**SUPPLEMENTAL INFORMATION**

ANALYZING FACTORS WHICH AFFECT *LEGIONELLA* OCCURENCE IN A FULL-SCALE GREEN BUILDING PREMISE PLUMBING SYSTEM

By

Ryan Julien

**File Description:**

The following explains each of the analyses conducted in this dissertation. Analyses from each chapter in the dissertation are separated into sections in this document. All code was executed in R (version 4.0.4), in the RStudio environment (version 1.4.1106).

# Variable selection analysis code

Variable selection analysis was conducted using a single RStudio script. The script is included below for reference:

# Load libraries

library(dplyr)

library(ggplot2)

library(GGally)

library(ggthemes)

scale\_colour\_discrete <- scale\_colour\_colorblind

# Import Analytical Data --------------------------------------------------

# Personal Laptop

setwd("D:/PhD/Publications/ReNEWW Exploratory Data Analysis/Model/Data/")

# MSU Desktop

#setwd("C:/Users/julienry/Desktop/Ryan's Folder/PhD/Publications/ReNEWW Exploratory Data Analysis/Model/Data")

#setwd("T:/Research BackUp/Ryan/ReNEWW House Data/Data/Analytical")

wq.df <- as.data.frame(read.csv("reneww\_data\_20190712.csv", stringsAsFactors = FALSE))

# Or choose these manually

#setwd(choose.dir())

#wq.df <- as.data.frame(read.csv(choose.files(), stringsAsFactors = FALSE))

# Arrange by fixture

wq.df$Location <- factor(wq.df$Location,

levels = c("SL", "cold.kitchen.sink", "cold.bath2.sink", "WH", "hot.kitchen.sink", "hot.bath2.sink", "mix.bath2.shower"),

ordered = TRUE)

wq.df <- arrange(wq.df, Location)

wq.df$System <- rep(NA, nrow(wq.df))

for (i in 1:nrow(wq.df)){

# Define Hot/Cold/Mixed systems

if (wq.df$Location[i] %in% c("WH", "hot.kitchen.sink", "hot.bath2.sink")){

wq.df$System[i] <- "H"

}

else if (wq.df$Location[i] %in% c("SL", "cold.kitchen.sink", "cold.bath2.sink")){

wq.df$System[i] <- "C"

}

else if (wq.df$Location[i] %in% c("mix.bath2.shower")){

wq.df$System[i] <- "M"

}

# Assume sample was collected at noon if not labeled

if (wq.df[i,"Time"] == "M"){

wq.df[i,"Time"] <- as.character("12:00")

}

}

rm(i)

# Transform Analytical Results --------------------------------------------

wq.df$date\_time <- with(wq.df, as.POSIXct(paste(Date, Time), format = "%m/%d/%Y %H:%M"))

# Note that missing sample times were set to noon

wq.df$Date <- as.Date(wq.df$Date, tryFormats = "%m/%d/%Y")

wq.df$Time <- as.POSIXct(wq.df$Time, format = "%H:%M")

wq.df$pH <- as.numeric(wq.df$pH)

wq.df$Temp <- as.numeric(wq.df$Temp)

wq.df$DO <- as.numeric(wq.df$DO)

# Assumes 50% of detection limit for all ND chlorine samples

wq.df$Total.Cl[wq.df$Total.Cl == 0 | wq.df$Total.Cl == "0.00"] <- 0.1/2

wq.df$Total.Cl <- as.numeric(wq.df$Total.Cl)

wq.df$Free.Cl[wq.df$Free.Cl == 0 | wq.df$Free.Cl == "0.00"] <- 0.1/2

wq.df$Free.Cl <- as.numeric(wq.df$Free.Cl)

wq.df$TCC.1 <- as.numeric(wq.df$TCC.1)

wq.df$TCC.2 <- as.numeric(wq.df$TCC.2)

wq.df$TCC.3 <- as.numeric(wq.df$TCC.3)

wq.df$TCC.4 <- as.numeric(wq.df$TCC.4)

wq.df$TCC.5 <- as.numeric(wq.df$TCC.5)

wq.df$TCC.6 <- as.numeric(wq.df$TCC.6)

wq.df$TCC <- rowMeans(cbind(wq.df$TCC.1, wq.df$TCC.2, wq.df$TCC.3, wq.df$TCC.4, wq.df$TCC.5, wq.df$TCC.6), na.rm = TRUE)

wq.df$TCC[is.nan(wq.df$TCC)] <- NA # Replace 'NaN' with 'NA'

wq.df$HPC <- as.numeric(wq.df$HPC)

# Legionella

# Assumes 50% of detection limit for all ND Legionella samples

wq.df$Leg.sp.1[!(wq.df$Leg.sp.1 > 0)] <- as.numeric(gsub("[^0-9.]", "", wq.df$Leg.sp.1[!(wq.df$Leg.sp.1 > 0)]))/2

wq.df$Leg.sp.2[!(wq.df$Leg.sp.2 > 0)] <- as.numeric(gsub("[^0-9.]", "", wq.df$Leg.sp.2[!(wq.df$Leg.sp.2 > 0)]))/2

wq.df$Leg.sp.3[!(wq.df$Leg.sp.3 > 0)] <- as.numeric(gsub("[^0-9.]", "", wq.df$Leg.sp.3[!(wq.df$Leg.sp.3 > 0)]))/2

wq.df$Leg.sp.1 <- as.numeric(wq.df$Leg.sp.1)

wq.df$Leg.sp.2 <- as.numeric(wq.df$Leg.sp.2)

wq.df$Leg.sp.3 <- as.numeric(wq.df$Leg.sp.3)

#Take average of all

wq.df$Leg.sp <- rowMeans(cbind(wq.df$Leg.sp.1, wq.df$Leg.sp.2, wq.df$Leg.sp.3), na.rm = TRUE)

# TOC

wq.df$TOC.1 <- as.numeric(wq.df$TOC.1)

wq.df$TOC.2 <- as.numeric(wq.df$TOC.2)

wq.df$TOC <- rowMeans(cbind(wq.df$TOC.1, wq.df$TOC.2), na.rm = TRUE)

wq.df$TOC[is.nan(wq.df$TOC)] <- NA # Replace 'NaN' with 'NA'

#DOC

wq.df$DOC.1 <- as.numeric(wq.df$DOC.1)

wq.df$DOC.2 <- as.numeric(wq.df$DOC.2)

wq.df$DOC <- rowMeans(cbind(wq.df$DOC.1, wq.df$DOC.2), na.rm = TRUE)

wq.df$DOC[is.nan(wq.df$DOC)] <- NA # Replace 'NaN' with 'NA'

# Alkalinity

wq.df$Alka.1 <- as.numeric(wq.df$Alka.1)

wq.df$Alka.2 <- as.numeric(wq.df$Alka.2)

wq.df$Alka <- rowMeans(cbind(wq.df$Alka.1, wq.df$Alka.2), na.rm = TRUE)

wq.df$Alka[is.nan(wq.df$Alka)] <- NA # Replace 'NaN' with 'NA'

# Assumes 50% of detection limit for all ND TTHM samples

wq.df$TTHM.1[wq.df$TTHM.1 == 0 | wq.df$TTHM.1 == "0.00"] <- 0.1/2

wq.df$TTHM.1 <- as.numeric(wq.df$TTHM.1)

wq.df$TTHM.2[wq.df$TTHM.2 == 0 | wq.df$TTHM.2 == "0.00"] <- 0.1/2

wq.df$TTHM.2 <- as.numeric(wq.df$TTHM.2)

wq.df$TTHM <- rowMeans(cbind(wq.df$TTHM.1, wq.df$TTHM.2), na.rm = TRUE)

wq.df$TTHM[is.nan(wq.df$TTHM)] <- NA # Replace 'NaN' with 'NA'

wq.df$System <- as.factor(wq.df$System)

# Import Usage Data -------------------------------------------------------

use.data <- as.data.frame(read.csv("allEvents 201706-201811.csv"))

# Fix timestamps

use.data$startTime <- as.POSIXct(strptime(use.data$startTime, "%m/%d/%Y %H:%M:%S"))

use.data$endTime <- as.POSIXct(strptime(use.data$endTime, "%m/%d/%Y %H:%M:%S"))

# Define naming convention

fixt.names <- rbind(c("Bath2\_Shower", "Bath2\_Sink\_Cold\_vol", "Bath2\_Sink\_Hot\_vol", "City\_Main\_vol", "DHW\_Cold\_Inlet\_vol", "DHW\_Hot\_Return\_vol", "Island\_Cold\_vol", "Island\_Hot\_vol", "Kitchen\_Cold\_vol", "Kitchen\_Sink\_Hot\_vol"),

c("mix.bath2.shower", "cold.bath2.sink", "hot.bath2.sink", "SL", "WH", "hot.recycle", "cold.kitchen.isl", "hot.kitchen.isl", "cold.kitchen.sink", "hot.kitchen.sink"))

# Develop Usage Stats -----------------------------------------------------

# Settings

metrics <- c("num.events", "vol.events", "meanTSL", "maxTSL")

#periods <- c(1,2,3,5,7,10,14,21,28,35,42)

#periods <- seq(from=1, to=45, by=1)

periods <- 14

age.names <- paste(rep(metrics, 3), rep(paste("day", periods, sep = ""), each = 3), sep = ".")

min.tSL <- 5 # Minimum duration between events (seconds)

num.events <- as.data.frame(matrix(NA, nrow = nrow(wq.df), ncol = length(periods), dimnames = list(NULL, paste0(rep("day"), periods))))

vol.events <- as.data.frame(matrix(NA, nrow = nrow(wq.df), ncol = length(periods), dimnames = list(NULL, paste0(rep("day"), periods))))

meanTSL <- as.data.frame(matrix(NA, nrow = nrow(wq.df), ncol = length(periods), dimnames = list(NULL, paste0(rep("day"), periods))))

maxTSL <- as.data.frame(matrix(NA, nrow = nrow(wq.df), ncol = length(periods), dimnames = list(NULL, paste0(rep("day"), periods))))

m95TSL <- as.data.frame(matrix(NA, nrow = nrow(wq.df), ncol = length(periods), dimnames = list(NULL, paste0(rep("day"), periods))))

# Calculate each water age metric

# For each sample, s

for (s in 1:nrow(wq.df)){

# For each time period, p

for (p in 1:length(periods)){

# Actually calculate water age metrics

# Number of Events

num.events.inst <- nrow(subset(use.data, variableName == fixt.names[1,][fixt.names[2,] %in% wq.df$Location[s]] &

endTime > wq.df$date\_time[s] - (periods[p]\*60\*60\*24) &

endTime <= wq.df$date\_time[s] &

timeSinceLastEvent >= min.tSL))

if (num.events.inst == 0){

num.events.inst <- 1

vol.events.inst <- 1

meanTSL.inst <- NA

} else {

# Cumulative Volume

vol.events.inst <- sum(subset(use.data, variableName == fixt.names[1,][fixt.names[2,] %in% wq.df$Location[s]] &

endTime > wq.df$date\_time[s] - (periods[p]\*60\*60\*24) &

endTime <= wq.df$date\_time[s],

select = sum),na.rm = TRUE)

# Mean timeSinceLast

meanTSL.inst <- mean(subset(use.data, variableName == fixt.names[1,][fixt.names[2,] %in% wq.df$Location[s]] &

endTime > wq.df$date\_time[s] - (periods[p]\*60\*60\*24) &

endTime <= wq.df$date\_time[s] &

timeSinceLastEvent >= min.tSL)[,"timeSinceLastEvent"])

# Max timeSinceLast

maxTSL.inst <- max(subset(use.data, variableName == fixt.names[1,][fixt.names[2,] %in% wq.df$Location[s]] &

endTime > wq.df$date\_time[s] - (periods[p]\*60\*60\*24) &

endTime <= wq.df$date\_time[s] &

timeSinceLastEvent >= min.tSL)[,"timeSinceLastEvent"])

# Max timeSinceLast

m95TSL.inst <- quantile(subset(use.data, variableName == fixt.names[1,][fixt.names[2,] %in% wq.df$Location[s]] &

endTime > wq.df$date\_time[s] - (periods[p]\*60\*60\*24) &

endTime <= wq.df$date\_time[s] &

timeSinceLastEvent >= min.tSL)[,"timeSinceLastEvent"], 0.975)

}

# Save output

num.events[s, p] <- num.events.inst

vol.events[s, p] <- vol.events.inst

meanTSL[s, p] <- meanTSL.inst

maxTSL[s, p] <- maxTSL.inst

m95TSL[s, p] <- m95TSL.inst

}

}

# Add desired usage metrics to wq.df

wq.df$num.events <- num.events$day14

wq.df$vol.events <- vol.events$day14

wq.df$meanTSL <- meanTSL$day14

wq.df$maxTSL <- maxTSL$day14

wq.df$m95TSL <- m95TSL$day14

# Remove unecessary info

rm(age.names, s, p, num.events.inst, vol.events.inst, meanTSL.inst, maxTSL.inst, use.data, num.events, vol.events, meanTSL, maxTSL, metrics)

#write.csv(wq.df, file = "wq.data.csv")

# Correlation Coefficients ------------------------------------------------

# Selected correlation matrix

my\_data <- as.matrix(select(wq.df, "pH", "Temp", "DO", "Total.Cl", "Free.Cl", "TOC", "DOC", "Alka", "TTHM", "TCC", "HPC", "Leg.sp","vol.events", "num.events", "meanTSL", "maxTSL"))

# Correlation matrix

library(Hmisc)

x <- rcorr(my\_data, type = "spearman")

#x <- cor(my\_data, method = "spearman", use = "complete.obs")

# I chose spearman over pearson because it is a non-parametric test

#setwd("C:/Users/Ryan/OneDrive - Michigan State University/Publications/ReNEWW Exploratory Data Analysis/Paper/Tables and Figures/")

#write.csv(x$r, file = "MatCor.csv")

#write.csv(x$n, file = "MatNum.csv")

#write.csv(x$P, file = "MatSig.csv")

# Select Data for PCA Analysis --------------------------------------------

# Uses all but data excluded during PCA (described in outline)

newdata.full <- na.omit(select(wq.df, Location, System, Date, Water.Age, pH, Temp, DO, Total.Cl, TCC, HPC, Leg.sp, TOC, Alka, TTHM, vol.events, maxTSL))

newdata.cat <- newdata.full[,1:4]

newdata <- newdata.full[,-(1:4)]

rm(newdata.full)

summary(newdata)

# Check whether to use PCA with Bartlett's Sphericty

# http://eric.univ-lyon2.fr/~ricco/tanagra/fichiers/en\_Tanagra\_KMO\_Bartlett.pdf

n <- nrow(newdata)

p <- ncol(newdata)

R <- cor(newdata)

chi2 <- -(n-1-(2\*p+5)/6)\*log(det(R))

ddl <- p\*(p-1)/2

pchisq(chi2,ddl,lower.tail=F)

# Sufficiently correlated to use PCA?

pchisq(chi2,ddl,lower.tail=F) < 0.05

# Confirm with KMO's MSA

library(psych)

KMO(newdata)

#MSA = 0.70; "middling"

# Yes. Proceed with PCA

# Conduct PCA -------------------------------------------------------------

pca <- prcomp(newdata, scale. = TRUE)

biplot(pca, scale = 0)

plot(pca, type = "line", main = "PCA Scree Plot")

abline(a = 1, b = 0, col = "red", lwd = 4)

summary(pca)

pca$rotation

# Print to file

setwd("C:/Users/Ryan/OneDrive - Michigan State University/Publications/ReNEWW Exploratory Data Analysis/Paper/Tables and Figures/")

write.csv(pca$rotation, file = "PCA.csv")

# Analysis with CSTAT -----------------------------------------------------

# 15 predictor variables initially available (plus legionella)

ncol(select(wq.df, Leg.sp, pH, Temp, DO, Total.Cl, TCC, HPC, TOC, Alka, TTHM, num.events, vol.events, meanTSL, maxTSL, Free.Cl, DOC))

# Missing 148 values (out of possible 406) for Legionella. No more than 258 observations can be include in GLM

sum(is.na(wq.df$Leg.sp))

# Decided to use remove Free.Cl and DOC before analysis because they were missing so often and had acceptable surrogates (Total.Cl and TOC)

sum(is.na(wq.df$Free.Cl)) # 147 missing

sum(is.na(wq.df$DOC)) # 35 missing

# Removing these Free.Cl and DOC allows for the inclusion of 89 additional observations (133 to 222)

nrow(na.omit(select(wq.df, Leg.sp, pH, Temp, DO, Total.Cl, TCC, HPC, TOC, Alka, TTHM, num.events, vol.events, meanTSL, maxTSL))) - nrow(na.omit(select(wq.df, Leg.sp, pH, Temp, DO, Total.Cl, TCC, HPC, TOC, Alka, TTHM, num.events, vol.events, meanTSL, maxTSL, Free.Cl, DOC)))

# Chose the following variables for analysis:

# Leg.sp, pH, Temp, DO, Total.Cl, TCC, HPC, TOC, Alka, TTHM, num.events, vol.events, meanTSL, maxTSL

detach("package:MASS", unload=TRUE)

library(dplyr)

data <- select(wq.df, date\_time, System, Location, Leg.sp, pH, Temp, DO, Total.Cl, TCC, HPC, TOC, Alka, TTHM, num.events, vol.events, meanTSL, maxTSL) # Selected all except Free.Cl and DOC

# Log transformations

# These are based on lowest Shapiro-Wilks W-statistic

data$Total.Cl <- log10(data$Total.Cl)

data$TOC <- log10(data$TOC)

data$Alka <- log10(data$Alka)

data$HPC <- log10(data$HPC)

data$TCC <- log10(data$TCC)

data$Leg.sp <- log10(data$Leg.sp)

# Did NOT log-transform use metrics

#data$num.events <- log10(data$num.events)

#data$vol.events <- log10(data$vol.events)

#data$meanTSL <- log10(data$meanTSL)

#data$maxTSL <- log10(data$maxTSL)

# Remove categorical variables

dat <- na.omit(data[,-(1:3)])

# Define GLM

global.glm <- glm(Leg.sp~.,dat, family = "gaussian", na.action = "na.fail")

# Results

plot(global.glm$y, global.glm$fitted.values)

cor(global.glm$y, global.glm$fitted.values)

summary(global.glm)

# Dredge GLM for best-fitting variable combinations

#install.packages("MuMIn")

library(MuMIn)

global.glm.dedge <- dredge(global.glm, rank = "AICc")

# Results

global.glm.dedge[1:11] # top 11 models have dAIC < 2.0

# Top: Leg.sp ~ HPC + num.events + TCC + Total.Cl

# All competing: Leg.sp ~ HPC + num.events + TCC + Total.Cl + DO + maxTSL + meanTSL + pH + Temp + TOC + TTHM + vol.events

#plot(global.glm, which = 1:6)

# Select new dataframe with only variables found in top competing models per 'dredge()'

dat.new <- select(data, c(Leg.sp, DO, HPC, maxTSL, meanTSL, num.events, pH, TCC, Temp, TOC, Total.Cl, TTHM, vol.events, Location))

dat.new <- na.omit(dat.new) # remove NAs

dat.loc <- dat.new["Location"]

dat.new <- subset(dat.new, select = -Location )

# Result has 235 observations of 13 variables

# Fit a GLM to the new data set

global.glm.new <- glm(Leg.sp~.,dat.new, family = "gaussian", na.action = "na.fail")

global.glm.dredge.new <- dredge(global.glm.new)

global.glm.dredge.new[1:4] # top 4 models have dAIC < 2.0

# Top ranked model (best): DO, HPC, maxTSL, meanTSL, num.events, pH, TCC, TOC, and Total.Cl

# Competing models (comp): DO, HPC, maxTSL, meanTSL, num.events, pH, TCC, TOC, Total.Cl, and TTHM

# Mixed model with random effects for fixture locations -------------------

# Add "Location" back into data

dat.new <- cbind(dat.new, dat.loc)

#install.packages("glmmTMB")

library(glmmTMB)

# Top-Ranked Model --------------------------------------------------------

global.glm.me.best <- glmmTMB(Leg.sp ~ DO + HPC + maxTSL + meanTSL + num.events + pH + TCC + TOC + Total.Cl + (1|Location),

data = dat.new, family = "gaussian", na.action = "na.fail")

# Check for convergence

global.glm.me.best$sdr$pdHess

# Check for singularity

library(performance)

performance::check\_singularity(global.glm.me.best)

# Residuals

summary(global.glm.me.best)

global.glm.me.best.resid <- resid(global.glm.me.best, "pearson")

plot(global.glm.me.best.resid)

hist(global.glm.me.best.resid)

# Q-Q residuals plot

qqnorm(global.glm.me.best.resid, pch = 1, frame = FALSE)

qqline(global.glm.me.best.resid, col = "steelblue", lwd = 2)

# Check homoscedasticity

performance::check\_heteroscedasticity(global.glm.me.best)

# Competing Models --------------------------------------------------------

global.glm.me.comp <- glmmTMB(Leg.sp ~ DO + HPC + maxTSL + meanTSL + num.events + pH + TCC + TOC + Total.Cl + TTHM + (1|Location),

data = dat.new, family = "gaussian", na.action = "na.fail")

# Check for convergence

global.glm.me.comp$sdr$pdHess

# Check for singularity

performance::check\_singularity(global.glm.me.comp)

# Residuals

summary(global.glm.me.comp)

global.glm.me.comp.resid <- resid(global.glm.me.comp, "pearson")

plot(global.glm.me.comp.resid)

hist(global.glm.me.comp.resid)

# Q-Q residuals plot

qqnorm(global.glm.me.comp.resid, pch = 1, frame = FALSE)

qqline(global.glm.me.comp.resid, col = "steelblue", lwd = 2)

# Check homoscedasticity

performance::check\_heteroscedasticity(global.glm.me.comp)

# Final Model Comparison --------------------------------------------------

#install.packages("DHARMa")

library(DHARMa)

sim <- simulateResiduals(global.glm.me.best, n = 10000)

plot(sim)

testOutliers(sim)

# Single outlier identified in top model

sim <- simulateResiduals(global.glm.me.comp, n = 10000)

plot(sim)

testOutliers(sim)

# Single outlier identified in competing models

# Check for Multicollinearity

check\_collinearity(global.glm.me.best, component = "all")

check\_collinearity(global.glm.me.comp, component = "all")

# Report model performance

model\_performance(global.glm.me.best, metrics = "all")

model\_performance(global.glm.me.comp, metrics = "all")

# Both models perform similarly (intuitive as only difference is TTHM)

# All 6 metrics suggest 'best' is indeed best

# Final Model Selection ---------------------------------------------------

# Select top-ranked model: Leg.sp ~ DO + HPC + maxTSL + meanTSL + num.events + pH + TCC + TOC + Total.Cl + (1|Location)

summary(global.glm.me.best)

# old version: Leg.sp ~ HPC + meanTSL + num.events + TCC + Total.Cl + Location

# Adding pH + TTHM maintains integrity of GLMM, but does not offer additional predictive power

plot(x = dat.new$Leg.sp, y = predict(global.glm.me.best, dat.new),

main ="Top GLMM Legionella Results",

xlab = "Measured Legionella spp. (Log GC/100 mL)", ylab = "Predicted Legionella spp. (Log GC/100 mL)")

abline(a = 0, b = 1, col = "red")

cor(x = dat.new$Leg.sp, y = predict(global.glm.me.best, dat.new))

plot(x = dat.new$Leg.sp, y = predict(global.glm.me.comp, dat.new),

main ="Competing Model Legionella Results",

xlab = "Measured Legionella spp. (Log GC/100 mL)", ylab = "Predicted Legionella spp. (Log GC/100 mL)")

abline(a = 0, b = 1, col = "red")

cor(x = dat.new$Leg.sp, y = predict(global.glm.me.comp, dat.new))

# And compare to previous GLM w/o location

plot(x = dat.new$Leg.sp, y = predict(global.glm.new, dat.new),

main ="GLM Legionella Results",

xlab = "Measured Legionella spp. (Log GC/100 mL)", ylab = "Predicted Legionella spp. (Log GC/100 mL)")

abline(a = 0, b = 1, col = "red")

cor(x = dat.new$Leg.sp ,y = predict(global.glm.new, dat.new))

# Water age model code

Water age modeling was conducted in RStudio using several scripts. First, flowmeter data, stored in several NetCDF files was imported to R using the script **reneww\_waterage\_ncin\_importer\_20200810.R**, to create a single data frame with all flowmeter data. That resulting data frame, along with a comma separated variable (CSV) file which described the plumbing system, were processed by the script **rennew\_waterage\_processor\_20201019.R** to develop discretized cumulative instantaneous flow descriptions for each monitored fixture. Those results were then passed to the script **reneww\_waterage\_20201019.R** to determine water age for each parcel at time of use. Plots of results were produced with the script **waterage\_output\_plots\_20201026.R**. Finally, water age was attributed to each sample collected from the ReNEWW house using the script **associate\_waterage\_with\_sample.R**. Variable selection including water age metrics was performed using the script **waterage\_statistical\_analysis\_20210316.R**. Comparisons with EPANET were prepared in a similar fashion. The same script was used to import NetCDF water use data. However, a different script (**reneww\_waterage\_processor\_20200923\_EPANET\_comparison.R**) was used to prepare this data for water age modeling, taking into account the simplifications made of the PPS. Water age was then determined for each parcel of water using the same water age script (**reneww\_waterage\_20201019.R**). Each of these scripts is provided under a separate heading, below.

## reneww\_waterage\_ncin\_importer\_20200810.R

#Load Required Libraries

library(ncdf4)

library(ggplot2)

library(ggthemes)

library(dplyr)

library(lubridate)

n <- NULL

mo <- c("01", "02", "03", "04", "05", "06", "07", "08", "09", "10", "11", "12")

yr <- c("2015", "2016", "2017", "2018", "2019")

for (y in yr){

for (m in mo){

n <- c(n, paste(y, m, sep = "\_"))

}

}

n <- n[8:52]

ncpath <- "D:/PhD/ReNEWW/Data/Electronic/Monthly/nc/"

dat.use <- NULL

dat.use.check <- NULL

for (ncname in n){

# Open the netCDF file

ncfname <- paste(ncpath, ncname, ".nc", sep="")

ncin <- nc\_open(ncfname)

#print(ncin)

#Converts separated date information into "datetime" timestamp

timestamp <- as\_datetime(paste(paste(ncvar\_get(ncin,"year"), ncvar\_get(ncin,"month"), ncvar\_get(ncin,"day"), sep = "/"), paste(ncvar\_get(ncin,"hour"), ncvar\_get(ncin,"minute"), ncvar\_get(ncin,"second"), sep = ":"), sep = " "))

# Add each flowmeter

SL <- ncvar\_get(ncin, "City\_Main\_vol")

RW <- ncvar\_get(ncin, "Rainwater\_Main\_vol")

GW.makeup <- ncvar\_get(ncin, "PGW\_MakeUp\_Rain\_vol")

GW.supply <- ncvar\_get(ncin, "PGW\_Supply\_vol")

cold.bath1.shower <- ncvar\_get(ncin, "Bath1\_Shower\_Cold\_vol")

cold.bath2.shower <- ncvar\_get(ncin, "Bath2\_Shower\_Cold\_vol")

cold.kitchen.sink <- ncvar\_get(ncin, "Kitchen\_Cold\_vol")

hot.bath2.shower <- ncvar\_get(ncin, "Bath2\_Shower\_Hot\_vol")

hot.clothes <- ncvar\_get(ncin, "WashMach\_Hot\_vol")

cold.clothes <- ncvar\_get(ncin, "WashMach\_Cold\_vol")

hot.bath1.shower <- ncvar\_get(ncin, "Bath1\_Shower\_Hot\_vol")

hot.kitchen.sink <- ncvar\_get(ncin, "Kitchen\_Sink\_Hot\_vol")

cold.heater <- ncvar\_get(ncin, "DHW\_Cold\_Inlet\_vol")

hot.recycle <- ncvar\_get(ncin, "DHW\_Hot\_Return\_vol")

hot.dish <- ncvar\_get(ncin, "Dishwasher\_Hot\_vol")

cold.kitchen.isl <- ncvar\_get(ncin, "Island\_Cold\_vol")

hot.kitchen.isl <- ncvar\_get(ncin, "Island\_Hot\_vol")

cold.bath1.sink <- ncvar\_get(ncin, "Bath1\_Sink\_Cold\_vol")

hot.bath1.sink <- ncvar\_get(ncin, "Bath1\_Sink\_Hot\_vol")

cold.bath2.sink <- ncvar\_get(ncin, "Bath2\_Sink\_Cold\_vol")

hot.bath2.sink <- ncvar\_get(ncin, "Bath2\_Sink\_Hot\_vol")

# Close the netCDF file - You risk losing information from the file if you don't do this!

nc\_close(ncin)

rm(ncin)

# Create two seperate data frames - one that will be passed to calculate water age, and one that will be used to confirm total water use

data.df <- data.frame(cbind(timestamp, cold.bath1.shower, cold.bath2.shower, cold.kitchen.sink, hot.bath2.shower, hot.clothes, cold.clothes, hot.bath1.shower, hot.kitchen.sink, GW.makeup,cold.heater, hot.recycle, hot.dish, cold.kitchen.isl, hot.kitchen.isl, cold.bath1.sink, hot.bath1.sink, cold.bath2.sink, hot.bath2.sink))

check.df <- data.frame(cbind(timestamp, SL, RW, GW.makeup, GW.supply))

rm(timestamp, SL, cold.bath1.shower, cold.bath2.shower, cold.kitchen.sink, hot.bath2.shower, hot.clothes, cold.clothes, hot.bath1.shower, hot.kitchen.sink, RW, GW.makeup, GW.supply, cold.heater, hot.recycle, hot.dish, cold.kitchen.isl, hot.kitchen.isl, cold.bath1.sink, hot.bath1.sink, cold.bath2.sink, hot.bath2.sink)

data.df$timestamp <- as\_datetime(data.df$timestamp)

check.df$timestamp <- as\_datetime(check.df$timestamp)

#start.date <- as\_datetime("2017-10-02")

#end.date <- as\_datetime("2017-10-15")

#df <- subset(df, timestamp >= start.date & timestamp < end.date) #subset by date

data.df <- data.df[rowSums(data.df[,-1]) > 0,] # subset by flow at any time (shorten files)

check.df <- check.df[rowSums(check.df[,-1]) > 0,] # subset by flow at any time (shorten files)

dat.use <- rbind(dat.use, data.df)

dat.use.check <- rbind(dat.use.check, check.df)

#Write finished time period to file

#setwd("D:/PhD/Publications/AEESP/Model/data")

write.csv(data.df, file = paste("D:/PhD/ReNEWW/Data/Electronic/Monthly/csv/",

"data\_",ncname, ".csv", sep = ""))

write.csv(check.df, file = paste("D:/PhD/ReNEWW/Data/Electronic/Monthly/csv/",

"check\_",ncname, ".csv", sep = ""))

}

write.csv(dat.use, file = paste("D:/PhD/ReNEWW/Data/Electronic/Monthly/csv/",

"data\_all\_", ".csv", sep = ""))

write.csv(dat.use.check, file = paste("D:/PhD/ReNEWW/Data/Electronic/Monthly/csv/",

"data\_check\_all\_", ".csv", sep = ""))

use.data.inst <- dat.use

rm(mo, yr, m, y, n, ncfname, ncname, ncpath, data.df, check.df, dat.use, dat.use.check)

save.image("C:/Users/Ryan/OneDrive - Michigan State University/Publications/Water Age/Data/use\_data\_201508\_201904.RData")

## rennew\_waterage\_processor\_20201019.R

# Set directory

getwd()

setwd("/mnt/ufs18/home-037/julienry/Desktop/Water Age/")

#setwd("/home/ryan/Everything/PhD/Water Age Paper/Model/")

timer.start <- Sys.time()

# Import Pipe Network

library(dplyr)

# Prep --------------------------------------------------------------------

# Choose 'delta.v'

delta.v <- 5 # mL # Unit volume for simulation. Explain this in more detail later

pipe.segments <- as.data.frame(read.csv("pipe\_segments\_rev.1\_20200615.csv", stringsAsFactors = TRUE))

pipe.segments$names <- paste(pipe.segments$upstream, pipe.segments$downstream, sep = "-")

pipe.segments[,c("L\_d0.75", "L\_d0.5")] <- pipe.segments[,c("L\_d0.75", "L\_d0.5")] \* 2.54 # (convert pipe length to cm)

# Set inner diameter for 3/4" and 1/2" pipe (assume PEX-A)

id\_0.75 <- 0.681 \* 2.54 # cm

id\_0.5 <- 0.485 \* 2.54 # cm

pipe.segments$vol <- round((pipe.segments$L\_d0.75 \* pi \* ((id\_0.75)/2)^2 +

pipe.segments$L\_d0.5 \* pi \* ((id\_0.5)/2)^2)

/ delta.v, 0) # convert to same units as 'delta.v'

# Add volume for WH tanks

#T1-T3 each at 409 L

pipe.segments$vol[pipe.segments$names == "T1-T3"] <-

as.integer(pipe.segments$vol[pipe.segments$names == "T1-T3"] + 3 \* 409 \* 1000 /delta.v)

#T4 at 141 L

pipe.segments$vol[pipe.segments$names == "WH-T4"] <-

as.integer(pipe.segments$vol[pipe.segments$names == "WH-T4"] + 141 \* 1000 /delta.v)

# Define Nodes

nodes <- unique(c(levels(pipe.segments$upstream), levels(pipe.segments$downstream)))

# Put nodes in order

nodes.comp <- nodes[nodes %in% c("SL")]

nodes.inc <- nodes[!nodes %in% nodes.comp]

nodes.inc.num <- length(nodes.inc)

# n <- nodes.inc[1]

while(length(nodes.inc) > 0){

for (n in nodes.inc) {

if (all(pipe.segments$upstream[pipe.segments$downstream %in% n][pipe.segments$upstream[pipe.segments$downstream %in% n] %in% nodes] %in% nodes.comp)){

nodes.comp <- c(nodes.comp, n)

nodes.inc <- nodes.inc[nodes.inc != n]

}

}

if (length(nodes.inc) == nodes.inc.num &

"hot.recycle" %in% nodes.inc){

nodes.comp <- c(nodes.comp, "hot.recycle")

nodes.inc <- nodes.inc[nodes.inc != "hot.recycle"]

}

nodes.inc.num <- length(nodes.inc)

}

if (all(nodes.comp %in% nodes, nodes %in% nodes.comp)){

nodes <- nodes.comp

rm(nodes.comp, nodes.inc)

} else{

stop("Error sorting nodes")

}

# Import Use Data ---------------------------------------------------------

#load("/home/ryan/Everything/PhD/Water Age Paper/Model/use\_data\_201508\_201904.RData")

load("/mnt/ufs18/home-037/julienry/Desktop/Water Age/use\_data\_201508\_201904.RData")

# Only for testing

#use.data.inst <- use.data.inst[1:100000,]

# Replace Faulty Sink Flowmeters ------------------------------------------

# Assume that ratio of hot/cold is the same for all kitchen sinks

r.ks <- sum(use.data.inst$hot.kitchen.sink) / sum(use.data.inst$cold.kitchen.sink)

# Thus the (faulty) hot.kitchen.isl may be replaced with a multiple of cold.kitchen.isl

use.data.inst$hot.kitchen.isl <- use.data.inst$cold.kitchen.isl \* r.ks

# The same can be done to replace hot.bath2.sink

r.bs <- sum(use.data.inst$hot.bath2.sink) / sum(use.data.inst$cold.bath2.sink)

use.data.inst$hot.bath2.sink <- use.data.inst$cold.bath2.sink \* r.bs

rm(r.ks, r.bs)

# Prepare data to develop cumulative flow ---------------------------------

library(lubridate)

use.data.inst$timestamp <- ymd\_hms(use.data.inst$timestamp, tz = Sys.timezone())

# 't' is time (seconds), columns are fixture names, rows are flow (in units of delta.v) at 1 sec resolution

use.data.inst <- as.data.frame(use.data.inst)

use.data.inst[,-1] <- use.data.inst[,-1] \* 3785.4 / delta.v # Convert flowrates from gallons to units of delta.v \* mL

# Remove all rows with no flow now that it's been rounded.

use.data.inst <- use.data.inst[rowSums(use.data.inst[,-(1:2)]) != 0,]

# Calculate cumulative flow for know nodes

use.data.cum <- as.data.frame(matrix(NA, nrow = nrow(use.data.inst), ncol = ncol(use.data.inst), dimnames = dimnames(use.data.inst)))

use.data.cum$timestamp <- use.data.inst$timestamp

use.data.cum[,-1] <- round(cumsum(use.data.inst[,-1]), 0)

# Create cumulative flow table

cum.flow <- as.data.frame(matrix(NA, nrow = nrow(use.data.cum), ncol = length(nodes), dimnames = list(NULL, nodes)))

# Add flow from use.data.cum to cum.flow

for (f in 2:ncol(use.data.cum)){

if (colnames(use.data.cum[f]) %in% colnames(cum.flow)){

cum.flow[,colnames(use.data.cum[f])] <- use.data.cum[f]

}

}

rm(f)

# Manually fill 'cum.flow.table' ------------------------------------------

# I'd like to change this in future versions to automatically calculate based on pipe network in 'pipe.segments'

library(data.table)

library(dplyr)

# Ensure demand at 'cold.heater' never exceeds hot water use

all.hot <- rowSums(select(cum.flow, hot.bath1.sink, hot.bath1.shower, hot.kitchen.isl, hot.bath2.shower, hot.bath2.sink, hot.clothes, hot.kitchen.sink, hot.dish))

cum.flow$cold.heater <- cumsum(apply(cbind(

c(cum.flow$cold.heater[1], diff(cum.flow$cold.heater)),

c(all.hot[1], diff(all.hot))), 1, FUN = min)

)

# Develop cumulative flow for all upstream nodes

cum.flow.matrix <- as.data.frame(cum.flow)

# Assume no flow at each unmetered fixture for now

cum.flow.matrix$cold.base.sink <- 0

cum.flow.matrix$cold.biowall <- 0

cum.flow.matrix$cold.fridge <- 0

cum.flow.matrix$cold.hose <- 0

cum.flow.matrix$hot.base.sink <- 0

# Assume no incoming rainwater flow

cum.flow.matrix$RW <- 0

# Fill each node where all downstream flow is known

cum.flow.matrix$H.3.3 <- cum.flow.matrix$hot.dish + cum.flow.matrix$hot.base.sink

cum.flow.matrix$H.3.2 <- cum.flow.matrix$H.3.3 + cum.flow.matrix$hot.kitchen.sink

cum.flow.matrix$H.3.1 <- cum.flow.matrix$H.3.2 + cum.flow.matrix$hot.clothes

cum.flow.matrix$H.6 <- cum.flow.matrix$hot.recycle + cum.flow.matrix$hot.bath2.sink

cum.flow.matrix$H.5 <- cum.flow.matrix$H.6 + cum.flow.matrix$hot.bath2.shower

cum.flow.matrix$H.4 <- cum.flow.matrix$H.5 + cum.flow.matrix$hot.kitchen.isl

cum.flow.matrix$H.3 <- cum.flow.matrix$H.4 + cum.flow.matrix$H.3.1

cum.flow.matrix$H.2 <- cum.flow.matrix$H.3 + cum.flow.matrix$hot.bath1.shower

cum.flow.matrix$H.1 <- cum.flow.matrix$H.2 + cum.flow.matrix$hot.bath1.sink

cum.flow.matrix$H <- cum.flow.matrix$H.1

#all.hot <- cum.flow.matrix$H - cum.flow.matrix$hot.recycle

cum.flow.matrix$C.07.1 <- all.hot

# Fixes for 'cold.heater'

# This is used to ensure that cold.heater observes no more flow than 'all.hot' at any given moment. This assumes there is no backward flow, and that 'cold.heater' is wrong if it does observe flow then.

#for (t in 2:nrow(cum.flow.matrix)){

# if ((cum.flow.matrix$cold.heater[t] - cum.flow.matrix$cold.heater[t-1]) > (cum.flow.matrix$C.07.1[t] - cum.flow.matrix$C.07.1[t-1])){

# cum.flow.matrix$cold.heater[t] <- cum.flow.matrix$cold.heater[t-1] + (cum.flow.matrix$C.07.1[t] - cum.flow.matrix$C.07.1[t-1])

# }

#

#}

cum.flow.matrix$C.07.2 <- cum.flow.matrix$C.07.1 - cum.flow.matrix$cold.heater

cum.flow.matrix$H.0 <- cum.flow.matrix$H - cum.flow.matrix$cold.heater

cum.flow.matrix$H.00 <- cum.flow.matrix$H.0

# Assume all nodes in WH network have the same flow as 'cold.heater'

cum.flow.matrix$T1 <- cum.flow.matrix$cold.heater

cum.flow.matrix$T3 <- cum.flow.matrix$cold.heater

cum.flow.matrix$WH <- cum.flow.matrix$cold.heater

cum.flow.matrix$T4 <- cum.flow.matrix$cold.heater

# Fill in cold water system

cum.flow.matrix$C.10 <- cum.flow.matrix$cold.biowall + cum.flow.matrix$cold.hose

cum.flow.matrix$C.09 <- cum.flow.matrix$C.10 + cum.flow.matrix$cold.bath2.sink

cum.flow.matrix$C.08 <- cum.flow.matrix$C.09 + cum.flow.matrix$cold.bath2.shower

cum.flow.matrix$C.07 <- cum.flow.matrix$C.08 + cum.flow.matrix$C.07.1

cum.flow.matrix$C.06.1 <- cum.flow.matrix$cold.bath1.shower + cum.flow.matrix$cold.bath1.sink

cum.flow.matrix$C.06 <- cum.flow.matrix$C.06.1 + cum.flow.matrix$C.07

cum.flow.matrix$C.05 <- cum.flow.matrix$C.06 + cum.flow.matrix$cold.kitchen.isl

cum.flow.matrix$C.04 <- cum.flow.matrix$C.05 + cum.flow.matrix$cold.fridge

cum.flow.matrix$C.03 <- cum.flow.matrix$C.04 + cum.flow.matrix$cold.kitchen.sink

cum.flow.matrix$C.02 <- cum.flow.matrix$C.03 + cum.flow.matrix$cold.clothes

cum.flow.matrix$C.01 <- cum.flow.matrix$C.02 + cum.flow.matrix$cold.base.sink

# Assume flow from SL and RW are mixed before coming into the house (i.e. they are indistinguishable)

cum.flow.matrix$C.00 <- cum.flow.matrix$C.01

cum.flow.matrix$RW.1 <- cum.flow.matrix$C.00

cum.flow.matrix$SL.1 <- cum.flow.matrix$RW.1 # - cum.flow.matrix$RW

cum.flow.matrix$SL.0 <- cum.flow.matrix$SL.1 + cum.flow.matrix$GW.makeup

cum.flow.matrix$WS <- cum.flow.matrix$SL.0

cum.flow.matrix$SL <- cum.flow.matrix$WS

cum.flow.matrix$B <- cum.flow.matrix$cold.bath2.shower + cum.flow.matrix$hot.bath2.shower

cum.flow.matrix$mix.bath2.shower <- cum.flow.matrix$B

time.sim.start <- (use.data.cum$timestamp[1] - 1)

time.index <- use.data.cum$timestamp - time.sim.start

# Remember: Subtracted 1 here to get time index to start at 1 instead of 0

# Thus, 'time.index' is relative to a second \*before\* the actual time

# Clean up the environment

rm(list=setdiff(ls(), c("cum.flow.matrix", "pipe.segments", "nodes", "time.index", "delta.v", "delta.t", "time.sim.start", "timer", "timer.start")))

save.image("processor\_output\_20201019.RData")

timer <- Sys.time() - timer.start

timer

## reneww\_waterage\_processor\_20200923\_EPANET\_comparison.R

# Set directory

getwd()

#setwd("/mnt/ufs18/home-037/julienry/Desktop/Water Age/")

#setwd("/home/ryan/Everything/PhD/Water Age Paper/Model/")

setwd("C:/Users/Ryan/OneDrive - Michigan State University/Publications/Water Age/Model")

timer.start <- Sys.time()

# Import Pipe Network

library(dplyr)

# Prep --------------------------------------------------------------------

# Choose 'delta.v'

delta.v <- 5 # mL # Unit volume for simulation. Explain this in more detail later

pipe.segments <- as.data.frame(read.csv("pipe\_segments\_rev.1\_20200615.csv", stringsAsFactors = TRUE))

pipe.segments$names <- paste(pipe.segments$upstream, pipe.segments$downstream, sep = "-")

pipe.segments[,c("L\_d0.75", "L\_d0.5")] <- pipe.segments[,c("L\_d0.75", "L\_d0.5")] \* 2.54 # (convert pipe length to cm)

# Set inner diameter for 3/4" and 1/2" pipe (assume PEX-A)

id\_0.75 <- 0.681 \* 2.54 # cm

id\_0.5 <- 0.485 \* 2.54 # cm

pipe.segments$vol <- round((pipe.segments$L\_d0.75 \* pi \* ((id\_0.75)/2)^2 +

pipe.segments$L\_d0.5 \* pi \* ((id\_0.5)/2)^2)

/ delta.v, 0) # convert to same units as 'delta.v'

# Add volume for WH tanks

#T1-T3 each at 409 L

pipe.segments$vol[pipe.segments$names == "T1-T3"] <-

as.integer(pipe.segments$vol[pipe.segments$names == "T1-T3"] + 3 \* 409 \* 1000 /delta.v)

#T4 at 141 L

pipe.segments$vol[pipe.segments$names == "WH-T4"] <-

as.integer(pipe.segments$vol[pipe.segments$names == "WH-T4"] + 141 \* 1000 /delta.v)

# Define Nodes

nodes <- unique(c(levels(pipe.segments$upstream), levels(pipe.segments$downstream)))

# Put nodes in order

nodes.comp <- nodes[nodes %in% c("SL")]

nodes.inc <- nodes[!nodes %in% nodes.comp]

nodes.inc.num <- length(nodes.inc)

# n <- nodes.inc[1]

while(length(nodes.inc) > 0){

for (n in nodes.inc) {

if (all(pipe.segments$upstream[pipe.segments$downstream %in% n][pipe.segments$upstream[pipe.segments$downstream %in% n] %in% nodes] %in% nodes.comp)){

nodes.comp <- c(nodes.comp, n)

nodes.inc <- nodes.inc[nodes.inc != n]

}

}

if (length(nodes.inc) == nodes.inc.num &

"hot.recycle" %in% nodes.inc){

nodes.comp <- c(nodes.comp, "hot.recycle")

nodes.inc <- nodes.inc[nodes.inc != "hot.recycle"]

}

nodes.inc.num <- length(nodes.inc)

}

if (all(nodes.comp %in% nodes, nodes %in% nodes.comp)){

nodes <- nodes.comp

rm(nodes.comp, nodes.inc)

} else{

stop("Error sorting nodes")

}

# Import Use Data ---------------------------------------------------------

#load("/home/ryan/Everything/PhD/Water Age Paper/Model/use\_data\_201508\_201904.RData")

load("use\_data\_201508\_201904.RData")

# Only for testing

#use.data.inst <- use.data.inst[1:100000,]

library(lubridate)

date.start <- ymd("2017-09-01")

date.end <- ymd("2018-11-01")

use.data.inst <- subset(use.data.inst, timestamp > date.start & timestamp < date.end)

library(lubridate)

use.data.inst$timestamp <- ymd\_hms(use.data.inst$timestamp, tz = Sys.timezone())

# 't' is time (seconds), columns are fixture names, rows are flow (in units of delta.v) at 1 sec resolution

use.data.inst <- as.data.frame(use.data.inst)

use.data.inst[,-1] <- use.data.inst[,-1] \* 3785.4 / delta.v # Convert flowrates from gallons to units of delta.v \* mL

# Remove all rows with no flow now that it's been rounded.

use.data.inst <- use.data.inst[rowSums(use.data.inst[,-(1:2)]) != 0,]

# Calculate cumulative flow for know nodes

use.data.cum <- as.data.frame(matrix(NA, nrow = nrow(use.data.inst), ncol = ncol(use.data.inst), dimnames = dimnames(use.data.inst)))

use.data.cum$timestamp <- use.data.inst$timestamp

use.data.cum[,-1] <- round(cumsum(use.data.inst[,-1]), 0)

# Create cumulative flow table

cum.flow <- as.data.frame(matrix(NA, nrow = nrow(use.data.cum), ncol = length(nodes), dimnames = list(NULL, nodes)))

# Add flow from use.data.cum to cum.flow

for (f in 2:ncol(use.data.cum)){

if (colnames(use.data.cum[f]) %in% colnames(cum.flow)){

cum.flow[,colnames(use.data.cum[f])] <- use.data.cum[f]

}

}

rm(f)

# Manually fill 'cum.flow.table' ------------------------------------------

# I'd like to change this in future versions to automatically calculate based on pipe network in 'pipe.segments'

library(data.table)

library(dplyr)

# Ensure demand at 'cold.heater' never exceeds hot water use

all.hot <- rowSums(select(cum.flow, hot.bath1.sink, hot.bath1.shower, hot.kitchen.isl, hot.bath2.shower, hot.bath2.sink, hot.clothes, hot.kitchen.sink, hot.dish))

cum.flow$cold.heater <- cumsum(apply(cbind(

c(cum.flow$cold.heater[1], diff(cum.flow$cold.heater)),

c(all.hot[1], diff(all.hot))), 1, FUN = min)

)

# Develop cumulative flow for all upstream nodes

cum.flow.matrix <- as.data.frame(cum.flow)

# Assume no flow at each unmetered fixture for now

cum.flow.matrix$cold.base.sink <- 0

cum.flow.matrix$cold.biowall <- 0

cum.flow.matrix$cold.fridge <- 0

cum.flow.matrix$cold.hose <- 0

cum.flow.matrix$hot.base.sink <- 0

# Zero out all meters except those used for the EPANET comparison

cum.flow.matrix$GW.makeup <- 0

cum.flow.matrix$cold.bath1.shower <- 0

cum.flow.matrix$cold.bath1.sink <- 0

cum.flow.matrix$cold.kitchen.isl <- 0

cum.flow.matrix$cold.clothes <- 0

# Assume no incoming rainwater flow

cum.flow.matrix$RW <- 0

# Fill each node where all downstream flow is known

cum.flow.matrix$H.3.3 <- cum.flow.matrix$hot.dish + cum.flow.matrix$hot.base.sink

cum.flow.matrix$H.3.2 <- cum.flow.matrix$H.3.3 + cum.flow.matrix$hot.kitchen.sink

cum.flow.matrix$H.3.1 <- cum.flow.matrix$H.3.2 + cum.flow.matrix$hot.clothes

cum.flow.matrix$H.6 <- cum.flow.matrix$hot.recycle + cum.flow.matrix$hot.bath2.sink

cum.flow.matrix$H.5 <- cum.flow.matrix$H.6 + cum.flow.matrix$hot.bath2.shower

cum.flow.matrix$H.4 <- cum.flow.matrix$H.5 + cum.flow.matrix$hot.kitchen.isl

cum.flow.matrix$H.3 <- cum.flow.matrix$H.4 + cum.flow.matrix$H.3.1

cum.flow.matrix$H.2 <- cum.flow.matrix$H.3 + cum.flow.matrix$hot.bath1.shower

cum.flow.matrix$H.1 <- cum.flow.matrix$H.2 + cum.flow.matrix$hot.bath1.sink

cum.flow.matrix$H <- cum.flow.matrix$H.1

#all.hot <- cum.flow.matrix$H - cum.flow.matrix$hot.recycle

cum.flow.matrix$C.07.1 <- all.hot

# Fixes for 'cold.heater'

# This is used to ensure that cold.heater observes no more flow than 'all.hot' at any given moment. This assumes there is no backward flow, and that 'cold.heater' is wrong if it does observe flow then.

#for (t in 2:nrow(cum.flow.matrix)){

# if ((cum.flow.matrix$cold.heater[t] - cum.flow.matrix$cold.heater[t-1]) > (cum.flow.matrix$C.07.1[t] - cum.flow.matrix$C.07.1[t-1])){

# cum.flow.matrix$cold.heater[t] <- cum.flow.matrix$cold.heater[t-1] + (cum.flow.matrix$C.07.1[t] - cum.flow.matrix$C.07.1[t-1])

# }

#

#}

cum.flow.matrix$C.07.2 <- cum.flow.matrix$C.07.1 - cum.flow.matrix$cold.heater

cum.flow.matrix$H.0 <- cum.flow.matrix$H - cum.flow.matrix$cold.heater

cum.flow.matrix$H.00 <- cum.flow.matrix$H.0

# Assume all nodes in WH network have the same flow as 'cold.heater'

cum.flow.matrix$T1 <- cum.flow.matrix$cold.heater

cum.flow.matrix$T3 <- cum.flow.matrix$cold.heater

cum.flow.matrix$WH <- cum.flow.matrix$cold.heater

cum.flow.matrix$T4 <- cum.flow.matrix$cold.heater

# Fill in cold water system

cum.flow.matrix$C.10 <- cum.flow.matrix$cold.biowall + cum.flow.matrix$cold.hose

cum.flow.matrix$C.09 <- cum.flow.matrix$C.10 + cum.flow.matrix$cold.bath2.sink

cum.flow.matrix$C.08 <- cum.flow.matrix$C.09 + cum.flow.matrix$cold.bath2.shower

cum.flow.matrix$C.07 <- cum.flow.matrix$C.08 + cum.flow.matrix$C.07.1

cum.flow.matrix$C.06.1 <- cum.flow.matrix$cold.bath1.shower + cum.flow.matrix$cold.bath1.sink

cum.flow.matrix$C.06 <- cum.flow.matrix$C.06.1 + cum.flow.matrix$C.07

cum.flow.matrix$C.05 <- cum.flow.matrix$C.06 + cum.flow.matrix$cold.kitchen.isl

cum.flow.matrix$C.04 <- cum.flow.matrix$C.05 + cum.flow.matrix$cold.fridge

cum.flow.matrix$C.03 <- cum.flow.matrix$C.04 + cum.flow.matrix$cold.kitchen.sink

cum.flow.matrix$C.02 <- cum.flow.matrix$C.03 + cum.flow.matrix$cold.clothes

cum.flow.matrix$C.01 <- cum.flow.matrix$C.02 + cum.flow.matrix$cold.base.sink

# Assume flow from SL and RW are mixed before coming into the house (i.e. they are indistinguishable)

cum.flow.matrix$C.00 <- cum.flow.matrix$C.01

cum.flow.matrix$RW.1 <- cum.flow.matrix$C.00

cum.flow.matrix$SL.1 <- cum.flow.matrix$RW.1 # - cum.flow.matrix$RW

cum.flow.matrix$SL.0 <- cum.flow.matrix$SL.1 + cum.flow.matrix$GW.makeup

cum.flow.matrix$WS <- cum.flow.matrix$SL.0

cum.flow.matrix$SL <- cum.flow.matrix$WS

cum.flow.matrix$B <- cum.flow.matrix$cold.bath2.shower + cum.flow.matrix$hot.bath2.shower

cum.flow.matrix$mix.bath2.shower <- cum.flow.matrix$B

time.sim.start <- (use.data.cum$timestamp[1] - 1)

time.index <- use.data.cum$timestamp - time.sim.start

# Remember: Subtracted 1 here to get time index to start at 1 instead of 0

# Thus, 'time.index' is relative to a second \*before\* the actual time

#

# Clean up the environment

rm(list=setdiff(ls(), c("cum.flow.matrix", "pipe.segments", "nodes", "time.index", "delta.v", "delta.t", "time.sim.start", "timer", "timer.start")))

save.image("processor\_output\_EPANET\_cold.RData")

timer <- Sys.time() - timer.start

timer

## reneww\_waterage\_20201019.R

# MSU HPCC

setwd("/mnt/ufs18/home-037/julienry/Desktop/Water Age/")

# Linux Desktop

#setwd("/home/ryan/Everything/PhD/Water Age Paper/Model/")

# Windows Laptop

#setwd("C:/Users/Ryan/OneDrive - Michigan State University/Publications/Water Age/Model")

load("processor\_output\_20201019.RData")

# Start Timer

timer.start <- Sys.time()

library(dplyr)

flow.nodes.cum <- cum.flow.matrix

rm(cum.flow.matrix)

# Determine mean HRT for WH based on input data

# -Assume WH is filled with water with age equal to the mean HRT

WH.avg.flow <- # Units are delta.v per second

sum(select(flow.nodes.cum[nrow(flow.nodes.cum),], hot.bath1.sink, hot.bath1.shower, hot.kitchen.isl, hot.bath2.shower, hot.bath2.sink, hot.clothes, hot.kitchen.sink, hot.dish)) /

as.numeric(diff(time.index[c(1, nrow(flow.nodes.cum))]))

WH.vol <- as.integer((3 \* 409 + 141) \* 1000 / delta.v) # WH volume in units of delta.v

WH.HRT <- round(WH.vol/WH.avg.flow, 0) # Assumed WH age in seconds

# Select subset of use data -----------------------------------------------

per <- 1

s <- c(1, 1500000, 3000000)

e <- c(1500001, 3000001, nrow(flow.nodes.cum))

s <- s[per]

e <- e[per]

if (per == 1){

flow.init <- matrix(0, nrow = 1, ncol = ncol(flow.nodes.cum), dimnames = list(NULL, colnames(flow.nodes.cum)))

} else{

flow.init <- flow.nodes.cum[s-1,]

}

flow.nodes.cum <- flow.nodes.cum[s:e,]

time.index <- time.index[s:e]

#rm(s, e)

# Other setup

pipes <- pipe.segments$names

# Vectorized calculations for loops ------------------------------------------------

# Instantaneous flowrate for nodes/pipes

flow.nodes.inst <- as.matrix(rbind((flow.nodes.cum[1,] - flow.init), diff(as.matrix(flow.nodes.cum))))

# Determine instantaneous flow from each pipe

flow.pipes.inst <- matrix(NA, nrow = nrow(flow.nodes.cum), ncol = length(pipes), dimnames = list(NULL, pipes))

for (p in pipes) {

up <- as.character(pipe.segments$upstream[pipe.segments$names %in% p])

dn <- as.character(pipe.segments$downstream[pipe.segments$names %in% p])

flow.pipes.inst[,p] <- pmin(flow.nodes.inst[,up], flow.nodes.inst[,dn])

}

rm(up, dn, p)

# And cumulative flowrate for pipes

flow.pipes.cum <- as.matrix(cumsum(as.data.frame(flow.pipes.inst)))

#simple visuals

#ss <- seq(1, nrow(flow.nodes.cum), round(nrow(flow.nodes.cum)/1000, 0))

#matplot(x = time.index[ss], y = flow.nodes.inst[ss,])

#plot(time.index[ss]+time.sim.start, flow.nodes.cum$SL[ss])

# Define 'time.flow', a record of the age of each unit of water to pass through each pipe

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

time.flow <- list()

for (p in pipes){

time.flow[[p]]$en.time <- as.integer(

rep(NA, # length of each vector = cumulative volume experienced by the pipe

as.integer(flow.pipes.cum[nrow(flow.nodes.cum),p]) + pipe.segments$vol[pipe.segments$names == p]) # plus the volume of each pipe (they start full)

)

}

# Fill 'time.flow' with initial pipe volume -------------------------------

if (per == 1){

# Assume values for initial pipe volume

for (p in pipes){

if (p == "T1-T3" | p == "WH-T4"){

time.flow[[p]]$en.time[1:pipe.segments$vol[pipe.segments$names == p]] <- -WH.HRT

} else {

time.flow[[p]]$en.time[1:pipe.segments$vol[pipe.segments$names == p]] <- 0

}

}

} else{

# Fill from previous

for (p in pipes){

time.flow[[p]]$en.time[1:pipe.segments$vol[pipe.segments$names == p]] <-

stored.in.pipes[[p]]$en.time

}

}

# Move water through pipes ------------------------------------------------

# Define counters for upstream and downstream pipes

# -These have the effect of shifting pipe volume forward

pipe.vol.used.up <- matrix(0, nrow = 1, ncol = ncol(flow.pipes.cum), dimnames = list(NULL, pipes))

pipe.vol.used.dn <- matrix(pipe.segments$vol[colnames(flow.pipes.cum) %in% pipe.segments$names],

nrow = 1, ncol = ncol(flow.pipes.cum), dimnames = list(NULL, pipes))

# Loop through each second of data

for (t in 1:nrow(flow.nodes.cum)) { #nrow(flow.nodes.cum)

# Loop through each active node

for (n in names(flow.nodes.inst[t, flow.nodes.inst[t,] > 0])) {

# Water from upstream pipes is added to this 'slug' which is then delivered to downstream pipes

# Initialize empty slug here

slug <- NULL

# If n == service line connection, add "fresh" water

if (n == "SL"){

slug <- rep(time.index[t] - 1, flow.pipes.inst[t,"SL-WS"]) # "Fresh" water with age of 1 sec is added to slug

# Otherwise collect 'slug' from upstream pipes

} else{

# Create slug from upstream pipes

upstrpipes <- pipe.segments$names[pipe.segments$downstream == n]

upstrpipes <- upstrpipes[upstrpipes %in% names(flow.pipes.inst[t,][flow.pipes.inst[t,] > 0])]

for (u in upstrpipes) {

vol.start.up <- as.numeric(pipe.vol.used.up[,u]) + 1

vol.end.up <- (vol.start.up - 1) + min(flow.nodes.inst[t,n], flow.pipes.inst[t,u])

# Confirm No NAs added to slug

#if (any(is.na(time.flow[[u]]$en.time[vol.start.up:vol.end.up]))){

# stop("At least one 'NA' to be added to slug")}

slug <- c(slug, time.flow[[u]]$en.time[vol.start.up:vol.end.up])

pipe.vol.used.up[,u] <- pipe.vol.used.up[,u] + (vol.end.up - (vol.start.up-1))

}

}# else

# Deliver slug onto the appropriate downstream pipe segments

dnstrpipes <- pipe.segments$names[pipe.segments$upstream == n]

dnstrpipes <- dnstrpipes[dnstrpipes %in% names(flow.pipes.inst[t,][flow.pipes.inst[t,] > 0])]

if (length(dnstrpipes) > 0){

for (d in dnstrpipes) {

vol.start.dn <- as.numeric(pipe.vol.used.dn[,d]) + 1

vol.end.dn <- (vol.start.dn - 1) + min(flow.nodes.inst[t,n], flow.pipes.inst[t,d])

vol.diff <- vol.end.dn - (vol.start.dn - 1)

time.flow[[d]]$en.time[vol.start.dn:vol.end.dn] <- slug[1:(vol.diff)]

pipe.vol.used.dn[,d] <- pipe.vol.used.dn[,d] + (vol.end.dn - (vol.start.dn-1))

slug <- slug[-(1:vol.diff)] # Delete the portion of the slug that was consumed

# Confirm no NAs in pasted values

#if (any(is.na(time.flow[[dnstrpipes[d]]]$en.time[vol.start.dn:vol.end.dn]))){

# stop("Pasted slug contains at least one 'NA'")}

}# for downstream pipes

}#if

}#n

}#t

# Identify any NAs in output

c <- rep(0, length(time.flow))

for (p in 1:length(time.flow)) {

c[p] <- length(time.flow[[p]]$en.time[is.na(time.flow[[p]]$en.time)])

#if (length(time.flow[[p]]$en.time[is.na(time.flow[[p]]$en.time)]) > 0){

# stop(paste(length(time.flow[[p]]$en.time[is.na(time.flow[[p]]$en.time)]), "NAs found in pipe", pipes[p]))}

}

cbind(names(time.flow), c)

rm(c)

# Clean up after loop

rm(dnstrpipes, vol.end.dn, vol.start.dn, d,

upstrpipes, vol.end.up, vol.start.up, u,

slug, vol.diff, t, n, p)

# Water age calculation ---------------------------------------------------

# Define the fixtures at which to calculate age

fixt.mon <- c("SL.1", "cold.kitchen.sink", "cold.bath2.sink",

"WH",

"hot.kitchen.sink", "hot.bath2.sink", "mix.bath2.shower")

# Define 'water.age' list to record water age for each monitored fixture ('fixt.mon')

water.age <- list()

for (f in fixt.mon){

water.age[[f]]$ex.time <- as.numeric(rep(NA, sum(flow.nodes.inst[, f]))) # time each unit water exits the building plumbing

water.age[[f]]$age <- as.numeric(rep(NA, sum(flow.nodes.inst[, f]))) # water age of each unit that exits the fixture

}

for (f in fixt.mon){#length(fixt.mon)

# Determine the time that each unit water entered the plumbing

ex <- water.age[[f]]$ex.time

p <- pipe.segments$names[pipe.segments$downstream == f]

vol.end.ex <- (flow.pipes.cum[,p]) # end volume for entry time

vol.start.ex <- (vol.end.ex - flow.pipes.inst[,p] + 1) # start volume for entry time

for (t in 1:nrow(flow.nodes.cum)){

if (flow.pipes.inst[t,p] > 0){

ex[vol.start.ex[t]:vol.end.ex[t]] <- time.index[t]

}

}

# Determine exit time for each unit volume

water.age[[f]]$ex.time <- ex

# Determine age for each unit volume

water.age[[f]]$age <- water.age[[f]]$ex.time -

time.flow[[p]]$en.time[1:sum(flow.nodes.inst[, f])]

rm(ex, vol.start.ex, vol.end.ex)

}

# Confirm no NAs in water age output

c <- rep(0, length(fixt.mon))

for (p in 1:length(fixt.mon)) {

c[p] <- length(water.age[[p]]$age[is.na(water.age[[p]]$age)])

#if (length(time.flow[[p]]$en.time[is.na(time.flow[[p]]$en.time)]) > 0){

# stop(paste(length(time.flow[[p]]$en.time[is.na(time.flow[[p]]$en.time)]), "NAs found in pipe", pipes[p]))}

}

cbind(fixt.mon, c)

rm(c, f, p, t,

flow.nodes.cum, flow.nodes.inst,

flow.pipes.cum, flow.pipes.inst,

pipe.vol.used.dn, pipe.vol.used.up)

# Prepare age dataframe for plotting --------------------------------------

library(data.table)

age <- data.frame(NULL)

for (f in 1:length(fixt.mon)) {

g <- as.data.table(water.age[[f]])

g$fixt <- fixt.mon[f]

age <- rbind(age, g)

rm(g)

}

rm(f)

age$ex.time <- age$ex.time + time.sim.start

# Set order for plotting

age$fixt <- factor(age$fixt,

levels = c("SL.1", "cold.kitchen.sink", "cold.bath2.sink", "WH", "hot.kitchen.sink", "hot.bath2.sink", "mix.bath2.shower"))

# Rename levels for plotting

#levels(age$fixt) <- c("SL", "Cold Kitchen Sink", "Cold Bath Sink", "WH", "Hot Kitchen Sink", "Hot Bath Sink", "Mix Shower")

# Save volume currently stored in pipes -----------------------------------

stored.in.pipes <- list()

for (p in pipes){

end.vol <- length(time.flow[[p]]$en.time)

start.vol <- (length(time.flow[[p]]$en.time) - pipe.segments$vol[pipe.segments$names == p]) + 1

stored.in.pipes[[p]]$en.time[1:pipe.segments$vol[pipe.segments$names == p]] <- time.flow[[p]]$en.time[start.vol:end.vol]

}

rm(time.flow)

# END OF LOOP -------------------------------------------------------------

if (per == 1){

save.image("waterage\_output\_20201019\_(1)1-1.5M")

}else if (per == 2){

save.image("waterage\_output\_20201019\_(2)1.5M-3M")

} else if (per == 3){

save.image("waterage\_output\_20201019\_(3)3M-end")

}

timer <- Sys.time() - timer.start

timer

## associate\_waterage\_with\_sample.R

library(lubridate)

library(dplyr)

wq.df <- read.csv("/mnt/ufs18/home-037/julienry/Desktop/Water Age/reneww\_data\_20190712.csv")

wq.df$Date <- mdy(wq.df$Date)

wq.df$Time <- hm(wq.df$Time)

wq.df$Location[wq.df$Location == "SL"] <- "SL.1"

time.delay <- 14 #days

fixt <- levels(age.all$fixt)

wq.df$age.mean <- NA

wq.df$age.median <- NA

wq.df$age.95per <-NA

wq.df$age.5per <- NA

for (d in 1:length(wq.df$Date)){

s <- age.all$age[age.all$fixt == wq.df$Location[d] &

age.all$ex.time >= wq.df$Date[d] - time.delay &

age.all$ex.time <= wq.df$Date[d]]

wq.df$age.mean[d] <- mean(s)

wq.df$age.median[d] <- median(s)

wq.df$age.95per[d] <- quantile(s, 0.95)

wq.df$age.5per[d] <- quantile(s, 0.05)

rm(s)

}

write.csv(wq.df, "/mnt/ufs18/home-037/julienry/Desktop/Water Age/Results/Assumed Sink Flows/reneww\_data\_20201020\_waterage.csv")

## waterage\_statistical\_analysis\_20210316.R.

# Load libraries

library(dplyr)

library(ggplot2)

library(GGally)

library(ggthemes)

library(lubridate)

# Import Analytical Data --------------------------------------------------

# Personal Laptop

setwd("C:/Users/Ryan/OneDrive - Michigan State University/Publications/Water Age/Model/Output")

# Personal Desktop

#setwd("/home/ryan/Everything/PhD/Water Age Paper/Model/")

wq.df <- as.data.frame(read.csv("reneww\_data\_20210310\_waterage.csv", stringsAsFactors = FALSE))

wq.df$Location[wq.df$Location == "SL.1"] <- "SL"

# Or choose these manually

#setwd(choose.dir())

#wq.df <- as.data.frame(read.csv(choose.files(), stringsAsFactors = FALSE))

# Arrange by fixture

wq.df$Location <- factor(wq.df$Location,

levels = c("SL", "cold.kitchen.sink", "cold.bath2.sink", "WH", "hot.kitchen.sink", "hot.bath2.sink", "mix.bath2.shower"),

ordered = TRUE)

wq.df <- arrange(wq.df, Location)

wq.df$System <- rep(NA, nrow(wq.df))

for (i in 1:nrow(wq.df)){

# Define Hot/Cold/Mixed systems

if (wq.df$Location[i] %in% c("WH", "hot.kitchen.sink", "hot.bath2.sink")){

wq.df$System[i] <- "H"

}

else if (wq.df$Location[i] %in% c("SL", "cold.kitchen.sink", "cold.bath2.sink")){

wq.df$System[i] <- "C"

}

else if (wq.df$Location[i] %in% c("mix.bath2.shower")){

wq.df$System[i] <- "M"

}

# Assume sample was collected at noon if not labeled

if (wq.df[i,"Time"] == "M"){

wq.df[i,"Time"] <- as.character("12:00")

}

}

rm(i)

# Transform Analytical Results --------------------------------------------

wq.df$Date <- mdy(wq.df$Date)

wq.df$Time <- hm(wq.df$Time)

wq.df$date\_time <- wq.df$Date + wq.df$Time

wq.df$pH <- as.numeric(wq.df$pH)

wq.df$Temp <- as.numeric(wq.df$Temp)

wq.df$DO <- as.numeric(wq.df$DO)

# Assumes 50% of detection limit for all ND chlorine samples

wq.df$Total.Cl[wq.df$Total.Cl == 0 | wq.df$Total.Cl == "0.00"] <- 0.1/2

wq.df$Total.Cl <- as.numeric(wq.df$Total.Cl)

wq.df$Free.Cl[wq.df$Free.Cl == 0 | wq.df$Free.Cl == "0.00"] <- 0.1/2

wq.df$Free.Cl <- as.numeric(wq.df$Free.Cl)

wq.df$TCC.1 <- as.numeric(wq.df$TCC.1)

wq.df$TCC.2 <- as.numeric(wq.df$TCC.2)

wq.df$TCC.3 <- as.numeric(wq.df$TCC.3)

wq.df$TCC.4 <- as.numeric(wq.df$TCC.4)

wq.df$TCC.5 <- as.numeric(wq.df$TCC.5)

wq.df$TCC.6 <- as.numeric(wq.df$TCC.6)

wq.df$TCC <- rowMeans(cbind(wq.df$TCC.1, wq.df$TCC.2, wq.df$TCC.3, wq.df$TCC.4, wq.df$TCC.5, wq.df$TCC.6), na.rm = TRUE)

wq.df$TCC[is.nan(wq.df$TCC)] <- NA # Replace 'NaN' with 'NA'

wq.df$HPC <- as.numeric(wq.df$HPC)

# Legionella

# Assumes 50% of detection limit for all ND Legionella samples

wq.df$Leg.sp.1[!(wq.df$Leg.sp.1 > 0)] <- as.numeric(gsub("[^0-9.]", "", wq.df$Leg.sp.1[!(wq.df$Leg.sp.1 > 0)]))/2

wq.df$Leg.sp.2[!(wq.df$Leg.sp.2 > 0)] <- as.numeric(gsub("[^0-9.]", "", wq.df$Leg.sp.2[!(wq.df$Leg.sp.2 > 0)]))/2

wq.df$Leg.sp.3[!(wq.df$Leg.sp.3 > 0)] <- as.numeric(gsub("[^0-9.]", "", wq.df$Leg.sp.3[!(wq.df$Leg.sp.3 > 0)]))/2

wq.df$Leg.sp.1 <- as.numeric(wq.df$Leg.sp.1)

wq.df$Leg.sp.2 <- as.numeric(wq.df$Leg.sp.2)

wq.df$Leg.sp.3 <- as.numeric(wq.df$Leg.sp.3)

#Take average of all

wq.df$Leg.sp <- rowMeans(cbind(wq.df$Leg.sp.1, wq.df$Leg.sp.2, wq.df$Leg.sp.3), na.rm = TRUE)

wq.df$Leg.sp[is.nan(wq.df$Leg.sp)] <- NA # Replace 'NaN' with 'NA'

# TOC

wq.df$TOC.1 <- as.numeric(wq.df$TOC.1)

wq.df$TOC.2 <- as.numeric(wq.df$TOC.2)

wq.df$TOC <- rowMeans(cbind(wq.df$TOC.1, wq.df$TOC.2), na.rm = TRUE)

wq.df$TOC[is.nan(wq.df$TOC)] <- NA # Replace 'NaN' with 'NA'

#DOC

wq.df$DOC.1 <- as.numeric(wq.df$DOC.1)

wq.df$DOC.2 <- as.numeric(wq.df$DOC.2)

wq.df$DOC <- rowMeans(cbind(wq.df$DOC.1, wq.df$DOC.2), na.rm = TRUE)

wq.df$DOC[is.nan(wq.df$DOC)] <- NA # Replace 'NaN' with 'NA'

# Alkalinity

wq.df$Alka.1 <- as.numeric(wq.df$Alka.1)

wq.df$Alka.2 <- as.numeric(wq.df$Alka.2)

wq.df$Alka <- rowMeans(cbind(wq.df$Alka.1, wq.df$Alka.2), na.rm = TRUE)

wq.df$Alka[is.nan(wq.df$Alka)] <- NA # Replace 'NaN' with 'NA'

# Assumes 50% of detection limit for all ND TTHM samples

wq.df$TTHM.1[wq.df$TTHM.1 == 0 | wq.df$TTHM.1 == "0.00"] <- 0.1/2

wq.df$TTHM.1 <- as.numeric(wq.df$TTHM.1)

wq.df$TTHM.2[wq.df$TTHM.2 == 0 | wq.df$TTHM.2 == "0.00"] <- 0.1/2

wq.df$TTHM.2 <- as.numeric(wq.df$TTHM.2)

wq.df$TTHM <- rowMeans(cbind(wq.df$TTHM.1, wq.df$TTHM.2), na.rm = TRUE)

wq.df$TTHM[is.nan(wq.df$TTHM)] <- NA # Replace 'NaN' with 'NA'

wq.df$System <- as.factor(wq.df$System)

# Import Usage Data -------------------------------------------------------

use.data <- as.data.frame(read.csv("allEvents 201706-201811.csv"))

# Fix timestamps

use.data$startTime <- as.POSIXct(strptime(use.data$startTime, "%m/%d/%Y %H:%M:%S"))

use.data$endTime <- as.POSIXct(strptime(use.data$endTime, "%m/%d/%Y %H:%M:%S"))

# Define naming convention

fixt.names <- rbind(c("Bath2\_Shower", "Bath2\_Sink\_Cold\_vol", "Bath2\_Sink\_Hot\_vol", "City\_Main\_vol", "DHW\_Cold\_Inlet\_vol", "DHW\_Hot\_Return\_vol", "Island\_Cold\_vol", "Island\_Hot\_vol", "Kitchen\_Cold\_vol", "Kitchen\_Sink\_Hot\_vol"),

c("mix.bath2.shower", "cold.bath2.sink", "hot.bath2.sink", "SL", "WH", "hot.recycle", "cold.kitchen.isl", "hot.kitchen.isl", "cold.kitchen.sink", "hot.kitchen.sink"))

# Develop Usage Stats -----------------------------------------------------

# Settings

metrics <- c("num.events", "vol.events", "meanTSL", "maxTSL")

#periods <- c(1,2,3,5,7,10,14,21,28,35,42)

#periods <- seq(from=1, to=45, by=1)

periods <- 14

age.names <- paste(rep(metrics, 3), rep(paste("day", periods, sep = ""), each = 3), sep = ".")

min.tSL <- 5 # Minimum duration between events (seconds)

num.events <- as.data.frame(matrix(NA, nrow = nrow(wq.df), ncol = length(periods), dimnames = list(NULL, paste0(rep("day"), periods))))

vol.events <- as.data.frame(matrix(NA, nrow = nrow(wq.df), ncol = length(periods), dimnames = list(NULL, paste0(rep("day"), periods))))

meanTSL <- as.data.frame(matrix(NA, nrow = nrow(wq.df), ncol = length(periods), dimnames = list(NULL, paste0(rep("day"), periods))))

maxTSL <- as.data.frame(matrix(NA, nrow = nrow(wq.df), ncol = length(periods), dimnames = list(NULL, paste0(rep("day"), periods))))

m95TSL <- as.data.frame(matrix(NA, nrow = nrow(wq.df), ncol = length(periods), dimnames = list(NULL, paste0(rep("day"), periods))))

# Calculate each water age metric

# For each sample, s

for (s in 1:nrow(wq.df)){

# For each time period, p

for (p in 1:length(periods)){

# Actually calculate water age metrics

# Number of Events

num.events.inst <- nrow(subset(use.data, variableName == fixt.names[1,][fixt.names[2,] %in% wq.df$Location[s]] &

endTime > wq.df$date\_time[s] - (periods[p]\*60\*60\*24) &

endTime <= wq.df$date\_time[s] &

timeSinceLastEvent >= min.tSL))

if (num.events.inst == 0){

num.events.inst <- 1

vol.events.inst <- 1

meanTSL.inst <- NA

maxTSL.inst <- NA

m95TSL.inst <- NA

} else {

# Cumulative Volume

vol.events.inst <- sum(subset(use.data, variableName == fixt.names[1,][fixt.names[2,] %in% wq.df$Location[s]] &

endTime > wq.df$date\_time[s] - (periods[p]\*60\*60\*24) &

endTime <= wq.df$date\_time[s],

select = sum),na.rm = TRUE)

# Mean timeSinceLast

meanTSL.inst <- mean(subset(use.data, variableName == fixt.names[1,][fixt.names[2,] %in% wq.df$Location[s]] &

endTime > wq.df$date\_time[s] - (periods[p]\*60\*60\*24) &

endTime <= wq.df$date\_time[s] &

timeSinceLastEvent >= min.tSL)[,"timeSinceLastEvent"])

# Max timeSinceLast

maxTSL.inst <- max(subset(use.data, variableName == fixt.names[1,][fixt.names[2,] %in% wq.df$Location[s]] &

endTime > wq.df$date\_time[s] - (periods[p]\*60\*60\*24) &

endTime <= wq.df$date\_time[s] &

timeSinceLastEvent >= min.tSL)[,"timeSinceLastEvent"])

# Max timeSinceLast

m95TSL.inst <- quantile(subset(use.data, variableName == fixt.names[1,][fixt.names[2,] %in% wq.df$Location[s]] &

endTime > wq.df$date\_time[s] - (periods[p]\*60\*60\*24) &

endTime <= wq.df$date\_time[s] &

timeSinceLastEvent >= min.tSL)[,"timeSinceLastEvent"], 0.975)

}

# Save output

num.events[s, p] <- num.events.inst

vol.events[s, p] <- vol.events.inst

meanTSL[s, p] <- meanTSL.inst

maxTSL[s, p] <- maxTSL.inst

m95TSL[s, p] <- m95TSL.inst

}

}

# Add desired usage metrics to wq.df

wq.df$num.events <- num.events$day14

wq.df$vol.events <- vol.events$day14

wq.df$meanTSL <- meanTSL$day14

wq.df$maxTSL <- maxTSL$day14

wq.df$m95TSL <- m95TSL$day14

# Remove unecessary info

rm(age.names, s, p, num.events.inst, vol.events.inst, meanTSL.inst, maxTSL.inst, m95TSL.inst, use.data, num.events, vol.events, meanTSL, maxTSL, m95TSL, metrics)

#write.csv(wq.df, file = "wq.data.csv")

# Box-Cox Transformation --------------------------------------------------

# library(dplyr)

# install.packages("dplyr", repos=c("http://rstudio.org/\_packages",

# "http://cran.rstudio.com",dependencies=TRUE))

# library("caret")

# dat <- as.numeric(my\_data[,"Leg.sp"])

# dat <- dat[!is.na(dat)]

#

# x <- BoxCoxTrans(dat)

# d <- predict(x, dat)

# plot(dat, d)

# #hist(dat)

# #hist(d)

#

# wq.df$Leg.sp.BC <- wq.df$Leg.sp

# wq.df$Leg.sp.BC[!is.na(wq.df$Leg.sp.BC)] <- d

# Correlation Coefficients ------------------------------------------------

# Selected for correlation matrix

my\_data <- as.matrix(dplyr::select(wq.df, "pH", "Temp", "DO", "Total.Cl", "Free.Cl", "TOC", "DOC", "Alka", "TTHM", "TCC", "HPC", "Leg.sp","vol.events", "num.events", "meanTSL", "maxTSL", "age.mean", "age.median", "age.95per"))

# Correlation matrix

library(Hmisc)

x <- rcorr(my\_data, type = "spearman")

#x <- cor(my\_data, method = "spearman", use = "complete.obs")

# I chose spearman over pearson because it is a non-parametric test

write.csv(x$r, file = "MatCor.csv")

write.csv(x$n, file = "MatNum.csv")

write.csv(x$P, file = "MatSig.csv")

# Select Data for PCA Analysis --------------------------------------------

# Uses all but data excluded during PCA (described in outline)

newdata.full <- na.omit(select(wq.df, Location, System, Date,

pH, Temp, DO, Total.Cl, TCC, HPC, Leg.sp, TOC, Alka, TTHM,

vol.events, meanTSL,

age.mean, age.median, age.95per))

newdata.cat <- newdata.full[,1:3]

newdata <- newdata.full[,-(1:3)]

rm(newdata.full)

summary(newdata)

# Check whether to use PCA with Bartlett's Sphericty

# http://eric.univ-lyon2.fr/~ricco/tanagra/fichiers/en\_Tanagra\_KMO\_Bartlett.pdf

n <- nrow(newdata)

p <- ncol(newdata)

R <- cor(newdata)

chi2 <- -(n-1-(2\*p+5)/6)\*log(det(R))

ddl <- p\*(p-1)/2

pchisq(chi2,ddl,lower.tail=F)

# Sufficiently correlated to use PCA?

pchisq(chi2,ddl,lower.tail=F) < 0.05

# Confirm with KMO's MSA

library(psych)

KMO(newdata)

# MSA = 0.74; "middling"

# Yes - Proceed with PCA

# Conduct PCA -------------------------------------------------------------

pca <- prcomp(newdata, scale. = TRUE)

biplot(pca, scale = 0)

plot(pca, type = "line",

main = "PCA Scree Plot")

# Consider only PCs with std dev > 1 per Kaiser's Rule

abline(a = 1, b = 0, col = "red", lwd = 4)

summary(pca)

# Loadings on significant PCs

pca$rotation[,1:length(pca$sdev[pca$sdev > 1])]

# Print all to file

write.csv(pca$rotation, file = "PCA.csv")

rm(chi2, ddl, R, pca, newdata, newdata.cat, n, p) # clean up

# Analysis with CSTAT -----------------------------------------------------

# 18 predictor variables initially available (plus legionella)

ncol(select(wq.df, Leg.sp, pH, Temp, DO, Total.Cl, TCC, HPC, TOC, Alka, TTHM, num.events, vol.events, meanTSL, maxTSL, Free.Cl, DOC, age.mean, age.median, age.95per))

# Missing 148 values (out of possible 406) for Legionella. No more than 258 observations can be include in GLM

sum(is.na(wq.df$Leg.sp))

# Decided to use remove Free.Cl and DOC before analysis because they were missing so often and had acceptable surrogates (Total.Cl and TOC)

sum(is.na(wq.df$Free.Cl)) # 147 missing

sum(is.na(wq.df$DOC)) # 35 missing

# Removing these Free.Cl and DOC allows for the inclusion of 89 additional observations (133 to 222)

nrow(na.omit(select(wq.df, Leg.sp, pH, Temp, DO, Total.Cl, TCC, HPC, TOC, Alka, TTHM, num.events, vol.events, meanTSL, maxTSL))) - nrow(na.omit(select(wq.df, Leg.sp, pH, Temp, DO, Total.Cl, TCC, HPC, TOC, Alka, TTHM, num.events, vol.events, meanTSL, maxTSL, Free.Cl, DOC)))

# Chose the following variables for analysis:

# Leg.sp, pH, Temp, DO, Total.Cl, TCC, HPC, TOC, Alka, TTHM, num.events, vol.events, meanTSL, maxTSL, age.mean, age.median, age.95per

library(dplyr)

data <- dplyr::select(wq.df, date\_time, System, Location, Leg.sp,

pH, Temp, DO, Total.Cl, TCC, HPC, TOC, Alka, TTHM,

num.events, vol.events, meanTSL, maxTSL,

age.mean, age.median, age.95per)

# Log transformations

# These are based on lowest Shapiro-Wilks W-statistic

data$Total.Cl <- log10(data$Total.Cl)

data$TOC <- log10(data$TOC)

data$Alka <- log10(data$Alka)

data$HPC <- log10(data$HPC)

data$TCC <- log10(data$TCC)

data$Leg.sp <- log10(wq.df$Leg.sp)

# Did NOT log-transform use metrics

#data$num.events <- log10(data$num.events)

#data$vol.events <- log10(data$vol.events)

#data$meanTSL <- log10(data$meanTSL)

#data$maxTSL <- log10(data$maxTSL)

#data$age.mean <- log10(data$age.mean)

#data$age.median <- log10(data$age.median)

# Remove categorical variables

dat <- na.omit(data[,-(1:3)])

# Define GLM

global.glm <- glm(Leg.sp~.,dat, family = "gaussian", na.action = "na.fail")

# Results

plot(global.glm$y, global.glm$fitted.values)

cor(global.glm$y, global.glm$fitted.values)

summary(global.glm)

# Dredge GLM for best-fitting variable combinations

#install.packages("MuMIn")

library(MuMIn)

global.glm.dedge <- dredge(global.glm, rank = "AICc")

# Results

global.glm.dedge[1:39] # top 39 models have dAIC < 2.0

# Top: Leg.sp ~ age.95per + HPC + meanTSL + TCC + Total.Cl + TTHM

# All competing: Leg.sp ~ age.95per + age.mean + age.median + HPC + maxTSL + meanTSL + num.events + pH + TCC + TOC + vol.events

# Select new dataframe with only variables found in top and competing models per 'dredge()'

dat.new <- select(data, c(Leg.sp, age.95per, age.mean, age.median, HPC,

maxTSL, meanTSL, num.events, pH, TCC, TOC,

vol.events, Location))

dat.new <- na.omit(dat.new) # remove NAs

dat.loc <- dat.new["Location"]

dat.new <- subset(dat.new, select = -Location )

# Result has 242 observations of 12 variables

# Fit a GLM to the new data set

global.glm.new <- glm(Leg.sp~.,dat.new, family = "gaussian", na.action = "na.fail")

global.glm.dredge.new <- dredge(global.glm.new)

global.glm.dredge.new[1:12] # the top 20 models have dAIC < 2.0

# Top ranked model (best): age.95per, HPC, maxTSL, mean.TSL, num.events, pH, TCC, TOC

# Competing models (comp): age.95per, age.mean, age.median, HPC, maxTSL, meanTSL, num.events, pH, TCC, TOC, vol.events

# Mixed model with random effects for fixture locations -------------------

# Add "Location" back into data

dat.new <- cbind(dat.new, dat.loc)

#install.packages("glmmTMB", type="source", dependencies = T)

# install.packages("glmmTMB",

# repos="https://glmmTMB.github.io/glmmTMB/repos",

# type="binary")

library(glmmTMB)

# Top-Ranked Model --------------------------------------------------------

global.glm.me.best <- glmmTMB(Leg.sp ~age.95per + HPC + maxTSL + meanTSL + num.events + pH + TCC + TOC + (1|Location),

data = dat.new, family = "gaussian", na.action = "na.fail")

# Check for convergence

global.glm.me.best$sdr$pdHess

# Check for singularity

library(performance)

performance::check\_singularity(global.glm.me.best)

# Residuals

summary(global.glm.me.best)

global.glm.me.best.resid <- resid(global.glm.me.best, "pearson")

plot(global.glm.me.best.resid)

hist(global.glm.me.best.resid)

# Q-Q residuals plot

qqnorm(global.glm.me.best.resid, pch = 1, frame = FALSE)

qqline(global.glm.me.best.resid, col = "steelblue", lwd = 2)

# Check homoscedasticity

performance::check\_heteroscedasticity(global.glm.me.best)

# Competing Models --------------------------------------------------------

global.glm.me.comp <- glmmTMB(Leg.sp ~ age.95per + age.mean + age.median + HPC + maxTSL + meanTSL + num.events + pH + TCC + TOC + vol.events + (1|Location),

data = dat.new, family = "gaussian", na.action = "na.fail")

# Check for convergence

global.glm.me.comp$sdr$pdHess

# MODEL FAILS TO CONVERGE

# This is likely due to overparameterization (12 predictor variables and only 242 observations)

# Check for singularity

performance::check\_singularity(global.glm.me.comp)

# Residuals

summary(global.glm.me.comp)

fixef(global.glm.me.comp)

global.glm.me.comp.resid <- resid(global.glm.me.comp, "pearson")

plot(global.glm.me.comp.resid)

hist(global.glm.me.comp.resid)

# Q-Q residuals plot

qqnorm(global.glm.me.comp.resid, pch = 1, frame = FALSE)

qqline(global.glm.me.comp.resid, col = "steelblue", lwd = 2)

# Check homoscedasticity

performance::check\_heteroscedasticity(global.glm.me.comp)

# Final Model Comparison --------------------------------------------------

#install.packages("DHARMa")

library(DHARMa)

sim <- simulateResiduals(global.glm.me.best, n = 10000)

plot(sim)

testOutliers(sim)

# Single outlier identified in top model

sim <- simulateResiduals(global.glm.me.comp, n = 10000)

plot(sim)

testOutliers(sim)

# Check for Multicollinearity

check\_collinearity(global.glm.me.best, component = "all")

check\_collinearity(global.glm.me.comp, component = "all")

# Report model performance

model\_performance(global.glm.me.best, metrics = "all")

model\_performance(global.glm.me.comp, metrics = "all")

# Both models perform similarly (intuitive as only difference is TTHM)

# All 6 metrics suggest 'best' is indeed best

# Final Model Selection ---------------------------------------------------

# Select top-ranked model: Leg.sp ~ DO + HPC + maxTSL + meanTSL + num.events + pH + TCC + TOC + Total.Cl + (1|Location)

summary(global.glm.me.best)

# old version: Leg.sp ~ HPC + meanTSL + num.events + TCC + Total.Cl + Location

# Adding pH + TTHM maintains integrity of GLMM, but does not offer additional predictive power

plot(x = dat.new$Leg.sp, y = predict(global.glm.me.best, dat.new),

main ="Top GLMM Legionella Results",

xlab = "Measured Legionella spp. (Log GC/100 mL)", ylab = "Predicted Legionella spp. (Log GC/100 mL)")

abline(a = 0, b = 1, col = "red")

cor(x = dat.new$Leg.sp, y = predict(global.glm.me.best, dat.new))

plot(x = dat.new$Leg.sp, y = predict(global.glm.me.comp, dat.new),

main ="Competing Model Legionella Results",

xlab = "Measured Legionella spp. (Log GC/100 mL)", ylab = "Predicted Legionella spp. (Log GC/100 mL)")

abline(a = 0, b = 1, col = "red")

cor(x = dat.new$Leg.sp, y = predict(global.glm.me.comp, dat.new))

# And compare to previous GLM w/o location

plot(x = dat.new$Leg.sp, y = predict(global.glm.new, dat.new),

main ="GLM Legionella Results",

xlab = "Measured Legionella spp. (Log GC/100 mL)", ylab = "Predicted Legionella spp. (Log GC/100 mL)")

abline(a = 0, b = 1, col = "red")

cor(x = dat.new$Leg.sp ,y = predict(global.glm.new, dat.new))

load("C:/Users/Ryan/OneDrive - Michigan State University/Publications/Water Age/Model/Output/messy\_data.RData")

# Clean environment

rm(list=setdiff(ls(), c("dat.new")))

# BGLR --------------------------------------------------------------------

# Top ranked model (best): age.95per, HPC, maxTSL, mean.TSL, num.events, pH, TCC, TOC

# Competing models (comp): age.95per, age.mean, age.median, HPC, maxTSL, meanTSL, num.events, pH, TCC, TOC, vol.events

# Select subset of predictor variables

vs.top <- select(dat.new, Leg.sp, age.95per, HPC, maxTSL, meanTSL, num.events, pH, TCC, TOC, Location)

vs.comp <- select(dat.new, Leg.sp, age.95per, age.mean, age.median, HPC, maxTSL, meanTSL, num.events, pH, TCC, TOC, vol.events, Location)

# Choose which subset to analyze (Choose correct ETA below, also)

data <- vs.comp

# Remove rows with NAs in predictors (requirement for BGLR)

data <- data[complete.cases(data[,-1]),]

rm(dat.new)

#data[is.na(data)] <- 0

# Run BGLR ----------------------------------------------------------------

library(BGLR)

x <- data[,-1] # Select predictors

y <- data[, 1] # Select target variable (Leg.sp)

gc()

#x[is.na(x)] <- 0

# Select the appropriate ETA (based on data set)

#vs.full

#ETA <- list(list(X=x, model = 'BayesC', counts = 2))

#vs.top

# ETA <- list(list(X=x[1:8], model = 'BayesC', counts = 2),

# list(~factor(Location), data=x, model='BayesC'))

#vs.comp

ETA <- list(list(X=x[1:11], model = 'BayesC', counts = 2),

list(~factor(Location), data=x, model='BayesC'))

fm <- BGLR(y=y, ETA=ETA,

nIter=100000, burnIn=2000, thin=5,

verbose = TRUE, saveAt = "dataFile-top")

save.image("C:/Users/Ryan/OneDrive - Michigan State University/Publications/Water Age/Model/Output/BGLR/BayesC\_VScomp\_100000.RData")

#save(fm, file='vs2.rda')

# Results -----------------------------------------------------------------

par(mfrow=c(1,1))

plot(fm$y, fm$yHat, xlab = "y", ylab = "yHat")

abline(a = 0, b = 1, col = "blue", lwd = 2)

cor(fm$y, fm$yHat, method = ("pearson"), use = "pairwise.complete")

# Estimated effects (absolute value)

plot(abs(fm$ETA[[1]]$b))

# Estimated probablity of non-zero effects (average)

fm$ETA[[1]]$probIn

# Estimated probablity of non-zero effects (per marker)

plot(fm$ETA[[1]]$d)

cbind(fm$ETA[[1]]$d, fm$ETA[[1]]$b, fm$ETA[[1]]$SD.b)

fm$fit$DIC

str(fm)

# Parameter Estimates Plots -----------------------------------------------

par(mfrow=c(3,3))

# Frequency plots of each estimated parameter

for (a in 1:length(fm$ETA[[1]]$b)) {

hist(rnorm(1000000, mean = fm$ETA[[1]]$b[a], sd = fm$ETA[[1]]$SD.b[a]), main = names(fm$ETA[[1]]$b[a]), breaks = 30)

abline(v = 0, col = "red", lwd = 3)

}

par(mfrow=c(1,1))

hist(rnorm(100000, mean = fm$ETA[[1]]$b[a], sd = fm$ETA[[1]]$SD.b[a]), main = names(fm$ETA[[1]]$b[a]), breaks = 30)

abline(v = 0, col = "red", lwd = 3)

fm$ETA[[2]]$probIn

fm$ETA[[1]]$R2

plot(fm$yHat ~ fm$y,xlab='Observed',ylab='Predicted')

abline(a=0,b=1,col=4,lwd=2)

# Plots -------------------------------------------------------------------

library(ggplot2)

library(ggthemes)

library(ggpubr)

ggarrange(

ggplot(wq.df, aes(age.mean, Leg.sp)) +

geom\_point(aes(color = System), size = 2) +

scale\_color\_colorblind() +

scale\_y\_log10() +

scale\_x\_log10(),

ggplot(wq.df, aes(meanTSL, Leg.sp)) +

geom\_point(aes(color = System), size = 2) +

scale\_color\_colorblind() +

scale\_y\_log10() +

scale\_x\_log10(),

common.legend = TRUE,

legend = "right"

)

cor(wq.df$Leg.sp, wq.df$age.mean, method = "spearman", use = "pairwise.complete")

cor(wq.df$Leg.sp, wq.df$meanTSL, method = "spearman", use = "pairwise.complete")

# Modeling thermal compliance code

library(lubridate)

# MSU HPCC

setwd("/mnt/ufs18/home-037/julienry/Desktop/Water Age/")

# Linux Desktop

#setwd("/home/ryan/Everything/PhD/Water Age Paper/Model/")

# Windows Laptop

#setwd("C:/Users/Ryan/OneDrive - Michigan State University/Publications/Water Age/Model")

load("/mnt/ufs18/home-037/julienry/Desktop/Water Temperature/Results/processor\_output\_20201130.RData")

load("/mnt/ufs18/home-037/julienry/Desktop/Water Temperature/Results/temp\_data\_201709\_201811.RData")

# Start Timer

timer.start <- Sys.time()

library(dplyr)

flow.nodes.cum <- cum.flow.matrix

rm(cum.flow.matrix)

# Determine mean HRT for WH based on input data

# -Assume WH is filled with water with age equal to the mean HRT

WH.avg.flow <- # Units are delta.v per second

sum(select(flow.nodes.cum[nrow(flow.nodes.cum),], hot.bath1.sink, hot.bath1.shower, hot.kitchen.isl, hot.bath2.shower, hot.bath2.sink, hot.clothes, hot.kitchen.sink, hot.dish)) /

as.numeric(diff(time.index[c(1, nrow(flow.nodes.cum))]))

WH.vol <- as.integer((3 \* 409 + 141) \* 1000 / delta.v) # WH volume in units of delta.v

WH.HRT <- round(WH.vol/WH.avg.flow, 0) # Assumed WH age in seconds

# Temperature Settings ----------------------------------------------------

# Note that these apply to \*any\* length of time

# Literature suggests that water should be kept at temps outside of a range

temp.lo <- 25 # Celsius

temp.hi <- 50 # Celsius

# Assume that temperature is reset to TRUE if temp exceeds threshold

temp.reset <- 60 # Celsius

# Fix timezone issue

tz(use.data.temp$timestamp) <- "America/Detroit"

# Select subset of use data -----------------------------------------------

per <- 1

s <- c(1, 500000, 1000000)

e <- c(500001, 1000001, nrow(flow.nodes.cum))

s <- s[per]

e <- e[per]

if (per == 1){

flow.init <- matrix(0, nrow = 1, ncol = ncol(flow.nodes.cum), dimnames = list(NULL, colnames(flow.nodes.cum)))

} else{

flow.init <- flow.nodes.cum[s-1,]

}

flow.nodes.cum <- flow.nodes.cum[s:e,]

time.index <- time.index[s:e]

#rm(s, e)

# Other setup

pipes <- pipe.segments$names

# Vectorized calculations for loops ------------------------------------------------

# Instantaneous flowrate for nodes/pipes

flow.nodes.inst <- as.matrix(rbind((flow.nodes.cum[1,] - flow.init), diff(as.matrix(flow.nodes.cum))))

# Determine instantaneous flow from each pipe

flow.pipes.inst <- matrix(NA, nrow = nrow(flow.nodes.cum), ncol = length(pipes), dimnames = list(NULL, pipes))

for (p in pipes) {

up <- as.character(pipe.segments$upstream[pipe.segments$names %in% p])

dn <- as.character(pipe.segments$downstream[pipe.segments$names %in% p])

flow.pipes.inst[,p] <- pmin(flow.nodes.inst[,up], flow.nodes.inst[,dn])

}

rm(up, dn, p)

# And cumulative flowrate for pipes

flow.pipes.cum <- as.matrix(cumsum(as.data.frame(flow.pipes.inst)))

#simple visuals

#ss <- seq(1, nrow(flow.nodes.cum), round(nrow(flow.nodes.cum)/1000, 0))

#matplot(x = time.index[ss], y = flow.nodes.inst[ss,])

#plot(time.index[ss]+time.sim.start, flow.nodes.cum$SL[ss])

# Define 'time.flow', a record of the age of each unit of water to pass through each pipe

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

time.flow <- list()

for (p in pipes){

time.flow[[p]]$en.time <- as.integer(

rep(NA, # length of each vector = cumulative volume experienced by the pipe

as.integer(flow.pipes.cum[nrow(flow.nodes.cum),p]) + pipe.segments$vol[pipe.segments$names == p]) # plus the volume of each pipe (they start full)

)

time.flow[[p]]$temp.comp <- as.logical(

rep(NA, # length of each vector = cumulative volume experienced by the pipe

as.integer(flow.pipes.cum[nrow(flow.nodes.cum),p]) + pipe.segments$vol[pipe.segments$names == p]) # plus the volume of each pipe (they start full)

)

}

# Fill 'time.flow' with initial pipe volume -------------------------------

if (per == 1){

# Assume values for initial pipe volume

for (p in pipes){

if (p == "T1-T3" | p == "WH-T4"){

time.flow[[p]]$en.time[1:pipe.segments$vol[pipe.segments$names == p]] <- -WH.HRT

} else {

time.flow[[p]]$en.time[1:pipe.segments$vol[pipe.segments$names == p]] <- 0

}

# Assign inital volume is temperature compliant

time.flow[[p]]$temp.comp[1:pipe.segments$vol[pipe.segments$names == p]] <- TRUE

}

} else{

# Fill from previous

for (p in pipes){

time.flow[[p]]$en.time[1:pipe.segments$vol[pipe.segments$names == p]] <-

stored.in.pipes[[p]]$en.time

}

}

# Move water through pipes ------------------------------------------------

# Define counters for upstream and downstream pipes

# -These have the effect of shifting pipe volume forward

pipe.vol.used.up <- matrix(0, nrow = 1, ncol = ncol(flow.pipes.cum), dimnames = list(NULL, pipes))

pipe.vol.used.dn <- matrix(pipe.segments$vol[colnames(flow.pipes.cum) %in% pipe.segments$names],

nrow = 1, ncol = ncol(flow.pipes.cum), dimnames = list(NULL, pipes))

# Loop through each second of data

for (t in 1:nrow(flow.nodes.cum)) { #nrow(flow.nodes.cum)

temp.time.index <- which(use.data.temp$timestamp == min(use.data.temp$timestamp[use.data.temp$timestamp > (time.index[t] + time.sim.start)]))

# Loop through each active node

for (n in names(flow.nodes.inst[t, flow.nodes.inst[t,] > 0])) {

# Water from upstream pipes is added to this 'slug' which is then delivered to downstream pipes

# Initialize empty slug here

slug <- NULL

slug.temp <- NULL

# If n == service line connection, add "fresh" water

if (n == "SL"){

# Add "fresh" water with age of 1 sec to slug

slug <- rep(time.index[t] - 1, flow.pipes.inst[t,"SL-WS"])

# Add "fresh" water for temp compliance to slug.temp

pipe.temp <- use.data.temp[temp.time.index, "SL.temp"]

slug.temp <- rep(!(pipe.temp < temp.lo & pipe.temp > temp.hi),

flow.pipes.inst[t,"SL-WS"])

# Otherwise collect 'slug' from upstream pipes

} else{

# Create slug from upstream pipes

upstrpipes <- pipe.segments$names[pipe.segments$downstream == n]

upstrpipes <- upstrpipes[upstrpipes %in% names(flow.pipes.inst[t,][flow.pipes.inst[t,] > 0])]

for (u in upstrpipes) {

vol.start.up <- as.numeric(pipe.vol.used.up[,u]) + 1

vol.end.up <- (vol.start.up - 1) + min(flow.nodes.inst[t,n], flow.pipes.inst[t,u])

# Confirm No NAs added to slug

#if (any(is.na(time.flow[[u]]$en.time[vol.start.up:vol.end.up]))){

# stop("At least one 'NA' to be added to slug")}

slug <- c(slug, time.flow[[u]]$en.time[vol.start.up:vol.end.up])

slug.temp <- c(slug.temp, time.flow[[u]]$temp.comp[vol.start.up:vol.end.up])

pipe.vol.used.up[,u] <- pipe.vol.used.up[,u] + (vol.end.up - (vol.start.up-1))

}

}# else

# Deliver slug onto the appropriate downstream pipe segments

dnstrpipes <- pipe.segments$names[pipe.segments$upstream == n]

dnstrpipes <- dnstrpipes[dnstrpipes %in% names(flow.pipes.inst[t,][flow.pipes.inst[t,] > 0])]

if (length(dnstrpipes) > 0){

for (d in dnstrpipes) {

vol.start.dn <- as.numeric(pipe.vol.used.dn[,d]) + 1

vol.end.dn <- (vol.start.dn - 1) + min(flow.nodes.inst[t,n], flow.pipes.inst[t,d])

vol.diff <- vol.end.dn - (vol.start.dn - 1)

time.flow[[d]]$en.time[vol.start.dn:vol.end.dn] <- slug[1:(vol.diff)]

time.flow[[d]]$temp.comp[vol.start.dn:vol.end.dn] <- slug.temp[1:(vol.diff)]

# Add temperature assignment ----------------------------------------------

# Temperature of the pipe

temp.zone <- as.character(pipe.segments$temp\_zone[pipe.segments$names == d])

pipe.temp <- use.data.temp[temp.time.index, temp.zone]

# If waterjust entered the house write over all entries

if (d == "SL-WS"){

time.flow[[d]]$temp.comp[vol.start.dn:vol.end.dn] <- !(pipe.temp < temp.lo & pipe.temp > temp.hi)

# Otherwise write over only entries that need to be changed

} else{

# Mark as FALSE IFF temp.lo < temperature < temp.hi

time.flow[[d]]$temp.comp[vol.start.dn:vol.end.dn][pipe.temp > temp.lo & pipe.temp < temp.hi] <- FALSE

# Reset water with temp exceeding temp.reset

time.flow[[d]]$temp.comp[vol.start.dn:vol.end.dn][!time.flow[[d]]$temp.comp[vol.start.dn:vol.end.dn] & pipe.temp > temp.reset] <- TRUE

}

pipe.vol.used.dn[,d] <- pipe.vol.used.dn[,d] + (vol.end.dn - (vol.start.dn-1))

# Delete the portion of the slug that was consumed

slug <- slug[-(1:vol.diff)]

slug.temp <- slug.temp[-(1:vol.diff)]

# Confirm no NAs in pasted values

#if (any(is.na(time.flow[[dnstrpipes[d]]]$en.time[vol.start.dn:vol.end.dn]))){

# stop("Pasted slug contains at least one 'NA'")}

}# for downstream pipes

}#if

}#n

}#t

#save.image("/mnt/ufs18/home-037/julienry/Desktop/Water Temperature/Results/waterage\_temp\_output\_20201127.RData")

# Identify any NAs in output

c <- rep(0, length(time.flow))

for (p in 1:length(time.flow)) {

c[p] <- length(time.flow[[p]]$en.time[is.na(time.flow[[p]]$temp.comp)])

#if (length(time.flow[[p]]$en.time[is.na(time.flow[[p]]$en.time)]) > 0){

# stop(paste(length(time.flow[[p]]$en.time[is.na(time.flow[[p]]$en.time)]), "NAs found in pipe", pipes[p]))}

}

cbind(names(time.flow), c)

rm(c)

# Clean up after loop

rm(dnstrpipes, vol.end.dn, vol.start.dn, d,

upstrpipes, vol.end.up, vol.start.up, u,

slug, vol.diff, t, n, p,

slug.temp, temp.zone)

# Water age calculation ---------------------------------------------------

# Define the fixtures at which to calculate age

fixt.mon <- c("SL.1", "cold.kitchen.sink", "cold.bath2.sink",

"WH",

"hot.kitchen.sink", "hot.bath2.sink", "mix.bath2.shower")

# Define 'water.age' list to record water age for each monitored fixture ('fixt.mon')

water.age <- list()

for (f in fixt.mon){

water.age[[f]]$ex.time <- as.numeric(rep(NA, sum(flow.nodes.inst[, f]))) # time each unit water exits the building plumbing

water.age[[f]]$age <- as.numeric(rep(NA, sum(flow.nodes.inst[, f]))) # water age of each unit that exits the fixture

water.age[[f]]$temp.compliance <- as.logical(rep(NA, sum(flow.nodes.inst[, f]))) # water age of each unit that exits the fixture

}

for (f in fixt.mon){#length(fixt.mon)

# Determine the time that each unit water entered the plumbing

ex <- water.age[[f]]$ex.time

p <- pipe.segments$names[pipe.segments$downstream == f]

vol.end.ex <- (flow.pipes.cum[,p]) # end volume for entry time

vol.start.ex <- (vol.end.ex - flow.pipes.inst[,p] + 1) # start volume for entry time

for (t in 1:nrow(flow.nodes.cum)){

if (flow.pipes.inst[t,p] > 0){

ex[vol.start.ex[t]:vol.end.ex[t]] <- time.index[t]

}

}

# Determine exit time for each unit volume

water.age[[f]]$ex.time <- ex

# Determine age for each unit volume

water.age[[f]]$age <- water.age[[f]]$ex.time -

time.flow[[p]]$en.time[1:sum(flow.nodes.inst[, f])]

# Copy over temperature compliance

water.age[[f]]$temp.compliance <- time.flow[[p]]$temp.comp[1:sum(flow.nodes.inst[, f])]

rm(ex, vol.start.ex, vol.end.ex)

}

# Confirm no NAs in water age output

c <- rep(0, length(fixt.mon))

for (p in 1:length(fixt.mon)) {

c[p] <- length(water.age[[p]]$age[is.na(water.age[[p]]$age)])

#if (length(time.flow[[p]]$en.time[is.na(time.flow[[p]]$en.time)]) > 0){

# stop(paste(length(time.flow[[p]]$en.time[is.na(time.flow[[p]]$en.time)]), "NAs found in pipe", pipes[p]))}

}

cbind(fixt.mon, c)

rm(c, f, p, t,

flow.nodes.cum, flow.nodes.inst,

flow.pipes.cum, flow.pipes.inst,

pipe.vol.used.dn, pipe.vol.used.up)

# Prepare age dataframe for plotting --------------------------------------

library(data.table)

age <- data.frame(NULL)

for (f in 1:length(fixt.mon)) {

g <- as.data.table(water.age[[f]])

g$fixt <- fixt.mon[f]

age <- rbind(age, g)

rm(g)

}

rm(f)

age$ex.time <- age$ex.time + time.sim.start

# Set order for plotting

age$fixt <- factor(age$fixt,

levels = c("SL.1", "cold.kitchen.sink", "cold.bath2.sink", "WH", "hot.kitchen.sink", "hot.bath2.sink", "mix.bath2.shower"))

# Rename levels for plotting

#levels(age$fixt) <- c("SL", "Cold Kitchen Sink", "Cold Bath Sink", "WH", "Hot Kitchen Sink", "Hot Bath Sink", "Mix Shower")

# Save volume currently stored in pipes -----------------------------------

stored.in.pipes <- list()

for (p in pipes){

end.vol <- length(time.flow[[p]]$en.time)

start.vol <- (length(time.flow[[p]]$en.time) - pipe.segments$vol[pipe.segments$names == p]) + 1

stored.in.pipes[[p]]$en.time[1:pipe.segments$vol[pipe.segments$names == p]] <- time.flow[[p]]$en.time[start.vol:end.vol]

}

rm(time.flow)

# END OF LOOP -------------------------------------------------------------

setwd("/mnt/ufs18/home-037/julienry/Desktop/Water Temperature/Results/")

if (per == 1){

save.image("waterage\_output\_20201130\_(1)1-0.5M")

}else if (per == 2){

save.image("waterage\_output\_20201130\_(2)0.5M-1M")

} else if (per == 3){

save.image("waterage\_output\_20201130\_(3)1M-end")

}

timer <- Sys.time() - timer.start

timer

#rm(age, pipe.segments, water.age, delta.v, end.vol, start.vol, fixt.mon, nodes, p, pipes,time.index, time.sim.start, WH.avg.flow, WH.HRT, WH.vol, timer.start)

#rm(list=setdiff(ls(),"stored.in.pipes"))

#rm(list=setdiff(ls(),c("water.age.1", "water.age.2", "water.age.3")))