# Data for: Partitioning inorganic carbon fluxes using paired $O_2$-$CO_2$ sensors in a headwater stream, Costa Rica, Submitted to Biogeochemistry

Nicholas S. Marzolf

Last updated: January 31, 2022

## Contents



### Publication information

Full authorship and affiliations:

Nicholas S. Marzolf[1] *, Gaston E. Small[2], Diana Oviedo-Vargas[3], Carissa N. Ganong[4], John H. Duff[5], Alonso Ramírez[6], Catherine M. Pringle[7], David P. Genereux[8], Marcelo Ardón[1,9]

1 Department of Forestry and Environmental Resources, North Carolina State University, NC, USA (ORCiD: 0000-0001-9146-1643)
2 College of Arts and Sciences, University of St. Thomas, MN, USA (ORDiD: 0000-0002-9018-7555)
3 Stroud Water Research Center, Avondale, PA, USA (ORCiD: 0000-0001-9333-0962)
4 Department of Biology, Missouri Western State University, MO, USA

5 US Geological Survey, Menlo Park, CA, USA
6 Department of Applied Ecology, North Carolina State University, NC, USA (ORCiD: 0000-0001-9985-5719)
7 Odum School of Ecology, University of Georgia, GA, USA (ORCiD: 0000-0002-3522-7315)
8 Marine, Earth, and Atmospheric Sciences, North Carolina State University, NC, USA (ORCiD: 0000-0001-5601-7088)
9 ORCiD: 0000-0001-7275-2672
* Corresponding author: nmarzol@ncsu.edu

## Load packages and explore the data

```
# data manipulation
library(dplyr)
library(tidyr)
library(lubridate)
library(tidyquant)
# plotting
library(ggplot2)
library(cowplot)
library(ggrepel)
library(scales)
library(ggsci)
# stats
library(lmodel2)
library(ellipse)
library(purrr)
library(lsmeans)
library(emmeans)
library(ARTool)
# hydrology
library(hydrostats)
# reproducibility
library(pander)

# Read in hourly data
Tac_all <- readRDS("Tac_all.rds")
```

This file contains hourly data from April 1 - September 27, including: stream and well Vaisala pCO2 sensors (CO2_ppm, wellCO2), YSI sonde (temp.water, pH, DO.obs, DO.sat, cond), discharge (sumQ_m3_hr, meanQ_m3_s), and meterological data (relative humidity (RH), Rainfall, barometric pressure (bp_mmHg)) from the weather station located at La Selva Biological Station. Rainfall is collected at 15-minute intervals, and there data here are aggregated to hourly sums. Discharge is collected at 15-minute intervals, and is shown as both mean discharge for the hour (m3/s) and sum of the 4 contributing measurements (m3/hr). Groundwater discharge (GW_Q) is described in the text. Mean depth is taken from the weir.

```
glimpse(Tac_all)
```

```
## Rows: 4,324
## Columns: 16
## $ Timestamp  <dttm> 2013-04-01 00:00:00, 2013-04-01 01:00:00, 2013-04-01 02:00~
## $ sumQ_m3_hr <dbl> 0.244, 0.535, 0.617, 0.710, 0.773, 0.834, 0.840, 0.812, 0.7~
## $ meanQ_m3_s <dbl> 0.002033333, 0.002229167, 0.002570833, 0.002958333, 0.00322~
```

```
## $ mean_depth <dbl> 0.24300, 0.24575, 0.25025, 0.25500, 0.25800, 0.26075, 0.261~
## $ GW_Q       <dbl> 0.0174460, 0.0382525, 0.0441155, 0.0507650, 0.0552695, 0.05~
## $ RH         <dbl> 92.850, 93.100, 92.900, 92.600, 93.300, 93.400, 93.350, 93.~
## $ Rainfall   <dbl> 0.254, 0.000, 1.778, 0.000, 1.524, 0.508, 1.016, 1.524, 0.0~
## $ bp_mmHg    <dbl> 755.6687, 755.8967, 756.2283, 756.0831, 755.5855, 755.3016,~
## $ CO2_ppm    <dbl> 6287.696, 6042.019, 6058.962, 6086.072, 6087.767, 6055.574,~
## $ temp.water <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
## $ pH         <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
## $ DO.obs     <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
## $ DO.sat     <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
## $ cond       <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
## $ wellCO2    <dbl> 41074.82, 42818.21, 44179.08, 45055.86, 45396.08, 46122.21,~
## $ PAR        <dbl> 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0~
```

We will use this dataset build on with the following calculations at the hourly resolution and aggregate to daily scale at the end of the document.

## Define constants

These constants define aspects of Taconazo and the lower study reach in this study. The widths were measured during propane injections from Oviedo-Vargas et al. (2015).

```
Tac_length = 1587.4       # total length of the main-stem Taconazo
Tac_reach = 75            # m
Tac_width_upper_dry = 1.4 # m
Tac_width_upper_wet = 1.8 # m
Tac_width_lower_dry = 1.5 # m
Tac_width_lower_wet = 2.8 # m
Tac_width_lower_dry = 1.5 # m
Tac_width_lower_wet = 2.8 # m
Tac_slope = 0.0024        # m/m
```

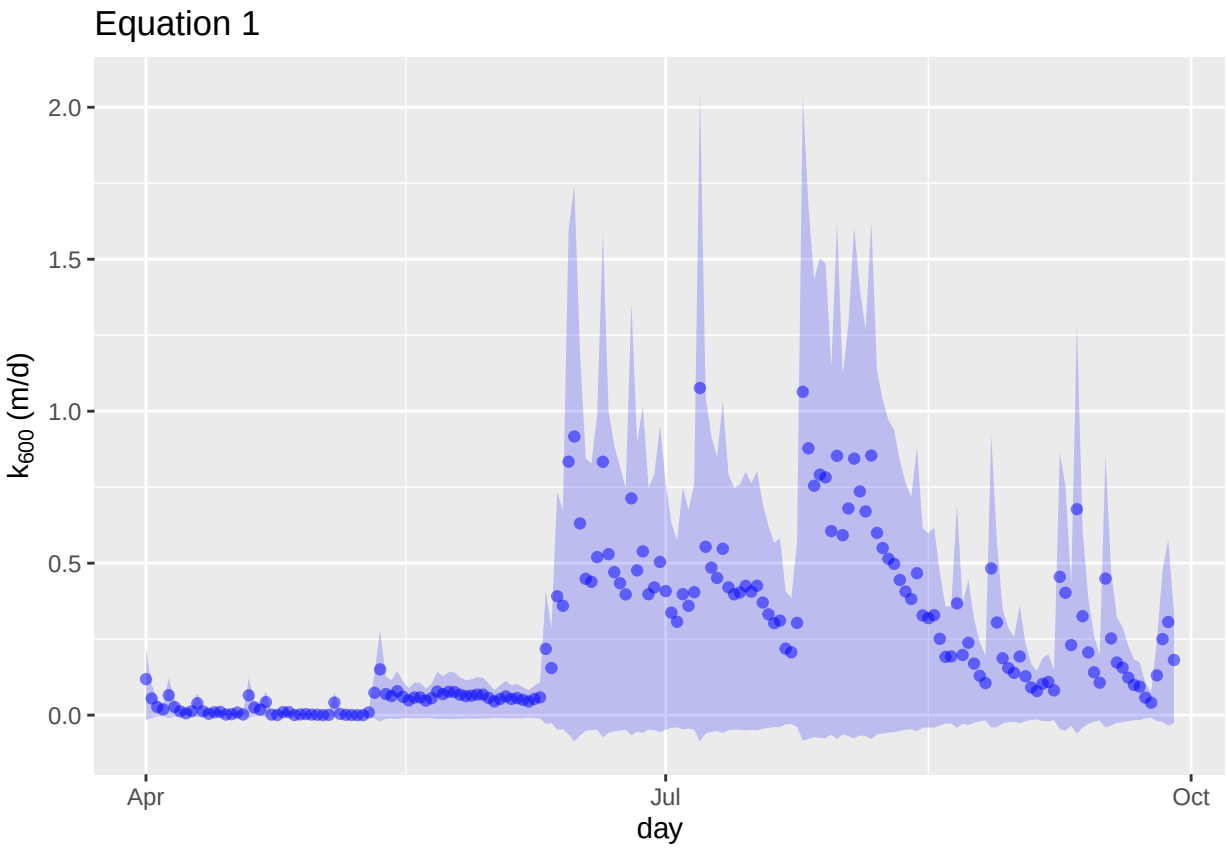
## Gas exchange

To estimate gas exchange, we use the equations from Raymond et al. (2012) best suited for small streams (1 and 2). We first estimate Schmidt numbers for relevant gases, CO[2] and O[2]. We use mean hourly depth to convert between k600 and K600. To determine which equation estimates gas exchange, we compare a rating curve of k600 and discharge using AIC and predict values based on mean hourly discharge.

```
# calculate mean temperature
temp_mean = mean(Tac_all$temp.water,
                 na.rm = TRUE)

# estimate Schmidt numbers
Sc_prop = 3545.60 - (203.41*temp_mean) + (4.78*(temp_mean^2)) - (0.0404*(temp_mean^3))
Sc_CO2 = 1686.08 - (89.66*temp_mean) + (2.07*(temp_mean^2)) - (0.018*(temp_mean^3))
Sc_O2 = 1568 - (86.04*temp_mean) + (2.142*(temp_mean^2)) - (0.0216*(temp_mean^3))

# estimate k600 from equations 1 and 2 from Raymond et al. (2012)
Tac_k600 <- Tac_all %>%
  select(Timestamp, meanQ_m3_s, mean_depth) %>%
```

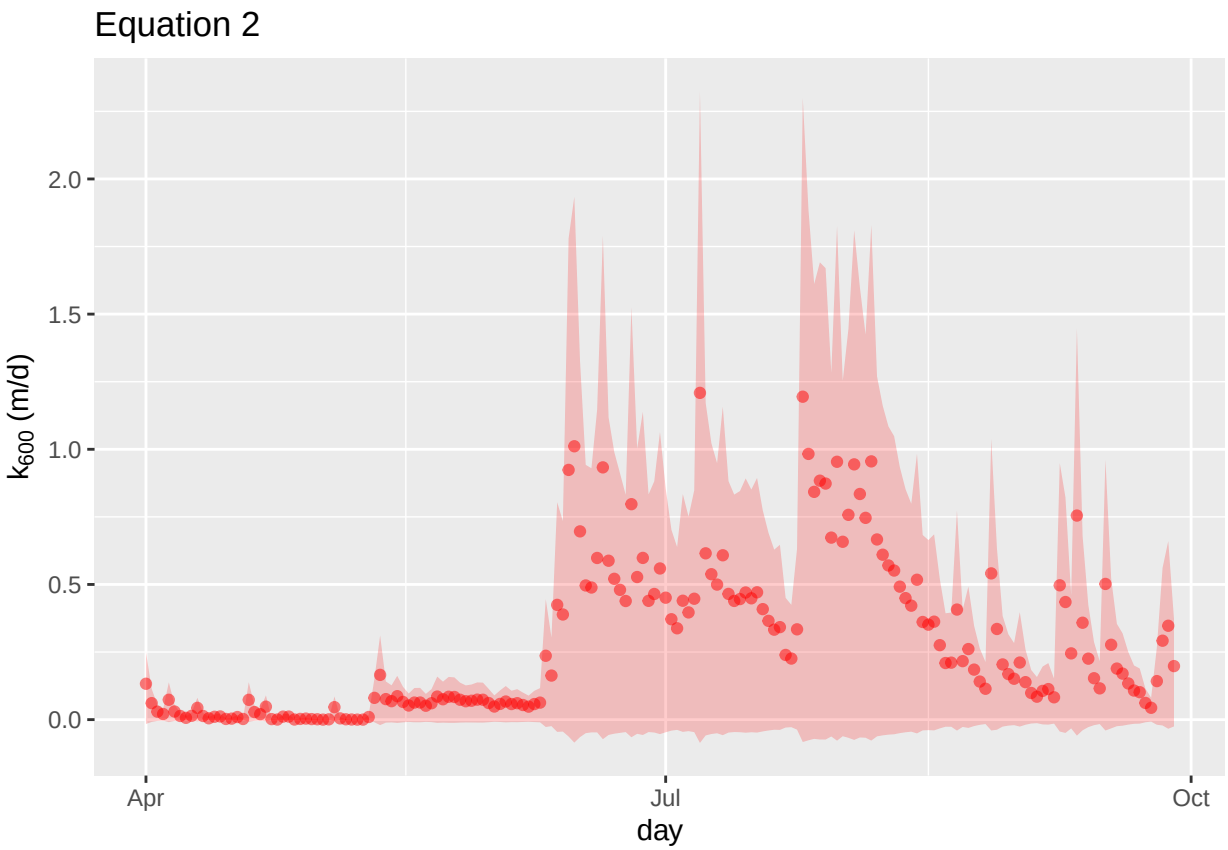
```
  # calculate hourly unit velocity and Froude number
  mutate(
    # velocity and Froude number
    Tac_unit_v = ifelse(month(Timestamp) == 'April',
                        meanQ_m3_s/(mean_depth * Tac_width_lower_dry),
                        meanQ_m3_s/(mean_depth * Tac_width_lower_wet)),
    Tac_froude = Tac_unit_v/(sqrt(9.81*mean_depth)),

    # scaling models from Raymond et al. (2012) and high/low errors
    k600_eq1 = 5037*
      ((Tac_unit_v*Tac_slope)^(0.89))*
      (mean_depth^(0.54)),
    k600_eq1_low = (5037-604)*
      ((Tac_unit_v*Tac_slope)^(0.89-0.02))*
      (mean_depth^(0.54-0.03)),
    k600_eq1_high = (5037+604)*
      ((Tac_unit_v*Tac_slope)^(0.89+0.02))*
      (mean_depth^(0.54+0.03)),

    k600_eq2 = 5937*
      (1 - (2.54*(Tac_froude^2)))*
      ((Tac_unit_v*Tac_slope)^(0.89))*
      (mean_depth^0.58),
    k600_eq2_low = (5937-606)*
      (1 - ((2.54-0.223)*(Tac_froude^2)))*
      ((Tac_unit_v*Tac_slope)^(0.89-0.017))*
      (mean_depth^(0.58-0.027)),
    k600_eq2_high = (5937+606)*
      (1 - ((2.54+0.223)*(Tac_froude^2)))*
      ((Tac_unit_v*Tac_slope)^(0.89+0.017))*
      (mean_depth^(0.58+0.027))
  )

# Mean daily k600 for equations 1 and 2
Tac_k600_sum <- Tac_k600 %>%
  group_by(day = date(Timestamp)) %>%                    # estimate at daily time-steps
  summarise(meanQ = mean(meanQ_m3_s, na.rm = TRUE),      # mean daily discharge
            mean_z = mean(mean_depth, na.rm = TRUE),     # mean daily depth

            # mean gas exchange velocities
            mean_k600_eq1 = mean(k600_eq1, na.rm = TRUE), # estimate EQ1 gas exchange velocity (m/d)
            mean_k600_eq2 = mean(k600_eq2, na.rm = TRUE), # estimate EQ2 gas exchange velocity (m/d)

            # mean error of gas exchange velocities
            mean_k600_eq1_low = mean(k600_eq1_low, na.rm = TRUE),
            mean_k600_eq1_high = mean(k600_eq1_high, na.rm = TRUE),
            mean_k600_eq2_low = mean(k600_eq2_low, na.rm = TRUE),
            mean_k600_eq2_high = mean(k600_eq2_high, na.rm = TRUE)
  )

plot_k600_eq1 <- ggplot(Tac_k600_sum,
                        aes(x = day))+
  geom_point(aes(y = mean_k600_eq1, color = 'Equation 1'),
```

```
            alpha = 0.5)+

  geom_ribbon(alpha = 0.2,
              aes(ymin = mean_k600_eq1 - mean_k600_eq1_low,
                  ymax = mean_k600_eq1 + mean_k600_eq1_high,
                  fill = 'Equation 1'))+
  scale_color_manual(values = 'blue')+
  scale_fill_manual(values = 'blue')+
  ylab(expression(paste(k[600], ' (m/d)')))+
  ggtitle('Equation 1')+
  theme(legend.position = 'none')
plot_k600_eq1
```

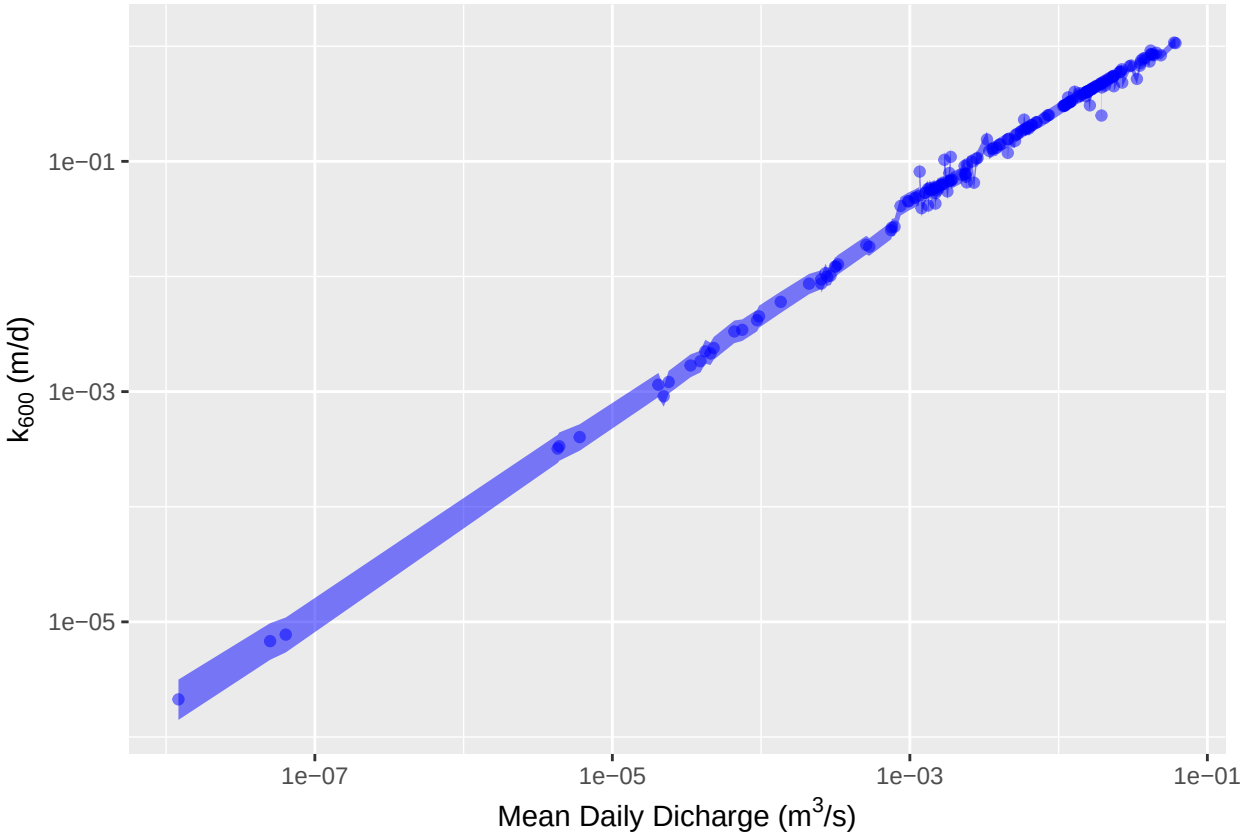



```
plot_k600_eq2 <- ggplot(Tac_k600_sum,
                        aes(x = day))+
  geom_point(aes(y = mean_k600_eq2, color = 'Equation 2'),
             alpha = 0.5)+
  geom_ribbon(alpha = 0.2,
              aes(ymin = mean_k600_eq2 - mean_k600_eq2_low,
                  ymax = mean_k600_eq2 + mean_k600_eq2_high,
                  fill = 'Equation 2'))+
  scale_color_manual(values = 'red')+
  scale_fill_manual(values = 'red')+
  ylab(expression(paste(k[600], ' (m/d)')))+
  ggtitle('Equation 2')+
```

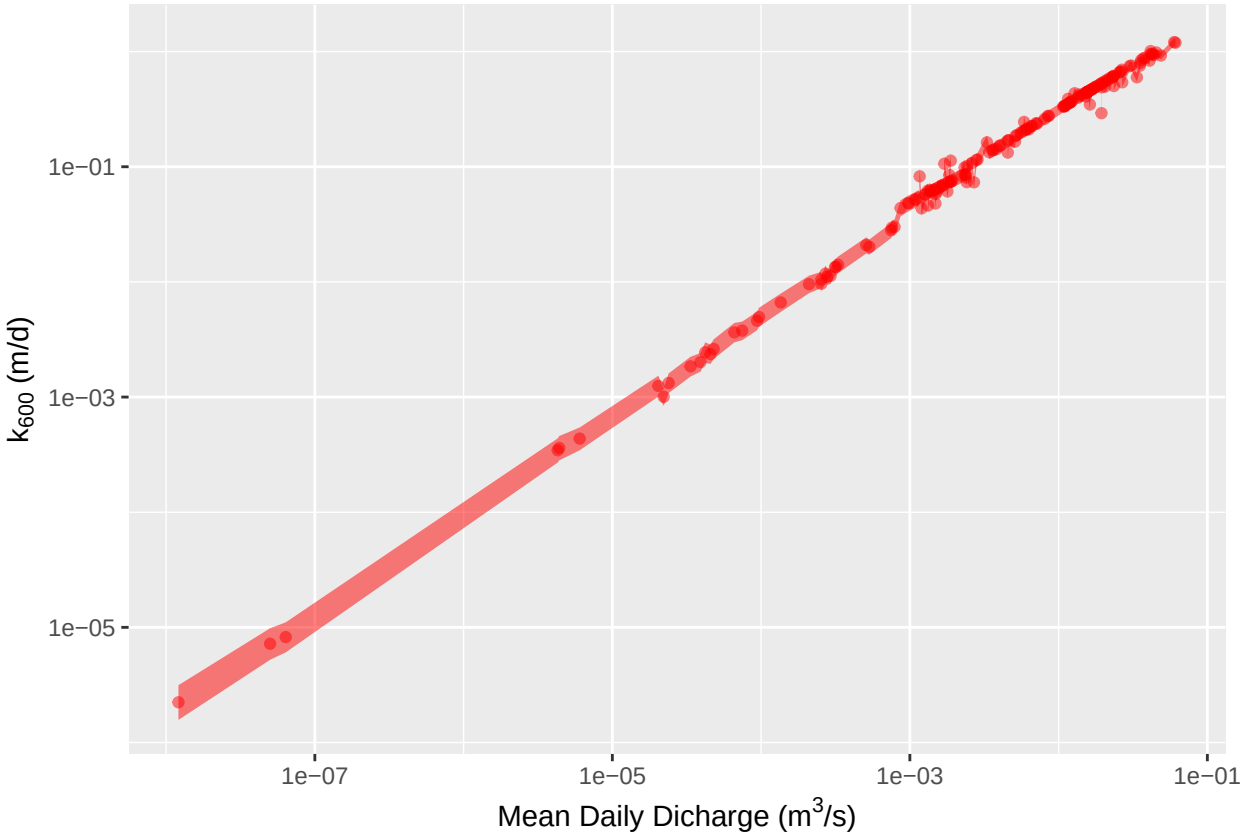
```
  theme(legend.position = 'none')
plot_k600_eq2
```



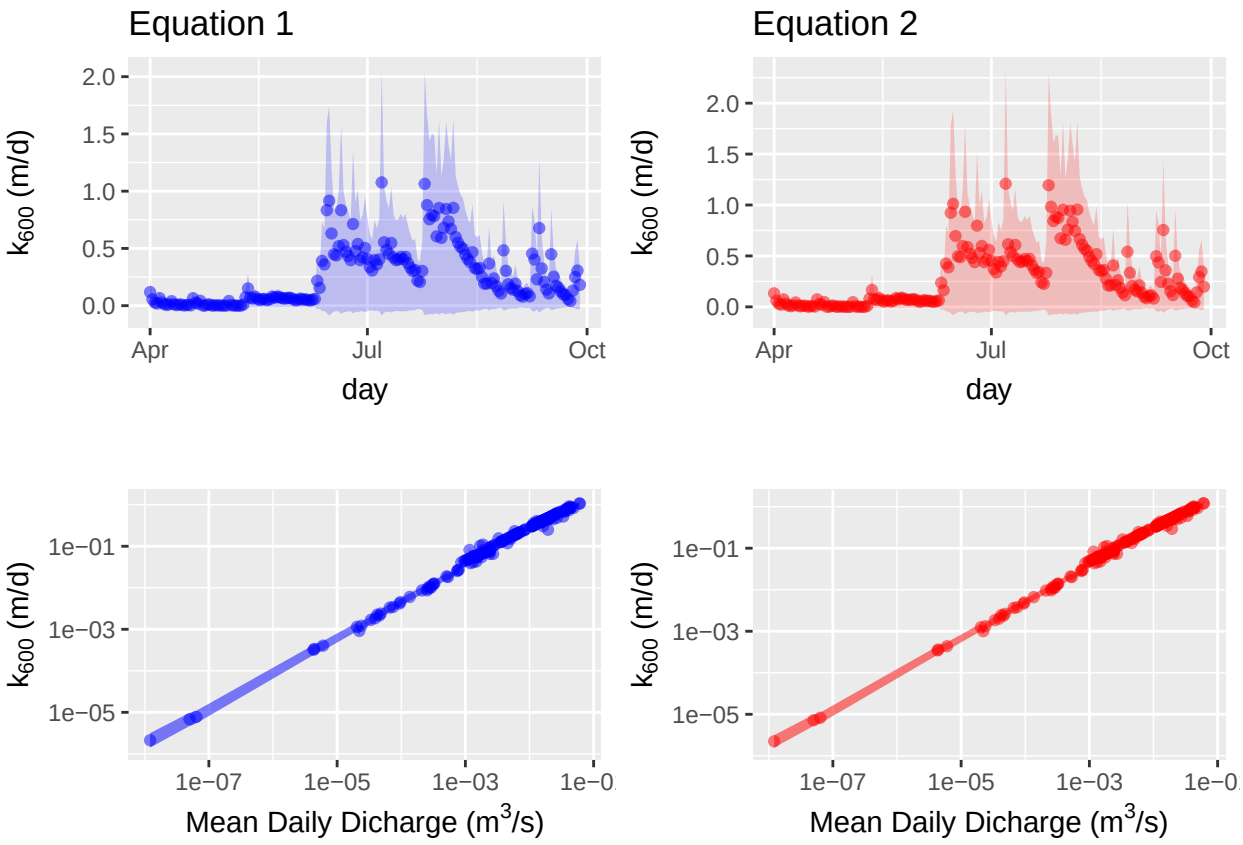

```
# which model fits the data best- plot and compare using AIC
k600_Q_ratingCurve_eq1 <- ggplot()+
  geom_point(data = Tac_k600_sum,
             aes(x = meanQ, y = mean_k600_eq1,
                 color = 'Equation 1'),
             alpha = 0.5)+
  geom_ribbon(data = Tac_k600_sum, alpha = 0.5,
              aes(x = meanQ,
                  ymin = mean_k600_eq1_low,
                  ymax = mean_k600_eq1_high,
                  fill = 'Equation 1'))+
  scale_x_log10()+
  scale_y_log10()+
  scale_color_manual(values = 'blue')+
  scale_fill_manual(values = 'blue')+
  labs(x = expression(paste('Mean Daily Dicharge (', m^3,'/s)')),
       y = expression(paste(k[600], ' (m/d)')))+
  theme(legend.position = 'none')
k600_Q_ratingCurve_eq1
```




```
k600_Q_ratingCurve_eq2 <- ggplot()+
  geom_point(data = Tac_k600_sum,
             aes(x = meanQ, y = mean_k600_eq2,
                 color = 'Equation 2'),
             alpha = 0.5)+
  geom_ribbon(data = Tac_k600_sum, alpha = 0.5,
              aes(x = meanQ,
                  ymin = mean_k600_eq2_low,
                  ymax = mean_k600_eq2_high,
                  fill = 'Equation 2'))+
  scale_color_manual(values = 'red')+
  scale_fill_manual(values = 'red')+
  scale_x_log10()+
  scale_y_log10()+
  labs(x = expression(paste('Mean Daily Dicharge (', m^3,'/s)')),
       y = expression(paste(k[600], ' (m/d)')))+
  theme(legend.position = 'none')
k600_Q_ratingCurve_eq2
```




```
FigS4 <- plot_grid(plot_k600_eq1, plot_k600_eq2,
                   k600_Q_ratingCurve_eq1, k600_Q_ratingCurve_eq2,
                   nrow = 2, ncol = 2,
                   align = 'hv')
FigS4
```




```
# which equation fits the data better
# use AIC to compare k600 ~ Q power law curves
eq1_ratingCurve <- lm(data = Tac_k600_sum,
                      log10(mean_k600_eq1) ~ log10(meanQ))
coef(eq1_ratingCurve)
```

```
##  (Intercept) log10(meanQ)
##    1.1717815    0.8680677
```

```
eq2_ratingCurve <- lm(data = Tac_k600_sum,
                      log10(mean_k600_eq2) ~ log10(meanQ))
coef(eq2_ratingCurve)
```

```
##  (Intercept) log10(meanQ)
##    1.2198808    0.8710842
```

```
AIC(eq1_ratingCurve, eq2_ratingCurve)
```

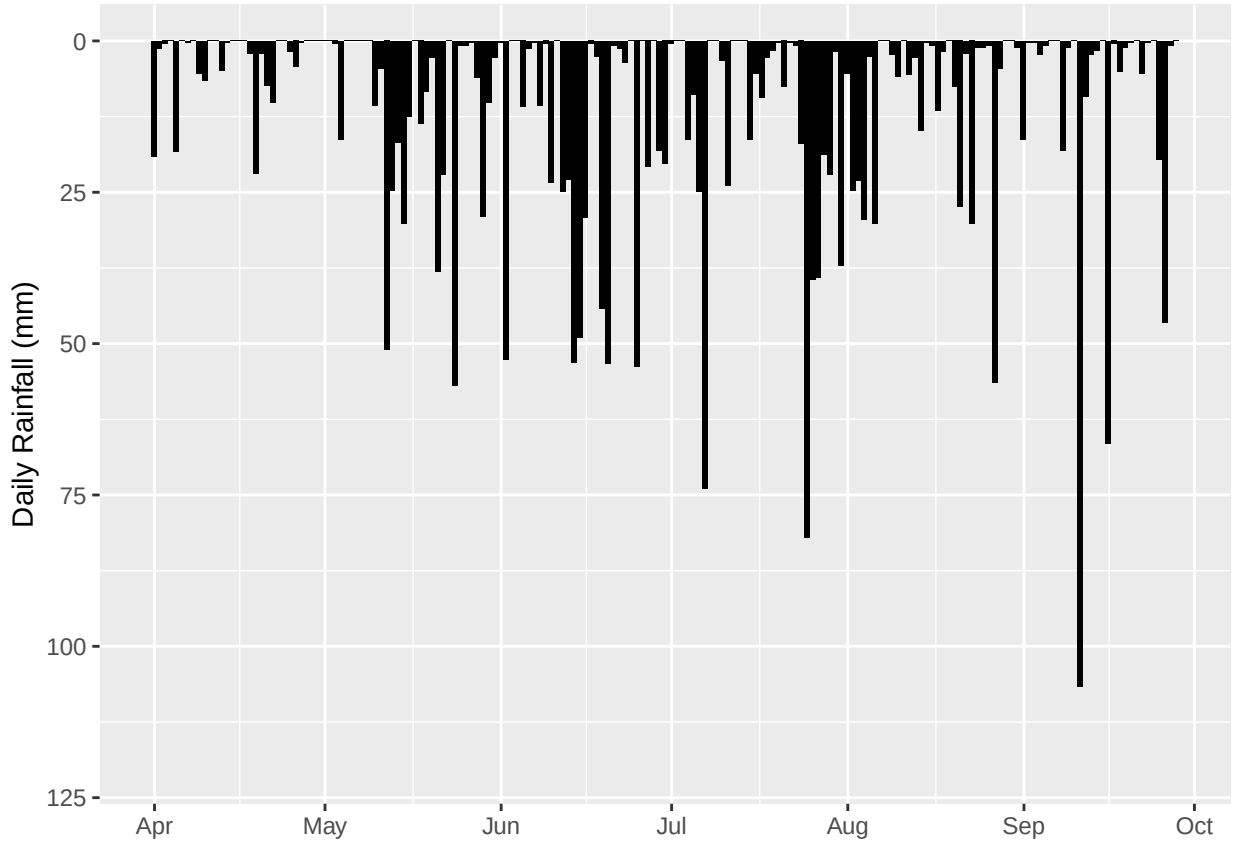
```
##                  df       AIC
## eq1_ratingCurve  3 -453.4729
## eq2_ratingCurve  3 -488.2466
```

```r
# convert k600 to KO2 and KCO2
Tac_gas_data <- Tac_k600_sum %>%
  mutate(
    # convert to kCO2 and kO2
    mean_k_CO2_eq2 = mean_k600_eq2*(600/Sc_CO2)^0.7, # Schmidt scaling to convert k600 to kCO2
    mean_k_O2_eq2 = mean_k600_eq2*(600/Sc_O2)^0.7,   # Schmidt scaling to convert k600 to kO2

    # calculate first order coefficients
    mean_K600_eq2 = mean_k600_eq2/mean_z,
    mean_K_CO2_eq2 = mean_k_CO2_eq2/mean_z,
    mean_K_O2_eq2 = mean_k_O2_eq2/mean_z
  ) %>%
  # select the data we need
  select(day, meanQ, mean_z,
         mean_k600_eq2, mean_k600_eq2_high, mean_k600_eq2_low,
         mean_K600_eq2, mean_K_CO2_eq2, mean_K_O2_eq2)
```

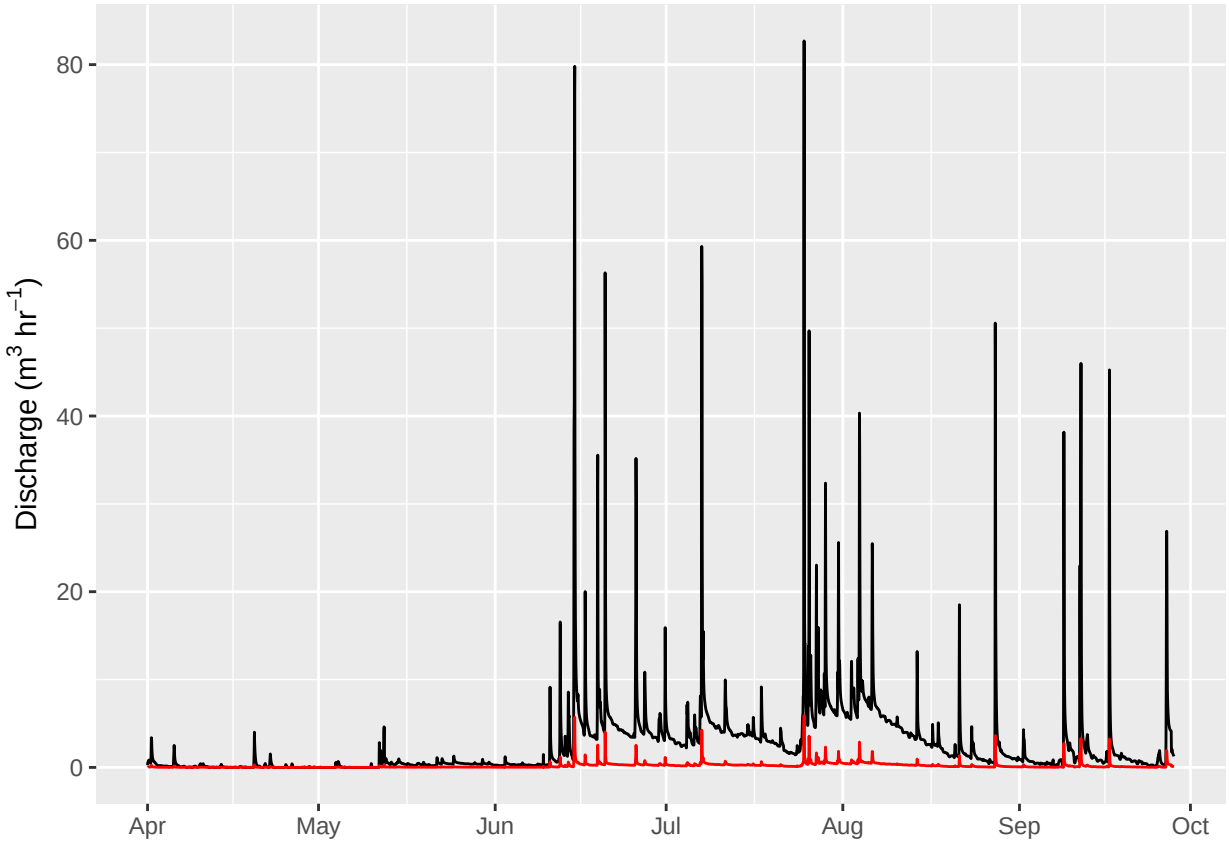
## Figure 2

Figure 2 details the stream measurements. We plot rainfall, discharge, groundwater discharge, dissolved oxygen, CO2 in the stream and in the riparian well.

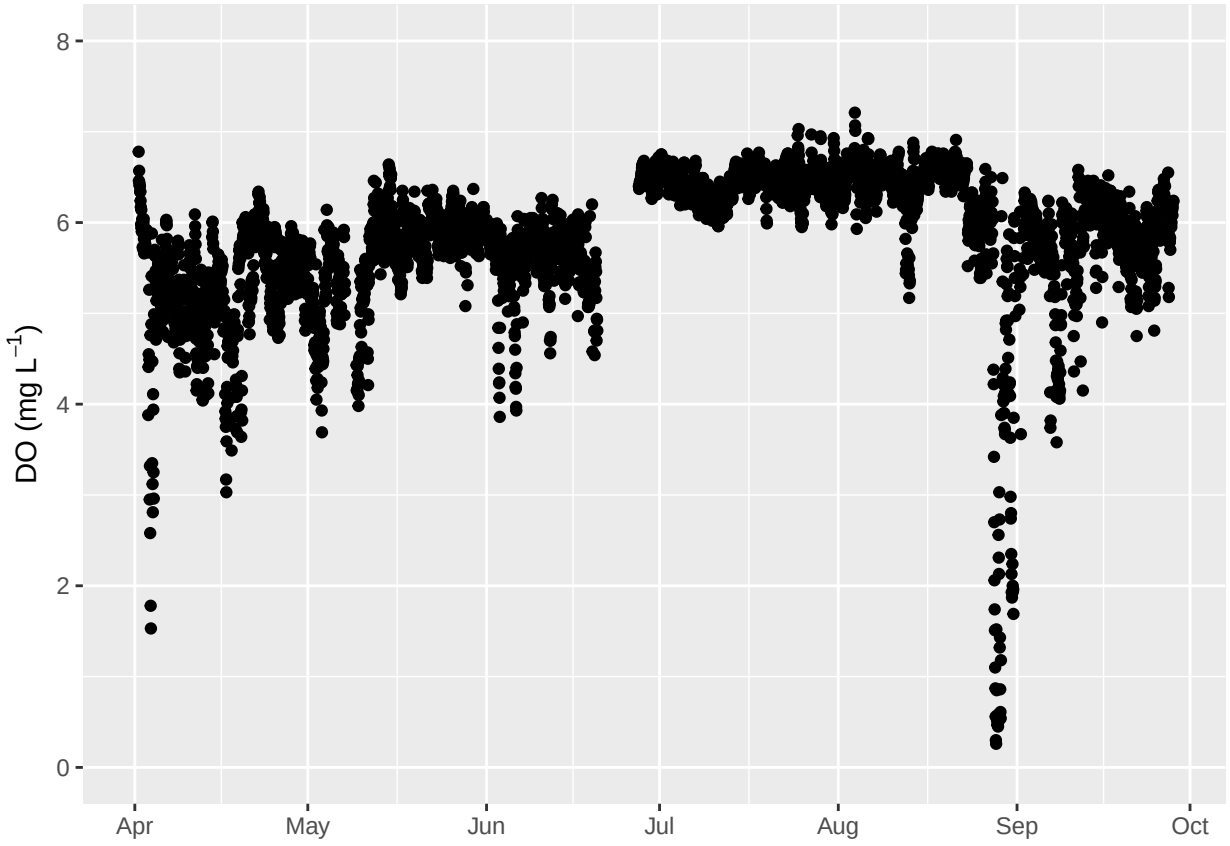
```r
plot_rain <- ggplot(Tac_all %>%
                      select(Timestamp, Rainfall) %>%
                      group_by(Date = date(Timestamp)) %>%
                      summarise(dayRain = sum(Rainfall, na.rm = TRUE)))+
  geom_bar(aes(x = as.POSIXct(Date),
               y = dayRain),
           stat = 'identity', fill = 'black')+
  ylim(120, 0)+
  ylab('Daily Rainfall (mm)')+
  scale_x_datetime(date_breaks = "1 month", date_labels =  "%b")+
  theme(axis.title.x = element_blank())
plot_rain
```



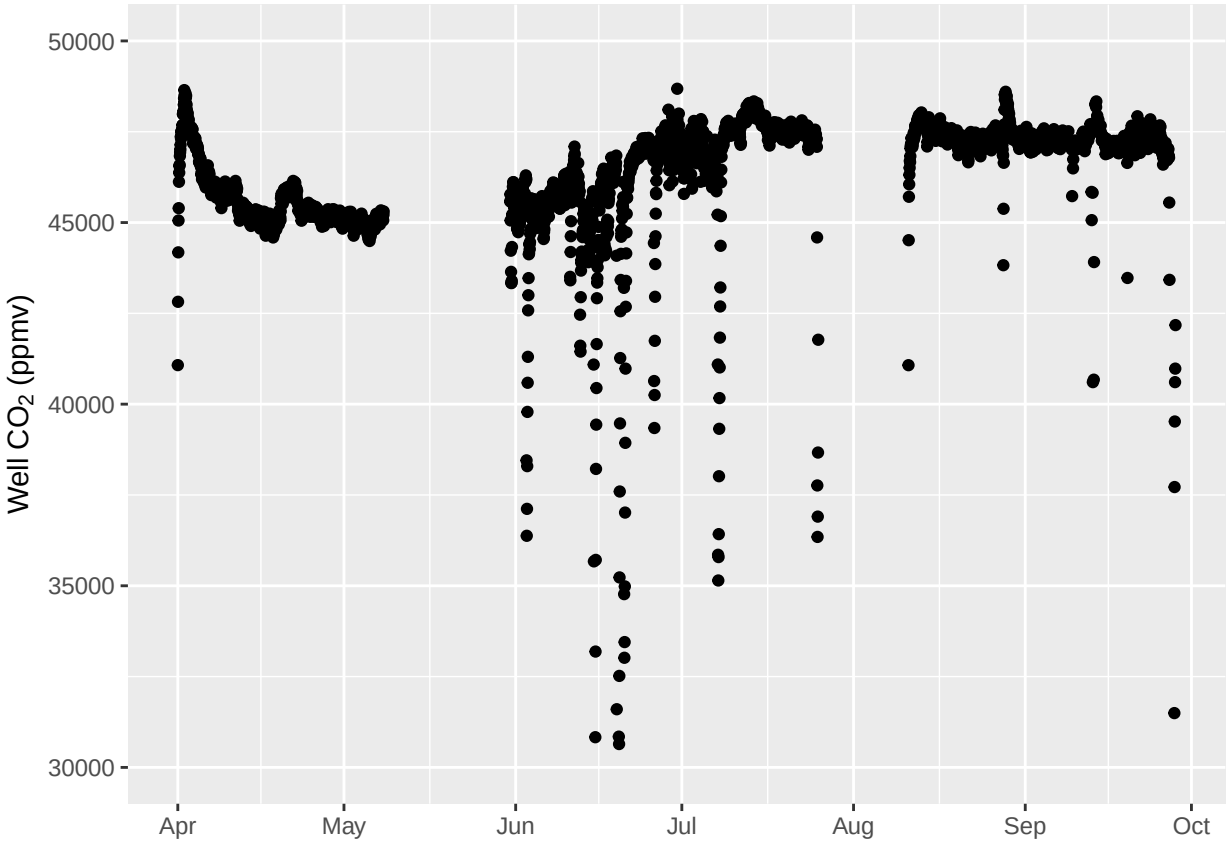

```
plot_Q <- ggplot(Tac_all,
                 aes(x = Timestamp))+
  geom_line(aes(y = sumQ_m3_hr),
            color = 'black')+
  geom_line(aes(y = GW_Q),
            color = 'red')+
  ylab(expression(paste("Discharge (", m^3," ", hr^-1,")")))+
  scale_color_manual(values = c('black',
                                'red'))+
  scale_x_datetime(date_breaks = "1 month",
                   date_labels =  "%b")+
  theme(axis.title.x = element_blank())
plot_Q
```



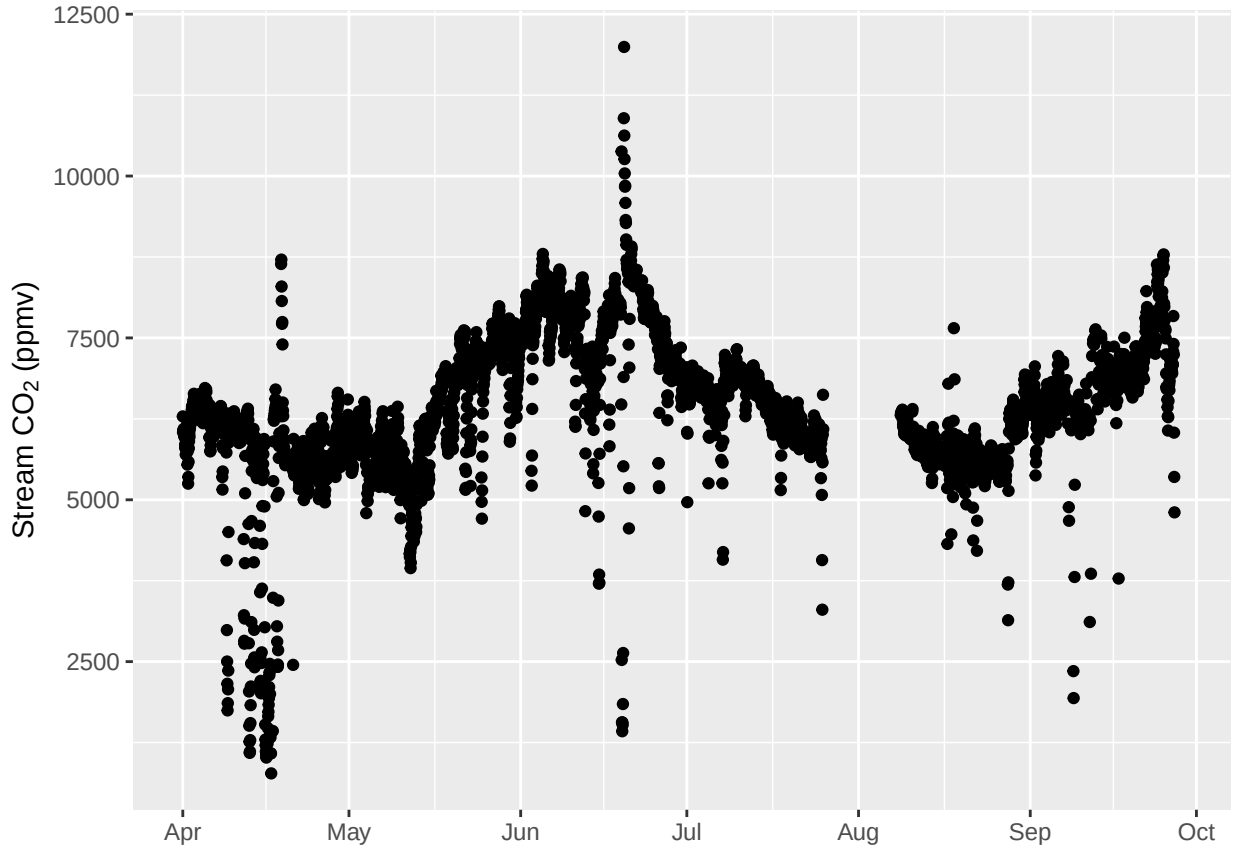

```
plot_do <- ggplot(data = Tac_all, aes(x = Timestamp))+
  geom_point(aes(y = DO.obs))+
  labs(x = "Time",
       y = expression(paste("DO (mg " ,L^{-1},")")))+
  ylim(0,8)+
  theme(axis.title.x = element_blank())+
  scale_x_datetime(date_breaks = "1 month",
                   labels = date_format("%b"))
plot_do
```




```
plot_wellCO2 <- ggplot(Tac_all,
                       aes(x = Timestamp))+
  geom_point(aes(y = wellCO2),
             color = 'black')+
  ylab(expression(paste("Well ", CO[2]," ", "(ppmv)")))+
  scale_x_datetime(date_breaks = "1 month",
                   date_labels =  "%b")+
  theme(axis.title.x = element_blank())+
  ylim(30000, 50000)
plot_wellCO2
```




```
plot_streamCO2 <- ggplot(Tac_all,
                         aes(x = Timestamp))+
  geom_point(aes(y = CO2_ppm),
             color = 'black')+
  ylab(expression(paste("Stream ", CO[2]," ", "(ppmv)")))+
  scale_x_datetime(date_breaks = "1 month",
                   date_labels =  "%b")+
  theme(axis.title.x = element_blank())
plot_streamCO2
```




```
Fig2 <- plot_grid(plot_rain, plot_Q, plot_do, plot_streamCO2, plot_wellCO2,
                  nrow = 5,
                  align = 'hv',
                  labels = 'auto', label_fontface = 'plain',
                  hjust = c(-9, -10, -10.5,-10,-9), vjust = 2)
Fig2
```

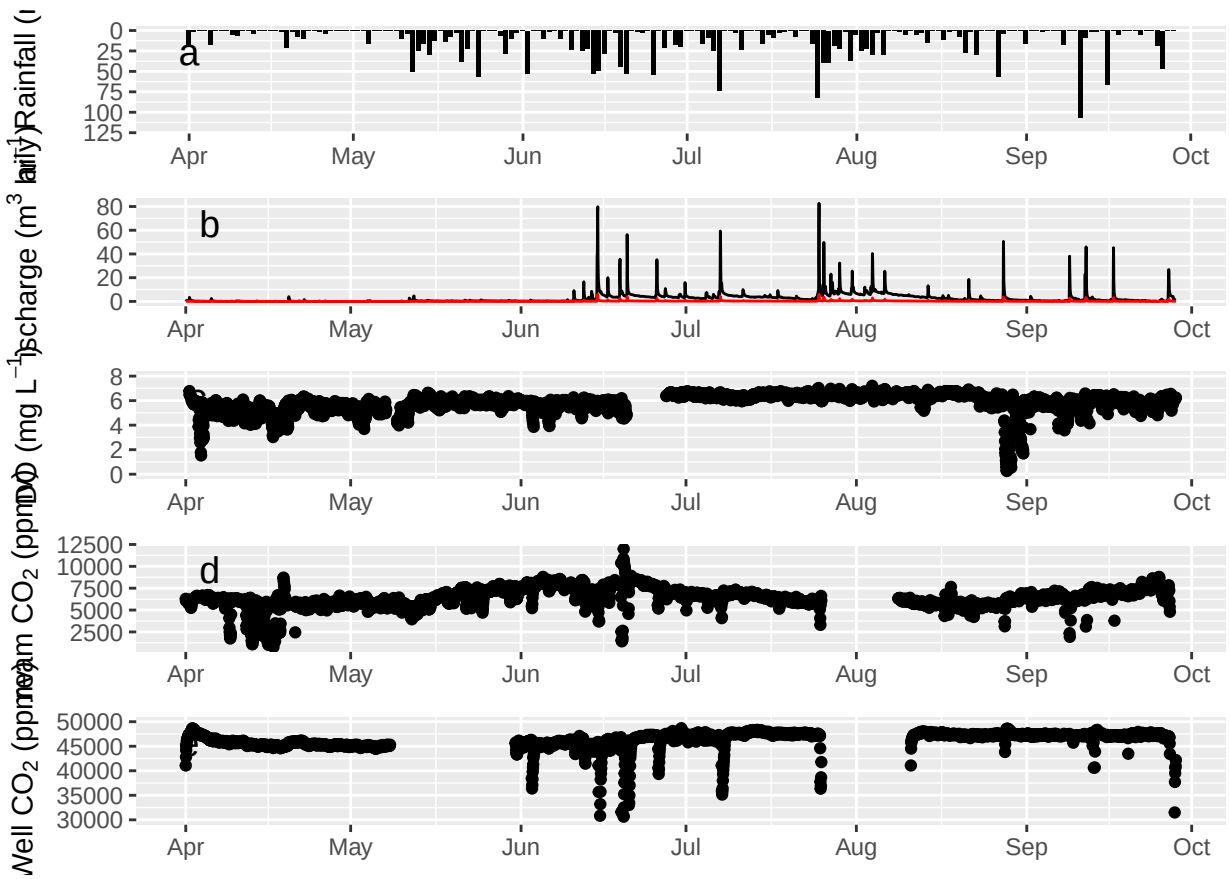
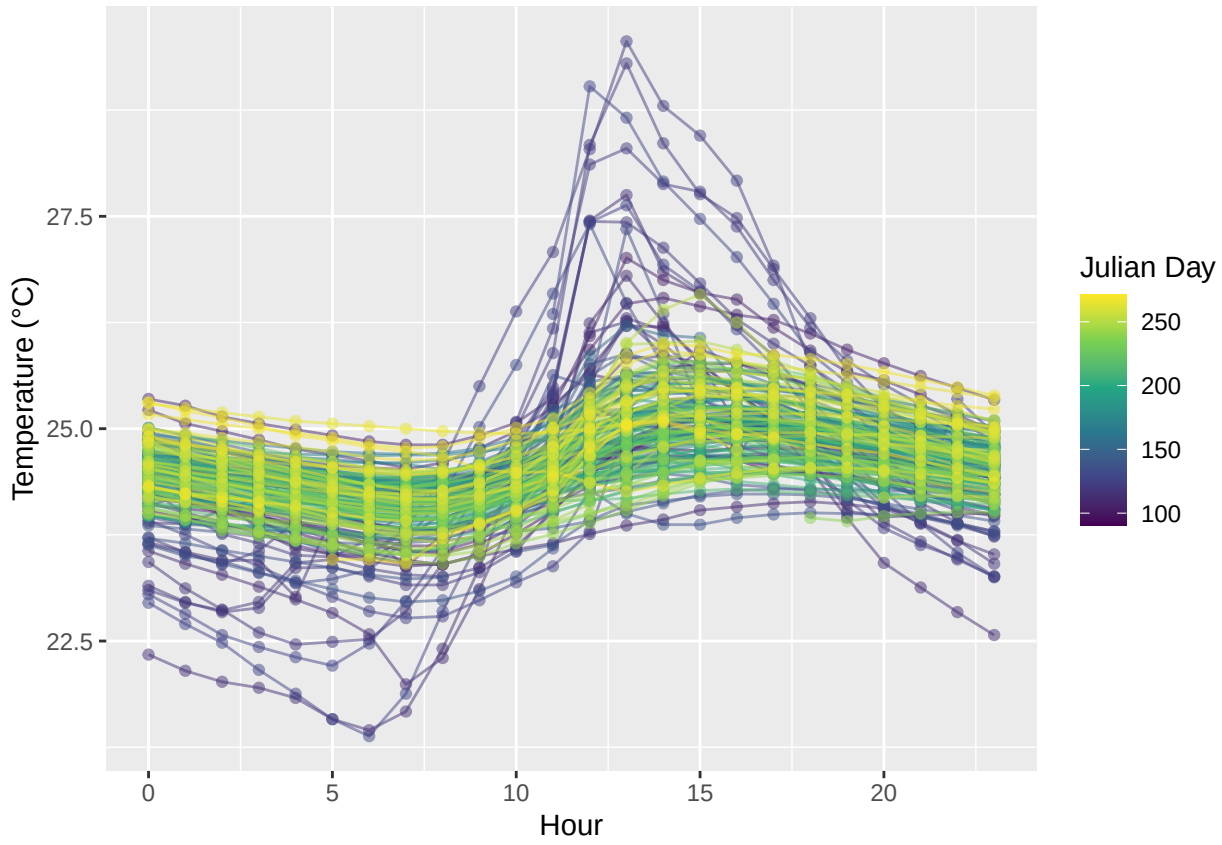


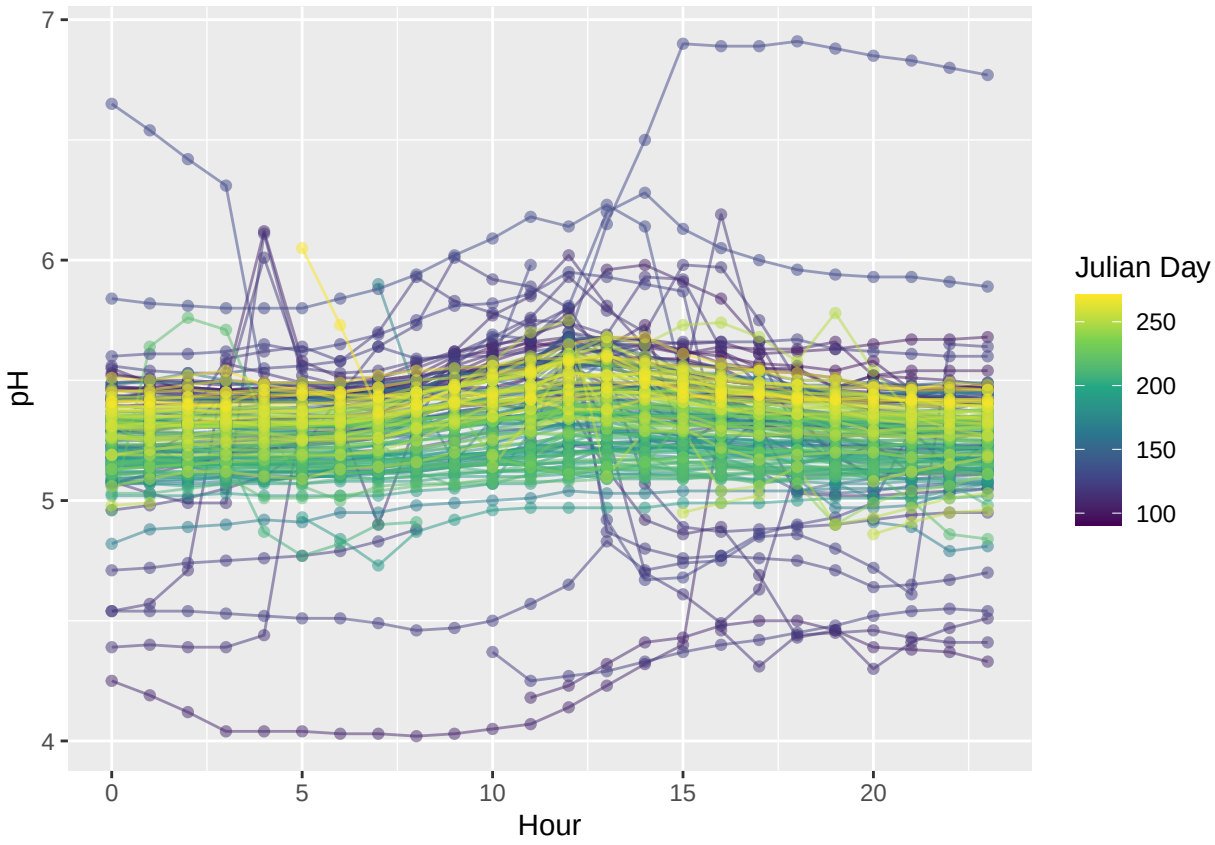

**Fig S2**

Here, we explore diel variation in stream parameters. This exploration is informative for estimating metabolism. As there is little diel variation, we use a direct approach to estimate net ecosystem production (NEP).

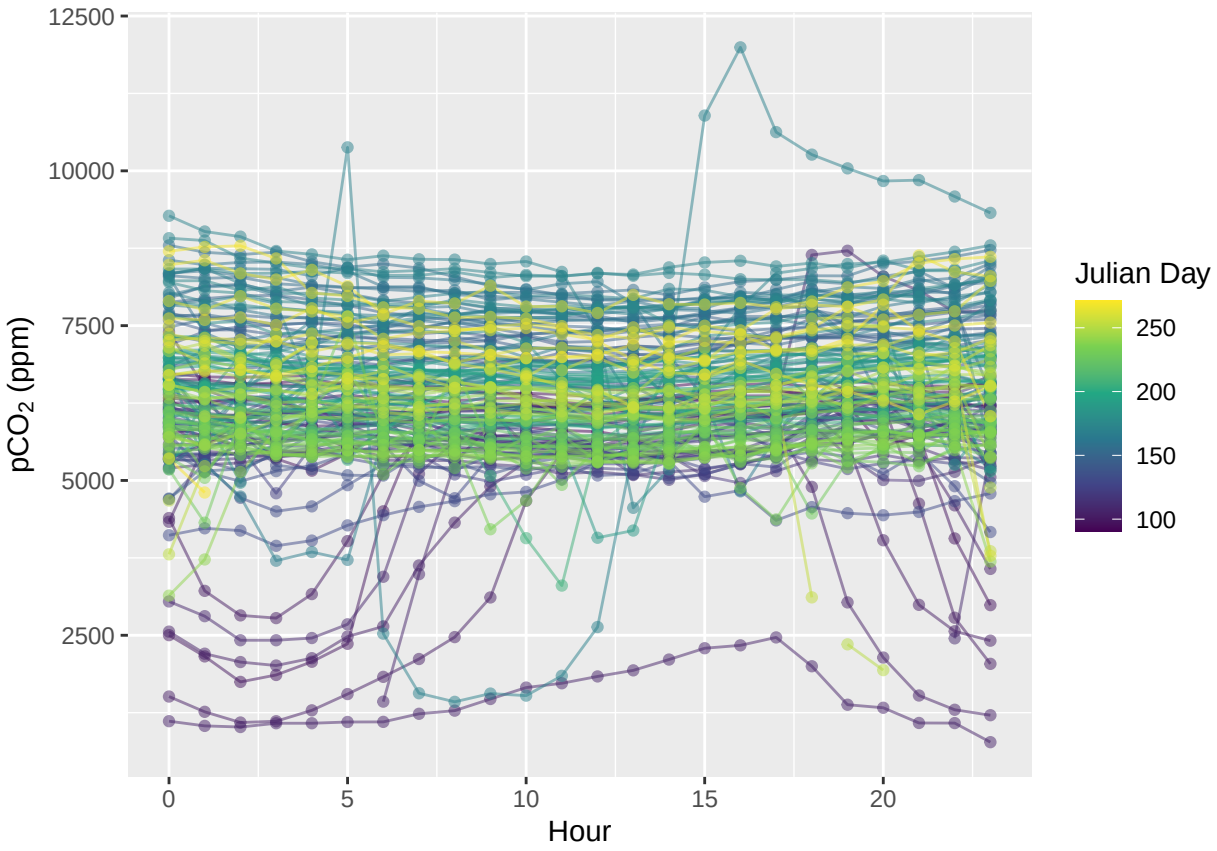
```
tempHour <- ggplot(Tac_all,
                   aes(x = hour(Timestamp), y = temp.water))+
  geom_line(alpha = 0.5,
            aes(group = yday(Timestamp),
                color = yday(Timestamp)))+
  geom_point(alpha = 0.5,
             aes(group = yday(Timestamp),
                 color = yday(Timestamp)))+
  xlab('Hour')+
  scale_colour_viridis_c(name = "Julian Day")+
  theme(legend.title = element_text(face = 'plain'))+
  ylab("Temperature (°C)")
tempHour
```



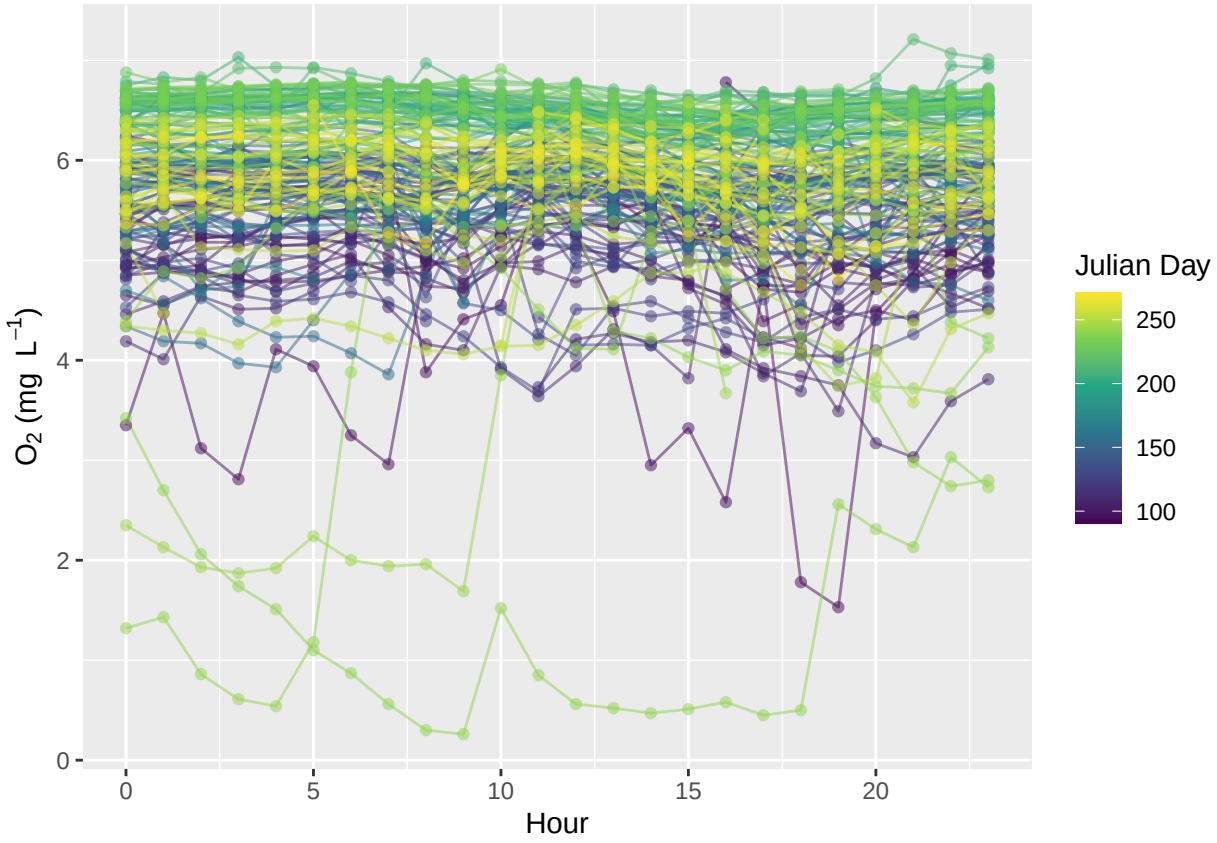

```
# pH
pHHour <- ggplot(Tac_all,
                 aes(x = hour(Timestamp), y = pH))+
  geom_line(alpha = 0.5,
            aes(group = yday(Timestamp),
                color = yday(Timestamp)))+
  geom_point(alpha = 0.5,
             aes(group = yday(Timestamp),
                 color = yday(Timestamp)))+
  labs(x = 'Hour', y = "pH")+
  scale_colour_viridis_c(name = "Julian Day")+
  theme(legend.title = element_text(face = 'plain'))
pHHour
```




```
# stream CO2, ppm
CO2Hour = ggplot(Tac_all,
                 aes(x = hour(Timestamp), y = CO2_ppm))+
  geom_line(alpha = 0.5,
            aes(group = yday(Timestamp), color = yday(Timestamp)))+
  geom_point(alpha = 0.5,
             aes(group = yday(Timestamp), color = yday(Timestamp)))+
  xlab('Hour')+
  ylab(expression(paste(pCO[2], ' (ppm)')))+
  scale_colour_viridis_c(name = "Julian Day")+
  theme(legend.title = element_text(face = 'plain'))
CO2Hour
```




```
# dissolved oxygen
DOHour = ggplot(Tac_all,
                aes(x = hour(Timestamp), y = DO.obs))+
  geom_line(alpha = 0.5,
            aes(group = yday(Timestamp), color = yday(Timestamp)))+
  geom_point(alpha = 0.5,
             aes(group = yday(Timestamp), color = yday(Timestamp)))+
  xlab('Hour')+
  ylab(expression(paste(O[2], " (mg "," ",L^{-1},")")))+
  scale_colour_viridis_c(name = 'Julian Day')+
  theme(legend.title = element_text(face = 'plain'))
DOHour
```




```
FigS2 <- plot_grid(tempHour, pHHour, CO2Hour, DOHour,
                   ncol = 2, nrow = 2, align = 'hv',
                   labels = 'auto')
FigS2
```

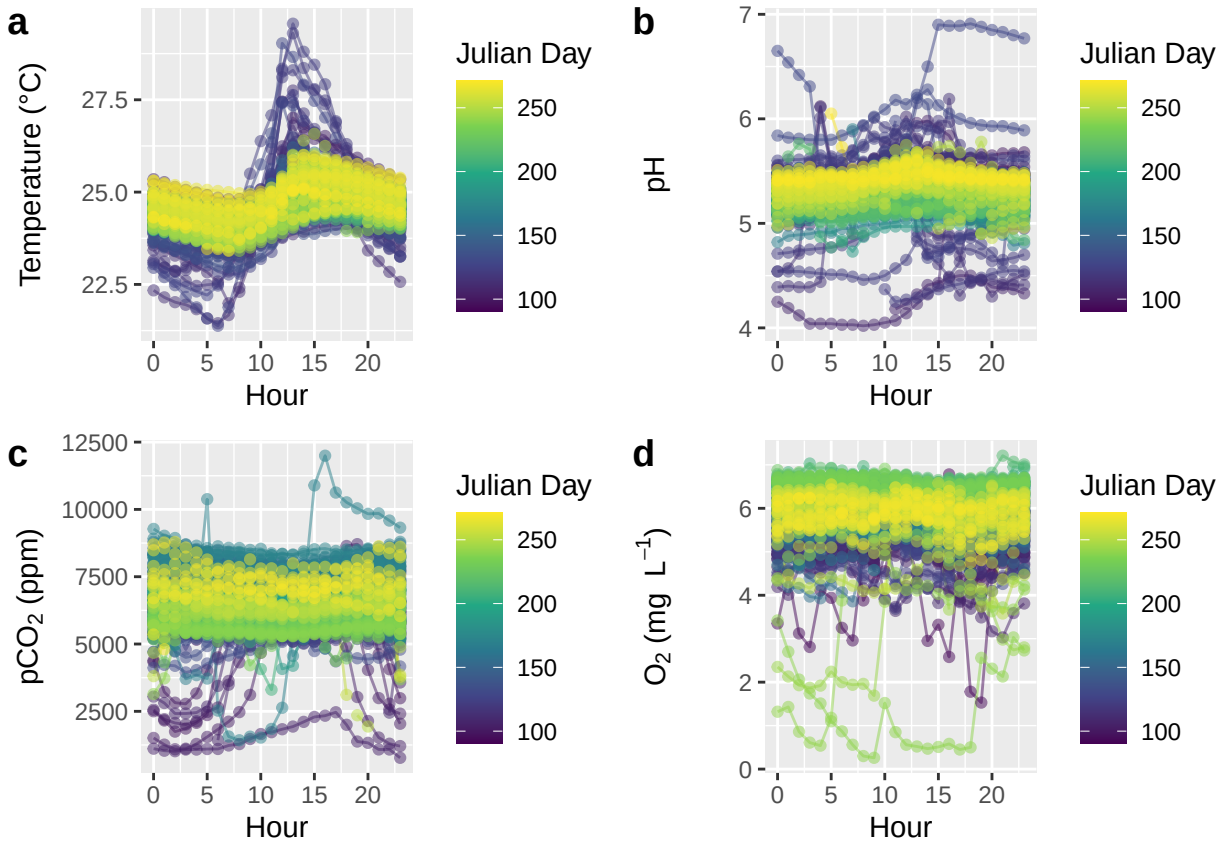



**Fig 3**

Here, we plot CO2 and O2 in respect to their atmospheric saturation concentrations. We calculate saturation concentrations for O2 using a barometric pressure model (Hall and Hotchkiss 2017) and for CO2 assuming atmopsheric CO2 of 400 ppm. We then explore the mean departure coordinates for the entire data collection period and for each month. The coordinates and slope inform processes that simultaneouly control O[2] and CO[2] in the stream.

```
Tac_departure <- Tac_all %>%
  mutate(
    # temperature-corrected Henry's Law constant (Plummer and Busenberg 1982)
    kH = 29.41*exp(-2400*(1/(273 + temp.water) - (1/298))),

    # in-stream CO2-aq calculations
    CO2_aq = ((CO2_ppm/1e6)/kH)*1000,       # aqueous CO2: mol CO2 m-3
    CO2_sat = ((400/1e6)/kH)*1000,

    ts = log((298.15 - temp.water)/(298.15 + temp.water)),    # function as part of O2 saturation, Hall
    u = 10^(8.10765 - (1750.3/(235 + temp.water))),           # function as part of O2 saturation

    Osat = exp(2.00907 +
                 (3.22014*ts) +
                 (4.0501*ts^2) +
                 (4.94457*ts^3) -
                 (0.256847*ts^4) +
```

```
                (3.88767*ts^5)) *
      ((bp_mmHg - u)/(760 - u))*1.42905,

    CO2_dep = (CO2_aq - CO2_sat)*1000,    # umol/L
    O2_dep = ((DO.obs - Osat)/32)*1000)  # umol/L


Tac_dep_clean <- na.omit(Tac_departure)

Tac_dep_clean$month = lubridate::month(Tac_dep_clean$Timestamp,
                                       abbr = TRUE, label = TRUE)

# calculate mean and sd
Tac_mu <- c(mean(Tac_dep_clean$CO2_dep, na.rm = TRUE),
            mean(Tac_dep_clean$O2_dep, na.rm = TRUE))
Tac_sd <- c(sd(Tac_dep_clean$CO2_dep, na.rm = TRUE),
            sd(Tac_dep_clean$O2_dep, na.rm = TRUE))
# correlation and covariance matrix
Tac_corMat <- cor(cbind(Tac_dep_clean$CO2_dep,
                        Tac_dep_clean$O2_dep))
Tac_covMat <- cov(cbind(Tac_dep_clean$CO2_dep,
                        Tac_dep_clean$O2_dep))
# eigen values
Tac_evals <- eigen(Tac_covMat)$values
Tac_ell_length <- 2*sqrt(5.991*Tac_evals)

# run linear model
Tac_reg <- lmodel2(data = Tac_dep_clean,
                   O2_dep ~ CO2_dep,
                   nperm = 99)
```

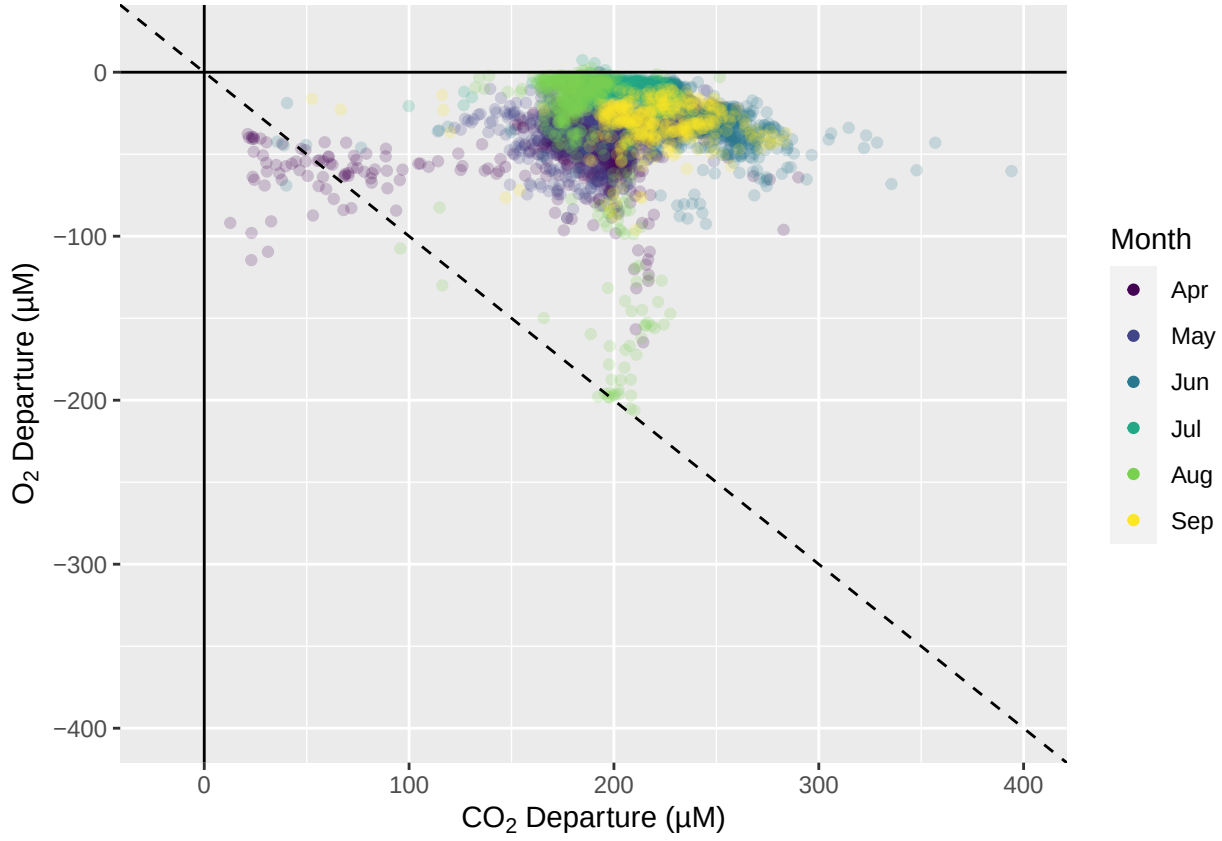
```
## RMA was not requested: it will not be computed.
```

```
# extract values from regression
Tac_inter = Tac_reg$regression.results[2,2]
Tac_slope = Tac_reg$regression.results[2,3]
Tac_correlation = Tac_reg$r

# create new data frame
Tac_metrics <- data.frame(meanCO2dep = Tac_mu[1],
                          meanO2dep = Tac_mu[2],
                          offset = Tac_mu[1] + Tac_mu[2],
                          EQ = 1/abs(Tac_slope),
                          width = Tac_ell_length[2],
                          stretch = Tac_ell_length[1])
Tac_metrics
```

```
##   meanCO2dep meanO2dep   offset       EQ    width  stretch
## 1   203.8241 -32.04432 171.7798 9.461728 128.9109 188.8561
```

```
# plot- all data points with 95% CI ellipsis
Fig3 <- ggplot(data = Tac_departure,
```

```
              aes(x = CO2_dep,
                  y = O2_dep,
                  color = factor(month(Timestamp))))+
  geom_point(alpha = 0.2)+
  geom_hline(yintercept = 0)+
  geom_vline(xintercept = 0)+
  geom_abline(intercept = 0, slope = -1, linetype = 2)+
  xlim(-20, 400)+
  ylim(-400, 20)+
  xlab(expression(paste(CO[2], " Departure (µM)")))+
  ylab(expression(paste(O[2], " Departure (µM)")))+
  guides(colour = guide_legend(override.aes = list(alpha = 1)))+
  scale_color_viridis_d(name = "Month",
                        labels = c("Apr", "May", "Jun", "Jul", "Aug", "Sep"))
Fig3
```




```
# summary for each month of the data
Tac_summary_mon <- Tac_dep_clean %>%
  group_by(month(Timestamp, label = TRUE, abbr = TRUE)) %>%
  dplyr::summarise(meanCO2_dep = mean(CO2_dep, na.rm = TRUE),
                   meanO2_dep = mean(O2_dep, na.rm = TRUE),
                   sdCO2_dep = sd(CO2_dep, na.rm = TRUE),
                   sdO2_dep = sd(O2_dep, na.rm = TRUE),
                   corMat = cor(cbind(CO2_dep, O2_dep)),
                   covMat = cor(cbind(CO2_dep, O2_dep))) %>%
```

```
  dplyr::mutate(offset = meanCO2_dep + meanO2_dep,
                evals_1 = eigen(covMat)$values[1],
                evals_2 = eigen(covMat)$values[2],
                ell_length_1 = 2*sqrt(5.991*evals_1),
                ell_length_2 = 2*sqrt(5.991*evals_2)) %>%
  dplyr::rename(month = `month(Timestamp, label = TRUE, abbr = TRUE)`) %>%
  group_by(month) %>%
  filter(row_number() %% 2==0)
```

```
## `summarise()` has grouped output by 'month(Timestamp, label = TRUE, abbr = TRUE)'. You can override u
```

```
Tac_model_metrics <- Tac_dep_clean %>%
  select(Timestamp, CO2_dep, O2_dep) %>%
  dplyr::mutate(month = factor(month(Timestamp, abbr= TRUE, label = TRUE))) %>%
  nest(-month) %>%
  mutate(model = map(data,
                     ~lmodel2(O2_dep ~ CO2_dep, nperm = 99,
                              data = .)),
         tidy = map(model,
                    broom::tidy),
         glance = map(model,
                      broom::glance)) %>%
  unnest(tidy, glance) %>%
  select(month, method, term, estimate, r.squared) %>%
  mutate(r = -sqrt(r.squared)) %>%
  filter(method == 'OLS') %>%
  spread(term,
         estimate)
```

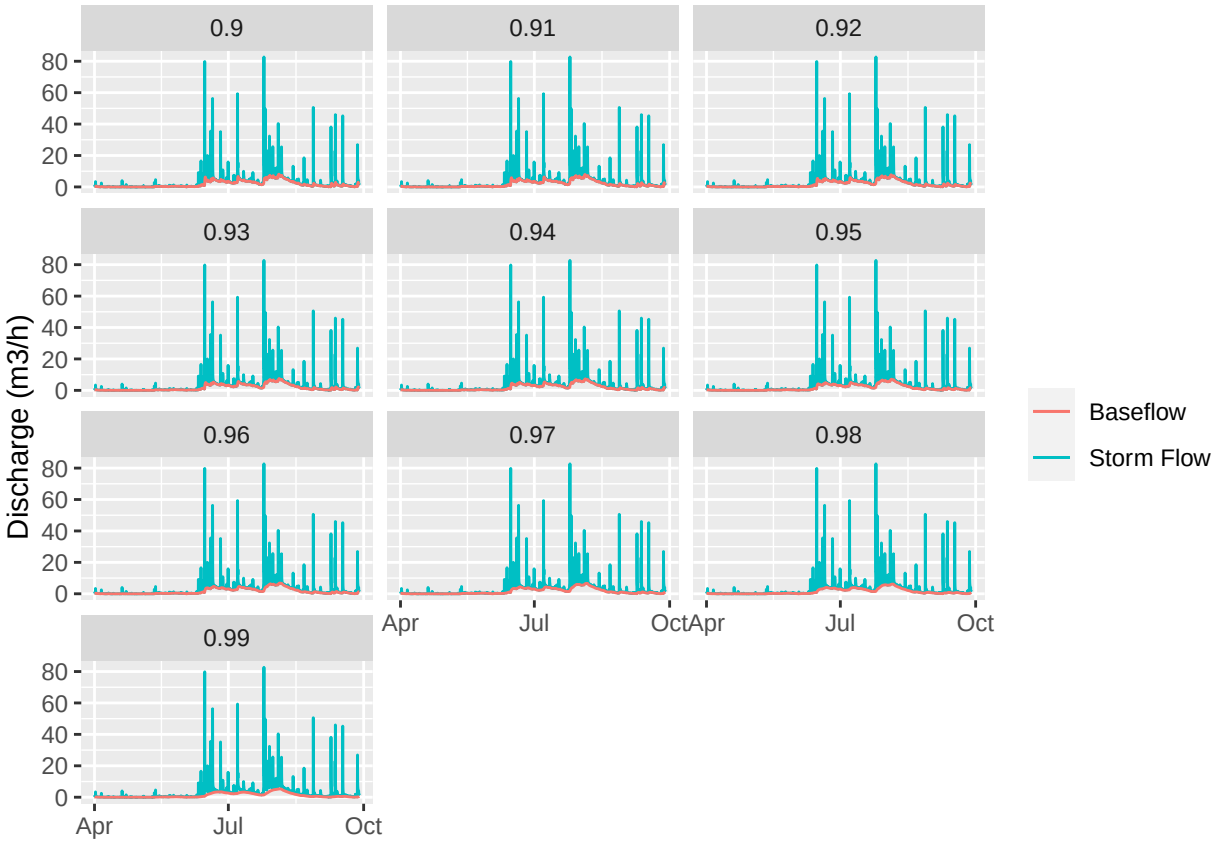
```
## RMA was not requested: it will not be computed.
##
## RMA was not requested: it will not be computed.
##
## RMA was not requested: it will not be computed.
##
## RMA was not requested: it will not be computed.
##
## RMA was not requested: it will not be computed.
##
## RMA was not requested: it will not be computed.
```

```
Tac_metrics_mon <- data.frame(month = Tac_summary_mon$month,
                              meanCO2_dep = Tac_summary_mon$meanCO2_dep,
                              meanO2_dep = Tac_summary_mon$meanO2_dep,
                              sdCO2_dep = Tac_summary_mon$sdCO2_dep,
                              sdO2_dep = Tac_summary_mon$sdO2_dep,
                              cor = Tac_summary_mon$corMat[,1],
                              offset = Tac_summary_mon$offset,
                              slope = 1/abs(Tac_model_metrics$Slope),
                              width = Tac_summary_mon$ell_length_2,
                              stretch = Tac_summary_mon$ell_length_1)
```

## Fig S3

Here, we explore the variation in pCO2 during high and low flow events. We use a baseflow separation approach to distinguish flow and storm conditions by means of a filter parameter, alpha. To assess which alpha fits our reach, we vary alpha and visually evaluate the resulting separation time series.

```
# baseflow separation approach
# iterate alphas to determine the 'best' to use
alphas <- seq(0.9, 0.99, by = 0.01)

hydro_output_all <- tibble()

for(p in 1:length(alphas)) {
  hydro_output <- baseflows(flow.ts = Tac_all %>%
                              select(Date = Timestamp,
                                     Q = sumQ_m3_hr),
                            a = alphas[p],
                            n.reflected = 30,
                            ts = 'daily') %>%
    mutate(bfi = bf/Q,
           alpha = !!alphas[p])

  hydro_output_all <- rbind(hydro_output, hydro_output_all)
}

# plot Q and bf across the different alphas
FigS3 <- ggplot(hydro_output_all,
       aes(x = Date))+
  geom_line(aes(y = Q, color = 'Storm Flow'))+
  geom_line(aes(y = bf, color = 'Baseflow'))+
  facet_wrap(. ~ alpha,
             ncol = 3)+
  ylab('Discharge (m3/h)')+
  theme(axis.title.x = element_blank(),
        legend.title = element_blank())
FigS3
```

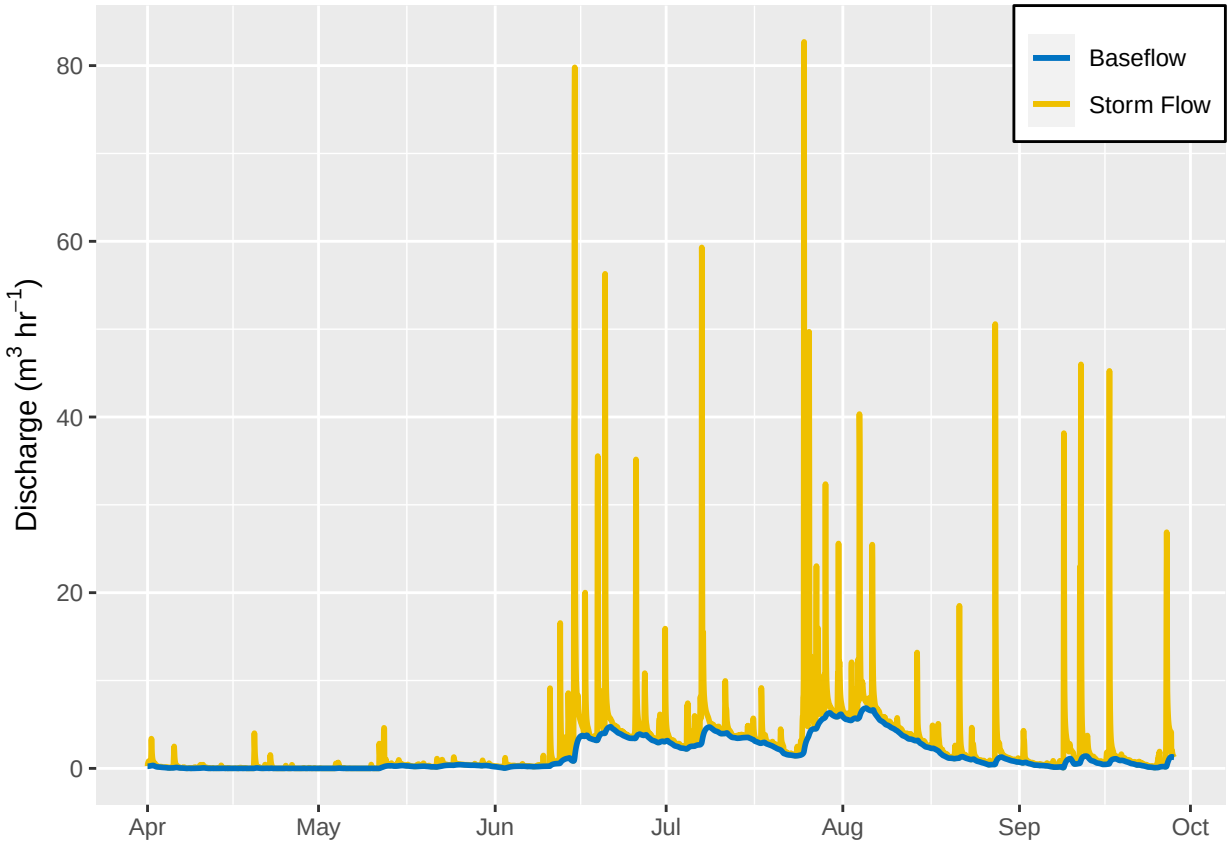



## Figure 4

We then merge the separated hydrograph with the pCO2 data and compare pCO2 at baseflows and storm flows. We use a non-parametric statistcal approach to compare pCO2 across interacting month and flow condition, given the non-normally distributed pCO[2] data.

```
Tac_baseflow <- baseflows(Tac_all %>%
                            select(Date = Timestamp,
                                   Q = sumQ_m3_hr),
                          a = 0.96,
                          n.reflected = 30,
                          ts = 'daily') %>%
  mutate(bfi = bf/Q,
         stormflow = Q - bf,
         source = ifelse(bfi >= 0.5,
                         'Baseflow',
                         'Storm Flow'),
         month = month(Date, label = TRUE, abbr = TRUE)) %>%
  rename(Timestamp = Date)
# bfi = fraction of baseflow calculated; bfi = 1 means all baseflow, 0 = all stormflow

# plot- baseflow on top of stream discharge
Fig4a <- ggplot(Tac_baseflow)+
  geom_line(aes(x = Timestamp,
                y = Q,
```

```
                color = 'Storm Flow'),
            size = 1)+
  geom_line(aes(x = Timestamp,
                y = bf,
                color = 'Baseflow'),
            size = 1)+
  scale_color_jco()+
  ylab(expression(paste('Discharge (', m^3, ' ',hr^-1,')')))+
  scale_x_datetime(date_breaks = "1 month",
                   date_labels =  "%b")+
  theme(axis.title.x = element_blank(),
        legend.position = c(1, 1),
        legend.justification = c(1, 1),
        legend.background = element_rect(color = 'black'),
        legend.title = element_blank())
Fig4a
```




```
# join hydrograph separation with pCO2 data
Tac_baseflow_hyst <- left_join(
  Tac_baseflow,
  Tac_all %>%
    select(Timestamp, CO2_ppm))
```

```
## Joining, by = "Timestamp"
```

```r
shapiro.test(Tac_baseflow_hyst$CO2_ppm)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  Tac_baseflow_hyst$CO2_ppm
## W = 0.91907, p-value < 2.2e-16
```

```r
pco2_source_art <- art(data = Tac_baseflow_hyst %>%
                         dplyr::mutate(source_f = factor(source)) %>%
                         dplyr::select(month, source_f, CO2_ppm) %>%
                         tidyr::drop_na(),
                       CO2_ppm ~ month * source_f)

anova(pco2_source_art)
```

```
## Analysis of Variance of Aligned Rank Transformed Data
##
## Table Type: Anova Table (Type III tests)
## Model: No Repeated Measures (lm)
## Response: art(CO2_ppm)
##
##                   Df Df.res F value     Pr(>F)
## 1 month            5   3717 478.189 < 2.22e-16 ***
## 2 source_f         1   3717 660.905 < 2.22e-16 ***
## 3 month:source_f   5   3717  41.348 < 2.22e-16 ***
## ---
## Signif. codes:   0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
art.con(pco2_source_art,
        'source_f')
```

```
## NOTE: Results may be misleading due to involvement in interactions

##  contrast              estimate   SE   df t.ratio p.value
##  Baseflow - Storm Flow     1146 44.6 3717  25.708  <.0001
##
## Results are averaged over the levels of: month
```

```r
art.con(pco2_source_art,
        'month',
        method = "pairwise")
```

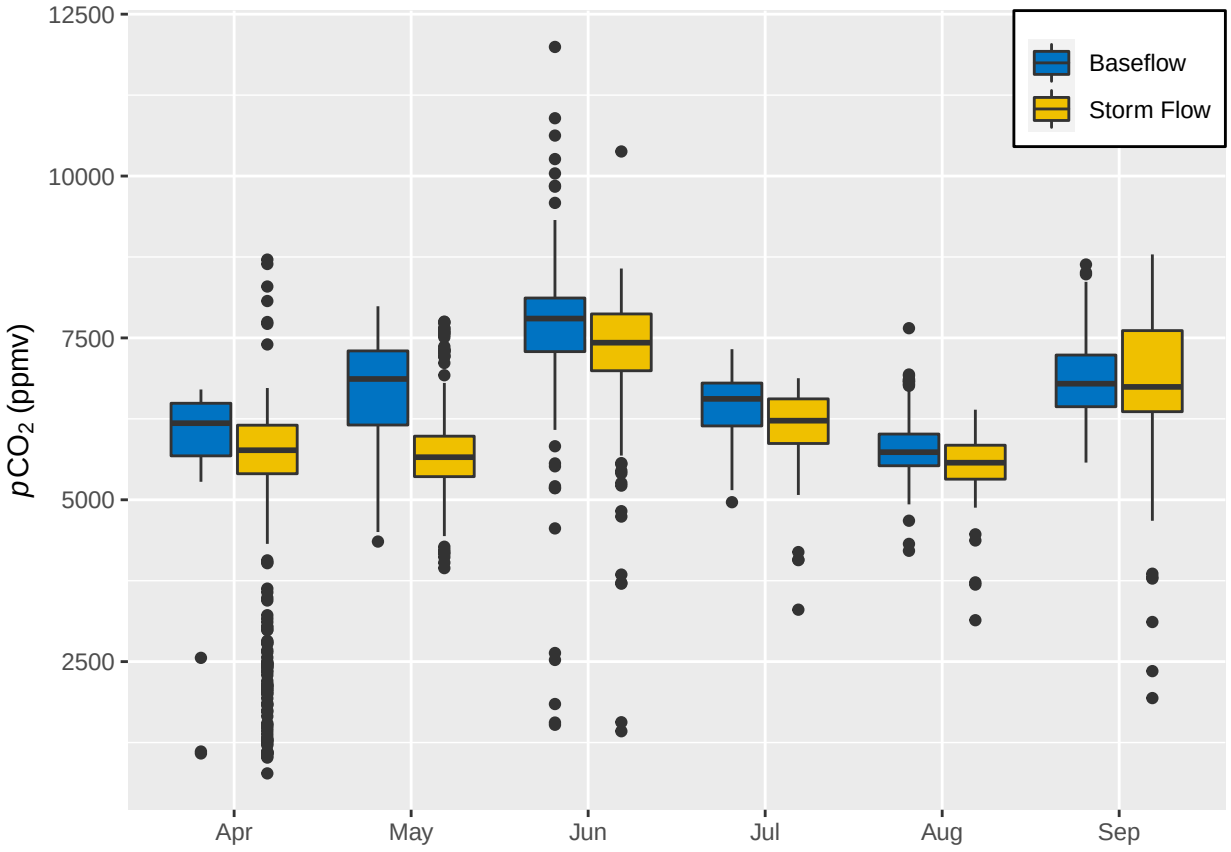
```
## NOTE: Results may be misleading due to involvement in interactions

##  contrast  estimate   SE   df t.ratio p.value
##  Apr - May   -816.4 46.0 3717 -17.767  <.0001
##  Apr - Jun  -2178.7 50.5 3717 -43.125  <.0001
##  Apr - Jul  -1086.8 63.2 3717 -17.196  <.0001
##  Apr - Aug    -71.4 68.1 3717  -1.048  0.9017
```

```
##  Apr - Sep  -1536.3 53.8 3717 -28.567  <.0001
##  May - Jun  -1362.3 44.3 3717 -30.774  <.0001
##  May - Jul   -270.4 58.3 3717  -4.636  0.0001
##  May - Aug    745.1 63.6 3717  11.709  <.0001
##  May - Sep   -719.9 48.0 3717 -15.012  <.0001
##  Jun - Jul   1091.9 62.0 3717  17.616  <.0001
##  Jun - Aug   2107.4 67.0 3717  31.452  <.0001
##  Jun - Sep    642.5 52.3 3717  12.273  <.0001
##  Jul - Aug   1015.4 77.0 3717  13.185  <.0001
##  Jul - Sep   -449.5 64.7 3717  -6.950  <.0001
##  Aug - Sep  -1464.9 69.5 3717 -21.080  <.0001
##
## Results are averaged over the levels of: source_f
## P value adjustment: tukey method for comparing a family of 6 estimates
```

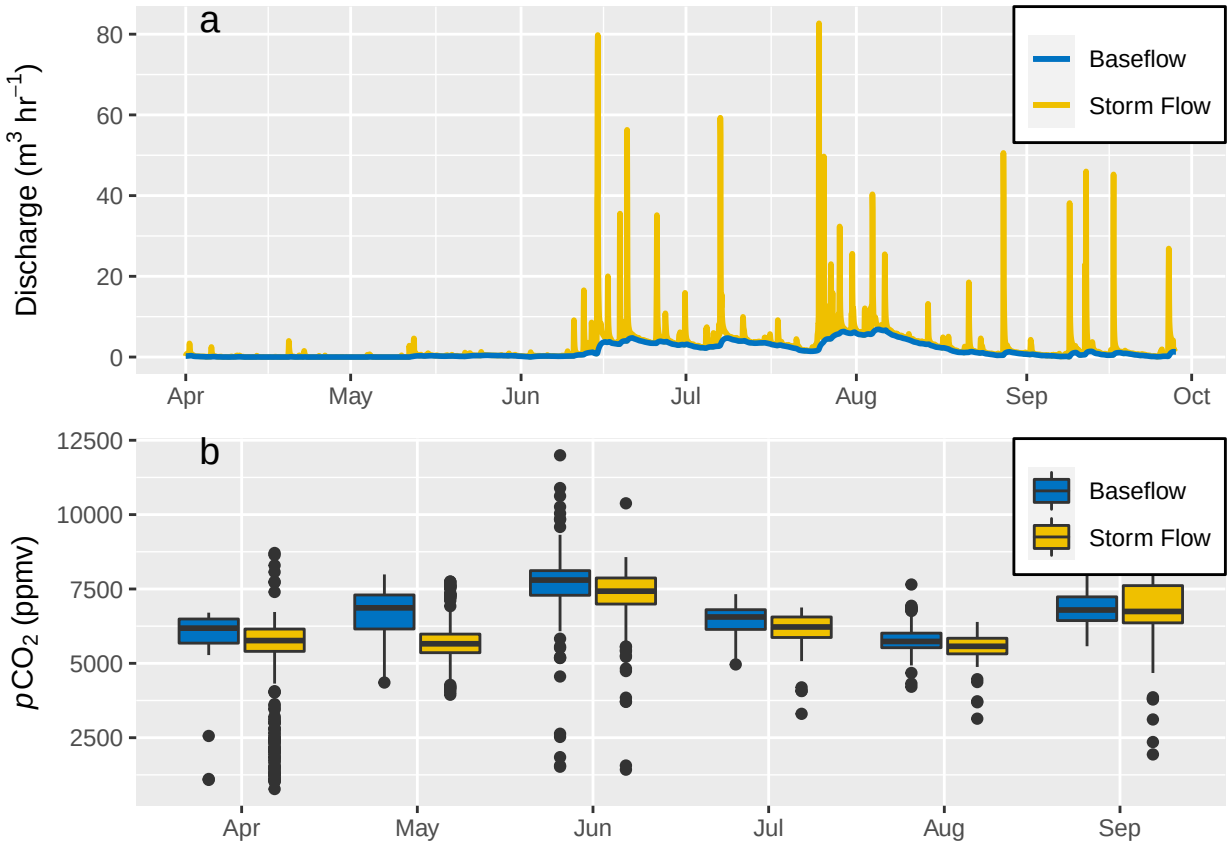
```
contrast(emmeans(artlm(pco2_source_art, 'month:source_f'),
                 ~ month:source_f),
         method = 'pairwise', interaction = TRUE)
```

```
##  month_pairwise source_f_pairwise     estimate  SE   df t.ratio p.value
##  Apr - May      Baseflow - Storm Flow     -784 119 3717  -6.611  <.0001
##  Apr - Jun      Baseflow - Storm Flow      240 130 3717   1.838  0.0661
##  Apr - Jul      Baseflow - Storm Flow      396 163 3717   2.430  0.0151
##  Apr - Aug      Baseflow - Storm Flow      576 176 3717   3.277  0.0011
##  Apr - Sep      Baseflow - Storm Flow      744 139 3717   5.361  <.0001
##  May - Jun      Baseflow - Storm Flow     1024 114 3717   8.960  <.0001
##  May - Jul      Baseflow - Storm Flow     1181 151 3717   7.842  <.0001
##  May - Aug      Baseflow - Storm Flow     1360 164 3717   8.283  <.0001
##  May - Sep      Baseflow - Storm Flow     1528 124 3717  12.348  <.0001
##  Jun - Jul      Baseflow - Storm Flow      157 160 3717   0.980  0.3273
##  Jun - Aug      Baseflow - Storm Flow      337 173 3717   1.946  0.0517
##  Jun - Sep      Baseflow - Storm Flow      505 135 3717   3.734  0.0002
##  Jul - Aug      Baseflow - Storm Flow      180 199 3717   0.904  0.3658
##  Jul - Sep      Baseflow - Storm Flow      348 167 3717   2.083  0.0373
##  Aug - Sep      Baseflow - Storm Flow      168 179 3717   0.936  0.3492
```

```
# plot CO2 vs stormflow
Fig4b <- ggplot(Tac_baseflow_hyst,
                aes(x = month,
                    y = CO2_ppm,
                    fill = source))+
  geom_boxplot()+
  ylab(expression(paste(italic(p), CO[2], ' (ppmv)')))+
  scale_fill_jco()+
  theme(axis.title.x = element_blank(),
        legend.position = c(1, 1),
        legend.justification = c(1, 1),
        legend.background = element_rect(color = 'black'),
        legend.title = element_blank())
Fig4b
```




```
# combine plots
Fig4 <- plot_grid(Fig4a, Fig4b,
                  align = 'v', nrow = 2,
                  labels = 'auto', label_fontface = 'plain',
                  hjust = -10)
Fig4
```



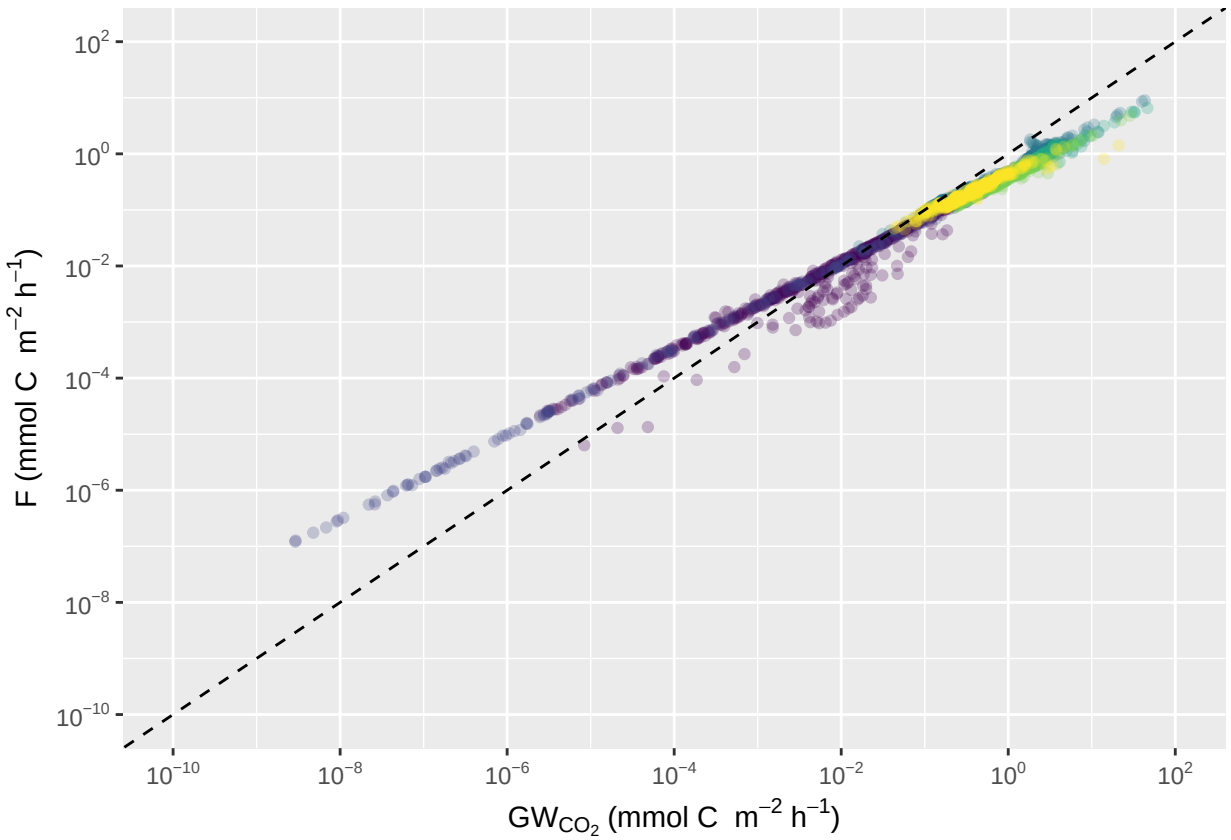

## Figure 5

Here, we estimate areal fluxes in the study reach, including net ecosystem production, evasion, and groundwater CO2, all in mol C m^{-2} h^{-1}.

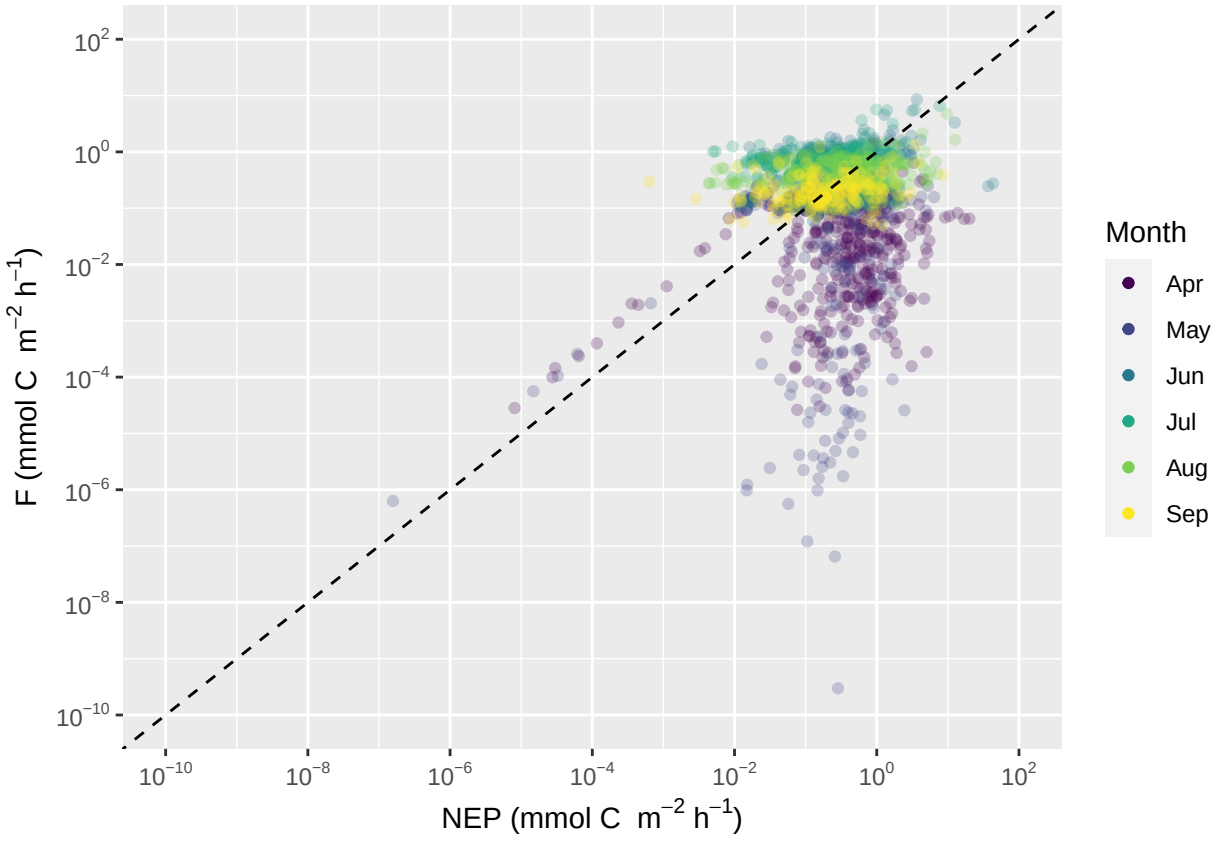
```
Tac_flux <- Tac_all %>%
  mutate(
    # calculate H+
    H = 10^(-1*pH),  # mol/L

    # temperature-scaled dissociation constants (Mook 1976)
    k1 = 10^(-1*((3404.71/(temp.water + 273.15)) + (0.032786*(temp.water + 273.15)) - 14.8435)),
    k2 = 10^(-1*((2902.39/(temp.water + 273.15)) + (0.02379*(temp.water + 273.15)) - 6.4980)),

    # DIC fractions
    alpha0 = (1 + (k1/H) + ((k1*k2)/(H^2)))^-1,
    alpha1 = (1 + (H/k1) + (k2/H))^-1,
    alpha2 = (1 + (H/k2) + ((H^2)/(k1*k2)))^-1,

    # temperature-corrected Henry's Law constant (Plummer and Busenberg 1982)
    kH = 29.41*exp(-2400*(1/(273 + temp.water) - (1/298))),

    # in-stream CO2-aq calculations
    CO2_aq = ((CO2_ppm/1e6)/kH)*1000,      # aqueous CO2: mol CO2 m-3
    CO2_sat = ((400/1e6)/kH)*1000,         # saturation CO2 at 400 ppm mol CO2 m-3
```

```
  # CO2-well
  wellCO2_aq = ((wellCO2/1e6)/kH)*1000,  # well aqueous CO2: mol CO2 m-3

  # DIC partitioning
  H2CO3 = (1/kH)*(CO2_ppm/1000),       # H2CO3* (mol/m3)
  TotDIC = H2CO3/alpha0,               # total stream DIC (mol/m3)
  HCO3 = TotDIC * alpha1,              # HCO3 (mol/m3)

  # Groundwater discharge (m3 h-1)
  GW_Q = ifelse(month(Timestamp) == 'April',    # calculate total groundwater discharge (m3/h)
                (0.1779*sumQ_m3_hr),            # 17.79% of stream Q in dry season (April); m3/h
                (0.0715*sumQ_m3_hr)),           # 7.15% of stream Q in wet season; m3/h

  # groundwater velocity (m h-1); divide GW discharge by reach area (length * width)
  GW_v = ifelse(month(Timestamp) == 'April',
                GW_Q/(Tac_reach * Tac_width_lower_dry),
                GW_Q/(Tac_reach * Tac_width_lower_wet)
  ),

  # upreach discharge (m3 h-1); Qu = Qweir - Qgw
  Q_upreach = sumQ_m3_hr - GW_Q,

  # upreach DIC
  upreach_DIC = Q_upreach * (CO2_aq + HCO3),

  # GWCO2 flux (mol CO2 m-2 h-1)
  GWCO2 = GW_v * wellCO2_aq,

  # estimate gas exchange based on scaling equations
  k600 = (1.74811528*(meanQ_m3_s)^0.7670773)/24,    # hourly gas velocity based on scaline equation
  K600 = k600/mean_depth,                           # convert to first order coefficient
  KCO2 = K600*(600/Sc_CO2)^0.7,                     # convert to CO2
  KO2 = K600*(600/Sc_O2)^0.7,                       # convert to O2

  # CO2 evasion (mol CO2 m-2 h-1)
  CO2_efflux = (CO2_aq - CO2_sat)*KCO2*mean_depth,  # hourly CO2 efflux (mol CO2 m-2 h-1)

  # hydrologic export
  CO2_aq_export = ((CO2_aq) * (sumQ_m3_hr)),        # aqueous CO2 export (mol C h-1)
  HCO3_aq_export = (HCO3 * (sumQ_m3_hr)),           # HCO3 export (mol C h-1)

  # Direct metabolism model
  ts = log((298.15 - temp.water)/(298.15 + temp.water)),   # function as part of O2 saturation, Hall
  u = 10^(8.10765 - (1750.3/(235 + temp.water))),          # function as part of O2 saturation

  # calculate O2 saturation
  Osat = exp(2.00907 +
               (3.22014*ts) +
               (4.0501*ts^2) +
               (4.94457*ts^3) -
               (0.256847*ts^4) +
               (3.88767*ts^5)) *
    ((bp_mmHg - u)/(760 - u))*1.42905,
```

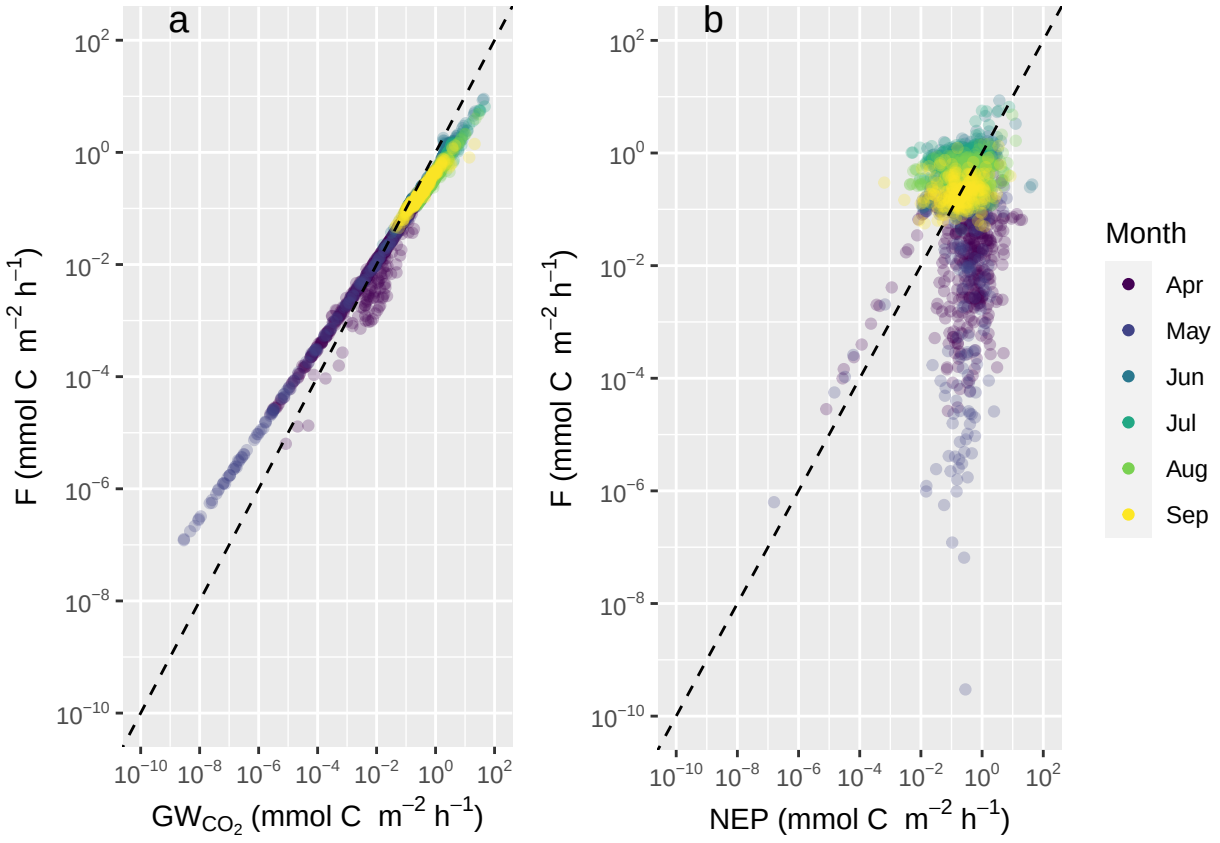
```
    # hourly NEP (mol C m-2 h-1)
    direct_met_mol =
      (diff(DO.obs) - (KO2*(Osat - DO.obs)))*(1/32)*mean_depth*-1    # -1 converts from mol O2 to mol C
  )

Fig5a <- ggplot(Tac_flux,
                aes(x = GWCO2*1000,
                    y = CO2_efflux*1000))+
  geom_point(aes(color = factor(month(Timestamp, label = TRUE, abbr = TRUE))),
             alpha = 0.25)+
  geom_abline(aes(slope = 1,
                  intercept = 0),
              linetype = 2)+
  scale_x_log10(limits = c(1e-10, 1e2),
                breaks = trans_breaks("log10", function(x) 10^x),
                labels = trans_format("log10", math_format(10^.x)))+
  scale_y_log10(limits = c(1e-10, 1e2),
                breaks = trans_breaks("log10", function(x) 10^x),
                labels = trans_format("log10", math_format(10^.x)))+
  scale_color_viridis_d(name = 'Month')+
  xlab(expression(paste(GW[CO[2]], ' (mmol C ', " ", m^-2," ", h^-1,")")))+
  ylab(expression(paste('F (mmol C ', " ", m^-2," ", h^-1,")")))+
  theme(legend.position = 'none')
Fig5a
```



```
Fig5b <- ggplot(Tac_flux ,
                aes(x = direct_met_mol*1000,
                    y = CO2_efflux*1000))+
  geom_point(aes(color = factor(month(Timestamp, label = TRUE, abbr = TRUE))),
             alpha = 0.25)+
  geom_abline(aes(slope = 1, intercept = 0), linetype = 2)+
  scale_x_log10(limits = c(1e-10, 1e2),
                breaks = trans_breaks("log10", function(x) 10^x),
                labels = trans_format("log10", math_format(10^.x)))+
  scale_y_log10(limits = c(1e-10, 1e2),
                breaks = trans_breaks("log10", function(x) 10^x),
                labels = trans_format("log10", math_format(10^.x)))+
  scale_color_viridis_d(name = 'Month')+
  xlab(expression(paste('NEP (mmol C ', " ", m^-2," ", h^-1,")")))+
  ylab(expression(paste('F (mmol C ', " ", m^-2," ", h^-1,")")))+
  guides(colour = guide_legend(override.aes = list(alpha = 1)))
Fig5b
```




```
Fig5 <- plot_grid(Fig5a, Fig5b,
                  ncol = 2,
                  rel_widths = c(0.75, 1),
                  labels = 'auto', label_fontface = 'plain',
                  hjust = -8.5, vjust = 1.5)
Fig5
```

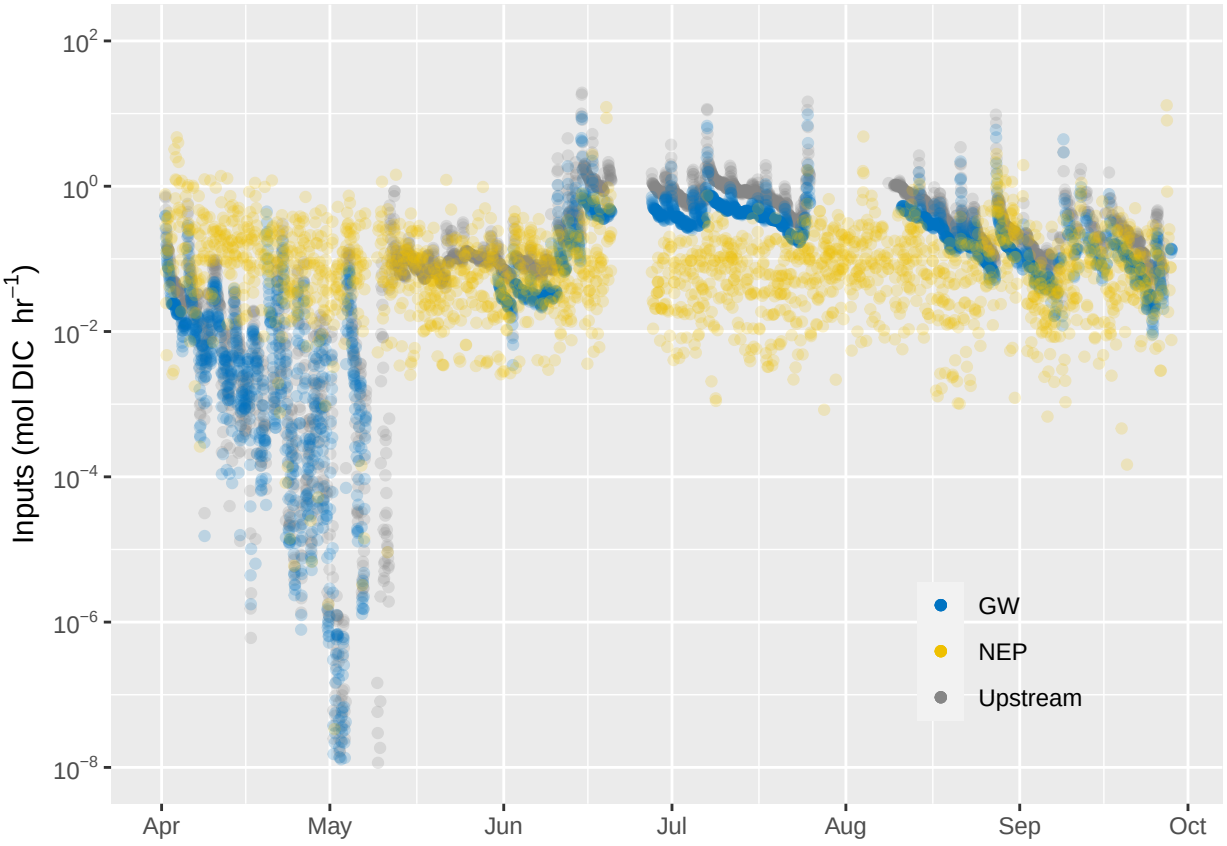



## Figure 6

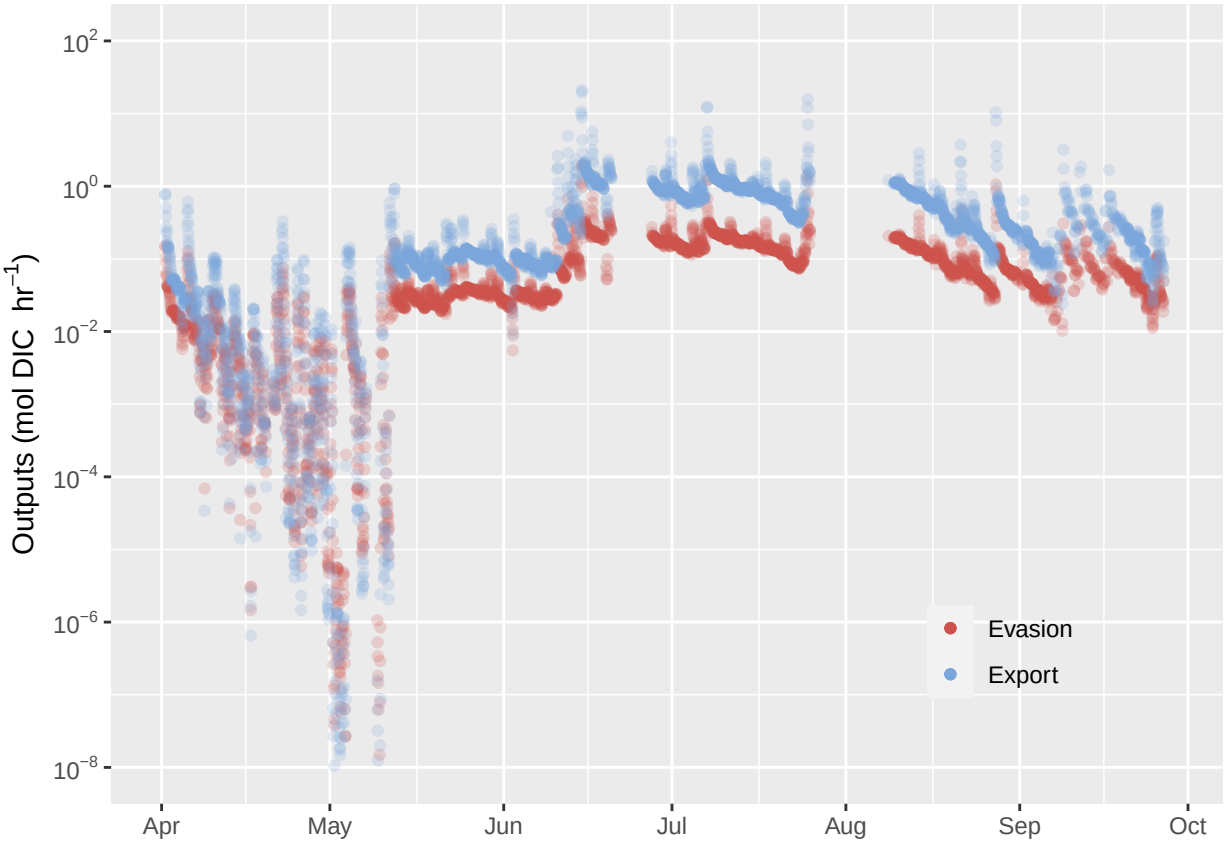
Here, we complete a reach scale volumetric DIC mass balance. The areal fluxes are converted to volumetric fluxes and converted to DIC by means of the carbonate chemistry dissociation constants. We include hydrologic inputs and losses. We plot cumulative inputs and exports and assess the mass balance by means of a change term.

```
Tac_scale <- Tac_flux %>%

  # select necessary columns and fluxes
  dplyr::select(Timestamp,
                Rainfall,
                pH,
                alpha0,
                Q_upreach,
                upreach_DIC,        # mol DIC h-1
                GWCO2,              # mol CO2 m-2 h-1
                CO2_efflux,         # mol CO2 m-2 h-1
                direct_met_mol,     # mol CO2 m-2 h-1
                CO2_aq_export,      # mol CO2 h-1
                HCO3_aq_export      # mol HCO3 h-1
  ) %>%

  # convert to mol DIC
  mutate(GW_DIC = GWCO2,                              # assumes mol CO2 = mol DIC
```

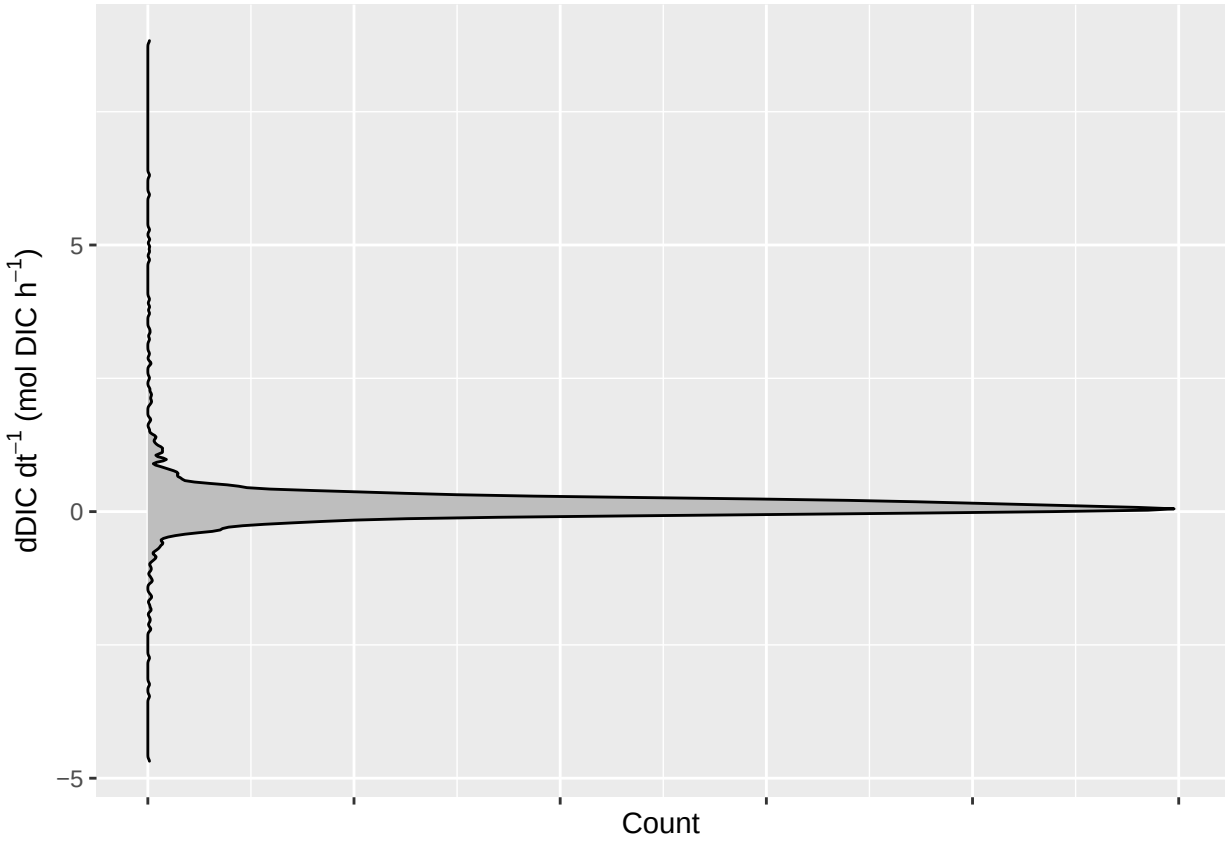
```
        F_DIC = CO2_efflux/alpha0,                    # convert to: mol DIC m-2 h-1
        NEP_DIC = direct_met_mol/alpha0,              # convert to: mol DIC m-2 h-1
        down_DIC = HCO3_aq_export + CO2_aq_export     # convert to: mol DIC h-1
  ) %>%

  # calculate mass balance terms
  mutate(DIC_in = upreach_DIC,                                        # mol DIC h-1
         GW_in = ifelse(month(Timestamp) == 'April',
                        GW_DIC*Tac_reach*Tac_width_lower_dry,
                        GW_DIC*Tac_reach*Tac_width_lower_wet),              # mol DIC h-1
         NEP_in = ifelse(month(Timestamp) == 'April',
                         NEP_DIC*Tac_reach*Tac_width_lower_dry,
                         NEP_DIC*Tac_reach*Tac_width_lower_wet),            # mol DIC h-1

         DIC_out = down_DIC,                                          # mol DIC h-1
         F_out = ifelse(month(Timestamp) == 'April',
                        F_DIC*Tac_reach*Tac_width_lower_dry,
                        F_DIC*Tac_reach*Tac_width_lower_wet),              # mol DIC h-1

         dC_dt = (DIC_in + NEP_in + GW_in) -
           (F_out + DIC_out)
  )

Tac_scaled_in <- ggplot(Tac_scale,
                        aes(x = Timestamp))+
  geom_point(aes(y = DIC_in, color = 'Upstream'), alpha = 0.2)+
  geom_point(aes(y = GW_in, color = 'GW'), alpha = 0.2)+
  geom_point(aes(y = NEP_in, color = 'NEP'), alpha = 0.2)+
  scale_color_manual(values = c('#0073C2FF', '#EFC000FF', '#868686FF'))+
  ylab(expression(paste("Inputs (mol DIC "," ",hr^-1,")")))+
  scale_y_log10(limits = c(.00000001, 1e2),
                breaks = trans_breaks("log10", function(x) 10^x),
                labels = trans_format("log10", math_format(10^.x)))+
  scale_x_datetime(date_breaks = "1 month", date_labels =  "%b")+
  theme(legend.position = c(0.8, 0.2),
        legend.title = element_blank(),
        legend.background = element_blank(),
        axis.title.x = element_blank())+
  guides(colour = guide_legend(override.aes = list(alpha = 1)))
Tac_scaled_in
```



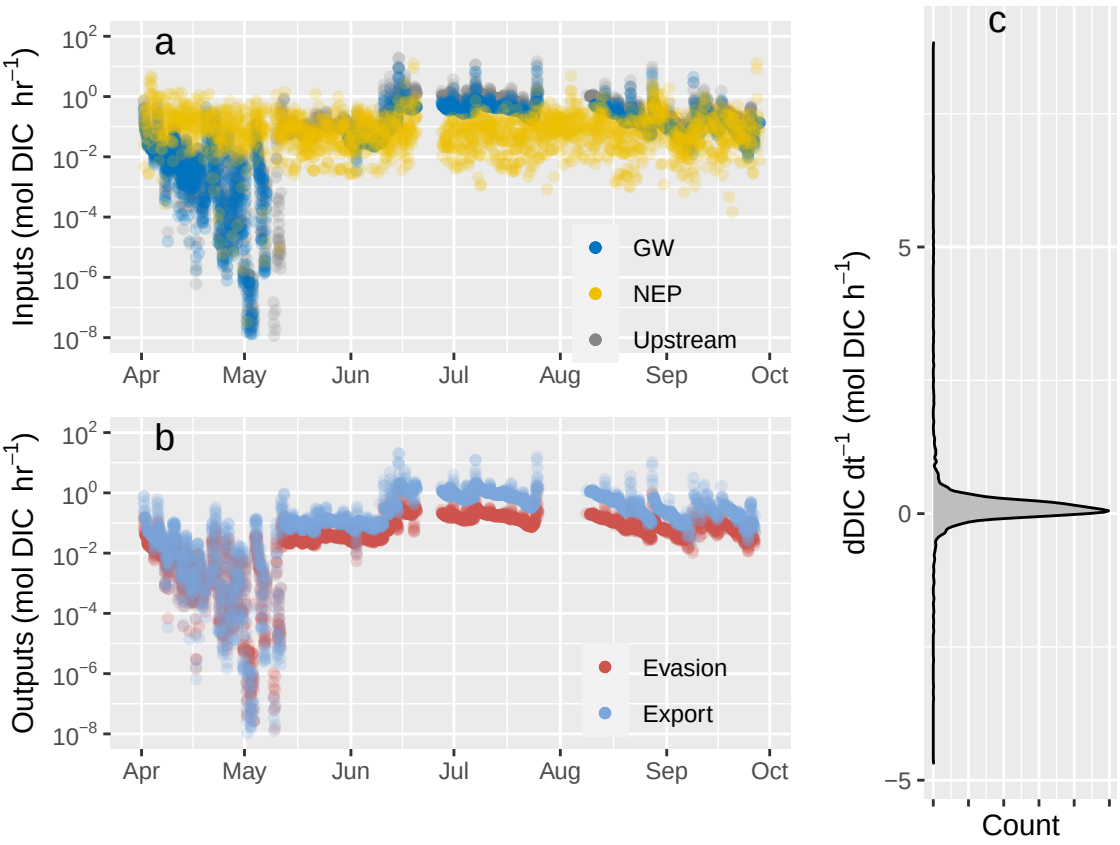

```
Tac_scaled_out <- ggplot(Tac_scale,
                         aes(x = Timestamp))+
  geom_point(aes(y = F_out, color = 'Evasion'), alpha = 0.2)+
  geom_point(aes(y = DIC_out, color = 'Export'), alpha = 0.2)+
  scale_color_manual(values = c("#CD534CFF", "#7AA6DCFF"))+
  ylab(expression(paste("Outputs (mol DIC "," ",hr^-1,")")))+
  scale_y_log10(limits = c(.00000001, 1e2),
                breaks = trans_breaks("log10", function(x) 10^x),
                labels = trans_format("log10", math_format(10^.x)))+
  scale_x_datetime(date_breaks = "1 month", date_labels =  "%b")+
  theme(legend.position = c(0.8, 0.2),
        legend.title = element_blank(),
        legend.background = element_blank(),
        axis.title.x = element_blank())+
  guides(colour = guide_legend(override.aes = list(alpha = 1)))
Tac_scaled_out
```




```
Tac_scaled_dCdt <- ggplot(Tac_scale)+
  geom_density(aes(x = dC_dt), fill = 'grey')+
  xlab(expression(paste("dDIC ", dt^-1, ' (mol DIC ', h^-1, ')')))+
  ylab('Count')+
  theme(axis.text.x = element_blank())+
  coord_flip()
Tac_scaled_dCdt
```




```
Fig6 <- plot_grid(plot_grid(Tac_scaled_in, Tac_scaled_out,
                            nrow = 2, align = 'hv',
                            labels = 'auto', label_fontface = 'plain',
                            hjust = -8, vjust = 1.75),
                  Tac_scaled_dCdt,
                  ncol = 2,
                  rel_widths = c(3,1),
                  labels = c('', 'c'), label_fontface = 'plain',
                  hjust = -9,
                  align = 'hv')
Fig6
```




## Table 3

Here, we assess the influence of rainfall on volumetric fluxes using the Pearson correlation coefficient.

```
Tac_corr <- Tac_scale %>%
  group_by(Timestamp = date(Timestamp)) %>%
  summarise(
    #fluxes
    sumGW = sum(GWCO2, na.rm = TRUE),
    sumF = sum(CO2_efflux, na.rm = TRUE),
    sumNEP = sum(direct_met_mol, na.rm = TRUE),
    sumDICin = sum(DIC_in, na.rm = TRUE),
    sumDICout = sum(DIC_out, na.rm = TRUE),

    # rainfall
    sumrain = sum(Rainfall, na.rm = TRUE),

    # pH
    meanpH = mean(pH, na.rm = TRUE),
    sdpH = sd(pH, na.rm = TRUE)
  )

cor.test(y = Tac_corr$sumGW,
         x = Tac_corr$sumrain,
         method = 'pearson')
```

```
##
##  Pearson's product-moment correlation
##
## data:  Tac_corr$sumrain and Tac_corr$sumGW
## t = 5.2348, df = 179, p-value = 4.593e-07
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.2307783 0.4844804
## sample estimates:
##       cor
## 0.3643711
```

```
cor.test(y = Tac_corr$sumDICin,
         x = Tac_corr$sumrain,
         method = 'pearson')
```

```
##
##  Pearson's product-moment correlation
##
## data:  Tac_corr$sumrain and Tac_corr$sumDICin
## t = 5.2913, df = 179, p-value = 3.52e-07
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.2344928 0.4874799
## sample estimates:
##       cor
## 0.3677718
```

```
cor.test(y = Tac_corr$sumNEP,
         x = Tac_corr$sumrain,
         method = 'pearson')
```

```
##
##  Pearson's product-moment correlation
##
## data:  Tac_corr$sumrain and Tac_corr$sumNEP
## t = 3.1205, df = 179, p-value = 0.002105
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.08406674 0.36103598
## sample estimates:
##       cor
## 0.2271394
```

```
cor.test(y = Tac_corr$sumF,
         x = Tac_corr$sumrain,
         method = 'pearson')
```

```
##
##  Pearson's product-moment correlation
##
## data:  Tac_corr$sumrain and Tac_corr$sumF
```

```
## t = 4.0203, df = 179, p-value = 8.555e-05
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.1481416 0.4161701
## sample estimates:
##       cor
## 0.2877811
```

```
cor.test(y = Tac_corr$sumDICout,
         x = Tac_corr$sumrain,
         method = 'pearson')
```

```
##
##  Pearson's product-moment correlation
##
## data:  Tac_corr$sumrain and Tac_corr$sumDICout
## t = 5.2913, df = 179, p-value = 3.52e-07
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.2344928 0.4874799
## sample estimates:
##       cor
## 0.3677718
```

## Figure 7

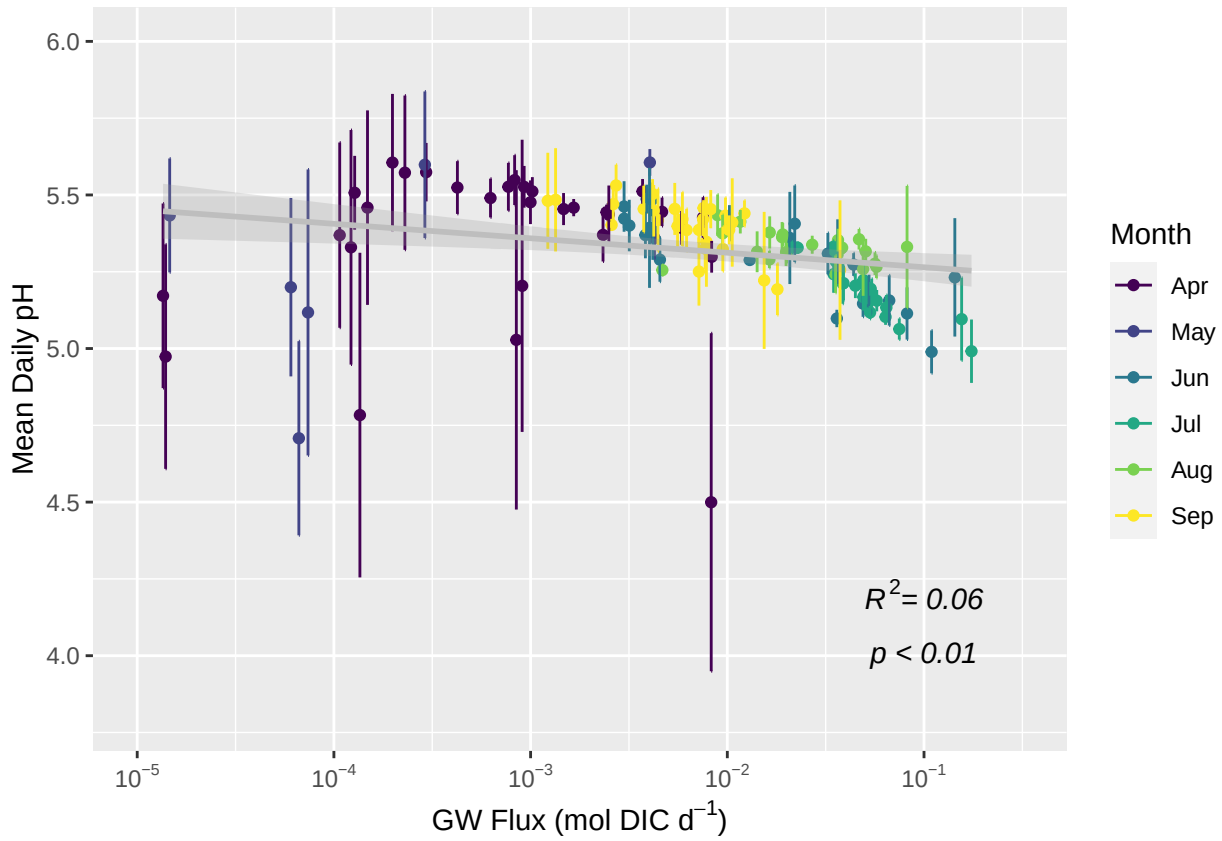
Finally, we determine the effect of groundwater DIC inputs to the reach as drivers of pH decreases, both at seasonal and episodic scales.

```
summary(lm(data = Tac_corr %>%
             filter(sumGW > 1e-5),
           meanpH ~ log10(sumGW))
)
```

```
##
## Call:
## lm(formula = meanpH ~ log10(sumGW), data = Tac_corr %>% filter(sumGW >
##     1e-05))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.81612 -0.07208  0.03779  0.10463  0.27544
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   5.21765    0.03695 141.208   <2e-16 ***
## log10(sumGW) -0.04710    0.01584  -2.973   0.0035 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1653 on 134 degrees of freedom
## Multiple R-squared:  0.06189,    Adjusted R-squared:  0.05489
## F-statistic:  8.84 on 1 and 134 DF,  p-value: 0.003496
```

```
Fig7 <- ggplot(Tac_corr %>%
                 filter(sumGW > 1e-5),
               aes(x = sumGW,
                   y = meanpH))+
  geom_point(aes(color = month(Timestamp, label = TRUE, abbr = TRUE)))+
  geom_errorbar(aes(ymin = meanpH - sdpH,
                    ymax = meanpH + sdpH,
                    color = month(Timestamp, label = TRUE, abbr = TRUE)))+
  geom_smooth(method = 'lm',
              color = 'grey',
              alpha = 0.25)+
  scale_x_log10(breaks = trans_breaks("log10", function(x) 10^x),
                labels = trans_format("log10", math_format(10^.x)),
                limits = c(1e-5, 10^(-0.5)))+
  ylim(3.8, 6)+
  scale_color_viridis_d(name = 'Month')+
  xlab(expression(paste('GW Flux (mol DIC ', d^-1, ")")))+
  ylab('Mean Daily pH')+
  annotate(x = 0.1, y = 4, 'text', label = expression(italic("p < 0.01")))+
  annotate(x = 0.1, y = 4.2, 'text', label = expression(italic(paste(R^2,"= 0.06"))))
Fig7
```

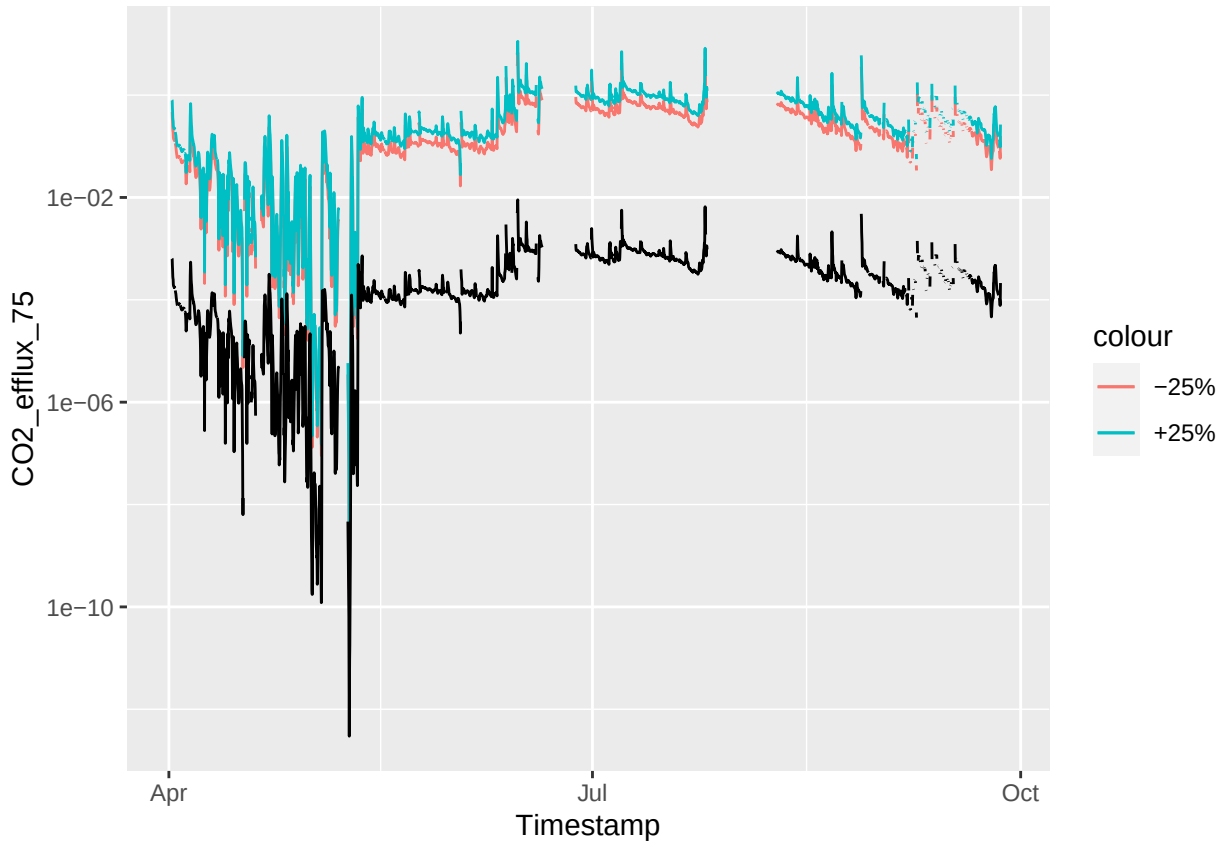
```
## `geom_smooth()` using formula 'y ~ x'
```



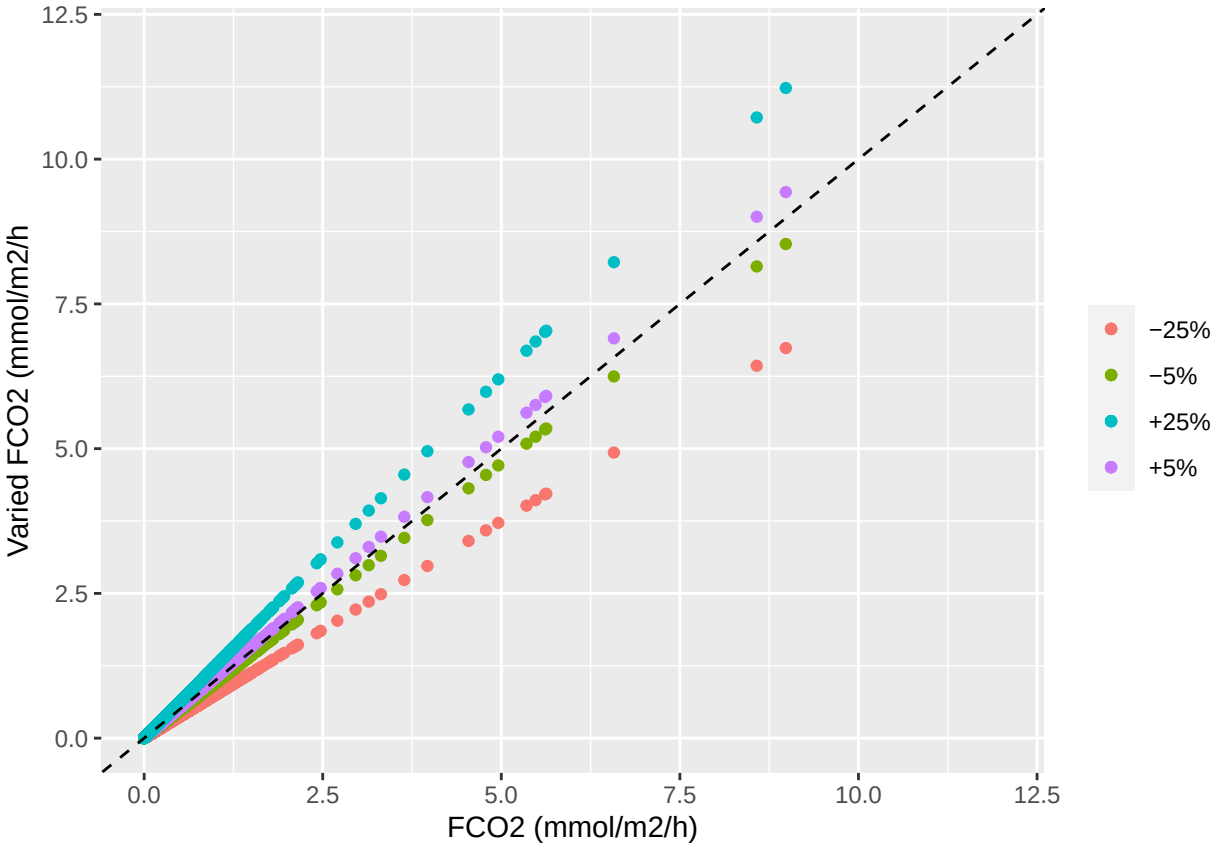
## Figure S5

Here, we perform a sensitivity analysis on areal fluxes. We determine to what extent variation in 1) gas exchange influences changes in F[CO2] and NEP and 2) groundwater discharge influences groudwater CO2 inputs. We explore variation at +-5% and +-25% of parameter values.

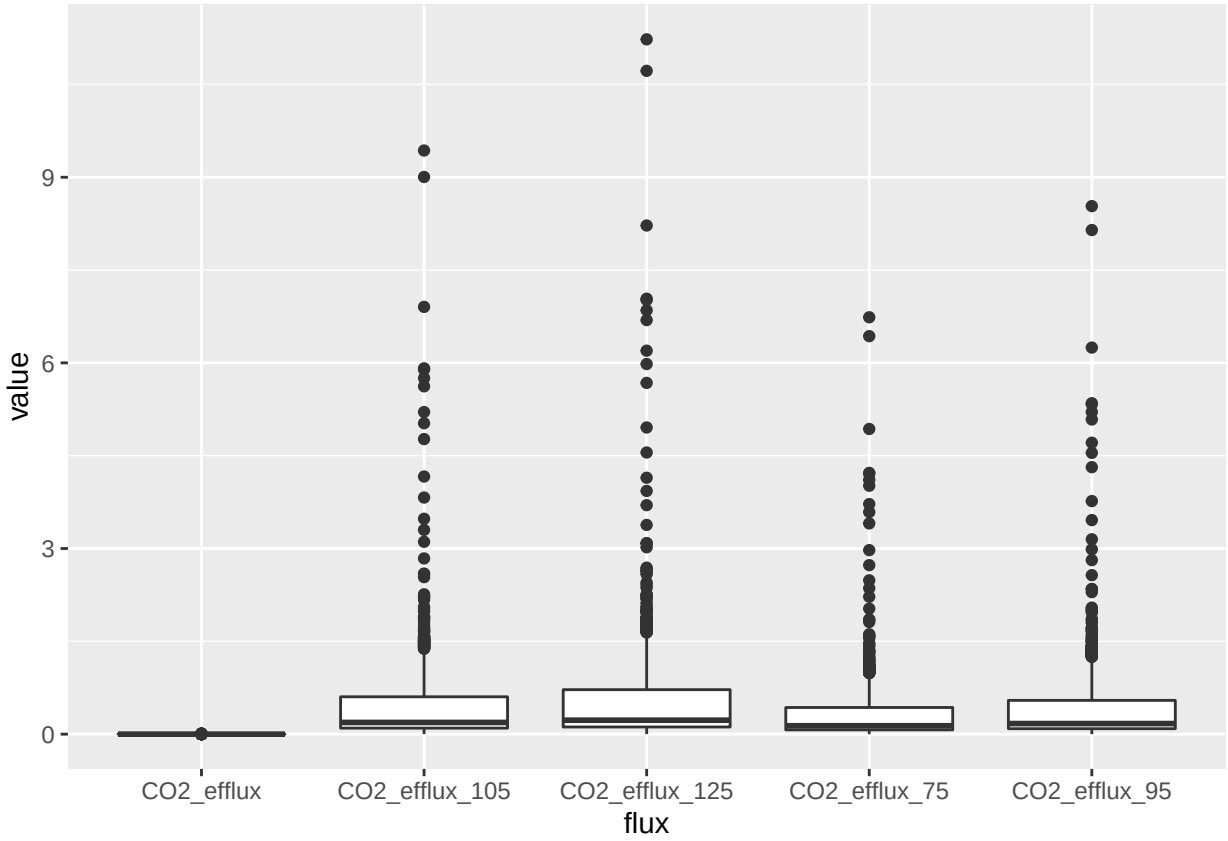
```
Tac_sens_gas <- Tac_flux %>%
  dplyr::select(Timestamp,                    # timestamp
                mean_depth,                   # for unit conversions
                k600, K600, KCO2, KO2,        # gas exchange measures
                CO2_aq, CO2_sat, CO2_efflux,  # CO2 efflux related measurements
                DO.obs, Osat, nep = direct_met_mol) # NEP related measurements

Tac_sens_gas_calc <- Tac_sens_gas %>%
  mutate(
    # modulate KCO2 by +- 25%
    KCO2_75 = KCO2 * 0.75,
    KCO2_95 = KCO2 * 0.95,
    KCO2_105 = KCO2 * 1.05,
    KCO2_125 = KCO2 * 1.25,
    # modulate KO2 by +- 25%
    KO2_75 = KO2 * 0.75,
    KO2_95 = KO2 * 0.95,
    KO2_105 = KO2 * 1.05,
    KO2_125 = KO2 * 1.25,
    # recalculate efflux with modulated KCO2s
    #CO2_efflux = CO2_efflux
    CO2_efflux_75 = ((CO2_aq - CO2_sat)*KCO2_75*mean_depth)*1000,
    CO2_efflux_95 = ((CO2_aq - CO2_sat)*KCO2_95*mean_depth)*1000,
    CO2_efflux_105 = ((CO2_aq - CO2_sat)*KCO2_105*mean_depth)*1000,
    CO2_efflux_125 = ((CO2_aq - CO2_sat)*KCO2_125*mean_depth)*1000,
    # recalculate NEP with modulated KO2s
    nep_75 = ((diff(DO.obs) - (KO2_75*(Osat - DO.obs)))*(1/32)*mean_depth*-1)*1000,
    nep_95 = ((diff(DO.obs) - (KO2_95*(Osat - DO.obs)))*(1/32)*mean_depth*-1)*1000,
    nep_105 = ((diff(DO.obs) - (KO2_105*(Osat - DO.obs)))*(1/32)*mean_depth*-1)*1000,
    nep_125 = ((diff(DO.obs) - (KO2_125*(Osat - DO.obs)))*(1/32)*mean_depth*-1)*1000
  )

# plot: CO2 efflux sensitivity analysis
ggplot(Tac_sens_gas_calc,
       aes(x = Timestamp))+
  geom_line(aes(y = CO2_efflux_75,
                color = '-25%'))+
  geom_line(aes(y = CO2_efflux_125,
                color = '+25%'))+
  geom_line(aes(y = CO2_efflux),
            color = 'black')+
  scale_y_log10()
```



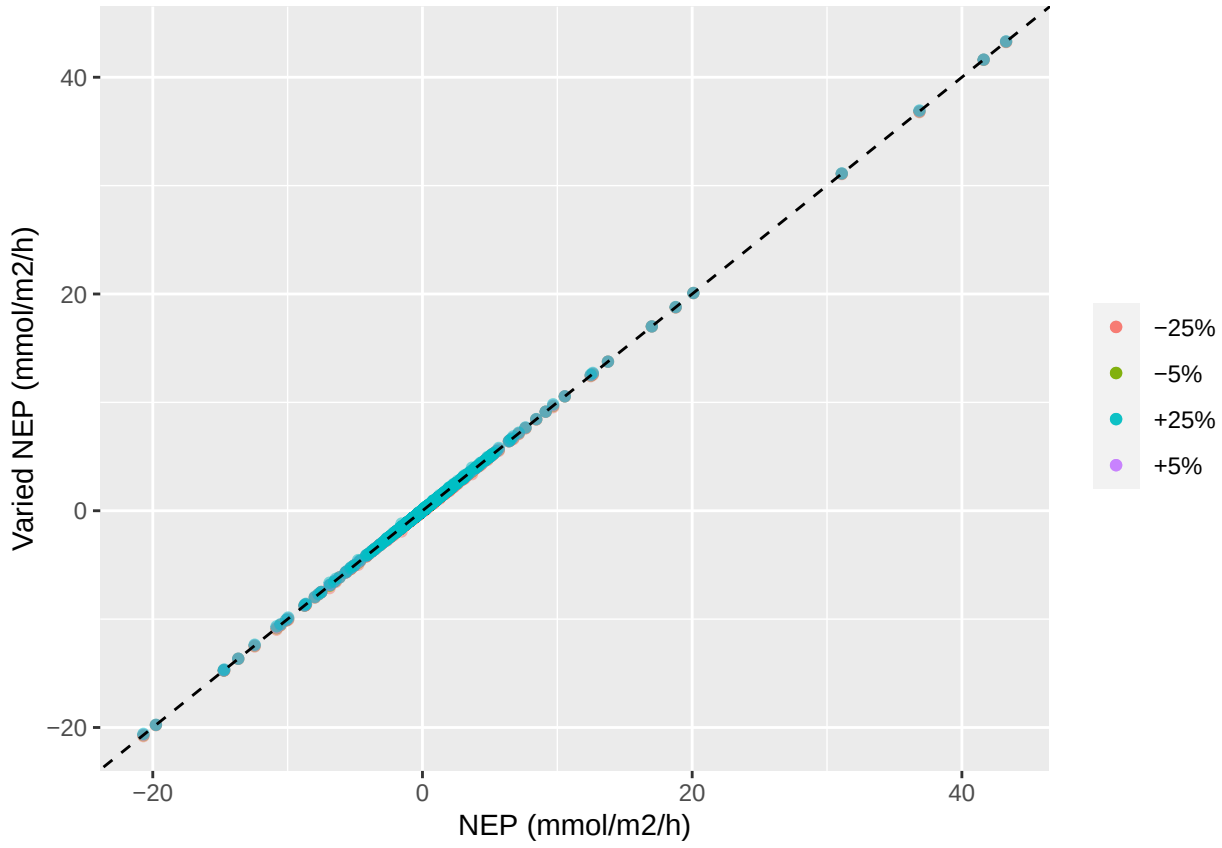

```
plot_sens_FCO2 <- ggplot(Tac_sens_gas_calc,
       aes(x = CO2_efflux*1000))+
  geom_point(aes(y = CO2_efflux_75,
                 color = '-25%'))+
  geom_point(aes(y = CO2_efflux_95,
                 color = '-5%'))+
  geom_point(aes(y = CO2_efflux_105,
                 color = '+5%'))+
  geom_point(aes(y = CO2_efflux_125,
                 color = '+25%'))+
  geom_abline(slope = 1, intercept = 0, linetype = 'dashed')+
  labs(x = 'FCO2 (mmol/m2/h)',
       y = 'Varied FCO2 (mmol/m2/h')+
  lims(x = c(0, 12),
       y  = c(0, 12))+
  theme(legend.title = element_blank())
plot_sens_FCO2
```



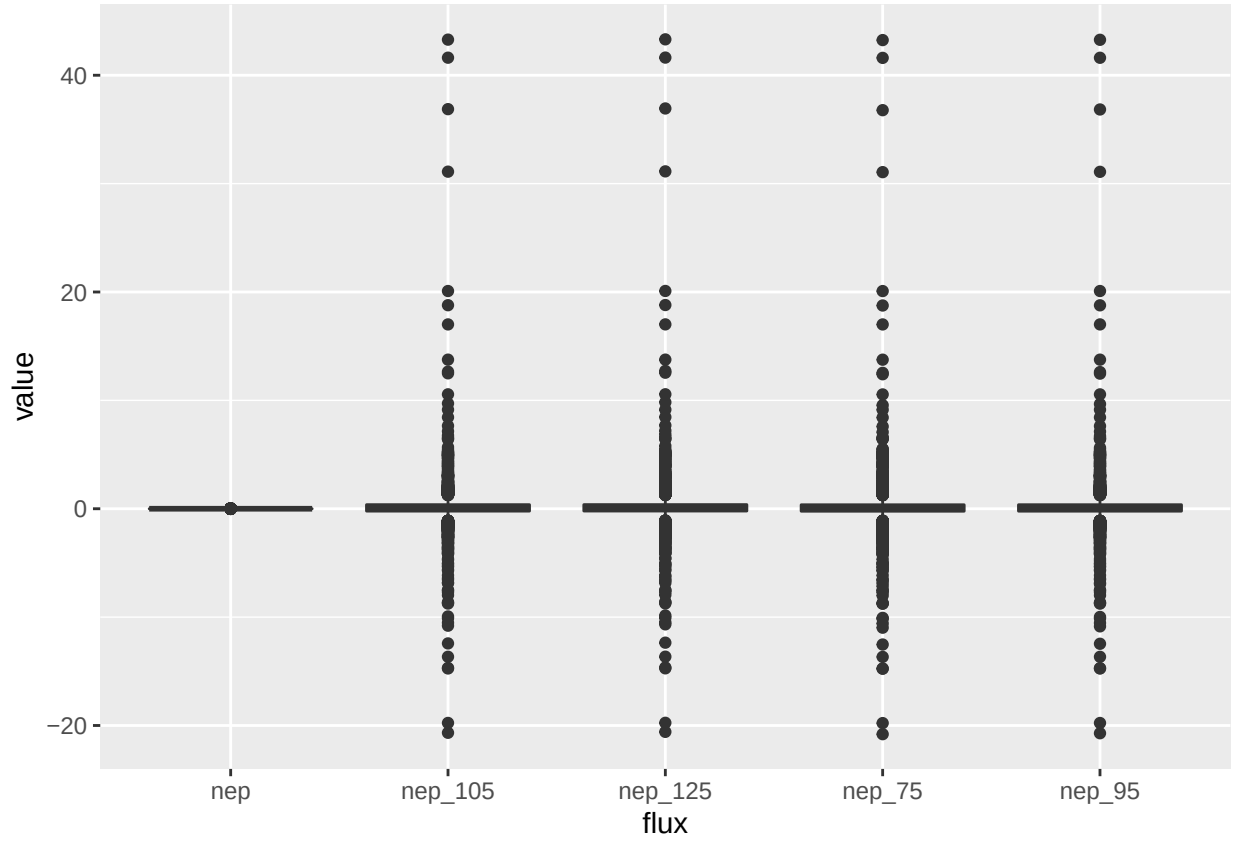

```
# as boxplots
Tac_sens_gas_calc %>%
  dplyr::select(CO2_efflux_75,CO2_efflux_95,CO2_efflux, CO2_efflux_105,CO2_efflux_125) %>%
  tidyr::pivot_longer(everything(),
                      names_to = 'flux', values_to = 'value') %>%
  ggplot(aes(x = flux, y = value))+
  geom_boxplot()
```




```
# change in gas exchange increases the highest rates of evasion

# plot: NEP sensitivity analysis
plot_sens_nep <- ggplot(Tac_sens_gas_calc,
       aes(x = nep*1000))+
  geom_point(aes(y = nep_75,
                 color = '-25%'),
             alpha = 0.5)+
  geom_point(aes(y = nep_95,
                 color = '-5%'),
             alpha = 0.5)+
  geom_point(aes(y = nep_105,
                 color = '+5%'),
             alpha = 0.5)+
  geom_point(aes(y = nep_125,
                 color = '+25%'),
             alpha = 0.5)+
  geom_abline(slope = 1, intercept = 0, linetype = 'dashed')+
  labs(x = 'NEP (mmol/m2/h)',
       y = 'Varied NEP (mmol/m2/h)')+
  theme(legend.title = element_blank())
plot_sens_nep
```

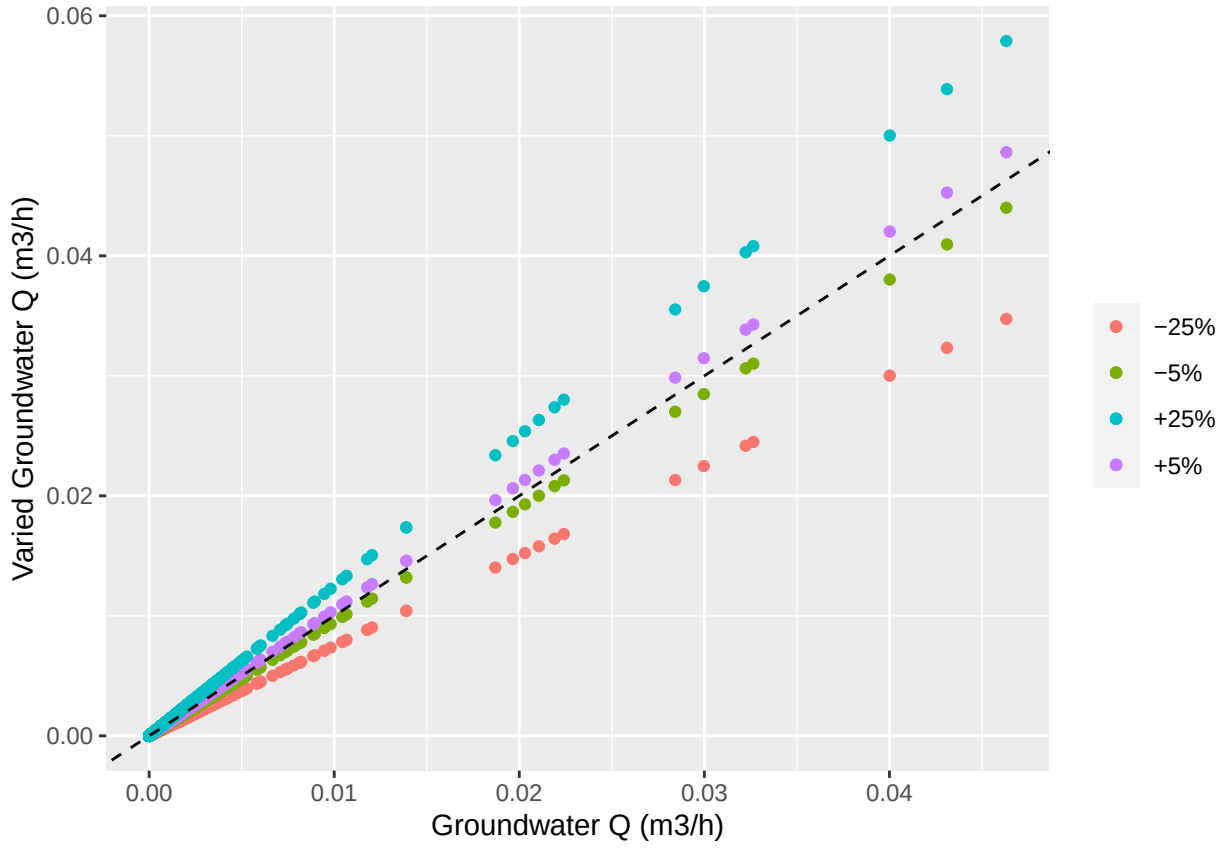



```
Tac_sens_gas_calc %>%
  dplyr::select(nep,nep_75,nep_125, nep_95,nep_105) %>%
  tidyr::pivot_longer(everything(),
                      names_to = 'flux', values_to = 'value') %>%
  ggplot(aes(x = flux, y = value))+
  geom_boxplot()
```

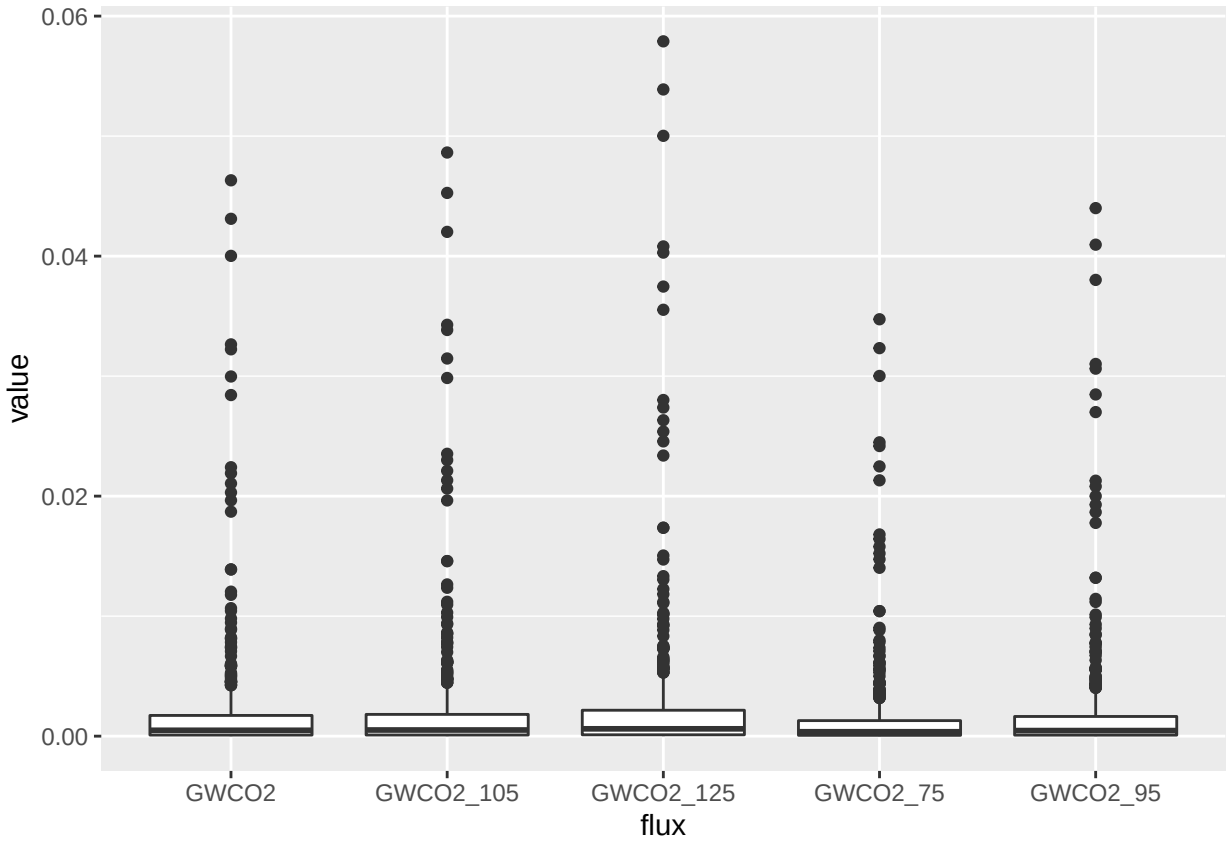



```
# gas exchange does not exert much control on NEP in the study reach


# on groundwater flux
Tac_sens_gw <- Tac_flux %>%
  dplyr::select(Timestamp,            # timestamp
                wellCO2_aq, GWCO2,    # aqueous CO2 in the well (mol/m3) and GWCO2 (mol/h)
                GW_v)                 # groundwater discharge (m3/h)

Tac_sens_gw_calc <- Tac_sens_gw %>%
  mutate(
    GW_v_75 = GW_v * 0.75,
    GW_v_95 = GW_v * 0.95,
    GW_v_105 = GW_v * 1.05,
    GW_v_125 = GW_v * 1.25,
    GWCO2_75 = GW_v_75*wellCO2_aq,
    GWCO2_95 = GW_v_95*wellCO2_aq,
    GWCO2_105 = GW_v_105*wellCO2_aq,
    GWCO2_125 = GW_v_125*wellCO2_aq
  )

plot_sens_gw <- ggplot(Tac_sens_gw_calc,
       aes(x = GWCO2))+
  geom_point(aes(y = GWCO2_75,
                 color = '-25%'))+
  geom_point(aes(y = GWCO2_95,
```

```
                color = '-5%'))+
  geom_point(aes(y = GWCO2_105,
                color = '+5%'))+
  geom_point(aes(y = GWCO2_125,
                color = '+25%'))+
  geom_abline(slope = 1, intercept = 0, linetype = 'dashed')+
  labs(x = 'Groundwater Q (m3/h)',
       y = 'Varied Groundwater Q (m3/h)')+
  theme(legend.title = element_blank())
plot_sens_gw
```




```
Tac_sens_gw_calc %>%
  dplyr::select(GWCO2,GWCO2_75,GWCO2_95, GWCO2_105,GWCO2_125) %>%
  tidyr::pivot_longer(everything(),
                      names_to = 'flux', values_to = 'value') %>%
  ggplot(aes(x = flux, y = value))+
  geom_boxplot()
```

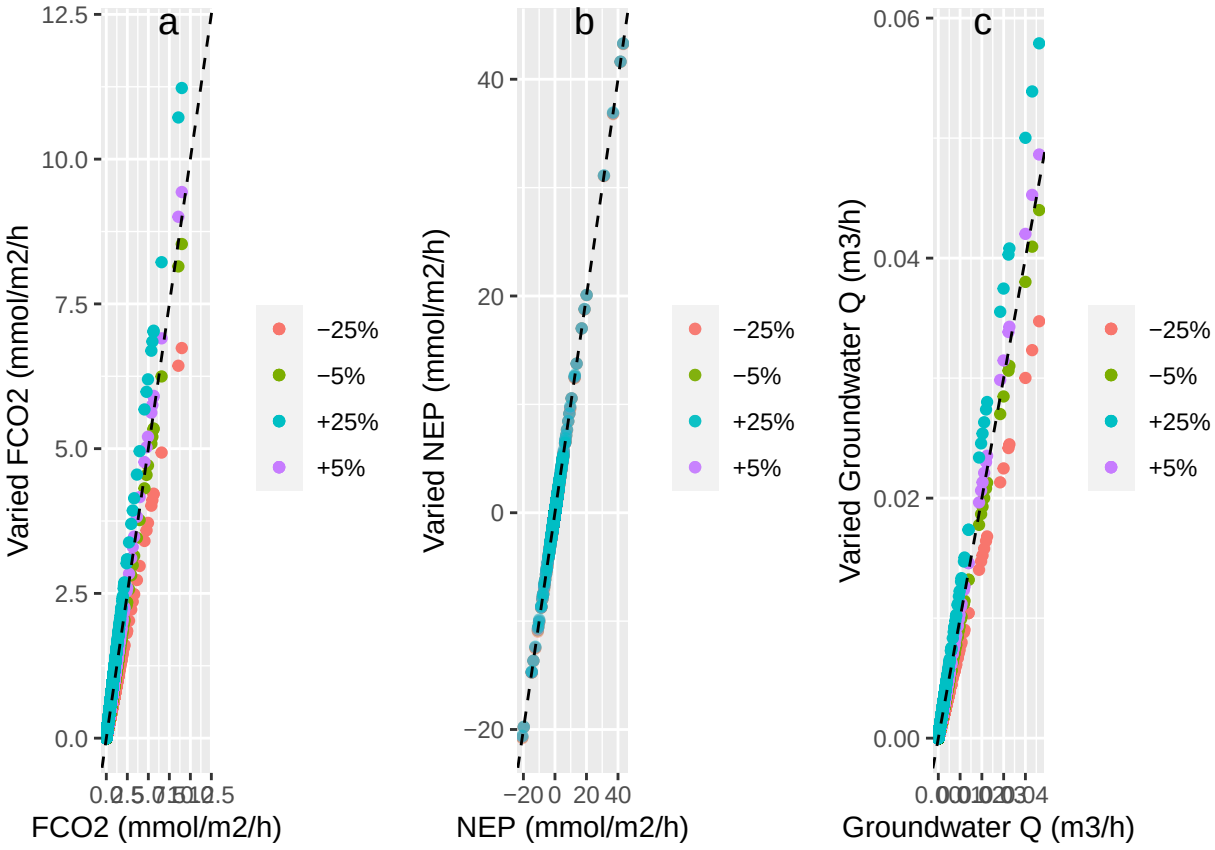



```
# GW velocity, and by proxy GW discharge, influence the input of CO2 from

# evaluate depth on FCO2 and NEP
Tac_flux %>%
  select(Timestamp, mean_depth, )
```

```
## # A tibble: 4,324 x 2
##    Timestamp           mean_depth
##    <dttm>                   <dbl>
##  1 2013-04-01 00:00:00      0.243
##  2 2013-04-01 01:00:00      0.246
##  3 2013-04-01 02:00:00      0.250
##  4 2013-04-01 03:00:00      0.255
##  5 2013-04-01 04:00:00      0.258
##  6 2013-04-01 05:00:00      0.261
##  7 2013-04-01 06:00:00      0.261
##  8 2013-04-01 07:00:00      0.260
##  9 2013-04-01 08:00:00      0.257
## 10 2013-04-01 09:00:00      0.253
## # ... with 4,314 more rows
```

```
FigS5 <- plot_grid(plot_sens_FCO2,
                   plot_sens_nep,
                   plot_sens_gw,
                   labels = 'auto', label_fontface = 'plain',
```

```
                    ncol = 3, align = 'hv',
                    hjust = -8)
FigS5
```




## Session info

```
pander(sessionInfo())
```

**R version 4.0.3 (2020-10-10)**

**Platform:** x86_64-w64-mingw32/x64 (64-bit)

**locale:** *LC_COLLATE=English_United States.1252*, *LC_CTYPE=English_United States.1252*, *LC_MONETARY=English_United States.1252*, *LC_NUMERIC=C* and *LC_TIME=English_United States.1252*

**attached base packages:** *stats*, *graphics*, *grDevices*, *utils*, *datasets*, *methods* and *base*

**other attached packages:** *pander(v.0.6.4)*, *hydrostats(v.0.2.8)*, *ARTool(v.0.11.1)*, *lsmeans(v.2.30-0)*, *emmeans(v.1.7.1-1)*, *purrr(v.0.3.4)*, *ellipse(v.0.4.2)*, *lmodel2(v.1.7-3)*, *ggsci(v.2.9)*, *scales(v.1.1.1)*, *ggrepel(v.0.9.1)*, *cowplot(v.1.1.1)*, *ggplot2(v.3.3.5)*, *tidyquant(v.1.0.3)*, *quantmod(v.0.4.18)*, *TTR(v.0.24.3)*, *PerformanceAnalytics(v.2.0.4)*, *xts(v.0.12.1)*, *zoo(v.1.8-9)*, *lubridate(v.1.8.0)*, *tidyr(v.1.1.4)* and *dplyr(v.1.0.7)*

**loaded via a namespace (and not attached):** *httr(v.1.4.2)*, *viridisLite(v.0.4.0)*, *jsonlite(v.1.7.2)*, *splines(v.4.0.3)*, *carData(v.3.0-4)*, *assertthat(v.0.2.1)*, *highr(v.0.9)*, *yaml(v.2.2.1)*, *backports(v.1.4.1)*,

*pillar(v.1.6.4)*, *lattice(v.0.20-45)*, *glue(v.1.6.0)*, *quadprog(v.1.5-8)*, *digest(v.0.6.29)*, *minqa(v.1.2.4)*, *colorspace(v.2.0-2)*, *sandwich(v.3.0-1)*, *htmltools(v.0.5.2)*, *Matrix(v.1.4-0)*, *plyr(v.1.8.6)*, *pkgconfig(v.2.0.3)*, *broom(v.0.7.10)*, *xtable(v.1.8-4)*, *mvtnorm(v.1.1-3)*, *lme4(v.1.1-27.1)*, *tibble(v.3.1.6)*, *mgcv(v.1.8-38)*, *farver(v.2.1.0)*, *generics(v.0.1.1)*, *car(v.3.0-12)*, *ellipsis(v.0.3.2)*, *TH.data(v.1.1-0)*, *withr(v.2.4.3)*, *cli(v.3.1.0)*, *survival(v.3.2-13)*, *magrittr(v.2.0.1)*, *crayon(v.1.4.2)*, *estimability(v.1.3)*, *evaluate(v.0.14)*, *fansi(v.0.5.0)*, *nlme(v.3.1-153)*, *MASS(v.7.3-54)*, *tools(v.4.0.3)*, *lifecycle(v.1.0.1)*, *multcomp(v.1.4-17)*, *stringr(v.1.4.0)*, *munsell(v.0.5.0)*, *compiler(v.4.0.3)*, *rlang(v.0.4.12)*, *grid(v.4.0.3)*, *nloptr(v.1.2.2.3)*, *rstudioapi(v.0.13)*, *labeling(v.0.4.2)*, *rmarkdown(v.2.11)*, *boot(v.1.3-28)*, *gtable(v.0.3.0)*, *codetools(v.0.2-18)*, *abind(v.1.4-5)*, *DBI(v.1.1.1)*, *curl(v.4.3.2)*, *R6(v.2.5.1)*, *knitr(v.1.37)*, *fastmap(v.1.1.0)*, *utf8(v.1.2.2)*, *Quandl(v.2.11.0)*, *stringi(v.1.7.6)*, *Rcpp(v.1.0.7)*, *vctrs(v.0.3.8)*, *tidyselect(v.1.1.1)* and *xfun(v.0.29)*