SUPPORTING INFORMATION

Title

Characterizing spatial information loss for wastewater surveillance using crAssphage: effect of decay, temperature, and population mobility

Authors

Corinne Wiesner-Friedman[a,b*], Nichole E. Brinkman[b], Emily Wheaton[b], Maitreyi Nagarkar[b], Chloe Hart[b], Scott P. Keely[b], Eunice Varughese[b], Jay Garland[b], Peter Klaver[c], Carrie Turner[c], John Barton[d], Marc Serre[e], Michael Jahne[b]

Author information

[a] Oak Ridge Institute for Science and Education, 26 West Martin Luther King Drive, Cincinnati, Ohio 45268, United States

[b] U.S. Environmental Protection Agency, 26 West Martin Luther King Drive, Cincinnati, Ohio 45268, United States

[c] LimnoTech, 501 Avis Drive, Ann Arbor, Michigan, 48108, United States

[d] Metropolitan Sewer District of Greater Cincinnati, 1081 Woodrow St, Cincinnati, Ohio 45204, United States

[e] Gillings School of Global Public Health, Department of Environmental Sciences and Engineering, University of North Carolina at Chapel Hill, Chapel Hill, North Carolina, 27599, United States

Number of pages

93

Number of Figures

9

1. Figure S1 Data available from census block group level (a) travel time to wastewater treatment plant (b) population (b) number of commercial parcels (representative of potential retail) and (d) the septic system count
2. Figure S2 The crAssphage load over time by WWTP (or sewershed)
3. Figure S3 The optimum signal decay hyperparameter α_{90} from residential locations for log10 crAssphage concentration with the SEDC SPM
4. Figure S4 The optimum signal decay hyperparameter α_{90} from residential locations for log10 crAssphage load with the SEDC SPM
5. Figure S5 The optimum signal decay hyperparameter $\alpha_{90}(0)$ and signal decay modifier from temperature γ_1 from residential locations for log10 crAssphage load with the SEDC-T SPM. The colorbar labeled C represents R^2. The values descend much lower, but we have cut them off at 0.560 to be able to show the maximum more clearly.
6. Figure S6 The optimum signal decay hyperparameter $\alpha_{90}(0)$ and signal decay modifiers from temperature γ_1 and mobility γ_2 from residential locations for log10 crAssphage load with the SEDC-TM SPM.

7. Figure S7 Information loss rates across the census block groups when looking at α_{90} values from the SEDC in relation to responses (a) flow (b) crAssphage concentration and (c) crAssphage load (also depicted in main text oof the manuscript)
8. Figure S8 Selection of the MSE and the 1-standard-error $log\ (\lambda)$ for the Test stage of FIT
9. Figure S9 Bootstrapped standardized regression coefficients and 95% confidence intervals for the variables selected with the LASSO regression. To compare the impact of the sum of exponentially decaying contributions (SEDC) from residentially located populations to other factors. ^ indicates mobility variable representing % time spent at the location compared to baseline
10. Figure S10 Selection of the MSE and the 1-standard-error $log\ (\lambda)$ for the LASSO-MM

Number of Tables

7

1. Table S1 Studies of RNA and DNA decay in wastewater
2. Table S2 Descriptive statistics for the responses and LASSO regression variables overall and by sewershed
3. Table S3 Summary of census block group populations within sewershed catchments
4. Table S4 Summary of variables, parameters, and hyperparameters
5. Table S5 Provides the key SPM values (α_{90},β_1, and Adj. R^2) from running the Inform stage when not excluding populations on septic systems and using Euclidean distance instead of travel time.
6. Table S6 Test stage bootstrapped LASSO regression results with confidence intervals around the regression coefficient estimates.
7. Table S7 Test stage bootstrapped LASSO with mobility as modifiers regression results with confidence intervals around the regression coefficient estimates.

S1. Data sources and Processing in R and ArcMap 10.5

1.1 Sewershed boundaries

Sewershed boundary data was obtained via e-mail correspondence with from the Metropolitan Sewer District of Greater Cincinnati (MSD).

1.2 Census block group data

Census block group data was obtained with the tidycensus package in R.

```
## Packages
library(readxl)
library(lubridate)
library(tidycensus)
library(sf)
library(lwgeom)
library(units)
#library(ipumsr)
library(srvyr)
library(tidyverse)

## Options
options(mc.cores = parallel::detectCores())

## Define general functions
`%notin%` = function(x,y) {!(x %in% y)}

## set census api key
census_api_key("50ae4c40f86f7f50a7e4ba3eae7f64a52e0adfe4")
Sys.getenv("CENSUS_API_KEY")

## age x sex
age_sex_var <- read_xlsx("~/Rworking/SARS-CoV-2/GISdata/census_codes.xlsx",
                         sheet = "B01001") %>%
  mutate(variable = str_c(acs_tab, var_id, sep = "_")) %>%
  select(variable, sex, age, universe)

age_sex_full <- get_acs(geography = "block group", table = "B01001",
                        state = "OH", geometry = FALSE, year = 2019,
                        output = "tidy", cache_table = TRUE,
                        show_call = TRUE) %>%
  left_join(age_sex_var, by = c("variable")) %>%
  rename(bg_id = GEOID)

age_sex <- age_sex_full %>%
  mutate(age = fct_collapse(age,
                            "17" = c("14", "17"),
```

```
                        "24" = c("19", "20", "21", "24"),
                        "34" = c("29", "34"),
                        "44" = c("39","44"),
                        "54" = c("49", "54"),
                        "64" = c("59", "61", "64"),
                        "74" = c("66", "69", "74"),
                        "75up" = c("79", "84", "85up")) %>%
          fct_relevel(c("4", "9", "17",
                        "24", "34", "44",
                        "54", "64", "74",
                        "75up", "tot"))) %>%
  group_by(bg_id, NAME,
           sex, age) %>%
  summarise(n = sum(estimate)) %>%
  ungroup()

saveRDS(age_sex, "~/Rworking/SARS-CoV-2/GISdata/age_sex.rds")
data.table::fwrite(age_sex, "~/Rworking/SARS-CoV-2/GISdata/age_sex.csv")

age <- age_sex %>%
  filter(sex != "tot") %>%
  group_by(bg_id, NAME,
           age) %>%
  summarise(n = sum(n)) %>%
  ungroup()
saveRDS(age, "~/Rworking/SARS-CoV-2/GISdata/age.rds")
data.table::fwrite(age, "~/Rworking/SARS-CoV-2/GISdata/age.csv")

sex <- age_sex %>%
  filter(!(sex != "tot" & age == "tot")) %>%
  group_by(bg_id, NAME,
           sex) %>%
  summarise(n = sum(n)) %>%
  ungroup()
saveRDS(sex, "~/Rworking/SARS-CoV-2/GISdata/sex.rds")
data.table::fwrite(sex, "~/Rworking/SARS-CoV-2/GISdata/sex.csv")

bg_codes <- readRDS("~/Rworking/SARS-CoV-2/GISdata/sex.rds") %>%
  filter(sex != "tot") %>%
  group_by(bg_id, NAME) %>%
  summarise(acs_pop = sum(n)) %>%
  ungroup()
saveRDS(bg_codes, "~/Rworking/SARS-CoV-2/GISdata/bg_names.rds")
```

Sewershed and census data join

```
## Packages
library(data.table)
library(dtplyr)
library(readxl)
```

```
library(lubridate)
library(units)
library(sf)
library(lwgeom)
library(tidycensus)
library(tidyverse)
library(rgdal)

## Options
options(mc.cores = parallel::detectCores())
options(dplyr.summarise.inform = FALSE)

## Define general functions
`%notin%` = function(x,y) {!(x %in% y)}

## Tidycensus
#census_api_key("50ae4c40f86f7f50a7e4ba3eae7f64a52e0adfe4", install = TRUE)
Sys.getenv("CENSUS_API_KEY")

## Load the Ohio 2019 Census shapefile
age_sex_var <- read_xlsx("~/Rworking/SARS-CoV-2/GISdata/census_codes.xlsx",
                         sheet = "B01001") %>%
  mutate(variable = str_c(acs_tab, var_id, sep = "_")) %>%
  select(variable, sex, age, universe)

age_sex_full <- get_acs(geography = "block group", table = "B01001",
                        state = "OH", geometry = TRUE, year = 2019,
                        output = "tidy", cache_table = TRUE,
                        show_call = TRUE) %>%
  left_join(age_sex_var, by = c("variable")) %>%
  rename(bg_id = GEOID)

## Just get the shapefile information
bg_geog<-age_sex_full[,c(1,9)]
remove(age_sex_full)
remove(age_sex_var)

st_write(bg_geog,"~/Rworking/SARS-CoV-2/GISdata/bg_geog_2019.shp")
## here we want to just get BGs that intersect with the sewersheds and know the percent of the
BGs that intersect. This is already done in ArcMap, but want to do this in R to automate the
process

## RELATE CENSUS BLOCK GROUPS TO SEWERSHEDS
Currently done in ArcMap (see processing notes below in section 1.3- preparing the
Census_Block_Sewersheds.shp for future use.

require(rgdal)
```

```
filename1<-"~/Rworking/SARS-CoV-2/GISdata/Sewersheds_Geographic.shp"
sewersheds<-read_sf(dsn=filename1)
sewersheds<-st_transform(sewersheds,4269)

filename2<-"~/Rworking/SARS-CoV-2/GISdata/Census_Block_Sewersheds.shp"
CensusBlocks<-read_sf(dsn=filename2)
CensusBlocks<-st_transform(CensusBlocks,4269)
CensusBlocksCopy<-CensusBlocks

## append census data to census blocks
age_sex<-read.csv(file="~/Rworking/SARS-CoV-2/GISdata/Census_2019_data/age_sex.csv")
age<-read.csv(file="~/Rworking/SARS-CoV-2/GISdata/Census_2019_data/age.csv")
bg_id<-unique(age$bg_id)
n_4<-vector(mode="numeric",length=length(bg_id))
n_9<-vector(mode="numeric",length=length(bg_id))
n_17<-vector(mode="numeric",length=length(bg_id))
n_24<-vector(mode="numeric",length=length(bg_id))
n_34<-vector(mode="numeric",length=length(bg_id))
n_44<-vector(mode="numeric",length=length(bg_id))
n_54<-vector(mode="numeric",length=length(bg_id))
n_64<-vector(mode="numeric",length=length(bg_id))
n_74<-vector(mode="numeric",length=length(bg_id))
n_75up<-vector(mode="numeric",length=length(bg_id))
n_tot<-vector(mode="numeric",length=length(bg_id))

for (i in 1:length(bg_id)){
  idxBG<-age$bg_id==bg_id[i]
  idxCAT<-age$age=="4"
  n_4[i]<-age$n[idxBG & idxCAT]
  idxCAT<-age$age=="9"
  n_9[i]<-age$n[idxBG & idxCAT]
  idxCAT<-age$age=="17"
  n_17[i]<-age$n[idxBG & idxCAT]
  idxCAT<-age$age=="24"
  n_24[i]<-age$n[idxBG & idxCAT]
  idxCAT<-age$age=="34"
  n_34[i]<-age$n[idxBG & idxCAT]
  idxCAT<-age$age=="44"
  n_44[i]<-age$n[idxBG & idxCAT]
  idxCAT<-age$age=="54"
  n_54[i]<-age$n[idxBG & idxCAT]
  idxCAT<-age$age=="64"
  n_64[i]<-age$n[idxBG & idxCAT]
  idxCAT<-age$age=="74"
  n_74[i]<-age$n[idxBG & idxCAT]
  idxCAT<-age$age=="75up"
  n_75up[i]<-age$n[idxBG & idxCAT]
```

```
  idxCAT<-age$age=="tot"
  n_tot[i]<-age$n[idxBG & idxCAT]
}

agedf<-data.frame(bg_id,n_4,n_9,n_17,n_24,n_34,n_44,n_54,n_64,n_74,n_75up,n_tot)


CensusBlocks<-merge(CensusBlocksCopy,agedf,by.x="GEOID", by.y="bg_id")

## GET PERCENT OF BLOCK GROUP IN SEWERSHED AREA.
Currently done in ArcMap

## Allocate counts to in-sewershed block groups

We weight the counts for each block group by what percent of the block group
spatially overlaps with the sewershed.

CensusBlocks[,27:80]<-CensusBlocks[,27:80]*CensusBlocks$Pcnt_Sewer
CensusBlocks2<-as.data.frame(CensusBlocks[,c(20,21,22,23,27:80)])

CensusBlocks3<-as.data.frame(CensusBlocks2[,5:58])

uSs<-unique(CensusBlocks2$Sewershed)
tmpmat<-matrix(data=NA,nrow=length(uSs),ncol=58)

for (i in 1:length(uSs)){
  idxss<-CensusBlocks2$Sewershed==uSs[i]
  tmpmat[i,1]<-uSs[i]
  tmpmat[i,2]<-unique(CensusBlocks2$Area_Ft[idxss])
  tmpmat[i,3]<-unique(CensusBlocks2$X_Centroid[idxss])
  tmpmat[i,4]<-unique(CensusBlocks2$Y_Centroid[idxss])
  tmp<-as.data.frame(CensusBlocks2[idxss,5:58])
  tmpmat[i,5:58]<-colSums(tmp)


}

SewerCensusData<-data.frame(tmpmat)
cns<-colnames(CensusBlocks2)
colnames(SewerCensusData)<-cns[1:58]

data.table::fwrite(SewerCensusData, "~/Rworking/SARS-CoV-
2/GISdata/SewerCensusData.csv")
```

1.3 Parcel and zoning data

We obtained data for both Hamilton (Cincinnati Area) and Montgomery County (Dayton Area), but only used Hamilton County data for this analysis.

Hamilton County

Hamilton County parcel data obtained from http://cagis.org/Opendata/Auditor/ with linkage file containing zoning categories created from the table here: https://www.hamiltoncountyauditor.org/pcapage2.asp

The table is saved as HamiltonCo_LandUseClasses.xlsx

Montgomery County

Downloaded zoning data from this link: http://www.mcauditor.org/downloads/gis_downloads.cfm

A linkage file was created from the data columns, ZONE_CODE and ZONE_DESC (MontgomeryCo_ZoningClasses.xlsx) to categorize each zone as industrial, residential, commercial, floodplain, or other.

- Anything with "Industrial" or codes with "I-" were categorized as industrial
- Anything with family, residence, residential, or "r-" or "-r" codes were viewed and determined to be residential or not
- Anything with "business", "commercial", or "office" in the description were categorized as commercial
- I also categorized floodplain classifications.
- Any remaining blanks were viewed for description and zone code to determine which classification based on other entries.
- If no classification could be found, I classified the zone as "other".

Joining the zoning via the linkage files to the shapefiles

1. Load the parcel shapefiles into ArcMap 10.8
2. Load the linkage .csv files into ArcMap 10.8
3. For Montgomery County data
    a. Right click the shapefile
    b. Click join
    c. Join by FID (retained from the export of the text file to make the linkage file)
    d. Check the table to make sure that Land_Use_Cat (which is zoning category) comes up.
4. For Hamilton County data
    a. Right click the shapefile (HamParcelMerged)
    b. The field in the layer that the join will be based on is "CLASS"
    c. Join by "Department of Tax Equalization"
    d. Check the table to make sure that "CATEGORY" comes up.

Preparing the Census_Block_Sewersheds shapefile for future use

1. Add two new fields to Census_Block_Sewersheds (will be used for distance calculations later)
    a. Xc_BG
    b. Yc_BG
2. Use selection tool to select either South or North Census Block Groups.
3. Set the data frame to either South or North StatePlane Coordinate System

4. Calculate geometry (X coordinate of centroid, Y coordinate of centroid). Use the NAD 1983 StatePlane Ohio (South/North) FIPS (3402/3401) Feet for the calculation. Calculate in Feet.
5. Calculate census block group area by adding a new field (CBG_Area) and calculating the area in Feet. Follow procedure above for either the North or South stateplane coordinate system

Getting number of commercial parcels per census block group

1. For Parcels_MC (Montgomery County parcel data) use the Field Calculator to get the longitude and latitude of each parcel
2. Export to a text file
3. Add the table to the map
4. Click on the table in the layers and select "Display XY Data"
5. Make sure the coordinate system is the same as for the parcel data polygon shapefile
6. Export the points to a shapefile called ParcelPoints_MC
7. Use the Spatial Join tool under Analysis > Overlay to relate the zonings from Zoning_MC.shp to the points (this will take a long time to run)
8. Select by attributes the zoning (Montgome_3) that is "Commercial"
9. Export the selected features as CommParcelPoints_MC.shp
10. For the Hamilton County Parcels, use the Feature to Point tool under Data Management > Features to get the parcel points. Select "Inside" so that it is the same as a centroid (this will take a long time to run).
11. Select by attributes the zoning (HamiltonCo) that is "COMMERCIAL"
12. Export the selected features as CommParcelPoints_HC.shp
13. Use the Merge tool under Data Management Tools > General Toolset to get CommParcelPoints.shp
14. Load the shapefile Census_Block_Sewersheds.shp
15. Follow these instructions https://support.esri.com/en/technical-article/000008599#:~:text=In%20the%20attribute%20table%20of,Count%20%3D%20%2C%20and%20click%20OK.
16. Save the output file as CBG_ComParcelCount.shp
17. Go to properties and uncheck irrelevant fields (anything with Sum_...)
18. Export as CBG_ComParcelCount2.shp
19. Then follow R code in GetCommercialParcelCounts.R:

```
#### load libraries ####
library(sf)

#### load the shape file ####

filename<-"~/Rworking/SARS-CoV-2/GISdata/CBG_ComParcelCount2.shp"
ComParcels<-read_sf(dsn=filename)

# drop the geometry
ComParcels<-st_drop_geometry(ComParcels)

#### load the CBG data for Hamilton and Montgomery counties ####
#see ArcMap data management protocols in word document (Data Sources for crAssphage.docx)
ShedData<-read.csv("~/Rworking/Stormwater/GIS_data/ShedData_nogeom.csv")
```

```
#### Append the parcel counts! ####

ShedData$ComParcels=NA
uGEOID=unique(ShedData$GEOID_Data)
for (i in 1:length(uGEOID)){
  idxGEOID=uGEOID[i]==ShedData$GEOID_Data
  prcls=ComParcels$Count_[idxGEOID]
  ShedData$ComParcels[idxGEOID]=prcls
}

ShedData$ComParcels_Pswrd=ShedData$ComParcels*ShedData$Pcnt_Sewer

#### Save the new dataset ####

data.table::fwrite(ShedData,"~/Rworking/Stormwater/crAssphage/ShedDatav2_nogeom.csv")
```

1.4 Septic system locations

The shapefile of septic systems for the greater Cincinnati area was obtained via e-mail correspondence with the Hamilton County Public Health Department and the addresses of septic systems within the boundaries of Cincinnati were obtained via e-mail correspondence with the Cincinnati Health Department.

We geocoded the addresses in R by adapting code written by Shane Lynn in 2013:

```
library(ggmap)
register_google(key ="AIzaSyDiATmSId1s801DEUs8j9shevgxbwNU9Vc")

# get the input data
data<- read.csv("~/Rworking/Stormwater/GIS_data/SepticSystems_Cincinnati.csv")
# get the address list, and append "Ireland" to the end to increase accuracy
# (change or remove this if your address already include a country etc.)
data$Address[95]= "3 Foxhall Ct Cincinnati OH 45219"

addresses = data$Address
addresses = paste0(addresses, ", United States")
#define a function that will process googles server responses for us.
getGeoDetails<- function(address){
  #use the gecode function to query google servers
  geo_reply = geocode(address, output='all', messaging=TRUE, override_limit=TRUE)
  #now extract the bits that we need from the returned list
  answer<- data.frame(lat=NA, long=NA, accuracy=NA, formatted_address=NA,
address_type=NA, status=NA)
  answer$status<- geo_reply$status
  #if we are over the query limit - want to pause for an hour
  while(geo_reply$status == "OVER_QUERY_LIMIT"){
    print("OVER QUERY LIMIT - Pausing for 1 hour at:")
```

```
    time<- Sys.time()
    print(as.character(time))
    Sys.sleep(60*60)
    geo_reply = geocode(address, output='all', messaging=TRUE, override_limit=TRUE)
    answer$status <- geo_reply$status
  }
  #return Na's if we didn't get a match:
  if (geo_reply$status != "OK"){
    return(answer)
  }
  #else, extract what we need from the Google server reply into a dataframe:
  answer$lat <- geo_reply$results[[1]]$geometry$location$lat
  answer$long <- geo_reply$results[[1]]$geometry$location$lng
  if (length(geo_reply$results[[1]]$types) > 0){
    answer$accuracy <- geo_reply$results[[1]]$types[[1]]
  }
  answer$address_type <- paste(geo_reply$results[[1]]$types, collapse=',')
  answer$formatted_address <- geo_reply$results[[1]]$formatted_address
  return(answer)
}
#initialise a dataframe to hold the results
geocoded <- data.frame()
# find out where to start in the address list (if the script was interrupted before):
startindex <- 1
#if a temp file exists - load it up and count the rows!
tempfilename <- paste0('CincySep', '_temp_geocoded.rds')
if (file.exists(tempfilename)){
  print("Found temp file - resuming from index:")
  geocoded <- readRDS(tempfilename)
  startindex <- nrow(geocoded)
  print(startindex)
}
# Start the geocoding process - address by address. geocode() function takes care of query speed
limit.
for (ii in seq(startindex, length(addresses))){
  print(paste("Working on index", ii, "of", length(addresses)))
  #query the google geocoder - this will pause here if we are over the limit.
  result = getGeoDetails(addresses[ii])
  print(result$status)
  result$index <- ii
  #append the answer to the results file.
  geocoded <- rbind(geocoded, result)
  #save temporary results as we are going along
  saveRDS(geocoded, tempfilename)
}

geocoded=geocoded[!duplicated(geocoded$index),]
```

```
#now we add the latitude and longitude to the main data
data$lat <- geocoded$lat
data$long <- geocoded$long
data$accuracy <- geocoded$accuracy
#finally write it all to the output files
# saveRDS(data, paste0("../data/", infile ,"_geocoded.rds"))
write.table(data, file=paste0("~/Rworking/Stormwater/GIS_data/", "CincySep
","_geocoded.csv"), sep=",", row.names=FALSE)
```

We then processed this in ArcMap to obtain the number of septic systems for each census block group that flows to each of the WWTPs defining the sewersheds.

ArcMap processing of septic systems in Hamilton County

1. Open the resulting comma-separated-value file (CincySep_geocoded.csv) in ArcMap 10.8.1 and display X,Y data, then export to own shapefile (Cincy_SewageTreatmentSystems_Active.shp).
2. Under Data Management Tools > General, use the merge tool to merge Cincy_SewageTreatmentSystems_Active.shp and HamCo_SewageTreatmentSystems_Active.shp and save as HamCo_SepticSystems.shp
3. Follow these instructions https://support.esri.com/en/technical-article/000008599#:~:text=In%20the%20attribute%20table%20of,Count%20%3D%20%2C%20and%20click%20OK using HamCo_SepticSystems and CBG_Sheds_HamMont_v3.shp and then save new file as CBG_Sheds_HamMont_v4.shp

1.5 Meteorological Data

Ambient temperature and precipitation data was obtained using the RNOAA package and the following R-code:

```
SewershedData<-read.csv("~/Rworking/SARS-CoV-2/GISdata/SewershedData_nogeom.csv")
#options(noaakey = "OhOlDoNELHhvByDIhuolnlOlGVPmDWpS")
wwtpdf<-SewershedData[c(1,5,6)]
names(wwtpdf)[names(wwtpdf)=="Sewershed"]<-"id"
names(wwtpdf)[names(wwtpdf)=="X_Centroid"]<-"longitude"
names(wwtpdf)[names(wwtpdf)=="Y_Centroid"]<-"latitude"

radiusVal=60 #pull stations within 20 km
nStations=30 # only use 30 nearest of the stations to draw data from.
fromdate=as.Date("2020-3-15")
todate=as.Date("2021-10-01")

library(rnoaa)
station_data <- ghcnd_stations() #load station data for use
nearby_stations <- meteo_nearby_stations(lat_lon_df =
wwtpdf,station_data=station_data,limit=nStations) # find stations within a radius of 20 km from
the wwtps (large enough to reduce NAs, small enough to capture local trends)
```

```
#get the precipitation data from december 2018 to december 2020
idnms<-names(nearby_stations) # get the station names
stations<-c() # create empty vector
origIds<-c() # create empty vector
noWWTPs=length(wwtpdf$id) #number of WWTPs
noDays=as.numeric(todate-fromdate)+1 #number of days
tmpval<-matrix(data=NA,nrow=noWWTPs,ncol=noDays) # create empty matrix with 18 rows
(no. of stations) and 732 columns (no. of days)
for (i in 1:length(idnms)) {
  stationIDs<-nearby_stations[[idnms[i]]]$id # get station IDs for those within distance of wwtp
id name
  placeid<-idnms[i] #store wwtp id
  dists<-nearby_stations[[idnms[i]]]$distance
  for (j in stationIDs) {
    newstation<-j # store the station id
    stations<-c(stations,newstation) # append station ids in list (before grouped by wwtp id)
    origIds<-c(origIds,placeid) # relate the wwtp id with station ids they are related to
  }
  alldata<-meteo_tidy_ghcnd(stationIDs,date_min = fromdate,date_max = todate,var = "tmin")
  alldata$tmin=alldata$tmin/10

## use inverse distance weighted approach to estimate the ambient temperature for each
sewershed by the WWTP location
  udates<-unique(alldata$date) # find all the unique days
  for (k in 1:length(udates)) {
    idx<-alldata$date==udates[k] # for each unique day index those days
    ustations<-unique(alldata$id[idx])
    idxStations<-nearby_stations[[idnms[i]]]$id %in% ustations
    dists2<-dists[idxStations]
    temp<-alldata$tmin[idx] # get the precipitation data from all stations for those days for that
wwtp ID
    idxna<-is.na(temp) # find NA values from the precipitation data from all stations for those days
for that wwtp ID
    A=temp*exp(-3*dists2/80)
    B=exp(-3*dists2/80)
    tmpval[i,k]<-sum(A,na.rm=TRUE)/sum(B,na.rm=TRUE) # inverse distance weighted
approach...stations within 10 km will have most influence
    #prcpval[i,k]<-mean(temp,na.rm=TRUE) # find the mean precipitation for that day from all
stations for that wwtp ID
  }

}
## replace NAs with mean value for that period of time (for temperature I think ok.)
for (i in 1:NCOL(tmpval)){
  idxna=is.na(tmpval[,i])
```

```
  val=mean(tmpval[,i],na.rm=TRUE)
  tmpval[idxna,i]=val
}

tmp<-as.character(udates) #store the unique dates as character class for assigning df column
names
colnames(tmpval)<-tmp # assign column names
rownames(tmpval)<-idnms # assign wwtp ID as rownames
tmpdf<-data.frame(tmpval) # make dataframe
colnames(tmpdf)<-colnames(tmpval)

data.table::fwrite(tmpdf, "~/Rworking/SARS-CoV-2/GISdata/OhioWW18SS_Temp_v2.csv")

wwtpdf<-SewershedData[c(1,5,6)]
names(wwtpdf)[names(wwtpdf)=="Sewershed"]<-"id"
names(wwtpdf)[names(wwtpdf)=="X_Centroid"]<-"longitude"
names(wwtpdf)[names(wwtpdf)=="Y_Centroid"]<-"latitude"

radiusVal=20 #pull stations within 20 km
nStations=20 # only use 6 nearest of the stations to draw data from.
fromdate=as.Date("2020-3-15")
todate=as.Date("2021-10-01")

library(rnoaa)
station_data <- ghcnd_stations() #load station data for use
nearby_stations <- meteo_nearby_stations(lat_lon_df =
wwtpdf,station_data=station_data,limit=nStations) # find stations within a radius of 20 km from
the wwtps (large enough to reduce NAs, small enough to capture local trends)


#get the precipitation data from december 2018 to december 2020
idnms<-names(nearby_stations) # get the station names
stations<-c() # create empty vector
origIds<-c() # create empty vector
noWWTPs=length(wwtpdf$id) #number of WWTPs
noDays=as.numeric(todate-fromdate)+1 #number of days
prcpval<-matrix(data=NA,nrow=noWWTPs,ncol=noDays) # create empty matrix with 18 rows
(no. of stations) and 732 columns (no. of days)
for (i in 1:length(idnms)) {
  stationIDs<-nearby_stations[[idnms[i]]]$id # get station IDs for those within distance of wwtp
id name
  placeid<-idnms[i] #store wwtp id
  dists<-nearby_stations[[idnms[i]]]$distance
  for (j in stationIDs) {
    newstation<-j # store the station id
    stations<-c(stations,newstation) # append station ids in list (before grouped by wwtp id)
    origIds<-c(origIds,placeid) # relate the wwtp id with station ids they are related to
```

```
  }
  alldata<-meteo_pull_monitors(stationIDs,var="PRCP",date_min = fromdate, date_max =
todate) # get the data for the 732 days for each station

  udates<-unique(alldata$date) # find all the unique days
  for (k in 1:length(udates)) {
    idx<-alldata$date==udates[k] # for each unique day index those days
    ustations<-unique(alldata$id[idx])
    idxStations<-nearby_stations[[idnms[i]]]$id %in% ustations
    dists2<-dists[idxStations]
    temp<-alldata$prcp[idx]/10 # get the precipitation data from all stations for those days for that
wwtp ID
    idxna<-is.na(temp) # find NA values from the precipitation data from all stations for those days
for that wwtp ID
    A=temp*exp(-3*dists2/10)
    B=exp(-3*dists2/10)
    prcpval[i,k]<-sum(A,na.rm=TRUE)/sum(B,na.rm=TRUE) # a kind of inverse distance
weighted approach...stations within 10 km will have most influence
    #prcpval[i,k]<-mean(temp,na.rm=TRUE) # find the mean precipitation for that day from all
stations for that wwtp ID
  }

}

tmp<-as.character(udates) #store the unique dates as character class for assigning df column
names
colnames(prcpval)<-tmp # assign column names
rownames(prcpval)<-idnms # assign wwtp ID as rownames
prcpdf<-data.frame(prcpval) # make dataframe
colnames(prcpdf)<-colnames(prcpval)

data.table::fwrite(prcpdf, "~/Rworking/SARS-CoV-2/GISdata/OhioWW18SS_Precip_v2.csv")
```

1.6 Mobility Data

Mobility data was obtained freely through https://www.google.com/covid19/mobility/ and then appended to our dataset with the following code:

```
#### load libraries ####
library(lubridate)
library(ggplot2)

##### load data ####
crAssFull<-read.csv("~/Rworking/SARS-CoV-2/GISdata/crAssphageFull_nogeom.csv")

# mobility data obtained from https://www.google.com/covid19/mobility/
M20<-read.csv("~/Rworking/Stormwater/crAssphage/2020_US_Region_Mobility_Report.csv")
```

```
M21<-read.csv("~/Rworking/Stormwater/crAssphage/2021_US_Region_Mobility_Report.csv")
M22<-read.csv("~/Rworking/Stormwater/crAssphage/2022_US_Region_Mobility_Report.csv")


##### subset data to Ohio ####
# get date in correct format

M20=M20[M20$sub_region_1=="Ohio",]
M20$date<-mdy(M20$date)
M21=M21[M21$sub_region_1=="Ohio",]
M21$date<-mdy(M21$date)
M22=M22[M22$sub_region_1=="Ohio",]
M22$date<-mdy(M22$date)

##### append all data together ####

mData<-rbind(M20,M21)
mData<-rbind(mData,M22)

#### subset by the crAssphage dataset dates #####

crAssFull$Date=as.Date(crAssFull$Date)
minDat<-min(crAssFull$Date)
maxDat<-max(crAssFull$Date)

mData2<-mData[mData$date>=minDat & mData$date<=maxDat, ]


##### separate out by Hamilton v Montgomery counties ####

mDataHam<-mData2[mData2$sub_region_2=="Hamilton County",]
mDataMont<-mData2[mData2$sub_region_2=="Montgomery County",]
mData3<-rbind(mDataHam,mDataMont)

#### get the dates and the county's data to line up with crAssphage data ####

idx=(crAssFull$Sewrshd=="LITTLE MIAMI"| crAssFull$Sewrshd=="MILL
CREEK"|crAssFull$Sewrshd=="MUDDY CREEK"|crAssFull$Sewrshd=="TAYLOR CREEK")
crAssFull$County=NA
crAssFull$County[idx==1]="Hamilton County"
crAssFull$County[idx==0]="Montgomery County"

#### append the data to the table #####
crAssFull2=merge(crAssFull,mData3, by.x=c("County","Date"), by.y=c("sub_region_2","date"))

#### eliminate unused columns ####
```

```
crAssFull2=crAssFull2[,-c(45:53,66:67,90:91,93)]

#### rename columns with long names ####

colnames(crAssFull2)[80:85]=c("retailrecdP","grocpharmdP","parksdP","transitdP","workdP","resdP")
data.table::fwrite(crAssFull2,"~/Rworking/Stormwater/crAssphage/crAssphageFullv2_nogeom.csv")
```

1.7 Wastewater temperature data

Wastewater temperatures were obtained through MSD and processed with the following R code:

```
library(readxl)
#### functions ####
GetCongruentCharStr<-function(charstrchange,charstrdesired){
  charstrchange=toupper(charstrchange)
  nmVex<-vector(mode="character",length=length(charstrchange))
  for (i in 1:length(charstrdesired)) {
    idx=grep(charstrdesired[i],charstrchange)
    nmVex[idx]=charstrdesired[i]
  }
  return(nmVex)
}

#### load data ####
WWTPdata<- read_excel("Rworking/SARS-CoV-2/WastewaterSampleData/20220228_Covid19WastewaterSample.xlsx",
sheet = "Sample_Collection", col_types = c("text", "text", "text", "text", "text", "text", "text",
"numeric", "numeric", "text",
"text", "numeric", "numeric", "numeric", "numeric", "numeric", "numeric", "numeric", "text",
"text", "text",
 "text", "text", "text", "text", "text", "text"))

crAssTmp<-read.csv("~/Rworking/SARS-CoV-2/GISdata/crAssphageFullv2_nogeom.csv")


idx1=is.na(WWTPdata$WQ_TOTHER)
WWTPdata$WQ_TOTHER[idx1]=WWTPdata$WQ_TMAX[idx1]

tmptbl=WWTP[is.na(WWTPdata$WQ_TOTHER),]

WWTPdata$Facility_Name=GetCongruentCharStr(WWTPdata$Facility_Name,crAssTmp$Sewrshd)

WWTPdata=WWTPdata[!(WWTPdata$Facility_Name==""),]

mm<-substr(WWTPdata$Collection_Date,1,2)
```

```
dd<-substr(WWTPdata$Collection_Date,3,4)
yyyy<-substr(WWTPdata$Collection_Date,5,9)
mmddyyyy<-paste(yyyy,mm,dd,sep="-")
WWTPdata$Collection_Date=as.Date(mmddyyyy)

#### Relate most recent date's temperature measurement to crAssphage measure ####

crAssTmp$Date=as.Date(crAssTmp$Date)
dtj=WWTPdata$Collection_Date
sRj=WWTPdata$Facility_Name
wwdata=WWTPdata$WQ_TOTHER
wwTempdata=data.frame(sRj,dtj,wwdata)

newdata=merge(crAssTmp,wwTempdata,by.x=c("Sewrshd","Date"), by.y=c("sRj","dtj"))


#### replace missing data with averages and remove outlying data (there are 2 based on the plots)

newdata$wwdata[max(newdata$wwdata, na.rm=TRUE)==newdata$wwdata]=NA
newdata$wwdata[min(newdata$wwdata, na.rm=TRUE)==newdata$wwdata]=NA

tmpval=aggregate(wwdata~Date, newdata, mean)

for (i in 1:length(newdata$wwdata)){
  d1=newdata$Date[i]
  idxWW=min(abs(outer(tmpval$Date,d1,"-")))==abs(outer(tmpval$Date,d1,"-"))
  if ((is.na(newdata$wwdata[i]))==TRUE) {
    newdata$wwdata[i]=tmpval$wwdata[idxWW]
  }
}

colnames(newdata)[colnames(newdata)=="wwdata"]="wwT"

data.table::fwrite(newdata,"~/Rworking/Stormwater/crAssphage/crAssphageFullv3_nogeom.csv"
)
```

1.8 Other wastewater treatment plant data

Flow, pH, the sewer type (i.e., combined or separate), estimated industrial flow to each wastewater treatment plant, and the Euclidean distance from each census block group centroid to each wastewater treatment plant were obtained using a file from MSD and the following code:

```
# Load and Define
```{r setup, message=FALSE, warning=FALSE}
rm(list = ls())
```
```

```
## Packages
library(data.table)
library(dtplyr)
library(readxl)
library(lubridate)
library(units)
library(sf)
library(lwgeom)
library(tidycensus)
library(tidyverse)
library(rgdal)
library(pracma)
library(reprex)
library(sp)
#library(EnvStats)

## Options
options(mc.cores = parallel::detectCores())
options(dplyr.summarise.inform = FALSE)

GetCongruentCharStr<-function(charstrchange,charstrdesired){
  charstrchange=toupper(charstrchange)
  nmVex<-vector(mode="character",length=length(charstrchange))
  for (i in 1:length(charstrdesired)) {
   idx=grep(charstrdesired[i],charstrchange)
   nmVex[idx]=charstrdesired[i]
  }
  return(nmVex)
}

substrRight <- function(x, n){
  substr(x, nchar(x)-n+1, nchar(x))
}

eucdist<- function(x1, x2) sqrt(sum((x1 - x2) ^ 2))

coord2dist<-function(x1,x2){
  dist<-NULL
  for(i in 1:NROW(x1)) {
    dist[i]<-eucdist(x1[i],x2)
  }
  return(dist)
}

SubsetByID<-function(refIDS, tarIDs, DatatoSubset){
  idx=vector("numeric",NROW(tarIDs))
    for (i in 1:NROW(tarIDs)){
```

```
    idx[i]=sum(refIDs==tarIDs[i])>0
   }
  DataOut<-DatatoSubset[idx==1,]
  return(DataOut)
}


## Residential inputs

### Load the shapefile data
filename1<-"~/Rworking/Stormwater/GIS_data/crAssphage/CBG_Sheds_HamMont_v3.shp"
SewershedData<-read_sf(dsn=filename1)

filename2<-"~/Rworking/Stormwater/GIS_data/OhioWWsamplingDashCoords.shp"
WWTPs<-read_sf(dsn=filename2)

### Get coordinates and transform them to stateplane
sCBG<-matrix(c(SewershedData$X_Centroid,SewershedData$Y_Centroid),
NROW(SewershedData),2)
sWWTP<-matrix(c(WWTPs$X, WWTPs$Y),NROW(WWTPs),2)
CBGdf<-
data.frame(SewershedData$Sewershed,SewershedData$Xc_BG,SewershedData$Yc_BG)
WWTPdf<-data.frame(WWTPs$Facility_N,WWTPs$X,WWTPs$Y)
colnames(CBGdf)=c("ID","X","Y")
colnames(WWTPdf)=c("ID","X","Y")

## rename the WWTPdf IDs
idxWrongEastern=(WWTPdf$ID=="Eastern Ohio RWA")
WWTPdf<-WWTPdf[!idxWrongEastern,] ## HARD CODED.
unms<-unique(CBGdf$ID)
nms<-GetCongruentCharStr(WWTPdf$ID,unms)
idxno=nms==""
WWTPdf$ID=nms
WWTPdf<-WWTPdf[!idxno,]



CBGdf.sp<-CBGdf
WWTPdf.sp<-WWTPdf

coordinates(CBGdf.sp)<-~X+Y
coordinates(WWTPdf.sp)<-~X+Y

proj4string(CBGdf.sp)<-CRS("+init=epsg:3735")

proj4string(WWTPdf.sp)<-CRS("+init=epsg:4326")
WWTPdf.sp<-spTransform(WWTPdf.sp,CRS("+init=epsg:3735"))
```

```
### plot sewersheds, census block centroids, and wwtps to confirm projections

filename3<-"~/Rworking/SARS-CoV-2/GISdata/SewershedData_v2.shp"
SewershedDataG<-read_sf(dsn=filename3)

refIDs<-WWTPdf$ID
tarIDs<-SewershedDataG$Sewrshd
ShedDataG<-SubsetByID(refIDS,tarIDs,SewershedDataG)

sf::st_crs(ShedDataG)<-CRS("+init=epsg:4326")
ShedDataG<-st_transform(select(ShedDataG,geometry),3735)

library(ggplot2)
library(ggspatial)
ggplot(data=NULL)+geom_sf(data=ShedDataG)+geom_point(data=NULL,aes(x=CBGdf.sp@co
ords[,1],y=CBGdf.sp@coords[,2]),color="black",size=1)+geom_point(data=NULL,
aes(x=WWTPdf.sp@coords[,1],y=WWTPdf.sp@coords[,2]),color="red",size=2)

ggplot(data=NULL)+geom_point(data=NULL,aes(x=CBGdf.sp@coords[,1],y=CBGdf.sp@coord
s[,2]),color="black",size=1)+geom_point(data=NULL,
aes(x=WWTPdf.sp@coords[,1],y=WWTPdf.sp@coords[,2]),color="red",size=2)

### Get the distance from each CBG to each WWTP

dists=matrix(data=NA,nrow=NROW(CBGdf.sp@coords),ncol =NROW(WWTPdf.sp@coords))
for (i in 1:NROW(CBGdf.sp@coords)) {
  for (j in 1:NROW(WWTPdf.sp@coords)) {
    dists[i,j]=eucdist(CBGdf.sp@coords[i,],WWTPdf.sp@coords[j,])
  }
}
uwwtp=unique(WWTPdf.sp@data$ID)
dists2=vector("numeric",NROW(CBGdf))
for(i in 1:length(uwwtp)){
  idxwwtp<-(CBGdf.sp@data$ID==uwwtp[i])
  dists2[idxwwtp]<-dists[idxwwtp,i]
}
library(ggplot2)
ggplot(data=NULL)+geom_point(data=NULL,aes(x=CBGdf.sp@coords[,1],y=CBGdf.sp@coord
s[,2],color=exp(-
3*(dists2/3281)/1000000)))+scale_color_continuous(low="yellow",high="red")+geom_point(dat
a=NULL, aes(x=WWTPdf.sp@coords[,1],y=WWTPdf.sp@coords[,2]),color="black",size=2)

### Attach the distance values to census block group centroid
Distance is in feet
```

```
SewershedData$Dist2WWTP=dists2

### Attach the census block data to the block group centroid data
cbg_age<-read.csv("~/Rworking/SARS-CoV-2/GISdata/Census_2019_data/age.csv")
idxTot<-cbg_age$age=="tot"
cbg_pop<-cbg_age[idxTot,]
CBGshedData<-merge(SewershedData,cbg_pop,by.x="GEOID",by.y="bg_id")
CBGshedData=rename(CBGshedData,pop=n)
CBGshedData=select(CBGshedData,-age)
CBGshedData2=st_drop_geometry(CBGshedData)
data.table::fwrite(CBGshedData2, "~/Rworking/Stormwater/GIS_data/ShedData_nogeom.csv")

library(readxl)
wwdata <- read_excel("~/Rworking/SARS-CoV-
2/WastewaterSampleData/20211213wwdata.xlsx")
mm<-substr(wwdata$Collection_Date,1,2)
dd<-substr(wwdata$Collection_Date,3,4)
yyyy<-substr(wwdata$Collection_Date,5,9)
mmddyyyy<-paste(yyyy,mm,dd,sep="-")
wwdata$Collection_Date=as.Date(mmddyyyy)

badpH=wwdata$WQ_pH
idxRange=grep("-",badpH)
lpH=as.numeric(sub("-.*","",badpH[idxRange]))
upH=as.numeric(sub(".*-","",badpH[idxRange]))
goodpH=as.numeric(badpH)
goodpH[idxRange]=(lpH+upH)/2
idxNA=is.na(badpH)
goodpH[idxNA]=mean(goodpH[!idxNA])
wwdata$WQ_pH=goodpH

newnames<-GetCongruentCharStr(wwdata$Facility_Name,crAss2$rID)
idx=newnames==""
wwdata2<-wwdata[!idx,]
wwdata2$Facility_Name<-newnames[!idx]

cDates<-as.Date(crAssFull$Date)
fDates<-as.Date(wwdata2$Collection_Date)
uSheds<-unique(wwdata2$Facility_Name)



shed=NULL
cDatesq=NULL
flowval=NULL
flowdaysaway=NULL
df2=data.frame(shed,cDatesq,flowval,flowdaysaway)
```

```
for (j in 1:length(uSheds)){
  idxshed1=(crAssFull$Sewrshd==uSheds[j])
  idxshed2=(wwdata2$Facility_Name==uSheds[j])
  cDatesq=cDates[idxshed1]
  fDatesq=fDates[idxshed2]
  Qq=wwdata2$Flow_Volume[idxshed2]
  pHq=wwdata2$WQ_pH[idxshed2]
  Dtmp=outer(cDatesq,fDatesq,"-")
  pH=vector("numeric",length(cDatesq))
  flowval=vector("numeric",length(cDatesq))
  flowdaysaway=vector("numeric",length(cDatesq))
  for (i in 1:length(cDatesq)) {
    q=Dtmp[i,] #crAssphage dates to all the flow dates
    q2=abs(as.numeric(q))
    idxnear=(min(q2,na.rm=TRUE)==q2)
    flowdaysaway[i]=min(q2)
    if ((flowdaysaway[i]>=10)==TRUE){
    flowval[i]=mean(Qq,na.rm=TRUE)
    pH[i]=mean(pHq)
    }
    if ((flowdaysaway[i]<10)==TRUE){
      flowval[i]=Qq[idxnear]
      pH[i]=pHq[idxnear]
    }
  }
  shed=crAssFull$Sewrshd[idxshed1]
  df1=data.frame(shed,cDatesq,pH,flowval,flowdaysaway)
  df2=bind_rows(df2,df1)
}
crAssFull$Date=as.Date(crAssFull$Date)
crAssFullB<-merge(crAssFull,df2,by.x=c("Sewrshd","Date"),by.y=c("shed","cDatesq"))

data.table::fwrite(crAssFullB,"~/Rworking/SARS-CoV-2/GISdata/crAssphageFull_nogeom.csv")
datatest<-read.csv("~/Rworking/SARS-CoV-2/GISdata/crAssphageFull_nogeom.csv")

## Add combined sewer info and other wwtp-level data (not temporal)

OHIODashWWTP<- read_excel("~/Rworking/SARS-CoV-
2/WastewaterSampleData/20211213wwtpdata.xlsx")
newnames<-GetCongruentCharStr(OHIODashWWTP$Facility_Name,crAss2$rID)
idx=newnames==""
OHdashWWTP2<-OHIODashWWTP[!idx,]
OHdashWWTP2$Facility_Name<-newnames[!idx]

OHdashdf<-OHdashWWTP2[,c(3,16,18,20)]

crAssFullC<-merge(crAssFullB,OHdashdf,by.x="Sewrshd",by.y="Facility_Name")
```

```
crAssFullC$Industrial_Flow=as.numeric(crAssFullC$Industrial_Flow)

data.table::fwrite(crAssFullC,"~/Rworking/SARS-CoV-2/GISdata/crAssphageFull_nogeom.csv")
```

1.9 Travel time estimation and relating to census block groups

Travel time estimation

The travel time, t_{ij}, from each spatially distributed population location *j* to the location of each i^{th} sample associated with a unique WWTP was obtained from running MSD's collection system models for the Mill Creek, Muddy Creek and Little Miami WWTPs. The models are built using U.S. EPA's Storm Water Management Model (SWMM),[1] and adaptations were made specifically for this study by adding in pollutant mass routing. In their original development, the models have dry weather sanitary sewage flow inputs distributed spatially across many of the system nodes. Each sanitary sewage input node in the model was tagged according to the census block group in which it resided, and a unique tracer species was defined for each census block group. Pulses of tracer mass were released at specified times in the model, and the travel time to the WWTPs was determined by the difference in the centroid of the mass pulse upon arrival at the WWTP and its release point. The process was repeated at six-hour intervals over the course of a day to evaluate the effect of diurnal variations in sewage flow on travel time.

Travel times ranged from a minimum of about 40 minutes to just over 19 hours, depending on the distance from the respective WWTP. Travel times for pulses released at 02:00 were consistently greater than other times of the day; however, in most instances the range of travel times (that is, the longest minus the shortest) was between 10% and 20% of the average time for all four pulses. In some areas, far from treatment plants and served entirely by lift stations, the range was larger, especially for the 2:00 release when lift station operation would be more intermittent. Overall, however, it appeared that an average of the four travel times from each census block group was sufficient to characterize the spatial variation and was successful in capturing how the details of sewer system layout differ from Euclidian distances.

Relating to census block groups

This is the file HamCo_TravelTimes.R that is referenced in the driver code (section S10)

```
#### load packages #####

library(MASS)

library(dplyr)

library(tidyr)

library(sf)
```

```
#### load data #####

library(readxl)

TTs <-
read_excel("~/Rworking/Stormwater/crAssphage/Hamilton_CBG_centroids_w_modeled_TT.xls
x")

CBGdata<-read.csv("~/Rworking/Stormwater/crAssphage/ShedDatav2_nogeom.csv")

filename2<-"~/Rworking/SARS-CoV-2/GISdata/CBG_Sheds_HamMont_v4.shp"

SepData<-read_sf(dsn=filename2)

SepData<-st_drop_geometry(SepData)

X=SepData$Xc_BG

Y=SepData$Yc_BG

SepCount=SepData$Sum_Count

Sewershed=SepData$Sewershed


SepData<-data.frame(X,Y,Sewershed,SepCount)


##### data cleaning and merging ####

##clean up the travel time data and merge with CBG data to compare the Euclidean

## distance with travel times

CBGdata=merge(CBGdata,SepData,by.x=c("Xc_BG","Yc_BG","Sewershed"),by.y=c("X","Y","
Sewershed"))



idxna=is.na(TTs$`Average Travel Time (hours)`)

TTs<-TTs[!idxna,]

TTs2<-TTs[,c(3,20:24)]

colnames(TTs2)<-c("GEOID_Data","TT2pulse","TT8pulse","TT14pulse","TT20pulse","TTavg")

TTs2$TTid=1:NROW(TTs2)


CBGdataHAM<-CBGdata[CBGdata$COUNTYFP==61,]

CBGdataHAM$CBGID=1:NROW(CBGdataHAM)

CBGdataHAM2<-CBGdataHAM
```

```
ShedData<-merge(CBGdataHAM,TTs2,by="GEOID_Data")
ShedData2<-ShedData
ShedData2<-ShedData2[,1:44]
ShedData2$TTavg=ShedData$TTavg



xBG<-CBGdataHAM$Xc_BG
yBG<-CBGdataHAM$Yc_BG

xTT<-ShedData$Xc_BG
yTT<-ShedData$Yc_BG

uGEOIDdata<-unique(CBGdataHAM$GEOID_Data)

idxMiss=vector("logical",length(uGEOIDdata))
for (i in 1:length(uGEOIDdata)){
  GEOID_Data<-uGEOIDdata[i]
  idxMiss[i]<-sum((GEOID_Data==ShedData$GEOID_Data))==0 #index the missing CBGs
  if (idxMiss[i]==TRUE) {
    idxCBG<- GEOID_Data==CBGdataHAM$GEOID_Data   #index the original file rows
    CBGID<-CBGdataHAM$CBGID[idxCBG]
    xCBGidx<- xBG[idxCBG]# the X value of the centroid of the CBG in question
    yCBGidx<- yBG[idxCBG]# the y value of the centroid of the CBG in question
    dii<- sqrt((xTT-xCBGidx)^2+(yTT-yCBGidx)^2) #get distance vector between that CBG
centroid and the travel time file centroids
    idxShed<- (min(dii)==dii)#index the output file rows that are nearest to those
    TTavg<- ShedData$TTavg[idxShed] # get the travel time of the nearest neighbor CBG
    dftmp<-data.frame(GEOID_Data,CBGID,TTavg)
    dftmp<-merge(dftmp,CBGdataHAM2, by=c("GEOID_Data","CBGID"))
    ShedData2=rbind(ShedData2,dftmp)
  }
```

```
}

ShedData<-ShedData2

rm(TTs,CBGdata,idxna,TTs2,CBGdataHAM,ShedData2,dftmp,TTavg,idxShed,dii,yCBGidx,xCBGidx,CBGID,idxCBG,Sewershed,idxMiss,i,GEOID_Data,uGEOIDdata,xTT,yTT, xBG, yBG, CBGdataHAM2,SepData,GEOID,SepCount)
```

1.10 Map summaries of travel time, population, commercial parcel county, and septic systems by census block group

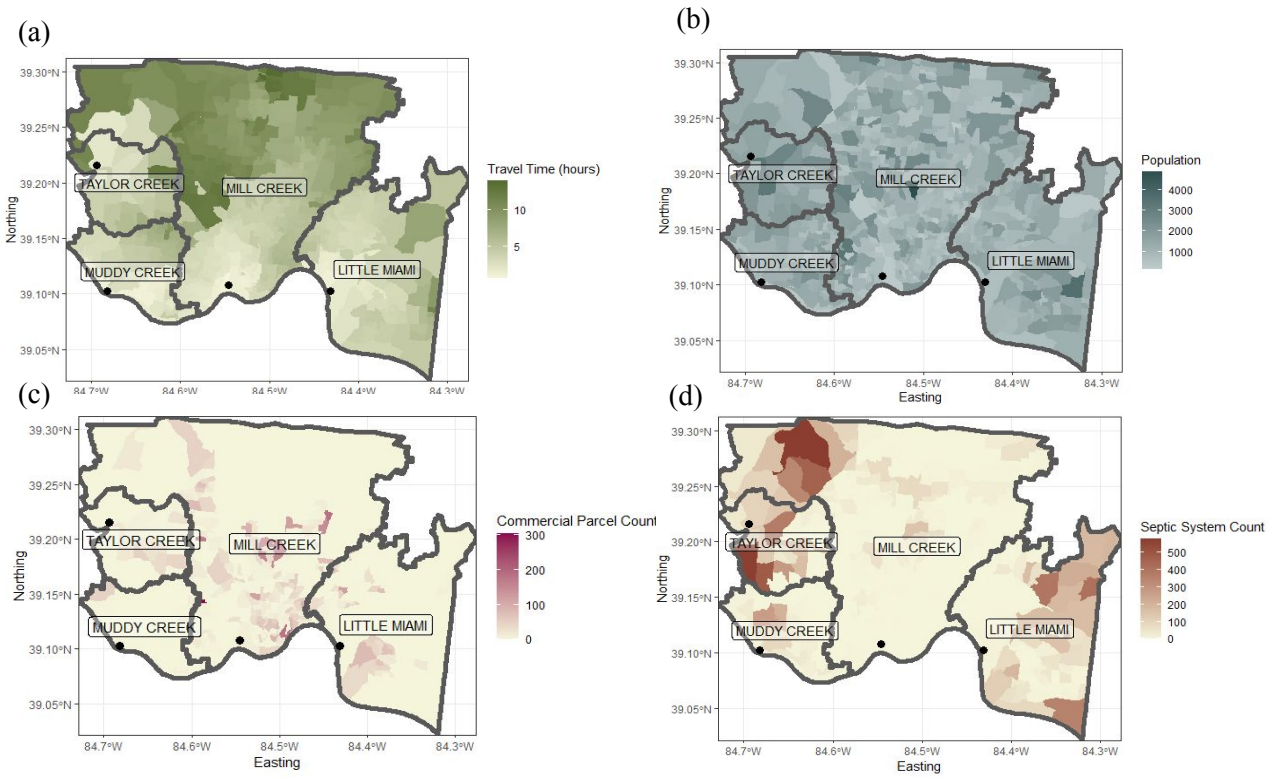

Figure S1 Data available from census block group level (a) travel time to wastewater treatment plant (b) population (b) number of commercial parcels (representative of potential retail) and (d) the septic system count

Figure S1 provides some of the databases to help represent spatially distributed sources for the *Find* stage of FIT. We only display here the commercial parcel count for interest.

S2. Details on crAssphage quantification results

Figure S2 describes the crAssphage load over time and by wastewater treatment plant. Here we can see that Mill Creek has the greatest load, followed by Little Miami, and then both Muddy Creek and Taylor Creek appear similar.

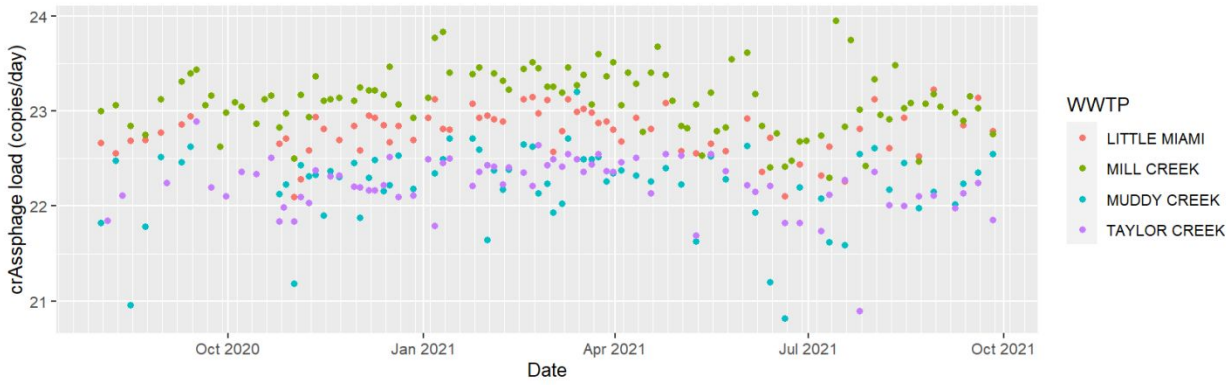

Figure S2 The crAssphage load over time by WWTP (or sewershed)

S3. Summary of the literature of RNA and DNA microbial marker decay in wastewater

Table S1 Studies of RNA and DNA decay in wastewater

Authors	*Publication Date*	*Access date*	*Microbial markers*	*Observed Decay as T90*	*Notes*
Ballesté, E; García-Aljaro, C; Blanch, A R	8/1/2018	10/15/2018	FIB, Bacteroidales species MST markers (DifHM, BifCW, BifPL, HF183/BFD, Rum2Bac, and Pig2Bac)	E. coli: 1.52-5.69 days (summer), 2.06-6.19 days (winter); enterocci: 1.15-3.10 days (summer), 1.05-1.91 days (winter) MST markers: 1.05-1.91 days (summer), 2.90-6.12 days (winter)	

Ahmed, Warish; Bertsch, Paul M; Bibby, Kyle; Haramoto, Eiji; Hewitt, Joanne; Huygens, Flavia; Gyawali, Pradip; Korajkic, Asja; Riddell, Shane; Sherchan, Samendra P; Simpson, Stuart L; Sirikanchana, Kwanrawee; Symonds, Erin M; Verhagen, Rory; Vasan, Seshadri S; Kitajima, Masaaki; Bivins, Aaron	2020-12	9/1/2021	SARS-CoV-2 RNA and murine hepatitis virus (MHV) measured with RT qPCR	SARS-CoV-2 RNA: 8.04 to 27.8 days; MHV RNA: 7.44 to 56.6 days	SARS-CoV-2 showed less sensitivity to elevated temperatures
Shi, Jiahua; Li, Xuan; Zhang, Shuxin; Sharma, Elipsha; Sivakumar, Muttucumaru; Sherchan, Samendra P; Jiang, Guangming	3/20/2022	9/19/2023	Human infection coronavirus 229E (HCoV-229E) and feline infectious peritonitis virus (FIPV)	99% of HCoV-229E and FIPV decayed within 2 hours under either gravity sewer or rising main sewers (pressurized sewers). Found limited difference in the decay of HCoV and FIPV in reactors operated as RM or GS with T90 differences of 7-10 minutes.	In-sewer decay

Zhang, Shuxin; Shi, Jiahua; Sharma, Elipsha; Li, Xuan; Gao, Shuhong; Zhou, Xu; O'Brien, Jake; Coin, Lachlan; Liu, Yanchen; Sivakumar, Muttucumaru; Hai, Faisal; Jiang, Guangming	4/15/2023	9/19/2023	C. jejuni and C. coli	C. jejuni: Ci95% 18.62 to 23.14 hours (rising main sewer, with biofilms), Ci95% 15.49 to 35.8 hours (rising main sewer, without biofilms), Ci95% 17.09 to 48.91 hours (gravity sewer, with biofilms), Ci95% 6.68 to 9.82 hours (gravity sewer, without biofilms); C. coli: Ci95% 5.58 to 7.67 hours (rising main sewer, with biofilms), Ci95% 2.97 to 9.01 hours (rising main sewer, without biofilms), Ci95% 7.20 to 13.95 hours (gravity sewer, with biofilms), Ci95% 1.70 to 2.27 hours (gravity sewer, without biofilms)	Biofilms play important role in decay processes
Guo, Ying; Sivakumar, Muttucumaru; Jiang, Guangming	5/1/2022	9/19/2023	*Campylobacter, Salmonella, Norovirus, Adenovirus*	For *Campylobacter, Salmonella, Norovirus, Adenovirus*, respectively k_20 reported with Arrhenius equation*:* 1.03, 1.13, 1.05, 1.23. which translates to 24.5 hours, 22.3 hours, 24 hours, 20.5 hours in terms of T90	"Stormwater dilution of domestic wastewater (i.e., sewer inflow might achieve 10 times volumetric dilution) was shown to play a role in increasing the sensitivity of WBE back-calculation to bacterial pathogens, but not viral pathogens. Hence, WBE back-calculation in real sewers should account for in-sewer decay of specific pathogen species under different wastewater temperatures and dilutions"

S4. Variable, parameter, and hyperparameter summary

4.1 Descriptive statistics

The average daily flow by sewershed (Table S2) is ranked with 1) Mill Creek having the greatest daily flow, followed by 2) Little Miami, 3) Muddy Creek, and 4) Taylor Creek. Our population estimates in the catchments (See Table S3) are ranked as 1) Mill Creek having the largest population, followed by 2) Little Miami, 3) Muddy Creek, and 4) Taylor Creek. The Greater Cincinnati Metropolitan Sewer District (MSD) provided some estimates of the population served in each catchment which are ranked in population as 1) Mill Creek having the largest, followed by 2) Taylor Creek, 3) Little Miami, and 4)

Muddy Creek. We note that in rank our population estimates correlate better with flow than those population estimates that the MSD provides, which was noted to be potentially out of date.

The average crAssphage load by sewershed (Table S2) is ranked as 1) Mill Creek having the greatest load, followed by 2) Little Miami, 3) Muddy Creek, and 4) Taylor Creek. Other details to note are that Taylor Creek is the only system that is a separated sewer system and had the largest estimate of people on septic systems compared to other sewersheds. Mill Creek has the greatest inputs of industrial flows and the wastewater is the warmest temperature on average.

Table S2 Descriptive statistics for the responses and LASSO regression variables overall and by sewershed

Data type	Data (s- spatial, t-temporal)	Mill Creek mean (std)	Taylor Creek mean (std)	Little Miami mean (std)	Muddy Creek mean (std)	**All mean (std)**
Response	Daily flow (s,t) in million gallons per day (MGD)	9.94e7 (4.40e7)	2.79e6 (8.39e6)	3.35e7 (9.71e6)	1.18e7 (4.15e6)	**4.27e7 (4.78e7)**
	crAssphage concentration (s,t) in copies-per-liter	4.90e8 (3.67e8)	1.99e9 (1.24e9)	5.96e8 (3.37e8)	5.98e8 (4.93e8)	**8.84e8 (9.18e8)**
	crAssphage load (s,t)	1.71e23 (1.41e23)	2.01e22 (1.15e22)	7.01e22 (3.51e22)	2.37e22 (2.07e22)	**8.05e22 (1.05e23)**
	Log10 crAssphage load (s,t)	23.1 (0.330)	22.2 (0.286)	22.8 (0.247)	22.2 (0.411)	**22.6 (0.509)**
Sewershed characteristic predictors ($x^{(w)}$)	Industrial flow (s) in MGD	9	0	1.5	0.1	**3.25 (4.01)**
	Daily wastewater temperature (s,t) in degrees Celsius	19.0 (3.81)	18.1 (3.77)	17.9 (3.65)	17.8 (3.65)	**18.3 (3.75)**
	Wastewater pH (s,t)	7.24 (0.217)	7.28 (0.160)	7.32 (0.171)	7.18 (0.385)	**7.26 (0.252)**
	Sewer type information [combined = 1, separate=0] (s)	1	0	1	1	**0.769 (0.422)**
	Daily ambient temperature (s,t) in degrees Celsius	7.72 (9.51)	5.32 (9.57)	6.48 (9.83)	6.51 (9.81)	**6.62 (9.66)**
	48-hour precipitation (s,t) in mm	3.17 (4.94)	2.71 (5.53)	2.86 (4.78)	2.68 (5.72)	**2.88 (5.21)**
	Percent time spent at retail and	-12.2 (8.92)	-14.3 (9.41)	-14.2 (9.16)	-14.6 (9.51)	**-13.7 (9.23)**

	recreation compared to a baseline (t)					
	Percent time spent at grocery or pharmacy compared to a baseline (t)	-5.83 (6.79)	-7.72 (6.67)	-8.01 (6.44)	-8.08 (6.63)	**-7.26 (6.69)**
	Percent time spent on transit compared to a baseline (t)	-32.5 (9.24)	-34.8 (9.56)	-34.0 (10.7)	-34.0 (10.6)	**-33.7 (9.93)**
	Percent time spent at work compared to a baseline (t)	-23.7 (8.16)	-23.5 (8.89)	-21.4 (8.20)	-21.6 (8.29)	**-22.6 (8.40)**
	Percent time spent at home compared to a baseline (t)	5.38 (4.26)	5.63 (4.82)	4.80 (4.66)	4.93 (4.80)	**5.21 (4.59)**

Table S3 Summary of census block group populations within sewershed catchments

Sewershed Name	Number of Census Block Groups included in analysis (average percent in catchment)	Number of those Census Block Groups overlapping the sewershed catchment boundary (average percent in catchment)	Average original population count in each Census Block Group (min, max)	Average catchment-adjusted population count in each Census Block Group (min, max)	Estimate of the number of people on septic systems in each sewershed catchment	Total number of people in the sewershed catchment	Greater Cincinnati Metropolitan Sewer District Population Served Estimate
Mill Creek	453 (0.702)	246 (0.552)	1,140 (149,4,864)	791 (1, 4864)	4,880	509,000	488,000
Taylor Creek	42 (0.397)	33 (0.340)	1,560 (384, 3,264)	667 (1, 3264)	7,960	520,00	143,000
Little Miami	154 (0.709)	79 (0.549)	1,060 (183, 3,600)	794 (1, 3600)	4,050	164,000	76,000
Muddy Creek	82 (0.508)	54 (0.409)	1,230 (361, 2436)	592 (1, 2436)	1,480	97,400	34,000
Total	649 (0.650)	330 (0.504)	1,170 (149, 4,864)	752 (1, 4864)	18,400	82,4000	74,1000

4.2 Summary of variables, parameters, and hyperparameter

Table S4 Summary of variables, parameters, and hyperparameters

Name	Description
y_i	Estimated wastewater concentration of crAssphage (crAv056-copies-per-liter) for the i^{th} space/time sample
q_i	the daily flow q_i (liter-wastewater-per-day) for the i^{th} space/time sample

z_i	daily load of crAssphage ($\log_{10}$ copies-crAv056-per-day) for the i^{th} space/time sample
$x_i^{(w)}$	Sewershed characteristic predictors (see above table for greater detail and SI1
$s_i(\boldsymbol{\alpha})$	The general term representing the standardized source contribution at space/time sample *i* from upstream residential populations with a set of variables and hyperparameters $\boldsymbol{\alpha}$
α_p	the distance at which we would expect to see a *p*% loss of information from source locations
β_0	An intercept to the land-use regression model
β_1	the increase in the response for a 1 standard-deviation increase in the source term
β_w	the increase in the response for a 1 standard-deviation increase in the w^{th} sewershed characteristic terms
$sedc_i(\alpha_{90})$	The un-standardized source term from the SEDC SPM (i.e., using the sum of exponentially decaying contributions from upstream populations)
$s_i(\alpha_{90})$	The standardized source term from the SEDC SPM
$sedc_i(\alpha_{90,T},\gamma_1,T_i)$	The un-standardized source term from the SEDC-T SPM (i.e., using the sum of exponentially decaying contributions from upstream populations modified by wastewater temperature)
$s_i(\alpha_{90,T},\gamma_1,T_i)$	The standardized source term from the SEDC-T SPM
γ_1	$\gamma_1 < 0$ is a fitted hyperparameter that captures the exponential decrease of α_{90} with z-scored temperature
$\alpha_{90,T}$	Is the value of α_{90} at the average wastewater temperature (i.e., when $WWT_i = 0$) of 18.3 °C for this study (see SI3 for more descriptive statistics)
T_i	The z-scored wastewater temperature (with a mean of 0 and standard deviation of 1)
$sedc_i(\alpha_{90,TM},\gamma_1,T_i,\gamma_2,Mob_i)$	The un-standardized source term from the SEDC-TM SPM (i.e., using the sum of exponentially decaying contributions from upstream populations modified by wastewater temperature and the amount of time that people spend at home)
$s_i(\alpha_{90,TM},\gamma_1,T_i,\gamma_2,Mob_i)$	The standardized source term from the SEDC-TM SPM
γ_2	$\gamma_2 > 0$ is a fitted parameter that captures the exponential increase of α_{90} with the z-scored population mobility factor
$\alpha_{90,TM}$	Is the value of α_{90} at average wastewater temperature (i.e., when $WWT_i = 0$) and average population mobility conditions (i.e., when $Mob_i = 0$)
Mob_i	The z-scored mobility factor (mean of 0 and standard deviation of 1), which in our work is the percent of time spent at home compared to baseline
P_{0j}	The associated populations at spatially distributed locations, *j*, which in our work represent the population information of the population at their residences
$P^{(CBG)}$	The census block group population according to the 2019 Census (the 2020 U.S. Census product will not be available until September 2023)
$A_j^{(total)}$	The total area of the census block group in square feet

$A_j^{(sewered)}$	Represents the area of that census block group that is sewered in square feet
Sep_j	The number of septic systems in area $A_j^{(sewered)}$
$p^{hh} = 2.41$	The median persons per household for Ohio based on 2016-2020 Census data[2]
t_{ij}	The travel time from each spatially distributed population location j to the location of each i^{th} sample associated with a unique WWTP
D_{ij}	The Euclidean distance from each spatially distributed population location j to the location of each i^{th} sample associated with a unique WWTP
$r_{ij}^{(SPM)}$	The information loss rate for each location j for the i^{th} space/time sample with the resulting hyperparameters from different SPMs
$t_{ijk}{}'$	All the noisy iterations of T_{ij} from the k=1 to 100 simulations of normal random error ϵ_{jk} scaled by $0.1\hat{\sigma}_{T_{ij}}$
$z_{ik}{}'$	All the noisy iterations of z_i from the k=1 to 100 simulations of normal random error ϵ_{ik} scaled by $0.1\hat{\sigma}_{z_i}$
$P_{0jk}{}'$	All the noisy iterations of P_{0j} from the k=1 to 100 simulations of normal random error ϵ_{jk} scaled by $0.1\hat{\sigma}_{P_{0j}}$
$P_{0j}{}'$	Population counts that do not account for those using septic that are not connected to the sewer network
$sedc_i(\alpha_{D90})$	The un-standardized source term from the SEDC SPM (i.e., using the sum of exponentially decaying contributions from upstream populations) but using Euclidean distance D_{ij} instead of travel time T_{ij}
$s_i(\alpha_{D90})$	The standardized source term from the SEDC SPM but using Euclidean distance D_{ij} instead of travel time T_{ij}
α_{D90}	The distance at which we would expect to see a 90% loss of information from the spatially distributed census block group centroid locations

S5. *Inform* stage results for residential contributions

5.1 Optimization with optim()

To do the optimization we used the R function optim(). See https://www.rdocumentation.org/packages/stats/versions/3.6.2/topics/optim for details. We used the L-BFGS-B method which allows for constrained optimization. To ensure that we were obtaining valid optimization results, we made comparisons to a brute force approach by plotting the objective function values across the hyperparameter space on a coarse grid. We were then able to compare the maxima observed in these plots to the ones we obtained with the optim function.

5.1.1 Visualizing the maxima

In Figures S3 and S4 we can see how the algorithm used in optim was able to obtain a maxima by displaying the objective function on a search grid for α_{90} at a 30 minute (0.5 hour) resolution for the

SEDC SPM. We see that when using concentration, no maximum was really found. The objective function is maximized as α_{90} approaches 0. In figure 4, when using load as the response, we see a clear maximum at 10.5 hours, which is consistent with the results from the optim function used in R which found a maximum at 10.3 hours (refer to main manuscript).

Figure S4 and S5 show the maximization of the objective function across the 2 and 3-dimensional hyperparameter spaces corresponding the SEDC-T and SEDC-TM SPMs, respectively.

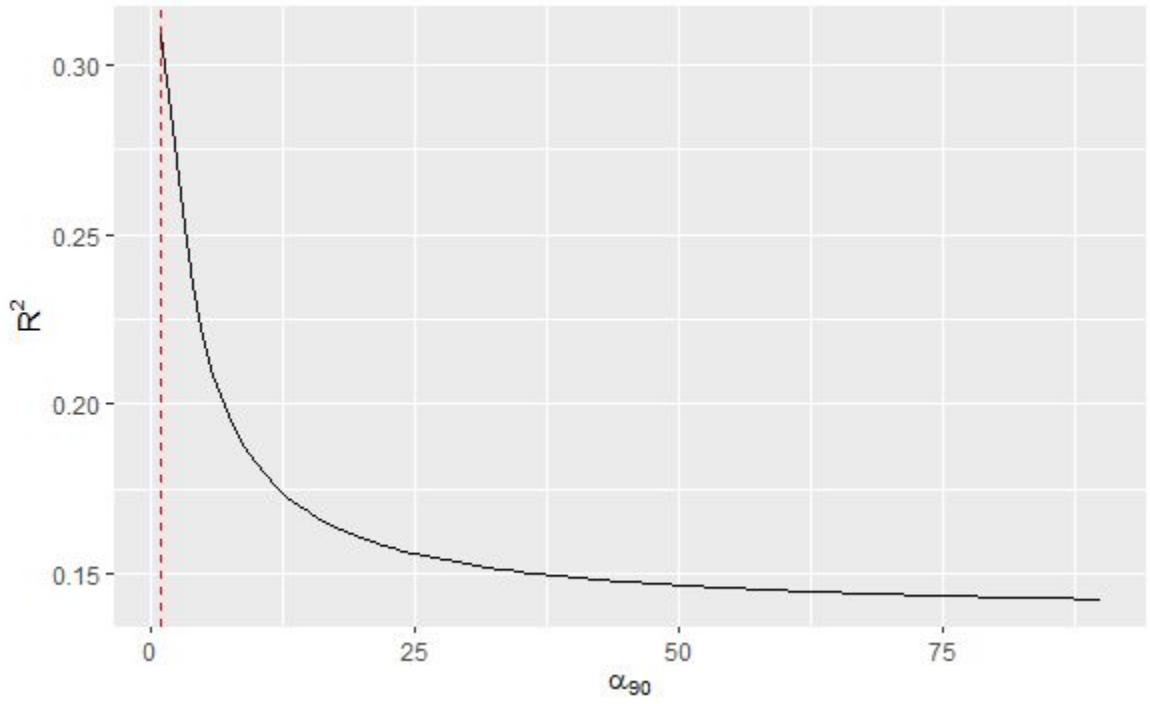

Figure S3 The optimum signal decay hyperparameter α_{90} from residential locations for log10 crAssphage concentration with the SEDC SPM

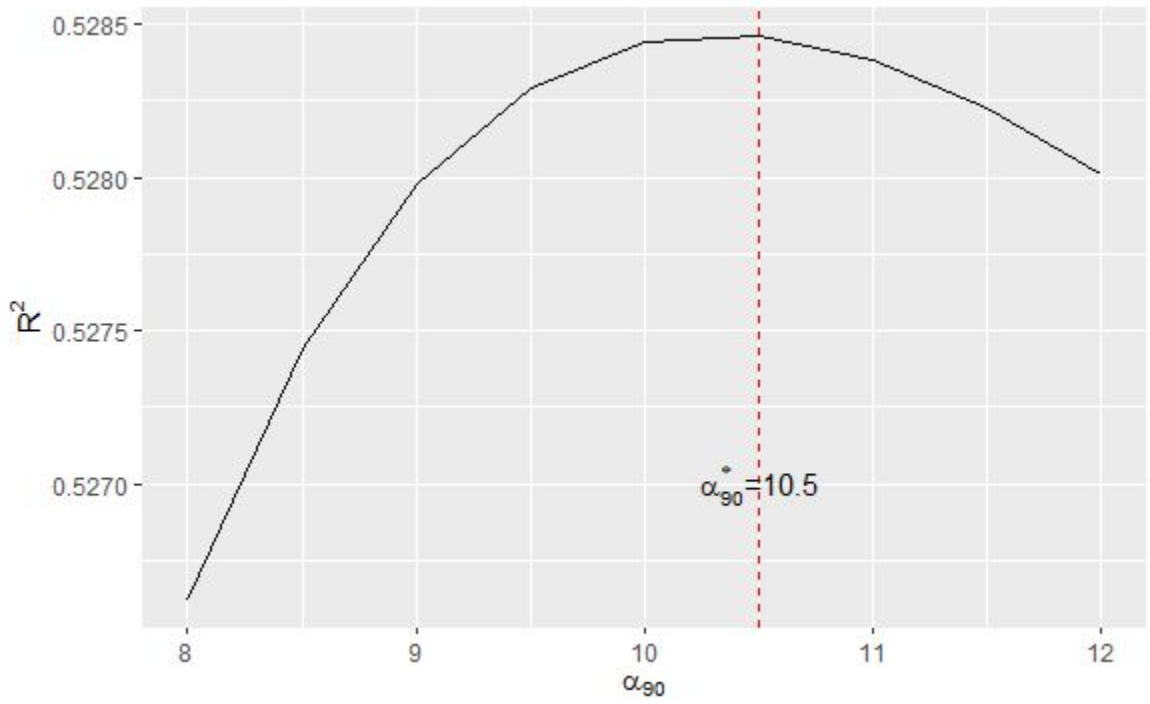

Figure S4 The optimum signal decay hyperparameter α_{90} from residential locations for log10 crAssphage load with the SEDC SPM

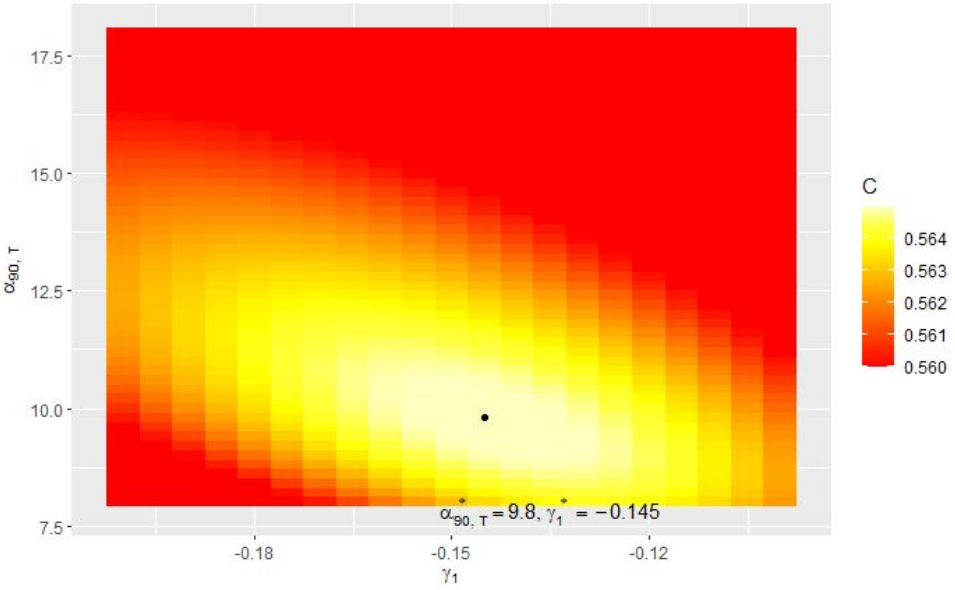

Figure S5 The optimum signal decay hyperparameter $\alpha_{90}(0)$ and signal decay modifier from temperature γ_1 from residential locations for log10 crAssphage load with the SEDC-T SPM. The colorbar labeled C represents R^2. The values descend much lower, but we have cut them off at 0.560 to be able to show the maximum more clearly.

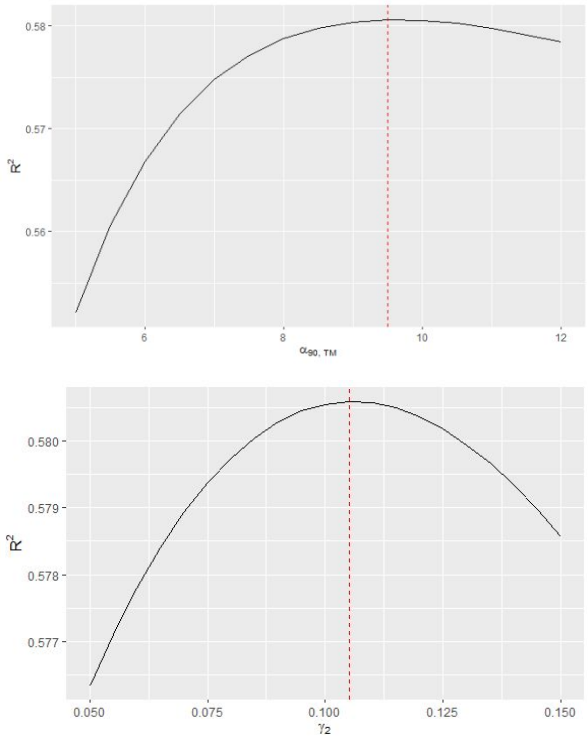

Figure S6 The optimum signal decay hyperparameter $\alpha_{90}(0)$ and signal decay modifiers from temperature γ_1 and mobility γ_2 from residential locations for log10 crAssphage load with the SEDC-TM SPM.

5.2 Additional information loss rate maps

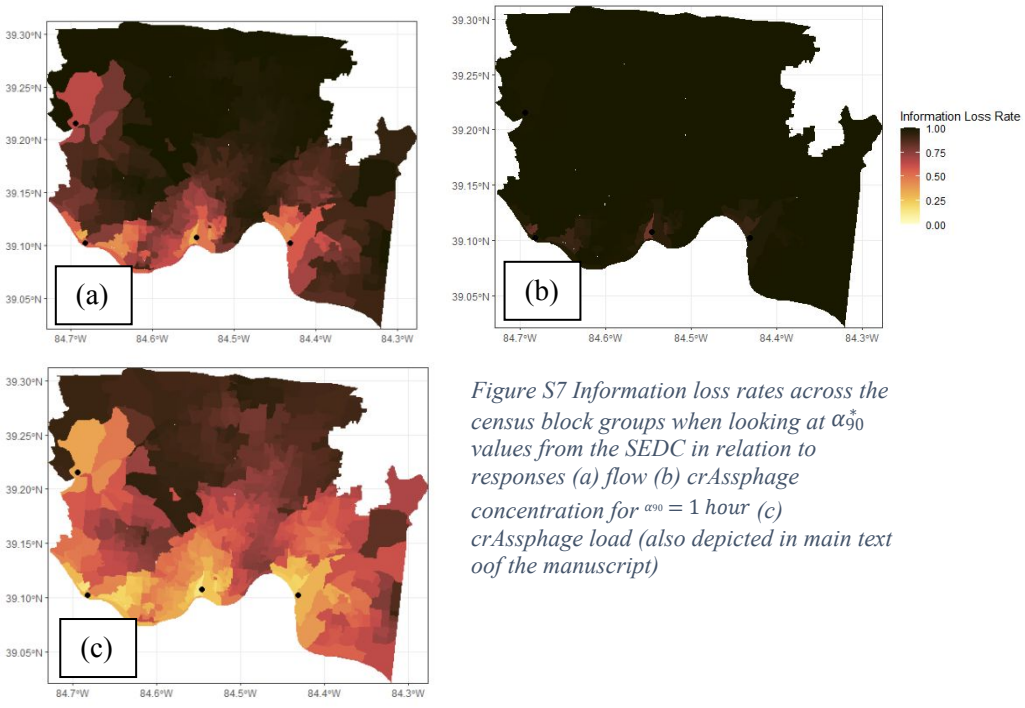

Figure S7 Information loss rates across the census block groups when looking at α_{90}^{*} values from the SEDC in relation to responses (a) flow (b) crAssphage concentration for $\alpha_{90} = 1\ hour$ (c) crAssphage load (also depicted in main text oof the manuscript)

S6. Details on preference for using crAssphage load over concentration

We find that accounting for dilution from flow is key to capturing the association between sewershed populations and crAssphage measurements. We compared the results of optimizing the signal decay hyperparameter of the SEDC SPM using crAssphage concentration (log_{10}-copies-per-liter) versus crAssphage load (log_{10}-copies-per-day). When using crAssphage concentration as the response, the spatial extent of population related to the measurement of crAssphage concentration at the WWTP was null (i.e., the optimal signal decay hyperparameter $\alpha_{90}^{*}$was always equal to the pre-set lower bound or very close to 0) which results in a map showing that no information is captured from the sewershed populations (see Figure S7). This indicates that when using a concentration response, information losses reduce the capacity to detect signals from residential locations almost immediately (α_{90}^{*} <1 hour) and result in poor association between signal contributions and measured crAssphage concentration at the WWTP ($\beta^{(residential)}$ <0) (Table 1).

When using log_{10} crAssphage load as the response (i.e., flow-adjusted concentration; Figure S2), a positive relationship emerges, and the spatial extent of the populations represented by the sample can be assessed; we find that a 90% loss of information occurs at approximately 10 hours of travel time for the sewersheds in this study (α_{90}^{*}=10.3 hours; Table 1). Therefore, using a flow-adjusted crAssphage response (i.e., load) reveals relationships between wastewater data and spatially distributed populations that are not evident based on concentration data alone. With flow-adjusted responses, our modeling that accounts for information losses reveals positive relationships between flow-adjusted responses and shedding populations across four sewersheds. Our results add to existing literature that load responses (i.e., the flow-adjusted responses) are preferred to concentrations for modeling.

S7. Sensitivity analysis for septic systems and Euclidean distance as a travel time proxy

Table S5 provides a demonstration of how the *Inform* stage approach changes little when there are errors on the population estimates by census block group but changes greatly when there are errors on the travel time (e.g., using Euclidean distance as a proxy). This is consistent with the results of the sensitivity analysis where we artificially added error many times to these different model components (see main manuscript).

Table S5 Provides the key SPM values (α_{90},β_1, and Adj. R^2) from running the Inform stage when not excluding populations on septic systems and using Euclidean distance instead of travel time.

	Septic system populations not excluded (k=1)	Euclidean distance instead of travel time (k=1)

α_{90}	**10.0 hours**	**100. m**
$\beta_{residentialSEDC}$	**0.372**	**0.330**
$Adj.\ R^2$	**0.533**	**0.419**

S8. Details on the results of the bootstrapped LASSO

8.1. Other sewershed factors which explain crAssphage load

In this work, we primarily focused on the *inform* stage of the microbial FIT framework to characterize the relationship between residential source locations and crAssphage load measured in WWTP influent samples. We explored SPMs that account for the modification of the signal decay hyperparameter **α** due to temperature (SEDC-T) or combined temperature and population mobility (SEDC-TM). Presented in Table 1, we find that the best model as defined by R^2 (SEDC-TM) explains 58.1% of the variability in crAssphage load, but statistically, all SEDC are comparable (i.e., within one standard-error of another from the 10-fold cross-validated MSE). Due to this, we selected the least complex of the three crAssphage load-based SPMs to be used in the in the *test* stage of FIT.

After running the LASSO at the *test* stage, we find the variables that were most greatly associated with crAssphage load in magnitude were those that varied spatially (i.e., contributions from populations at residential locations, average industrial flows, and whether the sewer was separate or combined) (Figure 2). Notably, after accounting for other factors, a 1 standard-deviation increase in the contributions from populations at residential locations was associated with a 1.17 (95%CI:0.491,1.86) increase in the log10 crAssphage load (log10-copies-crAssphage-per-day). A 1 standard-deviation increase in industrial flow was associated with a 0.670 decrease (95%CI: -1.25, -0.0928) in the log10 crAssphage load and combined sewer networks were associated with a 0.486 decrease (95%CI: -0.929, -0.0436) in the log10 crAssphage load. These decreases indicate that dilution is not fully accounted for by flow adjustment of concentration responses (i.e., to obtain load). Adding to this evidence is that precipitation was also associated with 0.0681 decrease (95%CI: -0.123, -0.0136) in log10 crAssphage load among combined sewer networks. Precipitation alone was evaluated but was not selected in the LASSO. This suggests that adjusting by flow is not sufficient to account for further dilution associated with precipitation events for combined sewer systems. Combined sewer overflows (CSOs) or exfiltration may explain this insufficiency, since they result in losses of flow from the system.[3] Specific knowledge of stormwater and industrial flow inputs, CSO events, and exfiltration processes may be useful for characterizing the dilution of wastewater responses in future modeling.

Temporal variables that were significant in explaining crAssphage load included wastewater temperature and mobility factors—the percent of time spent at different locations (retail and recreation, grocery or pharmacy, on transit, and home). Time spent at retail and recreation, on transit, and at home were all positively associated with crAssphage load, while time spent at grocery or pharmacy was negatively associated. Retail, recreation, and time spent on transit may reflect activity closer to downtown

areas, which are often near to WWTPs. We speculate that the reduction in time spent at grocery or pharmacy may also be inversely correlated with online shopping and may reflect quarantine and stay-at-home measures during the pandemic. Without a more spatially refined population mobility dataset, it is difficult to interpret the direction of these relationships. While there are more sophisticated methods to handle temperature and population mobility, as demonstrated by our SEDC-TM SPM and by Hart and Halden's deterministic modeling approach for temperature, [4,5] we find here that including these variables as predictors in a linear model may be sufficient.

We note that, overall, the magnitudes of the uniquely spatial effects are greater than those of the temporal effects. However, the spatial variables may also be explaining bulk differences between sewersheds related to population and sewer network characteristics not captured in this analysis. Likewise, the temporal variables may be explaining differences that are driven by underlying seasonal processes or long-term trends over time.

After the *test* stage of FIT, using the LASSO we explain 66.7% of the variability (see S6) in crAssphage load, with source contributions from populations at residential locations (i.e., SEDC) having the strongest impact. While there may be error associated with these population estimates (i.e., we used 2019 census data because 2020 data were not available yet), our sensitivity analysis (Section 3.5) demonstrates that small errors should not impact estimates greatly. Our results suggest that mobility is indeed a significant factor in describing crAssphage loads at WWTPs, warranting further study with spatially and temporally refined population mobility data.

Table S6 Test *stage bootstrapped LASSO regression results with confidence intervals around the regression coefficient estimates.*

Standardized variable	**Regression coefficient β**
Intercept	21.3 (20.1, 22.5)
Res. Cont.	1.17 (0.491, 1.86)
pH	0.0330 (-0.00504, 0.0711)
Combined Sewer	-0.486 (-0.929, -0.0436)
Retail/Recreation	0.0978 (0.0287, 0.167)
Grocery/Pharmacy	-0.133 (-0.210, -0.0571)
Transit	0.0487 (-0.00479, 0.102)
At Home	0.107 (0.0609, 0.154)
Industrial flow	-0.670 (-1.25, -0.0928)
Wastewater Temp.	-0.0611 (-0.101, -0.0209)
Precipitation (48h) and combined sewers	-0.0681 (-0.123, -0.0136)
R^2	*0.667*
Adjusted R^2	*0.655*

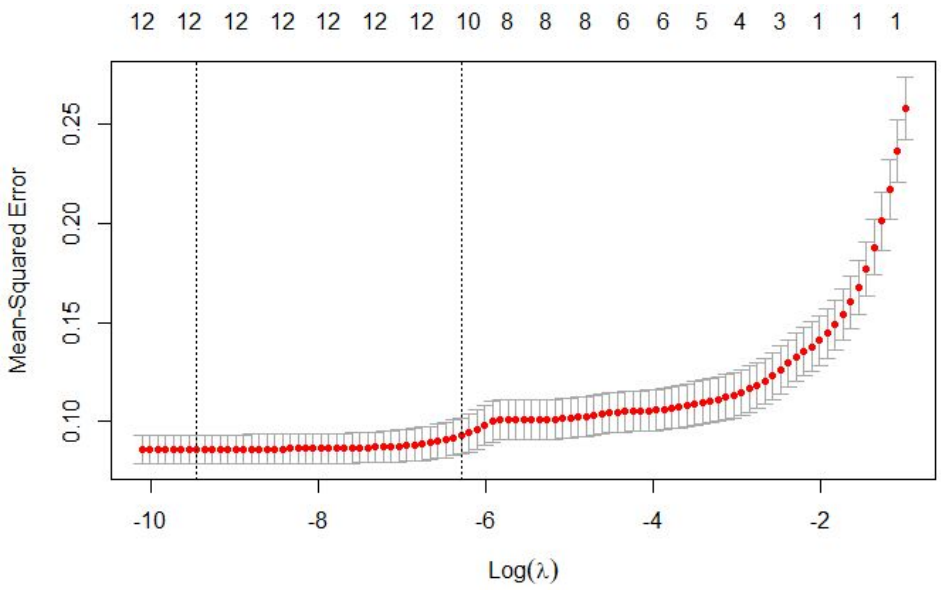

Figure S8 Selection of the MSE and the 1-standard-error $\log(\lambda)$ *for the* Test *stage of FIT*

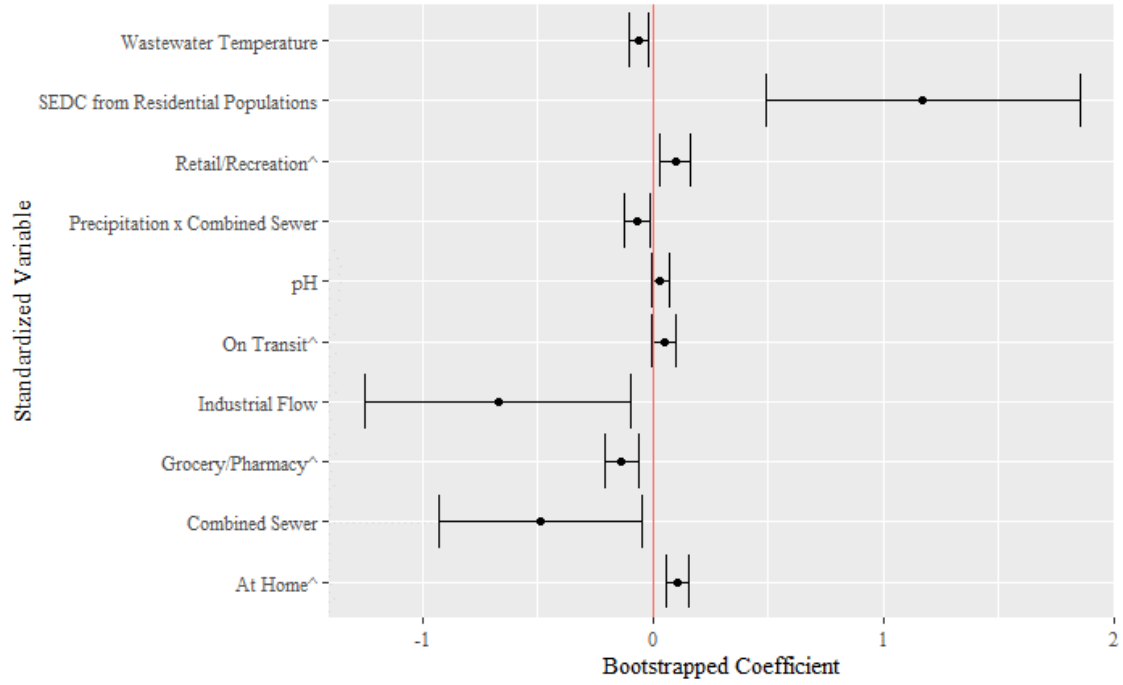

Figure S9 Bootstrapped standardized regression coefficients and 95% confidence intervals for the variables selected with the LASSO regression*. To compare the impact of the sum of exponentially decaying contributions (SEDC) from residentially-located populations to other factors. ^ indicates mobility variable representing % time spent at the location compared to baseline.*

S9. Bootstrapped LASSO with mobility as a modifier of residential signal contributions

Here, rather than considering the mobility variables as sewershed characteristics, we model them as modifiers of the signal contributions from residential locations.

In equation 2.1 of the main manuscript, here as S1 there are β_l corresponding to the modifying effects of mobility factors on contributions.

$$z_i = \beta_0 + \beta_1 s_i(\boldsymbol{\alpha}) exp\left\{\sum_{u=1}^{U} \theta_u m_i^{(u)}\right\} + \left\{\sum_{w=1}^{W} \beta_w x_i^{(w)}\right\} \quad \text{(Eq. S1)}$$

When there is only one source term and four mobility variables that modify the source term and we change the original FIT model with a linear approximation $exp(x) \approx (1 + x)$ for small x for each of the u^{th} modifier terms $m_i^{(u)}$. This leads us to the following:

$$z_i \approx \beta_0 + \beta_1 s_i(\boldsymbol{\alpha})\left\{1 + \sum_{u=1}^{U} \theta_u m_i^{(u)}\right\} + \left\{\sum_{w=1}^{W} \beta_w x_i^{(w)}\right\}$$

$$= \beta_0 + \left\{\beta_1 s_i(\boldsymbol{\alpha}) + \sum_{u=1}^{U} \beta_u s_i(\boldsymbol{\alpha}) m_i^{(u)}\right\} + \left\{\sum_{w=1}^{W} \beta_w x_i^{(w)}\right\} \quad \text{(Eq. S2)}$$

Where $\theta_u = \beta_u/\beta_1$, β_0 is the intercept, β_1 is the linear regression coefficient of our single population source term $s_i(\boldsymbol{\alpha})$, $m_i^{(u)}$ is the u^{th} modifier that amplifies ($\theta_u > 0$) or attenuates ($\theta_u < 0$) the population source term, and β_w is the linear regression coefficient for the w^{th} sewer characteristics $x_i^{(w)}$. If β_u is the resulting negative coefficient, it represents the diminishing effect of the u^{th} modifier on the contributing effect of the source term $s_i(\boldsymbol{\alpha})$.

This means that to find each θ_l, we would need to divide the regression coefficients, $\beta_2 \ldots \beta_5$, by β_1.

$$\theta_1 = \beta_2/\beta_1 \quad \text{(Eq. S3)}$$

$$\theta_2 = \beta_3/\beta_1 \quad \text{(Eq. S4)}$$

$$\theta_3 = \beta_4/\beta_1 \quad \text{(Eq. S5)}$$

$$\Theta_4 = \beta_5/\beta_1 \quad \text{(Eq. S6)}$$

After running the bootstrapped LASSO-MM, we find the following results:

Table S7 Test *stage bootstrapped LASSO with mobility as modifiers regression results with confidence intervals around the regression coefficient estimates.*

Standardized variable	**Regression coefficient β**	**Regression coefficient θ**	**Modifying effect for 1 std.-dev. increase** $\exp(\theta \times 1) \approx (1 + \theta \times 1)$

Intercept	21.3 (20.1, 22.5)	NA	NA
Res. Cont.	0.273 (0.214, 0.333)	NA	NA
Wastewater Temp.	-0.0394 (-0.0770, -0.00185)	NA	NA
Precipitation (48h) and combined sewers	-0.0248 (-0.0783, 0.0286)	NA	NA
At home	NA	0.111 (0.0392,0.157)	1.12
Grocery/Pharmacy	NA	-0.0370 (-0.137, 0.0274)	0.964
On transit	NA	-0.0202 (-0.100, 0.0312)	0.980
R^2	*0.589*		
Adjusted R^2	*0.581*		

The results from Table S7 indicate that there is some modification of contributions from residential locations by the percent of time that people spend at home. First, for a one-standard deviation increase in the modeled signal contributions from residential locations, we see a 0.273 increase in contributions from residential populations, we see a 0.273 increase in the log10 crAssphage load. For a 1 standard-deviation increase in the percent of time spent at home, we see that the effect of populations at residential locations on the wastewater signal increases by 12%. However, for a 1 standard-deviation increase in the percent of tiem spent at the grocery store/ pharmacy or on transit, we see small reductions in the effects of populations at residential locations on the wastewater signal.

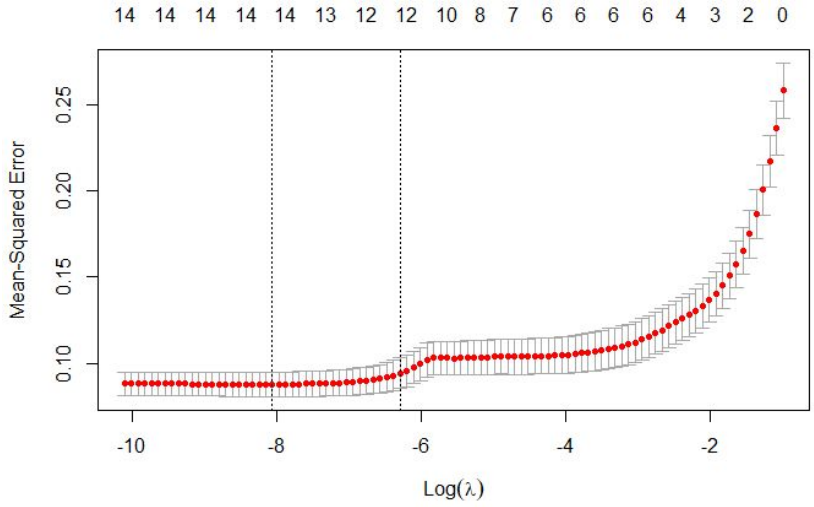

Figure S10 Selection of the MSE and the 1-standard-error $\log(\lambda)$ *for the LASSO-MM*

S10. Microbial Find, Inform, and Test code and functions in R for this application

10.1 Driver Code for running the analysis

This driver code is separated into different sections

1. Loading packages
2. Loading data from files
3. Defining functions
    a. Threshold functions

    b. Data processing
    c. LASSO Bootstrapping
    d. Spatial Predictor Models
        i. Sum of Exponentially Decaying Contributions
        ii. Upward capacities
    e. Objective functions
    f. SPM returning value functions
4. List of responses (e.g., flow, load, concentration, log10 flow, log10 load, log10 concentration)
5. Defining the other information needed for the SPMs
6. Defining the SPM source information (location and population)

```
#### Load package libraries####

library(pracma)
library(dplyr)
library(ggplot2)
library(glmnet)
library(latex2exp)
library(knitr)
library(lubridate)
library(scales)
library(gdata)
library(sf)
library(sp)
library(lmvar)
library(splitTools)

##### Load Data #####
crAssFull<-read.csv("~/Rworking/Stormwater/crAssphage/crAssphageFullv3_nogeom.csv")
source("~/Rworking/Stormwater/crAssphage/HamCo_TravelTimes.R")
filename2<-"~/Rworking/Stormwater/GIS_data/OhioWWsamplingDashCoords.shp"
WWTPs<-read_sf(dsn=filename2)
WWTPs2<-WWTPs[c(18,20,21,22,23,24,25),]
WWTPs2<-WWTPs2[WWTPs2$Site_Count=="Hamilton",]
WWTPdf<-data.frame(WWTPs2$Facility_N,WWTPs2$X,WWTPs2$Y)
colnames(WWTPdf)=c("ID","X","Y")
idxWrongEastern=(WWTPdf$ID=="Eastern Ohio RWA")
WWTPdf<-WWTPdf[!idxWrongEastern,] ## HARD CODED.
unms<-unique(crAssFull$Sewrshd)
nms<-GetCongruentCharStr(WWTPdf$ID,unms)
idxno=nms==""
WWTPdf$ID=nms
WWTPdf<-WWTPdf[!idxno,]
WWTPdf.sp<-WWTPdf
coordinates(WWTPdf.sp)<-~X+Y
proj4string(WWTPdf.sp)<-CRS("+init=epsg:4326")
WWTPdf.sp<-spTransform(WWTPdf.sp,CRS("+init=epsg:3735"))
```

```
filename1<-"~/Rworking/Stormwater/GIS_data/crAssphage/CBG_Sheds_HamMont_v3.shp"
SewershedData<-read_sf(dsn=filename1)



filename2<-"~/Rworking/SARS-CoV-2/GISdata/SewershedData_v2.shp"
SewershedDataG<-read_sf(dsn=filename2)
SewershedDataG<-SewershedDataG[c(3,5,6,9,10,15,18),]
SewershedDataG.spcs<-st_transform(SewershedDataG,3735)



filename3<-"~/Rworking/SARS-CoV-2/GISdata/crAssphageSheds.shp"
ShedOutlines<-read_sf(dsn=filename3)


crAssFull<-crAssFull[crAssFull$County=="Hamilton County",]
#### Define Functions ####
## THE SIGMOID FUNCTION -- to be used to constrain values equal/between 0 and 1
## proportion of another value may be val*sigmoid(x) and can be used as
## an alternative to the exp() function for modifiers or changes in T90.
## The exp() function has
## the advantage of
##
sigmoid<-function(x){
  y=1/(1+exp(-x))
  return(y)
}

## THE MAX FUNCTION -- to be used to constrain values equal to or above zero.
## essentially scales things if above 0. Potentially useful instead of exp()
## for modifiers or changes in T90
maxof<-function(x){
  tmp=x>0
  y=x*tmp
  return(y)
}

### this is for relating the datasets
GetCongruentCharStr<-function(charstrchange,charstrdesired){
  charstrchange=toupper(charstrchange)
  nmVex<-vector(mode="character",length=length(charstrchange))
  for (i in 1:length(charstrdesired)) {
    idx=grep(charstrdesired[i],charstrchange)
    nmVex[idx]=charstrdesired[i]
  }
```

```
  return(nmVex)
}

### if you want to try to standardize a variable by the location
zscorebyloc<-function(var1,locID){
  uloc=unique(locID)
  val=vector("numeric",length(locID))
  for (i in 1:length(uloc)){
    idx=locID==uloc
    mval=mean(var1[idx])
    sdval=std(var1[idx])
    val[idx]=(var1[idx]-mval)/sdval
  }
  return(val)
}

## Bootstrap lasso!
# To get 95% confidence intervals around the estimate!
BootLasso1<-function(X,zR,best_lambda,bootstraps){
  betas=matrix(NA,ncol(X)+1,bootstraps)
  for (i in 1:bootstraps) {
    sampleIdx=sample(1:length(zR),length(zR),replace=TRUE)
    Xdata=X[sampleIdx,]
    ydata=zR[sampleIdx]
    best_model <- glmnet(Xdata, ydata, alpha = 1, lambda = best_lambda)
    betas[,i]=as.vector(coef(best_model))
  }
  BSmeans=(rowSums(betas))/bootstraps
  BSstderr=sqrt(rowSums((betas-BSmeans)^2)/(bootstraps-1))
  CIlower=BSmeans-1.96*(BSstderr)
  CIupper=BSmeans+1.96*(BSstderr)
  df=data.frame(BSmeans,BSstderr,CIlower,CIupper)
  row.names(df)=c("intercept",colnames(X))
  return(df)
}

BootLasso2<-function(X,zR,best_lambda,p.fac,bootstraps){
  betas=matrix(NA,ncol(X)+1,bootstraps)
  for (i in 1:bootstraps) {
    sampleIdx=sample(1:length(zR),length(zR),replace=TRUE)
    Xdata=X[sampleIdx,]
    ydata=zR[sampleIdx]
    best_model <- glmnet(Xdata, ydata, alpha = 1, lambda = best_lambda, penalty.factor=p.fac)
    betas[,i]=as.vector(coef(best_model))
  }
  BSmeans=(rowSums(betas))/bootstraps
  BSstderr=sqrt(rowSums((betas-BSmeans)^2)/(bootstraps-1))
```

```
  CIlower=BSmeans-1.96*(BSstderr)
  CIupper=BSmeans+1.96*(BSstderr)
  df=data.frame(BSmeans,BSstderr,CIlower,CIupper)
  row.names(df)=c("intercept",colnames(X))
  return(df)
}
## the traditional SEDC
# INPUT
# T90 is the travel time at which you would expect a 90% reduction
# rID is the unique sewershed ID (location ID) for the response
# sIDj is the sewershed ID for the census block groups (flow-connected)
# zSj is the initial scale of the census block group
# Tij is the "travel time" between each census block group centroid and wwtp
sedc1<-function(T90,sIDj,zSj,Tij,rIDi,dti) {
  trID=unique(dti)
  tmin=min(trID)
  rID=NULL
  tID=NULL
  result=NULL
  df2<-data.frame(rID,tID,result)
  for (j in 1:length(trID)) {
    idxt=(dti==trID[j]) # index to remove temporal element
    rID=rIDi[idxt]
    result=vector("numeric",length(rID))
    tID=repmat(as.numeric(as.Date(trID[j])-as.Date(tmin)),length(rID),1)
    for (i in 1:length(rID)) {
      idx=(sIDj==rID[i])
      zStmp=zSj[idx]
      Tijtmp=Tij[idx]
      result[i]=(sum(zStmp*exp((-2.3*Tijtmp)/T90)))
    }
    df1<-data.frame(rID,tID,result)
    df2<-bind_rows(df2,df1)
  }
  df2$tID=as.Date(tmin)+df2$tID
  return(df2)
}

## the variable-adjusted hyperparameter SEDC
# T90 is the travel time at which you would expect a 90% reduction
# alpha is the change in T90 seen with a 1 unit increase in x
# zSj is the initial scale of the census block group
# Tij is the "travel time" between each census block group centroid and wwtp
# rIDi is the sewershed ID (location ID) for the response data
# dti is the date of the sample for the response data
# xi is the variable that afffects T90 (same size as response data)
# modfun determines which function to apply to alpha*T90modifierVar
```

```
# The functions for modfun are 1- max(0,x) 2) exp 3) sigmoid
sedc2<-function(T90,alpha,sIDj,zSj,Tij,rIDi,dti,xi,modfun) {
 trID=unique(dti)
 tmin=min(trID)
 rID=NULL
 tID=NULL
 result=NULL
 df2<-data.frame(rID,tID,result)
 for (j in 1:length(trID)) {
  idxt=(dti==trID[j]) # index to remove temporal element
  xq=xi[idxt]
  rID=rIDi[idxt]
  result=vector("numeric",length(rID))
  tID=repmat(as.numeric(as.Date(trID[j])-as.Date(tmin)),length(rID),1)
  for (i in 1:length(rID)) {
   idx=(sIDj==rID[i])
   zStmp=zSj[idx]
   Tijtmp=Tij[idx]
   if ((modfun==1)==TRUE) {
    result[i]=(sum(zStmp*exp(-2.3*Tijtmp/(T90*maxof(alpha*xq[i])))))
   }
   if ((modfun==2)==TRUE) {
    result[i]=(sum(zStmp*exp(-2.3*Tijtmp/(T90*exp(alpha*xq[i])))))
   }
   if ((modfun==3)==TRUE) {
    result[i]=(sum(zStmp*exp(-2.3*Tijtmp/(T90*sigmoid(alpha*xq[i])))))
   }
  }
  df1<-data.frame(rID,tID,result)
  df2<-bind_rows(df2,df1)
 }
 df2$tID=as.Date(tmin)+df2$tID
 return(df2)
}
## the traditional SEDC with flow
# INPUT
# T90 is the travel time at which you would expect a 90% reduction
# rID is the unique sewershed ID (location ID) for the response
# sIDj is the sewershed ID for the census block groups (flow-connected)
# zSj is the initial scale of the census block group
# Tij is the "travel time" between each census block group centroid and wwtp
# Qi is the flow at the wastewater treatment plant for that sample
sedc1F<-function(T90,sIDj,zSj,Tij,rIDi,dti,Qi) {
 trID=unique(dti)
 tmin=min(trID)
 rID=NULL
 tID=NULL
```

```
  result=NULL
  df2<-data.frame(rID,tID,result)
  for (j in 1:length(trID)) {
    idxt=(dti==trID[j]) # index to remove temporal element
    rID=rIDi[idxt]
    Qiq=Qi[idxt]
    result=vector("numeric",length(rID))
    tID=repmat(as.numeric(as.Date(trID[j])-as.Date(tmin)),length(rID),1)
    for (i in 1:length(rID)) {
      idx=(sIDj==rID[i])
      zStmp=zSj[idx]
      Tijtmp=Tij[idx]
      result[i]=(sum(zStmp*exp((-2.3*Tijtmp)/T90)))/Qiq[i]
    }
    df1<-data.frame(rID,tID,result)
    df2<-bind_rows(df2,df1)
  }
  df2$tID=as.Date(tmin)+df2$tID
  return(df2)
}

## the variable-adjusted hyperparameter SEDC with flow
# T90 is the travel time at which you would expect a 90% reduction
# alpha is the change in T90 seen with a 1 unit increase in x
# zSj is the initial scale of the census block group
# Tij is the "travel time" between each census block group centroid and wwtp
# rIDi is the sewershed ID (location ID) for the response data
# dti is the date of the sample for the response data
# xi is the variable that afffects T90 (same size as response data) T90modifierVar
# Qi is the flow at the wastewater treatment plant for that sample
# modfun determines which function to apply to alpha*T90modifierVar
# The functions for modfun are 1- max(0,x) 2) exp 3) sigmoid
sedc2F<-function(T90,alpha,sIDj,zSj,Tij,rIDi,dti,xi,Qi,modfun) {
  trID=unique(dti)
  tmin=min(trID)
  rID=NULL
  tID=NULL
  result=NULL
  df2<-data.frame(rID,tID,result)
  for (j in 1:length(trID)) {
    idxt=(dti==trID[j]) # index to remove temporal element
    xq=xi[idxt]
    Qiq=Qi[idxt]
    rID=rIDi[idxt]
    result=vector("numeric",length(rID))
    tID=repmat(as.numeric(as.Date(trID[j])-as.Date(tmin)),length(rID),1)
    for (i in 1:length(rID)) {
```

```
    idx=(sIDj==rID[i])
    zStmp=zSj[idx]
    Tijtmp=Tij[idx]
    if ((modfun==1)==TRUE) {
     result[i]=(sum(zStmp*exp(-2.3*Tijtmp/(T90*maxof(alpha*xq[i])))))/Qiq[i]
    }
    if ((modfun==2)==TRUE) {
     result[i]=(sum(zStmp*exp(-2.3*Tijtmp/(T90*exp(alpha*xq[i])))))/Qiq[i]
    }
    if ((modfun==3)==TRUE) {
     result[i]=(sum(zStmp*exp(-2.3*Tijtmp/(T90*sigmoid(alpha*xq[i])))))/Qiq[i]
    }
   }
   df1<-data.frame(rID,tID,result)
   df2<-bind_rows(df2,df1)
  }
  df2$tID=as.Date(tmin)+df2$tID
  return(df2)
}

sedc3<-function(alpha,delta,mew,sIDj,zSj,Tij,rIDi,dti,x1,x2,modfun) {
  trID=unique(dti) ### change alpha delta mew from T90 and alpha...
  tmin=min(trID)
  rID=NULL
  tID=NULL
  result=NULL
  df2<-data.frame(rID,tID,result)
  for (j in 1:length(trID)) {
   idxt=(dti==trID[j]) # index to remove temporal element
   xq1=x1[idxt]
   xq2=x2[idxt]
   rID=rIDi[idxt]
   result=vector("numeric",length(rID))
   tID=repmat(as.numeric(as.Date(trID[j])-as.Date(tmin)),length(rID),1)
   for (i in 1:length(rID)) {
    idx=(sIDj==rID[i])
    zStmp=zSj[idx]
    Tijtmp=Tij[idx]
    if ((modfun==1)==TRUE) {
     result[i]=(sum(zStmp*exp(-2.3*Tijtmp/(alpha*maxof(delta*xq1[i]+mew*xq2[i])))))
    }
    if ((modfun==2)==TRUE) {
     result[i]=(sum(zStmp*exp(-2.3*Tijtmp/(alpha*exp(delta*xq1[i]+mew*xq2[i])))))
    }
    if ((modfun==3)==TRUE) {
     result[i]=(sum(zStmp*exp(-2.3*Tijtmp/(alpha*sigmoid(delta*xq1[i]+mew*xq2[i])))))
    }
```

```
    }
    df1<-data.frame(rID,tID,result)
    df2<-bind_rows(df2,df1)
  }
  df2$tID=as.Date(tmin)+df2$tID
  return(df2)
}

sedc3F<-function(alpha,delta,mew,sIDj,zSj,Tij,rIDi,dti,x1,x2,Qi,modfun) {
  trID=unique(dti) ### change alpha delta mew from T90 and alpha...
  tmin=min(trID)
  rID=NULL
  tID=NULL
  result=NULL
  df2<-data.frame(rID,tID,result)
  for (j in 1:length(trID)) {
    idxt=(dti==trID[j]) # index to remove temporal element
    xq1=x1[idxt]
    xq2=x2[idxt]
    Qiq=Qi[idxt]
    rID=rIDi[idxt]
    result=vector("numeric",length(rID))
    tID=repmat(as.numeric(as.Date(trID[j])-as.Date(tmin)),length(rID),1)
    for (i in 1:length(rID)) {
      idx=(sIDj==rID[i])
      zStmp=zSj[idx]
      Tijtmp=Tij[idx]
      if ((modfun==1)==TRUE) {
        result[i]=(sum(zStmp*exp(-2.3*Tijtmp/(alpha*maxof(delta*xq1[i]+mew*xq2[i])))))/Qiq[i]
      }
      if ((modfun==2)==TRUE) {
        result[i]=(sum(zStmp*exp(-2.3*Tijtmp/(alpha*exp(delta*xq1[i]+mew*xq2[i])))))/Qiq[i]
      }
      if ((modfun==3)==TRUE) {
        result[i]=(sum(zStmp*exp(-2.3*Tijtmp/(alpha*sigmoid(delta*xq1[i]+mew*xq2[i])))))/Qiq[i]
      }
    }
    df1<-data.frame(rID,tID,result)
    df2<-bind_rows(df2,df1)
  }
  df2$tID=as.Date(tmin)+df2$tID
  return(df2)
}
```

```
## experimental sedc

sedc3expmt<-function(alpha,delta,mew,sIDj,zSj,Tij,rIDi,dti,x1,x2,modfun) {
 trID=unique(dti) ### change alpha delta mew from T90 and alpha...
 tmin=min(trID)
 rID=NULL
 tID=NULL
 result=NULL
 df2<-data.frame(rID,tID,result)
 for (j in 1:length(trID)) {
  idxt=(dti==trID[j]) # index to remove temporal element
  xq1=x1[idxt]
  xq2=x2[idxt]
  rID=rIDi[idxt]
  result=vector("numeric",length(rID))
  tID=repmat(as.numeric(as.Date(trID[j])-as.Date(tmin)),length(rID),1)
  for (i in 1:length(rID)) {
   idx=(sIDj==rID[i])
   zStmp=zSj[idx]
   Tijtmp=Tij[idx]
   if ((modfun==1)==TRUE) {
    result[i]=maxof(mew*xq2[i])*(sum(zStmp*exp(-2.3*Tijtmp/(alpha*maxof(delta*xq1[i])))))
   }
   if ((modfun==2)==TRUE) {
    result[i]=exp(mew*xq2[i])*(sum(zStmp*exp(-2.3*Tijtmp/(alpha*exp(delta*xq1[i])))))
   }
   if ((modfun==3)==TRUE) {
    result[i]=sigmoid(mew*xq2[i])*(sum(zStmp*exp(-
2.3*Tijtmp/(alpha*sigmoid(delta*xq1[i])))))
   }
  }
  df1<-data.frame(rID,tID,result)
  df2<-bind_rows(df2,df1)
 }
 df2$tID=as.Date(tmin)+df2$tID
 return(df2)
}

## just the associated sum of scales
Capacities<-function(sIDj,zSj,rIDi,dti) {
 trID=unique(dti)
 tmin=min(trID)
 rID=NULL
 tID=NULL
 result=NULL
 df2<-data.frame(rID,tID,result)
 for (j in 1:length(trID)) {
```

```
    idxt=(dti==trID[j]) # index to remove temporal element
    rID=rIDi[idxt]
    result=vector("numeric",length(rID))
    tID=repmat(as.numeric(as.Date(trID[j])-as.Date(tmin)),length(rID),1)
    for (i in 1:length(rID)) {
      idx=(sIDj==rID[i])
      zStmp=zSj[idx]
      result[i]=(sum(zStmp))
    }
    df1<-data.frame(rID,tID,result)
    df2<-bind_rows(df2,df1)
  }
  df2$tID=as.Date(tmin)+df2$tID
  return(df2)
}
# associated sum of scales divided by flow
CapacitiesF<-function(sIDj,zSj,rIDi,dti,Qi) {
  trID=unique(dti)
  tmin=min(trID)
  rID=NULL
  tID=NULL
  result=NULL
  df2<-data.frame(rID,tID,result)
  for (j in 1:length(trID)) {
    idxt=(dti==trID[j]) # index to remove temporal element
    rID=rIDi[idxt]
    Qiq=Qi[idxt]
    result=vector("numeric",length(rID))
    tID=repmat(as.numeric(as.Date(trID[j])-as.Date(tmin)),length(rID),1)
    for (i in 1:length(rID)) {
      idx=(sIDj==rID[i])
      zStmp=zS[idx]
      result[i]=(sum(zStmp))/Qiq[i]
    }
    df1<-data.frame(rID,tID,result)
    df2<-bind_rows(df2,df1)
  }
  df2$tID=as.Date(tmin)+df2$tID
  return(df2)
}

#### Objective Functions #####

ObjFun1<-function(par,rIDi,dti,zR,zSj,Tij){
  T90=par[1]
  dfR=data.frame(rIDi,dti,zR)
  dfR$dti=as.Date(dfR$dti)
```

```
  dfS=sedc1(T90,sIDj,zSj,Tij,rIDi,dti)
  dfS$lg10res=log10(dfS$result)
  dfS$stdl10res=dfS$lg10res/std(dfS$lg10res) # I did standardize here!
  dfS$stdRes=dfS$result/std(dfS$result)
  dfRS=merge(dfR,dfS,by.x=c("rIDi","dti"),by.y=c("rID","tID"))
  mdl=lm(zR~stdRes,dfRS)
  tmp=summary(mdl)
  val=1-tmp$r.squared
  return(val)
}


ObjFun1F<-function(par,rIDi,dti,zR,zSj,Qi,Tij){
  T90=par[1]
  dfR=data.frame(rIDi,dti,zR)
  dfR$dti=as.Date(dfR$dti)
  dfS=sedc1F(T90,sIDj,zSj,Tij,rIDi,dti,Qi)
  dfS$lg10res=log10(dfS$result)
  dfS$stdl10res=dfS$lg10res/std(dfS$lg10res) # I did standardize here!
  dfS$stdRes=dfS$result/std(dfS$result)
  dfRS=merge(dfR,dfS,by.x=c("rIDi","dti"),by.y=c("rID","tID"))
  mdl=lm(zR~stdl10res,dfRS)
  tmp=summary(mdl)
  val=1-tmp$r.squared
  return(val)
}

ObjFun2<-function(par,rIDi,dti,zR,zSj,m1,modfun,Tij){
  T90=par[1]
  alpha=par[2]
  dfR=data.frame(rIDi,dti,zR)
  dfR$dti=as.Date(dfR$dti)
  dfS=sedc2(T90,alpha,sIDj,zSj,Tij,rIDi,dti,m1,modfun)
  dfS$lg10res=log10(dfS$result)
  dfS$stdl10res=dfS$lg10res/std(dfS$lg10res) # I did standardize here!
  dfS$stdres=dfS$result/std(dfS$result)
  dfRS=merge(dfR,dfS,by.x=c("rIDi","dti"),by.y=c("rID","tID"))
  mdl=lm(zR~stdres,dfRS)
  tmp=summary(mdl)
  val= 1-tmp$r.squared
  return(val)
}

ObjFun2F<-function(par,rIDi,dti,zR,zSj,Qi,m1,modfun,Tij){
  T90=par[1]
  alpha=par[2]
  dfR=data.frame(rIDi,dti,zR)
```

```
  dfR$dti=as.Date(dfR$dti)
  dfS=sedc2F(T90,alpha,sIDj,zSj,Tij,rIDi,dti,m1,Qi,modfun)
  dfS$lg10res=log10(dfS$result)
  dfS$stdl10res=dfS$lg10res/std(dfS$lg10res) # I did standardize here!
  dfRS=merge(dfR,dfS,by.x=c("rIDi","dti"),by.y=c("rID","tID"))
  mdl=lm(zR~stdl10res,dfRS)
  tmp=summary(mdl)
  val= 1-tmp$r.squared
  return(val)
}

ObjFun2FB<-function(par,rIDi,dti,zR,zSj,Qi,m1,modfun, sIDj,Tij){
  T90=par[1]
  alpha=par[2]
  dfR=data.frame(rIDi,dti,zR)
  dfR$dti=as.Date(dfR$dti)
  dfS=sedc2F(T90,alpha,sIDj,zSj,Tij,rIDi,dti,m1,Qi,modfun)
  dfS$lg10res=log10(dfS$result)
  dfS$stdl10res=dfS$lg10res/std(dfS$lg10res) # I did standardize here!
  dfRS=merge(dfR,dfS,by.x=c("rIDi","dti"),by.y=c("rID","tID"))
  idx=any(is.nan(dfS$stdl10res))
  if (idx==TRUE){
    val=1} else {
      mdl=lm(zR~stdl10res,dfRS)
      tmp=summary(mdl)
      val= 1-tmp$r.squared
      return(val)
    }
}

ObjFun3<-function(par,rIDi,dti,zR,zSj,Qi,m1,m2,modfun,Tij){
  alpha=par[1]
  delta=par[2]
  mew=par[3]
  dfR=data.frame(rIDi,dti,zR)
  dfR$dti=as.Date(dfR$dti)
  dfS=sedc3(alpha,delta, mew,sIDj,zSj,Tij,rIDi,dti,m1,m2,modfun)
  dfS$stdres=dfS$result/std(dfS$result)
  dfRS=merge(dfR,dfS,by.x=c("rIDi","dti"),by.y=c("rID","tID"))
  mdl=lm(zR~stdres,dfRS)
  tmp=summary(mdl)
  val= 1-tmp$r.squared
  return(val)
}

ObjFun3F<-function(par,rIDi,dti,zR,zSj,Qi,m1,m2,modfun,Tij){
  alpha=par[1]
```

```
  delta=par[2]
  mew=par[3]
  dfR=data.frame(rIDi,dti,zR)
  dfR$dti=as.Date(dfR$dti)
  dfS=sedc3F(alpha,delta, mew,sIDj,zSj,Tij,rIDi,dti,m1,m2,Qi,modfun)
  dfS$lg10res=log10(dfS$result)
  dfS$stdl10res=dfS$lg10res/std(dfS$lg10res) # I did standardize here!
  dfRS=merge(dfR,dfS,by.x=c("rIDi","dti"),by.y=c("rID","tID"))
  mdl=lm(zR~stdl10res,dfRS)
  tmp=summary(mdl)
  val= 1-tmp$r.squared
  return(val)
}

ObjFun3expmt<-function(par,rIDi,dti,zR,zSj,Qi,m1,m2,modfun,Tij){
  alpha=par[1]
  delta=par[2]
  mew=par[3]
  dfR=data.frame(rIDi,dti,zR)
  dfR$dti=as.Date(dfR$dti)
  dfS=sedc3expmt(alpha,delta, mew,sIDj,zSj,Tij,rIDi,dti,m1,m2,modfun)
  dfS$stdres=dfS$result/std(dfS$result)
  dfRS=merge(dfR,dfS,by.x=c("rIDi","dti"),by.y=c("rID","tID"))
  mdl=lm(zR~stdres,dfRS)
  tmp=summary(mdl)
  val= 1-tmp$r.squared
  return(val)
}

#### Return Value Functions #####
GetMod1Vals<-function(par,rIDi,dti,zR,zSj,Tij){
  T90=par[1]
  dfS=sedc1(T90,sIDj,zSj,Tij,rIDi,dti)
  dfS$lg10res=log10(dfS$result)
  dfS$stdl10res=dfS$lg10res/std(dfS$lg10res) # I did standardize here!
  dfS$stdRes=dfS$result/std(dfS$result)
  return(dfS)
}

GetMod1FVals<-function(par,rIDi,dti,zR,zSj,Qi,Tij){
  T90=par[1]
  dfS=sedc1F(T90,sIDj,zSj,Tij,rIDi,dti,Qi)
  dfS$lg10res=log10(dfS$result)
  dfS$stdl10res=dfS$lg10res/std(dfS$lg10res) # I did standardize here!
  dfS$stdRes=dfS$result/std(dfS$result)
  return(dfS)
}
```

```
GetMod2Vals<-function(par,rIDi,dti,zR,zSj,m1,modfun,Tij){
  T90=par[1]
  alpha=par[2]
  dfS=sedc2(T90,alpha,sIDj,zSj,Tij,rIDi,dti,m1,modfun)
  dfS$lg10res=log10(dfS$result)
  dfS$stdres=dfS$result/std(dfS$result) # I did standardize here!
  return(dfS)
}


GetMod2FVals<-function(par,rIDi,dti,zR,zSj,Qi,m1,modfun,Tij){
  T90=par[1]
  alpha=par[2]
  dfS=sedc2F(T90,alpha,sIDj,zSj,Tij,rIDi,dti,m1,Qi,modfun)
  dfS$lg10res=log10(dfS$result)
  dfS$stdl10res=dfS$lg10res/std(dfS$lg10res) # I did standardize here!
  return(dfS)
}


GetMod2FValsB<-function(par,rIDi,dti,zR,zSj,Qi,m1,modfun,sIDj,Tij){
  T90=par[1]
  alpha=par[2]
  dfS=sedc2F(T90,alpha,sIDj,zSj,Tij,rIDi,dti,m1,Qi,modfun)
  dfS$lg10res=log10(dfS$result)
  dfS$stdl10res=dfS$lg10res/std(dfS$lg10res) # I did standardize here!
  return(dfS)
}

GetMod3Vals<-function(par,rIDi,dti,zR,zSj,m1,m2,modfun,Tij){
  alpha=par[1]
  delta=par[2]
  mew=par[3]
  dfS=sedc3(alpha,delta, mew,sIDj,zSj,Tij,rIDi,dti,m1,m2,modfun)
  dfS$stdres=dfS$result/std(dfS$result)
  return(dfS)
}


GetMod3FVals<-function(par,rIDi,dti,zR,zSj,Qi,m1,m2,modfun,Tij){
  alpha=par[1]
  delta=par[2]
  mew=par[3]
  dfS=sedc3F(alpha,delta, mew,sIDj,zSj,Tij,rIDi,dti,m1,m2,Qi,modfun)
  dfS$lg10res=log10(dfS$result)
  dfS$stdl10res=dfS$lg10res/std(dfS$lg10res) # I did standardize here!
```

```
  return(dfS)
}



#### Select the default response ####

flowpd=crAssFull$flowval*3785411.78 # MGD to liters per day
#### Set the response info for manual response selection ####
zRlist=list("Flow"=flowpd, "LogFlow"=log10(flowpd),"Load"=flowpd*crAssFull$c_L,
"LogLoad"=log10(flowpd*crAssFull$c_L), "Conc"=crAssFull$c_L,
"LogConc"=log10(crAssFull$c_L))

#### Set the other information required for the SEDCs ####
Tij=ShedData$TTavg
sIDj=ShedData$Sewershed
rID=unique(ShedData$Sewershed)
dti=crAssFull$Date
rIDi=crAssFull$Sewrshd
Qi=crAssFull$flowval/max(crAssFull$flowval)


#### Linear Model Define spatial predictor variables "source" ####
hhConst=2.41 # from census information
Acbg=ShedData$CBG_Area
pop=ShedData$pop*ShedData$Pcnt_Sewer
sep=ShedData$SepCount*ShedData$Pcnt_Sewer

# MODEL 1) will only use zSpop for this model. Limited mobility- people are where they
# are said to live according to the 2019 census
zSpop=pop
zScompar=ShedData$ComParcels_Pswrd

zSpopMod=(pop-(sep*hhConst))
zSpopMod[zSpopMod<0]=min(zSpopMod[!zSpopMod<0])/10

zSlist1=list("pop"=zSpop, "res"=0, "com"=0, "ind"=0, "oth"=0, "cbg"=0, "compar"=zScompar,
"popMod"=zSpopMod)
```

10.2 Code for reproducing the results of this manuscript

The following code can be pasted into a Rmarkdown notebook:

```
---
title: "Hamilton County crAssphage manuscript figures version 4"
output: html_notebook
```

Main manuscript

FIND and Set up the environment

```{r eval=FALSE}
# some of the wwtps are not consistently named across files, this makes them the same
GetCongruentCharStr<-function(charstrchange,charstrdesired){
  charstrchange=toupper(charstrchange)
  nmVex<-vector(mode="character",length=length(charstrchange))
  for (i in 1:length(charstrdesired)) {
   idx=grep(charstrdesired[i],charstrchange)
   nmVex[idx]=charstrdesired[i]
  }
  return(nmVex)
}

source("~/Manuscripts/crAssphage/code/crAssphage_Driver.R")
```
Plot the crAssphage over time by WWTP
```{r}
ggplot(crAssFull, aes(x=as.Date(crAssFull$Date), y=zR,
color=crAssFull$Sewrshd))+geom_point(group=crAssFull$Sewrshd)+scale_x_date(date_minor_
breaks = "1 week")+xlab("Date")+ ylab("crAssphage load (copies/day)")+
scale_colour_discrete("WWTP")
```

INFORM

1) Is flow important?

a. Estimate α_{90} with and without flow, also estimate for flow as a response
b. Compare R2
c. Compare standardized coefficients

log10 flow

```{r eval=FALSE}
resp=2
lgSEDC=0
defalphavals=0
zR=zRlist[[resp]]
zS=zSlist1[[2]]
T90=seq(3,5,0.5) # grid of points for alpha 90
dfR=data.frame(rIDi,dti,zR) # create the Response data frame
dfR$dti=as.Date(dfR$dti) # change the date string to be a Date
zSj=zSlist1[[2]] # get the population per census block group
R2=vector("numeric",length(T90))
for (i in 1:length(T90)){
```

```
  dfS=sedc1(T90[i],sIDj,zSj,Tij,rIDi,dti)
  dfS$lg10res=log10(dfS$result)/std(log10(dfS$result))
  dfS$stdres=dfS$result/std(dfS$result)
  dfRS=merge(dfR,dfS,by.x=c("rIDi","dti"),by.y=c("rID","tID"))
  mdl=lm(zR~stdres,dfRS)
  if ((lgSEDC==1)==TRUE) {
  mdl=lm(zR~lg10res,dfRS)
  }
  tmp=summary(mdl)
  R2[i]= tmp$r.squared
}
R2list=list()
R2list[[1]]=R2


```

```{r}
zSj=zSlist1[[2]]
zR=zRlist[[2]]


optimvals1<-optim(par=0.1,ObjFun1,rIDi=rIDi, dti=dti, zR=zR, zSj=zSj,Tij=Tij, method="L-
BFGS-B", lower=1, upper=90)
optimvals1$par

dfS=GetMod1Vals(optimvals1$par,rIDi,dti,zR,zSj,Tij)
dfS$stdRes=dfS$result/std(dfS$result)
dfR=data.frame(rIDi,dti,zR)
dfR$dti=as.Date(dfR$dti)
dfRS=merge(dfR,dfS,by.x=c("rIDi","dti"),by.y=c("rID","tID"))
mdlflow=lm(dfRS$zR~dfRS$stdRes,y=TRUE, x=TRUE)
```


Plot the optimum and resulting maps

```{r eval=FALSE}
R2=R2list[[1]]
T90=seq(3,5,0.5)


max(R2)
idx=max(R2)==R2
xint=T90[idx]


if ((resp==2)==TRUE){
ggplot(NULL, aes(x=T90,y=R2))+geom_line()+xlab(TeX("$\\alpha_{90}$"))+
ylab(TeX("$R^2$"))+geom_vline(xintercept=xint,linetype="dashed",color="red")+annotate(geo
```

```
m="text",x=4.75, y=0.880, label=
TeX(sprintf("$\\alpha_{90}^*=%g$",xint)))+ylim(min(R2),max(R2))+xlim(min(T90),max(T90))
}


ShedData$ContrRate=exp(-2.3*(ShedData$TTavg)/xint)

ggplot(data=NULL)+geom_point(data=ShedData,aes(x=Xc_BG,y=Yc_BG,color=ContrRate))+s
cale_color_continuous(low="yellow",high="red")+geom_point(
aes(x=WWTPdf.sp@coords[,1],y=WWTPdf.sp@coords[,2]))

ShedData$`Signal Contribution Rate`=exp(-2.3*(ShedData$TTavg)/xint)
ShedData$`Information Loss Rate`=1-exp(-2.3*(ShedData$TTavg)/xint)

 # ggplot(data=NULL)+geom_point(data=ShedData,aes(x=Xc_BG,y=Yc_BG,color=`Signal
Contribution
Rate`))+scale_color_continuous(low="yellow",high="red",limits=c(0.1,0.7),oob=squish)+geom_
point( aes(x=WWTPdf.sp@coords[,1],y=WWTPdf.sp@coords[,2]))

tmpdata=merge(SewershedData, ShedData, by="GEOID")

ggplot(data=tmpdata) + geom_sf(aes(fill=`Information Loss Rate`), color =
NA)+scico::scale_fill_scico(palette = "lajolla", limits=c(0,1))+geom_sf(data=WWTPs2,
color="black", size=2, show.legend="point")+  theme_bw()+coord_sf(expand = F)
```

save some info

```{r eval=FALSE}
PopCont=list()
mdlslist=list()
dfS=sedc1(xint,sIDj,zSj,Tij,rIDi,dti)
dfR=data.frame(rIDi,dti,zR)
dfR$dti=as.Date(dfR$dti)
dfRS=merge(dfR,dfS,by.x=c("rIDi","dti"),by.y=c("rID","tID"))
PopCont[[1]]=log10(dfRS$result)/std(log10(dfRS$result))
ggplot(dfRS, aes(result,zR,color=rIDi))+geom_point()
dfRS$stdres=dfRS$result/std(dfRS$result)
mdltmp<-lm(zR~stdres,dfRS)
mdlslist[[1]]<-summary(mdltmp)
```

log10 crAssphage concentration

```{r eval=FALSE}
resp=6
lgSEDC=0
defalphavals=1
zR=zRlist[[resp]]
T90=seq(1,90,1)
dfR=data.frame(rIDi,dti,zR) # create the Response data frame
```

```
dfR$dti=as.Date(dfR$dti) # change the date string to be a Date
zSj=zSlist1[[2]] # get the population per census block group
R2=vector("numeric",length(T90))
for (i in 1:length(T90)){
  dfS=sedc1(T90[i],sIDj,zSj,Tij,rIDi,dti)
  dfS$lg10res=log10(dfS$result)/std(log10(dfS$result))
  dfS$stdres=dfS$result/std(dfS$result)
  dfRS=merge(dfR,dfS,by.x=c("rIDi","dti"),by.y=c("rID","tID"))
  mdl=lm(zR~stdres,dfRS)
  if ((lgSEDC==1)==TRUE) {
  mdl=lm(zR~lg10res,dfRS)
  }
  tmp=summary(mdl)
  R2[i]= tmp$r.squared
}
R2list[[2]]=R2
```

Plot the optimum and resulting map

```{r eval=FALSE}
R2=R2list[[2]]
T90=seq(1,90,1)


max(R2)
idx=max(R2)==R2
xint=T90[idx]


ggplot(NULL, aes(x=T90,y=R2))+geom_line()+xlab(TeX("$\\alpha_{90}$"))+
ylab(TeX("$R^2$"))+geom_vline(xintercept=xint,linetype="dashed",color="red")+annotate(geo
m="text",x=30, y=0.71805, label=
TeX(sprintf("$\\alpha_{90}^*=%g$",xint)))+ylim(min(R2),max(R2))+xlim(min(T90),max(T90))


ShedData$ContrRate=exp(-2.3*(ShedData$TTavg)/xint)

ggplot(data=NULL)+geom_point(data=ShedData,aes(x=Xc_BG,y=Yc_BG,color=ContrRate))+s
cale_color_continuous(low="yellow",high="red")+geom_point(
aes(x=WWTPdf.sp@coords[,1],y=WWTPdf.sp@coords[,2]))

ShedData$`Signal Contribution Rate`=exp(-2.3*(ShedData$TTavg)/xint)
 # ggplot(data=NULL)+geom_point(data=ShedData,aes(x=Xc_BG,y=Yc_BG,color=`Signal
Contribution
Rate`))+scale_color_continuous(low="yellow",high="red",limits=c(0.1,0.7),oob=squish)+geom_
point( aes(x=WWTPdf.sp@coords[,1],y=WWTPdf.sp@coords[,2]))
ShedData$`Information Loss Rate`=1-exp(-2.3*(ShedData$TTavg)/xint)
tmpdata=merge(SewershedData, ShedData, by="GEOID")
```

```
ggplot(data=tmpdata) + geom_sf(aes(fill=`Information Loss Rate`), color =
NA)+scico::scale_fill_scico(palette = "lajolla", limits=c(0,1))+geom_sf(data=WWTPs2,
color="black", size=2, show.legend="point")+  theme_bw()+coord_sf(expand = F)
```

save some info

```{r eval=FALSE}
dfS=sedc1(xint,sIDj,zSj,Tij,rIDi,dti)
dfR=data.frame(rIDi,dti,zR)
dfR$dti=as.Date(dfR$dti)
dfRS=merge(dfR,dfS,by.x=c("rIDi","dti"),by.y=c("rID","tID"))
PopCont[[2]]=log10(dfRS$result)/std(log10(dfRS$result))
ggplot(dfRS, aes(result,zR,color=rIDi))+geom_point()
dfRS$stdres=dfRS$result/std(dfRS$result)
mdltmp<-lm(dfRS$zR~dfRS$stdres,x=TRUE,y=TRUE)
mdlslist[[2]]<-summary(mdltmp)
```

log10 load

```{r eval=FALSE}
resp=4
lgSEDC=0
zR=zRlist[[resp]]
T90=seq(10,30,0.5)

dfR=data.frame(rIDi,dti,zR) # create the Response data frame
dfR$dti=as.Date(dfR$dti) # change the date string to be a Date
zSj=zSlist1[[2]] # get the population per census block group
R2=vector("numeric",length(T90))
for (i in 1:length(T90)){
  dfS=sedc1(T90[i],sIDj,zSj,Tij,rIDi,dti)
  dfS$lg10res=log10(dfS$result)/std(log10(dfS$result))
  dfS$stdres=dfS$result/std(dfS$result)
  dfRS=merge(dfR,dfS,by.x=c("rIDi","dti"),by.y=c("rID","tID"))
  mdl=lm(zR~stdres,dfRS)
  if ((lgSEDC==1)==TRUE) {
  mdl=lm(zR~lg10res,dfRS)
  }
  tmp=summary(mdl)
  R2[i]= tmp$r.squared
}
R2list[[3]]=R2
```

Model 1 confirm that the objective function gets you the same results

```{r eval=FALSE}
zSj=zSlist1[[2]]
zR=zRlist[[4]]
```

```
foldMSE=list()
foldpars=list()

set.seed(1000)
folds=create_folds(zR,k=5)
pars=vector("numeric",length(folds))
MSE=vector("numeric",length(folds))
i=1
for (fold in folds){
  rIDiTrain=rIDi[fold]
  dtiTrain=dti[fold]
  zRTrain=zR[fold]

  rIDiTest=rIDi[-fold]
  dtiTest=dti[-fold]
  zRTest=zR[-fold]

  optimvals1<-optim(par=0.1,ObjFun1,rIDi=rIDiTrain, dti=dtiTrain, zR=zRTrain,
zSj=zSj,Tij=Tij, method="L-BFGS-B", lower=1, upper=90)
  pars[i]=optimvals1$par
  dfS=GetMod1Vals(optimvals1$par,rIDiTest,dtiTest,zRTest,zSj,Tij)
  dfS$stdRes=dfS$result/std(dfS$result)
  dfR=data.frame(rIDiTest,dtiTest,zRTest)
  dfR$dtiTest=as.Date(dfR$dtiTest)
  dfRS=merge(dfR,dfS,by.x=c("rIDiTest","dtiTest"),by.y=c("rID","tID"))
  mdltmp=lm(zRTest~stdRes, dfRS)
  yhat=mdltmp$fitted.values
  y=dfRS$zR
  MSE[i]=sum((y-yhat)^2)/length(y)
  i=i+1
}

foldMSE[[1]]=MSE
foldpars[[1]]=pars

optimvals1<-optim(par=0.1,ObjFun1,rIDi=rIDi, dti=dti, zR=zR, zSj=zSj,Tij=Tij, method="L-
BFGS-B", lower=1, upper=90)
optimvals1$par

dfS=GetMod1Vals(optimvals1$par,rIDi,dti,zR,zSj,Tij)
dfS$stdRes=dfS$result/std(dfS$result)
dfR=data.frame(rIDi,dti,zR)
dfR$dti=as.Date(dfR$dti)
dfRS=merge(dfR,dfS,by.x=c("rIDi","dti"),by.y=c("rID","tID"))
mdl1=lm(dfRS$zR~dfRS$stdRes,y=TRUE, x=TRUE)
PopCont[[3]]=dfRS$result
ggplot(dfRS, aes(result,zR,color=rIDi))+geom_point()

```
```
```

Make plots

```{r eval=FALSE}
R2=R2list[[3]]
T90=seq(10,30,0.5)


max(R2)
idx=max(R2)==R2
xint=T90[idx]


if ((resp==6)==TRUE){
ggplot(NULL, aes(x=T90,y=R2))+geom_line()+xlab(TeX("$\\alpha_{90}$"))+
ylab(TeX("$R^2$"))+geom_vline(xintercept=xint,linetype="dashed",color="red")+annotate(geo
m="text",x=30, y=0.71805, label=
TeX(sprintf("$\\alpha_{90}^*=%g$",xint)))+ylim(min(R2),max(R2))+xlim(min(T90),max(T90))
}

if ((resp==5)==TRUE){
ggplot(NULL, aes(x=T90,y=R2))+geom_line()+xlab(TeX("$\\alpha_{90}$"))+
ylab(TeX("$R^2$"))+geom_vline(xintercept=xint,linetype="dashed",color="red")+annotate(geo
m="text",x=30, y=0.71805, label=
TeX(sprintf("$\\alpha_{90}^*=%g$",xint)))+ylim(min(R2),max(R2))+xlim(min(T90),max(T90))
}

if ((resp==4)==TRUE){
ggplot(NULL, aes(x=T90,y=R2))+geom_line()+xlab(TeX("$\\alpha_{90}$"))+
ylab(TeX("$R^2$"))+geom_vline(xintercept=xint,linetype="dashed",color="red")+annotate(geo
m="text",x=10.5, y=0.527, label=
TeX(sprintf("$\\alpha_{90}^*=%g$",xint)))+ylim(min(R2),max(R2))+xlim(min(T90),max(T90))
}

if ((resp==3)==TRUE){
ggplot(NULL, aes(x=T90,y=R2))+geom_line()+xlab(TeX("$\\alpha_{90}$"))+
ylab(TeX("$R^2$"))+geom_vline(xintercept=xint,linetype="dashed",color="red")+annotate(geo
m="text",x=33, y=0.3817, label=
TeX(sprintf("$\\alpha_{90}^*=%g$",xint)))+ylim(min(R2),max(R2))+xlim(min(T90),max(T90))
}

if ((resp==2)==TRUE){
ggplot(NULL, aes(x=T90,y=R2))+geom_line()+xlab(TeX("$\\alpha_{90}$"))+
ylab(TeX("$R^2$"))+geom_vline(xintercept=xint,linetype="dashed",color="red")+annotate(geo
m="text",x=4.75, y=0.880, label=
TeX(sprintf("$\\alpha_{90}^*=%g$",xint)))+ylim(min(R2),max(R2))+xlim(min(T90),max(T90))
}


if ((resp==1)==TRUE){
ggplot(NULL, aes(x=T90,y=R2))+geom_line()+xlab(TeX("$\\alpha_{90}$"))+
ylab(TeX("$R^2$"))+geom_vline(xintercept=xint,linetype="dashed",color="red")+annotate(geo
```

```
m="text",x=30, y=0.71805, label=
TeX(sprintf("$\\alpha_{90}^*=%g$",xint)))+ylim(min(R2),max(R2))+xlim(min(T90),max(T90))
}

ShedData$ContrRate=exp(-2.3*(ShedData$TTavg)/xint)

ggplot(data=NULL)+geom_point(data=ShedData,aes(x=Xc_BG,y=Yc_BG,color=ContrRate))+s
cale_color_continuous(low="yellow",high="red")+geom_point(
aes(x=WWTPdf.sp@coords[,1],y=WWTPdf.sp@coords[,2]))

ShedData$`Signal Contribution Rate`=exp(-2.3*(ShedData$TTavg)/xint)
ShedData$`Information Loss Rate`=1-exp(-2.3*(ShedData$TTavg)/xint)

 # ggplot(data=NULL)+geom_point(data=ShedData,aes(x=Xc_BG,y=Yc_BG,color=`Signal
Contribution
Rate`))+scale_color_continuous(low="yellow",high="red",limits=c(0.1,0.7),oob=squish)+geom_
point( aes(x=WWTPdf.sp@coords[,1],y=WWTPdf.sp@coords[,2]))

tmpdata=merge(SewershedData, ShedData, by="GEOID")

ggplot(data=tmpdata) + geom_sf(aes(fill=`Information Loss Rate`), color =
NA)+scico::scale_fill_scico(palette = "lajolla", limits=c(0,1))+geom_sf(data=WWTPs2,
color="black", size=2, show.legend="point")+  theme_bw()+coord_sf(expand = F)
```

Save some info

```{r eval=FALSE}
dfS=sedc1(xint,sIDj,zSj,Tij,rIDi,dti)
dfR=data.frame(rIDi,dti,zR)
dfR$dti=as.Date(dfR$dti)
dfRS=merge(dfR,dfS,by.x=c("rIDi","dti"),by.y=c("rID","tID"))
PopCont[[3]]=dfRS$result/std(dfRS$result)
ggplot(dfRS, aes(result,zR,color=rIDi))+geom_point()
dfRS$stdres=dfRS$result/std(dfRS$result)
mdltmp<-lm(zR~stdres,dfRS)
mdlslist[[3]]<-summary(mdltmp)
```

2) Is inhibition important?

a. Estimate $\alpha_{90,T}$ and γ_1 (corresponding to temperature effect on
$\alpha_{90,T}$.
b. Compare the magnitude of the correlations to the model with flow
c. Compare the p-values of the correlations \#### log10 load

```{r eval=FALSE}
resp=4
zR=zRlist[[resp]]
modfun=2
```

```
temp=crAssFull$wwT
xi=(temp-mean(temp))/std(temp) ## need to standardize temperature or code breaks!

alpha=c(seq(-0.2,-0.1,0.005))
T90=seq(8,18,0.2)

dfR=data.frame(rIDi,dti,zR)
dfR$dti=as.Date(dfR$dti)
zSj=zSlist1[[2]]
R2=matrix(NA,length(alpha),length(T90))
for (i in 1:length(alpha)){
  for (j in 1:length(T90)) {
    dfS=sedc2(T90[j],alpha[i],sIDj,zSj,Tij,rIDi,dti,xi,modfun)
    dfS$result[dfS$result==0]=min(dfS$result[dfS$result>0])/2
    dfS$stdres=dfS$result/std(dfS$result)
    dfRS=merge(dfR,dfS,by.x=c("rIDi","dti"),by.y=c("rID","tID"))
    mdl=lm(zR~stdres,dfRS)
    tmp=summary(mdl)
    R2[i,j]= tmp$r.squared
    #b1[i,j]=tmp$coefficients[2,1]
  }
}
R2list[[4]]=R2


```

Plot the optima and resulting maps

```{r eval=FALSE}
alpha=c(seq(-0.2,-0.1,0.005))
T90=seq(8,18,0.2)

R2=R2list[[4]]

idx=max(R2)==R2

ALPHA=repmat(alpha,1,length(T90))
T90rs=repmat(T90,length(alpha),1)
A=as.vector(ALPHA)
B=as.vector(T90rs)
C=as.vector(R2)

alphastar=A[idx]
T90star=B[idx]

tmptbl2=data.frame(A,B,C)
library(ggplot2)
#ggplot(tmptbl2, aes(x=A, y=B, fill=C)) + geom_tile()+
scale_fill_gradientn(colours=topo.colors(7),TeX("$R^2$"),low = "red", high = "black",
limits=c(0.075,0.3313),oob=squish)+xlab(TeX("$\\Delta$"))+ ylab(TeX("$\\alpha_{90}$")) #
```

```
 ggplot(tmptbl2, aes(x=A, y=B, fill=C)) + geom_tile()+
scale_fill_gradientn(colours=heat.colors(10),limits=c(0.560,0.565), oob=squish)+
ylab(TeX("$\\alpha_{90,T}$"))+xlab(TeX("$\\gamma_1$"))+geom_point(data=NULL,
aes(x=alphastar, y=T90star))+annotate(geom="text",x=alphastar+0.01, y=T90star-2, label=
TeX(sprintf("$\\alpha_{90,T}^*=%g, \\gamma_1^*=%g$",T90star,alphastar))) #


summerval=(max(crAssFull$wwT)-mean(crAssFull$wwT))/std(crAssFull$wwT)
summeralpha=T90star*exp(alphastar*summerval)


winterval=(min(crAssFull$wwT)-mean(crAssFull$wwT))/std(crAssFull$wwT)
winteralpha=T90star*exp(alphastar*winterval)


ShedData$`Signal Contribution Rate`=exp(-2.3*(ShedData$TTavg)/summeralpha)

tmpdata=merge(SewershedData, ShedData, by="GEOID")

ggplot(data=tmpdata) + geom_sf(aes(fill=`Signal Contribution Rate`), color =
NA)+scico::scale_fill_scico(palette = "lajolla", limits=c(0,1))+geom_sf(data=WWTPs2,
color="black", size=2, show.legend="point")+  theme_bw()+coord_sf(expand = F)

ShedData$`Signal Contribution Rate`=exp(-2.3*(ShedData$TTavg)/winteralpha)


tmpdata=merge(SewershedData, ShedData, by="GEOID")

ggplot(data=tmpdata) + geom_sf(aes(fill=`Signal Contribution Rate`), color =
NA)+scico::scale_fill_scico(palette = "lajolla", limits=c(0,1))+geom_sf(data=WWTPs2,
color="black", size=2, show.legend="point")+  theme_bw()+coord_sf(expand = F)
```

Save some info

```{r eval=FALSE}
dfS=sedc2(T90star,alphastar,sIDj,zSj,Tij,rIDi,dti,xi,modfun)
dfR=data.frame(rIDi,dti,zR)
dfR$dti=as.Date(dfR$dti)
dfRS=merge(dfR,dfS,by.x=c("rIDi","dti"),by.y=c("rID","tID"))
PopCont[[4]]=dfRS$result/std(dfRS$result)
ggplot(dfRS, aes(result,zR,color=rIDi))+geom_point()
dfRS$stdres=dfRS$result/std(dfRS$result)
mdltmp<-lm(zR~stdres,dfRS)
mdlslist[[4]]<-summary(mdltmp)
```

Model 2 confirm that the objective function gets you the same results

```{r eval=FALSE}


```

```
zR=zRlist[[4]]
zSj=zSlist1[[2]]
temp=crAssFull$wwT
m1=(temp-mean(temp))/std(temp) ## need to standardize temperature or code breaks!
#m1=zscorebyloc(temp,wwtpID)
modfun=2

set.seed(1000)
folds=create_folds(zR,k=5)
pars=matrix(data=NA,nrow=length(folds),ncol=2)
MSE=vector("numeric",length(folds))
i=1
for (fold in folds){
  rIDiTrain=rIDi[fold]
  dtiTrain=dti[fold]
  zRTrain=zR[fold]

  rIDiTest=rIDi[-fold]
  dtiTest=dti[-fold]
  zRTest=zR[-fold]

  optimvals2<-optim(par=c(0.1,0.1),ObjFun2,rIDi=rIDiTrain, dti=dtiTrain, zR=zRTrain, zSj=zSj,
m1=m1, modfun=modfun,Tij=Tij, method="L-BFGS-B", lower=c(1,-1), upper=c(90,0))
  pars[i,]=optimvals2$par
  dfS=GetMod2Vals(optimvals2$par,rIDiTest,dtiTest,zRTest,zSj,m1,modfun, Tij)
  dfR=data.frame(rIDiTest,dtiTest,zRTest)
  dfR$dtiTest=as.Date(dfR$dtiTest)
  dfRS=merge(dfR,dfS,by.x=c("rIDiTest","dtiTest"),by.y=c("rID","tID"))
  mdltmp=lm(zRTest~stdres, dfRS)
  yhat=mdltmp$fitted.values
  y=dfRS$zR
  MSE[i]=sum((y-yhat)^2)/length(y)
  i=i+1
}

foldMSE[[2]]=MSE
foldpars[[2]]=pars

optimvals2<-optim(par=c(0.1,0.1),ObjFun2,rIDi=rIDi, dti=dti, zR=zR, zSj=zSj, m1=m1,
modfun=modfun,Tij=Tij, method="L-BFGS-B", lower=c(1,-1), upper=c(90,0))
optimvals2$par


dfS=GetMod2Vals(optimvals2$par,rIDi,dti,zR,zSj,m1,modfun, Tij)
dfR=data.frame(rIDi,dti,zR)
dfR$dti=as.Date(dfR$dti)
dfRS=merge(dfR,dfS,by.x=c("rIDi","dti"),by.y=c("rID","tID"))
mdl2=lm(dfRS$zR~dfRS$stdres,y=TRUE, x=TRUE)
PopCont[[4]]=dfRS$stdres
ggplot(dfRS, aes(result,zR,color=rIDi))+geom_point()
```

3) What is the effect of population mobility on crAssphage signal from residential locations?

Model 3 with objective function

```{r eval=FALSE}
temp=crAssFull$wwT
m1=(temp-mean(temp))/std(temp) ## need to standardize temperature or code breaks!
#m1=zscorebyloc(temp,wwtpID)
mob1=crAssFull$resdP
m2=(mob1-mean(mob1))/std(mob1) ## need to standardize temperature or code breaks!
#m2=zscorebyloc(mob1,wwtpID)
modfun=2

set.seed(1000)
folds=create_folds(zR,k=5)
pars=matrix(data=NA,nrow=length(folds),ncol=3)
MSE=vector("numeric",length(folds))
i=1
for (fold in folds){
  rIDiTrain=rIDi[fold]
  dtiTrain=dti[fold]
  zRTrain=zR[fold]

  rIDiTest=rIDi[-fold]
  dtiTest=dti[-fold]
  zRTest=zR[-fold]

  optimvals3<-optim(par=c(0.1,0.1,0.1),ObjFun3,rIDi=rIDiTrain, dti=dtiTrain, zR=zRTrain,
zSj=zSj, m1=m1,m2=m2, modfun=modfun, Tij=Tij, method="L-BFGS-B", lower=c(1,-1,0),
upper=c(90,0,1))
  pars[i,]=optimvals3$par
  dfS=GetMod3Vals(optimvals3$par,rIDiTest,dtiTest,zRTest,zSj,m1,m2,modfun, Tij)
  dfR=data.frame(rIDiTest,dtiTest,zRTest)
  dfR$dtiTest=as.Date(dfR$dtiTest)
  dfRS=merge(dfR,dfS,by.x=c("rIDiTest","dtiTest"),by.y=c("rID","tID"))
  mdltmp=lm(zRTest~stdres, dfRS)
  yhat=mdltmp$fitted.values
  y=dfRS$zR
  MSE[i]=sum((y-yhat)^2)/length(y)
  i=i+1
}

foldMSE[[3]]=MSE
foldpars[[3]]=pars

optimvals3<-optim(par=c(0.1,0.1,0.1),ObjFun3,rIDi=rIDi, dti=dti, zR=zR, zSj=zSj,
m1=m1,m2=m2, modfun=modfun, Tij=Tij, method="L-BFGS-B", lower=c(1,-1,0),
upper=c(90,0,1))
optimvals3$par
```

```
dfS=GetMod3Vals(optimvals3$par,rIDi,dti,zR,zSj,m1,m2,modfun, Tij)
dfR=data.frame(rIDi,dti,zR)
dfR$dti=as.Date(dfR$dti)
dfRS=merge(dfR,dfS,by.x=c("rIDi","dti"),by.y=c("rID","tID"))
mdl3=lm(dfRS$zR~dfRS$stdres,y=TRUE, x=TRUE)
PopCont[[5]]=dfRS$stdres
ggplot(dfRS, aes(result,zR,color=rIDi))+geom_point()
```

Calculate values for optimum figures

```{r eval=FALSE}
modfun=2
temp=crAssFull$wwT
m1=(temp-mean(temp))/std(temp) ## need to standardize temperature or code breaks!
#m1=zscorebyloc(temp,wwtpID)
mob1=crAssFull$resdP
m2=(mob1-mean(mob1))/std(mob1) ## need to standardize temperature or code breaks!
#m2=zscorebyloc(mob1,wwtpID)
## "Residential" contributions to crAssphage concentration (with flow)
dfR=data.frame(rIDi,dti,zR)
dfR$dti=as.Date(dfR$dti)
zSj=zSlist1[[2]]
## visualize optimum for alpha
alpha=c(seq(5,12,0.5))
R2=vector("numeric",length(alpha))
for (i in 1:length(alpha)){
  dfS=sedc3(alpha[i],optimvals3$par[2], optimvals3$par[3],sIDj,zSj,Tij,rIDi,dti,m1,m2,modfun)
  dfS$stdres=dfS$result/std(dfS$result)
  dfRS=merge(dfR,dfS,by.x=c("rIDi","dti"),by.y=c("rID","tID"))
  mdl=lm(zR~stdres,dfRS)
  tmp=summary(mdl)
  R2[i]= tmp$r.squared

}
R2list[[5]]=R2

## visualize optimum for delta
delta=c(seq(-0.15,-0.05,.005))
R2=vector("numeric",length(delta))
for (i in 1:length(delta)){
  dfS=sedc3(optimvals3$par[1],delta[i], optimvals3$par[3],sIDj,zSj,Tij,rIDi,dti,m1,m2,modfun)
  dfS$stdres=dfS$result/std(dfS$result)
  dfRS=merge(dfR,dfS,by.x=c("rIDi","dti"),by.y=c("rID","tID"))
  mdl=lm(zR~stdres,dfRS)
  tmp=summary(mdl)
  R2[i]= tmp$r.squared

}
R2list[[6]]=R2
```

```
## visualize optimum for mew
mew=c(seq(.05,0.15,.005))
R2=vector("numeric",length(mew))
for (i in 1:length(mew)){
  dfS=sedc3(optimvals3$par[1],optimvals3$par[2], mew[i],sIDj,zSj,Tij,rIDi,dti,m1,m2,modfun)
  dfS$stdres=dfS$result/std(dfS$result)
  dfRS=merge(dfR,dfS,by.x=c("rIDi","dti"),by.y=c("rID","tID"))
  mdl=lm(zR~stdres,dfRS)
  tmp=summary(mdl)
  R2[i]= tmp$r.squared

}
R2list[[7]]=R2
```

Plot the optima and resulting maps

```{r eval=FALSE}
R2=R2list[[5]]
max(R2)
idx=max(R2)==R2
xint=alpha[idx]

ggplot(NULL, aes(x=alpha,y=R2))+geom_line()+xlab(TeX("$\\alpha_{90,TM}$"))+
ylab(TeX("$R^2$"))+geom_vline(xintercept=xint,linetype="dashed",color="red")+annotate(geo
m="text",x=7.5, y=0.335, label=
TeX(sprintf("$\\alpha_{90,TM}^*=%g$",xint)))+xlim(min(alpha),max(alpha))+ylim(min(R2),m
ax(R2))

## MOBILITY W TEMP AVG

idxearly=(crAssFull$Date<as.Date("2021-01-01"))
idxlate=(crAssFull$Date>=as.Date("2021-01-01"))
lowresdP=optimvals3$par[1]*exp(optimvals3$par[2]*mean(m1)+optimvals3$par[3]*min(m2))
highresdP=optimvals3$par[1]*exp(optimvals3$par[2]*mean(m1)+optimvals3$par[3]*max(m2))


ShedData$`Signal Contribution Rate`=exp(-2.3*(ShedData$TTavg)/lowresdP)

tmpdata=merge(SewershedData, ShedData, by="GEOID")

ggplot(data=tmpdata) + geom_sf(aes(fill=`Signal Contribution Rate`), color =
NA)+scico::scale_fill_scico(palette = "lajolla", limits=c(0,1))+geom_sf(data=WWTPs2,
color="black", size=2, show.legend="point")+  theme_bw()+coord_sf(expand = F)

ShedData$`Signal Contribution Rate`=exp(-2.3*(ShedData$TTavg)/highresdP)

tmpdata=merge(SewershedData, ShedData, by="GEOID")
```

```
ggplot(data=tmpdata) + geom_sf(aes(fill=`Signal Contribution Rate`), color =
NA)+scico::scale_fill_scico(palette = "lajolla", limits=c(0,1))+geom_sf(data=WWTPs2,
color="black", size=2, show.legend="point")+  theme_bw()+coord_sf(expand = F)

## TEMPERATURE W MOB AVG

lowT=optimvals3$par[1]*exp(optimvals3$par[2]*min(m1)+optimvals3$par[3]*mean(m2))
highT=optimvals3$par[1]*exp(optimvals3$par[2]*max(m1)+optimvals3$par[3]*mean(m2))


ShedData$`Signal Contribution Rate`=exp(-2.3*(ShedData$TTavg)/lowT)

tmpdata=merge(SewershedData, ShedData, by="GEOID")

ggplot(data=tmpdata) + geom_sf(aes(fill=`Signal Contribution Rate`), color =
NA)+scico::scale_fill_scico(palette = "lajolla", limits=c(0,1))+geom_sf(data=WWTPs2,
color="black", size=2, show.legend="point")+  theme_bw()+coord_sf(expand = F)

ShedData$`Signal Contribution Rate`=exp(-2.3*(ShedData$TTavg)/highT)


tmpdata=merge(SewershedData, ShedData, by="GEOID")

ggplot(data=tmpdata) + geom_sf(aes(fill=`Signal Contribution Rate`), color =
NA)+scico::scale_fill_scico(palette = "lajolla", limits=c(0,1))+geom_sf(data=WWTPs2,
color="black", size=2, show.legend="point")+  theme_bw()+coord_sf(expand = F)
```

Plot figures from manuscript Figure 1

```{r eval=FALSE}
minTmaxM=optimvals3$par[1]*exp(optimvals3$par[2]*min(m1)+optimvals3$par[3]*max(m2))
maxTminM=optimvals3$par[1]*exp(optimvals3$par[2]*max(m1)+optimvals3$par[3]*min(m2))

ShedData$`Signal Contribution Rate`=exp(-2.3*(ShedData$TTavg)/minTmaxM)
ShedData$`Information Loss Rate`=1-exp(-2.3*(ShedData$TTavg)/minTmaxM)


tmpdata=merge(SewershedData, ShedData, by="GEOID")

ggplot(data=tmpdata) + geom_sf(aes(fill=`Information Loss Rate`), color =
NA)+scico::scale_fill_scico(palette = "lajolla", limits=c(0,1))+geom_sf(data=WWTPs2,
color="black", size=2, show.legend="point")+  theme_bw()+coord_sf(expand = F)

ShedData$`Signal Contribution Rate`=exp(-2.3*(ShedData$TTavg)/maxTminM)
ShedData$`Information Loss Rate`=1-exp(-2.3*(ShedData$TTavg)/maxTminM)


tmpdata=merge(SewershedData, ShedData, by="GEOID")
```

```
ggplot(data=tmpdata) + geom_sf(aes(fill=`Information Loss Rate`), color =
NA)+scico::scale_fill_scico(palette = "lajolla", limits=c(0,1))+geom_sf(data=WWTPs2,
color="black", size=2, show.legend="point")+  theme_bw()+coord_sf(expand = F)

ShedData$`Signal Contribution`=exp(-2.3*(ShedData$TTavg)/minTmaxM)*zSlist1[[2]]

tmpdata=merge(SewershedData, ShedData, by="GEOID")

ggplot(data=tmpdata) + geom_sf(aes(fill=`Signal Contribution`), color =
NA)+scico::scale_fill_scico(palette = "lajolla", limits=c(0,10e15))+geom_sf(data=WWTPs2,
color="black", size=2, show.legend="point")+  theme_bw()+coord_sf(expand = F)

ShedData$`Signal Contribution`=exp(-2.3*(ShedData$TTavg)/maxTminM)*zSlist1[[2]]


tmpdata=merge(SewershedData, ShedData, by="GEOID")

ggplot(data=tmpdata) + geom_sf(aes(fill=`Signal Contribution`), color =
NA)+scico::scale_fill_scico(palette = "lajolla", limits=c(0,10e15))+geom_sf(data=WWTPs2,
color="black", size=2, show.legend="point")+  theme_bw()+coord_sf(expand = F)

R2=R2list[[6]]
max(R2)
idx=max(R2)==R2
xint=delta[idx]

ggplot(NULL, aes(x=delta,y=R2))+geom_line()+xlab(TeX("$\\gamma_1$"))+
ylab(TeX("$R^2$"))+geom_vline(xintercept=xint,linetype="dashed",color="red")+annotate(geo
m="text",x=-.3, y=0.34, label=
TeX(sprintf("$\\Delta^*=%g$",xint)))+xlim(min(delta),max(delta))+ylim(min(R2),max(R2))


R2=R2list[[7]]
max(R2)
idx=max(R2)==R2
xint=mew[idx]

ggplot(NULL, aes(x=mew,y=R2))+geom_line()+xlab(TeX("$\\gamma_2$"))+
ylab(TeX("$R^2$"))+geom_vline(xintercept=xint,linetype="dashed",color="red")+annotate(geo
m="text",x=0.25, y=0.35, label=
TeX(sprintf("$M^*=%g$",xint)))+xlim(min(mew),max(mew))+ylim(min(R2),max(R2))

ggplot(crAssFull,aes(as.Date(Date), resdP, color=Sewrshd,
group=Sewrshd))+geom_point(aes(group=Sewrshd))+geom_smooth()

ggplot(crAssFull,aes(as.Date(Date), resdP,
color=Sewrshd))+geom_point()+geom_smooth(aes(group=1))+xlab("Date")+ylab("% Time spent
at home from baseline")
```

4) Compare the residential contribution models

```{r eval=FALSE}
cor1=cor(PopCont[[1]],dfRS$zR)
cor2=cor(PopCont[[2]],dfRS$zR)
cor3=cor(PopCont[[3]],dfRS$zR)

ggplot(NULL,
aes(PopCont[[1]],zR,color=dfRS$rIDi))+geom_point()+geom_smooth(method=lm,aes(group=1),
fullrange=TRUE, color="black")
ggplot(NULL,
aes(PopCont[[2]],zR,color=dfRS$rIDi))+geom_point()+geom_smooth(method=lm,aes(group=1),
fullrange=TRUE, color="black")
ggplot(NULL,
aes(PopCont[[3]],dfRS$zR,color=dfRS$rIDi))+geom_point()+geom_smooth(method=lm,aes(gro
up=1),fullrange=TRUE, color="black")

mdlflow.cv=cv.lm(mdlflow,k=10,ks_test=TRUE,seed=set.seed(1))
mdl1.cv=cv.lm(mdl1,k=10,ks_test=TRUE,seed=set.seed(1))
mdl2.cv=cv.lm(mdl2,k=10,ks_test=TRUE,seed=set.seed(1))
mdl3.cv=cv.lm(mdl3,k=10,ks_test=TRUE,seed=set.seed(1))

mdlsMSE=c(mdl1.cv$MSE$mean,mdl2.cv$MSE$mean,mdl3.cv$MSE$mean)
mdlsSD=c(mdl1.cv$MSE$sd,mdl2.cv$MSE$sd,mdl3.cv$MSE$sd)
mdlslb=mdlsMSE-1.96*mdlsSD
mdlsub=mdlsMSE+1.96*mdlsSD
mdlname=c("1. SEDC", "2. SEDC-T", "3. SEDC-TM")

cvdf<-data.frame(mdlname,mdlsMSE,mdlsSD,mdlslb,mdlsub)
kable(cvdf)
ggplot(cvdf, aes(mdlname, mdlsMSE)) +geom_point() +geom_errorbar(aes(ymin = mdlslb, ymax
= mdlsub))+xlab("Model")+ylab("rMSE")
```

TEST

5) What other factors may influence marker levels at the WWTPs?

a. Explore mobility factors, pH, industrial flow, and sewer type (i.e, combined or separate) and the contributions from residences and commercial parcels with a linear model, and the sewersheds as their own factors

```{r eval=FALSE}
zSj1=zSlist1[[2]]
Tij=ShedData$TTavg
dfS1=sedc1(optimvals1$par,sIDj,zSj1,Tij,rIDi,dti)
dfS1$SEDCstd=dfS1$result/std(dfS1$result) # I did standardize here!
dfRS=merge(dfR,dfS1,by.x=c("rIDi","dti"),by.y=c("rID","tID"))

#
# dfS1=sedc3(optimvals3$par[1],optimvals3$par[2],optimvals3$par[3],sIDj,zSj1,Tij,rIDi,dti, m1,
m2, modfun)
```

```
# dfS1$stdres1=dfS1$result/std(dfS1$result) # I did standardize here!
# dfRS=merge(dfR,dfS1,by.x=c("rIDi","dti"),by.y=c("rID","tID"))

mdl2b=lm(zR~result,dfRS)
summary(mdl2b)
```

```{r eval=FALSE}
crAssFull$Date=as.Date(crAssFull$Date)
tmp=merge(crAssFull,dfRS,by.x=c("Sewrshd","Date"),by.y=c("rIDi","dti"))
```

Simplest model- Original (MX)

```{r eval=FALSE}

tmp$dTstnd=(tmp$T)/std(tmp$T)
tmp$dpHstnd=(tmp$pH)/std(tmp$pH)
tmp$CS=tmp$Sewer_Type=="Combined"
tmp$P=tmp$P_48h/std(tmp$P_48h)
tmp$retailrecstd=tmp$retailrecdP/std(tmp$retailrecdP)
tmp$grocpharmstd=tmp$grocpharmdP/std(tmp$grocpharmdP)
tmp$transitstd=tmp$transitdP/std(tmp$transitdP)
tmp$workstd=tmp$workdP/std(tmp$workdP)
tmp$resstnd=tmp$resdP/std(tmp$resdP)
tmp$IndFlow=tmp$Industrial_Flow/std(tmp$Industrial_Flow)
tmp$PopServ=tmp$Population_Served/std(tmp$Population_Served)
tmp$dTwwstnd=(tmp$wwT)/std(tmp$wwT)
tmp$PxCS=tmp$CS*tmp$P




y=log10(tmp$c_L*tmp$flowval*3785411.78)

tmptbl=tmp[,c(89, 91:102)]  #tmp[,c(89:98)]
X=data.matrix(tmptbl)
CorrX=cor(X)
kable(CorrX)

library(glmnet)
set.seed(1795)
cv_model <- cv.glmnet(X, y, alpha = 1, type.measure = "mse", nfolds = 10)


best_lambda <- cv_model$lambda.1se
best_model <- glmnet(X, y, alpha = 1, lambda = best_lambda)
coef(best_model)

y_predicted <- predict(best_model, s = best_lambda, newx = X)
```

```
#find SST and SSE
sst <- sum((y - mean(y))^2)
sse <- sum((y_predicted - y)^2)

#find R-Squared
rsq <- 1 - sse/sst
rsq

idxCoef=!(as.numeric(coef(best_model))=="0")
X2=X[,idxCoef[2:length(idxCoef)]]
tmpcoefs=BootLasso1(X2,y,best_lambda,10000)
kable(tmpcoefs)
kable(10^tmpcoefs)

betas=tmpcoefs[,1]
y_predicted2<-betas[1]+X2%*%betas[-1]

#find SST and SSE
sst <- sum((y - mean(y))^2)
sse <- sum((y_predicted2 - y)^2)

#find R-Squared
rsq <- 1 - sse/sst
rsq

#find adj R2
n=NROW(X2)
k=NCOL(X2)
AdjRsq<-1-(((1-rsq)*(n-1))/(n-k-1))
```

```{r eval=FALSE}
plot(cv_model)

ggplot(data=NULL,aes(x=y_predicted2,
y=y,color=tmp$Sewrshd))+geom_point()+geom_smooth(method=lm,aes(group=1),fullrange=TR
UE, color="black")+xlab("Predicted")+ylab("log10 crAssphage copies per
liter")+labs(color='Sewershed')

tbl2=tmpcoefs[2:NROW(tmpcoefs),]
tbl2$varnames=c("SEDC from Residential Populations", "pH", "Combined Sewer",
"Retail/Recreation^", "Grocery/Pharmacy^", "On Transit^",  "At Home^", "Industrial Flow",
"Wastewater Temperature", "Precipitation x Combined Sewer")
tbl2=tbl2[order(abs(tbl2$BSmeans)),]
p1=ggplot(tbl2, aes(varnames, BSmeans)) +geom_point()+geom_errorbar(aes(ymin = CIlower,
ymax = CIupper))+xlab("Standardized Variable")+ylab("Bootstrapped
Coefficient")+geom_hline(data=NULL, aes(yintercept=0,
color="red"),show.legend=FALSE)+coord_flip()

p1+theme(text = element_text(family = "serif"))
```

```

#### Model with modifiers (MM)

```{r eval=FALSE}
crAssFull$Date=as.Date(crAssFull$Date)
tmp=merge(crAssFull,dfRS,by.x=c("Sewrshd","Date"),by.y=c("rIDi","dti"))

tmp$dTstnd=(tmp$T)/std(tmp$T)
tmp$dpHstnd=(tmp$pH)/std(tmp$pH)
tmp$CS=tmp$Sewer_Type=="Combined"
tmp$P=tmp$P_48h/std(tmp$P_48h)
# tmp$P=(tmp$P_48h*tmp$Area_Ft)
# tmp$P[tmp$P==0]=min(tmp$P[!tmp$P==0])/2
# tmp$P=log(tmp$P)/std(log(tmp$P))
tmp$retailrecstd=tmp$retailrecdP/std(tmp$retailrecdP)
tmp$grocpharmstd=tmp$grocpharmdP/std(tmp$grocpharmdP)
tmp$transitstd=tmp$transitdP/std(tmp$transitdP)
tmp$workstd=tmp$workdP/std(tmp$workdP)
tmp$resstnd=tmp$resdP/std(tmp$resdP)
tmp$IndFlow=tmp$Industrial_Flow/std(tmp$Industrial_Flow)
tmp$PopServ=tmp$Population_Served/std(tmp$Population_Served)
tmp$dTwwstnd=(tmp$wwT)/std(tmp$wwT)
tmp$PxCS=tmp$CS*tmp$P
tmp$SEDCxMobRes=tmp$SEDCstd*tmp$resstnd
tmp$SEDCxMobGP=tmp$SEDCstd*tmp$grocpharmstd
tmp$SEDCxMobT=tmp$SEDCstd*tmp$transitstd
tmp$SEDCxMobRR=tmp$SEDCstd*tmp$retailrecstd
tmp$SEDCxMobW=tmp$SEDCstd*tmp$workstd


y=log10(tmp$c_L*tmp$flowval*3785411.78)

tmptbl=tmp[,c(89, 92:95,101:107)]  #tmp[,c(89:98)]
X=data.matrix(tmptbl)
CorrX=cor(X)
kable(CorrX)

library(glmnet)
set.seed(1795)
cv_model <- cv.glmnet(X, y, alpha = 1, type.measure = "mse", nfolds = 10)


best_lambda <- cv_model$lambda.1se
best_model <- glmnet(X, y, alpha = 1, lambda = best_lambda)
coef(best_model)

y_predicted <- predict(best_model, s = best_lambda, newx = X)

#find SST and SSE
sst <- sum((y - mean(y))^2)
```
```

```
sse <- sum((y_predicted - y)^2)

#find R-Squared
rsq <- 1 - sse/sst
rsq

idxCoef=!(as.numeric(coef(best_model))=="0")
X2=X[,idxCoef[2:length(idxCoef)]]
tmpcoefs=BootLasso1(X2,y,best_lambda,10000)
kable(tmpcoefs)
kable(10^tmpcoefs)

betas=tmpcoefs[,1]
y_predicted2<-betas[1]+X2%*%betas[-1]

#find SST and SSE
sst <- sum((y - mean(y))^2)
sse <- sum((y_predicted2 - y)^2)

#find R-Squared
rsq <- 1 - sse/sst
rsq

#find adj R2
n=NROW(X2)
k=NCOL(X2)
AdjRsq<-1-(((1-rsq)*(n-1))/(n-k-1))
```

```{r eval=FALSE}
plot(cv_model)

ggplot(data=NULL,aes(x=y_predicted2,
y=y,color=tmp$Sewrshd))+geom_point()+geom_smooth(method=lm,aes(group=1),fullrange=TR
UE, color="black")+xlab("Predicted")+ylab("log10 crAssphage copies per
liter")+labs(color='Sewershed')

# tbl2=tmpcoefs[2:NROW(tmpcoefs),]
# tbl2$varnames=c("Res. Cont", "pH", "Retail/Rec", "Groc./Pharm.", "On Transit",  "At Home",
"Wastewater Temp.", "Precip x Combined")
# tbl2=tbl2[order(abs(tbl2$BSmeans)),]
# ggplot(tbl2, aes(varnames, BSmeans)) +geom_point()+geom_errorbar(aes(ymin = CIlower,
ymax = CIupper))+xlab("Standardized Variable")+ylab("Bootstrapped
Coefficient")+geom_hline(data=NULL, aes(yintercept=0,
color="red"),show.legend=FALSE)+coord_flip()
```

Supporting Information

S1) INFORM Euclidean distance for HamCo

```{r eval=FALSE}
GetCongruentCharStr<-function(charstrchange,charstrdesired){
  charstrchange=toupper(charstrchange)
  nmVex<-vector(mode="character",length=length(charstrchange))
  for (i in 1:length(charstrdesired)) {
   idx=grep(charstrdesired[i],charstrchange)
   nmVex[idx]=charstrdesired[i]
  }
  return(nmVex)
}

source("~/Manuscripts/crAssphage/code/crAssphage_Driver.R")
```

SEDC with Euc. Distance as Proxy

```{r eval=FALSE}
Tij=ShedData$Dist2WWTP*0.0003048 # feet to km
zR=zRlist[[4]]
zSj=zSlist1[[2]]
optimvals1<-optim(par=0.01,ObjFun1,rIDi=rIDi, dti=dti, zR=zR, zSj=zSj,Tij=Tij, method="L-
BFGS-B", lower=0.01, upper=90)
optimvals1$par

dfS=GetMod1Vals(optimvals1$par,rIDi,dti,zR,zSj,Tij)
dfR=data.frame(rIDi,dti,zR)
dfR$dti=as.Date(dfR$dti)
dfRS=merge(dfR,dfS,by.x=c("rIDi","dti"),by.y=c("rID","tID"))
PopCont[[6]]=dfRS$result
ggplot(dfRS, aes(result,zR,color=rIDi))+geom_point()

```

SEDC-T with Euc. Distance as Proxy

```{r eval=FALSE}
resp=4
zR=zRlist[[resp]]
modfun=2

temp=crAssFull$wwT
xi=temp/std(temp) ## need to standardize temperature or code breaks!

alpha=c(seq(-0.2,-0.1,0.005))
T90=seq(0.3,20,0.5)

dfR=data.frame(rIDi,dti,zR)
dfR$dti=as.Date(dfR$dti)
zSj=zSlist1[[2]]
R2=matrix(NA,length(alpha),length(T90))
for (i in 1:length(alpha)){
```

```
  for (j in 1:length(T90)) {
    dfS=sedc2(T90[j],alpha[i],sIDj,zSj,Tij,rIDi,dti,xi,modfun)
    dfS$result[dfS$result==0]=min(dfS$result[dfS$result>0])/2
    dfS$stdres=dfS$result/std(dfS$result)
    dfRS=merge(dfR,dfS,by.x=c("rIDi","dti"),by.y=c("rID","tID"))
    mdl=lm(zR~stdres,dfRS)
    tmp=summary(mdl)
    R2[i,j]= tmp$r.squared
    #b1[i,j]=tmp$coefficients[2,1]
  }
}
R2list[[4]]=R2


```

```{r eval=FALSE}
alpha=c(seq(-0.2,-0.1,0.005))
T90=seq(0.3,20,0.5)
R2=R2list[[4]]

idx=max(R2)==R2

ALPHA=repmat(alpha,1,length(T90))
T90rs=repmat(T90,length(alpha),1)
A=as.vector(ALPHA)
B=as.vector(T90rs)
C=as.vector(R2)

alphastar=A[idx]
T90star=B[idx]

tmptbl2=data.frame(A,B,C)
library(ggplot2)
# ggplot(tmptbl2, aes(x=A, y=B, fill=C)) + geom_tile()+ scale_fill_gradient(TeX("$R^2$"),low =
"red", high = "black", limits=c(0.075,0.3313),oob=squish)+xlab(TeX("$\\Delta$"))+
ylab(TeX("$\\alpha_{90}$")) #

ggplot(tmptbl2, aes(x=A, y=B, fill=C)) + geom_tile()+scale_fill_steps2(
   low = "yellow",
   mid = "red",
   high = "black",
   midpoint = 0.54, n.breaks=12)+
ylab(TeX("$\\alpha_{90}(0)$"))+xlab(TeX("$\\gamma_1$"))+geom_point(data=NULL,
aes(x=alphastar, y=T90star))+annotate(geom="text",x=alphastar+0.01, y=T90star-2, label=
TeX(sprintf("$\\alpha_{90}(0)^*=%g, \\gamma_1^*=%g$",T90star,alphastar))) #


summerval=max(crAssFull$wwT)/std(crAssFull$wwT)
summeralpha=T90star*exp(alphastar*summerval)
```

```
winterval=min(crAssFull$wwT)/std(crAssFull$wwT)
winteralpha=T90star*exp(alphastar*winterval)


ShedData$`Signal Contribution Rate`=exp(-2.3*(ShedData$TTavg)/summeralpha)

tmpdata=merge(SewershedData, ShedData, by="GEOID")

ggplot(data=tmpdata) + geom_sf(aes(fill=`Signal Contribution Rate`), color =
NA)+scico::scale_fill_scico(palette = "lajolla", limits=c(0,1))+geom_sf(data=WWTPs2,
color="black", size=2, show.legend="point")+  theme_bw()+coord_sf(expand = F)

ShedData$`Signal Contribution Rate`=exp(-2.3*(ShedData$TTavg)/winteralpha)


tmpdata=merge(SewershedData, ShedData, by="GEOID")

ggplot(data=tmpdata) + geom_sf(aes(fill=`Signal Contribution Rate`), color =
NA)+scico::scale_fill_scico(palette = "lajolla", limits=c(0,1))+geom_sf(data=WWTPs2,
color="black", size=2, show.legend="point")+  theme_bw()+coord_sf(expand = F)
```

```{r eval=FALSE}
dfS=sedc2(T90star,alphastar,sIDj,zSj,Tij,rIDi,dti,xi,modfun)
dfR=data.frame(rIDi,dti,zR)
dfR$dti=as.Date(dfR$dti)
dfRS=merge(dfR,dfS,by.x=c("rIDi","dti"),by.y=c("rID","tID"))
PopCont[[7]]=log10(dfRS$result)/std(log10(dfRS$result))
ggplot(dfRS, aes(result,zR,color=rIDi))+geom_point()
dfRS$stdres=dfRS$result/std(dfRS$result)
mdltmp<-lm(zR~stdres,dfRS)
mdlslist[[4]]<-summary(mdltmp)
```

```{r eval=FALSE}
zSj=zSlist1[[2]]
temp=crAssFull$wwT
m1=temp/std(temp) ## need to standardize temperature or code breaks!
#m1=zscorebyloc(temp,wwtpID)
modfun=2
optimvals2<-optim(par=c(0.1,0.1),ObjFun2,rIDi=rIDi, dti=dti, zR=zR, zSj=zSj, m1=m1,
modfun=modfun, Tij=Tij,method="L-BFGS-B", lower=c(0.3,-1), upper=c(20,0))
optimvals2$par


dfS=GetMod2Vals(optimvals2$par,rIDi,dti,zR,zSj,m1,modfun,Tij)
dfR=data.frame(rIDi,dti,zR)
dfR$dti=as.Date(dfR$dti)
dfRS=merge(dfR,dfS,by.x=c("rIDi","dti"),by.y=c("rID","tID"))
PopCont[[2]]=dfRS$stdres
```

```
ggplot(dfRS, aes(result,zR,color=rIDi))+geom_point()
```

SEDC-TM with Euc. Distance as Proxy

```{r eval=FALSE}
temp=crAssFull$wwT
m1=temp/std(temp) ## need to standardize temperature or code breaks!
#m1=zscorebyloc(temp,wwtpID)
mob1=crAssFull$resdP
m2=mob1/std(mob1) ## need to standardize temperature or code breaks!
#m2=zscorebyloc(mob1,wwtpID)
modfun=2
optimvals3<-optim(par=c(0.1,0.1,0.1),ObjFun3,rIDi=rIDi, dti=dti, zR=zR, zSj=zSj,
m1=m1,m2=m2, modfun=modfun,Tij=Tij, method="L-BFGS-B", lower=c(.3,-1,0),
upper=c(90,0,1))
optimvals3$par

dfS=GetMod3Vals(optimvals3$par,rIDi,dti,zR,zSj,m1,m2,modfun,Tij)
dfR=data.frame(rIDi,dti,zR)
dfR$dti=as.Date(dfR$dti)
dfRS=merge(dfR,dfS,by.x=c("rIDi","dti"),by.y=c("rID","tID"))
PopCont[[8]]=dfRS$stdres
ggplot(dfRS, aes(result,zR,color=rIDi))+geom_point()
```

Compare the residential contribution models using Euc. Distance as Proxy

```{r eval=FALSE}
cor1=cor(PopCont[[1]],dfRS$zR)
cor2=cor(PopCont[[2]],dfRS$zR)
cor3=cor(PopCont[[3]],dfRS$zR)

ggplot(NULL,
aes(PopCont[[1]],zR,color=dfRS$rIDi))+geom_point()+geom_smooth(method=lm,aes(group=1),
fullrange=TRUE, color="black")
ggplot(NULL,
aes(PopCont[[2]],zR,color=dfRS$rIDi))+geom_point()+geom_smooth(method=lm,aes(group=1),
fullrange=TRUE, color="black")
ggplot(NULL,
aes(PopCont[[3]],dfRS$zR,color=dfRS$rIDi))+geom_point()+geom_smooth(method=lm,aes(gro
up=1),fullrange=TRUE, color="black")


mdl1=lm(dfRS$zR~PopCont[[1]],x=TRUE,y=TRUE)
mdl2=lm(dfRS$zR~PopCont[[2]],x=TRUE,y=TRUE)
mdl3=lm(dfRS$zR~PopCont[[3]],x=TRUE,y=TRUE)

mdl1.cv=cv.lm(mdl1,k=10,ks_test=TRUE,seed=set.seed(1))
mdl2.cv=cv.lm(mdl2,k=10,ks_test=TRUE,seed=set.seed(1))
mdl3.cv=cv.lm(mdl3,k=10,ks_test=TRUE,seed=set.seed(1))
```

```
mdlsMSE=c(mdl1.cv$MSE_sqrt$mean,mdl2.cv$MSE_sqrt$mean,mdl3.cv$MSE_sqrt$mean)
mdlsSD=c(mdl1.cv$MSE_sqrt$sd,mdl2.cv$MSE_sqrt$sd,mdl3.cv$MSE_sqrt$sd)
mdlslb=mdlsMSE-1.96*mdlsSD
mdlsub=mdlsMSE+1.96*mdlsSD
mdlname=c("1. SEDC", "2. SEDC-T", "3. SEDC-TM")

cvdf<-data.frame(mdlname,mdlsMSE,mdlsSD,mdlslb,mdlsub)
kable(cvdf)
ggplot(cvdf, aes(mdlname, mdlsMSE)) +geom_point() +geom_errorbar(aes(ymin = mdlslb, ymax
= mdlsub))+xlab("Model")+ylab("rMSE")
```

S2) TEST

```{r eval=FALSE}
zSj1=zSlist1[[2]]
dfS1=sedc1(optimvals1$par,sIDj,zSj1,Tij,rIDi,dti)
dfS1$stdres1EUC=dfS1$result/std(dfS1$result) # I did standardize here!
dfRS=merge(dfR,dfS1,by.x=c("rIDi","dti"),by.y=c("rID","tID"))


mdl2b=lm(zR~stdres1EUC,dfRS)
summary(mdl2b)

```

```{r eval=FALSE}
crAssFull$Date=as.Date(crAssFull$Date)
tmp=merge(crAssFull,dfRS,by.x=c("Sewrshd","Date"),by.y=c("rIDi","dti"))
```

Euc. MX with SEDC

```{r eval=FALSE}

tmp$dTstnd=(tmp$T)/std(tmp$T)
tmp$dpHstnd=(tmp$pH)/std(tmp$pH)
tmp$CS=tmp$Sewer_Type=="Combined"
tmp$P=tmp$P_48h/std(tmp$P_48h)
# tmp$P=(tmp$P_48h*tmp$Area_Ft)
# tmp$P[tmp$P==0]=min(tmp$P[!tmp$P==0])/2
# tmp$P=log(tmp$P)/std(log(tmp$P))
tmp$retailrecstd=tmp$retailrecdP/std(tmp$retailrecdP)
tmp$grocpharmstd=tmp$grocpharmdP/std(tmp$grocpharmdP)
tmp$transitstd=tmp$transitdP/std(tmp$transitdP)
tmp$workstd=tmp$workdP/std(tmp$workdP)
tmp$resstnd=tmp$resdP/std(tmp$resdP)
tmp$IndFlow=tmp$Industrial_Flow/std(tmp$Industrial_Flow)
tmp$PopServ=tmp$Population_Served/std(tmp$Population_Served)
```

```
y=log10(tmp$c_L*tmp$flowval)

tmptbl=tmp[,c(89, 91:102)]  #tmp[,c(89:98)]
X=data.matrix(tmptbl)
CorrX=cor(X)
kable(CorrX)

library(glmnet)
set.seed(2)
cv_model <- cv.glmnet(X, y, alpha = 1, type.measure = "mse", nfolds = 10)


best_lambda <- cv_model$lambda.1se
best_model <- glmnet(X, y, alpha = 1, lambda = best_lambda)
coef(best_model)

y_predicted <- predict(best_model, s = best_lambda, newx = X)

#find SST and SSE
sst <- sum((y - mean(y))^2)
sse <- sum((y_predicted - y)^2)

#find R-Squared
rsq <- 1 - sse/sst
rsq

idxCoef=!(as.numeric(coef(best_model))=="0")
X2=X[,idxCoef[2:length(idxCoef)]]
tmpcoefs=BootLasso1(X2,y,best_lambda,10000)
kable(tmpcoefs)
kable(10^tmpcoefs)

betas=tmpcoefs[,1]
y_predicted2<-betas[1]+X2%*%betas[-1]

#find SST and SSE
sst <- sum((y - mean(y))^2)
sse <- sum((y_predicted2 - y)^2)

#find R-Squared
rsq <- 1 - sse/sst
rsq

#find adj R2
n=NROW(X2)
k=NCOL(X2)
AdjRsq<-1-(((1-rsq)*(n-1))/(n-k-1))
```

```{r eval=FALSE}
plot(cv_model)

ggplot(data=NULL,aes(x=y_predicted2,
y=y,color=tmp$Sewrshd))+geom_point()+geom_smooth(method=lm,aes(group=1),fullrange=TR
UE, color="black")+xlab("Predicted")+ylab("log10 crAssphage copies per
liter")+labs(color='Sewershed')

tbl2=tmpcoefs[2:NROW(tmpcoefs),]
tbl2$varnames=c("Res. Cont", "Temperature", "pH", "Combined", "Precip.", "Ret/Rec",
"Groc./Pharm.", "On Transit", "At Home", "Ind. Flow")
tbl2=tbl2[order(abs(tbl2$BSmeans)),]
ggplot(tbl2, aes(varnames, BSmeans)) +geom_point() +geom_errorbar(aes(ymin = CIlower,
ymax = CIupper))+xlab("Std. Variable")+ylab("Bootstrapped
Coefficient")+geom_hline(data=NULL, aes(yintercept=0,
color="red"),show.legend=FALSE)+coord_flip()
```

S3) Hyperparameter sensitivity analysis

Change the response

```{r eval=FALSE}

nsim=1000
seeds=sample(1:10000,nsim)
pars=vector("numeric",nsim)
R2=vector("numeric",nsim)
betaval=vector("numeric",nsim)
for (i in 1:nsim){
set.seed(seeds[i])
resp=4
Tij=ShedData$TTavg
zR=zRlist[[resp]]
zR=zR+0.10*std(zR)*as.numeric(randn(length(zR),1))
zSj=zSlist1[[2]] # get the population per census block group

optimvals1<-optim(par=0.1,ObjFun1,rIDi=rIDi, dti=dti, zR=zR, zSj=zSj,Tij=Tij, method="L-
BFGS-B", lower=1, upper=90)
pars[i]=optimvals1$par

dfS=GetMod1Vals(optimvals1$par,rIDi,dti,zR,zSj,Tij)
dfR=data.frame(rIDi,dti,zR)
dfR$dti=as.Date(dfR$dti)
dfRS=merge(dfR,dfS,by.x=c("rIDi","dti"),by.y=c("rID","tID"))
mdltmp2<-lm(zR~stdRes,dfRS)
betaval[i]=as.numeric(mdltmp2$coefficients[2])
tmp=summary(mdltmp2)
R2[i]=tmp$r.squared
}
dZR=list(R2,betaval,pars)
```

```

#### change the "source" (no septic systems accounted for)

```{r eval=FALSE}

zR=zRlist[[4]]
zSj=zSlist1[[1]]
Tij=ShedData$TTavg

optimvals1<-optim(par=0.1,ObjFun1,rIDi=rIDi, dti=dti, zR=zR, zSj=zSj, Tij=Tij, method="L-BFGS-B", lower=1, upper=90)
optimvals1$par

dfS=GetMod1Vals(optimvals1$par,rIDi,dti,zR,zSj,Tij)
dfR=data.frame(rIDi,dti,zR)
dfR$dti=as.Date(dfR$dti)
dfRS=merge(dfR,dfS,by.x=c("rIDi","dti"),by.y=c("rID","tID"))
mdltmp2<-lm(zR~stdRes,dfRS)


```

change the "source" just add error

```{r eval=FALSE}
nsim=100
seeds=sample(1:10000,nsim)
pars=vector("numeric",nsim)
R2=vector("numeric",nsim)
betaval=vector("numeric",nsim)
for (i in 1:nsim){
set.seed(seeds[i])
zR=zRlist[[4]]
zSj=zSlist1[[2]]
Tij=ShedData$TTavg
zSj=zSj+0.10*std(zSj)*as.numeric(randn(length(zSj),1))
optimvals1<-optim(par=0.1,ObjFun1,rIDi=rIDi, dti=dti, zR=zR, zSj=zSj, Tij=Tij, method="L-BFGS-B", lower=1, upper=90)

dfS=GetMod1Vals(optimvals1$par,rIDi,dti,zR,zSj,Tij)
dfR=data.frame(rIDi,dti,zR)
dfR$dti=as.Date(dfR$dti)
dfRS=merge(dfR,dfS,by.x=c("rIDi","dti"),by.y=c("rID","tID"))
mdltmp2<-lm(zR~stdRes,dfRS)
pars[i]=optimvals1$par
betaval[i]=as.numeric(mdltmp2$coefficients[2])
tmp=summary(mdltmp2)
R2[i]=tmp$r.squared
}
```
```

```

#### change the travel time

```{r eval=FALSE}
nsim=100
seeds=sample(1:10000,nsim)
pars=vector("numeric",nsim)
R2=vector("numeric",nsim)
betaval=vector("numeric",nsim)
for (i in 1:nsim){
set.seed(seeds[i])
zR=zRlist[[4]]
zSj=zSlist1[[1]]
Tij=ShedData$TTavg
Tij=Tij+0.10*std(Tij)*as.numeric(randn(length(Tij),1))
zSj=zSlist1[[2]]
optimvals1<-optim(par=0.1,ObjFun1,rIDi=rIDi, dti=dti, zR=zR, zSj=zSj, Tij=Tij, method="L-BFGS-B", lower=1, upper=90)
optimvals1$par

dfS=GetMod1Vals(optimvals1$par,rIDi,dti,zR,zSj,Tij)
dfR=data.frame(rIDi,dti,zR)
dfR$dti=as.Date(dfR$dti)
dfRS=merge(dfR,dfS,by.x=c("rIDi","dti"),by.y=c("rID","tID"))
mdltmp2<-lm(zR~stdRes,dfRS)
pars[i]=optimvals1$par
betaval[i]=as.numeric(mdltmp2$coefficients[2])
tmp=summary(mdltmp2)
R2[i]=tmp$r.squared
}
```

Extra code for SI

```{r eval=FALSE}
tmpdata$Population=tmpdata$pop
ggplot(data=tmpdata) + geom_sf(aes(fill=Population), color =
NA)+scale_fill_gradient(low="azure3",
high="darkslategray")+geom_sf(data=ShedOutlines,aes(),fill=NA,
size=2)+geom_sf(data=WWTPs2, color="black", size=3, show.legend="point")+
theme_bw()+coord_sf(expand = F)+geom_sf_label(data=ShedOutlines,aes(label = Sewershed),
alpha=0.3)+xlab("Easting")+ylab("Northing")

tmpdata$`Travel Time (hours)`=tmpdata$TTavg
ggplot(data=tmpdata) + geom_sf(aes(fill=`Travel Time (hours)`), color =
NA)+scale_fill_gradient(low="beige",
high="darkolivegreen")+geom_sf(data=ShedOutlines,aes(),fill=NA,
size=2)+geom_sf(data=WWTPs2, color="black", size=3, show.legend="point")+
theme_bw()+coord_sf(expand = F)+geom_sf_label(data=ShedOutlines,aes(label = Sewershed),
alpha=0.3)+xlab("Easting")+ylab("Northing")
```
```

```
tmpdata$`Septic System Count`=tmpdata$SepCount
ggplot(data=tmpdata) + geom_sf(aes(fill=`Septic System Count`), color =
NA)+scale_fill_gradient(low="beige",
high="coral4")+geom_sf(data=ShedOutlines,aes(),fill=NA, size=2)+geom_sf(data=WWTPs2,
color="black", size=3, show.legend="point")+  theme_bw()+coord_sf(expand =
F)+geom_sf_label(data=ShedOutlines,aes(label = Sewershed),
alpha=0.3)+xlab("Easting")+ylab("Northing")

tmpdata$`Commercial Parcel Count`=tmpdata$ComParcels
ggplot(data=tmpdata) + geom_sf(aes(fill=`Commercial Parcel Count`), color =
NA)+scale_fill_gradient(low="beige",
high="deeppink4")+geom_sf(data=ShedOutlines,aes(),fill=NA,
size=2)+geom_sf(data=WWTPs2, color="black", size=3, show.legend="point")+
theme_bw()+coord_sf(expand = F)+geom_sf_label(data=ShedOutlines,aes(label = Sewershed),
alpha=0.3)+xlab("Easting")+ylab("Northing")

```

```{r eval=FALSE}

tmpdata$`Combined Sewer`=(tmpdata$Sewershed.x=="MILL
CREEK"|tmpdata$Sewershed.x=="MUDDY CREEK"|tmpdata$Sewershed.x=="TAYLOR
CREEK")

tmpdata$Tortuosity[tmpdata$Sewershed.x=="MILL
CREEK"]=mean(tmpdata$TTavg[tmpdata$Sewershed.x=="MILL
CREEK"]*60*60*0.6/tmpdata$Dist2WWTP[tmpdata$Sewershed.x=="MILL CREEK"]*0.3048)

tmpdata$Tortuosity[tmpdata$Sewershed.x=="MUDDY
CREEK"]=mean(tmpdata$TTavg[tmpdata$Sewershed.x=="MUDDY
CREEK"]*60*60*0.6/tmpdata$Dist2WWTP[tmpdata$Sewershed.x=="MUDDY
CREEK"]*0.3048)

tmpdata$Tortuosity[tmpdata$Sewershed.x=="TAYLOR
CREEK"]=mean(tmpdata$TTavg[tmpdata$Sewershed.x=="TAYLOR
CREEK"]*60*60*0.6/tmpdata$Dist2WWTP[tmpdata$Sewershed.x=="TAYLOR
CREEK"]*0.3048)
tmpdata$Tortuosity[tmpdata$Sewershed.x=="LITTLE
MIAMI"]=mean(tmpdata$TTavg[tmpdata$Sewershed.x=="LITTLE
MIAMI"]*60*60*0.6/tmpdata$Dist2WWTP[tmpdata$Sewershed.x=="LITTLE
MIAMI"]*0.3048)

ggplot(tmpdata, aes(x=Tortuosity, y=`Combined Sewer`))+geom_point()

```
Create datatable for all variables in the analysis

```{r}
# some of the wwtps are not consistently named across files, this makes them the same
GetCongruentCharStr<-function(charstrchange,charstrdesired){
```

```
  charstrchange=toupper(charstrchange)
  nmVex<-vector(mode="character",length=length(charstrchange))
  for (i in 1:length(charstrdesired)) {
   idx=grep(charstrdesired[i],charstrchange)
   nmVex[idx]=charstrdesired[i]
  }
  return(nmVex)
}

source("~/Manuscripts/crAssphage/code/crAssphage_Driver.R")

zSj=zSlist1[[2]]
Tij=ShedData$TTavg
zR=zRlist[[4]]

# SEDC
optimvals1<-optim(par=0.1,ObjFun1,rIDi=rIDi, dti=dti, zR=zR, zSj=zSj,Tij=Tij, method="L-
BFGS-B", lower=1, upper=90)
dfS=GetMod1Vals(optimvals1$par,rIDi,dti,zR,zSj,Tij)
SEDC=dfS$result
SEDCstd=dfS$result/std(dfS$result) # I did standardize here!

# SEDC-T
optimvals2<-optim(par=c(0.1,0.1),ObjFun2,rIDi=rIDi, dti=dti, zR=zR, zSj=zSj, m1=m1,
modfun=modfun,Tij=Tij, method="L-BFGS-B", lower=c(1,-1), upper=c(90,0))
dfS=GetMod2Vals(optimvals2$par,rIDi,dti,zR,zSj,m1,modfun, Tij)
sedcT=dfS$result
sedcTstd=dfS$result/std(dfS$result) # I did standardize here!

# SEDC-TM
optimvals3<-optim(par=c(0.1,0.1,0.1),ObjFun3,rIDi=rIDi, dti=dti, zR=zR, zSj=zSj,
m1=m1,m2=m2, modfun=modfun, Tij=Tij, method="L-BFGS-B", lower=c(1,-1,0),
upper=c(90,0,1))
dfS=GetMod3Vals(optimvals3$par,rIDi,dti,zR,zSj,m1,m2,modfun, Tij)
sedcTM=dfS$result
sedcTMstd=dfS$result/std(dfS$result) # I did standardize here!

# Euclidean SEDC
Tij=ShedData$Dist2WWTP*0.0003048 # feet to km
optimvals1<-optim(par=0.1,ObjFun1,rIDi=rIDi, dti=dti, zR=zR, zSj=zSj,Tij=Tij, method="L-
BFGS-B", lower=1, upper=90)
dfS=GetMod1Vals(optimvals1$par,rIDi,dti,zRTest,zSj,Tij)
sedcEUC=dfS$result
sedcEUCstd=dfS$result/std(dfS$result) # I did standardize here!

# df with all models
rID=dfS$rID
tID=dfS$tID
dfMods=data.frame(rID, tID, SEDC, sedcT, sedcTM, sedcEUC,SEDCstd, sedcTstd, sedcTMstd,
sedcEUCstd)
dfRS=merge(dfR,dfMods,by.x=c("rIDi","dti"),by.y=c("rID","tID"))
```

```
crAssFull$Date=as.Date(crAssFull$Date)
tmp=merge(crAssFull,dfRS,by.x=c("Sewrshd","Date"),by.y=c("rIDi","dti"))

tmp$dTstnd=(tmp$T)/std(tmp$T)
tmp$dpHstnd=(tmp$pH)/std(tmp$pH)
tmp$CS=tmp$Sewer_Type=="Combined"
tmp$P=tmp$P_48h/std(tmp$P_48h)
tmp$retailrecstd=tmp$retailrecdP/std(tmp$retailrecdP)
tmp$grocpharmstd=tmp$grocpharmdP/std(tmp$grocpharmdP)
tmp$transitstd=tmp$transitdP/std(tmp$transitdP)
tmp$workstd=tmp$workdP/std(tmp$workdP)
tmp$resstnd=tmp$resdP/std(tmp$resdP)
tmp$IndFlow=tmp$Industrial_Flow/std(tmp$Industrial_Flow)
tmp$PopServ=tmp$Population_Served/std(tmp$Population_Served)
tmp$dTwwstnd=(tmp$wwT)/std(tmp$wwT)
tmp$PxCS=tmp$CS*tmp$P
tmp$SEDCxMobRes=tmp$SEDCstd*tmp$resstnd
tmp$SEDCxMobGP=tmp$SEDCstd*tmp$grocpharmstd
tmp$SEDCxMobT=tmp$SEDCstd*tmp$transitstd
tmp$SEDCxMobRR=tmp$SEDCstd*tmp$retailrecstd
tmp$SEDCxMobW=tmp$SEDCstd*tmp$workstd

```
Get hyperparameter values for commercial location as sources
```{r}
zSj=zSlist1[[3]]
zR=zRlist[[4]]

foldMSE=list()
foldpars=list()

set.seed(1000)
folds=create_folds(zR,k=5)
pars=vector("numeric",length(folds))
MSE=vector("numeric",length(folds))
i=1
for (fold in folds){
  rIDiTrain=rIDi[fold]
  dtiTrain=dti[fold]
  zRTrain=zR[fold]

  rIDiTest=rIDi[-fold]
  dtiTest=dti[-fold]
  zRTest=zR[-fold]

  optimvals1<-optim(par=0.1,ObjFun1,rIDi=rIDiTrain, dti=dtiTrain, zR=zRTrain,
zSj=zSj,Tij=Tij, method="L-BFGS-B", lower=1, upper=90)
  pars[i]=optimvals1$par
  dfS=GetMod1Vals(optimvals1$par,rIDiTest,dtiTest,zRTest,zSj,Tij)
  dfS$stdRes=dfS$result/std(dfS$result)
```

```
  dfR=data.frame(rIDiTest,dtiTest,zRTest)
  dfR$dtiTest=as.Date(dfR$dtiTest)
  dfRS=merge(dfR,dfS,by.x=c("rIDiTest","dtiTest"),by.y=c("rID","tID"))
  mdltmp=lm(zRTest~stdRes, dfRS)
  yhat=mdltmp$fitted.values
  y=dfRS$zR
  MSE[i]=sum((y-yhat)^2)/length(y)
  i=i+1
}


optimvals1<-optim(par=0.1,ObjFun1,rIDi=rIDi, dti=dti, zR=zR, zSj=zSj,Tij=Tij, method="L-
BFGS-B", lower=1, upper=90)
optimvals1$par

dfS=GetMod1Vals(optimvals1$par,rIDi,dti,zR,zSj,Tij)
dfS$stdRes=dfS$result/std(dfS$result)
dfR=data.frame(rIDi,dti,zR)
dfR$dti=as.Date(dfR$dti)
dfRS=merge(dfR,dfS,by.x=c("rIDi","dti"),by.y=c("rID","tID"))
mdl1=lm(dfRS$zR~dfRS$stdRes,y=TRUE, x=TRUE)
ggplot(dfRS, aes(result,zR,color=rIDi))+geom_point()


zR=zRlist[[4]]
zSj=zSlist1[[3]]
temp=crAssFull$wwT
m1=(temp-mean(temp))/std(temp) ## need to standardize temperature or code breaks!
#m1=zscorebyloc(temp,wwtpID)
modfun=2

set.seed(1000)
folds=create_folds(zR,k=5)
pars=matrix(data=NA,nrow=length(folds),ncol=2)
MSE=vector("numeric",length(folds))
i=1
for (fold in folds){
  rIDiTrain=rIDi[fold]
  dtiTrain=dti[fold]
  zRTrain=zR[fold]

  rIDiTest=rIDi[-fold]
  dtiTest=dti[-fold]
  zRTest=zR[-fold]

  optimvals2<-optim(par=c(0.1,0.1),ObjFun2,rIDi=rIDiTrain, dti=dtiTrain, zR=zRTrain, zSj=zSj,
m1=m1, modfun=modfun,Tij=Tij, method="L-BFGS-B", lower=c(1,-1), upper=c(90,0))
  pars[i,]=optimvals2$par
  dfS=GetMod2Vals(optimvals2$par,rIDiTest,dtiTest,zRTest,zSj,m1,modfun, Tij)
  dfR=data.frame(rIDiTest,dtiTest,zRTest)
  dfR$dtiTest=as.Date(dfR$dtiTest)
```

```
  dfRS=merge(dfR,dfS,by.x=c("rIDiTest","dtiTest"),by.y=c("rID","tID"))
  mdltmp=lm(zRTest~stdres, dfRS)
  yhat=mdltmp$fitted.values
  y=dfRS$zR
  MSE[i]=sum((y-yhat)^2)/length(y)
  i=i+1
}



optimvals2<-optim(par=c(0.1,0.1),ObjFun2,rIDi=rIDi, dti=dti, zR=zR, zSj=zSj, m1=m1,
modfun=modfun,Tij=Tij, method="L-BFGS-B", lower=c(1,-1), upper=c(90,0))
optimvals2$par

temp=crAssFull$wwT
m1=(temp-mean(temp))/std(temp) ## need to standardize temperature or code breaks!
#m1=zscorebyloc(temp,wwtpID)
mob1=crAssFull$resdP
m2=(mob1-mean(mob1))/std(mob1) ## need to standardize temperature or code breaks!
#m2=zscorebyloc(mob1,wwtpID)
modfun=2

set.seed(1000)
folds=create_folds(zR,k=5)
pars=matrix(data=NA,nrow=length(folds),ncol=3)
MSE=vector("numeric",length(folds))
i=1
for (fold in folds){
  rIDiTrain=rIDi[fold]
  dtiTrain=dti[fold]
  zRTrain=zR[fold]

  rIDiTest=rIDi[-fold]
  dtiTest=dti[-fold]
  zRTest=zR[-fold]

  optimvals3<-optim(par=c(0.1,0.1,0.1),ObjFun3,rIDi=rIDiTrain, dti=dtiTrain, zR=zRTrain,
zSj=zSj, m1=m1,m2=m2, modfun=modfun, Tij=Tij, method="L-BFGS-B", lower=c(1,-1,0),
upper=c(90,0,1))
  pars[i,]=optimvals3$par
  dfS=GetMod3Vals(optimvals3$par,rIDiTest,dtiTest,zRTest,zSj,m1,m2,modfun, Tij)
  dfR=data.frame(rIDiTest,dtiTest,zRTest)
  dfR$dtiTest=as.Date(dfR$dtiTest)
  dfRS=merge(dfR,dfS,by.x=c("rIDiTest","dtiTest"),by.y=c("rID","tID"))
  mdltmp=lm(zRTest~stdres, dfRS)
  yhat=mdltmp$fitted.values
  y=dfRS$zR
  MSE[i]=sum((y-yhat)^2)/length(y)
  i=i+1
}
```

```
foldMSE[[3]]=MSE
foldpars[[3]]=pars

optimvals3<-optim(par=c(0.1,0.1,0.1),ObjFun3,rIDi=rIDi, dti=dti, zR=zR, zSj=zSj,
m1=m1,m2=m2, modfun=modfun, Tij=Tij, method="L-BFGS-B", lower=c(1,-1,0),
upper=c(90,0,1))
optimvals3$par
```

```{r}
MainData<-tmp[,c(1,2,57,61, 62, 66, 70,71,73, 74, 75, 80:86,88:113)]
write.csv(MainData,"C:/Users/CWIESNER/OneDrive - Environmental Protection Agency
(EPA)/Profile/Documents/Manuscripts/crAssphage/data/crAssphage_main.csv", row.names =
FALSE)
write.csv(ShedData,"C:/Users/CWIESNER/OneDrive - Environmental Protection Agency
(EPA)/Profile/Documents/Manuscripts/crAssphage/data/ShedData.csv", row.names = FALSE)
```

S11 Additional calculations

11.1 The populations represented by wastewater data (code)

unique(crAssphage_main_v1$SEDC[crAssphage_main_v1$Sewrshd=="MILL
CREEK"])/sum(zS[ShedData$Sewershed=="MILL CREEK"])

[1] 0.2838719

unique(crAssphage_main_v1$SEDC[crAssphage_main_v1$Sewrshd=="LITTLE
MIAMI"])/sum(zS[ShedData$Sewershed=="LITTLE MIAMI"])

[1] 0.4068388

unique(crAssphage_main_v1$SEDC[crAssphage_main_v1$Sewrshd=="MUDDY
CREEK"])/sum(zS[ShedData$Sewershed=="MUDDY CREEK"])

[1] 0.4355066

unique(crAssphage_main_v1$SEDC[crAssphage_main_v1$Sewrshd=="TAYLOR
CREEK"])/sum(zS[ShedData$Sewershed=="TAYLOR CREEK"])

[1] 0.2917401

11.2 The expected travel time for 100 meters Euclidean distance

$$T_{ij}^{unknown} \approx \frac{\tau 100\ meters}{0.6\ \mathrm{ms}^{-1}} \quad \text{(Eq. S7)}$$

$$E[\tau] = E[\frac{v^{gravity} T_{ij}}{D_{ij}^{Euclidean}}] \quad \text{(Eq. S8)}$$

Using the Euclidean distances, $D_{ij}^{Euclidean}$, in meters from each census block group centroid to each wastewater treatment plant, and the pipe travel times, T_{ij}, based on the hydraulically-based census block group outlet to the wastewater treatment plant that we had available from modeling, we ran a regression without an intercept to estimate some measure of tortuosity, τ (i.e., measure of the curviness of the pipes). We assumed that the wastewater was flowing at a standard velocity for gravity pipes, $v^{gravity}$, of 0.6 ms^{-1} which would correspond to maximum velocity conditions, which are known. Using this estimate, $E[\tau]$, and $v^{gravity}$ we obtained an estimate of the travel time in the pipes we would expect for a 100 meter Euclidean distance.

We found E[τ]=8.08 which would mean that a 100 meter Euclidean distance would correspond with a travel time of approximately 22 minutes. We note that if we used a velocity of even 0.5 meters per second, this would still correspond to <1 hour travel time.

Bibliography

(1) Jacobs Engineering Group; Shamsi, U. M.; Gamble, B.; Metropolitan Sewer District of Greater Cincinnati; Koran, J.; Metropolitan Sewer District of Greater Cincinnati. Cincinnati's SWMM model: A journey through time. *JWMM* **2016**.

(2) U.S. Census Bureau. Persons Per Household 2016-2020 https://www.census.gov/quickfacts/fact/table/OH/BZA010220 (accessed Jul 14, 2022).

(3) Kumar, M.; Alamin, M.; Kuroda, K.; Dhangar, K.; Hata, A.; Yamaguchi, H.; Honda, R. Potential discharge, attenuation and exposure risk of SARS-CoV-2 in natural water bodies receiving treated wastewater. *npj Clean Water* **2021**, *4*, 8.

(4) Hart, O. E.; Halden, R. U. Computational analysis of SARS-CoV-2/COVID-19 surveillance by wastewater-based epidemiology locally and globally: Feasibility, economy, opportunities and challenges. *Sci. Total Environ.* **2020**, *730*, 138875.

(5) Hart, O. E.; Halden, R. U. Modeling wastewater temperature and attenuation of sewage-borne biomarkers globally. *Water Res.* **2020**, *172*, 115473.