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).

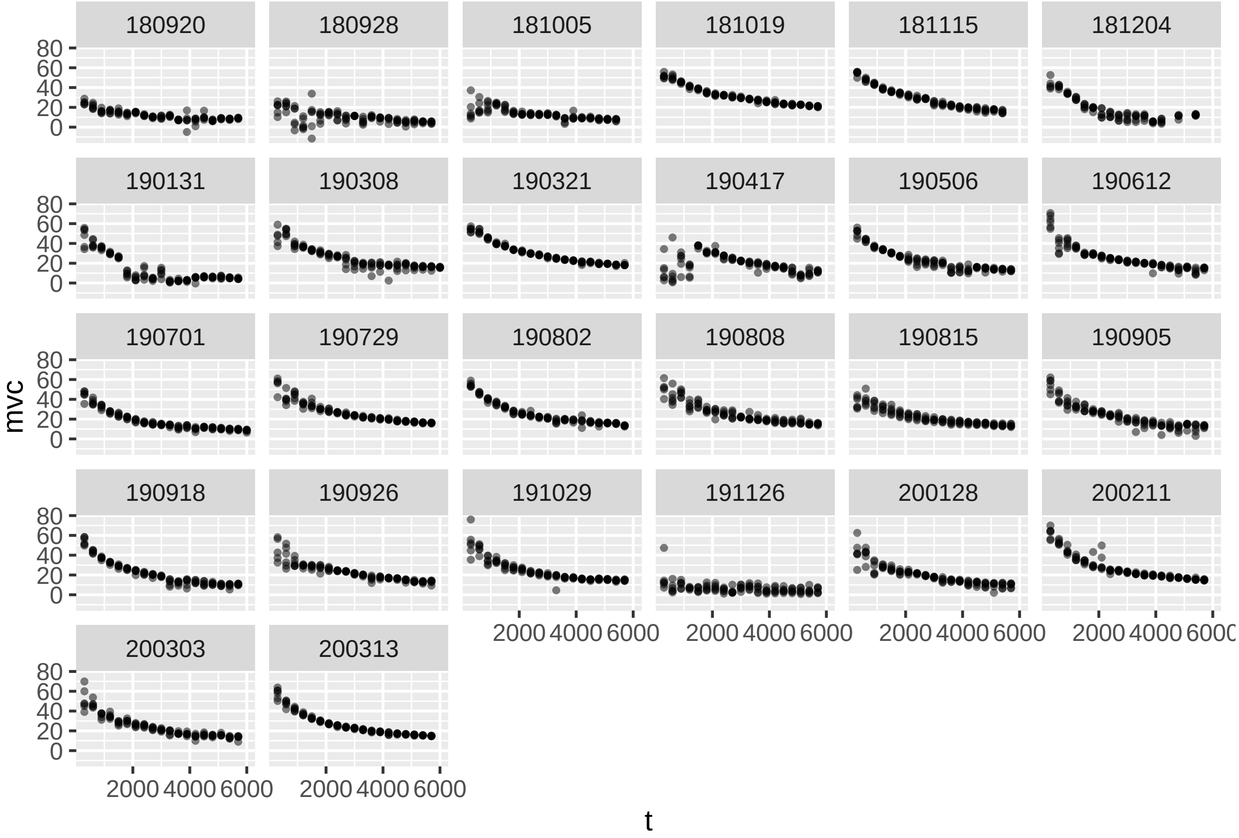
```
pumpdata <- lapply(pumpfiles, function(x) read_csv(paste0("../data/peristaltic pumps/",x)))

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)
```

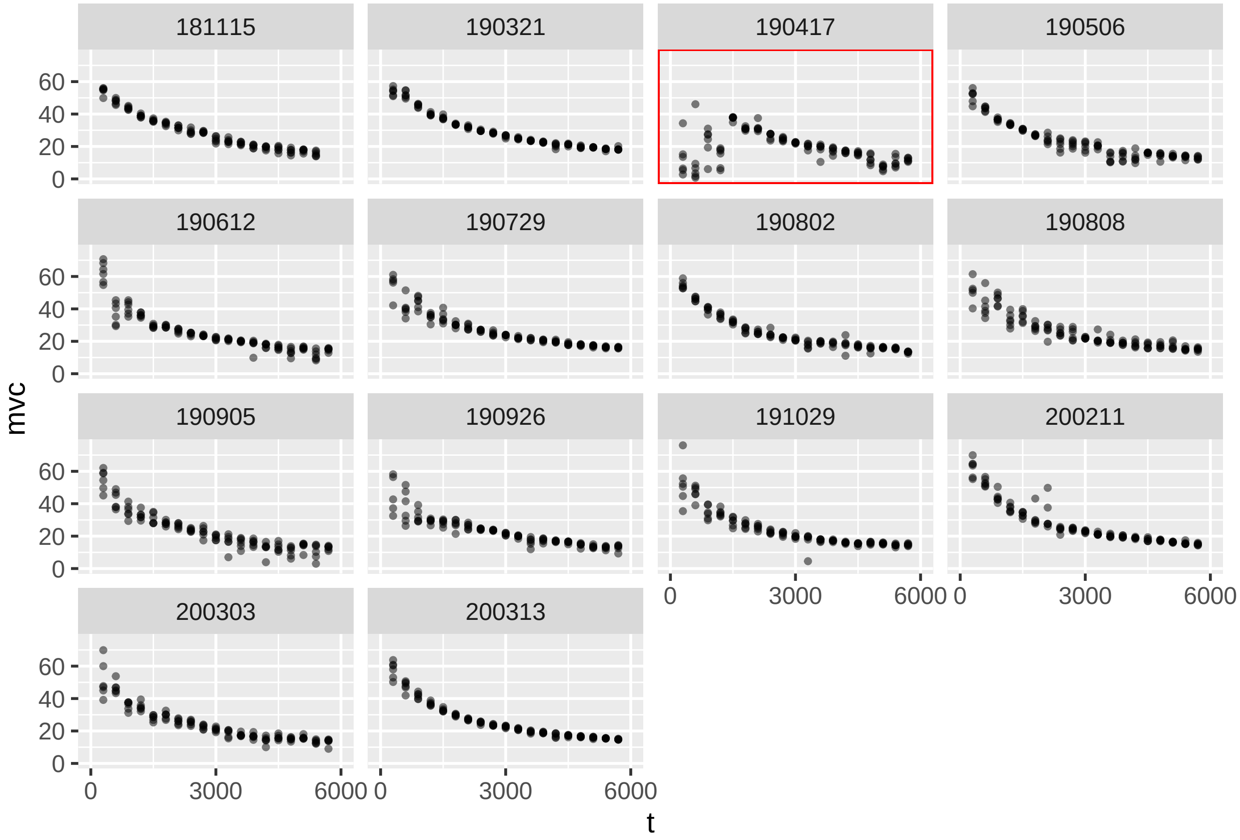
```
## [1] 20
```

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.

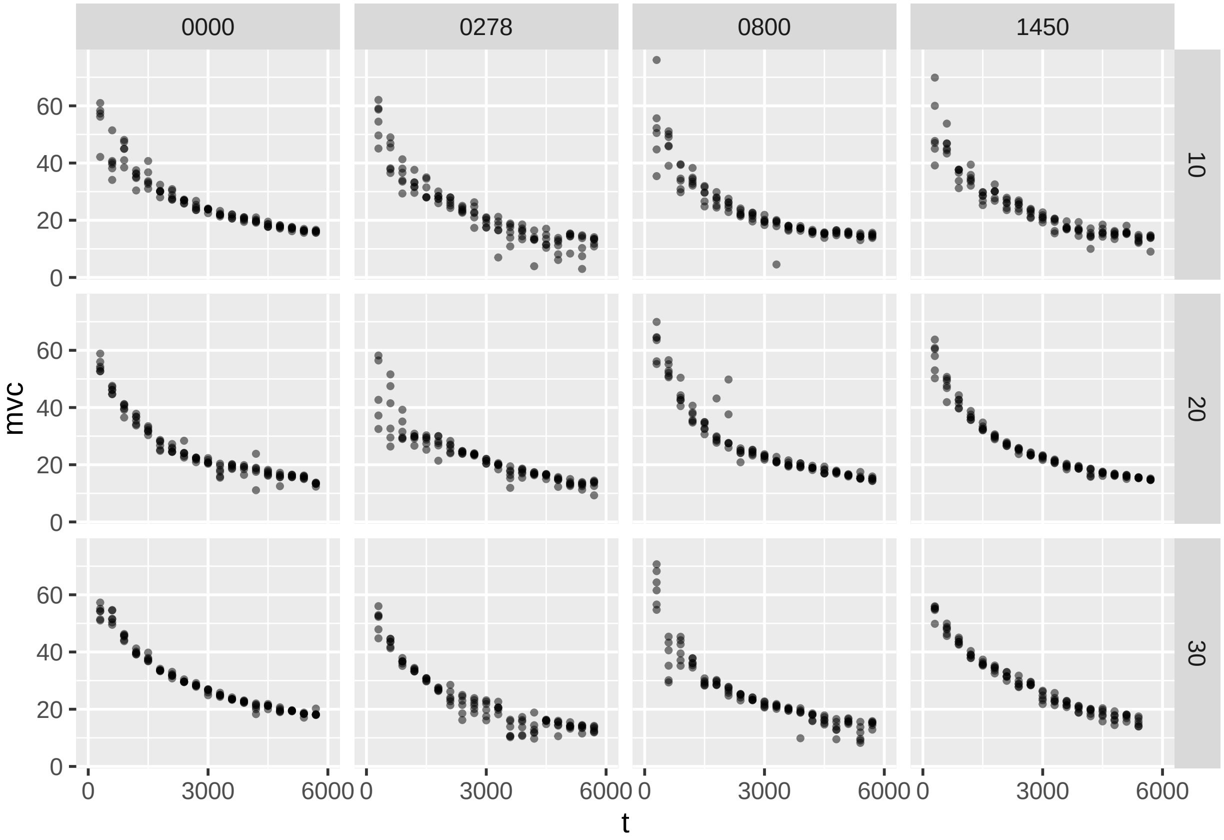
```
pump %>%
  filter(growth_days==0) %>%
  ggplot(aes(x = t, y = mvc)) +
  geom_point(alpha = .5, size = .75) +
  geom_rect(data = filter(pump, date == 190417),
                        fill = NA, colour = "red", xmin = -Inf,xmax = Inf,
            ymin = -Inf,ymax = Inf) +
  facet_wrap(~date) +
  scale_x_continuous(breaks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

```r
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
```

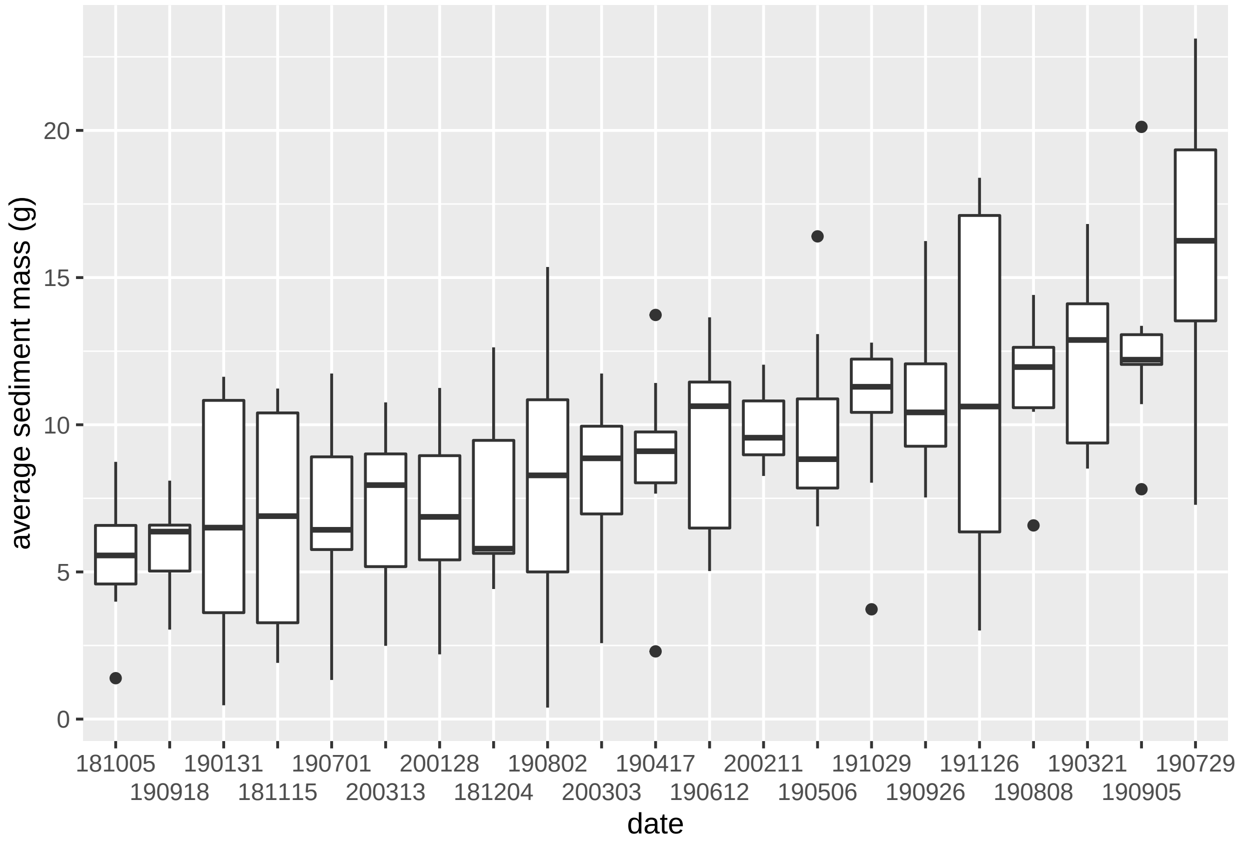
So, across 21 runs, we had 186 samples. We were set up to collect 9 per run, but traps broke, filters slipped and spilled sediment, et cetera. 186/21 = 8.8571429, so our success rate was actually pretty good, although some of those measurements were probably erroneous despite being measured. Let's see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

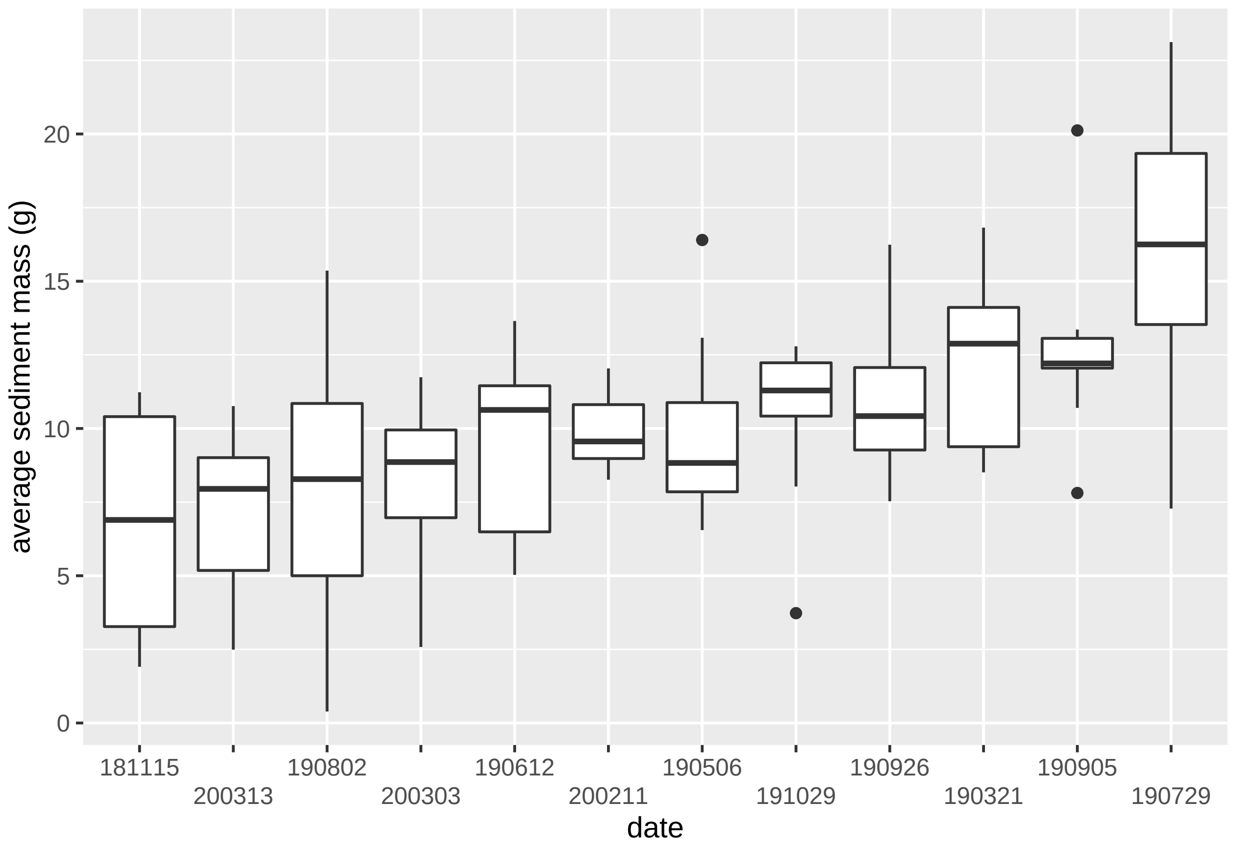
So, downstream traps tended to have less settling. This challenges our assumption of uniformity throughout the test section and might justify the use of a spline, as Justin and Candace had suggested.

```
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 = "average sediment mass (g)") +
  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.

```
left_join(trap, metadata, by = "date") %>%
  filter(date %in% pumpfinal$date) %>%
  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 = "average sediment mass (g)") +
  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 $1.5 \cdot \mathrm{IQR}$ 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.