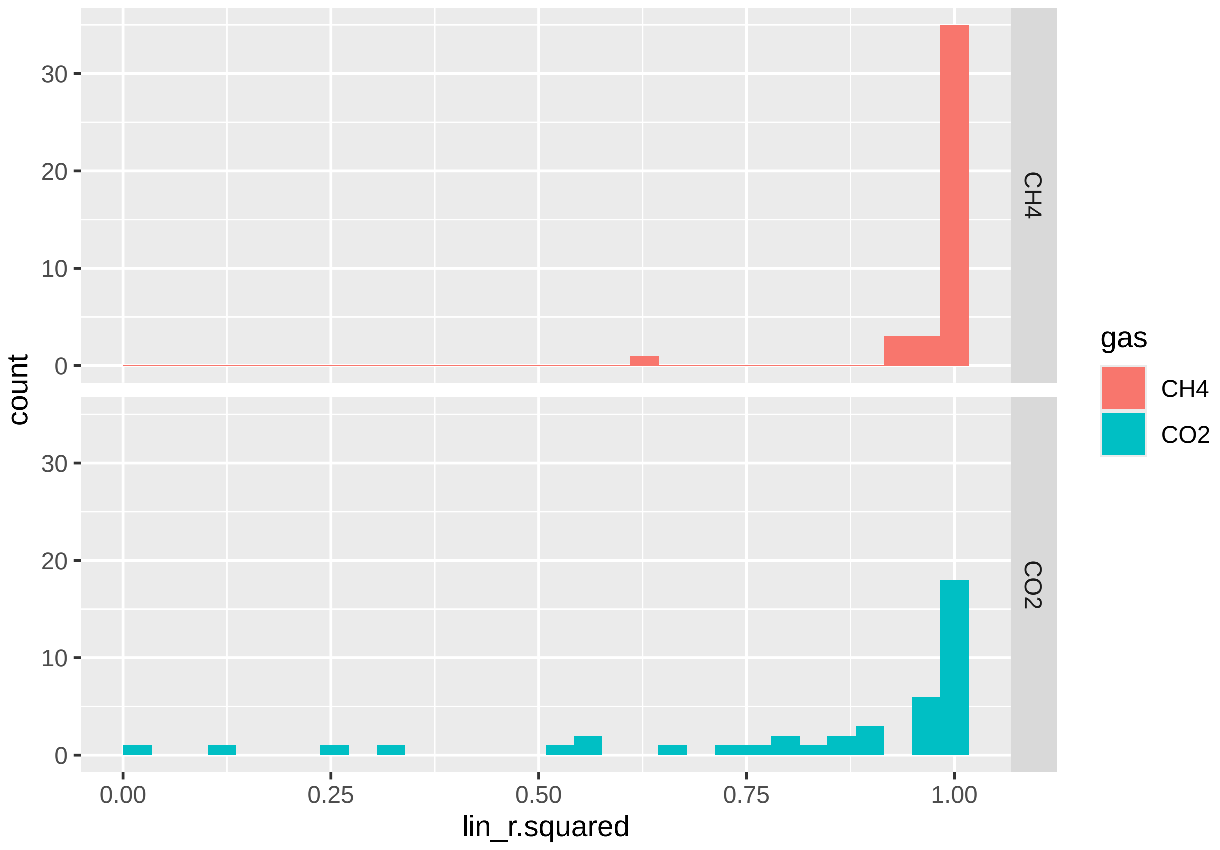
```
df %>%
  filter(gas == "CH4") %>%
  ggplot(aes(x = lin_flux.estimate)) +
  geom_histogram()
```

```
## `stat_bin()` using `bins = 30`. Pick better value `binwidth`.
```



```r
df %>% ggplot(aes(x = lin_r.squared, fill = gas)) +
  geom_histogram() +
  facet_grid(gas ~ .)
```

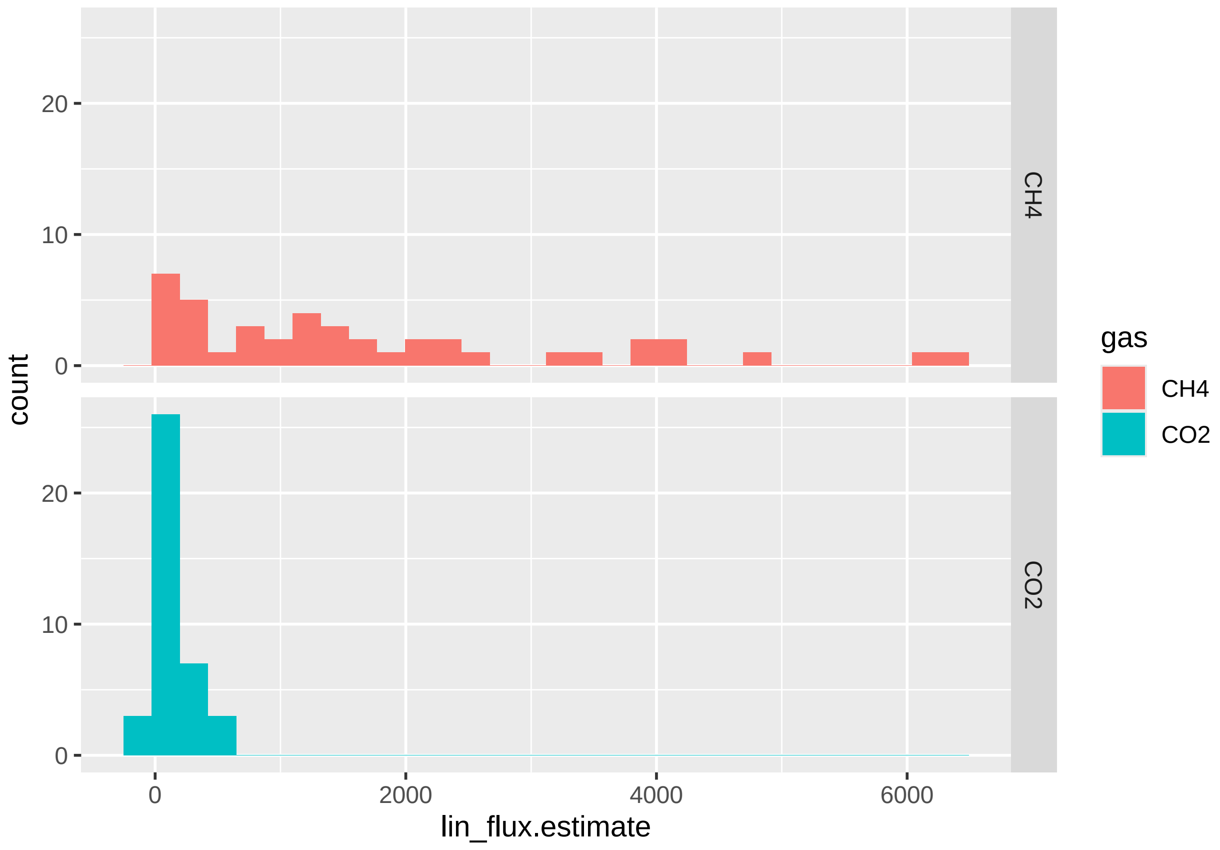
```
## ‘stat_bin()‘ using ‘bins = 30‘. Pick better value ‘binwidth‘.
```



```
# Basic filtering step

df_filt <- df %>%
  filter(lin_r.squared >= 0.33 & lin_p.value <= 0.05) # seen in the literature

# Adding date column
df_filt <- df_filt %>%
  mutate(
    date = as.character(date(TIMESTAMP))  # extracts YYYY-MM-DD from timestamp
  )
```

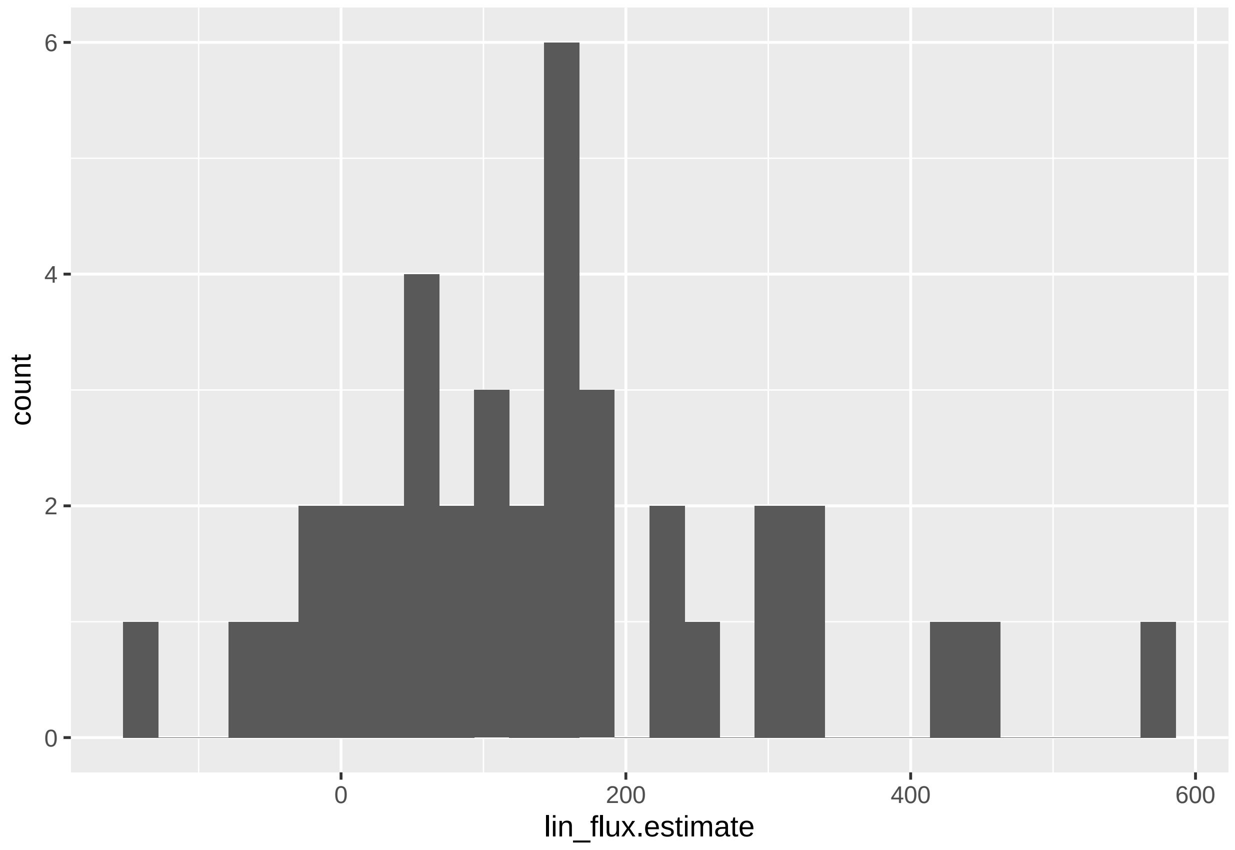
```
# Histograms after filtering

df_filt %>% ggplot(aes(x = lin_flux.estimate, fill = gas)) +
  geom_histogram() +
  facet_grid(gas ~ .)
```

```
## `stat_bin()` using `bins = 30`. Pick better value `binwidth`.
```



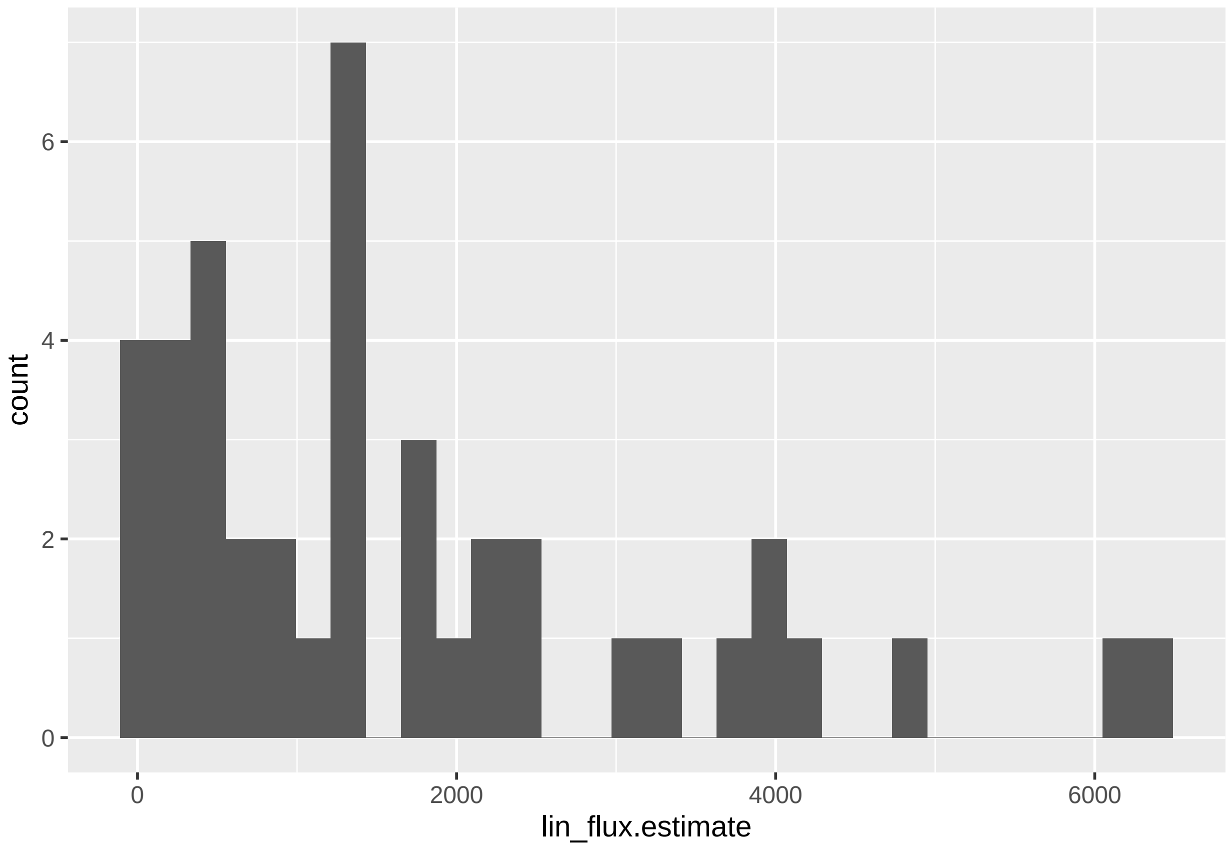
```
df_filt %>%
  filter(gas == "CO2") %>%
  ggplot(aes(x = lin_flux.estimate)) +
  geom_histogram()
```

```
## `stat_bin()` using `bins = 30`. Pick better value `binwidth`.
```



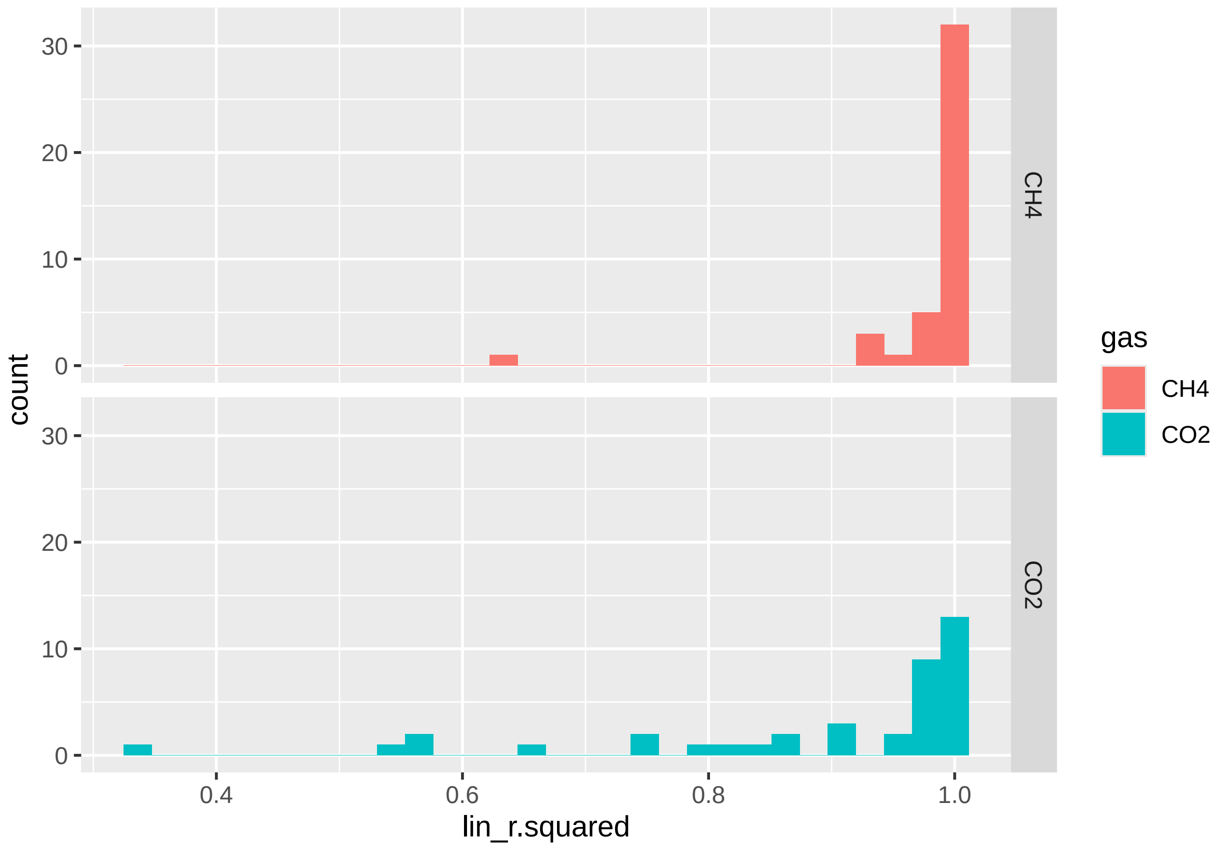
```
df_filt %>%
  filter(gas == "CH4") %>%
  ggplot(aes(x = lin_flux.estimate)) +
  geom_histogram()
```

```
## 'stat_bin()' using 'bins = 30'. Pick better value 'binwidth'.
```



```r
df_filt %>% ggplot(aes(x = lin_r.squared, fill = gas)) +
  geom_histogram() +
  facet_grid(gas ~ .)
```

```
## `stat_bin()` using `bins = 30`. Pick better value `binwidth`.
```



Visualization by plotting seasonal flux data

Taking all this data, I created plots to visualuze the seasonal fluxes across the elevation gradient.

Figure 1.

```r
# Adding a column where linear CH4 flux estimate is converted to umol m-2 s-1 from nmo
df <- df %>%
  mutate(lin_flux.estimate_umol = ifelse(
    gas == "CH4",
    lin_flux.estimate / 1000,  # convert CH4 from nmol → µmol
    lin_flux.estimate          # leave CO2 as is
  ))

# Data filtering
df_filt <- df %>%
  filter(lin_r.squared >= 0.33 & lin_p.value <= 0.05)

df <- df_filt
df %>%
  mutate(
    elevation = factor(
      elevation,
      levels = c("L", "M", "H"),
      labels = c("Low", "Mid", "High")
      )
    ) %>%
  ggplot(aes(x = elevation, y = lin_flux.estimate_umol, fill = elevation)) +
  geom_violin(trim = FALSE, alpha = 0.7) +
  geom_boxplot(width = 0.15, outlier.shape = NA, color = "black", alpha = 0.8) +
  facet_wrap(~gas, scales = "free_y") +
  geom_hline(yintercept = 0, linetype = "dashed") +
  scale_fill_viridis_d(option = "D", guide = "none") +
  labs(x = "Elevation",
       y = expression("Flux ("*mu*"mol m"^{-2}*" s"^{-1}*")")
       ) +
  theme_bw() +
  theme(
    legend.position = "n",
    strip.text = element_text(size = 16, face = "bold"),
    axis.text = element_text(size = 11),
    axis.text.x = element_text(size = 13),
    axis.text.y = element_text(size = 13),
    axis.title = element_text(size = 14)
    )
```

Figure 2.

```r
df_plot <- df %>%
  mutate(
    elevation = factor(elevation, levels = c("L", "M", "H")),
    date = as.factor(format(as.Date(TIMESTAMP), "%Y-%m-%d")) # discrete sampling dates
  )

ggplot(df_plot, aes(x = date, y = lin_flux.estimate_umol, fill = elevation)) +
  geom_boxplot(position = position_dodge(width = 0.8), alpha = 0.8, outlier.shape = NA)
  geom_hline(yintercept = 0, linetype = "dashed") +
  facet_wrap(~gas, scales = "free_y") +
  scale_fill_viridis_d(option = "D") +
  labs(
    x = "Sampling Date",
    y = expression("Flux ("*mu*"mol m"^{-2}*" s"^{-1}*")"),
    fill = "Elevation"
```

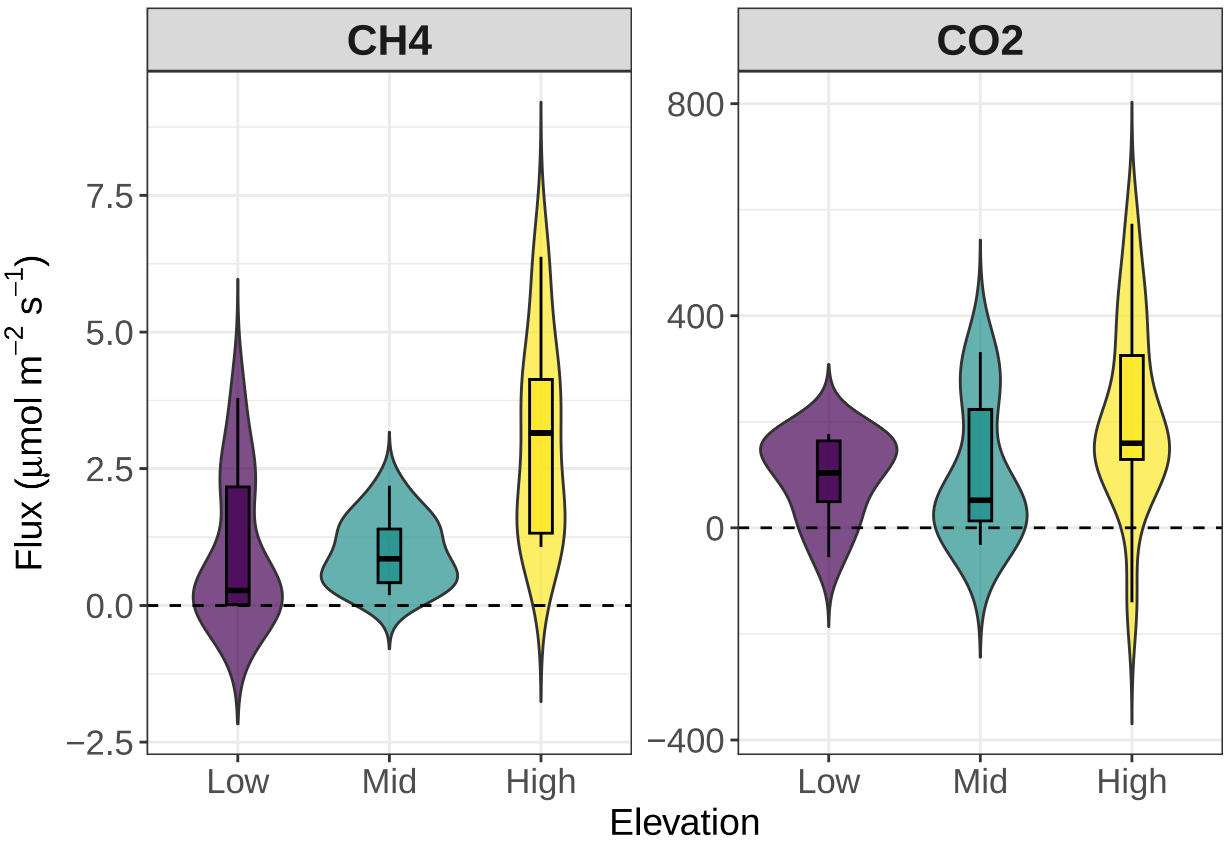


Figure 1: CO2 and CH4 fluxes show a tendency to be higher in high marsh areas than in mid or low marsh zones, although these differences are not statistically significant.

```
  ) +
  theme_bw(base_size = 14) +
  theme(
    strip.text = element_text(size = 16, face = "bold"),
    axis.text.x = element_text(size = 12, angle = 45, hjust = 1),
    axis.text.y = element_text(size = 12),
    legend.position = "right",
    axis.title = element_text(size = 14)

  )
```

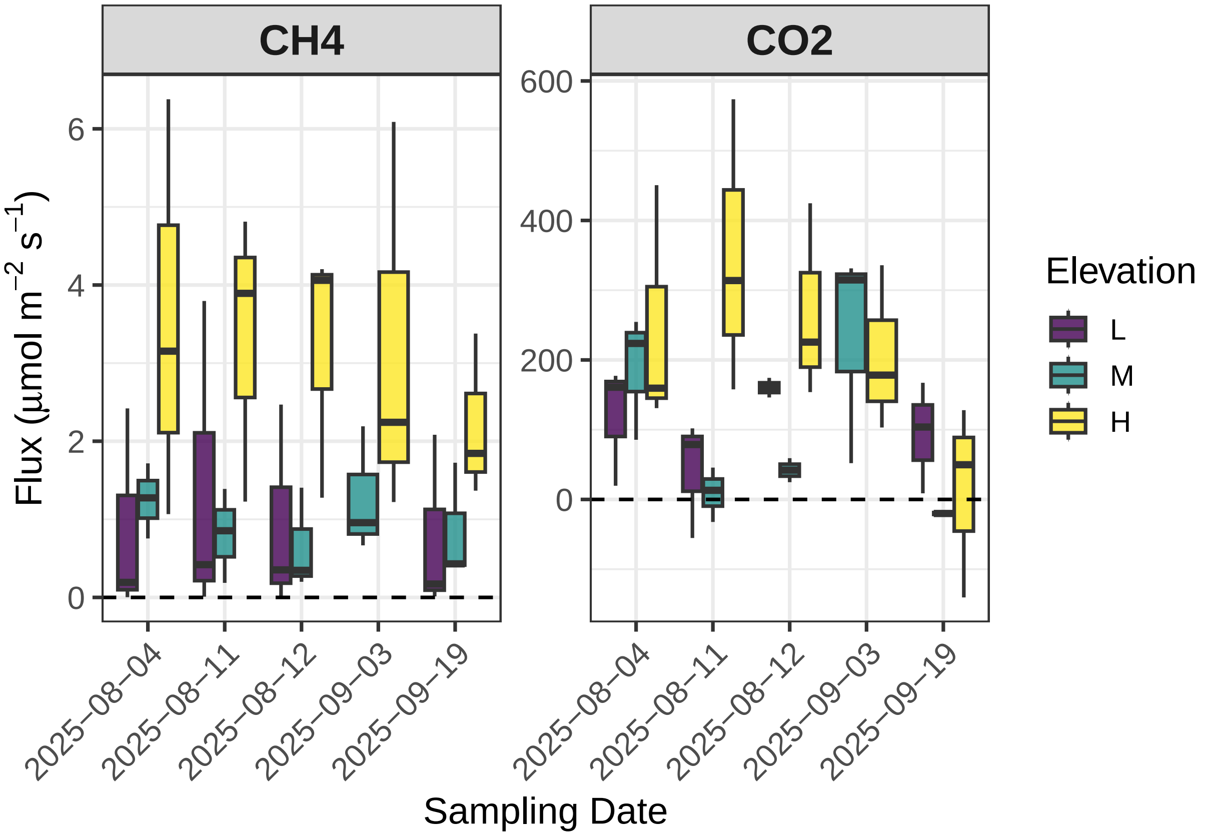


Figure 2: CO2 flux increases with elevation ($p = 0.027$) and shows a marginal temporal trend ($p = 0.074$), while CH4 flux is consistently higher in high marsh ($p < 0.001$) and stable over time. No elevation and date interactions were observed.

Data wrangling of the 24-hour high-marsh flux dataset to compute fluxes

I followed similar steps to compute fluxes as I did for the seasonal fluxes described above, with a few differences to note. The function used to generate the metadata was modified to detect where one measurement ends and the next begins by identifying large time gaps and assigning a new period ID each time a new session starts. The flux calculation was also slightly more complex. I looped through each chamber–measurement period combination and ran *fluxfinder* to calculate fluxes (the example below shows only CO_2), then combined these fluxes into a single dataframe. The corresponding metadata was also attached so each flux could later be analyzed. While the example shows only CO_2, the same code was applied to CH_4 with the appropriate variables replaced, and ultimately the CO_2 and CH_4 data files were joined into another master file. Note that I have not included all the code for wrangling the 24-hour data, as most steps are the same as those described in the seasonal flux data wrangling section. Additionally, many of these code chunks were written in separate R Markdown files, which is why the example code below has been commented out; the document will not knit if the code is ran as-is.

```r
# Preparing metadata function
# get_site_metadata <- function(df) {
#
#   df %>%
#     arrange(REMARK, ts) %>%
#     group_by(REMARK) %>%
#     mutate(
#       gap = ts - lag(ts),
#       period = cumsum(is.na(gap) | gap > 360) + 1
#     ) %>%
#     group_by(REMARK, period) %>%
#     summarise(
#       start = min(ts, na.rm = TRUE),
#       end = max(ts, na.rm = TRUE),
#       obs_length = as.numeric(difftime(max(ts, na.rm = TRUE), min(ts, na.rm = TRUE),
#       hour = first(HOUR),
#       date = unique(DATE),
#       .groups = "drop"
#     ) %>%
#     mutate(obs_length_mins = obs_length / 60)
# }
```

```r
# Compute 24 hour CO2 fluxes

# Get all unique combinations of ID and metadata_row
# groups <- unique(dat_changing_vol[, c("ID", "metadata_row")])
```

```
#
# # run fluxfinder for each combo
# fluxesCO2_list <- lapply(seq_len(nrow(groups)), function(i) {
#
#   id_i <- groups$ID[i]
#   mr_i <- groups$metadata_row[i]
#
#   dat_i <- dat_changing_vol %>%
#     filter(ID == id_i, metadata_row == mr_i)
#
#   out <- ffi_compute_fluxes(
#     dat_i,
#     group_column = "ID",
#     time_column = "TIMESTAMP",
#     gas_column = "CO2_umol_m2",
#     dead_band = 0
#   )
#
#   out$ID <- id_i
#   out$metadata_row <- mr_i
#   out
# })
#
# # combine into one dataframe
# fluxesCO2 <- do.call(rbind, fluxesCO2_list)
#
# # attach metadata (for hour, date, etc.)
# metadata_lookup <- metadata %>%
#   mutate(metadata_row = row_number()) %>%
#   select(metadata_row, date, hour, start, end)
#
# fluxesCO2 <- fluxesCO2 %>%
#   left_join(metadata_lookup, by = "metadata_row")
```

```
# # Save both CO2 and CH4 24 hour fluxes to a csv file
# fluxes_all <- fluxesCO2 %>%
#   left_join(fluxesCH4, by = c("ID", "TIMESTAMP", "TIMESTAMP_min", "TIMESTAMP_max", "

# write.csv(fluxes_all, file = "../Data/Script Outputs/8.21.2025/08_21_2025_Fluxes_Cal
```

Visualization by plotting 24 hour flux data

Taking all the 24-hour data, I created plots to visualize the hourly fluxes within a single day, specifically, August 21, 2025, in the high marsh. I also included the PAR data for this figure. To review, PAR simply measures how much light is available for plants to use to photosynthesize. I wanted to relate PAR back to the 24-hour measurements to see how different levels of photosynthesis activity influenced gas flux.

Figure 3.

```
# Read in 24-hour fluxes file
df_24 <- read.csv("../Kristine-GHG-Marsh-Work/Data/Script Outputs/8.21.2025/08_21_2025_F

df24_long <- df_24 %>%
  # Gather all CO2 and CH4 columns into long form
  pivot_longer(
    cols = matches("_(CO2|CH4)$"),                # select columns that end in _CO2 or
    names_to = c("model_metric", "gas"),          # split into model_metric and gas
    names_pattern = "(.*)_(CO2|CH4)$",
    values_to = "value"
  ) %>%
  # Spread them back out so each model_metric becomes its own column again
  pivot_wider(
    names_from = model_metric,
    values_from = value
  )

df_24 <- df24_long

# Filter
df_24_filt <- df_24 %>%
  filter(lin_r.squared >= 0.33 & lin_p.value <= 0.05)

df_24 <- df_24_filt

# Adding a column where linear CH4 flux estimate is converted to umol m-2 s-1 from nmo
df_24 <- df_24 %>%
  mutate(lin_flux.estimate_umol = ifelse(
    gas == "CH4",
    lin_flux.estimate / 1000,  # convert CH4 from nmol to µmol
    lin_flux.estimate          # leave CO2 as is
  ))
```

```r
# Data prep for plot
# Summarize mean and SE flux by hour for each gas
df_summary <- df_24 %>%
  group_by(gas, hour) %>%
  summarize(
    mean_flux = mean(lin_flux.estimate_umol, na.rm = TRUE),
    se_flux = sd(lin_flux.estimate_umol, na.rm = TRUE) / sqrt(n())
  )
```

```
## `summarise()` has grouped output by 'gas'. You can override using the `.groups`
## argument.
```

```r
# Split into CO2 and CH4 data frames for convenience
df_co2 <- df_summary %>% filter(gas == "CO2")
df_ch4 <- df_summary %>% filter(gas == "CH4")
```

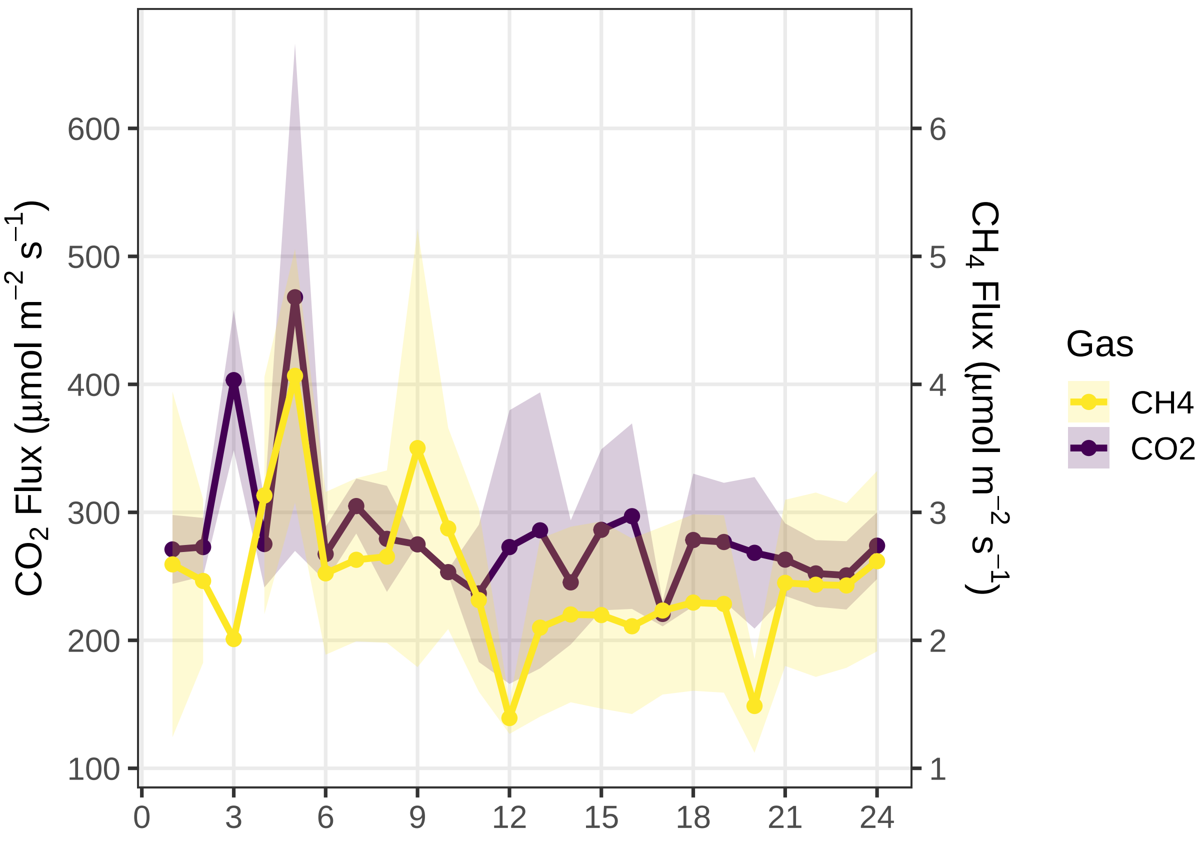
```r
# Reading in met tower data to use PAR
df_env <- read.csv("../Kristine-GHG-Marsh-Work/Data/Not mine/SSENRR Environmental Data/T

# Create a new column called ts for POSIXct timestamp
df_env <- df_env %>%
  mutate(ts = as.POSIXct(DateTimeStamp, format = "%m/%d/%Y %H:%M")) %>%
  mutate(hour = hour(ts) + 1)

# Only PAR along with timestamp columns
df_PAR <- df_env %>%
  select(DateTimeStamp, ts, hour, TotPAR)

# Get the mean of each hour
df_PAR_mean <- df_PAR %>%
  group_by(hour) %>%
  summarise(meanPAR = mean(TotPAR))

# Remove weird NA row
df_PAR_mean <- df_PAR_mean[-25,]
```

```r
# Flux plot (hourly)
fluxplot <- ggplot() +
  # CO2 (left y-axis)
  geom_ribbon(data = df_co2,
              aes(x = hour, ymin = mean_flux - se_flux, ymax = mean_flux + se_flux, fill
              alpha = 0.2) +
  geom_line(data = df_co2,
            aes(x = hour, y = mean_flux, color = "CO2"),
            size = 1.2) +
geom_point(data = df_co2,
             aes(x = hour, y = mean_flux, color = "CO2"),
             size = 2) +

# CH4 (rescaled to right axis)
geom_ribbon(data = df_ch4,
              aes(x = hour, ymin = (mean_flux - se_flux) * 100, ymax = (mean_flux + se_f
              alpha = 0.2) +
geom_line(data = df_ch4,
            aes(x = hour, y = mean_flux * 100, color = "CH4"),
            size = 1.2) +
geom_point(data = df_ch4,
             aes(x = hour, y = mean_flux * 100, color = "CH4"),
             size = 2) +

# Axes
scale_x_continuous(breaks = seq(0, 24, by = 3)) +
scale_y_continuous(
  name = expression("CO"[2]*" Flux ("*mu*"mol m"^{-2}*" s"^{-1}*")"),
  sec.axis = sec_axis(~./100, name = expression("CH"[4]*" Flux ("*mu*"mol m"^{-2}*" s
) +

# Color and fill
scale_color_manual(
  name = "Gas",
  values = c("CO2" = viridis(2)[1], "CH4" = viridis(2)[2])
) +
scale_fill_manual(
  name = "Gas",
  values = c("CO2" = viridis(2)[1], "CH4" = viridis(2)[2])
) +

# Labels and theme
theme_bw(base_size = 14) +
theme(
  axis.text = element_text(size = 12),
  axis.title = element_text(size = 14),
  panel.grid.minor = element_blank(),
  legend.position = "right",
  legend.title = element_text(size = 14),
  legend.text = element_text(size = 12)
) +
  labs(x = NULL)
```

```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use `linewidth` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
fluxplot
```

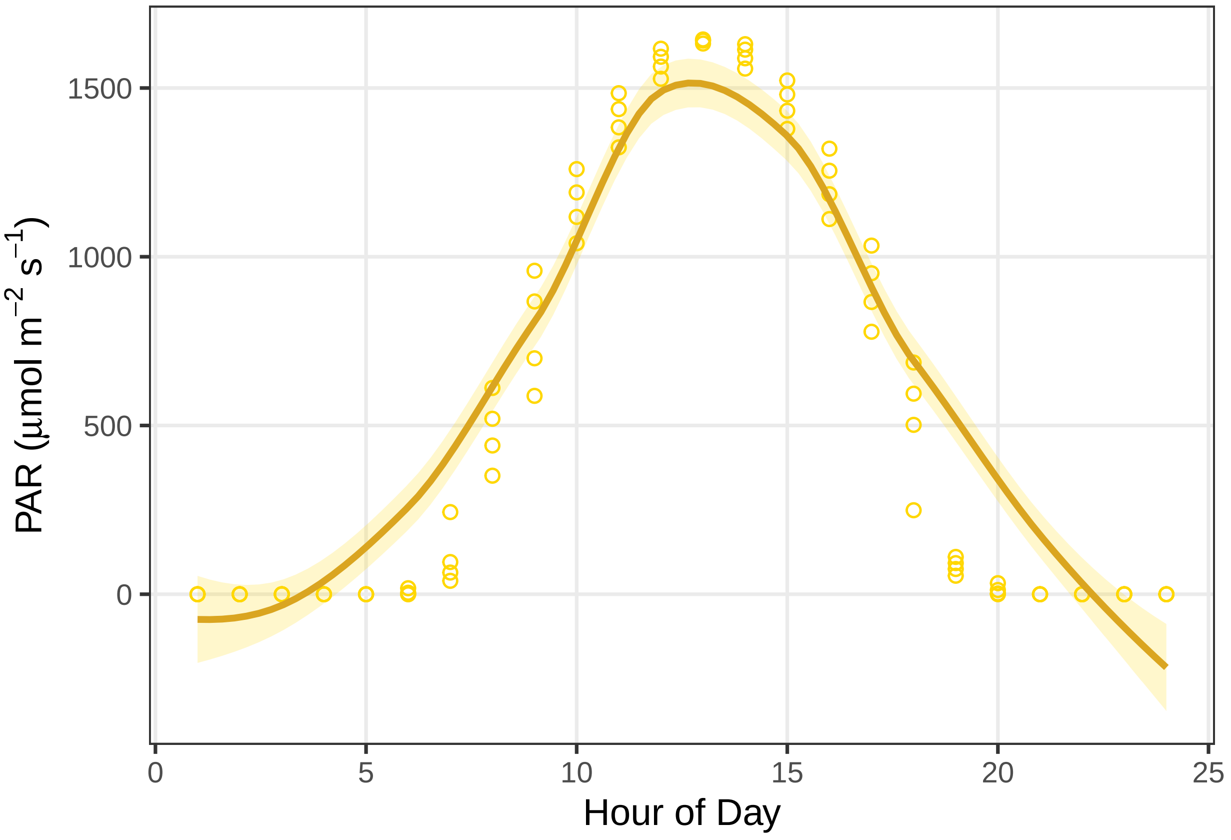
```
## Warning: Removed 1 row containing missing values or values outside the scale range
## (`geom_ribbon()`).
```



```
# Plotting PAR on 8/21/2025
PARplot <- ggplot(df_PAR, aes(x = hour, y = TotPAR)) +
  geom_point(shape = 1, fill = "gold", color = "gold", size = 2) +
  geom_smooth(method = "loess", color = "goldenrod", fill = "gold", alpha = 0.2, linewid
  labs(
    x = "Hour of Day",
```

```r
    y = expression("PAR ("*mu*"mol m"^{-2}*" s"^{-1}*")")
  ) +
  theme_bw(base_size = 14) +
  theme(
    panel.grid.minor = element_blank(),
    legend.position = "none"
  )
PARplot
```

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

## Warning: Removed 2 rows containing non-finite outside the scale range
## (`stat_smooth()`).

## Warning: Removed 2 rows containing missing values or values outside the scale range
## (`geom_point()`).
```



```r
# Combine with cowplot
combined_plot <- plot_grid(
  fluxplot,
  PARplot,
  ncol = 1,
  align = "v",
  rel_heights = c(2, 1)
)
```

```
## Warning: Removed 1 row containing missing values or values outside the scale range
## (‘geom_ribbon()‘).

## ‘geom_smooth()‘ using formula = ’y ~ x’

## Warning: Removed 2 rows containing non-finite outside the scale range
## (‘stat_smooth()‘).

## Warning: Removed 2 rows containing missing values or values outside the scale range
## (‘geom_point()‘).
```

```
combined_plot
```

Hypothesis test using a statistical test

We used linear mixed models to address key questions about gas flux. Linear mixed models are useful here because our data include repeated measurements over time and grouped observations by site, allowing us to account for these sources of variability. Specifically, we asked (1) does flux differ across elevations? (analyzed separately for CO_2 and CH_4) and (2) does flux differ across sampling dates within each elevation? (also analyzed separately for each gas).

Data prep

```
# Creating masterfile without 24 hour data
all_csvs <- list.files(
  path = "../Kristine-GHG-Marsh-Work/Data/Script Outputs/",
  pattern = "Trimmed\\.csv$",       # only CSV files
  recursive = TRUE,           # look into subfolders
  full.names = TRUE           # include full paths
)
all_csvs <- all_csvs[-3] # Remove 8/21/2025

combined_fluxes <- all_csvs %>%
  map_dfr(read_csv, .id = "source_file")
```

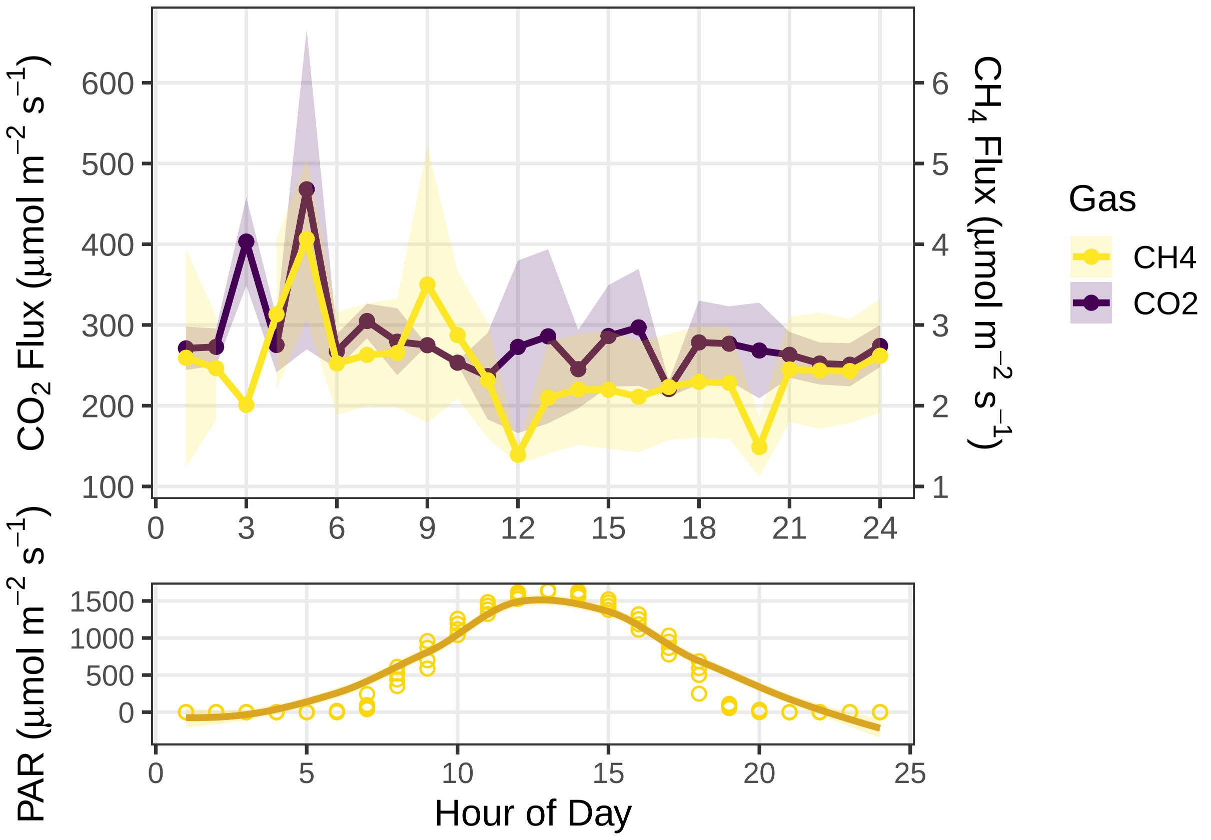


Figure 3: In the high marsh, CO2 and CH4 show distinct daily patterns: CO2 peaks before photosynthesis and declines once it starts, suggesting different influences of photosynthetic activity on the two gases.

```
## New names:
## Rows: 9 Columns: 47
## -- Column specification
## -------------------------------------------------------- Delimiter: "," chr
## (1): site dbl (41): ...1, HM81_AIC_CO2, HM81_flux.estimate_CO2,
## HM81_p.value_CO2, HM8... lgl (2): rob_converged_CO2, rob_converged_CH4 dttm
## (3): TIMESTAMP, TIMESTAMP_min, TIMESTAMP_max
## i Use ‘spec()‘ to retrieve the full column specification for this data. i
## Specify the column types or set ‘show_col_types = FALSE‘ to quiet this message.
## New names:
## Rows: 9 Columns: 47
## -- Column specification
## -------------------------------------------------------- Delimiter: "," chr
## (1): site dbl (41): ...1, HM81_AIC_CO2, HM81_flux.estimate_CO2,
## HM81_p.value_CO2, HM8... lgl (2): rob_converged_CO2, rob_converged_CH4 dttm
## (3): TIMESTAMP, TIMESTAMP_min, TIMESTAMP_max
## i Use ‘spec()‘ to retrieve the full column specification for this data. i
## Specify the column types or set ‘show_col_types = FALSE‘ to quiet this message.
## New names:
## Rows: 9 Columns: 47
## -- Column specification
## -------------------------------------------------------- Delimiter: "," chr
## (1): site dbl (41): ...1, HM81_AIC_CO2, HM81_flux.estimate_CO2,
## HM81_p.value_CO2, HM8... lgl (2): rob_converged_CO2, rob_converged_CH4 dttm
## (3): TIMESTAMP, TIMESTAMP_min, TIMESTAMP_max
## i Use ‘spec()‘ to retrieve the full column specification for this data. i
## Specify the column types or set ‘show_col_types = FALSE‘ to quiet this message.
## New names:
## Rows: 9 Columns: 47
## -- Column specification
## -------------------------------------------------------- Delimiter: "," chr
## (1): site dbl (41): ...1, HM81_AIC_CO2, HM81_flux.estimate_CO2,
## HM81_p.value_CO2, HM8... lgl (2): rob_converged_CO2, rob_converged_CH4 dttm
## (3): TIMESTAMP, TIMESTAMP_min, TIMESTAMP_max
## i Use ‘spec()‘ to retrieve the full column specification for this data. i
## Specify the column types or set ‘show_col_types = FALSE‘ to quiet this message.
## New names:
## Rows: 6 Columns: 47
## -- Column specification
## -------------------------------------------------------- Delimiter: "," chr
## (1): site dbl (41): ...1, HM81_AIC_CO2, HM81_flux.estimate_CO2,
## HM81_p.value_CO2, HM8... lgl (2): rob_converged_CO2, rob_converged_CH4 dttm
## (3): TIMESTAMP, TIMESTAMP_min, TIMESTAMP_max
## i Use ‘spec()‘ to retrieve the full column specification for this data. i
## Specify the column types or set ‘show_col_types = FALSE‘ to quiet this message.
## * `` -> `...1`
```

```
combined_fluxes <- combined_fluxes %>%
  rename("sample_num" = "...1")

# Converts flux data in long format from wide format

combined_fluxes_long <- combined_fluxes %>%
  # Gather all CO2 and CH4 columns into long form
  pivot_longer(
    cols = matches("_(CO2|CH4)$"), # select columns that end in _CO2 or _CH4
    names_to = c("model_metric", "gas"), # split into model_metric and gas
    names_pattern = "(.*)_(CO2|CH4)$",
    values_to = "value"
  ) %>%
  # Spread them back out so each model_metric becomes its own column again
  pivot_wider(
    names_from = model_metric,
    values_from = value
  )

df <- combined_fluxes_long

# Replace anywhere in site column where LM1retake exists to LM1
df <- df %>%
  mutate(site = ifelse(site == "LM1retake", "LM1", site))

# Adding elevation column
df <- df %>%
     mutate(elevation = case_when(
     grepl("^HM", site) ~ "H",
     grepl("^MM", site) ~ "M",
     grepl("^LM", site) ~ "L"
   ))

df$elevation <- as.factor(df$elevation)

# Adding a column where linear CH4 flux estimate is converted to umol m-2 s-1 from nmo
df <- df %>%
  mutate(lin_flux.estimate_umol = ifelse(
    gas == "CH4",
    lin_flux.estimate / 1000,  # convert CH4 from nmol → µmol
    lin_flux.estimate          # leave CO2 as is
  ))
```

```r
# Data cleaning
df_filt <- df %>%
  filter(lin_r.squared >= 0.33 & lin_p.value <= 0.05)

df <- df_filt

# Removes irrelevant columns (i.e. we only care about linear stuff)
df <- df[ ,-c(8:12, 21:28)]

# Add a date column
df <- df %>%
  mutate(date = as.Date(TIMESTAMP))
unique(df$date)
```

```
## [1] "2025-08-11" "2025-08-12" "2025-08-04" "2025-09-19" "2025-09-03"
```

```r
df_co2 <- df %>%
  filter(gas == "CO2")

df_ch4 <- df %>%
  filter(gas == "CH4")

df_co2$date <- as.factor(df_co2$date)
df_ch4$date <- as.factor(df_ch4$date)
```

Figure 1 fundamental question: Does flux differ across elevations? (2 separate models)

```r
# Baseline model
M1 <- lme(lin_flux.estimate_umol ~ elevation,
                         random = list(
                           site = pdDiag(~1),
                           date = pdDiag(~1)
                         ),
                         method = "REML",
                         data = df_co2)

M2 <- lme(lin_flux.estimate_umol ~ elevation,
                         random = list(
                           site = pdDiag(~1),
                           date = pdDiag(~1)
                         ),
                         method = "REML",
                         data = df_ch4)
```

```r
# No test for collinearity since there's only 1 predictor

anova(M1)
```

```
##             numDF denDF   F-value p-value
## (Intercept)     1    30 19.581793  0.0001
## elevation       2     6  1.616594  0.2744
```

```r
anova(M2)
```

```
##             numDF denDF   F-value p-value
## (Intercept)     1    33 15.864683  0.0004
## elevation       2     6  2.745447  0.1424
```

```r
# Tukey post-hoc for M1 (CO2)
emmeans_M1 <- emmeans(M1, pairwise ~ elevation, adjust = "tukey")
emmeans_M1
```

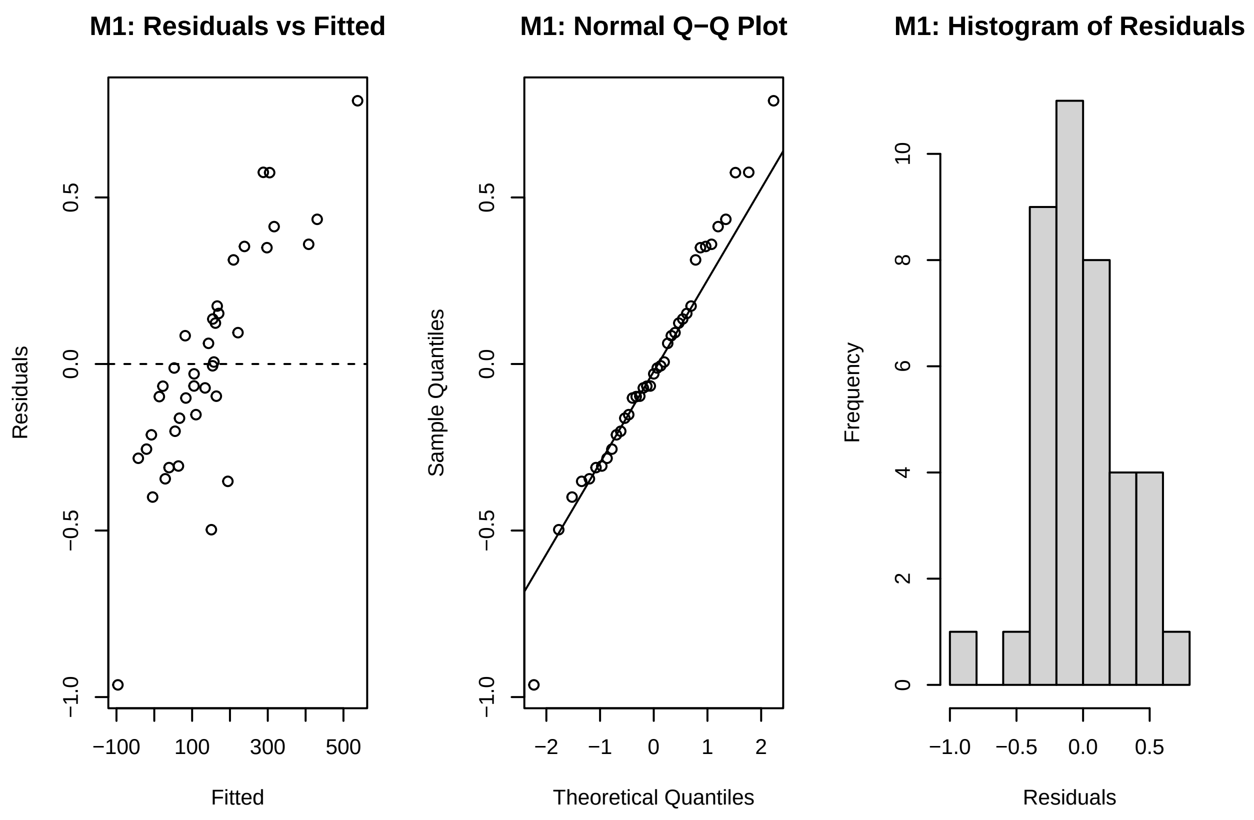
```
## $emmeans
##  elevation emmean   SE df lower.CL upper.CL
##  H          216.3 53.0  8     94.0      339
##  L           93.9 56.7  6    -44.9      233
##  M          101.4 54.6  6    -32.3      235
##
## Degrees-of-freedom method: containment
## Confidence level used: 0.95
##
## $contrasts
##  contrast estimate   SE df t.ratio p.value
##  H - L      122.48 77.6  6   1.578  0.3245
##  H - M      114.96 76.2  6   1.510  0.3515
##  L - M       -7.52 78.7  6  -0.095  0.9950
##
## Degrees-of-freedom method: containment
## P value adjustment: tukey method for comparing a family of 3 estimates
```

```r
# Tukey post-hoc for M2 (CH4)
emmeans_M2 <- emmeans(M2, pairwise ~ elevation, adjust = "tukey")
emmeans_M2
```

```
## $emmeans
##  elevation emmean    SE df lower.CL upper.CL
##  H          3.081 0.730  8    1.397     4.77
##  L          0.995 0.739  6   -0.813     2.80
##  M          0.966 0.730  6   -0.821     2.75
##
## Degrees-of-freedom method: containment
## Confidence level used: 0.95
##
## $contrasts
##  contrast estimate   SE df t.ratio p.value
##  H - L      2.0855 1.04  6   2.007  0.1913
##  H - M      2.1144 1.03  6   2.047  0.1818
##  L - M      0.0288 1.04  6   0.028  0.9996
##
## Degrees-of-freedom method: containment
## P value adjustment: tukey method for comparing a family of 3 estimates
```

Model Validation

```r
# M1: CO2
par(mfrow = c(1, 3))   # Show 3 plots side-by-side

res1 <- resid(M1, type = "normalized")
fit1 <- fitted(M1)

plot(fit1, res1,
     main = "M1: Residuals vs Fitted",
     xlab = "Fitted", ylab = "Residuals")
abline(h = 0, lty = 2)

qqnorm(res1, main = "M1: Normal Q-Q Plot")
qqline(res1)

hist(res1, main = "M1: Histogram of Residuals",
     xlab = "Residuals")
```



```r
# -----------------------------------------------------

# M2: CH4
par(mfrow = c(1, 3))

res2 <- resid(M2,
              type = "normalized")
fit2 <- fitted(M2)

plot(fit2, res2,
     main = "M2: Residuals vs Fitted",
     xlab = "Fitted",
     ylab = "Residuals")
abline(h = 0, lty = 2)

qqnorm(res2,
       main = "M2: Normal Q-Q Plot")
qqline(res2)

hist(res2,
     main = "M2: Histogram of Residuals",
    xlab = "Residuals")
```

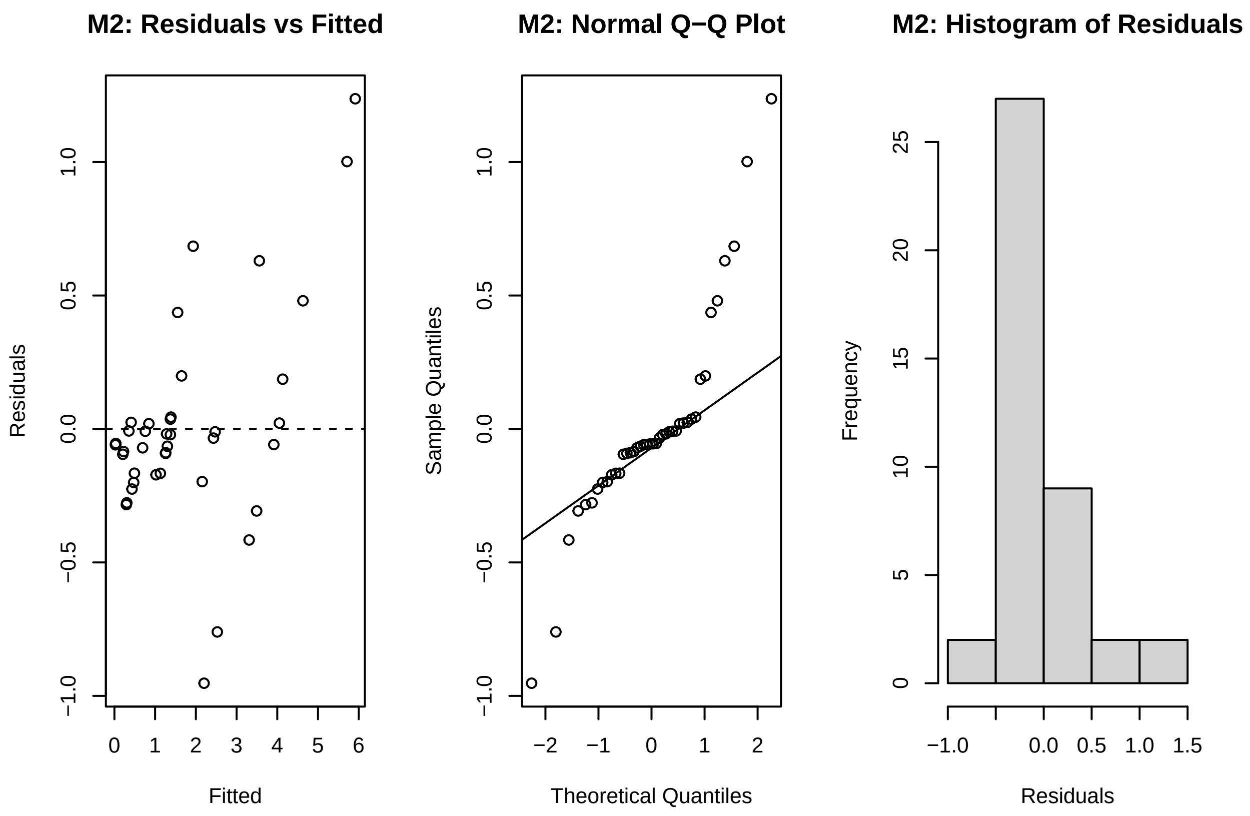


Figure 2 fundamental question: Does flux differ across the sampling dates for each elevation? (2 separate models)

```
# Baseline model
# No random effects since site and elevation are confounding?

M1 <- lm(lin_flux.estimate_umol ~ elevation * date, data = df_co2)

M2 <- lm(lin_flux.estimate_umol ~ elevation * date, data = df_ch4)


anova(M1)
```

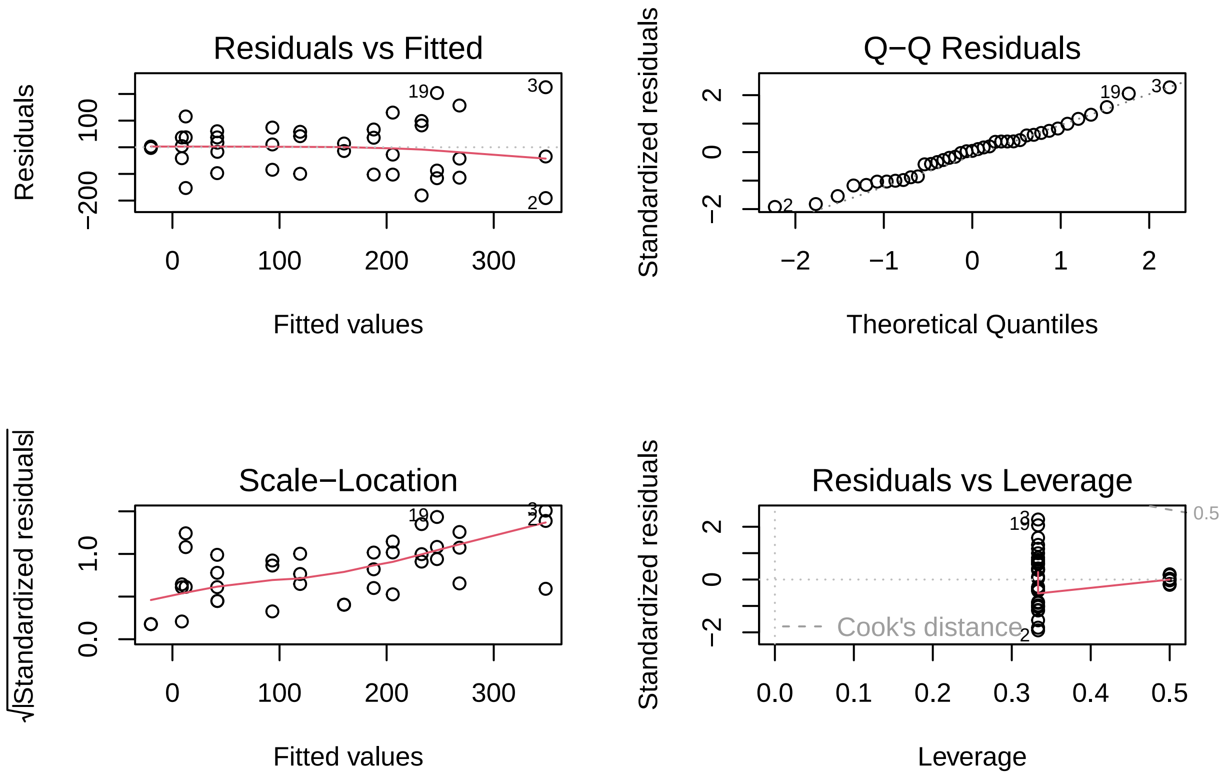
```
## Analysis of Variance Table
##
## Response: lin_flux.estimate_umol
##                 Df Sum Sq Mean Sq F value  Pr(>F)
## elevation        2 123638   61819  4.1995 0.02677 *
## date             4 143252   35813  2.4328 0.07391 .
## elevation:date  7 200162   28595  1.9425 0.10476
## Residuals      25 368019   14721
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
anova(M2)
```

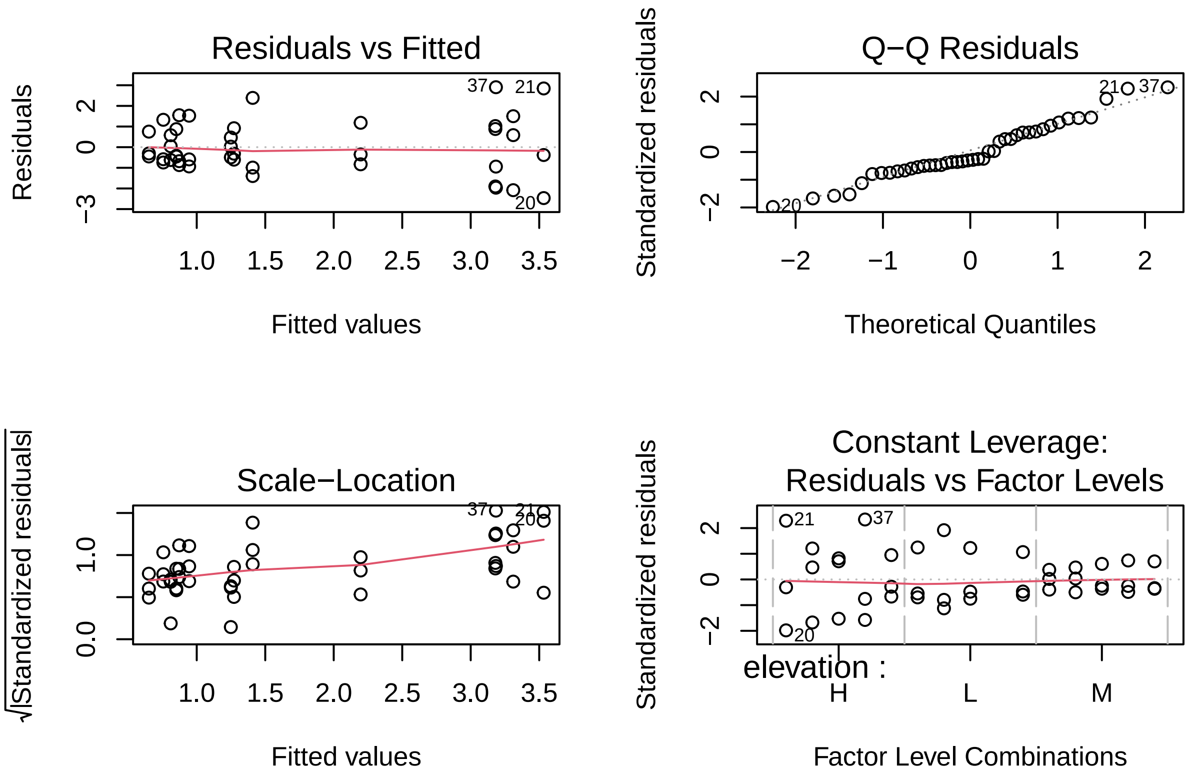
```
## Analysis of Variance Table
##
## Response: lin_flux.estimate_umol
##                 Df Sum Sq Mean Sq F value    Pr(>F)
## elevation        2 42.594 21.2969  9.1687 0.0008653 ***
## date             4  2.488  0.6221  0.2678 0.8961659
## elevation:date   7  2.359  0.3369  0.1451 0.9933567
## Residuals       28 65.038  2.3228
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Model validation

```
# M1: CO2
# Basic diagnostic plots for M1 (CO2)
par(mfrow = c(2,2))
plot(M1)
```



```
# Reset plotting layout
par(mfrow = c(1,1))

# Basic diagnostic plots for M2 (CH4)
par(mfrow = c(2,2))
plot(M2)
```



```r
par(mfrow = c(1,1))
```

Conclusion

Both CO_2 and CH_4 fluxes tended to be higher in the high marsh than in the mid and low marsh, although these differences were not statistically significant when averaging all measurments for each elevation (Fig 1). These general trends may reflect underlying differences in vegetation structure. Specifically the high and mid marshes are more densely vegetated, whereas the low marsh is largely bare, which may limit gas production and transport. Plants have specialized tissues called aerenchyma that facilitate gas exchange between the roots and the air in waterlogged plants. In wetland systems, aerenchyma have been shown to facilitate CH_4 transport. More vegetated areas might result in this sort of plant mediated transport. However, this is likely not the whole picture since low (barely vegetated) marsh and mid (similar vegetation to high marsh) have similar fluxes. Future work is needed to identify other factors that might be influencing this pattern.

When taking into account variability by looking across elevation and all the sampling dates (this time averaged by each day), the effect of elevation on flux becomes significant. Specifically, in the mixed-effects models, CO_2 flux increased significantly with elevation ($p = 0.027$) and showed a marginal trend across sampling dates ($p = 0.074$), suggesting some temporal sensitivity (Fig 2). CH_4 flux, in contrast, was significantly higher in the high marsh than in the mid and low marsh ($p < 0.001$) and remained relatively stable across dates, with no elevation and sampling date interaction detected for either gas (Fig 2). These patterns indicate that CH_4 emissions are likely primarily driven by marsh elevation, while CO_2 flux is shaped by both elevation and short-term environmental variability.For instance, CO_2 might be more sensitive to changes in temperature or water saturation.

Additionally, diel patterns suggest that photosynthesis may influence these gases differently: CO_2 flux tends to peak before photosynthesis begins and declines once photosynthetic activity starts, whereas CH_4 flux does not show the same response (Fig 3). Since we only have net ecosystem exchange, more work is needed to also consider the influence of microbial respiration on these patterns. I am currently planning on conducting assays to determine which microbes live within each elevation and how much of each microbe to further contextualize my initial results.

Cornu, C. E., & Sadro, S. (2002). Physical and Functional Responses to Experimental Marsh Surface Elevation Manipulation in Coos Bay's South Slough. *Restoration Ecology*, *10*(3), 474–486. https://doi.org/10.1046/j.1526-100X.2002.01035.x

Muñoz, I., Schmidt, J., & Weidema, B. P. (2024). The indirect global warming potential of methane oxidation in the IPCC's sixth assessment report. *Environmental Research Letters*, *19*(12), 121002. https://doi.org/10.1088/1748-9326/ad8edd

Salimi, S., Almuktar, S. A. A. A. N., & Scholz, M. (2021). Impact of climate change on wetland ecosystems: A critical review of experimental wetlands. *Journal of Environmental Management*, *286*, 112160. https://doi.org/10.1016/j.jenvman.2021.112160

Schultz, M. A., Janousek, C. N., Brophy, L. S., Schmitt, J., & Bridgham, S. D. (2023). How management interacts with environmental drivers to control greenhouse gas fluxes from Pacific Northwest coastal wetlands. *Biogeochemistry*, *165*(2), 165–190. https://doi.org/10.1007/s10533-023-01071-6

Shahan, J., Chu, H., Windham-Myers, L., Matsumura, M., Carlin, J., Eichelmann, E., Stuart-Haentjens, E., Bergamaschi, B., Nakatsuka, K., Sturtevant, C., & Oikawa, P. (2022). Combining Eddy Covariance and Chamber Methods to Better Constrain CO2 and CH4 Fluxes Across a Heterogeneous Restored Tidal Wetland. *Journal of Geophysical Research: Biogeosciences*, *127*(9), e2022JG007112. https://doi.org/10.1029/2022JG007112

Williams, T., Janousek, C. N., McKeon, M. A., Diefenderfer, H. L., Cornu, C. E., Borde, A. B., Apple, J., Brophy, L. S., Norwood, M., Schultz, M. A., & Bridgham, S. D. (2025). Methane and nitrous oxide fluxes from reference, restored, and disturbed estuarine wetlands in Pacific Northwest, USA. *Ecological Applications*, *35*(2), e70011. https://doi.org/10.1002/eap.70011