Data Analysis

Contents

1 Import and Tidy the Data

We have 60 sets of data in total. Each set represents a single trial of PM$_{2.5}$, PM$_{10}$ readings for one group from the four control and treatment groups. Each trial was conducted using one of three sensor-box combinations (1a, 2b, or 3d).

1.1 Define an `import` Function

Here, we define a function `import` which will

1. Import a trial from csv file,
2. Add a column to documents its sensor-box and another for its group,
3. Place the PM$_{2.5}$ and PM$_{10}$ readings into their own tibbles, then for each of the two reading types:
4. Trim away all the readings before the first time the sensor desaturates, and
5. Set the time column as being the seconds since the first desaturated reading.

By desaturation, we mean the first reading which falls within the sensor's specified range. For our model of PM sensor, this is defined as $[0, 999.9]$ µg m^{-3}.

```r
library(tidyverse)

import <- function(filename) {
  # Import a trial from csv file
  data <- read_csv(filename, col_names=c('time', 'type', 'value'))
  # Place the PM2.5 and PM10 readings into their own tibbles
  data <- list(
    pm2.5=(data %>%
      subset(type=='pm2.5') %>%
      mutate(value=as.numeric(value)) %>%
      mutate(time=as.numeric(as.POSIXct(time)))),
    pm10=(data %>%
      subset(type=='pm10') %>%
      mutate(value=as.numeric(value)) %>%
      mutate(time=as.numeric(as.POSIXct(time))))
  )

  # Add a column to documents its group
  if (grepl('nothing', filename, fixed=T)) {
    data$group <- 'nothing'
  }
  else if (
    grepl('plant', filename, fixed=T) &
    !grepl('deadplant', filename, fixed=T)
  ) {
    data$group <- 'plant'
  }
  else if (grepl('deadplant', filename, fixed=T)) {
    data$group <- 'deadplant'
  } else if (grepl('soil', filename, fixed=T)) {
    data$group <- 'soil'
  } else {
    stop(paste('import: Invaid group in', filename))
  }
  # Add a column to documents its sensor-box
  if(grepl('1[aA]', filename)) {
    data$sensorbox <- '1a'
  } else if (grepl('2[bB]', filename)) {
    data$sensorbox <- '2b'
  } else if (grepl('3[dD]', filename)) {
    data$sensorbox <- '3d'
  } else {
    stop(paste('import: Invalid sensor-box in', filename))
  }

  # Trim away all the readings before the first time the sensor desaturates
  pm2.5_last_saturated <- last(which(data$pm2.5$value>=999.9))
  data$pm2.5 <- data$pm2.5[-(1:pm2.5_last_saturated),]
  pm10_last_saturated <- last(which(data$pm10$value>=999.9))
  data$pm10 <- data$pm10[-(1:pm10_last_saturated),]

  # Set the time column as being the seconds since the first desaturated reading
  data$pm2.5$time <- data$pm2.5$time - data$pm2.5$time[1]
  data$pm10$time <- data$pm10$time - data$pm10$time[1]

  return(data)
}
```

1.2 Apply the `import` Function

Then, we import all our data by first building a vector of their filenames.

```
filenames <- c(
  '19-octa-nothing-1a.csv',  '19-octa-nothing-2b.csv', '19-octa-nothing-3d.csv',
  '20octa-nothing1a.csv',    '20octa-nothing2b.csv',   '20octa-nothing3d.csv',
  '21octa-nothing-1a.csv',   '21octa-nothing-2b.csv',  '21octa-nothing-3d.csv',
  '21octb-nothing-1a.csv',   '21octb-nothing-2b.csv',  '21octb-nothing-3d.csv',
  '22octa-nothing-1a.csv',   '22octa-nothing-2b.csv',  '22octa-nothing-3d.csv',
  '25octa-deadplant-1a.csv', '25octa-plant-3D.csv',    '25octa-soil-2b.csv',
  '25octb-deadplant-1a.csv', '25octb-plant-3D.csv',    '25octb-soil-2b.csv',
  '26octa-deadplant-1a.csv', '26octa-plant-3d.csv',    '26octa-soil-2b.csv',
  '27octa-deadplant-1a.csv', '27octa-plant-3d.csv',    '27octa-soil-2b.csv',
  '28octa-deadplant-1a.csv', '28octa-plant-3d.csv',    '28octa-soil-2b.csv',
  '2nova-deadplant-2b.csv',  '2nova-plant-1a.csv',     '2nova-soil-3d.csv',
  '2novb-deadplant-2b.csv',  '2novb-plant-1a.csv',     '2novb-soil-3d.csv',
  '30octa-deadplant-2b.csv', '30octa-plant-1a.csv',    '30octa-soil-3d.csv',
  '30octb-deadplant-2b.csv', '30octb-plant-1a.csv',    '30octb-soil-3d.csv',
  '31octb-deadplant-2b.csv', '31octb-plant-1a.csv',    '31octb-soil-3d.csv',
  '3nova-deadplant-3d.csv',  '3nova-plant-2b.csv',     '3nova-soil-1a.csv',
  '4nova-deadplant-3d.csv',  '4nova-plant-2b.csv',     '4nova-soil-1a.csv',
  '5nova-deadplant-3d.csv',  '5nova-plant-2b.csv',     '5nova-soil-1a.csv',
  '6nova-deadplant-3d.csv',  '6nova-plant-2b.csv',     '6nova-soil-1a.csv',
  '7nova-plant-2b.csv',      '7nova-deadplant-3d.csv', '7nova-soil-1a.csv'
)
```

Followed by an `lapply` of the `import` function over the filenames vector to obtain a list of lists. Each nested list is a trial.

```r
data <- lapply(filenames, function(fn) {
  import(paste('05-analysis-proper/', fn, sep=''))})
```

2 Calibrate Sensors

We observed that each sensor is not calibrated with the others, and varies sometimes in a non-linear fashion (and so cannot be easily controlled for in the linear model). In order to calibrate these sensors, we ran another trial where

- Each sensor is placed in the same box and reads during the same session, and
- Each sensor reads within 5s of each other.

An arbitrary sensor, the one labelled 3, is selected as the baseline for calibration. We then run a linear regression with quadratic terms, in a pairwise manner between the sensors, in order to obtain estimates for a 'calibration function'.

2.1 Import and Tidy the Calibration Data

First, load the readings for the individual sensors. Then, combine them into one tibble. Note that each sensor's time stamp is synchronised because they are given by the same raspberry pi zero computer.

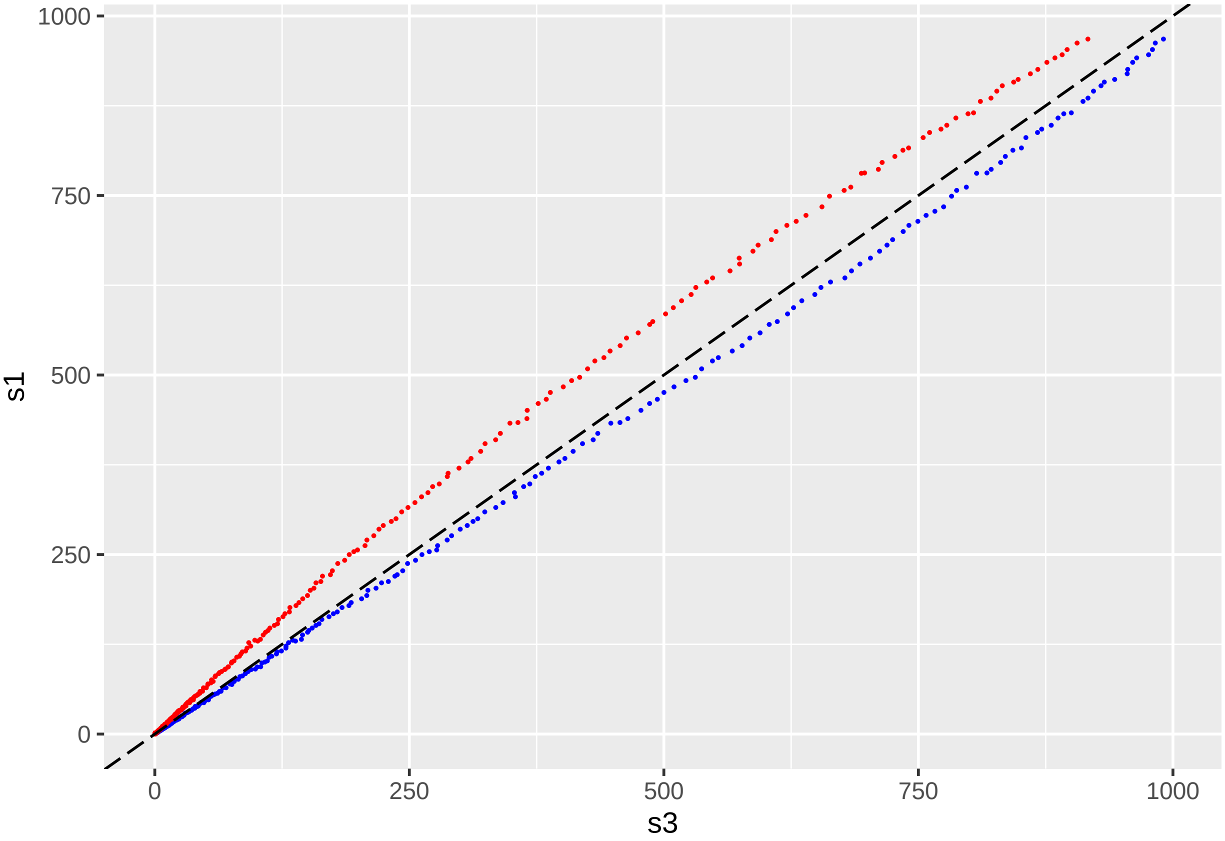
```
import_cdata <- function(filename) {
  data <- filename %>%
    read_csv(col_names=c('time', 'type', 'value')) %>%
    subset(type != 'comment') %>%
    mutate(value=as.numeric(value)) %>%
    mutate(time=as.numeric(as.POSIXct(time)))
}

# The three sensors were set-up such that they all began at the same time, +- 1s.
sensor1 <- import_cdata('03-sensor-calibration/16octc-nothing-1bigbox.csv')
sensor2 <- import_cdata('03-sensor-calibration/16octc-nothing-2bigbox.csv')
sensor3 <- import_cdata('03-sensor-calibration/16octc-nothing-3bigbox.csv')

# Sometimes, the an interval (supposedly per-minute) fails to read, and a row is missing.
# So, we want to make sure that each row index of sensor 1/2 aligns with sensor 3.
grouped_pm2.5 <- tibble(s1=double(), s2=double(), s3=double())
sensor1_pm2.5 <- subset(sensor1, type=='pm2.5')
sensor2_pm2.5 <- subset(sensor2, type=='pm2.5')
sensor3_pm2.5 <- subset(sensor3, type=='pm2.5')

# Also, trim the pre-desaturation data. The 999.9 are outliers with high leverage, and
# the data leading up to saturation is high variance.
last_saturated <- max(
  last(which(sensor1_pm2.5$value >= 999.9)),
  last(which(sensor2_pm2.5$value >= 999.9)),
  last(which(sensor3_pm2.5$value >= 999.9))
)
sensor1_pm2.5 <- sensor1_pm2.5[-(1:last_saturated),]
sensor2_pm2.5 <- sensor2_pm2.5[-(1:last_saturated),]
sensor3_pm2.5 <- sensor3_pm2.5[-(1:last_saturated),]
for (i in seq(1, nrow(sensor3_pm2.5))) {
  s3 <- sensor3_pm2.5[i,]
  s1 <- sensor1_pm2.5 %>% mutate(time=time-s3[1,]$time) %>% subset(abs(time) <= 5)
  s2 <- sensor2_pm2.5 %>% mutate(time=time-s3[1,]$time) %>% subset(abs(time) <= 5)
  if (nrow(s1) == 0 & nrow(s2) == 0) {
    next
  } else if (nrow(s1) == 0) {
    grouped_pm2.5 <- bind_rows(
      grouped_pm2.5, list(s1=NA, s2=s2[1,]$value, s3=s3[1,]$value))
  } else if (nrow(s2) == 0) {
    grouped_pm2.5 <- bind_rows(
      grouped_pm2.5, list(s1=s1[1,]$value, s2=NA, s3=s3[1,]$value))
  } else {
    grouped_pm2.5 <- bind_rows(
      grouped_pm2.5, list(s1=s1[1,]$value, s2=s2[1,]$value, s3=s3[1,]$value))
  }
}

# Repeat for PM10...
grouped_pm10 <- tibble(s1=double(), s2=double(), s3=double())
sensor1_pm10 <- subset(sensor1, type=='pm10')
sensor2_pm10 <- subset(sensor2, type=='pm10')
sensor3_pm10 <- subset(sensor3, type=='pm10')
last_saturated <- max(
  last(which(sensor1_pm10$value >= 999.9)),
  last(which(sensor2_pm10$value >= 999.9)),
  last(which(sensor3_pm10$value >= 999.9))
)
sensor1_pm10 <- sensor1_pm10[-(1:last_saturated),]
sensor2_pm10 <- sensor2_pm10[-(1:last_saturated),]
sensor3_pm10 <- sensor3_pm10[-(1:last_saturated),]
for (i in seq(1, nrow(sensor3_pm10))) {
  s3 <- sensor3_pm10[i,]
  s1 <- sensor1_pm10 %>% mutate(time=time-s3[1,]$time) %>% subset(abs(time) <= 5)
  s2 <- sensor2_pm10 %>% mutate(time=time-s3[1,]$time) %>% subset(abs(time) <= 5)
  if (nrow(s1) == 0 & nrow(s2) == 0) {
    next
  } else if (nrow(s1) == 0) {
    grouped_pm10 <- bind_rows(
      grouped_pm10, list(s1=NA, s2=s2[1,]$value, s3=s3[1,]$value))
  } else if (nrow(s2) == 0) {
    grouped_pm10 <- bind_rows(
      grouped_pm10, list(s1=s1[1,]$value, s2=NA, s3=s3[1,]$value))
  } else {
    grouped_pm10 <- bind_rows(
      grouped_pm10, list(s1=s1[1,]$value, s2=s2[1,]$value, s3=s3[1,]$value))
  }
}

data_calibration <- list(pm2.5=grouped_pm2.5, pm10=grouped_pm10)
```

Similar to the previous section, readings before the saturation points were trimmed away.

A quick graphical check shows that the sensors are indeed different:

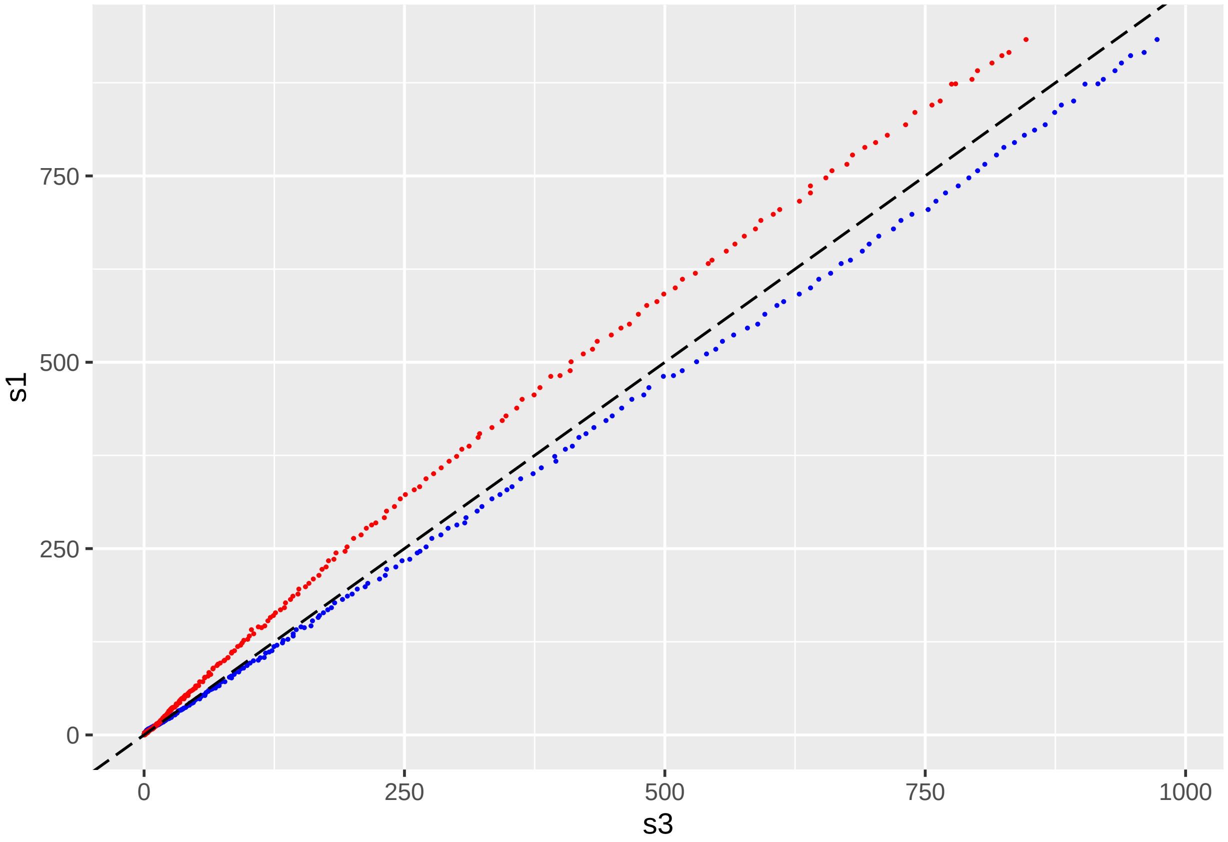
```
ggplot(data_calibration$pm2.5) +
  geom_point(mapping=aes(x=s3, y=s1), size=.2, colour='blue') +
  geom_point(mapping=aes(x=s2, y=s1), size=.2, colour='red') +
  geom_abline(linetype='longdash')
```

```
## Warning: Removed 10 rows containing missing values (geom_point).
## Warning: Removed 17 rows containing missing values (geom_point).
```



```
ggplot(data_calibration$pm10) +
  geom_point(mapping=aes(x=s3, y=s1), size=.2, colour='blue') +
  geom_point(mapping=aes(x=s2, y=s1), size=.2, colour='red') +
  geom_abline(linetype='longdash')
```

```
## Warning: Removed 10 rows containing missing values (geom_point).

## Warning: Removed 17 rows containing missing values (geom_point).
```



The missing values are a result of a glitch in obtaining readings from the PM sensors (sometimes, readings fail to be obtained from the sensors).

2.2 Calibration via Regression

Now, the calibration via regression.

```
calibration <- list(
  pm2.5=list(
    s1=lm(s1~s3+I(s3^2), data=data_calibration$pm2.5),
    s2=lm(s2~s3+I(s3^2), data=data_calibration$pm2.5)
  ),
  pm10=list(
    s1=lm(s1~s3+I(s3^2), data=data_calibration$pm10),
    s2=lm(s2~s3+I(s3^2), data=data_calibration$pm10)
  )
)

summary(calibration$pm2.5$s1)
```

```
##
## Call:
## lm(formula = s1 ~ s3 + I(s3^2), data = data_calibration$pm2.5)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -11.9730  -0.3237  -0.2314   0.2253  10.5947
```

```
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 3.696e-01  7.396e-02   4.997 7.19e-07 ***
## s3          9.236e-01  1.121e-03 823.930  < 2e-16 ***
## I(s3^2)     4.531e-05  1.392e-06  32.559  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.851 on 783 degrees of freedom
##   (10 observations deleted due to missingness)
## Multiple R-squared:  0.9999, Adjusted R-squared:  0.9999
## F-statistic: 5.05e+06 on 2 and 783 DF,  p-value: < 2.2e-16
```

```
summary(calibration$pm2.5$s2)
```

```
##
## Call:
## lm(formula = s2 ~ s3 + I(s3^2), data = data_calibration$pm2.5)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -14.6745  -0.4492  -0.3537  -0.0691  13.3057
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 5.581e-01  9.175e-02   6.083 1.84e-09 ***
## s3          6.517e-01  1.384e-03 471.028  < 2e-16 ***
## I(s3^2)     2.618e-04  1.705e-06 153.538  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.303 on 786 degrees of freedom
##   (7 observations deleted due to missingness)
## Multiple R-squared:  0.9999, Adjusted R-squared:  0.9999
## F-statistic: 2.642e+06 on 2 and 786 DF,  p-value: < 2.2e-16
```

```
summary(calibration$pm10$s1)
```

```
##
## Call:
## lm(formula = s1 ~ s3 + I(s3^2), data = data_calibration$pm10)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -10.1701  -0.8193  -0.3620   0.7817  11.9315
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 1.140e+00  6.993e-02   16.31   <2e-16 ***
## s3          9.297e-01  1.130e-03  822.51   <2e-16 ***
## I(s3^2)     2.480e-05  1.478e-06   16.78   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 1.74 on 763 degrees of freedom
##   (10 observations deleted due to missingness)
## Multiple R-squared:  0.9999, Adjusted R-squared:  0.9999
## F-statistic: 4.267e+06 on 2 and 763 DF,  p-value: < 2.2e-16
```

```
summary(calibration$pm10$s2)
```

```
##
## Call:
## lm(formula = s2 ~ s3 + I(s3^2), data = data_calibration$pm10)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -10.4328  -0.9058  -0.6122   0.8488   8.6659
##
## Coefficients:
##              Estimate  Std. Error t value Pr(>|t|)
## (Intercept) 1.070e+00  7.097e-02    15.08  <2e-16 ***
## s3          6.773e-01  1.137e-03  595.55   <2e-16 ***
## I(s3^2)     1.937e-04  1.476e-06  131.26   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.772 on 766 degrees of freedom
##   (7 observations deleted due to missingness)
## Multiple R-squared:  0.9999, Adjusted R-squared:  0.9999
## F-statistic: 3.118e+06 on 2 and 766 DF,  p-value: < 2.2e-16
```

2.3 Define a calibrate Function

Finally, package it all into a function.

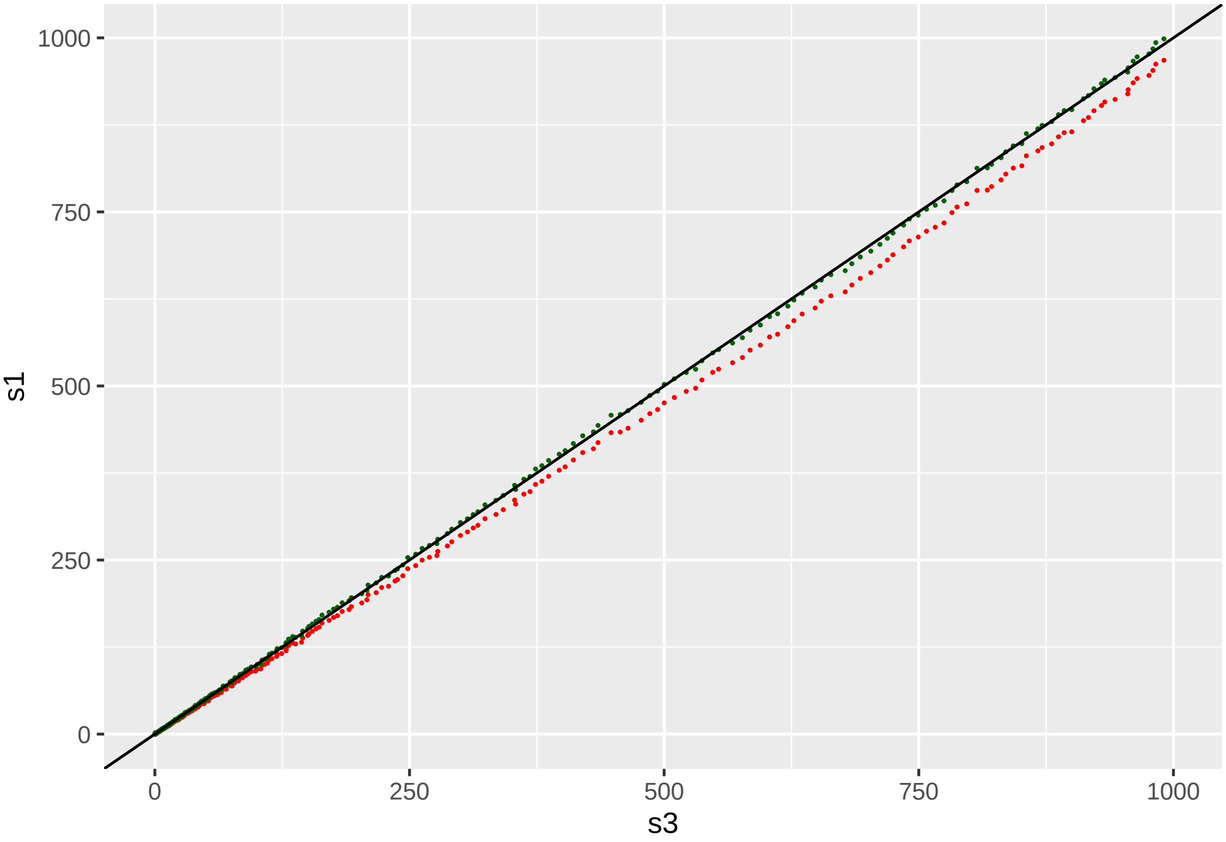
```
calibrate_one <- function(pm, sensorbox, x) {
  if (is.na(x)) return(NA)
  if (pm == 'pm2.5' & sensorbox == '1a') {
    lm_obj <- calibration$pm2.5$s1
  } else if (pm == 'pm2.5' & sensorbox == '2b') {
    lm_obj <- calibration$pm2.5$s2
  } else if (pm == 'pm10' & sensorbox == '1a') {
    lm_obj <- calibration$pm10$s1
  } else if (pm == 'pm10' & sensorbox == '2b') {
    lm_obj <- calibration$pm10$s2
  } else {
    stop(paste('Invalid pm/sensorbox: ', pm, sensorbox))
  }
  coeff <- lm_obj$coefficients - c(x, 0, 0)
  roots <- Re(polyroot(coeff))
  root <- roots[roots>=0]
  if (length(root) == 1) return(root)
  return(NA)  # would use NULL, but can't have NA in the middle of a vector
}

calibrate <- function(pm, sensorbox, xs) {
  return(unlist(Map(
    function(x) calibrate_one(pm, sensorbox, x), xs)))
}
```

Some closing graphical sanity checks:

```
ggplot(data_calibration$pm2.5 %>% mutate(s1.fix=calibrate('pm2.5', '1a', s1))) +
  geom_point(mapping=aes(x=s3, y=s1), colour='red', size=.2) +
  geom_point(mapping=aes(x=s3, y=s1.fix), colour='darkgreen', size=.2) +
  geom_abline()
```

```
## Warning: Removed 10 rows containing missing values (geom_point).
```

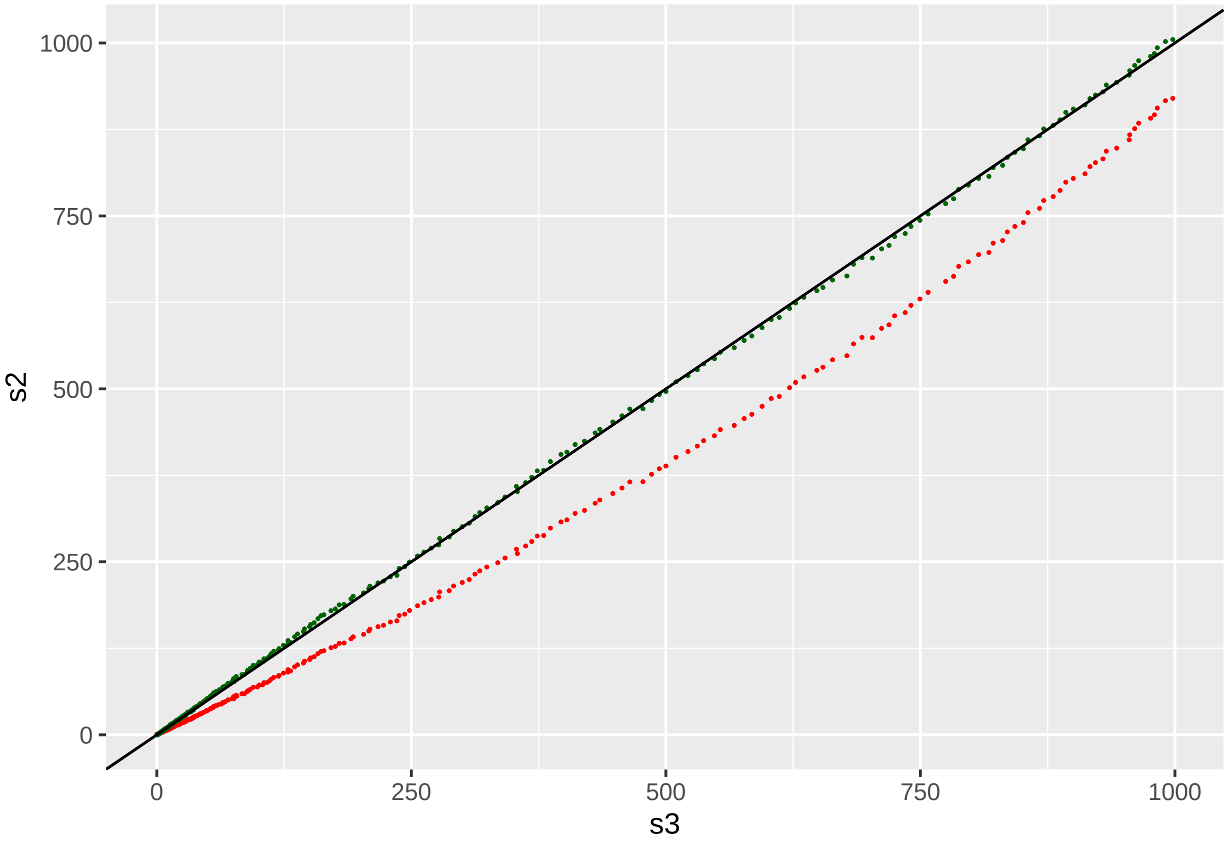
```
## Warning: Removed 127 rows containing missing values (geom_point).
```



```
ggplot(data_calibration$pm2.5 %>% mutate(s2.fix=calibrate('pm2.5', '2b', s2))) +
  geom_point(mapping=aes(x=s3, y=s2), colour='red', size=.2) +
  geom_point(mapping=aes(x=s3, y=s2.fix), colour='darkgreen', size=.2) +
  geom_abline()
```

```
## Warning: Removed 7 rows containing missing values (geom_point).
```

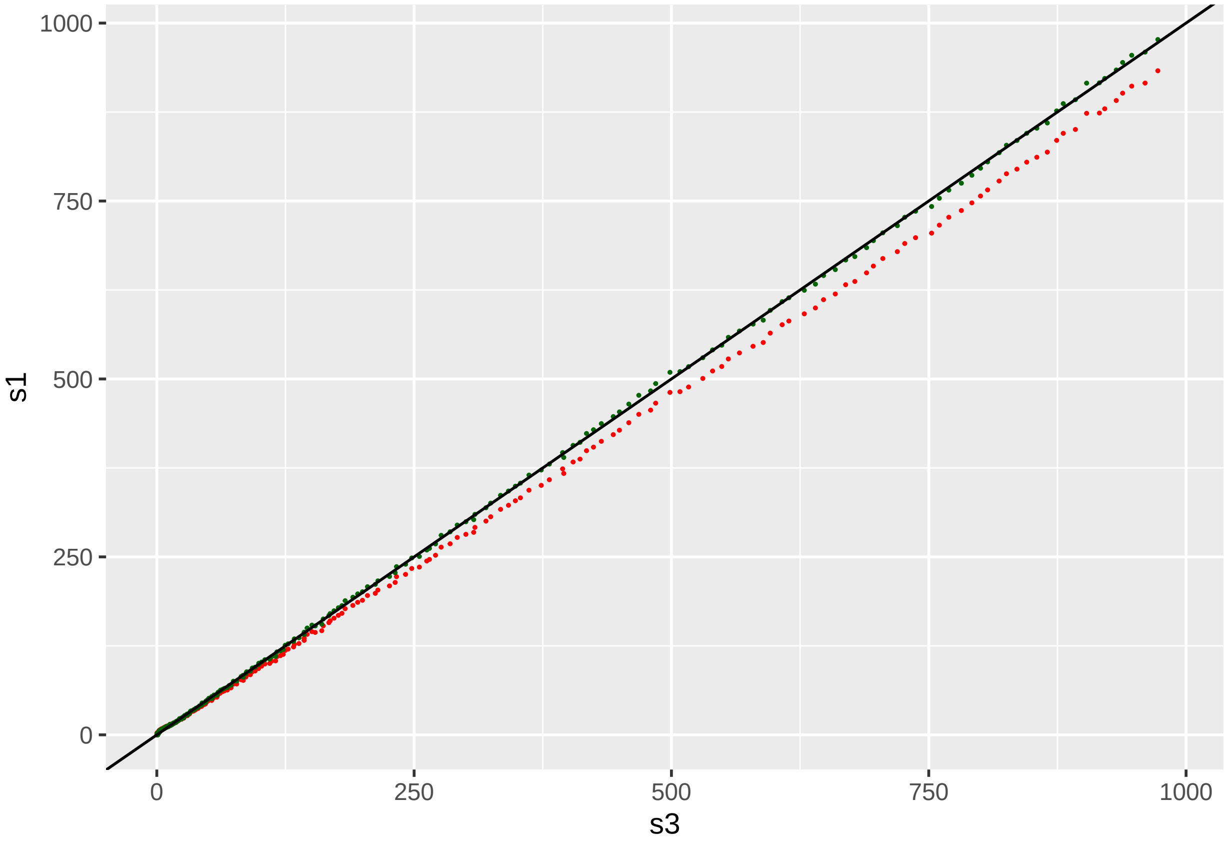
```
## Warning: Removed 377 rows containing missing values (geom_point).
```



```
ggplot(data_calibration$pm10 %>% mutate(s1.fix=calibrate('pm10', '1a', s1))) +
  geom_point(mapping=aes(x=s3, y=s1), colour='red', size=.2) +
  geom_point(mapping=aes(x=s3, y=s1.fix), colour='darkgreen', size=.2) +
  geom_abline()
```

```
## Warning: Removed 10 rows containing missing values (geom_point).
```

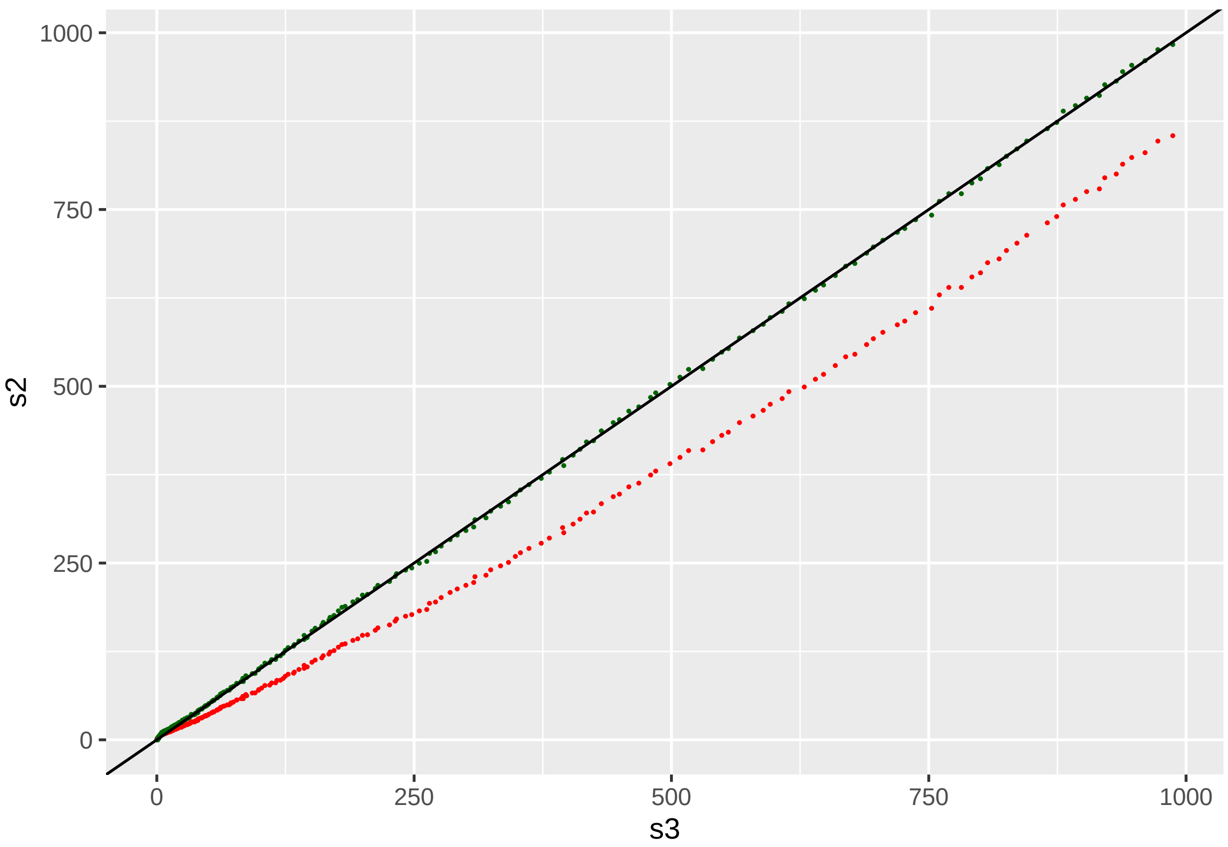
```
## Warning: Removed 274 rows containing missing values (geom_point).
```



```
ggplot(data_calibration$pm10 %>% mutate(s2.fix=calibrate('pm10', '2b', s2))) +
  geom_point(mapping=aes(x=s3, y=s2), colour='red', size=.2) +
  geom_point(mapping=aes(x=s3, y=s2.fix), colour='darkgreen', size=.2) +
  geom_abline()
```

```
## Warning: Removed 7 rows containing missing values (geom_point).
```

```
## Warning: Removed 399 rows containing missing values (geom_point).
```


1000
750
500
250
0
s2
0
250
500
750
1000
s3

3 Box Plot & Regression

First, we consider PM$_{2.5}$.

3.1 Transform the Data

First, calibrate the data using the previously-defined `calibrate` function.

```r
data_calibrated <- data %>%
  lapply(function(trial) {
    if (trial$sensorbox == '1a') {
      trial$pm2.5$value <- calibrate('pm2.5', '1a', trial$pm2.5$value)
      trial$pm10$value <- calibrate('pm10', '1a', trial$pm10$value)
    } else if (trial$sensorbox == '2b') {
      trial$pm2.5$value <- calibrate('pm2.5', '2b', trial$pm2.5$value)
      trial$pm10$value <- calibrate('pm10', '2b', trial$pm10$value)
    } else if (trial$sensorbox == '3d') {
      # noop
    } else {
      stop(paste('Invalid sensorbox', trial$sensorbox))
    }
    return(trial)
  })
```

Now, create a new tibble which records the time taken for PM$_{2.5}$ and PM$_{10}$ to decrease from $900\,\mu\mathrm{g\,m^{-3}}$ to $450\,\mu\mathrm{g\,m^{-3}}$. Since our readings are limited in resolution (per-minute), we pick the readings closest to $900\,\mu\mathrm{g\,m^{-3}}$ and $450\,\mu\mathrm{g\,m^{-3}}$.

```r
pm2.5_aggregated <- lapply(data_calibrated, function(trial) {
  # This function is applied for each trial.
  # First, find the index of the 900 reading.
  index_900 <- which.min(abs(trial$pm2.5$value - 900))
  index_450 <- which.min(abs(trial$pm2.5$value - 450))
  time <- (trial$pm2.5[index_450,]$time - trial$pm2.5[index_900,]$time) / 60
  list(sensorbox=trial$sensorbox, group=trial$group, time=time)
})

pm2.5_decrease <- tibble(
  sensorbox=sapply(pm2.5_aggregated, function(xs) xs$sensorbox),
  group=sapply(pm2.5_aggregated, function(xs) xs$group),
  time=sapply(pm2.5_aggregated, function(xs) xs$time)
)

pm2.5_decrease$has_soil <- ifelse(
  pm2.5_decrease$group %in% c('soil', 'deadplant', 'plant'), 1, 0)
pm2.5_decrease$has_sa <- ifelse(
  pm2.5_decrease$group %in% c('deadplant', 'plant'), 1, 0)
pm2.5_decrease$has_life <- ifelse(
  pm2.5_decrease$group == 'plant', 1, 0)
```

3.2 Regression

```r
pm2.5_decrease.lm <- lm(
  time~has_soil + has_sa + has_life + sensorbox, data=pm2.5_decrease)
summary(pm2.5_decrease.lm)
```

```
##
## Call:
## lm(formula = time ~ has_soil + has_sa + has_life + sensorbox,
##     data = pm2.5_decrease)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -14.7500  -5.1403  -0.0528   4.7758  15.1956
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   88.707      2.345  37.827  < 2e-16 ***
## has_soil     -16.536      2.708  -6.106 1.15e-07 ***
## has_sa        -9.406      2.708  -3.473  0.00102 **
## has_life       8.001      2.708   2.955  0.00463 **
## sensorbox2b  -20.957      2.345  -8.936 3.13e-12 ***
## sensorbox3d   20.650      2.345   8.806 5.05e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.416 on 54 degrees of freedom
## Multiple R-squared:  0.8841, Adjusted R-squared:  0.8733
## F-statistic: 82.36 on 5 and 54 DF,  p-value: < 2.2e-16
```

```r
confint(pm2.5_decrease.lm)
```

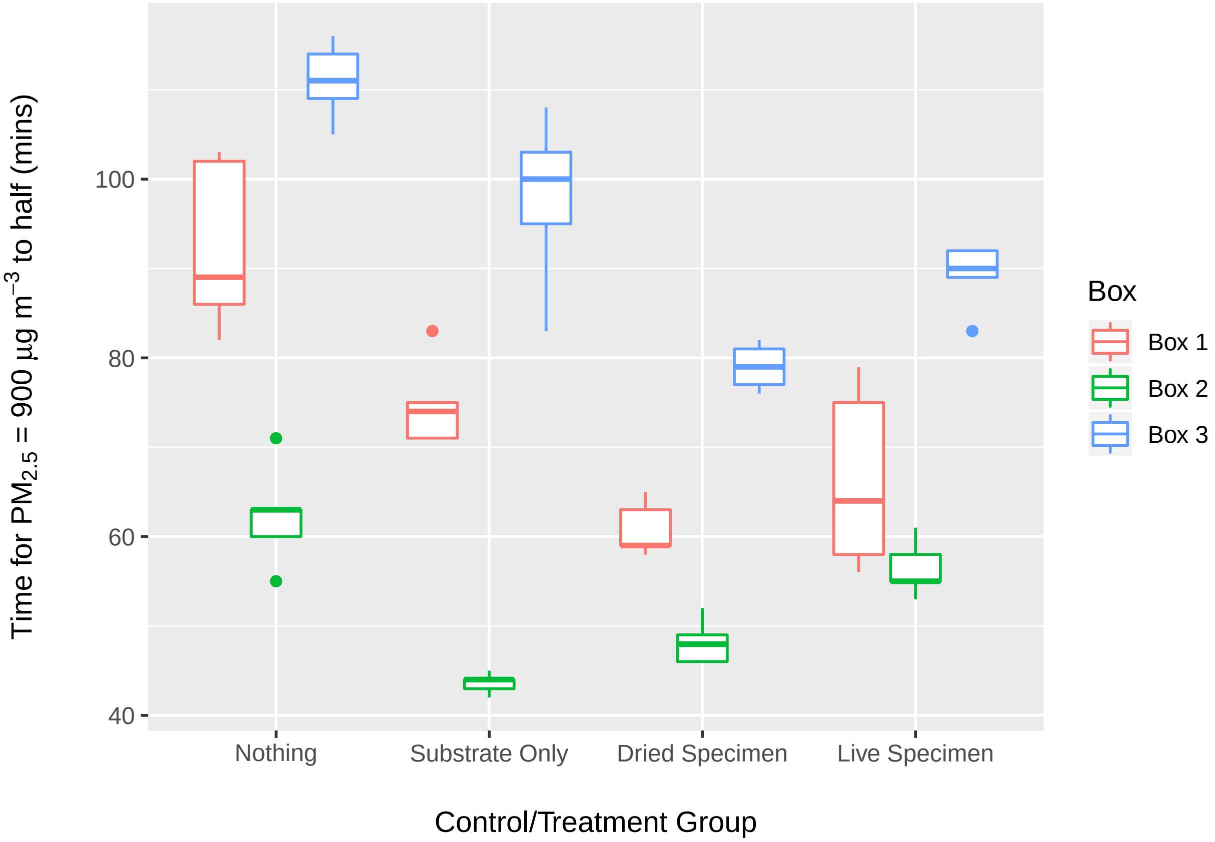
```
##                  2.5 %     97.5 %
## (Intercept)  84.005042  93.408292
## has_soil    -21.964524 -11.106587
## has_sa      -14.834524  -3.976587
## has_life      2.572142  13.430080
## sensorbox2b -25.658292 -16.255042
## sensorbox3d  15.948375  25.351625
```

```r
(confint(pm2.5_decrease.lm)[,2] - confint(pm2.5_decrease.lm)[,1]) / 2
```

```
## (Intercept)    has_soil      has_sa    has_life sensorbox2b sensorbox3d
##    4.701625    5.428969    5.428969    5.428969    4.701625    4.701625
```

3.3 Boxplot

```r
library(ggpubr)

groups <- factor(pm2.5_decrease$group, levels=c('nothing', 'soil', 'deadplant', 'plant'))
pm2.5_decrease$group <- groups

ggboxplot(pm2.5_decrease, x='group', y='time', color='sensorbox') +
    ylab(expression(atop(
      "Time for PM"[2.5]*" = 900 "*mu*"g m"^-3*" to half (mins)", ""))) +
    xlab("\nControl/Treatment Group") +
    guides(color=guide_legend(title='Box')) +
    scale_x_discrete(labels=c(
        'nothing'='Nothing',
        'soil'='Substrate Only',
        'deadplant'=expression('Dried Specimen'),
        'plant'=expression('Live Specimen'))) +
    scale_color_discrete(labels=c('Box 1', 'Box 2', 'Box 3')) +
    theme_grey()
```



```
ggsave('boxplot-pm25.eps', width=9, height=5)
```

3.4 Repeat for PM_{10}

```
pm10_aggregated <- lapply(data_calibrated, function(trial) {
  index_900 <- which.min(abs(trial$pm10$value - 900))
  index_450 <- which.min(abs(trial$pm10$value - 450))
  time <- (trial$pm10[index_450,]$time - trial$pm10[index_900,]$time) / 60
  list(sensorbox=trial$sensorbox, group=trial$group, time=time)
})

pm10_decrease <- tibble(
  sensorbox=sapply(pm10_aggregated, function(xs) xs$sensorbox),
  group=sapply(pm10_aggregated, function(xs) xs$group),
  time=sapply(pm10_aggregated, function(xs) xs$time)
)

pm10_decrease$has_soil <- ifelse(
  pm10_decrease$group %in% c('soil', 'deadplant', 'plant'), 1, 0)
pm10_decrease$has_sa <- ifelse(
  pm10_decrease$group %in% c('deadplant', 'plant'), 1, 0)
pm10_decrease$has_life <- ifelse(
  pm10_decrease$group == 'plant', 1, 0)

pm10_decrease.lm <- lm(time~has_soil + has_sa + has_life + sensorbox, data=pm10_decrease)
summary(pm10_decrease.lm)
```

```
## 
## Call:
## lm(formula = time ~ has_soil + has_sa + has_life + sensorbox, 
##     data = pm10_decrease)
## 
## Residuals:
##      Min       1Q   Median       3Q      Max 
## -11.3003  -4.3371  -0.0531   4.3937  10.4322 
## 
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept)   77.568      1.898  40.877  < 2e-16 ***
## has_soil     -15.464      2.191  -7.058 3.32e-09 ***
## has_sa        -7.873      2.191  -3.593 0.000707 ***
## has_life       6.672      2.191   3.045 0.003592 ** 
## sensorbox2b  -18.251      1.898  -9.618 2.67e-13 ***
## sensorbox3d   16.151      1.898   8.511 1.49e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 6.001 on 54 degrees of freedom
## Multiple R-squared:  0.893,  Adjusted R-squared:  0.8831 
## F-statistic: 90.17 on 5 and 54 DF,  p-value: < 2.2e-16
```

```
confint(pm10_decrease.lm)
```

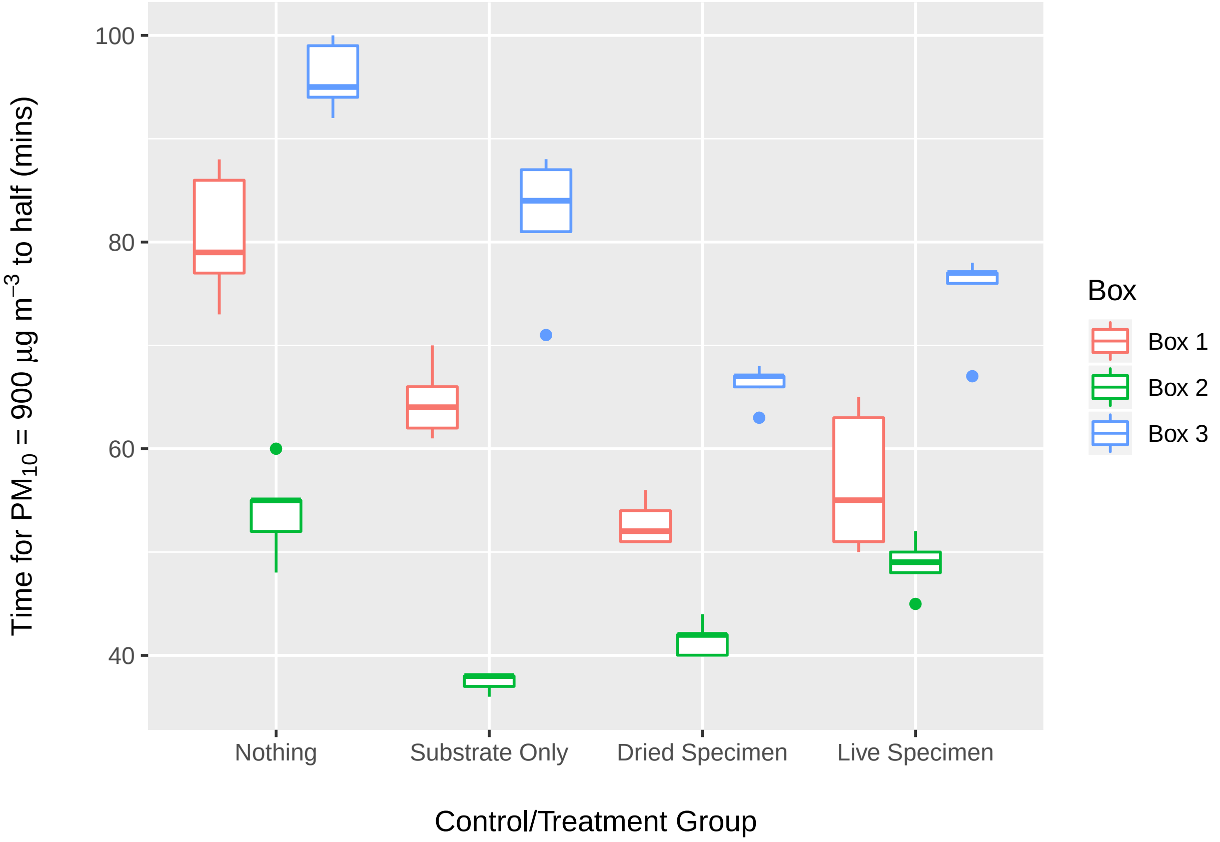
```
##                  2.5 %    97.5 %
## (Intercept)  73.763334  81.37222
## has_soil    -19.857438 -11.07145
## has_sa      -12.266327  -3.48034
## has_life      2.279229  11.06522
## sensorbox2b -22.055277 -14.44639
## sensorbox3d  12.346389  19.95528
```

```
(confint(pm10_decrease.lm)[,2] - confint(pm10_decrease.lm)[,1]) / 2
```

```
## (Intercept)    has_soil      has_sa    has_life sensorbox2b sensorbox3d 
##    3.804444    4.392994    4.392994    4.392994    3.804444    3.804444
```

```
groups <- factor(pm10_decrease$group, levels=c('nothing', 'soil', 'deadplant', 'plant'))
pm10_decrease$group <- groups
```

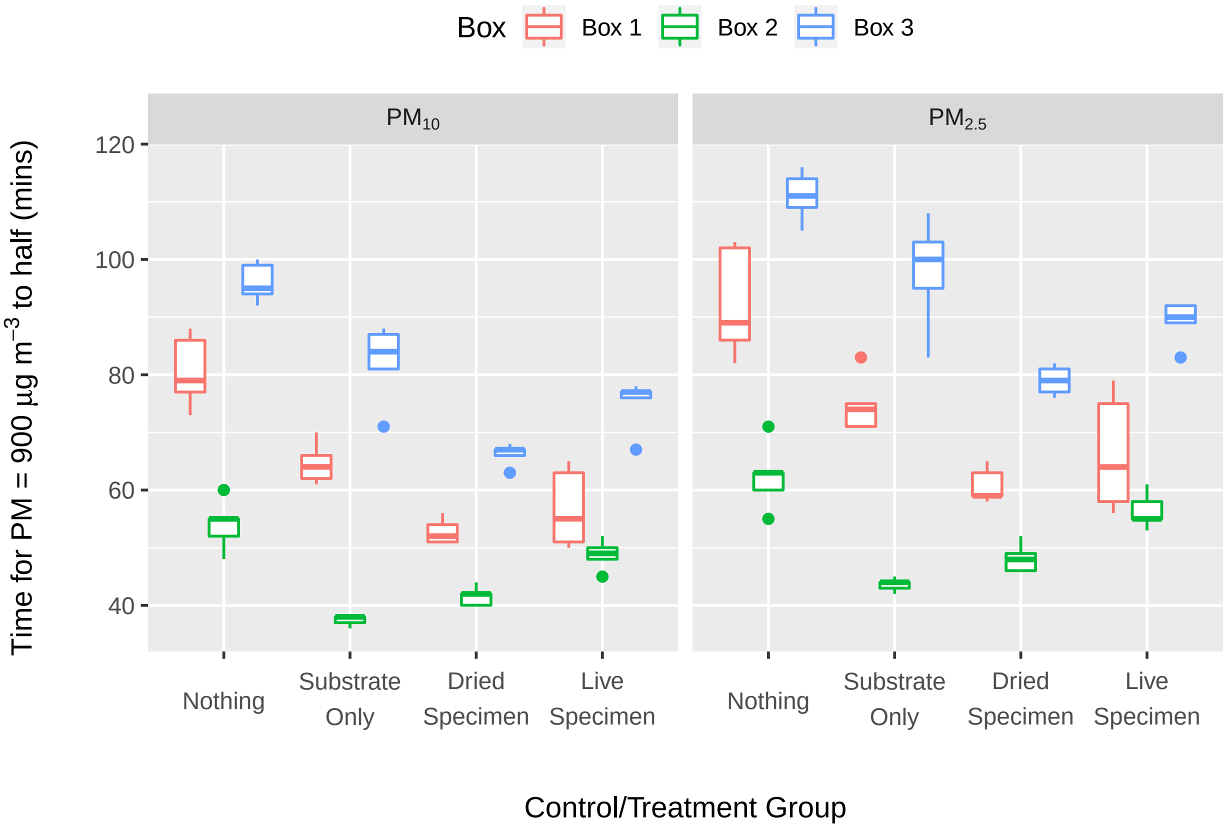
```
ggboxplot(pm10_decrease, x='group', y='time', color='sensorbox') +
    ylab(expression(atop(
      "Time for PM"[10]*" = 900 "*mu*"g m"^-3*" to half (mins)", ""))) +
    xlab("\nControl/Treatment Group") +
    guides(color=guide_legend(title='Box')) +
    scale_x_discrete(labels=c(
        'nothing'='Nothing',
        'soil'='Substrate Only',
        'deadplant'=expression('Dried Specimen'),
        'plant'=expression('Live Specimen'))) +
        scale_color_discrete(labels=c('Box 1', 'Box 2', 'Box 3')) +
    theme_grey()
```



```
ggsave('boxplot-pm10.eps', width=9, height=5)
```

3.5 Combine Boxplots for Publication

```
pm10_decrease$pm <- 'PM[10]'
pm2.5_decrease$pm <- 'PM[2.5]'

ggboxplot(bind_rows(pm10_decrease, pm2.5_decrease),
  x='group', y='time', color='sensorbox') +
  facet_grid(. ~ pm, labeller=label_parsed) +
  ylab(expression(atop("Time for PM = 900 "*mu*"g m"^-3*" to half (mins)", ""))) +
  xlab("\nControl/Treatment Group") +
  guides(color=guide_legend(title='Box')) +
  scale_x_discrete(labels=c(
      'nothing'=expression(
        atop(NA, textstyle('Nothing'))),
      'soil'=expression(atop(
        atop(NA, textstyle('Substrate')), atop(textstyle('Only'), NA))),
      'deadplant'=expression(
        atop(atop(NA, textstyle('Dried')), atop(textstyle('Specimen'), NA))),
      'plant'=expression(
        atop(atop(NA, textstyle('Live')), atop(textstyle('Specimen'), NA))))) +
      scale_color_discrete(labels=c('Box 1', 'Box 2', 'Box 3')) +
  theme_gray() +
  theme(legend.position="top")
```



```
ggsave('boxplot.eps', width=9, height=5)
```