corrections

Tony Ni

6/23/2020

Libraries

```r
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 3.6.2
```

```
## -- Attaching packages ------------------------------------------------------------------------------
```

```
## v ggplot2 3.2.1     v purrr   0.3.3
## v tibble  2.1.3     v dplyr   0.8.4
## v tidyr   1.0.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0
```

```
## Warning: package 'ggplot2' was built under R version 3.6.1
```

```
## Warning: package 'tibble' was built under R version 3.6.1
```

```
## Warning: package 'tidyr' was built under R version 3.6.2
```

```
## Warning: package 'purrr' was built under R version 3.6.1
```

```
## Warning: package 'forcats' was built under R version 3.6.1
```

```
## -- Conflicts ------------------------------------------------------------------------------------------
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```r
library(Rmisc)
```

```
## Warning: package 'Rmisc' was built under R version 3.6.3
```

```
## Loading required package: lattice
```

```
## Loading required package: plyr
```

```
## Warning: package 'plyr' was built under R version 3.6.2
## -------------------------------------------------------------------------------
```

```
## You have loaded plyr after dplyr - this is likely to cause problems.
## If you need functions from both plyr and dplyr, please load plyr first, then dplyr:
## library(plyr); library(dplyr)
```

```
## -------------------------------------------------------------------------------
```

```
##
## Attaching package: 'plyr'
```

```
## The following objects are masked from 'package:dplyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize
```

```
## The following object is masked from 'package:purrr':
##
##     compact
```

Read in data

```
setwd("~/harvard-summer-biostats")
df <- read_csv("data/wide_illinois.csv") #read in data
```

```
## Parsed with column specification:
## cols(
##   .default = col_double(),
##   well_id = col_character(),
##   site = col_character(),
##   disposal_area = col_character(),
##   type = col_character(),
##   gradient = col_character()
## )
```

```
## See spec(...) for full column specifications.
```

```
df1 <- na.omit(df) #get rid of na's

df1 <- df1[-69, ] %>%
  filter(gradient == "Upgradient")
```

Correcting upgradient values

Following up on our conversation yesterday, here's a little schematic of how I was thinking we might impute corrected values for the contaminated upgradient wells (which you have identified via clustering) and use those corrected values to estimate contamination caused by a given disposal area.

Bootstrap-type procedure for imputing "corrected" chemical measures for contaminated upgradient wells and estimating contaminant concentrations caused by a given coal ash disposal area

Set B=1000 (or some large number)

For each chemical do the following:

1. Separate the upgradient well data into 2 datasets, one containing the non-contaminated wells and one containing the contaminated wells

```
contam <- df1 %>%
  filter(well_id %in% c("G01D", "G02D", "G03D", "MW201", "MW203", "MW24D"))

'%ni%' <- Negate('%in%')

n_contam <- df1 %>%
  filter(well_id %ni% c("G01D", "G02D", "G03D", "MW201", "MW203", "MW24D"))
```

2. Randomly sample (with replacement) B times from the measurements of the chemical from non-contaminated upgradient wells to create an empirical distribution of naturally occurring chemical levels. These will serve as the set of imputed "corrected" measurements of the chemical for each contaminated upgradient well.

```
#sample with replacement n times from non contaminated well
set.seed(7271999)
n <- 500
sample_n_contam <- sample_n(n_contam, n, replace = TRUE) %>%
  select(6:26)
```

3. For each contaminated upgradient well, compute the average of the downgradient well chemical measurements for the corresponding disposal area.

```
#checking which disposal areas our contaminated wells belong to
unique(contam$disposal_area)
```

```
## [1] "Joppa East Ash Pond" "Near Field Landfill"
```

The wells belonging to our contaminated data set are "Joppa East Ash Pond" and "Near Field Landfill".

```
#for each contaminated upgradient well, compute the average of the downgradient
#well chemical measurements for the corresponding disposal area
downgrad <- df %>%
  filter(disposal_area %in% c("Joppa East Ash Pond", "Near Field Landfill")) %>%
  filter(gradient == "Downgradient")

#avg of corresp. downgrad well for corresp. disposal area
mean_df <- downgrad %>%
  gather("contaminant", "concentration", 6:26) %>%
  group_by(disposal_area, contaminant) %>%
  summarise_each(funs(mean)) %>%
  select(2, 7) %>%
  spread(contaminant, concentration) %>%
  ungroup() %>%
  select(2:22)
```

```
## Adding missing grouping variables: `disposal_area`
```

Subtract each of the B imputed upgradient measurements from the average downgradient measure. This creates a distribution of B values of the contaminant concentrations caused by the disposal area. We can then take the mean of these B values as the estimate of the contamination caused by the disposal area (for the given chemical) and then use the 2.5 percentile and 97.5 percentile of the distribution as a bootstrap-type confidence interval.

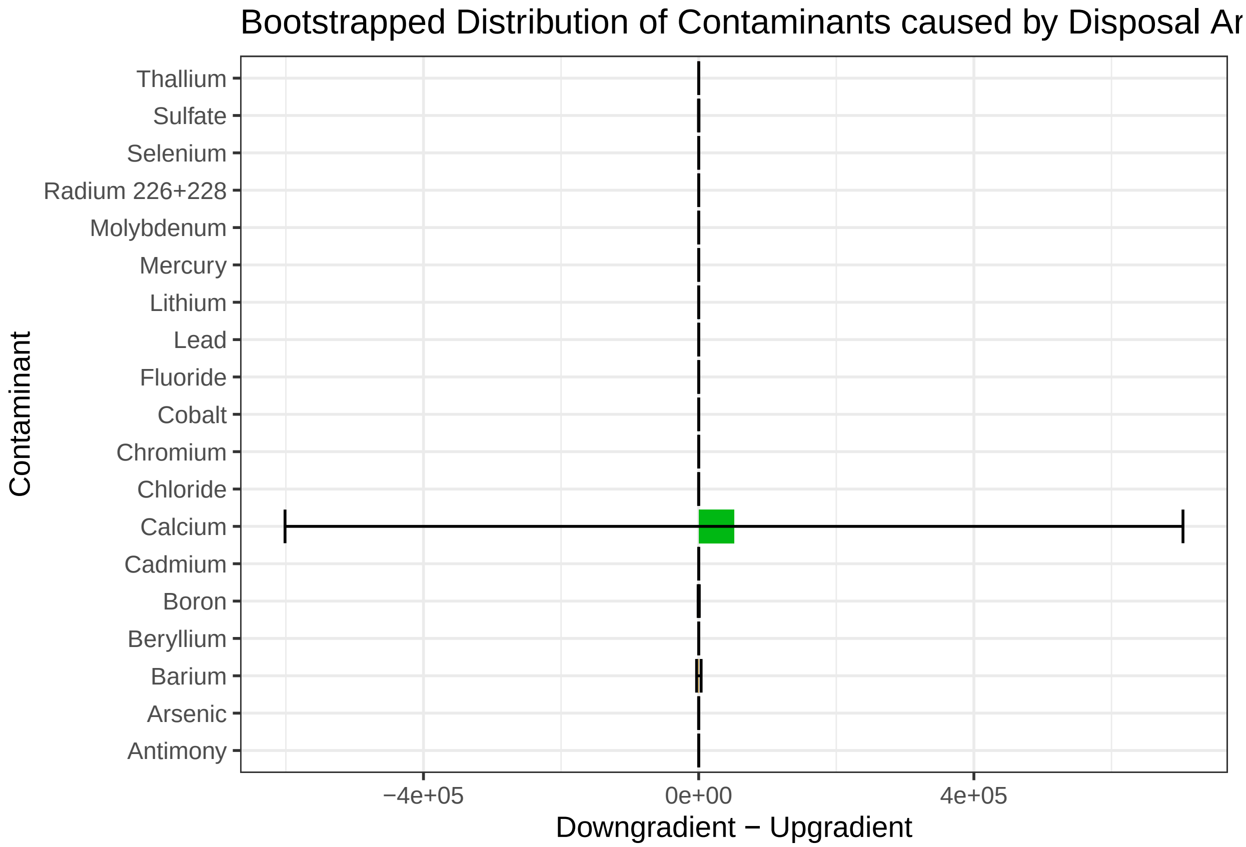
```r
#downgrad - upgrad
#mean_df (avg of contam valuse) - sample_n_contam (imputed correct valuse)
first <- map2_df(mean_df, sample_n_contam[1, ], `-`)
second <- map2_df(mean_df, sample_n_contam[2, ], `-`)

combined <- full_join(first, second) %>%
  gather("contaminant", "concentration", 1:21)
```

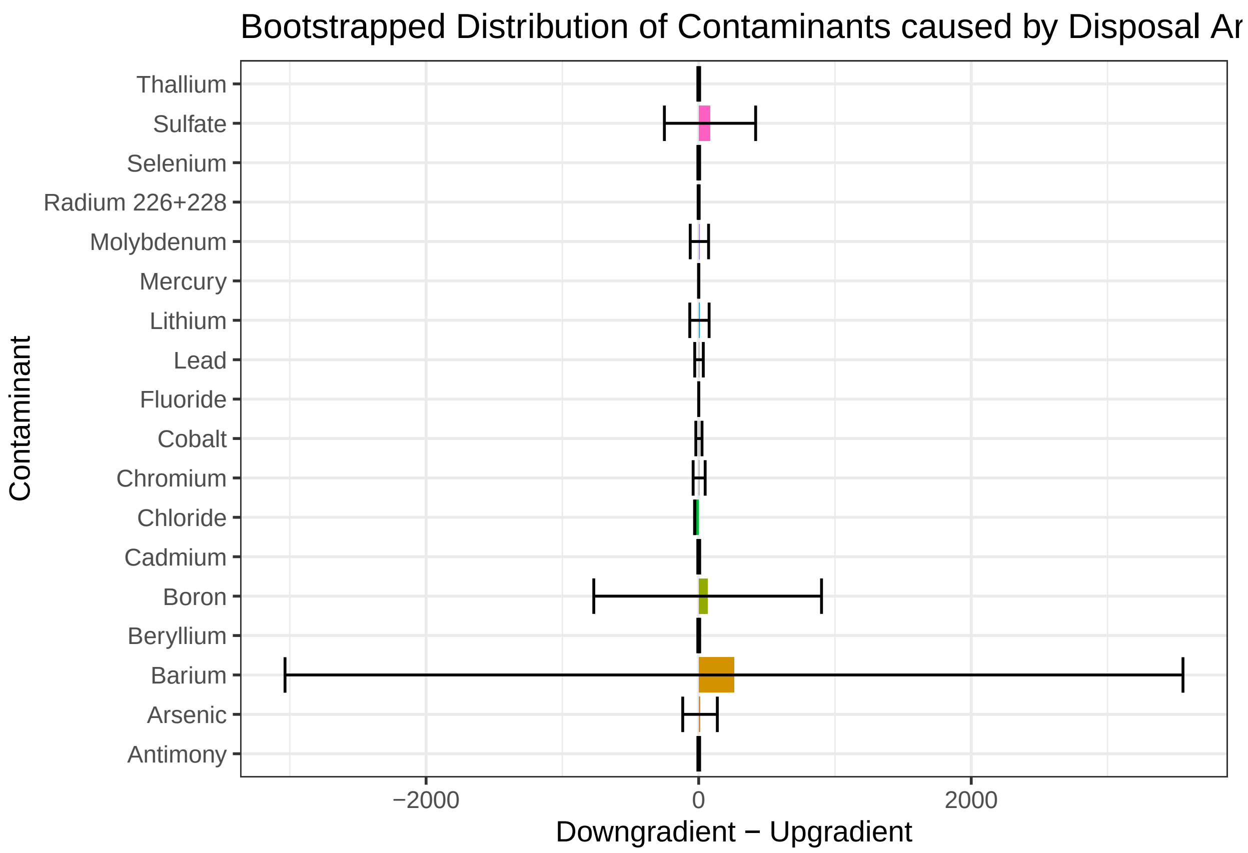
```
## Joining, by = c("Antimony", "Arsenic", "Barium", "Beryllium", "Boron", "Cadmium", "Calcium", "Chlori
```

```r
combined2 <- combined %>%
  group_by(contaminant) %>%
  dplyr::summarize(avg_conc = mean(concentration),
            upper = CI(concentration)[1],
            lower = CI(concentration)[3])
```

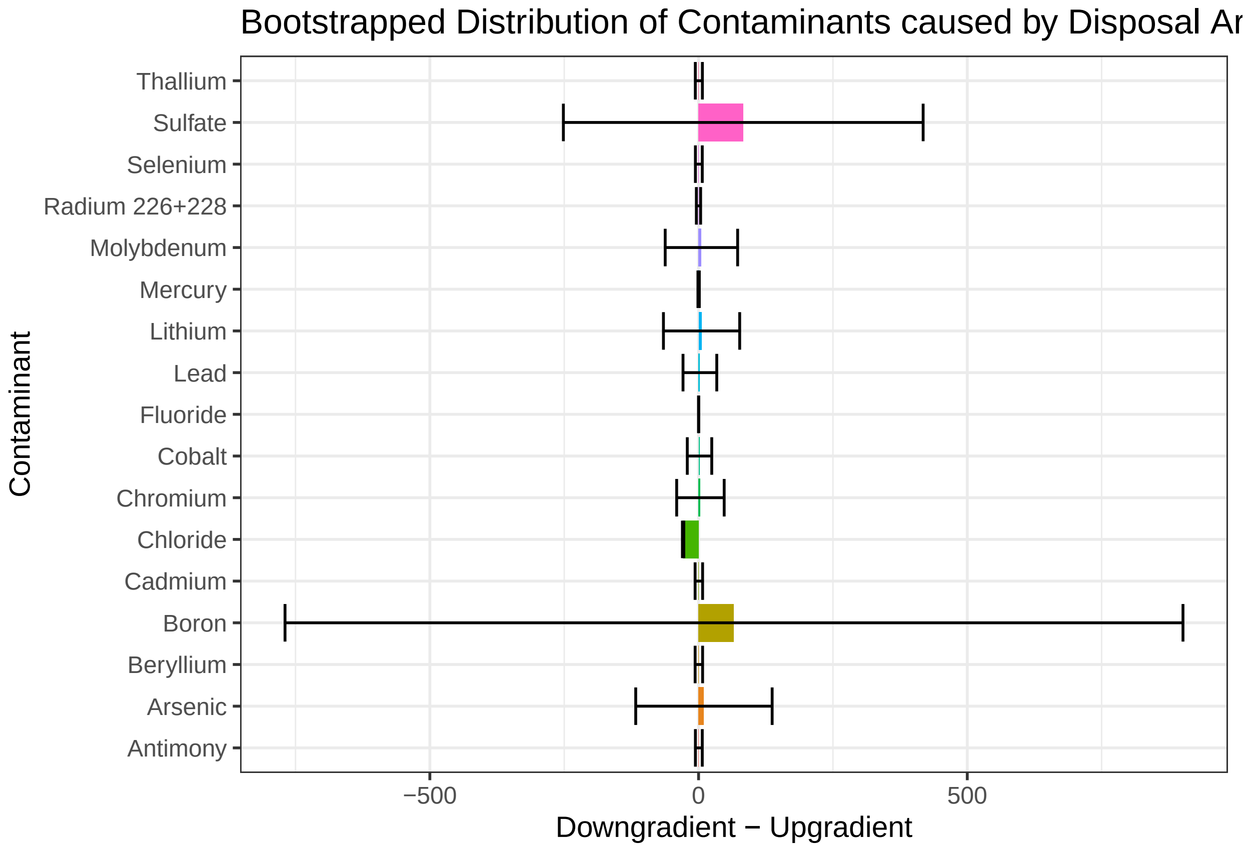
```r
#er... calcium is extreme...
combined2 %>%
  filter(contaminant %ni% c("Total Dissolved Solids", "pH"))%>%
  ggplot(aes(x = contaminant, y = avg_conc, fill = contaminant)) +
  geom_bar(stat = "identity", position = "dodge") +
  geom_errorbar(aes(ymin = lower, ymax = upper), position = "dodge") +
  ggtitle("Bootstrapped Distribution of Contaminants caused by Disposal Area (n=500)") +
  xlab("Contaminant") +
  ylab("Downgradient - Upgradient") +
  theme_bw() +
  guides(fill=FALSE) +
  coord_flip()
```



```
#w/o calcium
combined2 %>%
  filter(contaminant %ni% c("Calcium", "Total Dissolved Solids", "pH")) %>%
  ggplot(aes(x = contaminant, y = avg_conc, fill = contaminant)) +
  geom_bar(stat = "identity", position = "dodge") +
  geom_errorbar(aes(ymin = lower, ymax = upper), position = "dodge") +
  ggtitle("Bootstrapped Distribution of Contaminants caused by Disposal Area (n=500)") +
  xlab("Contaminant") +
  ylab("Downgradient - Upgradient") +
  theme_bw() +
  guides(fill=FALSE) +
  coord_flip()
```



```
#w/o calcium and w/o barium
combined2 %>%
  filter(contaminant %ni% c("Calcium", "Total Dissolved Solids", "pH", "Barium")) %>%
  ggplot(aes(x = contaminant, y = avg_conc, fill = contaminant)) +
  geom_bar(stat = "identity", position = "dodge") +
  geom_errorbar(aes(ymin = lower, ymax = upper), position = "dodge") +
  ggtitle("Bootstrapped Distribution of Contaminants caused by Disposal Area (n=500)") +
  xlab("Contaminant") +
  ylab("Downgradient - Upgradient") +
  theme_bw() +
  guides(fill=FALSE) +
  coord_flip()
```



We can compare these estimates to the contaminant concentrations caused by disposal areas with non-contaminated upgradient wells (which we can compute just by taking the difference in the average downgradient and upgradient measurements for those disposal areas).