Particle Interception as a Function of Reynolds Number and Collector Density

Jordan Wingenroth

8/26/2020

Contents

Objective(s)

In the interest of documenting our analytic methods, I've decided to make one comprehensive, hopefully lay-readable document including all steps from our raw laboratory data to the model results. I won't be able to include the turbulence analyses in time for our meeting (8/27), but that required Laurel's Matlab script anyways.

Data sourcing and tidying

We entered data into Google Sheets as we processed samples in the lab. I downloaded .csv versions of these data in 2019, then made a few modifications as I explored confounders. Today, I added data from runs done in my absence to this dataset. Field names, date syntax, and other conventions weren't kept consistent, so some data wrangling is necessary. I also removed the blank data at the beginning of some files manually.

```
library(tidyverse)
```

```
pumpfiles <- list.files("../data/peristaltic pumps/")
trapfiles <- list.files("../data/sediment traps/")

length(pumpfiles)
```

```
## [1] 26
```

```
length(trapfiles)
```

```
## [1] 21
```

As you see, there are different numbers of files for sediment trap versus peristaltic pump data.

```
pumpdate <- str_sub(pumpfiles,0,6)
trapdate <- str_sub(trapfiles,0,6)

tibble(pumpdate, pumpdate %in% trapdate)
```

```
## # A tibble: 26 x 2
##    pumpdate `pumpdate %in% trapdate`
##    <chr>    <lgl>
##  1 180920   FALSE
##  2 180928   FALSE
##  3 181005   TRUE
##  4 181019   FALSE
##  5 181115   TRUE
##  6 181204   TRUE
##  7 190131   TRUE
##  8 190308   FALSE
##  9 190321   TRUE
## 10 190417   TRUE
## # ... with 16 more rows
```

In all, five runs are missing sediment trap data, meaning they won't be used in our final, published analyses.

Suspended sediment data

Let's turn our attention to the suspended concentration data (i.e., pump data).

```
pumpdata <- lapply(pumpfiles, function(x) read_csv(paste0("../data/peristaltic pumps/",x)))

names(pumpdata) <- pumpdate

x <- pumpdata

tidypump <- lapply(seq_along(x), function(i) {
  select(x[[i]],
         loc = Location,
         ht = Height,
         t = `time series`,
         mvc = contains("(ppm)")) %>%
    filter(t < 21) %>% #filter a few timepoints outside the normal window
    mutate(t = (t-min(t)+1)*300, #convert from timestep to seconds
           mvc = as.numeric(mvc),
           date = as.numeric(names(x)[[i]])) %>%
    filter(mvc < 80) #filter out a few early-timestep anomalies
  }
  )

pump <- bind_rows(tidypump)

pump
```

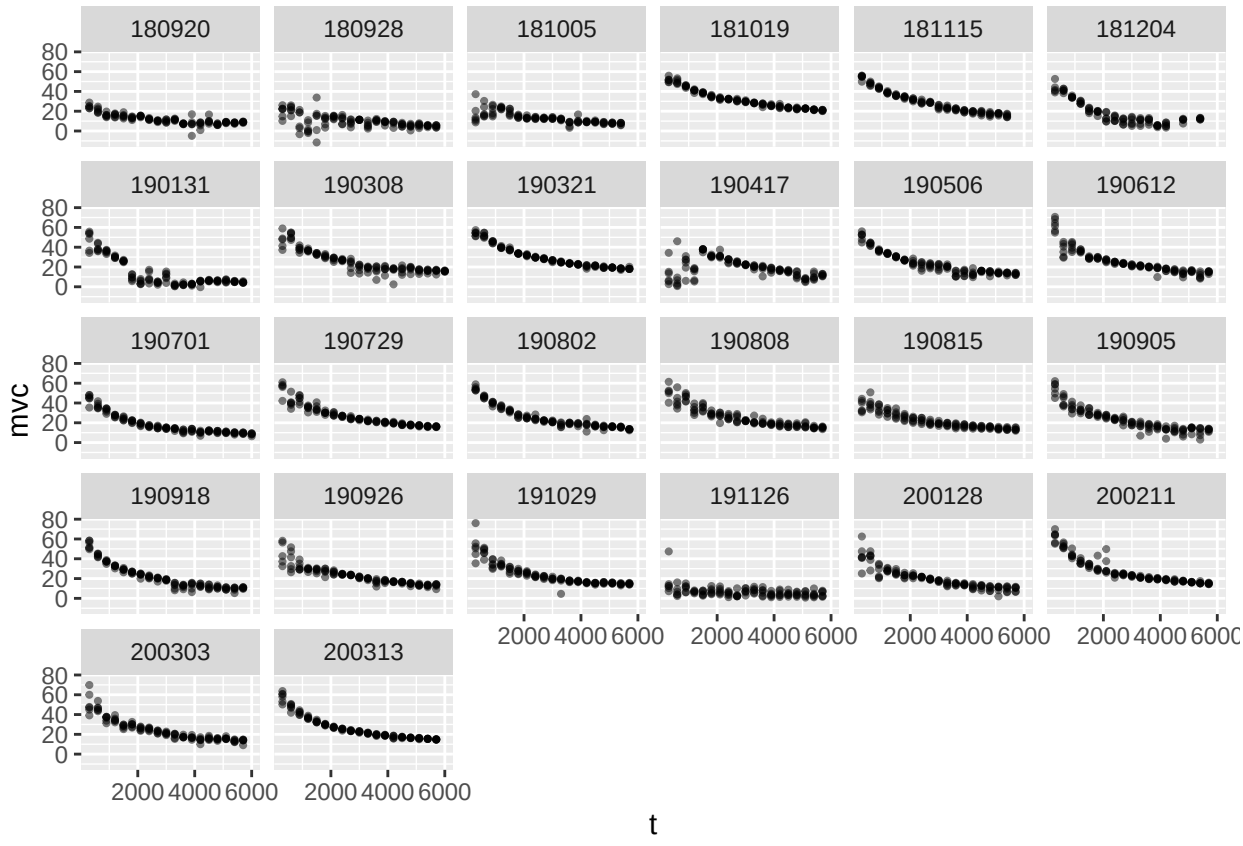
```
## # A tibble: 2,935 x 5
##    loc      ht     t   mvc   date
##    <chr> <dbl> <dbl> <dbl>  <dbl>
```

```
##  1 U         5   300  23.0 180920
##  2 U        14   300  28.8 180920
##  3 U        27   300  25.2 180920
##  4 D        14   300  23.0 180920
##  5 D        27   300  24.6 180920
##  6 U         5   600  19.6 180920
##  7 U        14   600  18.8 180920
##  8 U        27   600  18.4 180920
##  9 D         5   600  24.6 180920
## 10 D        14   600  22.8 180920
## # ... with 2,925 more rows
```

So now we have a long table of our pump data (time × concentration) stratified by run (i.e., treatment), height, and upstream/downstream location.

Let's plot the decay curves separately to see what the data looks like qualitatively.

```
pump %>%
  ggplot(aes(x = t, y = mvc)) +
  geom_point(alpha = .5, size = .75) +
  facet_wrap(~date)
```



Aside from a few runs where **part** or **all** of the data don't follow the decay pattern, the results look pretty good! If we repeated runs as necessary to replace those with erroneous data, that shouldn't be a problem. Here I'll narrow down our dataset by knocking out runs with clear issues.

```
pump <- pump %>%
```

```
#first 3 runs had starting sediment mass of 100g rather than 200g

    filter(date > 181005) %>%

#we can't use runs without sediment mass

    filter(as.character(date) %in% trapdate)

length(unique(pump$date))
```

```
## [1] 20
```

So this leaves 20 rows. Let's join this to our run metadata table.

```
metadata <- read_csv("../data/run_metadata.csv")
```

```
## Parsed with column specification:
## cols(
##   date = col_double(),
##   dowel_density = col_character(),
##   pump_freq = col_double(),
##   growth_days = col_double()
## )
```

```
sum(unique(pump$date) %in% metadata$date)
```

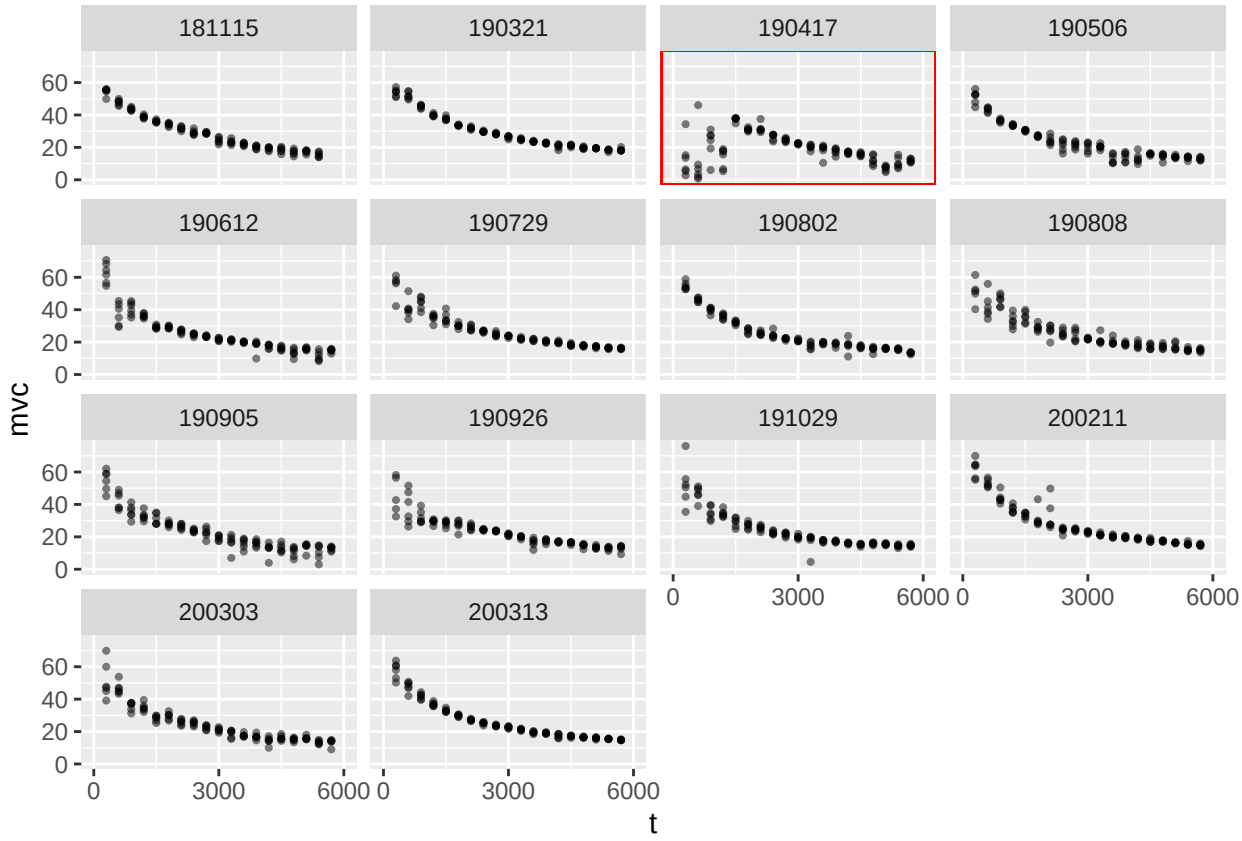
```
## [1] 20
```

Hoorah! We have metadata for all of our runs of interest.

```
pump <- left_join(pump, metadata, by = "date")
```

Now then, let's remove the biofouled runs, which won't be used in assessing our primary hypothesis about Reynolds number and collector density, and we'll see what we're left with.

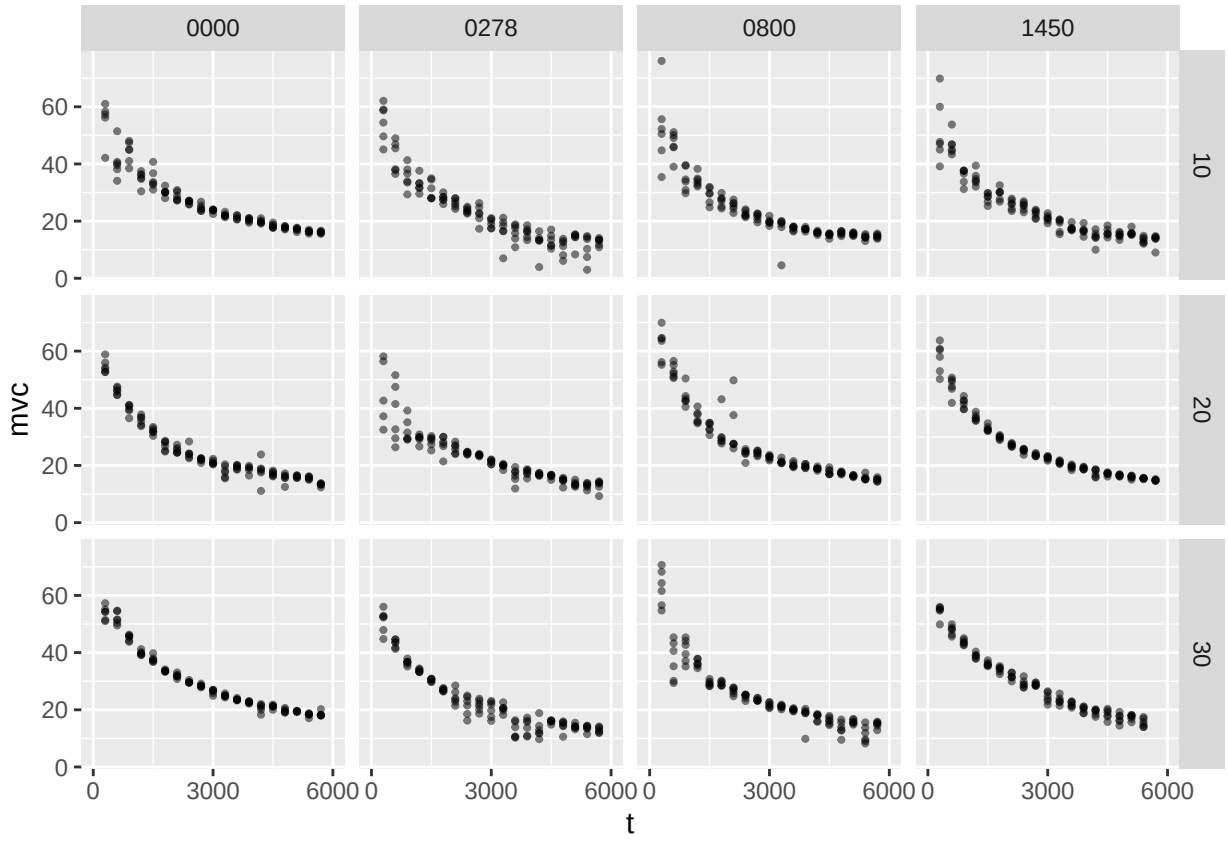
```
pump %>%
  filter(growth_days==0) %>%
  ggplot(aes(x = t, y = mvc)) +
  geom_point(alpha = .5, size = .75) +
  geom_rect(data = filter(pump, date == 190417),
                          fill = NA, colour = "red", xmin = -Inf,xmax = Inf,
            ymin = -Inf,ymax = Inf) +
  facet_wrap(~date) +
  scale_x_continuous(breaks = c(0,3000,6000), limits = c(0,6000))
```



These runs do look quite familiar to me because I think this is about my 10th time through this analysis. `190417` is the run with clear issues, in the early timesteps more specifically. During one of those previous analyses, I found that there were several runs where **part** of the timespan was clearly out of line with the expected decay pattern. Fortunately we tracked metadata about the logistics of laboratory analysis, and it just so happens that the clearly erroneous data were all from samples processed by one researcher in particular. `190417` is the only run out of these 13 where they were involved in processing samples, at least since we started recording initials on our work. We repeated that run on `190506` and got results more in line with exponential decay.

So that leaves 13 runs. Aside from one run at 232 collectors/m^2, the result of a miscalculation during dowel installation, the 12 remaining are a perfect match for our parameter space of 4 densities (including 0 control) and 3 velocities/Re values.

```
pumpfinal <- pump %>%
  filter(growth_days==0, date != 190417, dowel_density != "0232")

pumpfinal %>%
  ggplot(aes(x = t, y = mvc)) +
  geom_point(alpha = .5, size = .75) +
  facet_grid(pump_freq~dowel_density) +
  scale_x_continuous(breaks = c(0,3000,6000), limits = c(0,6000))
```



There are certainly no trends jumping out immediately, which isn't necessarily worrisome since exponential decay is hard to compare by eye. More on that later though.

Sediment trap data

Data import and wrangling follows a similar structure to the pump data.

```
trapdata <- lapply(trapfiles, function(x) read_csv(paste0("../data/sediment traps/",x)))

names(trapdata) <- trapdate

x <- trapdata

tidytrap <- lapply(seq_along(x), function(i) {
  select(x[[i]], station = 1, pre = contains("pre"), post = contains("post"), sed = contains("sed")) %>%
  mutate(date = names(x)[i]) %>%
  mutate_at(vars(pre,post,sed,date), as.numeric)
  }
  )

trap <- bind_rows(tidytrap) %>%
  filter(!is.na(sed))

trap
```

```
## # A tibble: 186 x 5
##    station   pre  post   sed   date
```

```
##    <chr>   <dbl> <dbl> <dbl>  <dbl>
##  1 A        55.2  63.1  7.87 181005
##  2 B        59.1  64.3  5.16 181005
##  3 C        57.6  66.3  8.74 181005
##  4 D        57.1  62.9  5.82 181005
##  5 E        58.4  63.0  4.59 181005
##  6 F        57.6  64.2  6.58 181005
##  7 G        57.0  62.5  5.56 181005
##  8 H        56.0  57.4  1.39 181005
##  9 I        50.6  54.6  3.99 181005
## 10 A        58.5  61.5  3.04 181115
## # ... with 176 more rows
```

So, across 21 runs, we had 186 samples. We were set up to collect 9 per run, but traps broke, filters slipped and spilled sediment, et cetera. $186/21 = 8.8571429$, so our success rate was actually pretty good, although some of those measurements were probably erroneous despite being measured. Let's see what we're working with and then we can maybe try to remove outliers.

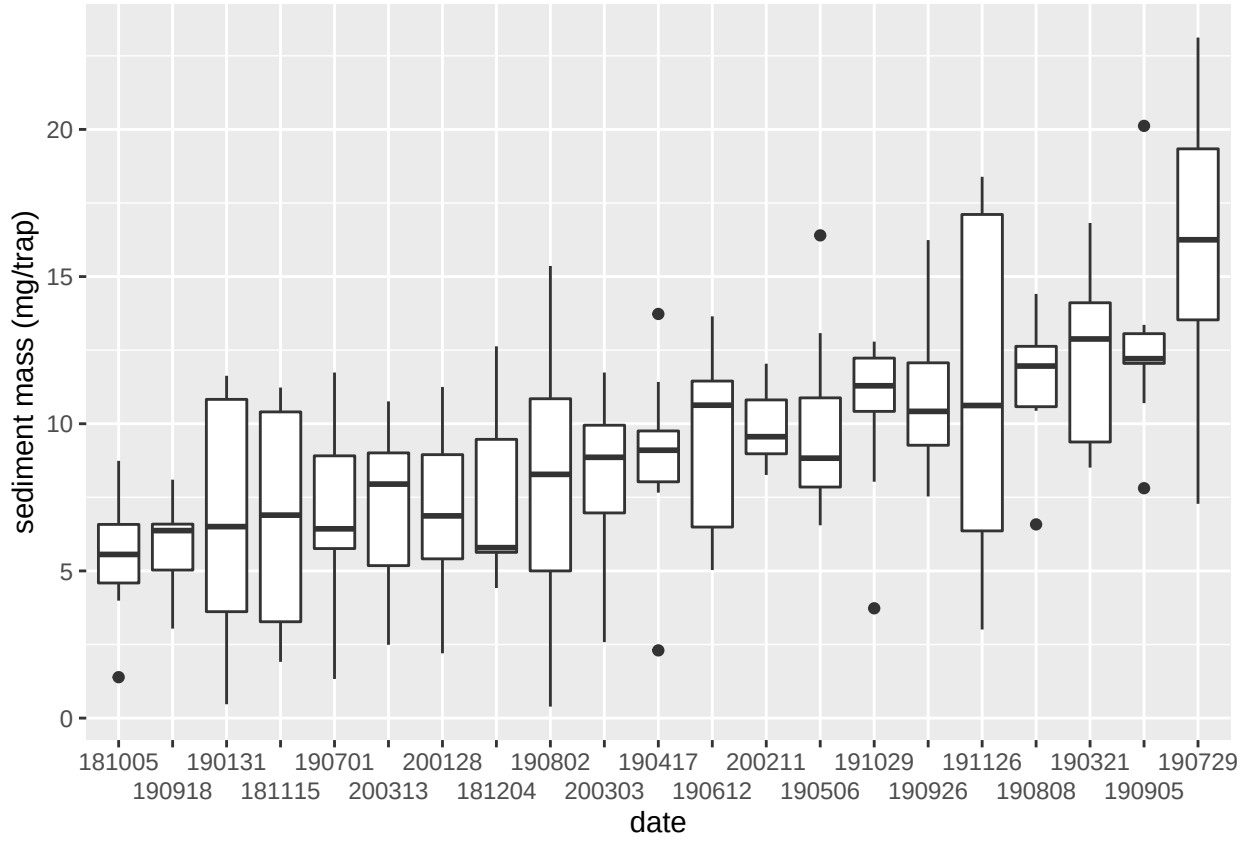
```
lm(sed~station, data = trap) %>% summary()
```

```
##
## Call:
## lm(formula = sed ~ station, data = trap)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8.5862 -2.5888  0.1437  2.0968 11.2300
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  10.7862     0.8399  12.842  < 2e-16 ***
## stationB     -0.9282     1.2026  -0.772  0.44125
## stationC      1.1038     1.1879   0.929  0.35403
## stationD     -1.7638     1.1879  -1.485  0.13936
## stationE     -0.8190     1.1879  -0.690  0.49140
## stationF     -2.0105     1.1879  -1.693  0.09231 .
## stationG     -3.0552     1.2026  -2.540  0.01193 *
## stationH     -3.7190     1.1879  -3.131  0.00204 **
## stationI     -3.2637     1.2026  -2.714  0.00731 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.849 on 177 degrees of freedom
## Multiple R-squared:  0.1395, Adjusted R-squared:  0.1006
## F-statistic: 3.588 on 8 and 177 DF,  p-value: 0.0006955
```

So, downstream traps tended to have less settling. This challenges our assumption of uniformity throughout the test section and might justify the use of a spline, as Justin and Candace had suggested.

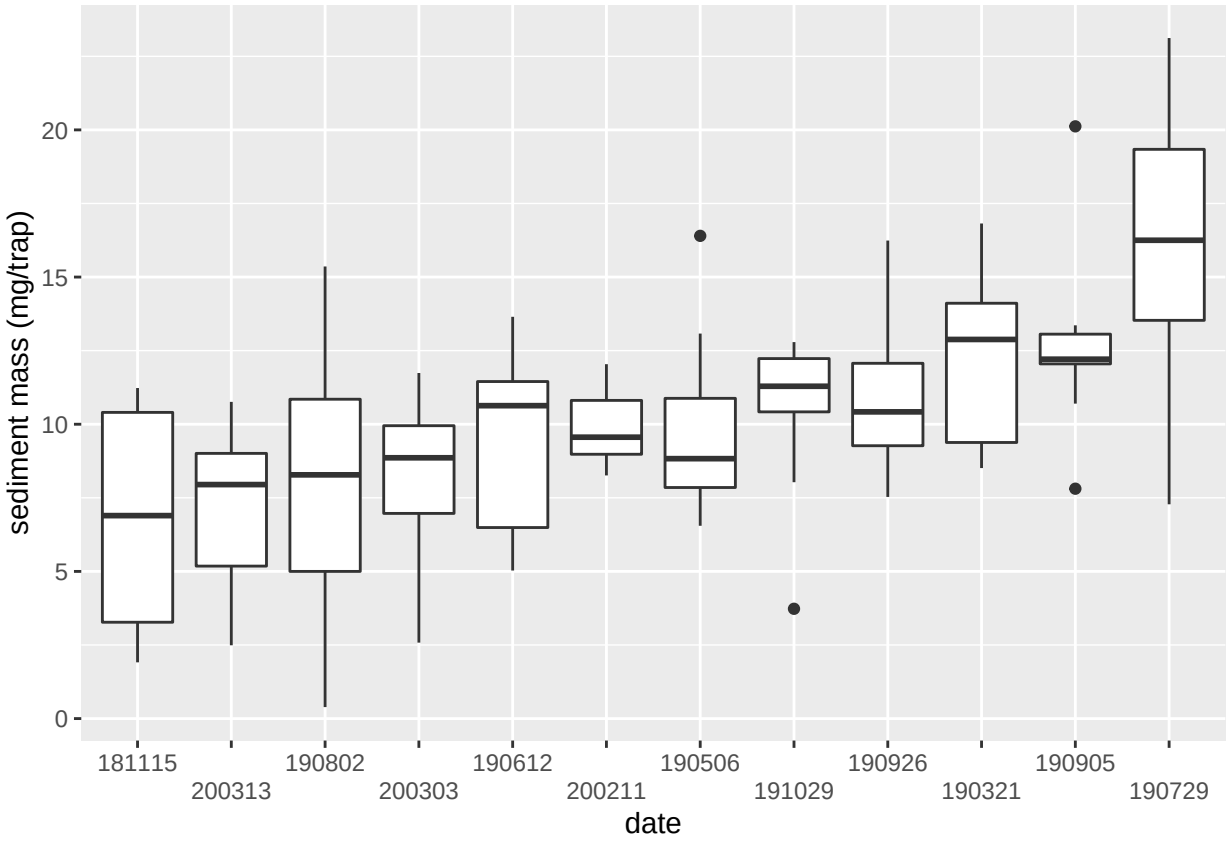
```
trap %>%
  group_by(date) %>%
  mutate(mean = mean(sed)) %>%
  arrange(mean) %>%
  ggplot(aes(x = factor(date, levels = unique(date)), y = sed)) +
  geom_boxplot() +
  labs(x = "date", y = "sediment mass (mg/trap)") +
```

```
  scale_x_discrete(guide = guide_axis(n.dodge = 2))
```



On the whole, the sediment data seems pretty normal and independent. Let's zoom in to just the 12 runs used in the main analysis.

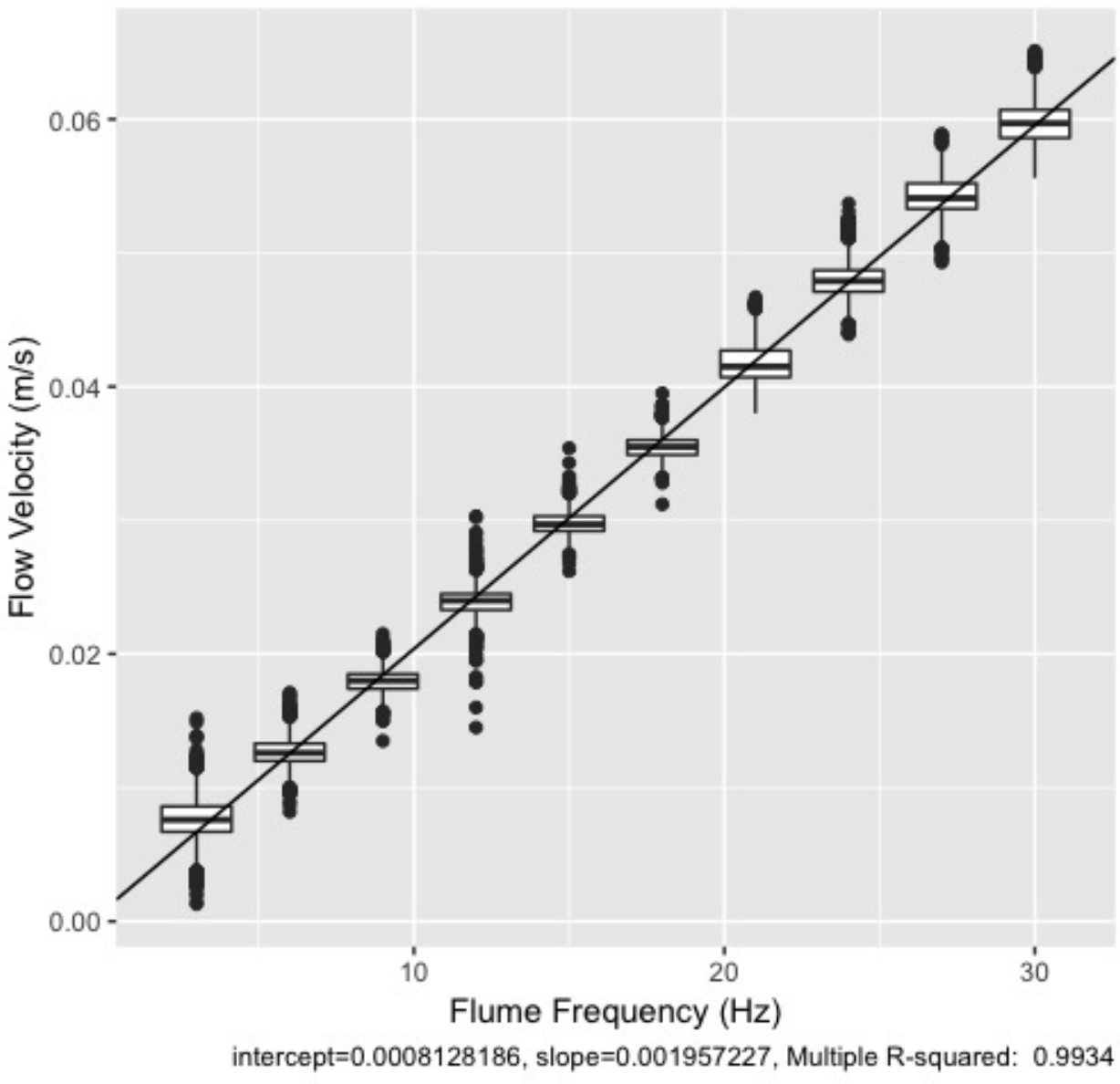
```
trapfinal <- left_join(trap, metadata, by = "date") %>%
  filter(date %in% pumpfinal$date)

trapfinal %>%
  group_by(date) %>%
  mutate(mean = mean(sed)) %>%
  arrange(mean) %>%
  ggplot(aes(x = factor(date, levels = unique(date)), y = sed)) +
  geom_boxplot() +
  labs(x = "date", y = "sediment mass (mg/trap)") +
  scale_x_discrete(guide = guide_axis(n.dodge = 2))
```



A pretty broad range, without any clear outliers to knock out, at least by my sense of it. The whiskers are limited to $1.5 \cdot \text{IQR}$ each (both up and down), so while there are 4 points outside that range for their respective runs, they're definitely all within the range of the dataset as a whole.

Velocity

We estimated Reynolds number from a flow velocity experiment in the open channel conducted in early 2019.



It looks like there are a lot of outliers, but that's because there are simply a ton of Vectrino points and the distribution clearly has long tails. See how tiny the IQR boxes are, which by definition encompass 50% of points.

It should also be noted that the back-of-envelope estimate of $v = f/500$ is extremely close, probably by design.

The data and code for this analysis is MIA at the moment, but should turn up soon.

Flume volume

The test section is modeled as a rectangular prism, making calculation of volume trivial. However, the water in the flume is in an irregular form. In order to correct for the time water spends flowing outside the test section, where collectors are not acting on particles, we must estimate the total volume of water. We did so using a simple integration of the volumetric flow rate at the drain. We ran an experiment previously in 2018 or 2019, but my physics common sense lapsed: We picked the drain hose up off the ground to take measurements, then stuck it back down in the drain for most of the flow time. This led to an underestimate

of about 5-10%. I re-did the analysis. 1st through 4th degree polynomial regression all yield the same volume +/- <0.5%:

```
# flows in ml/s:
flow <- c(131, 132, 132, 131, 134, 129, 130, 129, 130, 129, 130, 127, 127, 127, 126,
          123, 126, 125, 124, 125, 127, 125, 123, 122, 125, 124, 118, 123, 125, 125,
          120, 122, 124, 126, 122, 117, 123, 118, 120, 120, 121, 118, 118, 118, 119,
          118, 120, 117, 117, 114, 115, 115, 116, 112, 112, 110, 111, 108, 105, 105,
          102, 100, 100, 102)

# times in fractional hrs of day:
time <- c(12.13333, 12.21667, 12.28333, 12.41667, 12.5, 12.56667, 12.6, 13.11667,
          13.15, 13.18333, 13.58333, 13.61667, 13.65, 14.25, 14.28333, 14.31667,
          14.35, 14.38333, 14.68333, 14.71667, 14.75, 14.78333, 14.81667, 14.85,
          14.88333, 14.93333, 15.35, 15.38333, 15.41667, 15.45, 15.48333, 15.76667,
          15.8, 15.86667, 15.9, 15.93333, 15.96667, 16.18333, 16.23333, 16.28333,
          16.31667, 16.33333, 16.36667, 16.55, 16.58333, 16.61667, 16.81667, 16.85,
          16.9, 16.93333, 16.98333, 17.03333, 17.06667, 17.2, 17.23333, 17.35,
          17.36667, 17.53333, 17.58333, 17.61667, 17.73333, 17.75, 17.78333, 17.8)

#flow start time:
t_0 <- 11.59166667

#time at which flume was drained to top of test-array holding basin
t_b <- 16.96666667

#time at which basin was drained
t_f <- 17.825

#polynomial model fits:
fits <- lapply(1:4, function(x) {lm(flow ~ poly(time, x, raw = TRUE))})

#finite integral calculator
poly_fin_integr <- function(model,lim1,lim2) {
  sum(coefficients(model)/(1:length(coefficients(model)))*
        lim2^(1:length(coefficients(model))))-
  sum(coefficients(model)/(1:length(coefficients(model)))*
        lim1^(1:length(coefficients(model))))
}

estimates <- lapply(fits, function(x) poly_fin_integr(x, lim1 = t_0, lim2 = t_b))

#convert from mL/s*h to cubic m: 1 mL/s*h = 3600 mL = 3.6 L = 0.0036 cubic m

estimates <- unlist(estimates)*.0036

names(estimates) <- c("linear", "quadratic", "cubic", "4th order")

estimates
```

```
##    linear quadratic     cubic 4th order
##  2.430926  2.427712  2.437273  2.429381
```

I rounded off to 2.43 m^3. The test section has been measured as $1.95 \times .6 \times .4 = 0.468$ m^3. Hence the corrective factor is 5.1923077.

Modelling for eta (and all the other vars of interest)

So at this point, we really only need `pumpfinal`, `trapfinal`, maybe `metadata` for good measure, the frequency-to-velocity regression coefficients, and individual constants about the physical setup such as dowel diameter, flume volume, starting sediment mass, dynamic viscosity etc.

Let's clean up for the sake of sanity (if you're reading the .pdf, this is unimportant):

```
rm(fits, pump, pumpdata, tidypump, tidytrap, trap, trapdata, x, estimates, flow, pumpdate, pumpfiles, t
```

I'll also mention here that our model is based on the general equation:

$$k_{tot} = k_s + k_c + k_{bg}$$

where the k's represent total decay rate and portions of it due to settling, collection, and (background) settling in the rest of the flume outside the test section, respectively.

k_tot (total time-decay rate)

We estimate k_tot from our pump data by linear regression of `log(mvc)`:

```
library(lme4)
```

```
## Loading required package: Matrix
```

```
##
## Attaching package: 'Matrix'
```

```
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
```

```
fits <- lmList(data = pumpfinal, log(mvc)~t | date)

summary(fits)
```

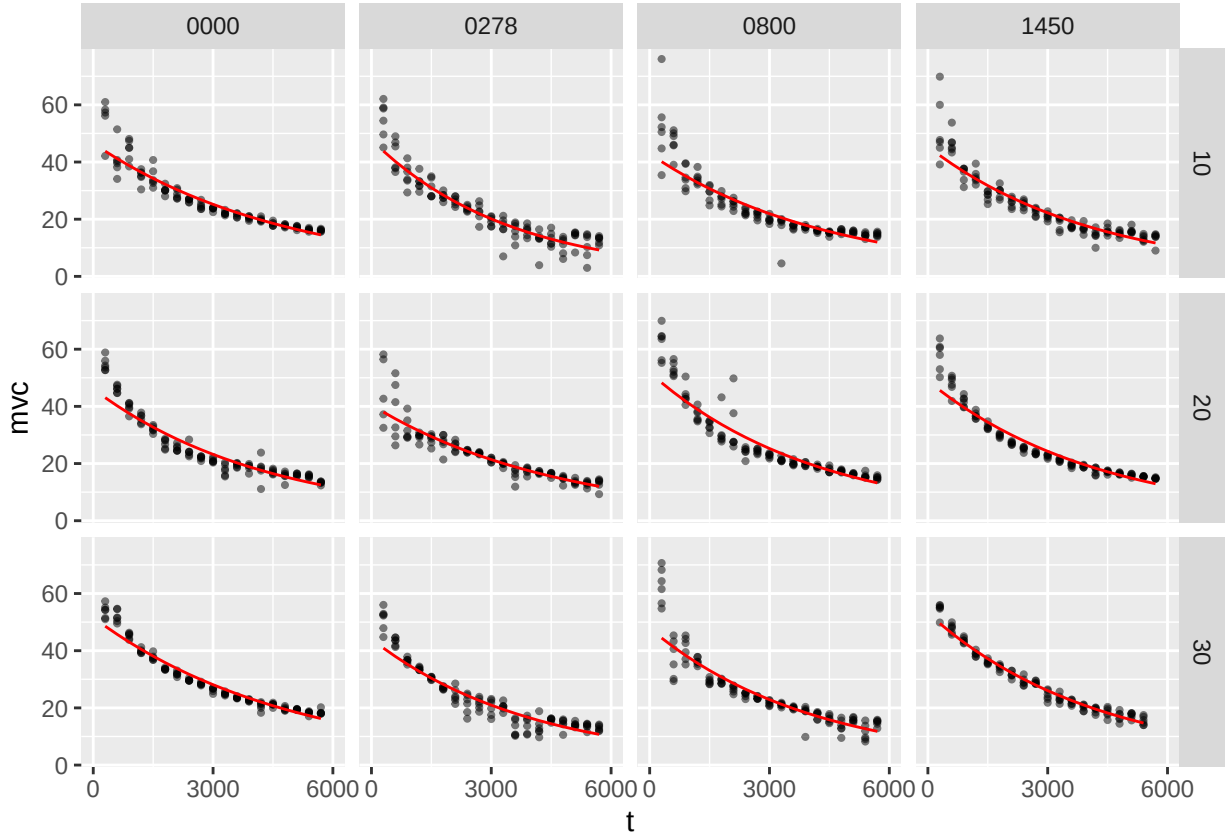
```
## Call:
##   Model: log(mvc) ~ t | NULL
##    Data: pumpfinal
##
## Coefficients:
##    (Intercept)
##        Estimate Std. Error  t value Pr(>|t|)
## 181115 3.973846 0.02998373 132.5334        0
## 190321 3.943840 0.02911822 135.4423        0
## 190506 3.785685 0.02911822 130.0109        0
## 190612 3.867590 0.02911822 132.8237        0
## 190729 3.840790 0.02960512 129.7340        0
## 190802 3.830722 0.02911822 131.5576        0
## 190905 3.866608 0.02911822 132.7900        0
## 190926 3.701444 0.02960157 125.0421        0
## 191029 3.758973 0.02911822 129.0935        0
## 200211 3.948348 0.02911822 135.5972        0
## 200303 3.815678 0.02911822 131.0409        0
## 200313 3.889987 0.02911822 133.5929        0
##    t
##             Estimate   Std. Error   t value      Pr(>|t|)
## 181115 -0.0002381660 9.233416e-06 -25.79392 2.328639e-119
## 190321 -0.0002025636 8.512789e-06 -23.79521 1.316401e-104
```

```
## 190506 -0.0002477993 8.512789e-06 -29.10906 1.244289e-144
## 190612 -0.0002455980 8.512789e-06 -28.85048 1.242828e-142
## 190729 -0.0002045307 8.621156e-06 -23.72428 4.310903e-104
## 190802 -0.0002286986 8.524112e-06 -26.82962 3.672883e-127
## 190905 -0.0002885331 8.512789e-06 -33.89408 3.614730e-182
## 190926 -0.0002130646 8.616350e-06 -24.72795 1.946542e-111
## 191029 -0.0002236461 8.512789e-06 -26.27177 6.016420e-123
## 200211 -0.0002397320 8.512789e-06 -28.16139 2.525016e-137
## 200303 -0.0002382292 8.512789e-06 -27.98487 5.713696e-136
## 200313 -0.0002333927 8.512789e-06 -27.41671 1.263618e-131
##
## Residual standard error: 0.1493503 on 1334 degrees of freedom
```

The `t` coeffient estimates are the `k_tot` values in units s^{-1}, no transformation needed aside from a sign change ($\log(\bar{\phi}) = \log(\phi_0 e^{-kt}) = log(\phi_0) - kt$).

All the t values are high, though the coefficient t-values are relative to 0, meaning they're really just expressing that we're very certain we did indeed add sediment (phew!). But let's see what the fits look like with the data:

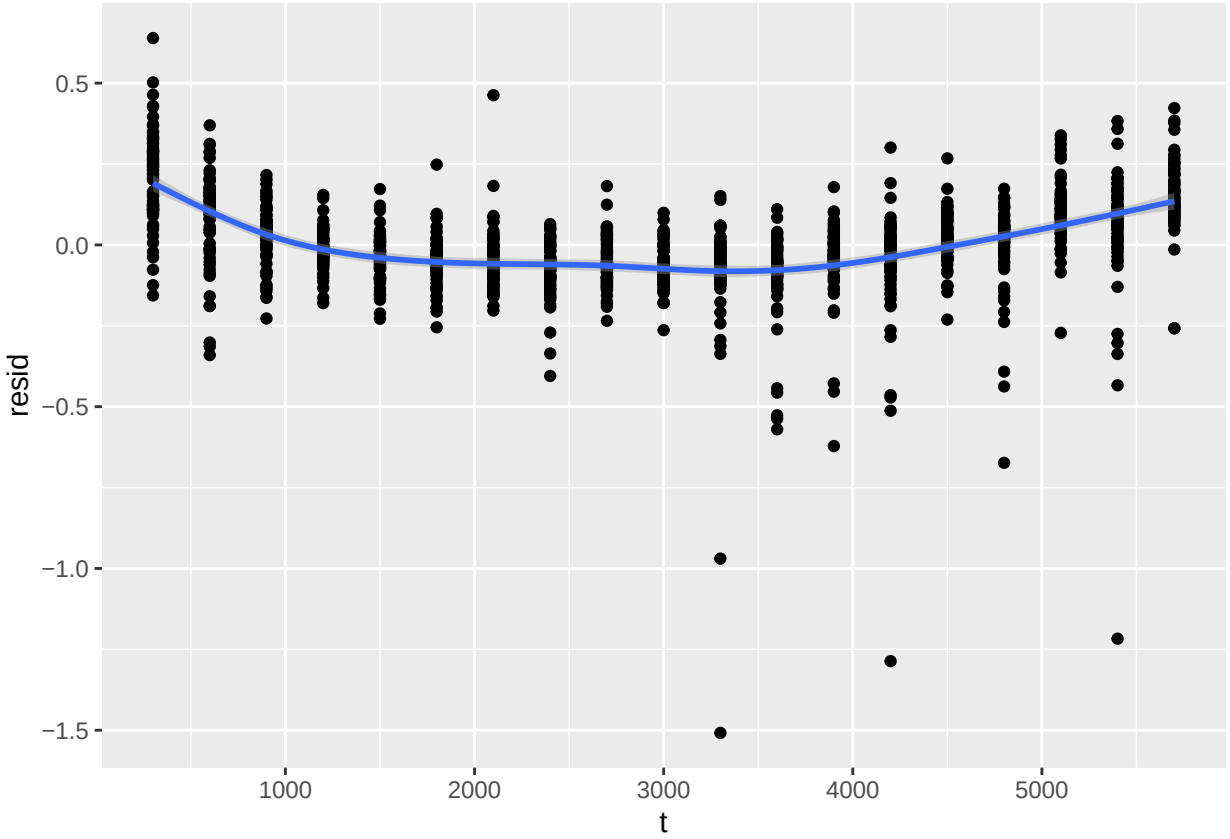
```
cbind(pumpfinal, pred = predict(fits)) %>%
  ggplot() +
  geom_point(aes(x = t, y = mvc), size = .75, alpha = .5) +
  geom_line(aes(x = t, y = exp(pred)), color = "red") +
  facet_grid(pump_freq~dowel_density) +
  scale_x_continuous(breaks = c(0,3000,6000), limits = c(0,6000))
```



As I noticed back in early 2019, these curves seem to hint at a U-shaped pattern in the residuals.

```
cbind(pumpfinal, resid = residuals(fits)) %>%
  ggplot(aes(x = t, y = resid)) +
  geom_point() +
  geom_smooth()
```

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



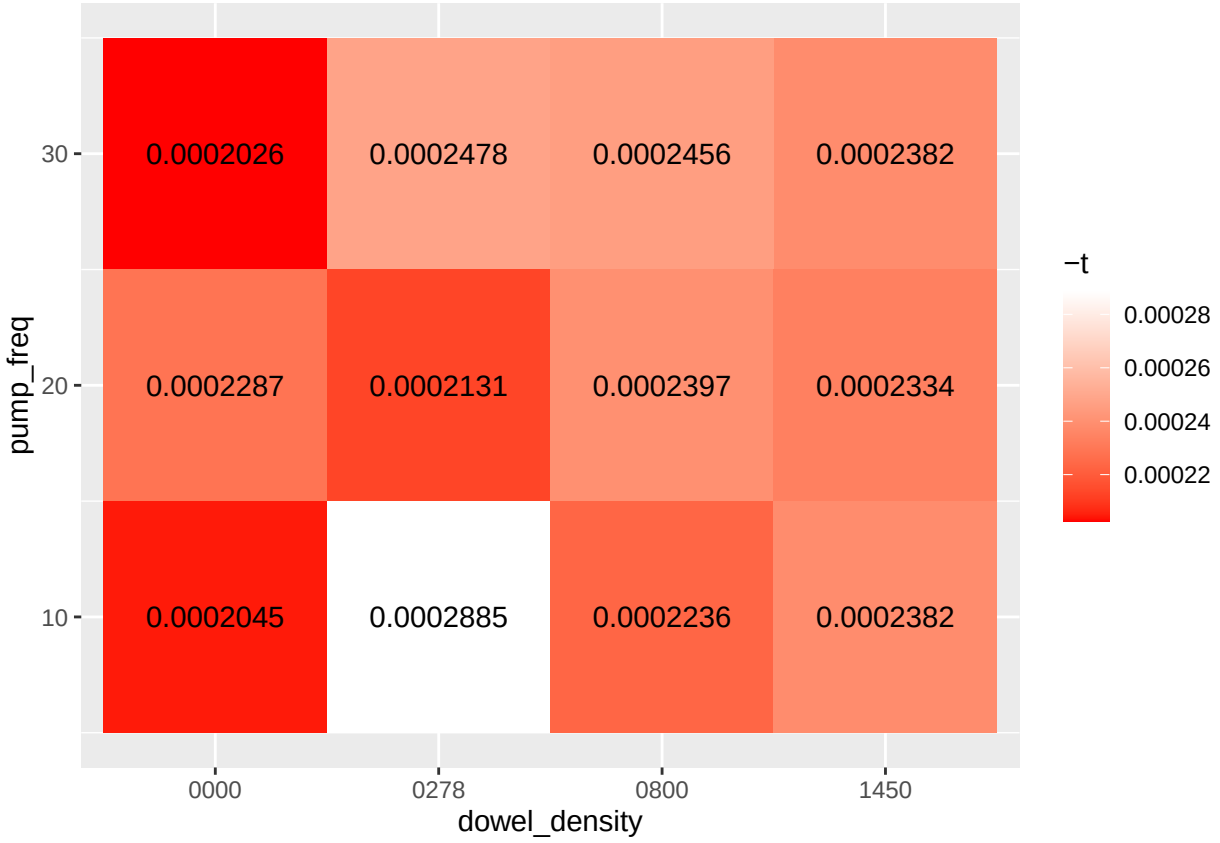
I think this is simply an artifact of nonlinear processes not accounted for in the exponential model. As an example of what I mean, let's imagine there were a hidden variable called "stickiness" differentiating walnut shell particles from one another. If by way of accellerated settling due to floculation, or accellerated collection, those particles were disproportionately removed from suspension early in the timespan of the experiment, the average physical properties of the suspended sediment would change over its course. It also seems likely to me that as sediment adheres to collectors, their efficiency decreases on account of their having less open surface area.

Whatever the explanation is, we (and other researchers using the exponential model) assume it is uniform across our independent variables. I don't know how safe that assumption is, but that'll have to wait for a future paper. It might be testable from our data but `n = 12` seems a bit of a small sample. Then again, we've probably done more like `n = 100` runs over the entire history of the project.

Aside from that U-curve, the fit seems reasonably good. Though there are high **absolute** measurements at the beginning, on the **log-scale** they are far less influential than the lower values (`mvc < 10`) at the end. Removing the lower outliers towards the end (`t > 3000`) might be worth trying later if our results don't satisfy.

So then, k_s as a function of our independent vars:

```
fitdata <- coefficients(fits) %>%
cbind(date = as.numeric(row.names(coefficients(fits)))) %>%
  left_join(metadata, by = "date")

fitdata %>%
  ggplot(aes(x = dowel_density, y = pump_freq, fill = -t, label = round(-t,7))) +
    geom_tile() +
    geom_text() +
  scale_fill_gradient(low = "red", high = "white")
```



Mixed feelings. On the one hand it's reassuring that aside from the switcharoo at 20 Hz, all runs with dowels had greater k_t than runs without. However, it seems like there's some serious variance. **What confounders could we be missing???**

THE PROBLEM IN A NUTSHELL

Having a greater decay rate for the control run than the dowel run, by an amount greater than the difference in k_s between the two runs, implies a negative k_c. This seems unrealistic, or at least very out of step with our operating theory. For the rest of this analysis, I'll do what I slipped into my past code and then forgot about: leave out that control run and estimate its values as the mean of the other controls (it does lie at the mean of their independent vars. so weighting the average seems unnecessary). Not ideal! Maybe we should give this set of parameters another shot? But I don't feel great about cherrypicking experimental results that way either.

```
fitdata[fitdata$date==190802,"t"] <- (fitdata[fitdata$date==190729,"t"] +
                                     fitdata[fitdata$date==190321,"t"])/2
```

```
fitdata
```

```
##    (Intercept)             t   date dowel_density pump_freq growth_days
## 1     3.973846 -0.0002381660 181115          1450        30           0
## 2     3.943840 -0.0002025636 190321          0000        30           0
## 3     3.785685 -0.0002477993 190506          0278        30           0
## 4     3.867590 -0.0002455980 190612          0800        30           0
## 5     3.840790 -0.0002045307 190729          0000        10           0
## 6     3.830722 -0.0002035472 190802          0000        20           0
## 7     3.866608 -0.0002885331 190905          0278        10           0
## 8     3.701444 -0.0002130646 190926          0278        20           0
## 9     3.758973 -0.0002236461 191029          0800        10           0
## 10    3.948348 -0.0002397320 200211          0800        20           0
## 11    3.815678 -0.0002382292 200303          1450        10           0
## 12    3.889987 -0.0002333927 200313          1450        20           0
```

k_s (time-decay due to settling)

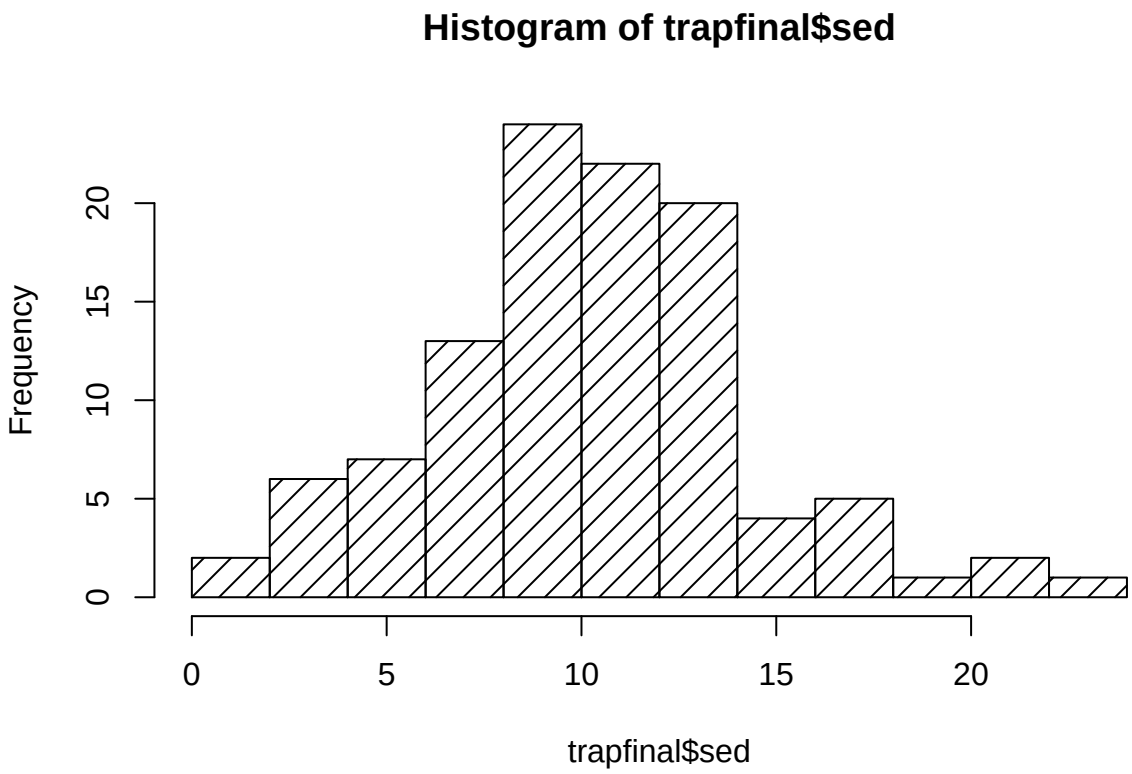
$$m_s = \frac{k_s}{k_{tot}}(1 - e^{-kT})m_0$$

Or, expressed for k_s,

$$k_s = \frac{m_s}{m_0}(1 - e^{-kT})k_{tot}$$

where m_s is mass settled in the test section (estimated from sediment traps), m_0 is starting sediment mass (200 g for this study), and T is the total exposure time of the sediment traps (100 min, i.e., 6000 s, for this study).

```
hist(trapfinal$sed, breaks = 10, density = 10)
```

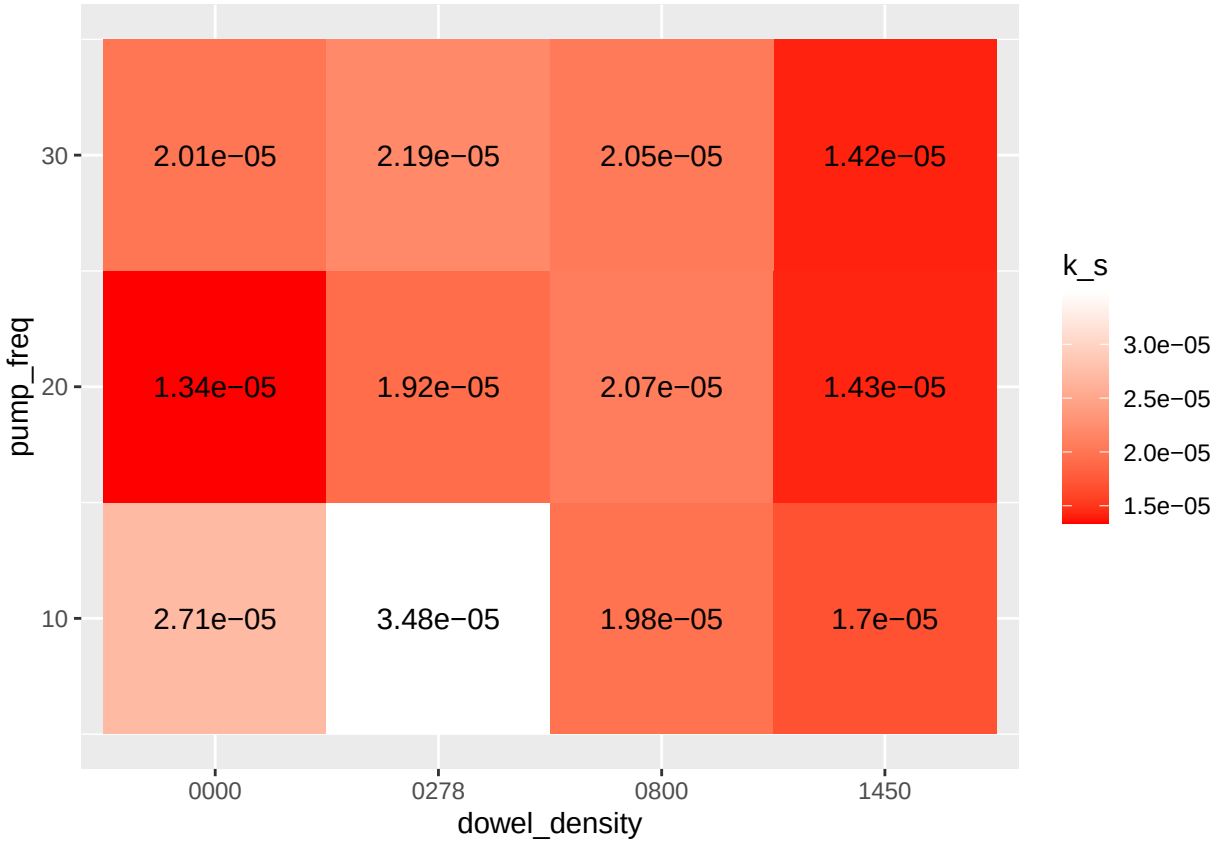


There aren't really any clear outliers, as we established in the boxplots above. While splines might be our final tool, for now an average will do.

```
final <- trapfinal %>%
  group_by(date) %>%
  summarise(m_trap = mean(sed)/1000) %>% #average sediment in trap converted from mg to g
  mutate(m_s = m_trap*1.95*.6/(pi*.0127^2)) %>% #times test section area divided by area of 1 trap
  left_join(fitdata, by = "date") %>%
  mutate(k_s = m_s/200*(1-exp(t*6000))*(-t))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
final %>%
  ggplot(aes(x = dowel_density, y = pump_freq, fill = k_s, label = round(k_s,7))) +
    geom_tile() +
    geom_text() +
  scale_fill_gradient(low = "red", high = "white")
```



If we're not using the k_{tot} from the 20 Hz control, I think we should probably do the same with k_s, not least because it gives us linear k_{bg} values, though this all still feels a little iffy.

```
final[final$date==190802,"k_s"] <- (final[final$date==190729,"k_s"] +
                                    final[final$date==190321,"k_s"])/2
```

Gladly, settling across the dowel treatments does seem to decrease, with a less clear pattern across flow velocity.

k_bg (time-decay due to settling in the rest of the flume)

In control runs, $k_c = 0$, so we can estimate the decay rate in the rest of the flume as $k_{bg} = k_{tot} - k_s$

```
bgvals <- final %>%
  filter(dowel_density=="0000") %>%
  transmute(pump_freq, k_bg = -t - k_s) %>%
  arrange(pump_freq)

bgvals
```

```
## # A tibble: 3 x 2
##   pump_freq     k_bg
##       <dbl>    <dbl>
## 1        10 0.000177
## 2        20 0.000180
## 3        30 0.000182
```

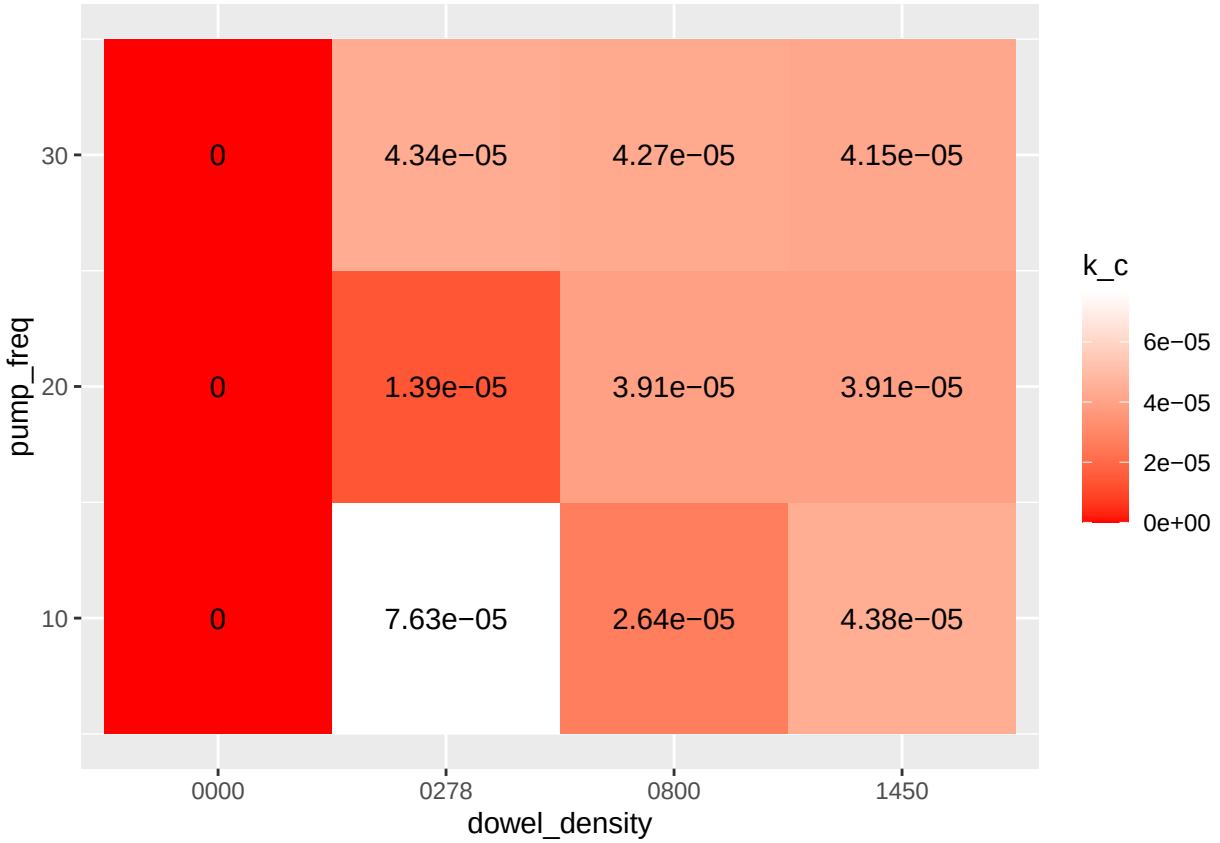
To add it to the table:

```
final <- left_join(final, bgvals, by = "pump_freq")
```

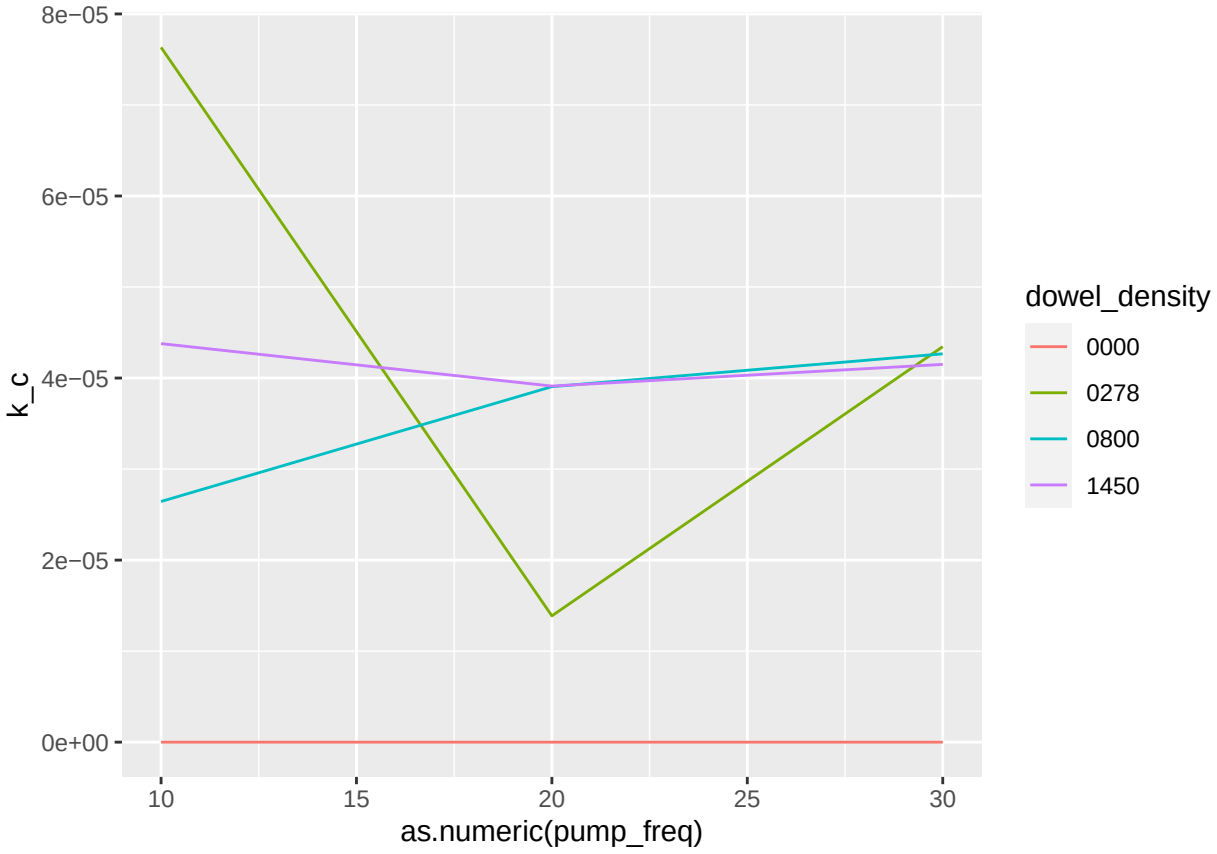
k_c (time-decay due to collection)

So, now that we have our values in the table, k_c is just a subtraction away:

```
final <- final %>%
  mutate(k_c = -t - k_s - k_bg)

final %>%
  ggplot(aes(x = dowel_density, y = pump_freq, fill = k_c, label = round(k_c,7))) +
    geom_tile() +
    geom_text() +
  scale_fill_gradient(low = "red", high = "white")
```



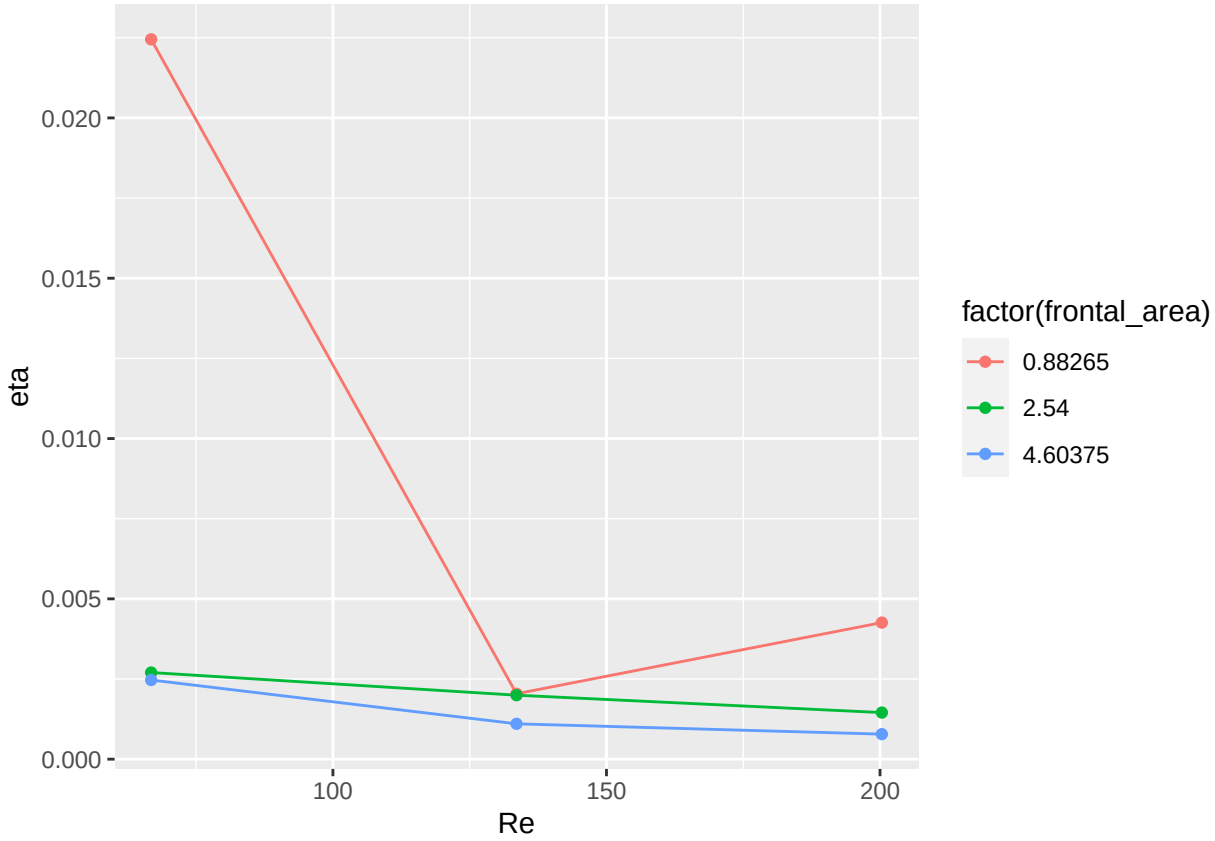

```
final %>%
  ggplot(aes(x = as.numeric(pump_freq), y = k_c, color = dowel_density)) +
  geom_line()
```



Unlike the calculation with the measured values at the 20 Hz control, all k_c are in the green, but the pattern is still quite odd.

eta

```
#temp in flume measured at 22.2C with a calibrated thermometer
#plugged into https://www.engineeringtoolbox.com/water-dynamic-kinematic-viscosity-d_596.html
visc = 9.509e-7 #kinematic viscosity, m2/s

d = .003175 #dowel diameter

final <- final %>%
  transmute(frontal_area = as.numeric(dowel_density)*.003175,
            u = as.numeric(pump_freq)/500, #velocity
            Re = u*.003175/visc, #Reynolds #
            k_t = -t,
            k_c, k_s, k_bg) %>%
  mutate(eta = k_c/u/frontal_area) %>%
  mutate(eta = eta * 2.43/(1.95*.4*.6)) # don't forget to correct for time out of test section!

final %>%
  filter(frontal_area != 0) %>%
  ggplot(aes(x = Re, y = eta, color = factor(frontal_area))) +
  geom_line() +
  geom_point()
```



Possible additions

- Turbulence data
  - The calculations are done
  - But it might be good to include here for the sake of comprehensiveness
- i did find a way to propogate error through these calculations quite easily
  - I think including error bars on our main graph (eta over Rey) might be a good idea?
  - Is it irresponsible to do so with `n = 1` per treatment?