Cross-Sectional G-Formula

A. Keil and M.Kamenetsky

2025-02

```
#Load in libraries
library(ggplot2) #for plotting
library(dplyr) #for data cleaning
library(survival) #for survival analysis
library(boot) #for bootstrapping
library(ggcorrplot)
library(ggdag)
```

Research Questions:

- To estimate the joint effects of air co-pollutants from a coal-fired power plant on 2 year old mental development index (MDI) score under various reduction scenarios

Learning Objectives:

By the end of this tutorial, you will be able to:

- Develop a model based on an directed acylic graph (DAG)
- Complete a parametric g-formula analysis using `R` software
- Estimate different joint effects under different reduction scenarios
- Bootstrap confidence intervals for effect estimates under different reduction scenarios
- Interpret results in a causal inference framework

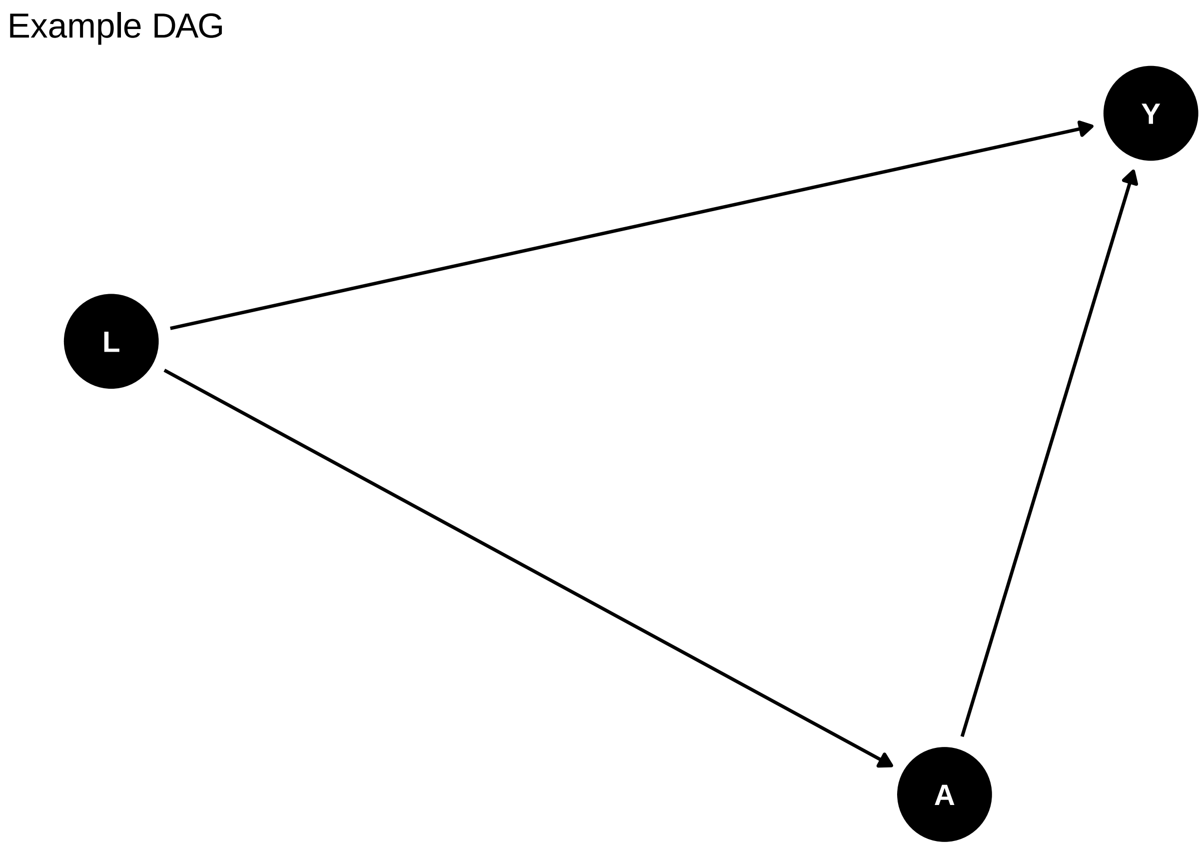
Key Points:

- Causal assumptions are assumed to be met. They are:
  - 1) Exchangeability
  - 2) Consistency
  - 3) Positivity
  - 4) No interference
- The parametric g-formula allows us to answer key public health questions, such as expected outcomes under various regimes. It allows us to make inference about effects of interventions or treatments
- The g-formula is useful for:
  - target parameters that do not come from a model
  - estimating population-level impacts
  - complex, longitudinal data
  - dynamic exposure and treatment regimes
  - potential outcomes models
- In order to get valid confidence intervals, we must use the bootstrap

The g-formula (briefly)

```r
exdag <- ggdag::dagify(A ~ L,
                        Y ~ L,
                        Y ~ A)

#set.seed(2)
set.seed(18)
ggdag::ggdag(exdag)+
  theme_void() +
  ggtitle("Example DAG")
```



1) Start with the distribution of observed data: $p(y, a, l) = p(y|a, l)p(a|l)p(l)$
2) Replace $p(a|l)$ with degenerate distribution $p_d(a|l)$ that is equal to 1 at $A = g$ and is 0 everywhere else
3) Marginalize over $p(l)$: $\int p(y|a, l)p_d(g|l)p(l)dl = \int p(y|g, l)p(l)dl$
4) $p(y|a, l)$ can be estimated via regression and $p(y|a, l, \beta)$ can be estimated by marginalizing over $p(l)$ by taking the sample average of predictions from that model.

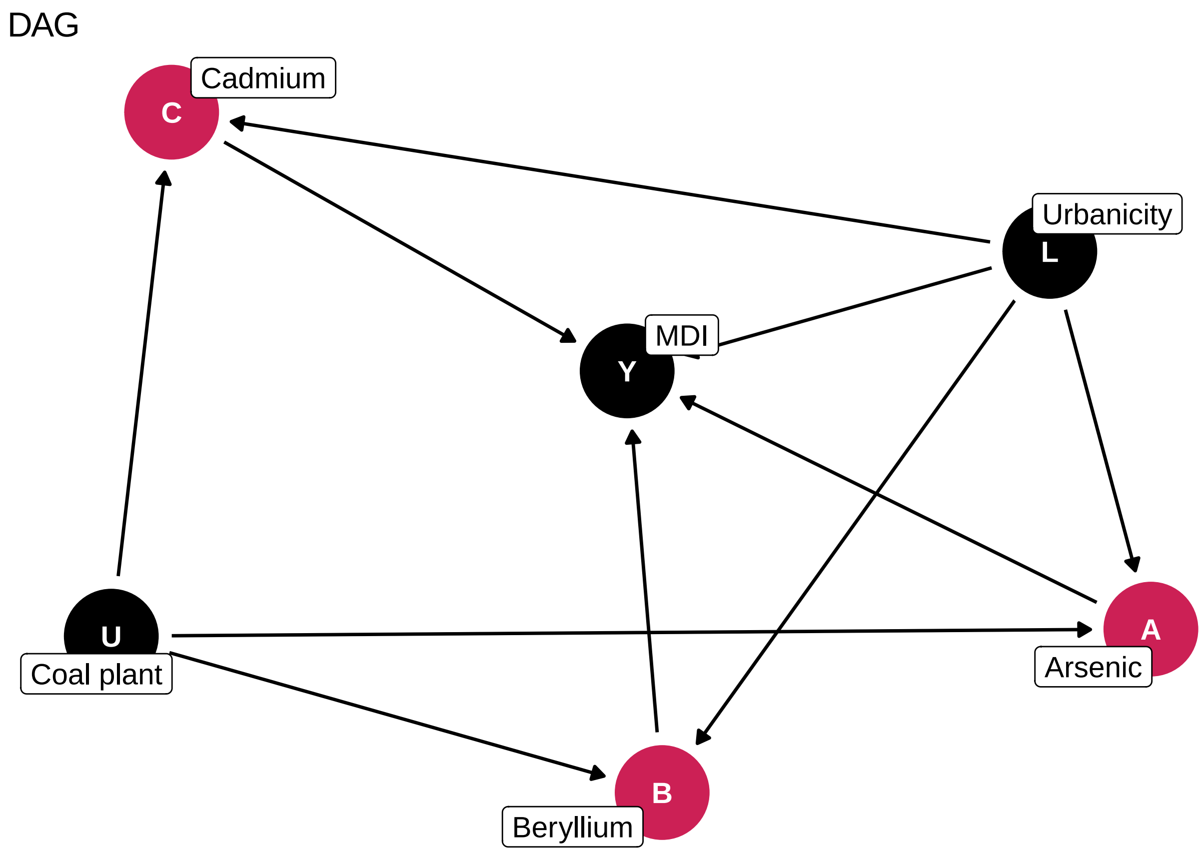
Coal-Fire Power Plant Example: Cross-Sectional G-Formula

Causal Contrasts

First, we load in the libraries needed for this analysis. If you have not installed these libraries yet, please be sure to do so using `install.packages("packagename")`.

Our data set is a (simulated) birth cohort of 3,961 individuals, followed up to 2 years of age in a U.S. city. The outcome of interest is the Mental Development Index (MDI) measured at age 2. The exposures of interest are 3 metals known to be emitted from coal-fired power plants: arsenic, beryllium, and cadmium. These exposures are measured as annual ambient levels from birth to age 1 via passive monitoring. The confounder of interest is urbanicity.

Our directed acyclic graph (DAG) of the research question is:



Next, we load in the data set `coalplant` and perform an initial exploratory data analysis:

```
coalplant <- read.csv("data/coalplant.csv")
str(coalplant)
```

```
## 'data.frame':    3961 obs. of  6 variables:
##  $ id        : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ as        : num  0.285 0.342 0.877 1.231 0.613 ...
##  $ be        : num  0.31 0.302 1.315 1.731 0.649 ...
##  $ cd        : num  0.313 0.245 1.495 2.218 0.649 ...
##  $ mdi       : num  110 74 140 81 85 104 90 79 69 97 ...
##  $ urbanicity: int  1 1 1 0 1 1 1 1 1 1 ...
```

By using the `str()` command, we identify the following 4 variables:

- `id`: a unique identifier for each individual
- `as`: arsenic levels
- `be`: beryllium levels
- `cd`: cadmium levels
- `mdi`: Mental Development Index (MDI) measured at age 2
- `urbanicity`: 0/1 indicator taking 1 if participant lived in an urban area or 0 otherwise

We first look at the summary statistics for the continuous variables and a table for the binary variable, `urbanicity`.

```r
summary(coalplant[, 2:5])
```

```
##        as               be               cd               mdi
##  Min.   :0.1464   Min.   :0.1057   Min.   :0.06395   Min.   : 60.00
##  1st Qu.:0.3656   1st Qu.:0.3650   1st Qu.:0.34904   1st Qu.: 87.00
##  Median :0.5995   Median :0.6073   Median :0.60318   Median : 97.00
##  Mean   :0.7208   Mean   :0.7561   Mean   :0.79716   Mean   : 97.32
##  3rd Qu.:1.0058   3rd Qu.:1.0054   3rd Qu.:1.03619   3rd Qu.:107.00
##  Max.   :2.2589   Max.   :4.3836   Max.   :6.51724   Max.   :140.00
```

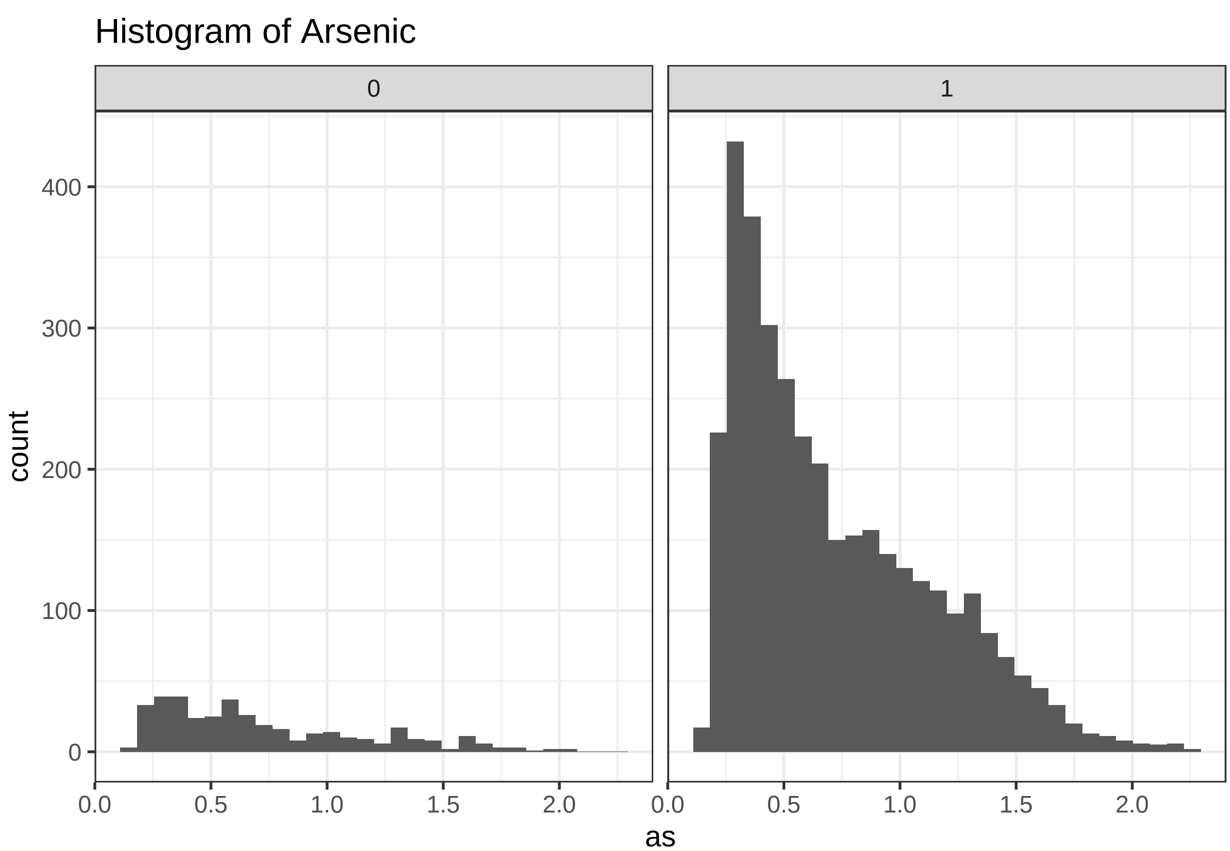
```r
table(coalplant$urbanicity)
```

```
##
##    0    1
##  385 3576
```

We observe similar distributions for the three exposures. The mean and median MDI scores are both around 97. Most participants live in an urban setting 3576.

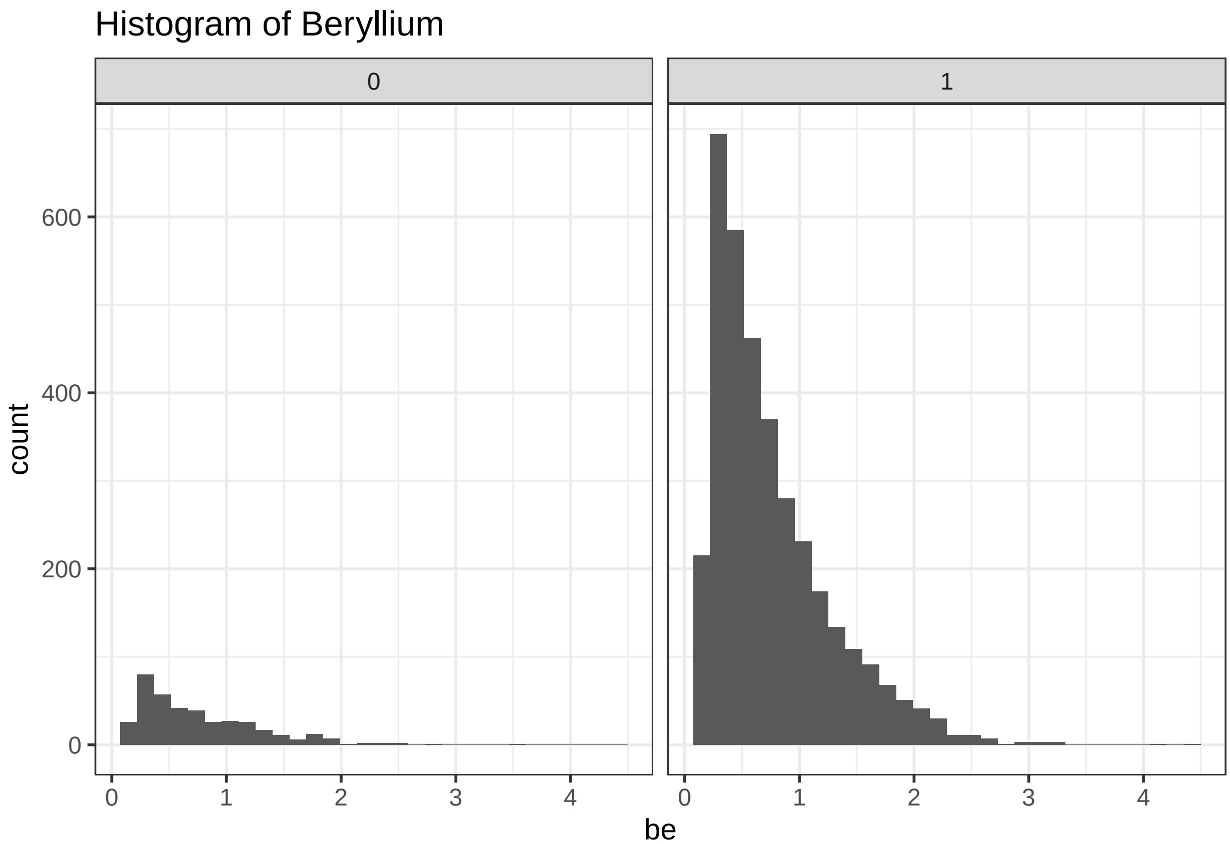
To visualize the data, we create histograms and facet by urbanicity status:

```r
ggplot(data=coalplant) +
  geom_histogram(aes(x=as)) +
  theme_bw() +
  ggtitle("Histogram of Arsenic") +
  facet_grid(.~ urbanicity)
```

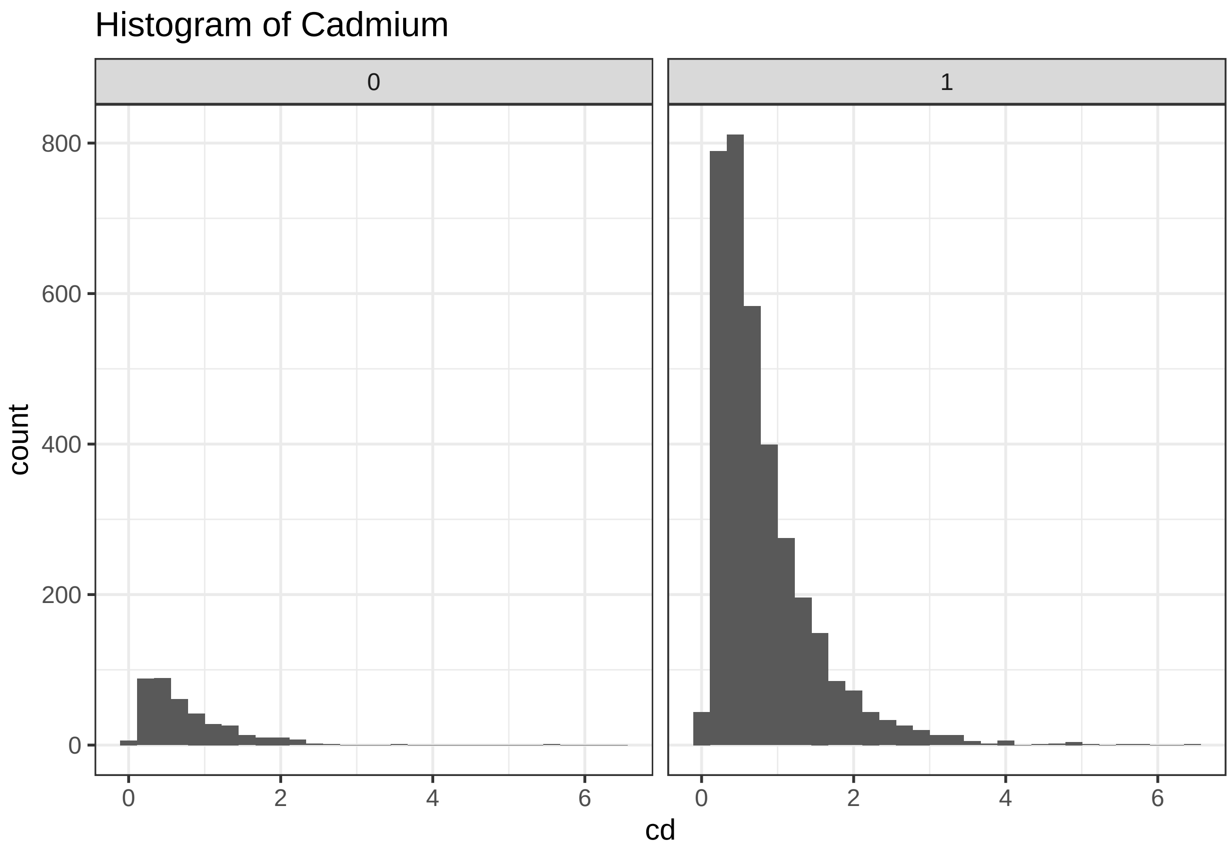



```r
ggplot(data=coalplant) +
  geom_histogram(aes(x=be)) +
```

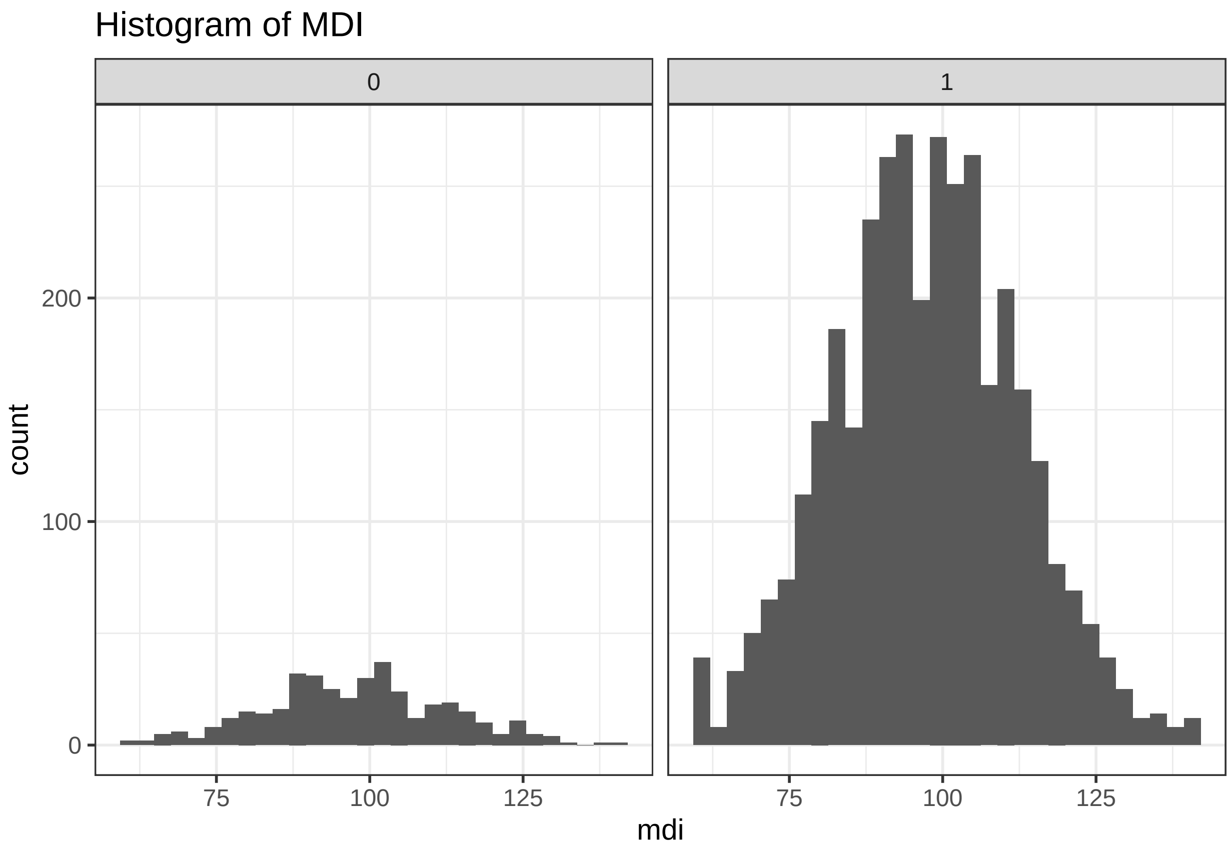
```
  theme_bw() +
  ggtitle("Histogram of Beryllium") +
  facet_grid(.~ urbanicity)
```




```
ggplot(data=coalplant) +
  geom_histogram(aes(x=cd)) +
  theme_bw() +
  ggtitle("Histogram of Cadmium") +
  facet_grid(.~ urbanicity)
```




```r
ggplot(data=coalplant) +
  geom_histogram(aes(x=mdi)) +
  theme_bw() +
  ggtitle("Histogram of MDI") +
  facet_grid(.~ urbanicity)
```



We observe large differences in distributions for the exposures by urbanicity. Intuitively, this makes sense based on background knowledge - living in a more urban area you exposure to co-pollutants will be different from those living in more rural or suburban areas. We examine summary statistics once more by urbanicity status using the `dplyr` functions. We pipe (`%>%`) the data set to a `filter()` function and subset based on urbanicity and then pipe to a `summary()` function:

```
coalplant %>%
  filter(urbanicity==1) %>%
  summary()
```

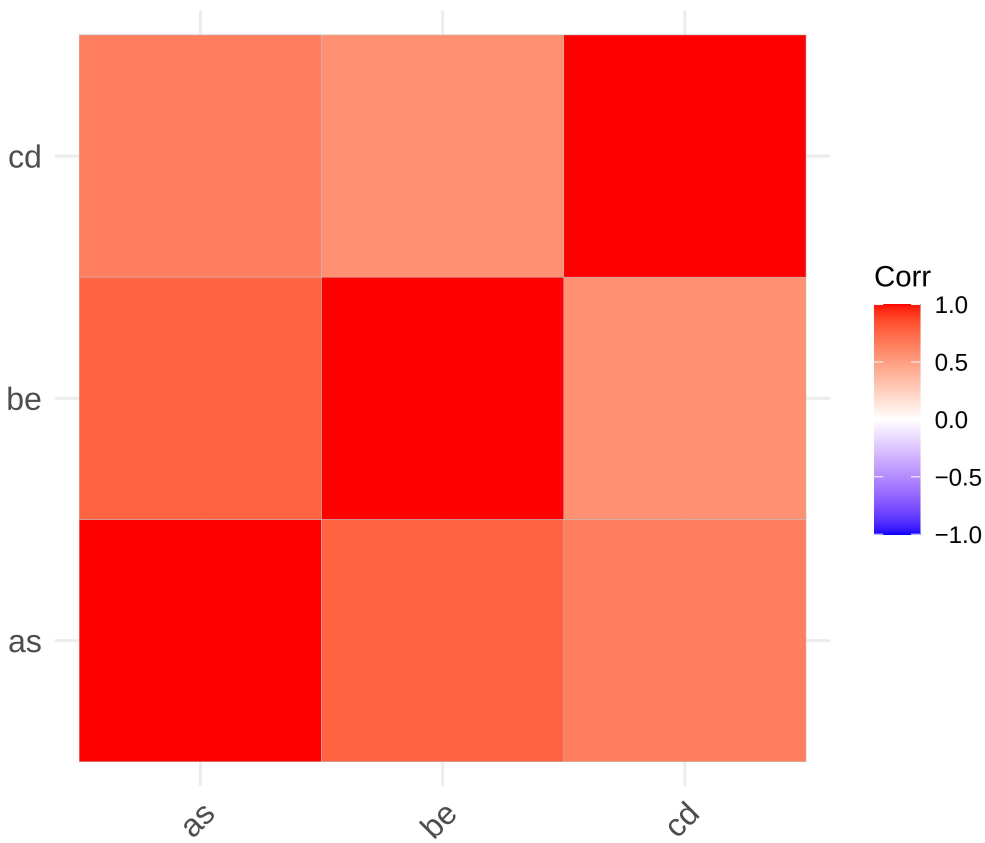
```
##        id               as               be               cd         
##  Min.   :   1.0   Min.   :0.1617   Min.   :0.1057   Min.   :0.06395  
##  1st Qu.: 984.8   1st Qu.:0.3658   1st Qu.:0.3660   1st Qu.:0.35008  
##  Median :1984.5   Median :0.5997   Median :0.6054   Median :0.60277  
##  Mean   :1979.6   Mean   :0.7196   Mean   :0.7554   Mean   :0.80107  
##  3rd Qu.:2973.2   3rd Qu.:1.0047   3rd Qu.:1.0006   3rd Qu.:1.03817  
##  Max.   :3960.0   Max.   :2.2589   Max.   :4.3836   Max.   :6.51724  
##       mdi           urbanicity
##  Min.   : 60.00   Min.   :1   
##  1st Qu.: 87.00   1st Qu.:1   
##  Median : 97.00   Median :1   
##  Mean   : 97.27   Mean   :1   
##  3rd Qu.:107.00   3rd Qu.:1   
##  Max.   :140.00   Max.   :1
```

```
coalplant %>%
  filter(urbanicity==0) %>%
  summary()
```

```
##        id             as               be               cd         
##  Min.   :   4   Min.   :0.1464   Min.   :0.1141   Min.   :0.07428  
##  1st Qu.:1081   1st Qu.:0.3575   1st Qu.:0.3568   1st Qu.:0.34105  
##  Median :1941   Median :0.5986   Median :0.6177   Median :0.60498  
##  Mean   :1994   Mean   :0.7325   Mean   :0.7625   Mean   :0.76083  
##  3rd Qu.:2942   3rd Qu.:1.0118   3rd Qu.:1.0687   3rd Qu.:1.02718  
##  Max.   :3961   Max.   :2.0288   Max.   :3.5153   Max.   :5.49568  
##       mdi           urbanicity
##  Min.   : 60.00   Min.   :0   
##  1st Qu.: 88.00   1st Qu.:0   
##  Median : 98.00   Median :0   
##  Mean   : 97.74   Mean   :0   
##  3rd Qu.:108.00   3rd Qu.:0   
##  Max.   :140.00   Max.   :0   
```

As part of the exploratory data analysis, we want to create a correlation plot. We first create a vector of co-pollutants or exposures. Then we calculate the correlations using the `cor()` function on the data set, only subsetting to the co-pollutant columns (`cor(coalplant[, exposures])`). Finally, we use the `ggcorrplot()` function from the `ggcorrplot` package to create the plot:

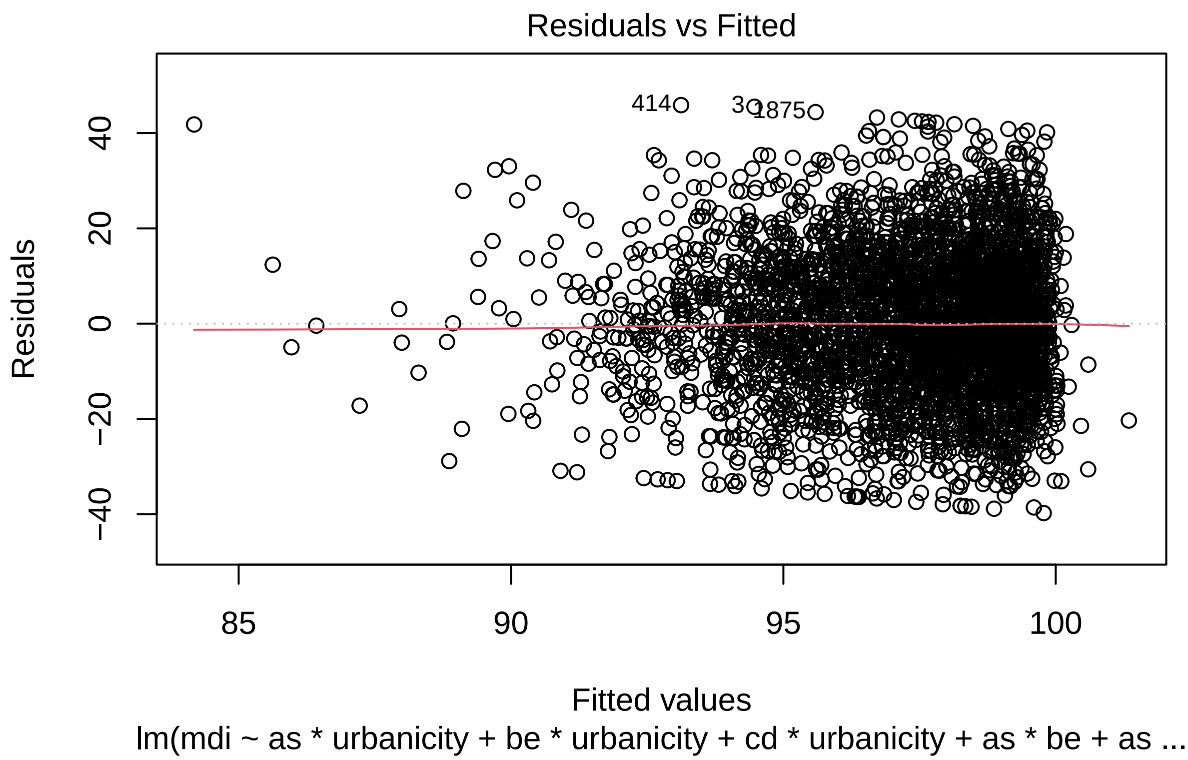
```r
exposures <- c("as", "be","cd")
corr <- cor(coalplant[, exposures])
ggcorrplot(corr)
```

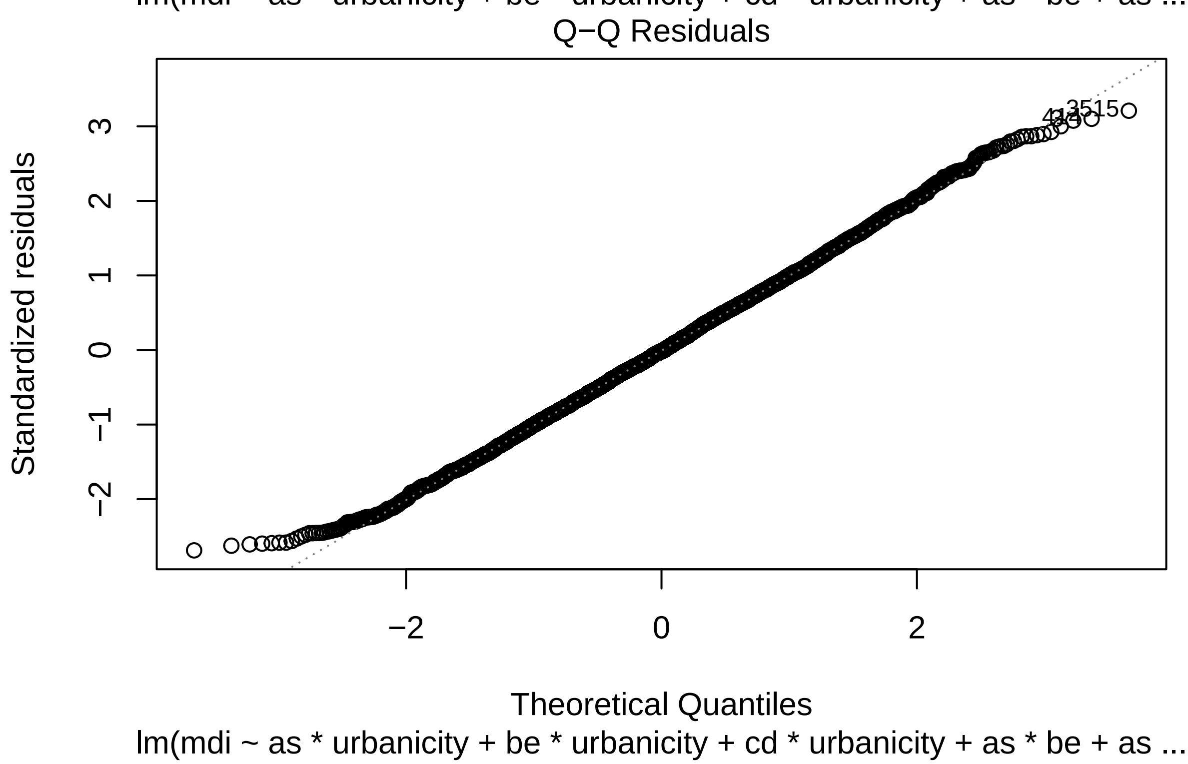


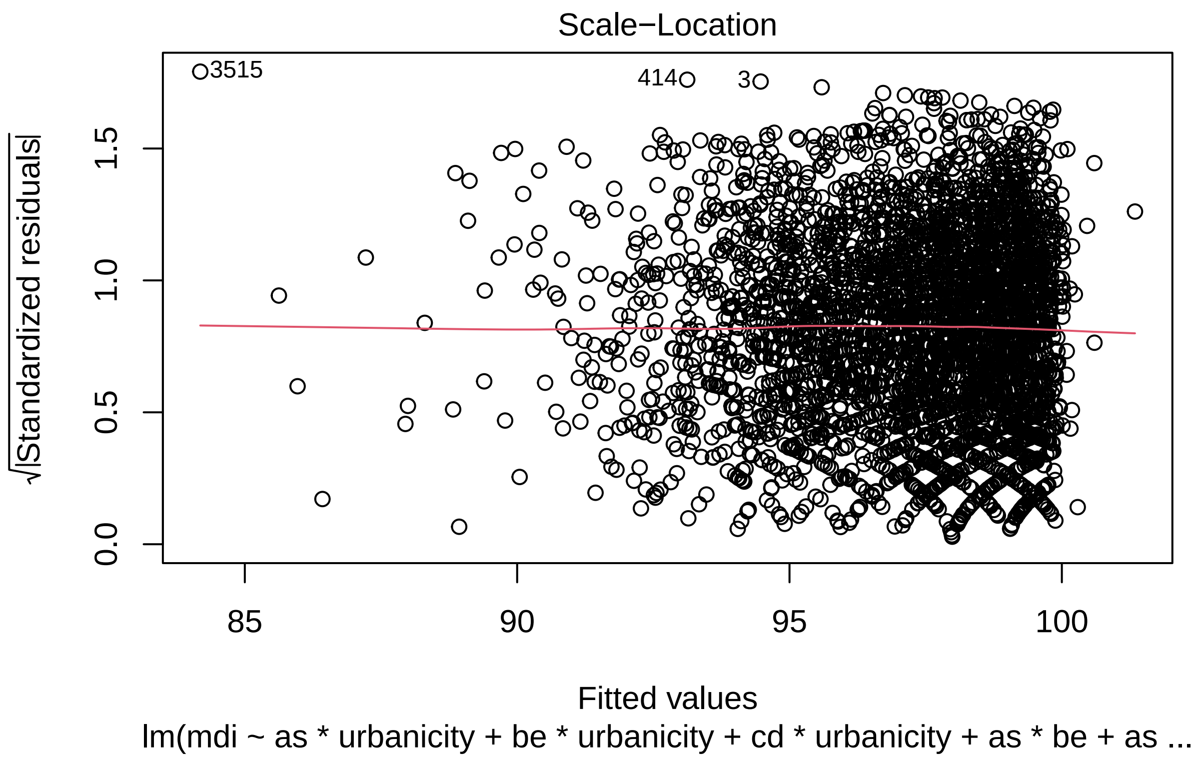
We also observe high-correlations among the three exposures.

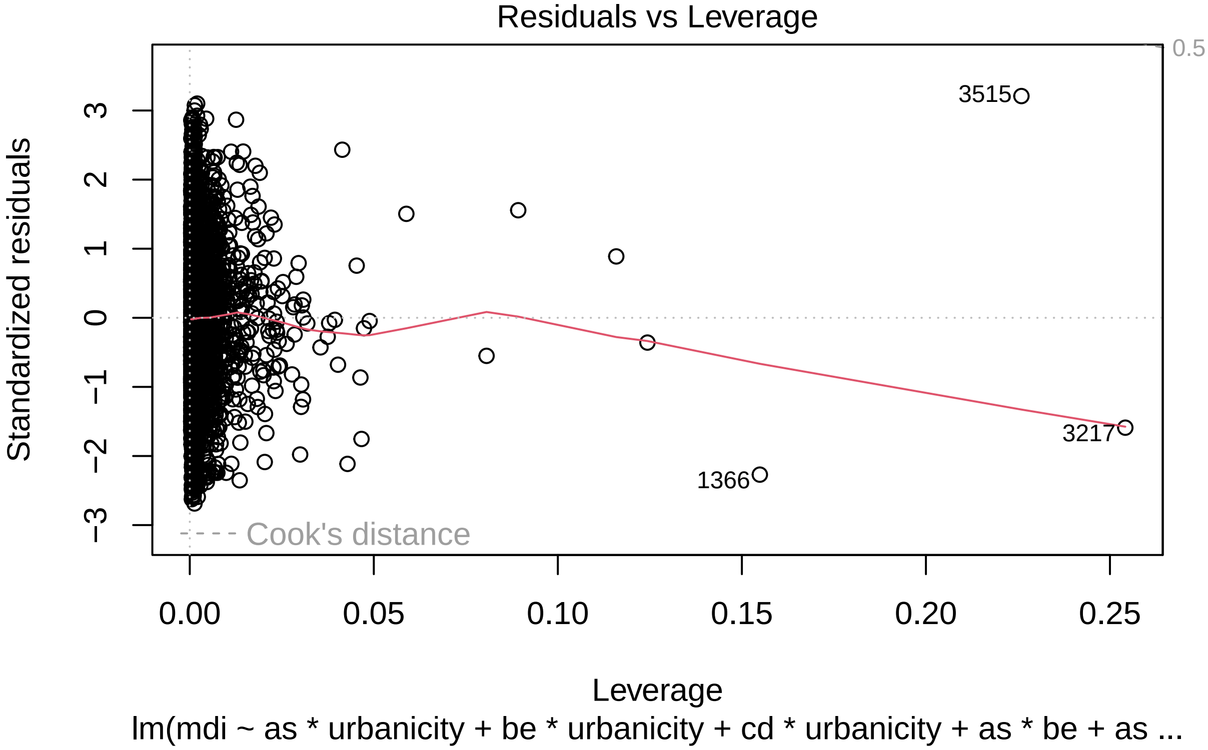
We will use a linear model using the `lm()` function in `R`. We include interactions between urbanicity and each of the three exposure pollutants, as well as interactions between the exposures as well. We look at model diagnostic plots using the `plot()` function and model summary statistics using the `summary()` function:

```r
mdimod = lm(mdi ~ as*urbanicity + be*urbanicity + cd*urbanicity + as*be + as*cd + be*cd, data=coalplant
plot(mdimod)
```










```r
summary(mdimod)
```

```
##
## Call:
## lm(formula = mdi ~ as * urbanicity + be * urbanicity + cd * urbanicity +
##     as * be + as * cd + be * cd, data = coalplant)
##
```

```
## Residuals:
##     Min      1Q  Median      3Q     Max
## -39.780 -10.121  -0.252   9.881  45.878
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)   100.90256    1.64323  61.405   <2e-16 ***
## as             -3.59765    3.29159  -1.093   0.2745
## urbanicity     -0.07433    1.55570  -0.048   0.9619
## be             -3.24839    2.89896  -1.121   0.2626
## cd              1.65248    2.16155   0.764   0.4446
## as:urbanicity   3.38784    3.15931   1.072   0.2836
## urbanicity:be  -0.16971    2.55152  -0.067   0.9470
## urbanicity:cd  -3.49894    1.79880  -1.945   0.0518 .
## as:be           0.97723    1.42876   0.684   0.4940
## as:cd          -0.09528    1.17825  -0.081   0.9356
## be:cd           0.04008    0.91097   0.044   0.9649
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 14.81 on 3950 degrees of freedom
## Multiple R-squared:  0.01889,    Adjusted R-squared:  0.01641
## F-statistic: 7.606 on 10 and 3950 DF,  p-value: 3.952e-12
```

Under the "natural course" (*nc*) (coal-fire power plant continues to operate), we can estimate the average joint effect of the three exposures on a given individual by taking the mean of the predictions:

```r
mean(predict(mdimod))
```

```
## [1] 97.31583
```

Under the natural course, where the coal-fire power plant continues to emit arsenic, beryllium, and cadmium, we expect the average MDI of a given child to be 97.316.

In other words, we are expecting no reduction in any of the exposures. More explicitly, we can use the `mutate()` and `predict()` functions to more explicitly calculate the same thing. First, we pipe (`%>%`) the data set to the `mutate()` function, which allows us to create new variables in the data set. We multiply each of the co-pollutants by the percent reduction expected. In the natural course case, we expect *no reduction*, or 100%-0%. We create this new data set called `dat_nc` for prediction and use the `predict()` function to use the `mdimod` model to predict on the new data set, `dat_nc`. Lastly, we take the `mean()` of those predictions to get our joint effect estimate under no reductions:

```r
#create new data set for prediction
dat_nc <- coalplant %>%
    mutate(as = as*(1-0.00), #no reduction in as
         be = be*(1-0.00), #no reduction in be
         cd = cd*(1-0.00)) #no reduction in cd

#predict mean MDI
(nc <- mean(predict(mdimod, newdata = dat_nc)))
```

```
## [1] 97.31583
```

Suppose that based on prior research, we know that 91% of arsenic, 96% of beryllium, and 45% of cadmium ambient levels come from a local coal-fired power plant. One **joint effect** of interest would be what would happen if we reduced the exposures by the proportion expected by decommissioning the power plant?

```r
#create new data set based on new intervention
dat_no_coal <- coalplant %>%
  mutate(as = as*(1-0.96),
         be = be*(1-0.91),
         cd = cd*(1-0.45))
#predict mean MDI
(no_coal <- mean(predict(mdimod, newdata = dat_no_coal)))
```

```
## [1] 99.92355
```

What would be the joint effect from only setting arsenic levels to 0?

```r
dat_no_as <- coalplant %>%
  mutate(as = as*0)
#predict mean MDI
(no_as <- mean(predict(mdimod, newdata = dat_no_as)))
```

```
## [1] 97.08229
```

What would be the joint effect from only setting beryllium levels to 0?

```r
dat_no_be <- coalplant %>%
  mutate(be = be*0)
#predict mean MDI
(no_be <- mean(predict(mdimod, newdata = dat_no_be)))
```

```
## [1] 99.15785
```

What would be the joint effect from only setting cadmium levels to 0?

```r
dat_no_cd <- coalplant %>%
  mutate(cd = cd*0)
#predict mean MDI
(no_cd <- mean(predict(mdimod, newdata = dat_no_cd)))
```

```
## [1] 98.56913
```

```r
c(naturalcourse = nc,
  no_coal = no_coal,
  no_arsenic = no_as,
  no_beryllium = no_be,
  no_cadmium = no_cd) %>%
  knitr::kable(col.names = c("Model","Predicted Mean MDI"))
```

Model	Predicted Mean MDI
naturalcourse	97.31583
no_coal	99.92355
no_arsenic	97.08229
no_beryllium	99.15785
no_cadmium	98.56913

Under *causal assumptions*, these estimates are equal to counterfactual means under the hypothetical means.

If we wanted to estimate the effect of decommissioning the power plant, we could subtract the mean expected MDI under the natural course from the mean expected MDI under the intervention:

$$\Delta M\bar{D}I = E(MDI^{as\times[1-0.96];be\times[1-0.91];cd\times[1-0.45]}) - E(MDI^{as;be;cd})$$

```
no_coal-nc
```

```
## [1] 2.607721
```

By decomissioning the coal-fire power plant, we can expect and increase of 2.608 MDI of 2 year olds.

Bootstrapping

When using the g-formula, bootstrapping is odten the only way to get valid confidence intervals, though it can be computationally-intensive. Briefly, bootstrapping uses resampling of the data to create an empirical distribution. From that, we can estimate the standard errors of the parameter of interest.

In order to obtain valid confidence intervals, we must bootstrap using the `boot` package. We will create a `function` to loop through the sampled data, estimate the joint effects under the different scenarios `R=500` times. Lastly, the `boot()` function will calculate the *bias* and *standard error* for us. We will then be able to calculate *bootstrapped confidence intervals*.

First, the function will take the arguments, `data` and `index`. Inside the function, we will sample the data set with replacement and create a new resampled data set called `mcsample`. We then will run the same linear model on the resampled data and store those results in the object, `bootmod`. Then, for each of the scenarios explored above, we create the new data sets and then take the mean predictions. In order to be able to replicate results, we use the `set.seed()` function before performing the bootstrapping

```
bootfunc <- function(data, index){
  mcsample = data[index,] #resampled data set
  bootmod = lm(mdi ~ as*urbanicity + be*urbanicity + cd*urbanicity + as*be + as*cd + be*cd, data=mcsampl

  #scenarios
  ###natural course
  dat_nc <- mcsample %>%
    mutate(as = as*(1-0.00), #no reduction in as
           be = be*(1-0.00), #no reduction in be
           cd = cd*(1-0.00)) #no reduction in cd
  nc <- mean(predict(bootmod, newdata = dat_nc))

  ###no coal-fire power plant
  dat_no_coal <- mcsample  %>%
  mutate(as = as*(1-0.96),
         be = be*(1-0.91),
         cd = cd*(1-0.45))
  no_coal <- mean(predict(bootmod, newdata = dat_no_coal))

  ###no arsenic
  dat_no_as <- mcsample  %>%
  mutate(as = as*0)
  no_as <- mean(predict(bootmod, newdata = dat_no_as))

  ###no beryllium
  dat_no_be <- mcsample %>%
  mutate(be = be*0)
  no_be <- mean(predict(bootmod, newdata = dat_no_be))

  ###no cadmium
  dat_no_cd <- mcsample  %>%
  mutate(cd = cd*0)
  no_cd <- mean(predict(bootmod, newdata = dat_no_cd))
```

```r
  #combine estiamtes together to export
  c(nc_mdi=nc, shutdown=no_coal-nc, attrmdi_cd=no_cd-nc, attrmdi_as=no_as-nc, attrmdi_be=no_be-nc)

}
set.seed(2)
bootsamples = boot(data=coalplant, statistic=bootfunc, R=500)
bootsamples
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = coalplant, statistic = bootfunc, R = 500)
##
##
## Bootstrap Statistics :
##       original       bias    std. error
## t1* 97.3158293 -0.003408230   0.2308952
## t2*  2.6077209  0.135491829   0.8707088
## t3*  1.2532988 -0.019145011   0.4770534
## t4* -0.2335398  0.043268799   0.7519097
## t5*  1.8420160 -0.004491634   0.5888464
```

We can look at the first 20 bootstrapped samples for each of the 5 scenarios by accessing the `t` object:

```r
bootsamples$t[1:20,]
```

```
##           [,1]     [,2]      [,3]        [,4]      [,5]
##  [1,] 97.66574 1.749377 1.0924607 -0.03985812 1.3574135
##  [2,] 97.38702 2.860693 1.2537652 -0.98013269 2.1568462
##  [3,] 97.52310 3.003203 1.2019686 -0.52773022 2.4650956
##  [4,] 97.15526 4.580490 0.7917178  0.43885199 2.4795414
##  [5,] 97.86973 2.796064 2.0635324  0.22111691 0.7969950
##  [6,] 97.60919 1.870814 0.9502873  0.10129677 1.5905147
##  [7,] 96.85610 3.310334 1.7113058  0.34905467 1.6996682
##  [8,] 96.93916 2.454644 0.8849235 -0.81974236 1.8002753
##  [9,] 97.62080 3.359343 1.2563761  0.42470447 1.4761066
## [10,] 96.97854 2.182605 1.1992109 -0.85064454 2.6017198
## [11,] 97.21257 3.900217 0.5401451 -0.31040268 2.4964559
## [12,] 97.20853 1.576355 1.8816106 -1.25022482 2.0651108
## [13,] 97.03307 2.300369 2.1177154  0.19205230 1.9079592
## [14,] 97.30725 3.067565 1.3164318  0.05608466 2.3632042
## [15,] 97.05226 3.366381 0.7793274 -0.76153589 2.3723907
## [16,] 97.63570 1.879225 1.0185941  0.19900097 1.6370528
## [17,] 97.12497 2.570180 0.9257450 -0.59776958 1.8073224
## [18,] 97.26332 2.760461 0.7190303 -0.63816341 2.5150579
## [19,] 97.09568 2.091166 1.2999673  0.64562866 0.9232546
## [20,] 97.58167 2.650061 1.6363082  0.61125944 1.1603666
```

To get the standard error we take the standard deviation of the bootstrapped samples using the `sd()` function. We can do this for each of the 5 scenarios by using `apply()` to apply the `sd()` function to each column:

```r
se = apply(bootsamples$t, 2, sd)
se
```

```
## [1] 0.2308952 0.8707088 0.4770534 0.7519097 0.5888464
```

Lastly, we print our table with the bootstrapped estimates and 95% confidence intervals by combining everything using `cbind()`:

```
print(cbind(estimate=bootsamples$t0, lowerCI=bootsamples$t0-1.96*se, upperCI=bootsamples$t0+1.96*se), 3)
```

```
##            estimate lowerCI upperCI
## nc_mdi       97.316  96.863   97.77
## shutdown      2.608   0.901    4.31
## attrmdi_cd    1.253   0.318    2.19
## attrmdi_as   -0.234  -1.707    1.24
## attrmdi_be    1.842   0.688    3.00
```

By shutting down the coal plant, we would expect an increase in mean MDI of 2 year olds to increase by 2.6 points, relative to doing nothing (natural course). This joint effect results in a larger difference in expected MDI score than completely eliminating any one of the co-pollutants on its own.

The parametric g-formula for a point exposure involves a standard regression model, but allows for more flexibility in inference.

Appendix - R Code

```
knitr::opts_chunk$set(echo = TRUE, warning = FALSE, message = FALSE)

#Load in libraries
library(ggplot2) #for plotting
library(dplyr) #for data cleaning
library(survival) #for survival analysis
library(boot) #for bootstrapping
library(ggcorrplot)
library(ggdag)
exdag <- ggdag::dagify(A ~ L,
                         Y ~ L,
                         Y ~ A)

#set.seed(2)
set.seed(18)
ggdag::ggdag(exdag)+
  theme_void() +
  ggtitle("Example DAG")

our_dag <- ggdag::dagify(B ~ U,
                         Y ~ B,
                         Y ~ A,
                         Y ~ C,
                         A ~ U,
                         C~ U,
                         Y ~ L,
                         A ~ L,
                         B ~ L,
                         C~ L,
                         labels = c("B" = "Beryllium",
                                    "U" = "Coal plant",
                                    "Y" = "MDI",
                                    "A" = "Arsenic",
                                    "C" = "Cadmium",
                                    "L" = "Urbanicity"))


#set.seed(2)
set.seed(18)
p <- ggdag::ggdag(our_dag,
             use_labels = "label") +
  theme_void() +
  ggtitle("DAG")
p$layers[[3]]$mapping <-
  aes(colour = c("Exposures", "Confounders")[as.numeric(name %in%  c("U","L","Y")) + 1])
set.seed(18)
p + scale_color_manual(values = c("black", "#cc2055")) +
  theme(legend.position = "none")


coalplant <- read.csv("data/coalplant.csv")
str(coalplant)
```

```
summary(coalplant[, 2:5])
table(coalplant$urbanicity)

ggplot(data=coalplant) +
  geom_histogram(aes(x=as)) +
  theme_bw() +
  ggtitle("Histogram of Arsenic") +
  facet_grid(.~ urbanicity)

ggplot(data=coalplant) +
  geom_histogram(aes(x=be)) +
  theme_bw() +
  ggtitle("Histogram of Beryllium") +
  facet_grid(.~ urbanicity)

ggplot(data=coalplant) +
  geom_histogram(aes(x=cd)) +
  theme_bw() +
  ggtitle("Histogram of Cadmium") +
  facet_grid(.~ urbanicity)


ggplot(data=coalplant) +
  geom_histogram(aes(x=mdi)) +
  theme_bw() +
  ggtitle("Histogram of MDI") +
  facet_grid(.~ urbanicity)

coalplant %>%
  filter(urbanicity==1) %>%
  summary()

coalplant %>%
  filter(urbanicity==0) %>%
  summary()


exposures <- c("as", "be","cd")
corr <- cor(coalplant[, exposures])
ggcorrplot(corr)

mdimod = lm(mdi ~ as*urbanicity + be*urbanicity + cd*urbanicity + as*be + as*cd + be*cd, data=coalplant
plot(mdimod)
summary(mdimod)


mean(predict(mdimod))
#create new data set for prediction
dat_nc <- coalplant %>%
    mutate(as = as*(1-0.00), #no reduction in as
         be = be*(1-0.00), #no reduction in be
         cd = cd*(1-0.00)) #no reduction in cd
```

```
#predict mean MDI
(nc <- mean(predict(mdimod, newdata = dat_nc)))

#create new data set based on new intervention
dat_no_coal <- coalplant %>%
  mutate(as = as*(1-0.96),
         be = be*(1-0.91),
         cd = cd*(1-0.45))
#predict mean MDI
(no_coal <- mean(predict(mdimod, newdata = dat_no_coal)))


dat_no_as <- coalplant %>%
  mutate(as = as*0)
#predict mean MDI
(no_as <- mean(predict(mdimod, newdata = dat_no_as)))
dat_no_be <- coalplant %>%
  mutate(be = be*0)
#predict mean MDI
(no_be <- mean(predict(mdimod, newdata = dat_no_be)))
dat_no_cd <- coalplant %>%
  mutate(cd = cd*0)
#predict mean MDI
(no_cd <- mean(predict(mdimod, newdata = dat_no_cd)))
c(naturalcourse = nc,
  no_coal = no_coal,
  no_arsenic = no_as,
  no_beryllium = no_be,
  no_cadmium = no_cd) %>%
  knitr::kable(col.names = c("Model","Predicted Mean MDI"))

no_coal-nc

bootfunc <- function(data, index){
  mcsample = data[index,] #resampled data set
  bootmod = lm(mdi ~ as*urbanicity + be*urbanicity + cd*urbanicity + as*be + as*cd + be*cd, data=mcsampl

  #scenarios
  ###natural course
  dat_nc <- mcsample %>%
    mutate(as = as*(1-0.00), #no reduction in as
           be = be*(1-0.00), #no reduction in be
           cd = cd*(1-0.00)) #no reduction in cd
  nc <- mean(predict(bootmod, newdata = dat_nc))

  ###no coal-fire power plant
  dat_no_coal <- mcsample  %>%
  mutate(as = as*(1-0.96),
         be = be*(1-0.91),
         cd = cd*(1-0.45))
  no_coal <- mean(predict(bootmod, newdata = dat_no_coal))

  ###no arsenic
```

```
  dat_no_as <- mcsample  %>%
  mutate(as = as*0)
  no_as <- mean(predict(bootmod, newdata = dat_no_as))

  ###no beryllium
  dat_no_be <- mcsample %>%
  mutate(be = be*0)
  no_be <- mean(predict(bootmod, newdata = dat_no_be))

  ###no cadmium
  dat_no_cd <- mcsample  %>%
  mutate(cd = cd*0)
  no_cd <- mean(predict(bootmod, newdata = dat_no_cd))

  #combine estiamtes together to export
  c(nc_mdi=nc, shutdown=no_coal-nc, attrmdi_cd=no_cd-nc, attrmdi_as=no_as-nc, attrmdi_be=no_be-nc)

}
set.seed(2)
bootsamples = boot(data=coalplant, statistic=bootfunc, R=500)
bootsamples

bootsamples$t[1:20,]
se = apply(bootsamples$t, 2, sd)
se
print(cbind(estimate=bootsamples$t0, lowerCI=bootsamples$t0-1.96*se, upperCI=bootsamples$t0+1.96*se), 3
knitr::purl(input = "cross_sectional_gformula.Rmd", output = "cross_sectional_gformula.R",documentation
```

```
knitr::purl(input = "cross_sectional_gformula.Rmd", output = "cross_sectional_gformula.R",documentation
```

```
## [1] "cross_sectional_gformula.R"
```