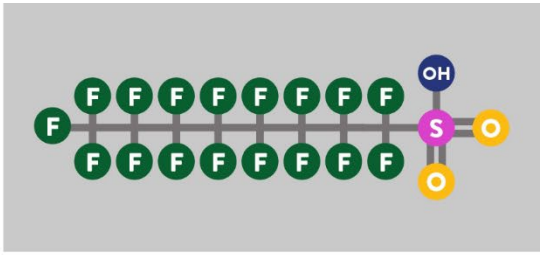
Water quality standards

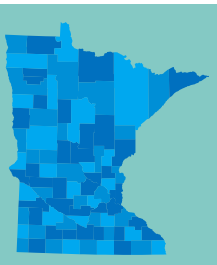
May 2024

Human Health Protective Water Quality Criteria for Per- and Polyfluoroalkyl Substances (PFAS) in Mississippi River, Miles 820 to 812

Supporting Data



MINNESOTA POLLUTION CONTROL AGENCY



Authors

Laura Lyle (MPCA, Water Quality Standards Unit)

Summer Streets (MPCA, Water Quality Standards Unit)

Contributors/acknowledgements

Emily Brault (MPCA, Effluent Limits Unit)

Scott Kyser (MPCA, Effluent Limits Unit)

Photos

Minnesota Department of Natural Resources

Minnesota Pollution Control

Minnesota Pollution Control Agency

520 Lafayette Road North | Saint Paul, MN 55155-4194 |

651-296-6300 | 800-657-3864 | Or use your preferred relay service. | Info.pca@state.mn.us

This report is available in alternative formats upon request, and online at www.pca.state.mn.us.

Document number: wq-s6-69b

Supporting data overview

In order to determine PFAS site-specific water quality criteria (SSC) for the Mississippi River, Miles 820 to 812, the Minnesota Pollution Control Agency (MPCA) calculated bioaccumulation factors (BAFs) with PFAS surface water and fish measurements from the Mississippi River near Cottage Grove, MN. BAF calculation requires geometric means per PFAS compound for surface water and fish. Since some measurements are less than the reported detection limit of the analytical method, the MPCA applied non-detection analyses to the data.

Prior to calculating PFAS surface water and fish means and BAFs, we processed the data. We converted data to be in a consistent unit (ng/L and ng/g), removed measurements from quality control samples, retained data resulting from a single analytical method (method 537.1 and ETS-8-045 for surface water and fish samples, respectively), and addressed duplicate data.

We calculated PFAS means using six different non-detection approaches and PFAS geometric means with five different non-detection methods. The six methods are described in Table 1. We did not calculate means via the Kaplan-Meier and Regression on Order Statistics (ROS) for PFAS compounds when certain criteria were not met: (1) two or fewer values in the given dataset were detected or (2) two or fewer values in the given dataset were not detected. Application of the Kaplan-Meier or ROS methods is not appropriate when datasets are composed of nearly all detected or non-detected measurements. We did not determine geometric means via the Kaplan-Meier technique since it is not possible with negative values, which results when using log-transformed data in geometric mean calculations. All data processing and statistical calculations were performed in R statistical software (2024).[1]

This Supporting Data package includes R scripts for surface water and fish and raw data files for surface water and fish that we used in the derivation of SSC in the Mississippi River Miles 820 to 812.

[1]R Core Team (2024). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

R script for non-detect analysis in surface water

```
#################
### OBJECTIVE

### import surface water PFAS data and process it (e.g., address non-detects)

###################
### SET OBJECTS

directory <- "X:\\Agency_Files\\Water\\Standards\\R_Data_Analyses\\PFAS_EQuIS"

#####################
### SET DIRECTORY

setwd(directory)
getwd()

############################################
### INSTALL AND/OR CALL DESIRED PACKAGES

### Install the below packages if you haven't yet. You only need to install them once.
#install.packages(c('dplyr','readr'))
library(dplyr)
library(readr)
library(stringr)
library(sf)
library(lubridate)
library(tidyverse)
library(EnvStats)
library(NADA)
library(ggplot2)
```

```
#library(insight)

####################
### BRING IN DATA

water <-
read_csv("X:\\Agency_Files\\Water\\Standards\\R_Data_Analyses\\PFAS_EQuIS\\Water_2.4and3.1.csv"
)

#####################################################################
### ADDRESS MULTIPLE UNITS AND MULTIPLE SAME STATION ENTRIES

unique(water$RESULT_UNIT)
### convert ng/ml to ng/L

### note: issue with same station having multiple description or counties in EQuIS, have same
coordinates - delete attributes to address

options(scipen = 999)
water_ed <- water %>%
 dplyr::select(-c(LOC_DESC,CITY,COUNTY_NAME)) %>%
 dplyr::mutate(RESULT_NUMERIC_ADJ=case_when(RESULT_UNIT=='ng/mL' ~ RESULT_NUMERIC*1000,
                                RESULT_UNIT=='ng/L' ~ RESULT_NUMERIC),
        RESULT_UNIT_ADJ=case_when(RESULT_UNIT=='ng/mL' ~ 'ng/L',
                              RESULT_UNIT=='ng/L' ~ RESULT_UNIT),
        DETECTION_LIMIT_ADJ=case_when(DETECTION_LIMIT_UNIT=='ng/mL' ~
REPORTING_DETECTION_LIMIT*1000,
                                  DETECTION_LIMIT_UNIT=='ng/L' ~ REPORTING_DETECTION_LIMIT),
        DETECTION_LIMIT_UNIT_ADJ=case_when(DETECTION_LIMIT_UNIT=='ng/mL' ~ 'ng/L',
                                       DETECTION_LIMIT_UNIT=='ng/L' ~ DETECTION_LIMIT_UNIT),
        id=paste(SYS_LOC_CODE,SHORT_NAME,SAMPLE_DATE,RESULT_UNIT_ADJ,sep='_')) %>%
 dplyr::filter(ANALYTIC_METHOD!='ETS-8-044',
        ) %>%
 distinct()
```

```
water_ed$id[duplicated(water_ed$id)] ### no duplicates

##########################
### LOOK AT SAMPLE TYPE

unique(water_ed$SAMPLE_TYPE_CODE) ### all sample - no QC in this dataset

##########################
### ADDRESS NON-DETECTS

#undetected <- water %>%
#  dplyr::filter(is.na(RESULT_NUMERIC))
#detected <- water %>%
#  dplyr::filter(!is.na(RESULT_NUMERIC))

water_nd_eds <- water_ed %>%
  dplyr::mutate(CENSORED=as.logical(case_when(is.na(RESULT_NUMERIC_ADJ) ~ 'TRUE', ### if 'TRUE',
then need to censor data
                                              RESULT_NUMERIC_ADJ < DETECTION_LIMIT_ADJ ~ 'TRUE', ### if 'TRUE', then
need to censor data
                                              TRUE ~ 'FALSE')),
         RESULT_SUB_HALF_DL=ifelse(CENSORED,DETECTION_LIMIT_ADJ*0.5,RESULT_NUMERIC_ADJ),
         RESULT_SUB_DL=ifelse(CENSORED,DETECTION_LIMIT_ADJ,RESULT_NUMERIC_ADJ),
         RESULT_SUB_ZERO=ifelse(CENSORED,0,RESULT_NUMERIC_ADJ))

Means_byCompound <- water_nd_eds %>%
  dplyr::group_by(SHORT_NAME,RESULT_UNIT_ADJ) %>%
  dplyr::summarise(CENSOR_Adequate=case_when(sum(CENSORED)>2 &
n_distinct(RESULT_SUB_DL[CENSORED!=TRUE])>2 ~ TRUE,
                                             TRUE ~ FALSE),
            TOTAL_COUNT=n(),
            NOT_DETECTED_COUNT=sum(is.na(RESULT_NUMERIC_ADJ)),
            PERCENT_ND=round((NOT_DETECTED_COUNT/TOTAL_COUNT)*100,digits=1),
            Raw_Mean=round(mean(RESULT_NUMERIC_ADJ,na.rm=T),digits=3),
            Raw_SD=round(sd(RESULT_NUMERIC_ADJ,na.rm=T),digits=3),
```

```
            SubHalfDL_Mean=round(mean(RESULT_SUB_HALF_DL),digits=3),
            SubHalfDL_SD=round(sd(RESULT_SUB_HALF_DL),digits=3),
            SubDL_Mean=round(mean(RESULT_SUB_DL),digits=3),
            SubDL_SD=round(sd(RESULT_SUB_DL),digits=3),
            SubZero_Mean=round(mean(RESULT_SUB_ZERO),digits=3),
            SubZero_SD=round(sd(RESULT_SUB_ZERO),digits=3),

KaplanMeier_Mean=round(ifelse(CENSOR_Adequate,enparCensored(RESULT_SUB_DL,CENSORED)$parameters[[1]],NA),digits=3),

KaplanMeier_SD=round(ifelse(CENSOR_Adequate,enparCensored(RESULT_SUB_DL,CENSORED)$parameters[[2]],NA),digits=3),

ROS_Mean=round(ifelse(CENSOR_Adequate,enormCensored(RESULT_SUB_DL,CENSORED,method="rROS",ci=F,lb.impute=min(DETECTION_LIMIT_ADJ)*0.10,ub.impute=max(DETECTION_LIMIT_ADJ))$parameters[[1]],NA),digits=3),

ROS_SD=round(ifelse(CENSOR_Adequate,enormCensored(RESULT_SUB_DL,CENSORED,method="rROS",ci=F,lb.impute=min(DETECTION_LIMIT_ADJ)*0.10,ub.impute=max(DETECTION_LIMIT_ADJ))$parameters[[2]],NA),digits=3)) %>%
  ungroup() %>%
  dplyr::mutate(across(Raw_Mean,~replace(.,is.nan(.),NA))) %>%
  dplyr::arrange(SHORT_NAME)

water_nd_forNADA <- water_nd_eds
NADA_water <-
cenfit(Cen(water_nd_forNADA$RESULT_SUB_DL,water_nd_forNADA$CENSORED)~water_nd_forNADA$SHORT_NAME)
print(NADA_water)

### note: you can't use method "mle" because the data needs a lower boundary set; you can set boundaries with method "impute.w.mle" but this doesn't work with multiple detection limits as is the case with these data; using method "mle" won't work for these data
#
RESULT_MLE_MEAN=ifelse(CENSOR_Adequate,enormCensored(RESULT_SUB_DL,CENSORED,method="mle")$parameters[[1]],NA)

GeoMeans_byCompound <- water_nd_eds %>%
  dplyr::group_by(SHORT_NAME,RESULT_UNIT_ADJ) %>%
```

```
  dplyr::summarise(CENSOR_Adequate=case_when(sum(CENSORED)>2 &
n_distinct(RESULT_SUB_DL[CENSORED!=TRUE])>2 ~ TRUE,
                                   TRUE ~ FALSE),
            TOTAL_COUNT=n(),
            NOT_DETECTED_COUNT=sum(is.na(RESULT_NUMERIC_ADJ)),
            PERCENT_ND=round((NOT_DETECTED_COUNT/TOTAL_COUNT)*100,digits=1),
            Raw_Mean=round(exp(mean(log(RESULT_NUMERIC_ADJ),na.rm=T)),digits=3),
            Raw_SD=round(exp(sd(log(RESULT_NUMERIC_ADJ),na.rm=T)),digits=3),
            SubHalfDL_Mean=round(exp(mean(log(RESULT_SUB_HALF_DL))),digits=3),
            SubHalfDL_SD=round(exp(sd(log(RESULT_SUB_HALF_DL))),digits=3),
            SubDL_Mean=round(exp(mean(log(RESULT_SUB_DL))),digits=3),
            SubDL_SD=round(exp(sd(log(RESULT_SUB_DL))),digits=3),
            SubZero_Mean=round(exp(mean(log(RESULT_SUB_ZERO))),digits=3),
            SubZero_SD=round(exp(sd(log(RESULT_SUB_ZERO))),digits=3),

ROS_Mean=round(exp(ifelse(CENSOR_Adequate,enormCensored(log(RESULT_SUB_DL),CENSORED,meth
od="rROS",ci=F,lb.impute=log(min(DETECTION_LIMIT_ADJ))*0.10,ub.impute=max(log(DETECTION_LIMIT
_ADJ)))$parameters[[1]],NA)),digits=3),

ROS_SD=round(exp(ifelse(CENSOR_Adequate,enormCensored(log(RESULT_SUB_DL),CENSORED,method
="rROS",ci=F,lb.impute=log(min(DETECTION_LIMIT_ADJ))*0.10,ub.impute=max(log(DETECTION_LIMIT_
ADJ)))$parameters[[2]],NA)),digits=3)) %>%
  ungroup()  %>%
  dplyr::mutate(across(Raw_Mean, ~replace(.,is.nan(.),NA)),
           across(SubZero_SD, ~replace(.,is.nan(.),NA))) %>%
  dplyr::arrange(SHORT_NAME)

###################
### EXPORT DATA

Means_byCompound_print <- Means_byCompound %>%

dplyr::rename(Compound=SHORT_NAME,Unit=RESULT_UNIT_ADJ,Censored_Data_Adequate=CENSOR_
Adequate,Total_Count=TOTAL_COUNT,Not_Detected_Count=NOT_DETECTED_COUNT,Percent_ND=PER
CENT_ND)
getwd()
write_csv(Means_byCompound_print,'MeanStats_SurfaceWater.csv')
```

```
GeoMeans_byCompound_print <- GeoMeans_byCompound %>%

dplyr::rename(Compound=SHORT_NAME,Unit=RESULT_UNIT_ADJ,Censored_Data_Adequate=CENSOR_
Adequate,Total_Count=TOTAL_COUNT,Not_Detected_Count=NOT_DETECTED_COUNT,Percent_ND=PER
CENT_ND)
getwd()
write_csv(GeoMeans_byCompound_print,'GeoMeanStats_SurfaceWater.csv')

#################
### MAKE PLOT

### arithmetic means

lapply(Means_byCompound,class)
names(Means_byCompound)

forplot <- Means_byCompound %>%
  dplyr::select(SHORT_NAME,Raw_Mean:ROS_SD)

pivot1 <- forplot %>%
  pivot_longer(!SHORT_NAME,
               names_to="Type",
               values_to="Value")

pivot2 <- pivot1 %>%
  separate_wider_delim(Type,delim="_",names=c("METHOD","STATISTIC"))

pivot3 <- pivot2 %>%
  pivot_wider(names_from="STATISTIC",
              values_from="Value"
              ) %>%
  mutate(METHOD=case_when(METHOD=='KaplanMeier' ~ 'Kaplan-Meier',
                          METHOD=='SubDL' ~ 'Detection Limit',
                          METHOD=='SubHalfDL' ~ 'Half Detection Limit',
```

```
                METHOD=='SubZero' ~ 'Zero',
                TRUE ~ METHOD))

results_by_method <- ggplot(pivot3,aes(x=METHOD,y=Mean)) +
 geom_point() +
 facet_wrap(~SHORT_NAME,scales="free") +
 theme(axis.text.x=element_text(angle=90,vjust=0.5,hjust=1)) +
 geom_errorbar(aes(ymin=ifelse(Mean-SD<0,0,Mean-SD),ymax=Mean+SD,width=0.2)) +
 xlab("Non-Detection Method") +
 ylab("Mean +/- SD (ng/L)")
getwd()
ggsave("Water_PFAS_by_NDMethod.tiff",path=directory,width=5,height=5,device='tiff',dpi=700)

### geometic means

lapply(GeoMeans_byCompound,class)
names(GeoMeans_byCompound)

Geo_forplot <- GeoMeans_byCompound %>%
 dplyr::select(SHORT_NAME,Raw_Mean:ROS_SD)

Geo_pivot1 <- Geo_forplot %>%
 pivot_longer(!SHORT_NAME,
          names_to="Type",
          values_to="Value")

Geo_pivot2 <- Geo_pivot1 %>%
 separate_wider_delim(Type,delim="_",names=c("METHOD","STATISTIC"))

Geo_pivot3 <- Geo_pivot2 %>%
 pivot_wider(names_from="STATISTIC",
        values_from="Value"
 ) %>%
 mutate(METHOD=case_when(METHOD=='KaplanMeier' ~ 'Kaplan-Meier',
```

```
                METHOD=='SubDL' ~ 'Detection Limit',
                METHOD=='SubHalfDL' ~ 'Half Detection Limit',
                METHOD=='SubZero' ~ 'Zero',
                TRUE ~ METHOD)) %>%
  dplyr::filter(METHOD!='Kaplan-Meier',
           METHOD!='Zero')

geo_results_by_method <- ggplot(Geo_pivot3,aes(x=METHOD,y=Mean)) +
  geom_point() +
  facet_wrap(~SHORT_NAME,scales="free") +
  theme(axis.text.x=element_text(angle=90,vjust=0.5,hjust=1)) +
  geom_errorbar(aes(ymin=ifelse(Mean-SD<0,0,Mean-SD),ymax=Mean+SD,width=0.2)) +
  xlab("Non-Detection Method") +
  ylab("Geometric Mean +/- SD (ng/L)")
getwd()
ggsave("Water_PFAS_by_NDMethod_Geo.tiff",path=directory,width=5,height=5,device='tiff',dpi=700)
```

R script for non-detect analysis in fish tissue

```
#################
### OBJECTIVE

### import fish PFAS data and process it (e.g., address non-detects)

###################
### SET OBJECTS

directory <- "X:\\Agency_Files\\Water\\Standards\\R_Data_Analyses\\PFAS_EQuIS"

#####################
### SET DIRECTORY
```

```
setwd(directory)
getwd()

##########################################
### INSTALL AND/OR CALL DESIRED PACKAGES

### Install the below packages if you haven't yet. You only need to install them once.
#install.packages(c('dplyr','readr'))
library(dplyr)
library(readr)
library(stringr)
library(sf)
library(lubridate)
library(tidyverse)
library(EnvStats)
library(NADA)
library(viridis)
library(ggbreak)
#library(insight)

####################
### BRING IN DATA

fish <-
read_csv("X:\\Agency_Files\\Water\\Standards\\R_Data_Analyses\\PFAS_EQuIS\\Filet_2.4and3.1.csv")

########################################
### ADD GROUP AND LOOK FOR DUPLICATES

names(fish)
unique(fish$TAXON)
unique(fish$SPECIES)
unique(fish$TISSUE_ANATOMY) ### fillet only
unique(fish$RESULT_UNIT) ### all in ng/g
```

```
unique(fish$ANALYTIC_METHOD) ### all used ETS-8-045
unique(fish$SAMPLE_TYPE_CODE) ### all samples, no QC
sort(unique(fish$SAMPLE_NAME))
unique(str_detect(unique(fish$SAMPLE_NAME),'W')) ### no W's which are actually whole fish

options(scipen = 999)

fish_ed <- fish %>%
  dplyr::select(-c(NAME,LOC_DESC,CITY,COUNTY_NAME)) %>%
  dplyr::mutate(TAXON_2=case_when(TAXON=='Sander vitreus' ~ 'Sander Genus',
                                  TAXON=='Sander canadensis' ~ 'Sander Genus',
                                  TRUE ~ TAXON),
                id=paste(SYS_LOC_CODE,SAMPLE_NAME,SHORT_NAME,SAMPLE_DATE,sep='_')) %>%
  distinct()

fish_ed$id[duplicated(fish_ed$id)] ### no duplicates

############################
### FILL IN NON-DETECTS

#undetected <- fish_ed %>%
#  dplyr::filter(is.na(RESULT_NUMERIC))
#detected <- fish_ed %>%
#  dplyr::filter(!is.na(RESULT_NUMERIC))

fish_nd_eds <- fish_ed %>%
  dplyr::mutate(CENSORED=as.logical(case_when(is.na(RESULT_NUMERIC) ~ 'TRUE', ### if 'TRUE', then need to censor data
                                              RESULT_NUMERIC < REPORTING_DETECTION_LIMIT ~ 'TRUE', ### if 'TRUE', then need to censor data
                                              TRUE ~ 'FALSE')),

RESULT_SUB_HALF_DL=ifelse(CENSORED,REPORTING_DETECTION_LIMIT*0.5,RESULT_NUMERIC),
                RESULT_SUB_DL=ifelse(CENSORED,REPORTING_DETECTION_LIMIT,RESULT_NUMERIC),
                RESULT_SUB_ZERO=ifelse(CENSORED,0,RESULT_NUMERIC))
```

```
###################################
### CALCULATE MEANS BY FISH TAXA


names(fish_nd_eds)
GeoMeans_byTaxa <- fish_nd_eds %>%
  dplyr::group_by(TAXON_2,SHORT_NAME,RESULT_UNIT) %>%
  dplyr::summarise(CENSOR_Adequate=case_when(sum(CENSORED)>2 &
n_distinct(RESULT_SUB_DL[CENSORED!=TRUE])>2 ~ TRUE,
                                             TRUE ~ FALSE),
                   TOTAL_COUNT=n(),
                   NOT_DETECTED_COUNT=sum(is.na(RESULT_NUMERIC)),
                   PERCENT_ND=round((NOT_DETECTED_COUNT/TOTAL_COUNT)*100,digits=1),
                   Raw_Mean=round(exp(mean(log(RESULT_NUMERIC),na.rm=T)),digits=3),
                   Raw_SD=round(exp(sd(log(RESULT_NUMERIC),na.rm=T)),digits=3),
                   SubHalfDL_Mean=round(exp(mean(log(RESULT_SUB_HALF_DL))),digits=3),
                   SubHalfDL_SD=round(exp(sd(log(RESULT_SUB_HALF_DL))),digits=3),
                   SubDL_Mean=round(exp(mean(log(RESULT_SUB_DL))),digits=3),
                   SubDL_SD=round(exp(sd(log(RESULT_SUB_DL))),digits=3),
                   SubZero_Mean=round(exp(mean(log(RESULT_SUB_ZERO))),digits=3),
                   SubZero_SD=round(exp(sd(log(RESULT_SUB_ZERO))),digits=3),

ROS_Mean=round(exp(ifelse(CENSOR_Adequate,enormCensored(log(RESULT_SUB_DL),CENSORED,meth
od="rROS",ci=F,lb.impute=log(min(REPORTING_DETECTION_LIMIT))*0.10,ub.impute=max(log(REPORTIN
G_DETECTION_LIMIT)))$parameters[[1]],NA)),digits=3),

ROS_SD=round(exp(ifelse(CENSOR_Adequate,enormCensored(log(RESULT_SUB_DL),CENSORED,method
="rROS",ci=F,lb.impute=log(min(REPORTING_DETECTION_LIMIT))*0.10,ub.impute=max(log(REPORTING
_DETECTION_LIMIT)))$parameters[[2]],NA)),digits=3)) %>%
  ungroup()  %>%
  dplyr::mutate(across(Raw_Mean, ~replace(.,is.nan(.),NA)),
                across(SubZero_SD, ~replace(.,is.nan(.),NA)),
                Trophic_Level=case_when(TAXON_2=='Aplodinotus grunniens' ~ 3,
                                        TAXON_2=='Cyprinus carpio' ~ 3,
                                        TAXON_2=='Lepomis macrochirus' ~ 3,
                                        TAXON_2=='Morone chrysops' ~ 4,
```

```
                    TAXON_2=='Micropterus dolomieu' ~ 4,
                    TAXON_2=='Pomoxis nigromaculatus' ~ 3,
                    TAXON_2=='Sander Genus' ~ 4)) %>%
  dplyr::arrange(TAXON_2,SHORT_NAME)


### note: you can't use method "mle" because the data needs a lower boundary set; you can set boundaries with method "impute.w.mle" but this doesn't work with multiple detection limits as is the case with these data; using method "mle" won't work for these data
#
RESULT_MLE_MEAN=ifelse(CENSOR_Adequate,enormCensored(RESULT_SUB_DL,CENSORED,method="mle")$parameters[[1]],NA)


Means_byTaxa <- fish_nd_eds %>%
  dplyr::group_by(TAXON_2,SHORT_NAME,RESULT_UNIT) %>%
  dplyr::summarise(CENSOR_Adequate=case_when(sum(CENSORED)>2 & n_distinct(RESULT_SUB_DL[CENSORED!=TRUE])>2 ~ TRUE,
                                             TRUE ~ FALSE),
            TOTAL_COUNT=n(),
            NOT_DETECTED_COUNT=sum(is.na(RESULT_NUMERIC)),
            PERCENT_ND=round((NOT_DETECTED_COUNT/TOTAL_COUNT)*100,digits=1),
            Raw_Mean=round(mean(RESULT_NUMERIC,na.rm=T),digits=3),
            Raw_SD=round(sd(RESULT_NUMERIC,na.rm=T),digits=3),
            SubHalfDL_Mean=round(mean(RESULT_SUB_HALF_DL),digits=3),
            SubHalfDL_SD=round(sd(RESULT_SUB_HALF_DL),digits=3),
            SubDL_Mean=round(mean(RESULT_SUB_DL),digits=3),
            SubDL_SD=round(sd(RESULT_SUB_DL),digits=3),
            SubZero_Mean=round(mean(RESULT_SUB_ZERO),digits=3),
            SubZero_SD=round(sd(RESULT_SUB_ZERO),digits=3),

KaplanMeier_Mean=round(ifelse(CENSOR_Adequate,enparCensored(RESULT_SUB_DL,CENSORED)$parameters[[1]],NA),digits=3),


KaplanMeier_SD=round(ifelse(CENSOR_Adequate,enparCensored(RESULT_SUB_DL,CENSORED)$parameters[[2]],NA),digits=3),


ROS_Mean=round(ifelse(CENSOR_Adequate,enormCensored(RESULT_SUB_DL,CENSORED,method="rRO
```

```
S",ci=F,lb.impute=min(REPORTING_DETECTION_LIMIT)*0.10,ub.impute=max(REPORTING_DETECTION_L
IMIT))$parameters[[1]],NA),digits=3),

ROS_SD=round(ifelse(CENSOR_Adequate,enormCensored(RESULT_SUB_DL,CENSORED,method="rROS",
ci=F,lb.impute=min(REPORTING_DETECTION_LIMIT)*0.10,ub.impute=max(REPORTING_DETECTION_LIM
IT))$parameters[[2]],NA),digits=3)) %>%
 ungroup()  %>%
 dplyr::mutate(across(Raw_Mean, ~replace(.,is.nan(.),NA)),
        Trophic_Level=case_when(TAXON_2=='Aplodinotus grunniens' ~ 3,
                TAXON_2=='Cyprinus carpio' ~ 3,
                TAXON_2=='Lepomis macrochirus' ~ 3,
                TAXON_2=='Morone chrysops' ~ 4,
                TAXON_2=='Micropterus dolomieu' ~ 4,
                TAXON_2=='Pomoxis nigromaculatus' ~ 3,
                TAXON_2=='Sander Genus' ~ 4)) %>%
 dplyr::arrange(TAXON_2,SHORT_NAME)

##################
### EXPORT DATA

GeoMeans_byTaxa_print <- GeoMeans_byTaxa %>%

dplyr::rename(Compound=SHORT_NAME,Unit=RESULT_UNIT,Censored_Data_Adequate=CENSOR_Adeq
uate,Total_Count=TOTAL_COUNT,Not_Detected_Count=NOT_DETECTED_COUNT,Percent_ND=PERCENT
_ND,Taxon=TAXON_2)
getwd()
write_csv(GeoMeans_byTaxa_print,'GeoMeanStats_byTaxa.csv')

Means_byTaxa_print <- Means_byTaxa %>%

dplyr::rename(Compound=SHORT_NAME,Unit=RESULT_UNIT,Censored_Data_Adequate=CENSOR_Adeq
uate,Total_Count=TOTAL_COUNT,Not_Detected_Count=NOT_DETECTED_COUNT,Percent_ND=PERCENT
_ND,Taxon=TAXON_2)
getwd()
write_csv(Means_byTaxa_print,'MeanStats_byTaxa.csv')

################
```

```
### MAKE PLOT

### arithmetic means

lapply(Means_byTaxa,class)
names(Means_byTaxa)

forplot <- Means_byTaxa %>%
  dplyr::select(TAXON_2,SHORT_NAME,Raw_Mean:ROS_SD)
#dplyr::mutate(id=paste(TAXON_2,SHORT_NAME,sep="_")) %>%

pivot1 <- forplot %>%
  pivot_longer(-c(TAXON_2,SHORT_NAME),
               names_to="Type",
               values_to="Value")

pivot2 <- pivot1 %>%
  separate_wider_delim(Type,delim="_",names=c("METHOD","STATISTIC"))

pivot3 <- pivot2 %>%
  pivot_wider(names_from="STATISTIC",
              values_from="Value") %>%
  mutate(METHOD=case_when(METHOD=='KaplanMeier' ~ 'Kaplan-Meier',
                          METHOD=='SubDL' ~ 'Detection Limit',
                          METHOD=='SubHalfDL' ~ 'Half Detection Limit',
                          METHOD=='SubZero' ~ 'Zero',
                          TRUE ~ METHOD))

unique(pivot3$TAXON_2)

results_by_MethodandTaxa <-
ggplot(pivot3,aes(x=METHOD,y=Mean,group=TAXON_2,colour=TAXON_2)) +
  geom_point() +
  facet_wrap(~SHORT_NAME,scales="free") +
```

```
  theme(axis.text.x=element_text(angle=90,vjust=0.5,hjust=1)) +
  geom_errorbar(aes(ymin=ifelse(Mean-SD<0,0,Mean-SD),ymax=Mean+SD,width=0.2)) +
  xlab("Non-Detection Method") +
  ylab("Mean +/- SD (ng/L)") +
  guides(color=guide_legend(title="Taxon")) +
  theme(legend.position="bottom") +
  theme(legend.title=element_blank(),legend.text=element_text(size=6))
getwd()
ggsave("Taxa_PFAS_by_NDMethod.tiff",path=directory,width=5,height=6,device='tiff',dpi=700)

########################
### BOXPLOTS BY TAXA

names(fish_nd_eds)

raw_by_taxa <- ggplot(fish_nd_eds,aes(x=TAXON_2,y=log(RESULT_NUMERIC))) +
  geom_boxplot() +
  facet_wrap(~SHORT_NAME,scales="free") +
  theme(axis.text.x=element_text(angle=90,vjust=0.5,hjust=1)) +
  ylab("Log Fish Filet PFAS (ng/g)") +
  theme(axis.title.x = element_blank()) +
  theme(axis.text.x = element_text(size=9))
getwd()
ggsave("Raw_Taxa_PFAS.tiff",path=directory,width=5,height=6,device='tiff',dpi=700)

DL_by_taxa <- ggplot(fish_nd_eds,aes(x=TAXON_2,y=log(RESULT_SUB_DL))) +
  geom_boxplot() +
  facet_wrap(~SHORT_NAME,scales="free") +
  theme(axis.text.x=element_text(angle=90,vjust=0.5,hjust=1)) +
  ylab("Log Fish Filet PFAS (ng/g)") +
  theme(axis.title.x = element_blank()) +
  theme(axis.text.x = element_text(size=9))
getwd()
ggsave("DL_Taxa_PFAS.tiff",path=directory,width=5,height=6,device='tiff',dpi=700)
```

```
HalfDL_by_taxa <- ggplot(fish_nd_eds,aes(x=TAXON_2,y=log(RESULT_SUB_HALF_DL))) +
  geom_boxplot() +
  facet_wrap(~SHORT_NAME,scales="free") +
  theme(axis.text.x=element_text(angle=90,vjust=0.5,hjust=1)) +
  ylab("Log Fish Filet PFAS (ng/g)") +
  theme(axis.title.x = element_blank()) +
  theme(axis.text.x = element_text(size=9))
getwd()
ggsave("HalfDL_Taxa_PFAS.tiff",path=directory,width=5,height=6,device='tiff',dpi=700)

Zero_by_taxa <- ggplot(fish_nd_eds,aes(x=TAXON_2,y=log(RESULT_SUB_ZERO))) +
  geom_boxplot() +
  facet_wrap(~SHORT_NAME,scales="free") +
  theme(axis.text.x=element_text(angle=90,vjust=0.5,hjust=1)) +
  ylab("Log Fish Filet PFAS (ng/g)") +
  theme(axis.title.x = element_blank()) +
  theme(axis.text.x = element_text(size=9))
getwd()
ggsave("Zero_Taxa_PFAS.tiff",path=directory,width=5,height=6,device='tiff',dpi=700)

########################################
### CALCULATE MEANS BY TROPHIC LEVEL

GeoMeans_byTL <- fish_nd_eds %>%
  dplyr::mutate(Trophic_Level=case_when(TAXON_2=='Aplodinotus grunniens' ~ 3,
                                        TAXON_2=='Cyprinus carpio' ~ 3,
                                        TAXON_2=='Lepomis macrochirus' ~ 3,
                                        TAXON_2=='Morone chrysops' ~ 4,
                                        TAXON_2=='Micropterus dolomieu' ~ 4,
                                        TAXON_2=='Pomoxis nigromaculatus' ~ 3,
                                        TAXON_2=='Sander Genus' ~ 4)) %>%
  dplyr::group_by(Trophic_Level,SHORT_NAME,RESULT_UNIT) %>%
```

```
  dplyr::summarise(CENSOR_Adequate=case_when(sum(CENSORED)>2 &
n_distinct(RESULT_SUB_DL[CENSORED!=TRUE])>2 ~ TRUE,
                                   TRUE ~ FALSE),
            TOTAL_COUNT=n(),
            NOT_DETECTED_COUNT=sum(is.na(RESULT_NUMERIC)),
            PERCENT_ND=round((NOT_DETECTED_COUNT/TOTAL_COUNT)*100,digits=1),
            Raw_Mean=round(exp(mean(log(RESULT_NUMERIC),na.rm=T)),digits=3),
            Raw_SD=round(exp(sd(log(RESULT_NUMERIC),na.rm=T)),digits=3),
            SubHalfDL_Mean=round(exp(mean(log(RESULT_SUB_HALF_DL))),digits=3),
            SubHalfDL_SD=round(exp(sd(log(RESULT_SUB_HALF_DL))),digits=3),
            SubDL_Mean=round(exp(mean(log(RESULT_SUB_DL))),digits=3),
            SubDL_SD=round(exp(sd(log(RESULT_SUB_DL))),digits=3),
            SubZero_Mean=round(exp(mean(log(RESULT_SUB_ZERO))),digits=3),
            SubZero_SD=round(exp(sd(log(RESULT_SUB_ZERO))),digits=3),

ROS_Mean=round(exp(ifelse(CENSOR_Adequate,enormCensored(log(RESULT_SUB_DL),CENSORED,meth
od="rROS",ci=F,lb.impute=log(min(REPORTING_DETECTION_LIMIT))*0.10,ub.impute=max(log(REPORTIN
G_DETECTION_LIMIT)))$parameters[[1]],NA)),digits=3),

ROS_SD=round(exp(ifelse(CENSOR_Adequate,enormCensored(log(RESULT_SUB_DL),CENSORED,method
="rROS",ci=F,lb.impute=log(min(REPORTING_DETECTION_LIMIT))*0.10,ub.impute=max(log(REPORTING
_DETECTION_LIMIT)))$parameters[[2]],NA)),digits=3)) %>%
  ungroup()  %>%
  dplyr::mutate(across(Raw_Mean, ~replace(.,is.nan(.),NA)),
           across(SubZero_SD, ~replace(.,is.nan(.),NA))) %>%
  dplyr::arrange(Trophic_Level,SHORT_NAME)


names(fish_nd_eds)
Means_byTL <- fish_nd_eds %>%
  dplyr::mutate(Trophic_Level=case_when(TAXON_2=='Aplodinotus grunniens' ~ 3,
                                TAXON_2=='Cyprinus carpio' ~ 3,
                                TAXON_2=='Lepomis macrochirus' ~ 3,
                                TAXON_2=='Morone chrysops' ~ 4,
                                TAXON_2=='Micropterus dolomieu' ~ 4,
                                TAXON_2=='Pomoxis nigromaculatus' ~ 3,
                                TAXON_2=='Sander Genus' ~ 4)) %>%
```

```
  dplyr::group_by(Trophic_Level,SHORT_NAME,RESULT_UNIT) %>%
  dplyr::summarise(CENSOR_Adequate=case_when(sum(CENSORED)>2 &
n_distinct(RESULT_SUB_DL[CENSORED!=TRUE])>2 ~ TRUE,
                                             TRUE ~ FALSE),
                   TOTAL_COUNT=n(),
                   NOT_DETECTED_COUNT=sum(is.na(RESULT_NUMERIC)),
                   PERCENT_ND=round((NOT_DETECTED_COUNT/TOTAL_COUNT)*100,digits=1),
                   Raw_Mean=round(mean(RESULT_NUMERIC,na.rm=T),digits=3),
                   Raw_SD=round(sd(RESULT_NUMERIC,na.rm=T),digits=3),
                   SubHalfDL_Mean=round(mean(RESULT_SUB_HALF_DL),digits=3),
                   SubHalfDL_SD=round(sd(RESULT_SUB_HALF_DL),digits=3),
                   SubDL_Mean=round(mean(RESULT_SUB_DL),digits=3),
                   SubDL_SD=round(sd(RESULT_SUB_DL),digits=3),
                   SubZero_Mean=round(mean(RESULT_SUB_ZERO),digits=3),
                   SubZero_SD=round(sd(RESULT_SUB_ZERO),digits=3),

KaplanMeier_Mean=round(ifelse(CENSOR_Adequate,enparCensored(RESULT_SUB_DL,CENSORED)$para
meters[[1]],NA),digits=3),

KaplanMeier_SD=round(ifelse(CENSOR_Adequate,enparCensored(RESULT_SUB_DL,CENSORED)$parame
ters[[2]],NA),digits=3),

ROS_Mean=round(ifelse(CENSOR_Adequate,enormCensored(RESULT_SUB_DL,CENSORED,method="rRO
S",ci=F,lb.impute=min(REPORTING_DETECTION_LIMIT)*0.10,ub.impute=max(REPORTING_DETECTION_L
IMIT))$parameters[[1]],NA),digits=3),

ROS_SD=round(ifelse(CENSOR_Adequate,enormCensored(RESULT_SUB_DL,CENSORED,method="rROS",
ci=F,lb.impute=min(REPORTING_DETECTION_LIMIT)*0.10,ub.impute=max(REPORTING_DETECTION_LIM
IT))$parameters[[2]],NA),digits=3)) %>%
  ungroup()  %>%
  dplyr::mutate(across(Raw_Mean, ~replace(.,is.nan(.),NA))) %>%
  dplyr::arrange(Trophic_Level,SHORT_NAME)

### note: you can't use method "mle" because the data needs a lower boundary set; you can set
boundaries with method "impute.w.mle" but this doesn't work with multiple detection limits as is the
case with these data; using method "mle" won't work for these data
```

```
#
RESULT_MLE_MEAN=ifelse(CENSOR_Adequate,enormCensored(RESULT_SUB_DL,CENSORED,method="
mle")$parameters[[1]],NA)


##################
### EXPORT DATA


GeoMeans_byTL_print <- GeoMeans_byTL %>%

dplyr::rename(Compound=SHORT_NAME,Unit=RESULT_UNIT,Censored_Data_Adequate=CENSOR_Adeq
uate,Total_Count=TOTAL_COUNT,Not_Detected_Count=NOT_DETECTED_COUNT,Percent_ND=PERCENT
_ND)
getwd()
write_csv(GeoMeans_byTL_print,'GeoMeanStats_byTL.csv')


Means_byTL_print <- Means_byTL %>%

dplyr::rename(Compound=SHORT_NAME,Unit=RESULT_UNIT,Censored_Data_Adequate=CENSOR_Adeq
uate,Total_Count=TOTAL_COUNT,Not_Detected_Count=NOT_DETECTED_COUNT,Percent_ND=PERCENT
_ND)
getwd()
write_csv(Means_byTL_print,'MeanStats_byTL.csv')


#################
### MAKE PLOT


### arithmetic means


lapply(Means_byTL,class)
names(Means_byTL)


unique(Means_byTL$Trophic_Level)
class(Means_byTL$Trophic_Level)


forplot <- Means_byTL %>%
  dplyr::select(Trophic_Level,SHORT_NAME,Raw_Mean:ROS_SD) %>%
```

```
  dplyr::mutate(Trophic_Level=as.character(Trophic_Level))
#dplyr::mutate(id=paste(TAXON_2,SHORT_NAME,sep="_")) %>%

pivot1 <- forplot %>%
  pivot_longer(-c(Trophic_Level,SHORT_NAME),
               names_to="Type",
               values_to="Value")

pivot2 <- pivot1 %>%
  separate_wider_delim(Type,delim="_",names=c("METHOD","STATISTIC"))

pivot3 <- pivot2 %>%
  pivot_wider(names_from="STATISTIC",
              values_from="Value") %>%
  mutate(METHOD=case_when(METHOD=='KaplanMeier' ~ 'Kaplan-Meier',
                          METHOD=='SubDL' ~ 'Detection Limit',
                          METHOD=='SubHalfDL' ~ 'Half Detection Limit',
                          METHOD=='SubZero' ~ 'Zero',
                          TRUE ~ METHOD))

results_by_MethodandTrophicLevel <-
ggplot(pivot3,aes(x=METHOD,y=Mean,group=Trophic_Level,colour=Trophic_Level)) +
  geom_point() +
  facet_wrap(~SHORT_NAME,scales="free") +
  theme(axis.text.x=element_text(angle=90,vjust=0.5,hjust=1)) +
  geom_errorbar(aes(ymin=ifelse(Mean-SD<0,0,Mean-SD),ymax=Mean+SD,width=0.2)) +
  xlab("Non-Detection Method") +
  ylab("Mean +/- SD (ng/g)") +
  guides(color=guide_legend(title="Trophic Level")) +
  theme(legend.position="bottom") +
  theme(legend.title=element_text(size=7),legend.text=element_text(size=7)) +
  scale_color_manual(values=c("red","blue"))
getwd()
ggsave("TrophicLevel_PFAS_by_NDMethod.tiff",path=directory,width=5,height=6,device='tiff',dpi=700)
```

```
### geometic means

lapply(GeoMeans_byTL,class)
names(GeoMeans_byTL)

unique(GeoMeans_byTL$Trophic_Level)
class(GeoMeans_byTL$Trophic_Level)

Geo_forplot <- GeoMeans_byTL %>%
  dplyr::select(Trophic_Level,SHORT_NAME,Raw_Mean:ROS_SD) %>%
  dplyr::mutate(Trophic_Level=as.character(Trophic_Level))
#dplyr::mutate(id=paste(TAXON_2,SHORT_NAME,sep="_")) %>%

Geo_pivot1 <- Geo_forplot %>%
  pivot_longer(-c(Trophic_Level,SHORT_NAME),
               names_to="Type",
               values_to="Value")

Geo_pivot2 <- Geo_pivot1 %>%
  separate_wider_delim(Type,delim="_",names=c("METHOD","STATISTIC"))

Geo_pivot3 <- Geo_pivot2 %>%
  pivot_wider(names_from="STATISTIC",
              values_from="Value") %>%
  mutate(METHOD=case_when(METHOD=='KaplanMeier' ~ 'Kaplan-Meier',
                          METHOD=='SubDL' ~ 'Detection Limit',
                          METHOD=='SubHalfDL' ~ 'Half Detection Limit',
                          METHOD=='SubZero' ~ 'Zero',
                          TRUE ~ METHOD)) %>%
  dplyr::filter(METHOD!='Kaplan-Meier',
                METHOD!='Zero')
```

```
geo_results_by_MethodandTrophicLevel <-
ggplot(Geo_pivot3,aes(x=METHOD,y=Mean,group=Trophic_Level,colour=Trophic_Level)) +
 geom_point() +
 facet_wrap(~SHORT_NAME,scales="free") +
 theme(axis.text.x=element_text(angle=90,vjust=0.5,hjust=1)) +
 geom_errorbar(aes(ymin=Mean-SD,ymax=Mean+SD,width=0.2)) +
 #geom_errorbar(aes(ymin=ifelse(Mean-SD<0,0,Mean-SD),ymax=Mean+SD,width=0.2)) +
 xlab("Non-Detection Method") +
 ylab("Geometric Mean +/- SD (ng/g)") +
 guides(color=guide_legend(title="Trophic Level")) +
 theme(legend.position="bottom") +
 theme(legend.title=element_text(size=7),legend.text=element_text(size=7)) +
 scale_color_manual(values=c("red","blue"))
getwd()
ggsave("TrophicLevel_PFAS_by_NDMethod_Geo.tiff",path=directory,width=5,height=6,device='tiff',dpi=
700)

###################################
### CALCULATE MEANS FOR ALL FISH

GeoMeans_AllFish <- fish_nd_eds %>%
 dplyr::group_by(SHORT_NAME,RESULT_UNIT) %>%
 dplyr::summarise(CENSOR_Adequate=case_when(sum(CENSORED)>2 &
n_distinct(RESULT_SUB_DL[CENSORED!=TRUE])>2 ~ TRUE,
                                  TRUE ~ FALSE),
           TOTAL_COUNT=n(),
           NOT_DETECTED_COUNT=sum(is.na(RESULT_NUMERIC)),
           PERCENT_ND=round((NOT_DETECTED_COUNT/TOTAL_COUNT)*100,digits=1),
           Raw_Mean=round(exp(mean(log(RESULT_NUMERIC),na.rm=T)),digits=3),
           Raw_SD=round(exp(sd(log(RESULT_NUMERIC),na.rm=T)),digits=3),
           SubHalfDL_Mean=round(exp(mean(log(RESULT_SUB_HALF_DL))),digits=3),
           SubHalfDL_SD=round(exp(sd(log(RESULT_SUB_HALF_DL))),digits=3),
           SubDL_Mean=round(exp(mean(log(RESULT_SUB_DL))),digits=3),
           SubDL_SD=round(exp(sd(log(RESULT_SUB_DL))),digits=3),
           SubZero_Mean=round(exp(mean(log(RESULT_SUB_ZERO))),digits=3),
```

```
            SubZero_SD=round(exp(sd(log(RESULT_SUB_ZERO))),digits=3),

ROS_Mean=round(exp(ifelse(CENSOR_Adequate,enormCensored(log(RESULT_SUB_DL),CENSORED,meth
od="rROS",ci=F,lb.impute=log(min(REPORTING_DETECTION_LIMIT))*0.10,ub.impute=max(log(REPORTIN
G_DETECTION_LIMIT)))$parameters[[1]],NA)),digits=3),

ROS_SD=round(exp(ifelse(CENSOR_Adequate,enormCensored(log(RESULT_SUB_DL),CENSORED,method
="rROS",ci=F,lb.impute=log(min(REPORTING_DETECTION_LIMIT))*0.10,ub.impute=max(log(REPORTING
_DETECTION_LIMIT)))$parameters[[2]],NA)),digits=3)) %>%
  ungroup()  %>%
  dplyr::mutate(across(Raw_Mean, ~replace(.,is.nan(.),NA)),
            across(SubZero_SD, ~replace(.,is.nan(.),NA))) %>%
  dplyr::arrange(SHORT_NAME)


names(fish_nd_eds)
Means_AllFish <- fish_nd_eds %>%
  dplyr::group_by(SHORT_NAME,RESULT_UNIT) %>%
  dplyr::summarise(CENSOR_Adequate=case_when(sum(CENSORED)>2 &
n_distinct(RESULT_SUB_DL[CENSORED!=TRUE])>2 ~ TRUE,
                                    TRUE ~ FALSE),
            TOTAL_COUNT=n(),
            NOT_DETECTED_COUNT=sum(is.na(RESULT_NUMERIC)),
            PERCENT_ND=round((NOT_DETECTED_COUNT/TOTAL_COUNT)*100,digits=1),
            Raw_Mean=round(mean(RESULT_NUMERIC,na.rm=T),digits=3),
            Raw_SD=round(sd(RESULT_NUMERIC,na.rm=T),digits=3),
            SubHalfDL_Mean=round(mean(RESULT_SUB_HALF_DL),digits=3),
            SubHalfDL_SD=round(sd(RESULT_SUB_HALF_DL),digits=3),
            SubDL_Mean=round(mean(RESULT_SUB_DL),digits=3),
            SubDL_SD=round(sd(RESULT_SUB_DL),digits=3),
            SubZero_Mean=round(mean(RESULT_SUB_ZERO),digits=3),
            SubZero_SD=round(sd(RESULT_SUB_ZERO),digits=3),

KaplanMeier_Mean=round(ifelse(CENSOR_Adequate,enparCensored(RESULT_SUB_DL,CENSORED)$para
meters[[1]],NA),digits=3),

KaplanMeier_SD=round(ifelse(CENSOR_Adequate,enparCensored(RESULT_SUB_DL,CENSORED)$parame
ters[[2]],NA),digits=3),
```

```
ROS_Mean=round(ifelse(CENSOR_Adequate,enormCensored(RESULT_SUB_DL,CENSORED,method="rROS",ci=F,lb.impute=min(REPORTING_DETECTION_LIMIT)*0.10,ub.impute=max(REPORTING_DETECTION_LIMIT))$parameters[[1]],NA),digits=3),

ROS_SD=round(ifelse(CENSOR_Adequate,enormCensored(RESULT_SUB_DL,CENSORED,method="rROS",ci=F,lb.impute=min(REPORTING_DETECTION_LIMIT)*0.10,ub.impute=max(REPORTING_DETECTION_LIMIT))$parameters[[2]],NA),digits=3)) %>%
  ungroup()  %>%
  dplyr::mutate(across(Raw_Mean, ~replace(.,is.nan(.),NA))) %>%
  dplyr::arrange(SHORT_NAME)

### note: you can't use method "mle" because the data needs a lower boundary set; you can set boundaries with method "impute.w.mle" but this doesn't work with multiple detection limits as is the case with these data; using method "mle" won't work for these data
#
RESULT_MLE_MEAN=ifelse(CENSOR_Adequate,enormCensored(RESULT_SUB_DL,CENSORED,method="mle")$parameters[[1]],NA)

##################
### EXPORT DATA

GeoMeans_AllFish_print <- GeoMeans_AllFish %>%

dplyr::rename(Compound=SHORT_NAME,Unit=RESULT_UNIT,Censored_Data_Adequate=CENSOR_Adequate,Total_Count=TOTAL_COUNT,Not_Detected_Count=NOT_DETECTED_COUNT,Percent_ND=PERCENT_ND)
getwd()
write_csv(GeoMeans_AllFish_print,'GeoMeanStats_AllFish.csv')

Means_AllFish_print <- Means_AllFish %>%

dplyr::rename(Compound=SHORT_NAME,Unit=RESULT_UNIT,Censored_Data_Adequate=CENSOR_Adequate,Total_Count=TOTAL_COUNT,Not_Detected_Count=NOT_DETECTED_COUNT,Percent_ND=PERCENT_ND)
getwd()
write_csv(Means_AllFish_print,'MeanStats_AllFish.csv')

################
```

```
### MAKE PLOT

### arithmetic means

lapply(Means_AllFish,class)
names(Means_AllFish)

forplot <- Means_AllFish %>%
  dplyr::select(SHORT_NAME,Raw_Mean:ROS_SD)

pivot1 <- forplot %>%
  pivot_longer(!SHORT_NAME,
               names_to="Type",
               values_to="Value")

pivot2 <- pivot1 %>%
  separate_wider_delim(Type,delim="_",names=c("METHOD","STATISTIC"))

pivot3 <- pivot2 %>%
  pivot_wider(names_from="STATISTIC",
              values_from="Value") %>%
  mutate(METHOD=case_when(METHOD=='KaplanMeier' ~ 'Kaplan-Meier',
                          METHOD=='SubDL' ~ 'Detection Limit',
                          METHOD=='SubHalfDL' ~ 'Half Detection Limit',
                          METHOD=='SubZero' ~ 'Zero',
                          TRUE ~ METHOD))

results_by_MethodandAllFish <- ggplot(pivot3,aes(x=METHOD,y=Mean)) +
  geom_point() +
  facet_wrap(~SHORT_NAME,scales="free") +
  theme(axis.text.x=element_text(angle=90,vjust=0.5,hjust=1)) +
  geom_errorbar(aes(ymin=ifelse(Mean-SD<0,0,Mean-SD),ymax=Mean+SD,width=0.2)) +
  xlab("Non-Detection Method") +
  ylab("Mean +/- SD (ng/g)")
```

```
getwd()
ggsave("AllFish_PFAS_by_NDMethod.tiff",path=directory,width=5,height=5,device='tiff',dpi=700)

### geometic means

lapply(GeoMeans_AllFish,class)
names(GeoMeans_AllFish)

Geo_forplot <- GeoMeans_AllFish %>%
  dplyr::select(SHORT_NAME,Raw_Mean:ROS_SD)

Geo_pivot1 <- Geo_forplot %>%
  pivot_longer(!SHORT_NAME,
               names_to="Type",
               values_to="Value")

Geo_pivot2 <- Geo_pivot1 %>%
  separate_wider_delim(Type,delim="_",names=c("METHOD","STATISTIC"))

Geo_pivot3 <- Geo_pivot2 %>%
  pivot_wider(names_from="STATISTIC",
              values_from="Value") %>%
  mutate(METHOD=case_when(METHOD=='KaplanMeier' ~ 'Kaplan-Meier',
                          METHOD=='SubDL' ~ 'Detection Limit',
                          METHOD=='SubHalfDL' ~ 'Half Detection Limit',
                          METHOD=='SubZero' ~ 'Zero',
                          TRUE ~ METHOD)) %>%
  dplyr::filter(METHOD!='Kaplan-Meier',
                METHOD!='Zero')

geO_results_by_MethodandAllFish <- ggplot(Geo_pivot3,aes(x=METHOD,y=Mean)) +
  geom_point() +
  facet_wrap(~SHORT_NAME,scales="free") +
  theme(axis.text.x=element_text(angle=90,vjust=0.5,hjust=1)) +
```

```
  geom_errorbar(aes(ymin=Mean-SD,ymax=Mean+SD,width=0.2)) +
  #geom_errorbar(aes(ymin=ifelse(Mean-SD<0,0,Mean-SD),ymax=Mean+SD,width=0.2)) +
  xlab("Non-Detection Method") +
  ylab("Geometric Mean +/- SD (ng/g)")
getwd()
ggsave("AllFish_PFAS_by_NDMethod_Geo.tiff",path=directory,width=5,height=5,device='tiff',dpi=700)
```