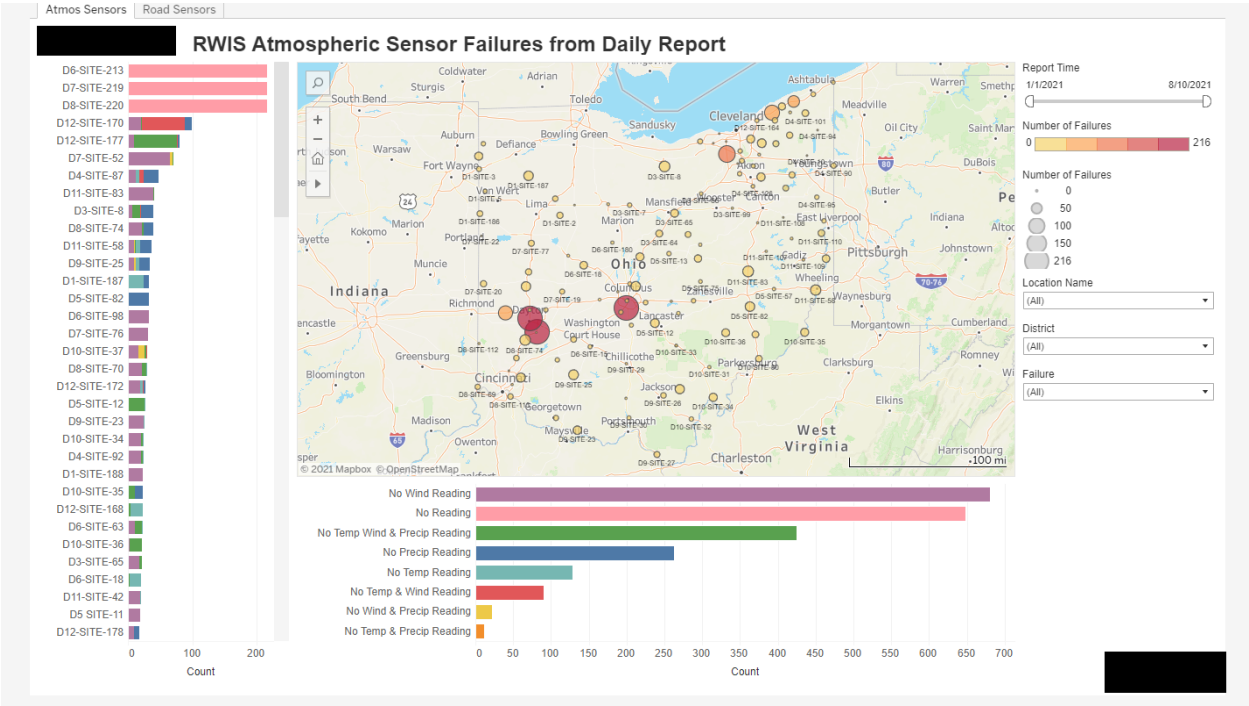
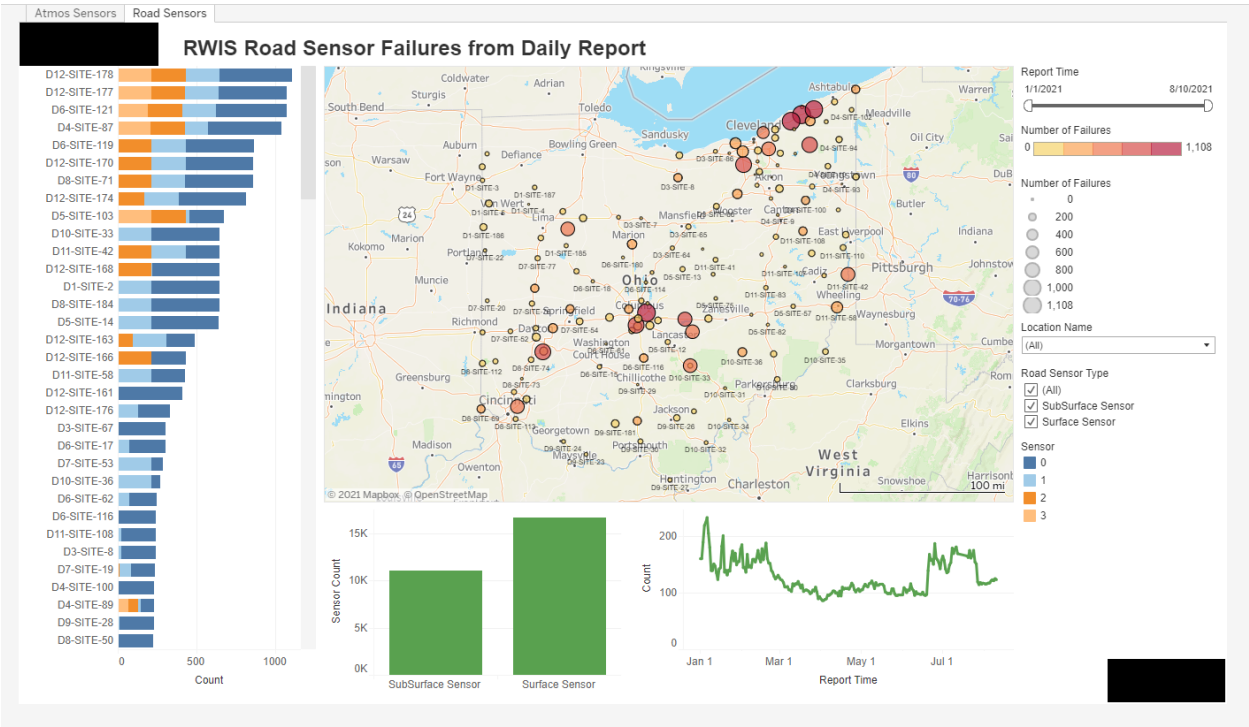
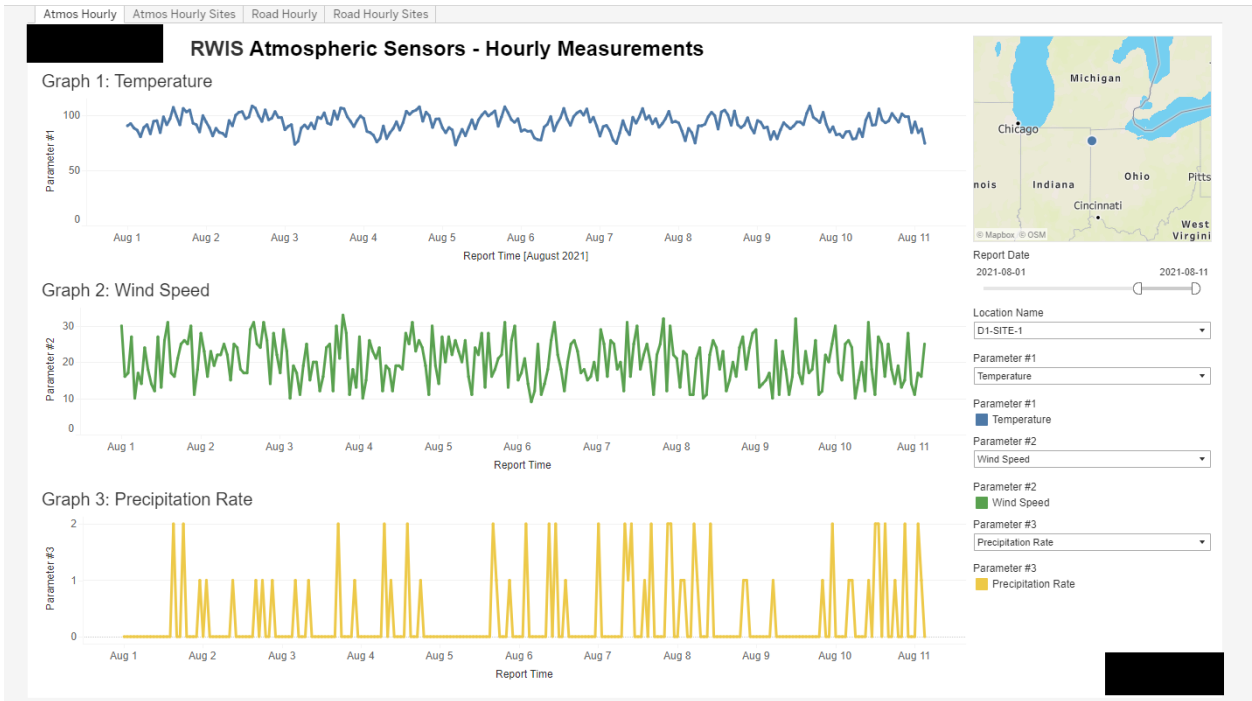
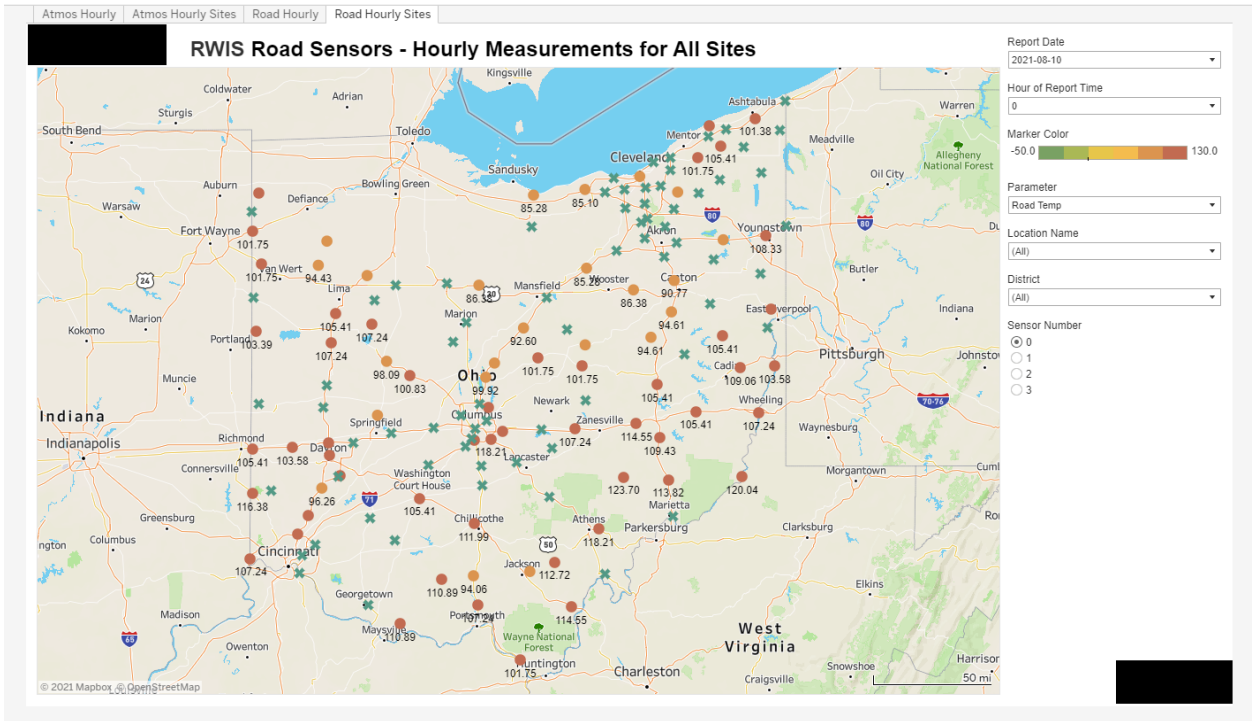
Kyle Vertin
Internship May 2021 - August 2021

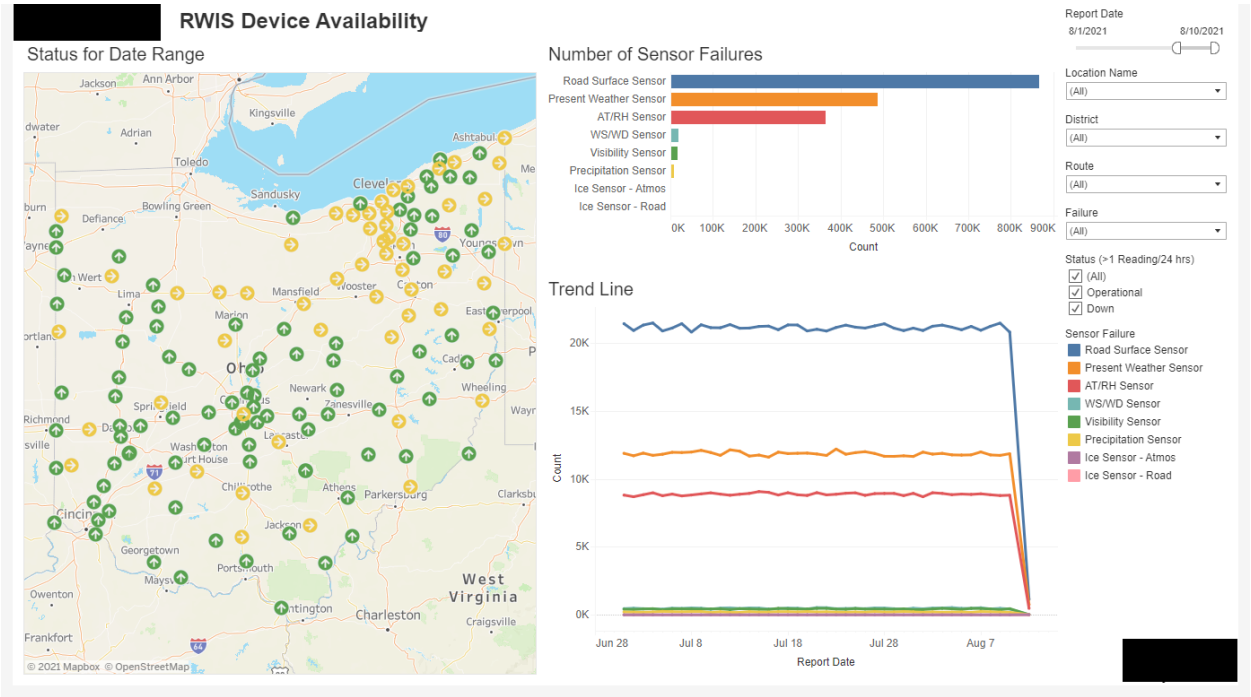
Dashboards:






Atmos Hourly
Atmos Hourly Sites
Road Hourly
Road Hourly Sites
RWIS Atmospheric Sensors - Hourly Measurements
Graph 1: Temperature
Parameter #1
Report Time [August 2021]
Graph 2: Wind Speed
Parameter #2
Report Time
Graph 3: Precipitation Rate
Parameter #3
Report Time
Aug 1
Aug 2
Aug 3
Aug 4
Aug 5
Aug 6
Aug 7
Aug 8
Aug 9
Aug 10
Aug 11
Michigan
Chicago
Indiana
Ohio
Cincinnati
Report Date
2021-08-01
2021-08-11
Location Name
D1-SITE-1
Parameter #1
Temperature
Parameter #2
Wind Speed
Parameter #3
Precipitation Rate


Atmos Hourly
Atmos Hourly Sites
Road Hourly
Road Hourly Sites
RWIS Road Sensors - Hourly Measurements for All Sites
Report Date
2021-08-10
Hour of Report Time
0
Marker Color
-50.0
130.0
Parameter
Road Temp
Location Name
(All)
District
(All)
Sensor Number
0
1
2
3
Cleveland
Pittsburgh
Cincinnati
Columbus
Indianapolis
Toledo
Akron
Youngstown
Canton
Mansfield
Dayton
Springfield
Zanesville
Parkersburg
Charleston
West Virginia
Huntington
Morgantown
Wheeling
Weirton
Marietta
Athens
Chillicothe
Lancaster
Newark
Portsmouth
Jackson
Wayne National Forest
Allegheny National Forest
© 2021 Mapbox © OpenStreetMap
50 mi



Scripts:

----------PARSE REPORT AND STORE IN DATABASE----------

```
library(lubridate)
library(stringr)
library(plyr)
library(dplyr)
library(openxlsx)
library(tools)

rm(list=ls())
Sys.setenv(TZ='America/New_York')

home <- "setwd('~/analytics/scripts')"
eval(parse(text = home))


setwd("../database")
r_failure <- read.csv("r_failure.csv", stringsAsFactors = FALSE, na.strings = c("NA",""))
r_failure <- subset(r_failure, select = c(failure_id, failure, failure_datasource))

t_location <- read.csv("t_location.csv", stringsAsFactors = FALSE, na.strings = c("NA",""))
```

```
#t_location$location <- t_location$location_number
t_location$location_number <- as.integer(gsub("^.*-", "", t_location$location_number))
t_location <- t_location[!is.na(t_location$location_number),]
t_location <- subset(t_location, select = c(location_id, location_description, location_number))


t_dailyatmos <- read.csv("t_dailyatmos.csv", stringsAsFactors = FALSE, na.strings = c("NA",""))
t_dailysensor <- read.csv("t_dailysensor.csv", stringsAsFactors = FALSE, na.strings =
c("NA",""))

      t_dailyatmos  <- join(t_dailyatmos,
r_failure[r_failure$failure_datasource=="daily_report_atmospheric",], by = "failure_id",) %>%
            join(.,t_location, by="location_id")

      if(sum(is.na(t_dailyatmos$failure)) > 0) { warning("daily_parse: The failure in t_dailyatmos
is NA") }
      if(sum(is.na(t_dailyatmos$location)) > 0) { warning("daily_parse: The location in
t_dailyatmos is NA") }

      t_dailyatmos <-  t_dailyatmos[,c("location_number", "failure", "location_description",
"report_time",
                        "dailyatmos_inserted", "dailyatmos_inserted_by", "dailyatmos_updated")]


      t_dailysensor  <- join(t_dailysensor,
r_failure[r_failure$failure_datasource=="daily_report_sensor",], by = "failure_id") %>%
            join(.,t_location, by="location_id")


      t_dailysensor <-  t_dailysensor[,c("location_number", "failure", "location_description",
"sensor", "sensor_name", "report_time",
                              "dailysensor_inserted", "dailysensor_inserted_by",
"dailysensor_updated")]

setwd("../input_files/daily_report")
input_files <- list.files(pattern = ".txt", all.files = TRUE)

      for(file in input_files) {

            #file <- input_files[1]
            rdr <- readLines(file)
            rdr <- data.frame(x = rdr, stringsAsFactors = FALSE)

            report_time <- str_sub(rdr$x[1], 43, nchar(rdr$x[1]))
```

```
        report_time <- gsub("\\.", "", report_time)

        report_time <- date(strptime(report_time, "%m/%d/%Y", tz = ""))


        newfile <- paste0("daily_report_", report_time, ".txt")

        if(!newfile %in% input_files) {
            file.rename(file, newfile)
        }
    }

    input_files <- sort(list.files(pattern = ".txt", all.files = TRUE))
    outage_flag <- 0
    sensor_flag <- 0

for(file in input_files) {

    #file <- input_files[1]
    rdr <- readLines(file)
    rdr <- data.frame(x = rdr, stringsAsFactors = FALSE)

    report_time <- str_sub(rdr$x[1], 43, nchar(rdr$x[1]))
    # gsub use regex, meaning regular expressions
    report_time <- gsub("\\.", "", report_time)

    # Put in UTC format
    report_time <- date(strptime(report_time, "%m/%d/%Y", tz = ""))
    # print(report_time)


    rdr$location_number <- NA
    rdr$location_description <- NA
    rdr$sensor <- NA
    rdr$sensor_name <- NA
    rdr$failure <-   NA
    rdr$report_time <- paste0(format(report_time, "%Y-%m-%d"), " 10:00:00")

    rdr$x <- gsub("^ *$", "", rdr$x)
    rdr <- rdr[rdr$x != "",]

    row.names(rdr) <- NULL

    if(sum(t_dailyatmos$report_time == report_time) == 0) {
```

```
        outage_flag <- 1

        start <- which(grepl("RWIS ATMOSPHERIC REPORT", rdr$x))
        end <-  which(grepl("RWIS SURFACE SENSOR REPORT", rdr$x))
        t1 <- rdr[start:(end-1),]

        t1 <- as.data.frame(t1)

        t1 <- t1[grepl("^[0-9]", t1$x),]
        t1$report_name <- "RWIS ATMOSPHERIC REPORT"

        t1$dailyatmos_inserted <- t1$report_time
        t1$dailyatmos_inserted_by <- "daily_parse"
        t1$dailyatmos_updated <- t1$report_time

        d_start <- c(1,11,71)
        d_end <- c(10,70,255)

        for(i in 1:nrow(t1)) {
                t1$location_number[i] <- substr(t1$x[i],d_start[1],d_end[1])
                t1$location_description[i] <- substr(t1$x[i],d_start[2],d_end[2])
                t1$failure[i] <- substr(t1$x[i],d_start[3],d_end[3])
        }

        t1$failure <- gsub("^.*No", "No", t1$x)
        t1$location_number <- as.integer(t1$location_number)
        t1$location_description <- gsub(" *$", "", t1$location_description)
        t1$failure <- gsub(" *$", "", t1$failure)

        t1 <- subset(t1, select = -c(x, sensor, sensor_name, report_name))

        if(nrow(t1) > 0) {
                # Column names must be identical for rbind
                t_dailyatmos <- rbind(t_dailyatmos,t1)
        }
}

if(sum(t_dailysensor$report_time == report_time) == 0) {

        sensor_flag <- 1
        # Middle
        start <- which(grepl("RWIS SURFACE SENSOR REPORT", rdr$x))
```

```
end <-  which(grepl("RWIS SUB SENSOR REPORT", rdr$x))
t2 <- rdr[start:(end-1),]
t2 <- t2[grepl("^[0-9]", t2$x),]
t2$report_name <- "RWIS SURFACE SENSOR REPORT"

t2$dailysensor_inserted <- t2$report_time
t2$dailysensor_inserted_by <- "daily_parse"
t2$dailysensor_updated <- t2$report_time


d_start <- c(1,11,71,81,131)
d_end <- c(10,70,80,130,255)

for(i in 1:nrow(t2)) {
	t2$location_number[i] <- substr(t2$x[i],d_start[1],d_end[1])
	t2$location_description[i] <- substr(t2$x[i],d_start[2],d_end[2])
	t2$sensor[i] <- substr(t2$x[i],d_start[3],d_end[3])
	t2$sensor_name[i] <- substr(t2$x[i],d_start[4],d_end[4])
	t2$failure[i] <- substr(t2$x[i],d_start[5],d_end[5])
}

t2$location_number <- as.integer(t2$location_number)
t2$sensor <- as.integer(t2$sensor)
t2$location_description <- gsub(" *$", "", t2$location_description)
t2$sensor_name <- gsub(" *$", "", t2$sensor_name)
t2$failure <- gsub(" *$", "", t2$failure)

start <- which(grepl("RWIS SUB SENSOR REPORT", rdr$x))
end <-  nrow(rdr)
t3 <- rdr[start:end,]
t3 <- t3[grepl("^[0-9]", t3$x),]
t3$report_name <- "RWIS SURFACE SENSOR REPORT"

t3$dailysensor_inserted <- t3$report_time
t3$dailysensor_inserted_by <- "daily_parse"
t3$dailysensor_updated <- t3$report_time


d_start <- c(1,11,71,81,131)
d_end <- c(10,70,80,130,255)

for(i in 1:nrow(t3)) {
	t3$location_number[i] <- substr(t3$x[i],d_start[1],d_end[1])
	t3$location_description[i] <- substr(t3$x[i],d_start[2],d_end[2])
```

```
            t3$sensor[i] <- substr(t3$x[i],d_start[3],d_end[3])
            t3$sensor_name[i] <- substr(t3$x[i],d_start[4],d_end[4])
            t3$failure[i] <- substr(t3$x[i],d_start[5],d_end[5])
        }

        t3$location_number <- as.integer(t3$location_number)
        t3$sensor <- as.integer(t3$sensor)
        t3$location_description <- gsub(" *$", "", t3$location_description)
        t3$sensor_name <- gsub(" *$", "", t3$sensor_name)
        t3$failure <- gsub(" *$", "", t3$failure)

        t2 <- subset(t2, select = -c(x,report_name))
        t3 <- subset(t3, select = -c(x,report_name))

        if(nrow(t2) > 0 | nrow(t3) > 0 ) {
            # Column names must be identical for rbind
            t_dailysensor <- rbind(t_dailysensor, rbind(t2,t3))
    }
    }
    }

    t_dailyatmos <- join(t_dailyatmos, t_location, by = "location_number") %>%
        join(., r_failure[r_failure$failure_datasource=="daily_report_atmospheric",], by =
"failure")

    if(sum(is.na(t_dailyatmos$failure_id)) > 0) {
        warning("daily_parse: Daily outage failure is not in r_failure table, add it")
        print("daily_parse: Daily outage failure is not in r_failure table, add it")
        print(sort(unique(t_dailyatmos$failure[is.na(t_dailyatmos$failure_id)])))
    }

    t_dailyatmos <- t_dailyatmos[!is.na(t_dailyatmos$location_id),]

    t_dailyatmos$dailyatmos_id <- 1:nrow(t_dailyatmos)

    t_dailyatmos <-
t_dailyatmos[,c("dailyatmos_id","location_id","failure_id","location_description", "report_time",

"dailyatmos_inserted","dailyatmos_inserted_by","dailyatmos_updated")]


    t_dailysensor <- join(t_dailysensor, t_location, by = "location_number") %>%
        join(., r_failure[r_failure$failure_datasource=="daily_report_sensor",], by = "failure")
```

```
	t_dailysensor <- t_dailysensor[!is.na(t_dailysensor$location_id),]


	t_dailysensor$dailysensor_id <- 1:nrow(t_dailysensor)


	t_dailysensor <-
t_dailysensor[,c("dailysensor_id","location_id","failure_id","location_description",
"sensor","sensor_name","report_time",

"dailysensor_inserted","dailysensor_inserted_by","dailysensor_updated")]


	setwd("../../database")

	if(outage_flag == 1) {
		write.table(t_dailyatmos, "t_dailyatmos.csv", row.names = FALSE, sep=",")
	}

	if(sensor_flag == 1) {
		write.table(t_dailysensor, "t_dailysensor.csv", row.names = FALSE, sep=",")
	}
```

----------SCRIPT TO READ API DATA AND INSERT INTO DATABASE----------

```
suppressMessages(library(lubridate))
suppressMessages(library(plyr))
suppressMessages(library(dplyr))
suppressMessages(library(stringr))
suppressMessages(library(openxlsx))
suppressMessages(library(tools))
suppressMessages(library(httr))
suppressMessages(library(jsonlite))
suppressMessages(library(digest))



	# API
```

```
# https://api.com/random/random/exampletext/

# How to HTTP in R
# https://medium.com/@traffordDataLab/querying-apis-in-r-39029b73d5f1`


#path <- "https://api.com/random/random/exampletext/"

# atmospheric
# path <- "https://api.com/random/random/exampletext/"

# Sensors
#path <- "https://api.com/random/random/exampletext/"

    # API Connection info
    #myquery$failure_code
    #myquery$request

rm(list = ls())
options(max.print=10000)
options(digits=6)
Sys.setenv(TZ='America/New_York')

# Staging

    # OPTIONS FOR TESTING
    # These options are for testing and are not going to be in the production code
    # Option 1: Make input database tables t_atmos, t_atmos_item, t_sesnor, t_sensor_item
    # blank with header only if empty_tables == 1, leave as-is if == 0
    empty_tables <- 0
    # Option 2: Use simulated report_time dates if simulation == 1, do not simulate if simulation
== 0
    simulation <- 1


    tday <- Sys.time()
    attr(tday,"tzone") <- "America/Denver"
    tdayf2 <- format(tday, "%y%m%d_%H%M%S")
    attr(tday,"tzone") <- "UTC"
    tdayf <- format(tday, "%Y-%m-%d %H:%M:%S")


    # Choose one type of query, and one site.  Site is "OH001" in the URL.
    home <- "setwd('~/analytics/scripts')"
```

```
	eval(parse(text = home))

	setwd("../database")
	site_cref <- read.csv("r_site_crossref.csv", stringsAsFactors = FALSE, na.strings =
c("NA",""))
	setwd("../input_files")

	setwd("../database")

	t_location <- read.csv("t_location.csv", stringsAsFactors = FALSE, na.strings = c("NA",""))
	t_location <- subset(t_location, select = c(location_id, ws_location_name,
ws_location_number))

	r_failure <- read.csv("r_failure.csv", stringsAsFactors = FALSE, na.strings = c("NA",""))
	r_failure <- subset(r_failure, select = c(failure_id, failure, failure_datasource))

	t_atmos <- read.csv("t_atmos.csv", stringsAsFactors = FALSE, na.strings = c("NA",""))
	t_atmos_item <- read.csv("t_atmos_item.csv", stringsAsFactors = FALSE, na.strings =
c("NA",""))
	t_sensor <- read.csv("t_sensor.csv", stringsAsFactors = FALSE, na.strings = c("NA",""))
	t_sensor_item <- read.csv("t_sensor_item.csv", stringsAsFactors = FALSE, na.strings =
c("NA",""))

	# Option for testing
	if(empty_tables == 1) {
		t_atmos <- t_atmos[0,]
		t_atmos_item <- t_atmos_item[0,]
		t_sensor <- t_sensor[0,]
		t_sensor_item <- t_sensor_item[0,]
	}

	# Options for testing - simulate the report_time
	# This applies only to data already in t_atmos and t_sensor
	# Scramble up the dates
	if(simulation == 1 & nrow(t_atmos) > 0) {
		# Date simulation is going to go in the ws_collect script
		adt <- as.POSIXct(paste0(date(tdayf), "06:00:00"))
		attr(adt,"tzone") <- "UTC"
		for(k in 1:30) {
			adt <- c(adt, adt[1] - days(k))
		}
		adt <- format(adt,"%Y-%m-%d %H:%M:%S")
		t_atmos$report_time <- sample(adt, nrow(t_atmos),replace = TRUE)
		t_sensor$report_time <- sample(adt, nrow(t_sensor),replace = TRUE)
```

```
	}


	atmos_cols <-
c("location_id","atmos_item_set_id","report_time","temperature","dew_point","relative_humidity",
"wind_speed","wind_direction","wind_gust",

"visibility","precipitation_rate","atmos_inserted","atmos_inserted_by","atmos_updated")

	t_atmos <- t_atmos[,atmos_cols]

	atmos_item_cols <- c("atmos_item_set_id","failure_id","count","atmos_item_inserted",
			"atmos_item_inserted_by","atmos_item_updated")

	t_atmos_item <- t_atmos_item[,atmos_item_cols]




# Get Atmospheric Measurements from API query for t_atmos

	# df has all data for all sites
	atmos <- NULL
	# failure has failure data for all sites
	failure <- NULL
	# StationId
	i <- 0

	for(j in 1:250) {
		i <- i + 1
		print(i)

		# Skip if the site is not in t_location
		if(i %in% t_location$ws_location_number) {

		a <- str_pad(as.character(i), 3, "left", pad = "0")
		path <- paste0("example", a, "/example")
		myquery <-  GET(url = path)
		output <- httr::content(myquery, as = "text")
		z <- fromJSON(output)

		if("stationId" %in% names(z)) {
			if("qcFailures" %in% names(z) & sum(!is.na(z$qcFailures)) > 0) {
				temp <- bind_rows(z$qcFailures)
```

```
                temp$StationId <- paste0("OH",a)

                                        if(is.null(failure)){
                        failure <- temp
                }else {
                        failure <- rbind(failure, temp)
                }
        }

                                z$precipitation_rate <- z$precipitation$rate

        keepcols <- c("stationId",
                        #"latitude",
                        #"longitude",
                        "utcTime",
                        "temperature",
                        "dewPoint",
                        #"wetBulbTemp",
                        "relativeHumidity",
                        "windSpeed",
                        "windDirection",
                        "windGust",
                        "visibility",
                        "precipitation_rate"
                        #"pressureSeaLevel"
        )

        z <- z[nrow(z), keepcols]

        if(is.null(atmos)){
                atmos <- z
                report_time <- as.POSIXct(strptime(z$utcTime[nrow(z)],
"%Y-%m-%dT%H:%M:%SZ"), tz = "UTC")
                print(paste0("Time of the WeatherSentry API Query in UTC = ",
report_time))
        }
        else {
                atmos <- bind_rows(atmos, z)
        }
    }else {

        temp <- data.frame(attribute = "Not Reporting", arrayIndex = NA, causes = "",
                        StationId = paste0("OH",a))
        if(is.null(failure)){
```

```
                failure <- temp
            }else {
                failure <- rbind(failure, temp)
            }
        }
        }
    }

      atmos$atmos_item_set_id <- 0
    atmos <- subset(atmos, select = -c(utcTime))
    report_time <- format(report_time,"%Y-%m-%d %H:%M:%S")

    atmos$report_time <- report_time
    atmos$atmos_inserted <- tdayf
    atmos$atmos_inserted_by <- "ws_collect"
    atmos$atmos_updated <- tdayf

    names(atmos) <- c(
        "ws_location_name",
        "temperature",
        "dew_point",
        "relative_humidity",
        "wind_speed",
        "wind_direction",
        "wind_gust",
        "visibility",
        "precipitation_rate",
        "atmos_item_set_id",
        "report_time",
        "atmos_inserted",
        "atmos_inserted_by",
        "atmos_updated"
    )
    sort(unique(failure$attribute))
       failure_all_col <- c("temperature","windSpeed","windDirection","windGusts",
"wetBulbTemp",
                    "dewPoint","precipitation.accumulation.6H","precipitation.accumulation.24H",
                    "precipitation.precipDetected","precipitation.rate","relativeHumidity",
"visibility",
                    "precipitation.accumulation.1H","precipitation.accumulation.3H",
                    "precipitation.accumulation.12H", "Not Reporting")

        if(sum(!failure$attribute %in% failure_all_col) > 0) {
          print("Need to add a failure attribute to the df data frame: ")
```

```
      print(unique(failure$attribute[!failure$attribute %in% failure_all_col]))
}

failure <- failure[failure$attribute %in% failure_all_col,c("attribute","StationId")]

  failure$num <- 1
fagg <- aggregate(num ~ attribute + StationId, data = failure, sum)
fagg$attribute <- gsub("\\.","_",fagg$attribute)

# Names in schema
names(fagg) <- c("failure","ws_location_name","count")
fagg$atmos_item_set_id <- 0
fagg$atmos_item_inserted <- tdayf
fagg$atmos_item_inserted_by <- "ws_collect"
fagg$atmos_item_updated <- tdayf

      if(nrow(t_atmos) == 0) {
       index <- 0
}else {
       index <- max(t_atmos$atmos_item_set_id)
}

sites <- unique(atmos$ws_location_name)

   for(i in sites) {
                  if(i %in% fagg$ws_location_name) {
              index <- index + 1
              atmos$atmos_item_set_id[atmos$ws_location_name == i] <- index
              fagg$atmos_item_set_id[fagg$ws_location_name == i] <- index
       }
}




atmos <- join(atmos, t_location, by = "ws_location_name")
atmos <- atmos[,atmos_cols]

t_atmos <- rbind(t_atmos, atmos)

     t_atmos$atmos_id <- 1:nrow(t_atmos)
t_atmos <- t_atmos[,c("atmos_id",atmos_cols)]
```

```
	write.table(t_atmos, "t_atmos.csv", row.names = FALSE, sep=",")

	fagg <- join(fagg, r_failure, by="failure")
	fagg <- fagg[,atmos_item_cols]

	t_atmos_item <- rbind(t_atmos_item, fagg)

	t_atmos_item$atmos_item_id <- 1:nrow(t_atmos_item)
	t_atmos_item <- t_atmos_item[,c("atmos_item_id",atmos_item_cols)]

	write.table(t_atmos_item, "t_atmos_item.csv", row.names = FALSE, sep=",")




	sensor_cols <- c("location_id","sensor_item_set_id","report_time","sensor_number",
			"sensor_name",
			"road_temp",
			"bridge_temp",
			"freeze_temp",
			"pavement_sensor_obs_error",
			"subsurface_temperature",
			"sensor_inserted","sensor_inserted_by","sensor_updated")

	t_sensor <- t_sensor[,sensor_cols]

	sensor_item_cols <- c("sensor_item_set_id","failure_id","count","sensor_item_inserted",
				"sensor_item_inserted_by","sensor_item_updated")

	t_sensor_item <- t_sensor_item[,sensor_item_cols]




sensor <- NULL

failure <- NULL

i <- 0

	for(j in 1:25) {
		i <- i + 1
		print(i)
```

```
                if(i %in% t_location$ws_location_number) {

        a <- str_pad(as.character(i), 3, "left", pad = "0")
        path <- paste0("example", a, "/example")

        myquery <-  GET(url = path)
        output <- httr::content(myquery, as = "text")
        z2 <- fromJSON(output)

        if("stationId" %in% names(z2)) {

                        if("qcFailures" %in% names(z2) & sum(!is.na(z2$qcFailures)) > 0) {

                temp <- NULL

                for(k in 1:nrow(z2)) {
                        if(is.null(z2$qcFailures[[k]]$attribute)) {
                                fail <- data.frame(attribute = NA,
                                                StationId = paste0("OH",a),
                                                sensorId = z2$sensorId[k], stringsAsFactors =
FALSE)
                        }else {
                                fail <- data.frame(attribute = z2$qcFailures[[k]]$attribute,
                                                StationId = paste0("OH",a),
                                                sensorId = z2$sensorId[k], stringsAsFactors =
FALSE)
                        }
                        if(is.null(temp)) {
                                temp <- fail
                        }else {
                                temp <- rbind(temp,fail)
                        }

                }

                                        if(is.null(failure)){
                        failure <- temp
                }else {
                        failure <- rbind(failure, temp)
                }
        }

        if("subsurfaceTemps" %in% names(z2) & sum(!is.na(z2$subsurfaceTemps)) > 0
) {
```

```
                    sstemp <- NULL

                    for(x in 1:nrow(z2)) {
                        if(is.null(z2$subsurfaceTemps[[x]]$temperature)) {
                            ssvec <- NA
                        }else {

if(length(z2$subsurfaceTemps[[x]]$temperature) > 1) {
                                print(paste0("WS API has returned more than 1 temperature
per sensor: OH", a))
                            }
                            ssvec <- z2$subsurfaceTemps[[x]]$temperature[1]
                        }
                        if(is.null(sstemp)) {
                            sstemp <- ssvec
                        }else {
                            sstemp <- c(sstemp, ssvec)
                        }
                    }
                    z2$subsurfaceTemps <- sstemp
                } else {
                    z2$subsurfaceTemps <- NA
                }

                z2 <- z2[z2$utcTime == max(z2$utcTime),
                         c("stationId",
                         #"latitude",
                         #"longitude",
                         "utcTime",
                         #"precipitation" this is an embedded dataframe may need to
process later
                         "sensorId",
                         "sensorName",
                         #"surfaceCondition",
                         #"mobileFriction",
                         "roadTemp",
                         "bridgeTemp",
                         "freezeTemp",
                         #"chemicalPercent",
                         #"chemicalFactor",
                         #"waterLevel",
                         #"icePercent",
                         #"conductivity",
```

```
                        #"salinity",
                        "subsurfaceTemps", # We may need to add this back in later
                        "pavementSensorObservationError"
                )]

                if(is.null(sensor)){
                        sensor <- z2
                        report_time <- as.POSIXct(strptime(z2$utcTime[nrow(z2)],
"%Y-%m-%dT%H:%M:%SZ"), tz = "UTC")
                        attr(report_time,"tzone") <- "America/New_York"
                        print(paste0("Time of the WeatherSentry API Query in Eastern Time = ",
report_time))
                }
                else {
                        sensor <- bind_rows(sensor, z2)
                }
        }
        }
    }



    sensor$sensor_item_set_id <- 0
    sensor <- subset(sensor, select = -c(utcTime))
    report_time <- format(report_time,"%Y-%m-%d %H:%M:%S")

    sensor$report_time <- report_time
    sensor$sensor_inserted <- tdayf
    sensor$sensor_inserted_by <- "ws_collect"
    sensor$sensor_updated <- tdayf

    names(sensor) <- c(
        "ws_location_name",
        "sensor_number",
        "sensor_name",
        #"surface_condition",
        #"mobile_friction",
        "road_temp",
        "bridge_temp",
        "freeze_temp",
        "subsurface_temperature",
        "pavement_sensor_obs_error",
        "sensor_item_set_id",
        "report_time",
```

```
        "sensor_inserted",
        "sensor_inserted_by",
        "sensor_updated"
)


sort(unique(failure$attribute))


if(sum(!failure$attribute %in% failure_all_col) > 0) {
        print("Need to add a failure attribute to the df2 data frame: ")
        print(unique(failure$attribute[!failure$attribute %in% failure_all_col]))
}

failure <- failure[failure$attribute %in% failure_all_col,c("attribute","StationId", "sensorId")]

      failure$num <- 1
fagg <- aggregate(num ~ attribute + StationId, data = failure, sum)
fagg$attribute <- gsub("\\.","_",fagg$attribute)

       names(fagg) <- c("failure","ws_location_name","count")
fagg$sensor_item_set_id <- 0
fagg$sensor_item_inserted <- tdayf
fagg$sensor_item_inserted_by <- "ws_collect"
fagg$sensor_item_updated <- tdayf

      if(nrow(t_sensor) == 0) {
        index <- 0
}else {
        index <- max(t_sensor$sensor_item_set_id)
}

sites <- unique(sensor$ws_location_name)

      for(i in sites) {
                  if(i %in% fagg$ws_location_name) {
              index <- index + 1
              sensor$sensor_item_set_id[sensor$ws_location_name == i] <- index
              fagg$sensor_item_set_id[fagg$ws_location_name == i] <- index
       }
}
```

```
sensor <- join(sensor, t_location, by = "ws_location_name")
sensor <- sensor[,sensor_cols]

t_sensor <- rbind(t_sensor, sensor)


t_sensor$sensor_id <- 1:nrow(t_sensor)
t_sensor <- t_sensor[,c("sensor_id",sensor_cols)]

write.table(t_sensor, "t_sensor.csv", row.names = FALSE, sep=",")


fagg <- join(fagg, r_failure, by="failure")
fagg <- fagg[,sensor_item_cols]

t_sensor_item <- rbind(t_sensor_item, fagg)
      t_sensor_item$sensor_item_id <- 1:nrow(t_sensor_item)
t_sensor_item <- t_sensor_item[,c("sensor_item_id",sensor_item_cols)]

write.table(t_sensor_item, "t_sensor_item.csv", row.names = FALSE, sep=",")
```

-----SCRIPT TO READ API DATA AND INSERT INTO DATABASE WITH AGGREGATIONS----

```
suppressMessages(library(lubridate))
suppressMessages(library(plyr))
suppressMessages(library(dplyr))
suppressMessages(library(stringr))
suppressMessages(library(openxlsx))
suppressMessages(library(tools))
suppressMessages(library(httr))
suppressMessages(library(jsonlite))
suppressMessages(library(digest))

#suppressMessages(library(RMariaDB))
#suppressMessages(library(rsyslog))



# API
# https://api.com/random/random/exampletext/

# How to HTTP in R
```

```
# https://medium.com/@traffordDataLab/querying-apis-in-r-39029b73d5f1`


#path <- "https://api.com/random/random/exampletext/"

# atmospheric
# path <- "https://api.com/random/random/exampletext/"

# Sensors
#path <- "https://api.com/random/random/exampletext/"

# API Connection info
#myquery$failure_code
#myquery$request

rm(list = ls())
options(max.print=10000)
options(digits=6)
Sys.setenv(TZ='America/New_York')


      home <- "setwd('~/analytics/scripts')"
      eval(parse(text = home))



      tday <- Sys.time()
      attr(tday,"tzone") <- "America/Denver"
      tdayf2 <- format(tday, "%y%m%d_%H%M%S")
      attr(tday,"tzone") <- "UTC"
      tdayf <- format(tday, "%Y-%m-%d %H:%M:%S")

setwd("../database")
site_cref <- read.csv("r_site_crossref.csv", stringsAsFactors = FALSE, na.strings = c("NA",""))
setwd("../input_files")

setwd("../database")

t_atmos <- read.csv("t_atmos.csv", stringsAsFactors = FALSE, na.strings = c("NA",""))
t_atmos_item <- read.csv("t_atmos_item.csv", stringsAsFactors = FALSE, na.strings =
c("NA",""))
t_sensor <- read.csv("t_sensor.csv", stringsAsFactors = FALSE, na.strings = c("NA",""))
t_sensor_item <- read.csv("t_sensor_item.csv", stringsAsFactors = FALSE, na.strings =
c("NA",""))
```

```
t_availability <- read.csv("t_availability.csv", stringsAsFactors = FALSE, na.strings = c("NA",""))

r_failure <- read.csv("r_failure.csv", stringsAsFactors = FALSE, na.strings = c("NA",""))
unneeded <-
c("precipitation_accumulation_3H","precipitation_accumulation_6H","precipitation_accumulation
_12H",
            "salinity","surfaceCondition")
r_failure <- r_failure[!r_failure$failure %in% unneeded,]
r_failure_atmos <- r_failure$failure_id[r_failure$failure_datasource == "api_atmospheric"]
r_failure_sensor <- r_failure$failure_id[r_failure$failure_datasource == "api_sensor"]

t_location <- read.csv("t_location.csv", stringsAsFactors = FALSE, na.strings = c("NA",""))
t_location_vec <- t_location$location_id[t_location$active == 1]

t_asset <- read.csv("t_asset.csv", stringsAsFactors = FALSE, na.strings = c("NA",""))
t_asset <- t_asset[,c("asset_status_id","location_id")]


t_availability_cols <- c("location_id",
              "failure_id",
              "asset_status_id",
              "report_date",
              "count",
              "availability_inserted",
              "availability_inserted_by",
              "availability_updated")

t_availability <- t_availability[,t_availability_cols]

      alldates <- as.character(unique(date(t_atmos$report_time)))
      newdates <- alldates[!alldates %in% t_availability$report_date]



      if(length(newdates) > 0) {

            t_atmos <- t_atmos[as.character(date(t_atmos$report_time)) %in% newdates, ]

                  t_atmos_item <- join(t_atmos, t_atmos_item, by = "atmos_item_set_id")

            t_atmos_item$report_time <-  as.POSIXct(t_atmos_item$report_time, tz = "UTC")
            attr(t_atmos_item$report_time, "tzone") <- "America/New_York"
```

```
        t_atmos_item$report_date <- date(t_atmos_item$report_time)


        aagg <- aggregate(count ~ location_id + failure_id + report_date, t_atmos_item, sum)

        aagg$report_date <- format(aagg$report_date, "%Y-%m-%d")

        df <- expand.grid(location_id = t_location_vec, failure_id = r_failure_atmos,
report_date = newdates)

        aagg <- join(df, aagg, by = c("location_id", "failure_id", "report_date"))
        aagg$count[is.na(aagg$count)] <- 0


        aagg <- join(aagg, t_asset, by = "location_id")

        aagg$availability_inserted <- tdayf
        aagg$availability_inserted_by <- "ws_aggregate"
        aagg$availability_updated <- tdayf
        names(aagg)[names(aagg) == "count"] <- "count"

        aagg <- aagg[,t_availability_cols]

                t_availability <- rbind(t_availability, aagg)
    }


    if(length(newdates) > 0) {




                t_sensor_item <- join(t_sensor, t_sensor_item, by = "sensor_item_set_id")

        t_sensor_item$report_time <-  as.POSIXct(t_sensor_item$report_time, tz = "UTC")
        attr(t_sensor_item$report_time, "tzone") <- "America/New_York"

        t_sensor_item$report_date <- date(t_sensor_item$report_time)

        sagg <- aggregate(count ~ location_id + failure_id + report_date, t_sensor_item, sum)

        sagg$report_date <- format(sagg$report_date, "%Y-%m-%d")
```

```
        df <- expand.grid(location_id = t_location_vec, failure_id = r_failure_sensor,
report_date = newdates)

        sagg <- join(df, sagg, by=c("location_id", "failure_id", "report_date"))
        sagg$count[is.na(sagg$count)] <- 0



        sagg <- join(sagg, t_asset, by = "location_id")

        sagg$availability_inserted <- tdayf
        sagg$availability_inserted_by <- "ws_aggregate"
        sagg$availability_updated <- tdayf
        names(sagg)[names(sagg) == "count"] <- "count"

        sagg <- sagg[,t_availability_cols]

        # now check that you match the schema exactly before doing rbind (t_availability)
        t_availability <- rbind(t_availability, sagg)
}

       t_availability$availability_id <- 1:nrow(t_availability)
    t_availability_cols <- c("availability_id",t_availability_cols)
    t_availability <- t_availability[,t_availability_cols]

    t_availability$availability_id <- 1:nrow(t_availability)
    t_availability <- t_availability[,c("availability_id",t_availability_cols)]

    write.table(t_availability, "t_availability.csv", row.names = FALSE, sep=",")

    t_atmos <- read.csv("t_atmos.csv", stringsAsFactors = FALSE, na.strings = c("NA",""))
    t_atmos_item <- read.csv("t_atmos_item.csv", stringsAsFactors = FALSE, na.strings =
c("NA",""))
    t_sensor <- read.csv("t_sensor.csv", stringsAsFactors = FALSE, na.strings = c("NA",""))
    t_sensor_item <- read.csv("t_sensor_item.csv", stringsAsFactors = FALSE, na.strings =
c("NA",""))

    retention <- Sys.Date() - 10

    retain_keys  <- t_atmos$atmos_item_set_id[date(t_atmos$report_time) >= retention &
t_atmos$atmos_item_set_id != 0]
    t_atmos <- t_atmos[date(t_atmos$report_time) >= retention, ]
    t_atmos_item <- t_atmos_item[t_atmos_item$atmos_item_set_id %in% retain_keys,]
```

```
        retain_keys  <- t_sensor$sensor_item_set_id[date(t_sensor$report_time) >= retention &
t_sensor$sensor_item_set_id != 0]
        t_sensor <- t_sensor[date(t_sensor$report_time) >= retention, ]
        t_sensor_item <- t_sensor_item[t_sensor_item$sensor_item_set_id %in% retain_keys,]
```