

analysis

```
library(MASS)
library(broom)
library(tidyverse)
options(pillar.sigfig=4)
```

Data import

```
csv2tibble <- function(fn, show_plot = FALSE) {

  # Determine the condition of this trial
  if (grepl("-nothing", fn)) {
    condition <- "control"
  } else if (grepl("-plant", fn)) {
    condition <- "liveplant"
  } else if (grepl("-deadplant", fn)) {
    condition <- "deadplant"
  } # else if (grepl("-soil", fn)) {
  #   condition <- "soil"
  } else {
    stop("Cannot infer condition for file ", fn)
  }

  # Determine the enclosure/sensor of this trial
  if (grepl("1a", fn)) {
    sensor <- "A"
  } else if (grepl("2b", fn)) {
    sensor <- "B"
  } else if (grepl("3d", fn)) {
    sensor <- "C"
  } else {
    stop("Cannot infer enclosure/sensor for file ", fn)
  }

  tib <- fn %>%
    { suppressMessages(
      read_csv(., col_names = c("datetime", "type", "value"))) } %>%
    # obs of type "comment" are not used
    filter(type %in% c("pm2.5", "pm10")) %>%
    mutate(value=as.numeric(value)) %>%
    mutate(
      condition=condition, sensor=sensor, fn=fn,
      has_area=condition %in% c("deadplant", "liveplant"),
      has_bio=condition == "liveplant")

  # Experiment starts when PM2.5 first drops below 900.
}
```

```

first_pm2.5_999.99 <- tib %>%
  filter(type == "pm2.5" & value == 999.9) %>%
  .$datetime %>%
  last()
first_pm2.5_sub900 <- tib %>%
  filter(datetime > first_pm2.5_999.99) %>%
  filter(type == "pm2.5" & value <= 900) %>%
  .$datetime %>%
  first()

# Experiment ends when PM2.5/PM10 first drops below 10.
first_sub10 <- tib %>%
  filter(datetime > first_pm2.5_sub900 & value < 10) %>%
  .$datetime %>%
  first()

if (show_plot) {
  g <- ggplot(tib, aes(x=datetime, y=value, colour=type)) +
    geom_point() +
    geom_hline(yintercept = c(900, 999.9)) +
    geom_vline(xintercept = c(
      first_pm2.5_999.99, first_pm2.5_sub900, first_sub10))
  print(g)
}

tib <- filter(tib, between(datetime, first_pm2.5_sub900, first_sub10)) %>%
  # use minutes instead of seconds, otherwise will get Inf on exp later
  mutate(elapsed=as.numeric(datetime-first_pm2.5_sub900)/60)

tib
}

# Make a list of two tibbles, one for PM2.5, the other for PM10.
data <- list.files("05-analysis-proper/", "*.csv") %>%
  Filter(function(x) !grepl("-soil", x), .) %>%
  paste0("05-analysis-proper/", .) %>%
  lapply(csv2tibble) %>%
  do.call(bind_rows, .) %>%
  mutate(
    condition=
      factor(condition, levels=c("control", "deadplant", "liveplant")),
    type=factor(type, levels=c("pm2.5", "pm10"))) %>%
  {list(
    pm2.5 = filter(., type=="pm2.5"),
    pm10 = filter(., type=="pm10")
  )}

```

Note that we omit the soil controls because 5 of 15 of those trials suffer from systematic errors, likely due to a leaky seal on the enclosure.

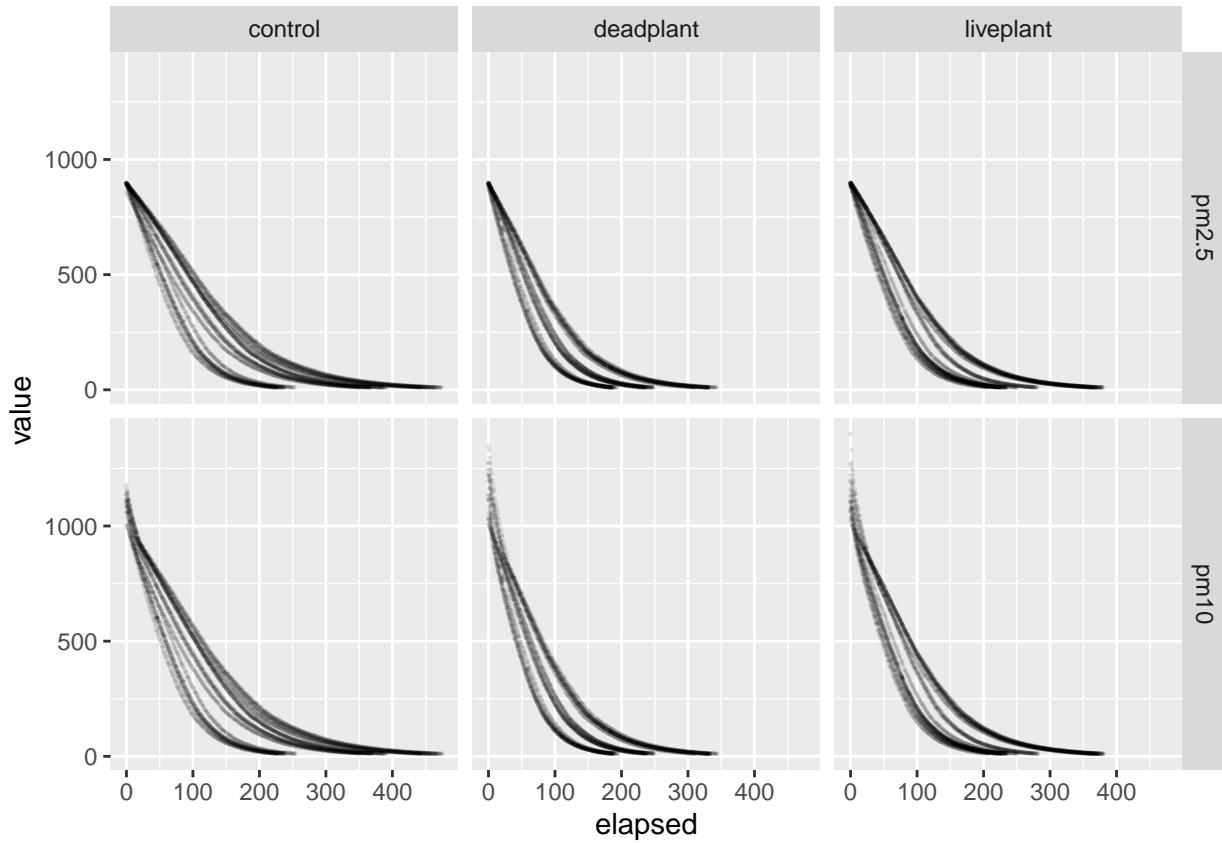
Sanity check:

```

do.call(bind_rows, data) %>%
  filter(value > 10) %>%
  ggplot(aes(x=elapsed, y=value)) +

```

```
geom_point(size=.1, alpha=.1) +
facet_grid(type~condition)
```



Model selection

Stepwise BIC optimisation reports a complex model to be most suitable for both PM2.5 and PM10.

```
model_full <- (
  value ~ elapsed*sensor + elapsed*has_area + elapsed*has_bio)
glm_bare <- glm(value ~ 1, family=Gamma("log"), data=data$pm2.5)
glm_full <- glm(model_full, family=Gamma("log"), data=data$pm2.5)
stepAIC(
  glm_bare, scope=list(lower=glm_bare, upper=glm_full),
  direction="both", trace = FALSE,
  k=log(nrow(data$pm2.5)))

##
## Call: glm(formula = value ~ elapsed + sensor + has_area + has_bio +
##           elapsed:sensor + elapsed:has_area + elapsed:has_bio, family = Gamma("log"),
##           data = data$pm2.5)
##
## Coefficients:
##             (Intercept)          elapsed          sensorB
##             7.117660        -0.012917        0.059531
##             sensorC          has_areaTRUE      has_bioTRUE
##             0.115783        -0.094929        0.050745
```

```

##      elapsed:sensorB      elapsed:sensorC  elapsed:has_areaTRUE
##              -0.006360          0.002730        -0.004514
##  elapsed:has_bioTRUE
##          0.001663
##
## Degrees of Freedom: 12916 Total (i.e. Null); 12907 Residual
## Null Deviance: 19600
## Residual Deviance: 555.6 AIC: 120000
glm_bare <- glm(value ~ 1, family=Gamma("log"), data=data$pm10)
glm_full <- glm(model_full, family=Gamma("log"), data=data$pm10)
stepAIC(
  glm_bare, scope=list(lower=glm_bare, upper=glm_full),
  direction="both", trace = FALSE,
  k=log(nrow(data$pm10)))

##
## Call: glm(formula = value ~ elapsed + sensor + has_area + has_bio +
##           elapsed:sensor + elapsed:has_area + elapsed:has_bio, family = Gamma("log"),
##           data = data$pm10)
##
## Coefficients:
##             (Intercept)      elapsed      sensorB
##             7.213408     -0.012854     0.064428
##             sensorC      has_areaTRUE  has_bioTRUE
##             0.141045     -0.083706     0.045912
##      elapsed:sensorB      elapsed:sensorC  elapsed:has_areaTRUE
##      -0.006393       0.002594      -0.004585
##  elapsed:has_bioTRUE
##          0.001686
##
## Degrees of Freedom: 12919 Total (i.e. Null); 12910 Residual
## Null Deviance: 19880
## Residual Deviance: 513.9 AIC: 121800

```

However, we will opt for a simpler model instead, which has comparable BIC:

```

model_simpler <- (
  value ~ elapsed*sensor + elapsed:has_area + elapsed:has_bio)
tibble(
  pm2.5_simpler = BIC(glm(model_simpler, family=Gamma("log"), data=data$pm2.5)),
  pm2.5_full = BIC(glm(model_full, family=Gamma("log"), data=data$pm2.5)),
  pm10_simpler = BIC(glm(model_simpler, family=Gamma("log"), data=data$pm10)),
  pm10_full = BIC(glm(model_full, family=Gamma("log"), data=data$pm10))
)

## # A tibble: 1 x 4
##   pm2.5_simpler pm2.5_full pm10_simpler pm10_full
##   <dbl>        <dbl>        <dbl>        <dbl>
## 1 120148.      120042.      121924.      121837.

```

Regression

For PM2.5

```
fits <- list()
fits$pm2.5 <- data$pm2.5 %>%
  glm(model_simpler, family=Gamma(link="log"), data=.)
summary(fits$pm2.5)

##
## Call:
## glm(formula = model_simpler, family = Gamma(link = "log"), data = .)
##
## Deviance Residuals:
##       Min      1Q      Median      3Q      Max
## -0.93346 -0.14335   0.00321   0.12225   0.48891
##
## Coefficients:
##                               Estimate Std. Error t value Pr(>|t|)
## (Intercept)                7.077e+00  5.820e-03 1215.896 < 2e-16 ***
## elapsed                  -1.276e-02  3.211e-05 -397.467 < 2e-16 ***
## sensorB                   5.400e-02  9.023e-03   5.984 2.23e-09 ***
## sensorC                   1.127e-01  7.823e-03   14.411 < 2e-16 ***
## elapsed:sensorB           -6.306e-03  6.374e-05  -98.930 < 2e-16 ***
## elapsed:sensorC           2.750e-03  3.990e-05   68.925 < 2e-16 ***
## elapsed:has_areaTRUE      -4.961e-03  2.515e-05 -197.256 < 2e-16 ***
## elapsed:has_bioTRUE       1.932e-03  2.704e-05   71.450 < 2e-16 ***
##
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for Gamma family taken to be 0.0402728)
##
## Null deviance: 19600.52 on 12916 degrees of freedom
## Residual deviance:  560.99 on 12909 degrees of freedom
## AIC: 120081
##
## Number of Fisher Scoring iterations: 6
```

For PM10

```
fits$pm10 <- data$pm10 %>%
  glm(model_simpler, family=Gamma(link="log"), data=.)
summary(fits$pm10)

##
## Call:
## glm(formula = model_simpler, family = Gamma(link = "log"), data = .)
##
## Deviance Residuals:
##       Min      1Q      Median      3Q      Max
## -0.8482 -0.1408   0.0002   0.1171   0.4756
##
## Coefficients:
##                               Estimate Std. Error t value Pr(>|t|)
```

```

## (Intercept)      7.178e+00  5.605e-03 1280.601 < 2e-16 ***
## elapsed        -1.272e-02  3.094e-05 -411.155 < 2e-16 ***
## sensorB         5.947e-02  8.685e-03   6.847 7.87e-12 ***
## sensorC         1.381e-01  7.533e-03   18.327 < 2e-16 ***
## elapsed:sensorB -6.345e-03  6.137e-05 -103.396 < 2e-16 ***
## elapsed:sensorC  2.612e-03  3.843e-05   67.981 < 2e-16 ***
## elapsed:has_areaTRUE -4.978e-03  2.422e-05 -205.544 < 2e-16 ***
## elapsed:has_bioTRUE   1.928e-03  2.604e-05   74.057 < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for Gamma family taken to be 0.03733007)
##
## Null deviance: 19878.40 on 12919 degrees of freedom
## Residual deviance: 518.09 on 12912 degrees of freedom
## AIC: 121857
##
## Number of Fisher Scoring iterations: 6

```

Confidence interval for estimates

Since the p-values are all non-informatively small, we report the confidence intervals instead.

```

coeffs <- lapply(names(fits), function(pm) {
  tidy(fits[[pm]]) %>% mutate(pm=pm)
})
names(coeffs) <- names(fits) # was erased
cis <- lapply(fits, confint)

## Waiting for profiling to be done...
## Waiting for profiling to be done...

stopifnot(all(coeffs$pm2.5$term == rownames(cis$pm2.5)))
stopifnot(all(coeffs$pm10$term == rownames(cis$pm10)))
stopifnot(all(names(coeffs) == names(cis))) # in the same order

bind_cols(
  do.call(bind_rows, coeffs),
  do.call(bind_rows, lapply(cis, as_tibble))) %>%
  dplyr::select(pm, term, `^2.5 %`, estimate, `^97.5 %`)

## # A tibble: 16 x 5
##       pm     term          `^2.5 %` estimate `^97.5 %`
##       <chr>  <chr>        <dbl>     <dbl>     <dbl>
## 1 pm2.5 (Intercept)    7.066     7.077     7.088
## 2 pm2.5 elapsed      -0.01283   -0.01276   -0.01270
## 3 pm2.5 sensorB      0.03603    0.05400    0.07197
## 4 pm2.5 sensorC      0.09716    0.1127     0.1283
## 5 pm2.5 elapsed:sensorB -0.006435 -0.006306 -0.006177
## 6 pm2.5 elapsed:sensorC  0.002670   0.002750   0.002830
## 7 pm2.5 elapsed:has_areaTRUE -0.005011 -0.004961 -0.004910
## 8 pm2.5 elapsed:has_bioTRUE   0.001879  0.001932  0.001985
## 9 pm10 (Intercept)     7.167     7.178     7.189
## 10 pm10 elapsed      -0.01278   -0.01272   -0.01266
## 11 pm10 sensorB      0.04227    0.05947    0.07668
## 12 pm10 sensorC      0.1231     0.1381    0.1530

```

```

## 13 pm10 elapsed:sensorB      -0.006469 -0.006345 -0.006222
## 14 pm10 elapsed:sensorC      0.002535  0.002612  0.002689
## 15 pm10 elapsed:has_areaTRUE -0.005026 -0.004978 -0.004929
## 16 pm10 elapsed:has_bioTRUE   0.001877  0.001928  0.001979

```

Effect on half-life

Let λ_1 be the coefficient of time t under some first condition, and λ_2 the same coefficient under a different second condition. We can identify the marginal multiplicative effect of the second condition over the first on the half-life as follows:

$$\begin{aligned}\frac{t_{1/2, 2}}{t_{1/2, 1}} &= \frac{\ln 2/\lambda_2}{\ln 2/\lambda_1} \\ &= \frac{\lambda_1}{\lambda_2},\end{aligned}$$

where $t_{1/2, 1}$ is the half-life under the first condition, and $t_{1/2, 2}$ under the second.

Note that since the half-life is a ratio of coefficient sets of a conditions, there is no easy way to factorise out the coefficients associated with the control variables. So, we report the marginal percentage change in half-life by presence of surface area and then biological activity, once for each sensor.

```

diff_hl <- function(c2, c1, pm) {
  stopifnot(pm %in% c("pm2.5", "pm10"))
  coeff <- coeffs[[pm]]
  ci <- cis[[pm]]
  stopifnot(all(c1 %in% rownames(ci)))
  stopifnot(all(c2 %in% rownames(ci)))

  lambda1s <- c1 %>%
    sapply(function(c) {
      lower <- ci[c, 1]
      est <- filter(coeff, term == c) %>% pull(estimate)
      upper <- ci[c, 2]
      c(lower, est, upper)
    }) %>%
    t()

  lambda2s <- c2 %>%
    sapply(function(c) {
      lower <- ci[c, 1]
      est <- filter(coeff, term == c) %>% pull(estimate)
      upper <- ci[c, 2]
      c(lower, est, upper)
    }) %>%
    t()

  apply(lambda1s, 2, sum) / apply(lambda2s, 2, sum)
}

hl_design <- tibble(
  pm = c(rep("pm2.5", 6), rep("pm10", 6)),
  effect_of = rep(c(rep("area", 3), rep("bio", 3)), 2),
  sensor = rep(c("A", "B", "C"), 4)
)
hl_effects <- apply(hl_design, 1, function(row) {

```

```

pm <- row[1]; effect_of <- row[2]; sensor <- row[3]
stopifnot(pm %in% c("pm2.5", "pm10"))
stopifnot(effect_of %in% c("area", "bio"))
stopifnot(sensor %in% c("A", "B", "C"))

# Construct c1
c1 <- "elapsed"
if (sensor == "B") c1 <- c(c1, "elapsed:sensorB")
if (sensor == "C") c1 <- c(c1, "elapsed:sensorC")
if (effect_of == "bio") c1 <- c(c1, "elapsed:has_areaTRUE")

# Construct c2
if (effect_of == "area") c2 <- c(c1, "elapsed:has_areaTRUE")
if (effect_of == "bio") c2 <- c(c1, "elapsed:has_bioTRUE")

diff_hl(c2, c1, pm)
}) %>%
t() %>%
as.data.frame()

bind_cols(hl_design, hl_effects) %>%
mutate(lower=V1, estimate=V2, upper=V3) %>%
dplyr::select(-V1, -V2, -V3)

## # A tibble: 12 x 6
##   pm    effect_of sensor  lower estimate  upper
##   <chr> <chr>     <chr>   <dbl>    <dbl>   <dbl>
## 1 pm2.5 area      A     0.7191   0.7201  0.7212
## 2 pm2.5 area      B     0.7936   0.7936  0.7936
## 3 pm2.5 area      C     0.6697   0.6687  0.6678
## 4 pm2.5 bio       A     1.118    1.122   1.127
## 5 pm2.5 bio       B     1.084    1.087   1.091
## 6 pm2.5 bio       C     1.141    1.148   1.155
## 7 pm10  area      A     0.7178   0.7188  0.7198
## 8 pm10  area      B     0.7930   0.7930  0.7930
## 9 pm10  area      C     0.6709   0.6701  0.6692
## 10 pm10 bio        A     1.118    1.122   1.127
## 11 pm10 bio        B     1.084    1.087   1.091
## 12 pm10 bio        C     1.140    1.147   1.153

```

Visualisation

Pre-computation

For the purpose of visualization, we first ‘normalize’ readings by factoring out sensor effects. The resulting value is named the `\texttt{sc_value}`, as in sensor-normalized value.

```

pm2.5_glm.tb <- tidy(fits$pm2.5)

sensorB_coeff <- pm2.5_glm.tb %>%
  filter(term == "sensorB") %>%
  .$estimate
sensorC_coeff <- pm2.5_glm.tb %>%
  filter(term == "sensorC") %>%

```

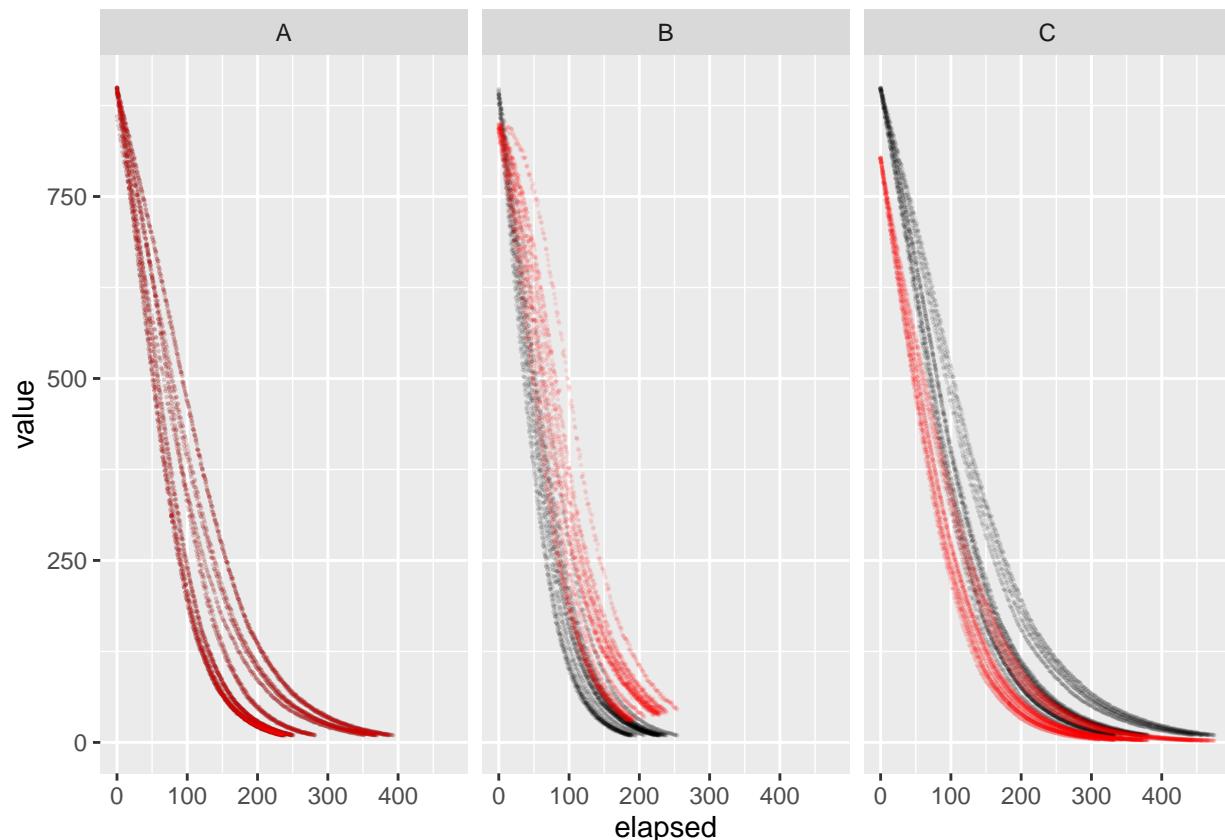
```

.$estimate
sensorB_int_coeff <- pm2.5_glm.tb %>%
  filter(term == "elapsed:sensorB") %>%
  .$estimate
sensorC_int_coeff <- pm2.5_glm.tb %>%
  filter(term == "elapsed:sensorC") %>%
  .$estimate

data$pm2.5 <- data$pm2.5 %>%
  mutate(sc_value=recode(
    sensor,
    A=value,
    B=value / exp(sensorB_coeff + sensorB_int_coeff*elapsed),
    C=value / exp(sensorC_coeff + sensorC_int_coeff*elapsed)
  ))

data$pm2.5 %>%
  ggplot(aes(x=elapsed)) +
  geom_point(aes(y=value), size=.1, alpha=.1) +
  geom_point(aes(y=sc_value), size=.1, alpha=.1, colour="red") +
  facet_grid(~sensor)

```



Repeat for PM10.

```

pm10_glm.tb <- tidy(fits$pm10)

sensorB_coeff <- pm10_glm.tb %>%

```

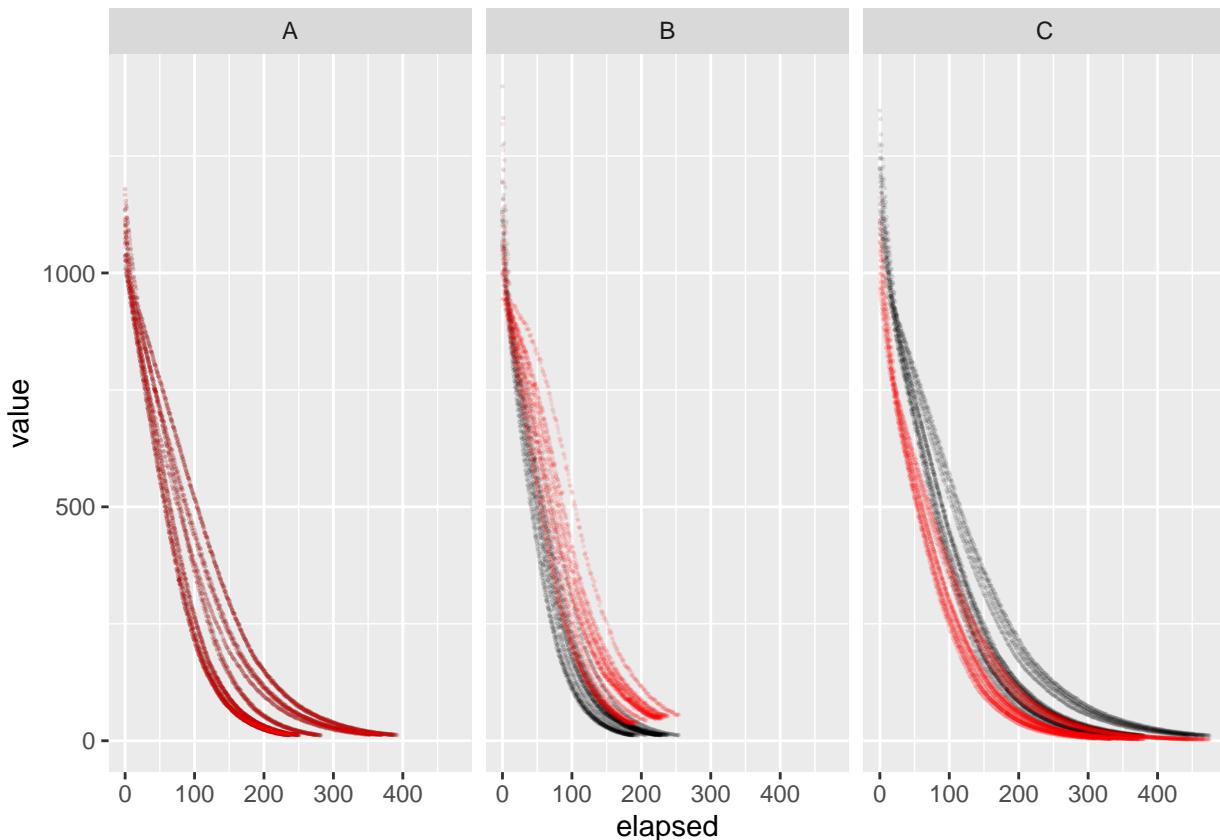
```

filter(term == "sensorB") %>%
  .$estimate
sensorC_coeff <- pm10_glm.tb %>%
  filter(term == "sensorC") %>%
  .$estimate
sensorB_int_coeff <- pm10_glm.tb %>%
  filter(term == "elapsed:sensorB") %>%
  .$estimate
sensorC_int_coeff <- pm10_glm.tb %>%
  filter(term == "elapsed:sensorC") %>%
  .$estimate

data$pm10 <- data$pm10 %>%
  mutate(sc_value=recode(
    sensor,
    A=value,
    B=value / exp(sensorB_coeff + sensorB_int_coeff*elapsed),
    C=value / exp(sensorC_coeff + sensorC_int_coeff*elapsed)
  ))

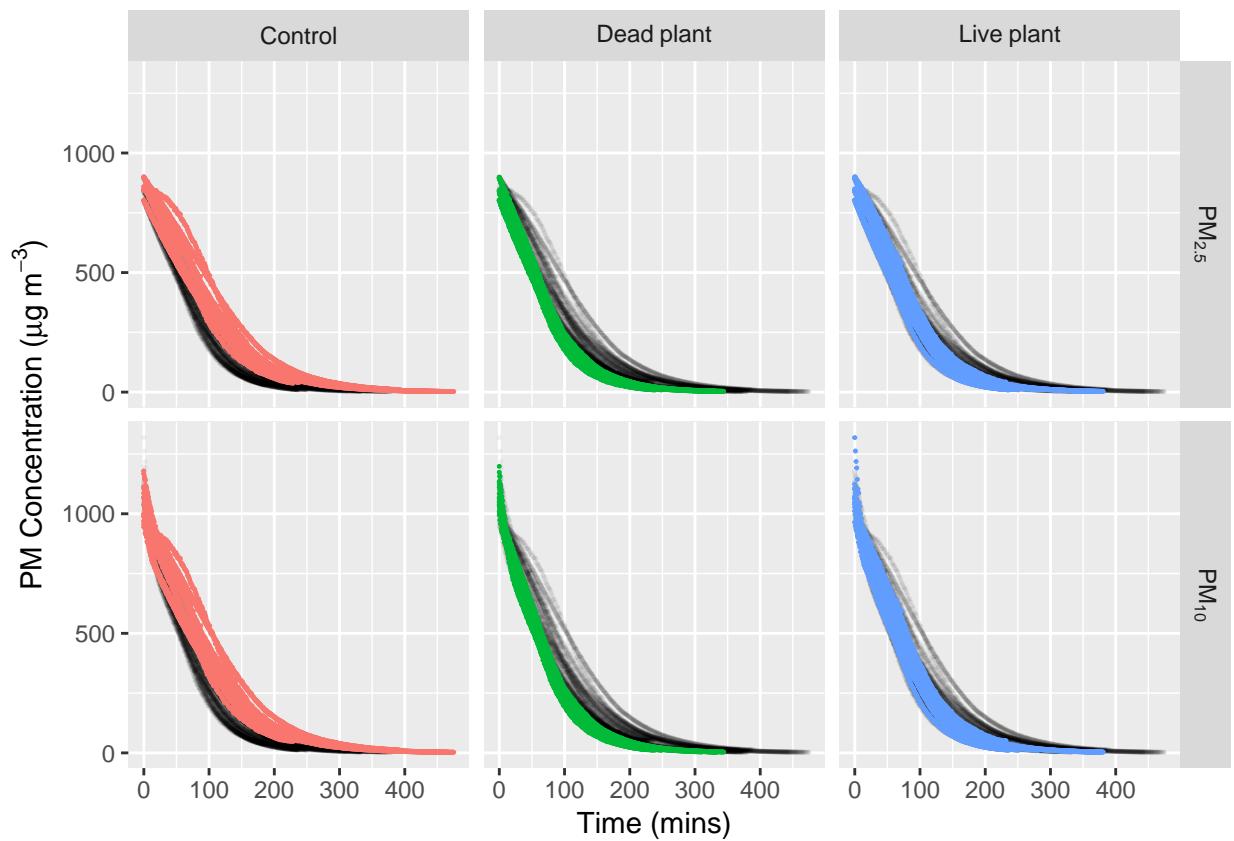
data$pm10 %>%
  ggplot(aes(x=elapsed)) +
  geom_point(aes(y=value), size=.1, alpha=.1) +
  geom_point(aes(y=sc_value), size=.1, alpha=.1, colour="red") +
  facet_grid(~sensor)

```



Final graph

```
data %>%
  do.call(bind_rows, .) %>%
  mutate(
    type=factor(
      ifelse(type == "pm2.5", "PM[2.5]", "PM[10]"),
      levels=c("PM[2.5]", "PM[10]")),
    # `~` represent spaces when parsed by ggplot:label_parsed
    condition=recode(
      condition,
      control = "Control",
      deadplant = "Dead-plant",
      liveplant = "Live~plant"
    )) %>%
  ggplot(aes(x=elapsed, y=sc_value)) +
  geom_point(
    data=bind_rows(data) %>%
      dplyr::select(-condition) %>%
      mutate(type=factor(
        ifelse(type == "pm2.5", "PM[2.5]", "PM[10]"),
        levels=c("PM[2.5]", "PM[10]"))),
    aes(x=elapsed, y=sc_value),
    size=.2, alpha=.05
  ) +
  geom_point(aes(fill=condition, colour=condition), size=.2, shape=21) +
  facet_grid(type ~ condition, labeller=label_parsed) +
  theme(legend.position = "none") +
  ylab(expression("PM Concentration (*mu*g m^-3*")")) +
  xlab(expression("Time (mins)"))
```



```
ggsave("concentration-time.eps", device=cairo_ps, width=9, height=6, units="in")
```