Wailupe Stormwater Network

This is code to process the stormwater network for use in SWMM.

Erica Johnson

May 7, 2020

Code Setup

Read libraries and data file here

```
#Libraries
library(tidyverse)
library(dplyr)
library(data.table)
library(geosphere)
library(janitor)



#Data with clean names
data <- read_csv("sw_network_endpoints_wailupe.csv") %>% clean_names()
vertices_dt <- read_csv("sw_network_allpoints_wailupe.csv") %>% clean_names()
```

Data Tidying

Select columns with the data we need to use. Rename them for easy reference.

```
network <- data %>%
  select(
  objectid,
  point_x,
  point_y,
  elevation,
  subcatch_r,
  roughness,
  type,
  diameter,
  width,
  height,
  type_1
  ) %>%
  rename(
    name = objectid,
    x = point_x,
    y = point_y,
    subc = subcatch_r,
    elevation = elevation,
    shape = type,
```

```r
    structure = type_1
    ) %>%
  mutate (shape = str_replace_all(shape, "Reinforced Concrete Pipe", "CIRCULAR")) %>%
  mutate (shape = str_replace_all(shape, "Box Culvert", "RECT_CLOSED")) %>%
  mutate (shape = str_replace_all(shape, "Channel", "TRAPEZOIDAL")) %>%
  mutate (shape = str_replace_all(shape, "Ditch", "RECT_OPEN")) %>%
  mutate (shape = str_replace_all(shape, "Other", "RECT_OPEN")) %>%
  distinct()
```

"Length" provided by USGS is distance between xy points. This "length"" is not the actual length of the conduit because it does not take into consideration height (xyz), so we will use the difference between lower distance between xy points calcualte the actual length further down in the code.

We will also re-calculate distance between xy points because different sources return different values for some of the conduits and some conduits need to have this distance calculated anyway because it is blank.

Step 1 - Create unique names for nodes with the same x and y coordinates, and unique names for conduits

```r
#Index xy coordinates with unique IDs if different, same IDs if repeated
unique <- network %>%
  mutate(
    node = group_indices(
      network, x, y
      )
    )

#Assign nodes the letter j for "junction" (SWMM terminology) and conduits the letter c for
#"conduit" (SWMM terminology)

unique$c <- "C"
unique$j <- "J"

#conduits
unique$name= paste(unique$c,unique$name)

#remove space
unique$name <- gsub(
  '\\s+', '', unique$name
  )
#node
unique$node= paste(unique$j,unique$node)

#remove space
unique$node <- gsub(
  '\\s+', '', unique$node
  )
```

2. Arrange and reshape data

Note: each conduit has start and end coordinates and nodes, - so there are duplicate rows for each conduit. We want to reshape this data to have both xy and nodes in the same row.

```r
#arrange by conduit name and descending elevation
arrange <- unique[
  with(
  unique, order(
    name,
    -elevation,
    na.last=FALSE)
  ),
  ]

#reshape data
reshape_dt <- dcast(
  setDT(arrange),
  name + roughness ~ rowid(name, prefix="node"),
  value.var=c("node", "x", "y", "elevation"))
```

3. Length

We must now calculate the length of the conduits using the following steps: a. Find distance "length" between xy coordinates of each conduit using geosphere. b. Use difference in elevation to calculate height c. Use pythag. theorem to calculate length

```r
#part a -  distance (adjust code based on number of pairs.
#This dataset has 18 based on the longest conduit)

dist <- reshape_dt %>%
  rowwise(
  ) %>%
  mutate (
    dist_m = distm(c(x_node1, y_node1),
                   c(x_node2, y_node2),
                   fun = distHaversine
                   )
    ) %>%
  mutate(
    dist_ft = dist_m*3.28084
    )

#part b and c - height then length
lengths <- dist %>%
  mutate (
    length = sqrt(
      (dist_ft)^2 + (elevation_node1-elevation_node2)^2)
          )%>%
  rename (
    from_node = node_node1,
    to_node = node_node2
  )
```

4. file output for conduits

[CONDUITS]
;;Name From Node To Node Length Roughness InOffset OutOffset InitFlow MaxFlow
;;———- ———- ———- ———- ———- ———- ———- ———- ———-

** The channelized stream needs to connect with the stormwater infrastructure via junctions/nodes in the urban region. It is feasible to connect them manually by drawing in the stream conduit in SWMM.

Use roughness value 0.01, for concrete pipes found in Appendix A-8 pg. 184 of EPA manual

```
conduits <- lengths %>%
  mutate(
    roughness = ifelse(is.na(roughness), 0.01, roughness)
    ) %>%
  select(
    name,
    from_node,
    to_node,
    length,
    roughness
    )

conduits$inoffset <- 0
conduits$outoffset <- 0
conduits$initflow <- 0
conduits$maxflow <- 0


write.csv(conduits,"inp_conduits.csv", row.names = FALSE)
```

5. file output for lengths

[XSECTIONS]
;;Link Shape Geom1 Geom2 Geom3 Geom4 Barrels Culvert
;;————— ———— ——————- ————-

```
xsection_dt <- merge(
  lengths,
  unique,
  by = "name"
  ) %>%
  select (
    name,
    length,
    shape,
    diameter,
    width,
    height
    )
```

a - concrete pipe dimensions

```
pipes<- xsection_dt %>%
  filter(
    shape == "CIRCULAR"
    ) %>%
  rename(
    geom1 = diameter,
    link = name
    ) %>%
  mutate(
```

```r
    geom1 = ifelse(is.na(geom1), "23", geom1)
    ) %>%
  mutate(
    geom1 = ifelse((geom1=="Other"), "23", (geom1))
    )%>%
  select(
    link,
    shape,
    length,
    geom1
  )
pipes$geom2 <- 0
pipes$geom3 <- 0
pipes$geom4 <- 0
pipes$barrels <-
pipes$culvert <- 0
```

b - box culverts, ditches, and “other” conduit dimensions

```r
ditch_box <- xsection_dt %>%
  filter(
    shape == "RECT_CLOSED" | shape == "RECT_OPEN"
    ) %>%
  rename (
    geom1 = width,
    geom2 = height,
    link = name
    ) %>%
  mutate(
    geom1 = ifelse(is.na(geom1), 5, geom1)
    ) %>%
  mutate(geom2 = ifelse(is.na(geom2), 5, geom2)
         ) %>%
  select(
    link,
    shape,
    length,
    geom1,
    geom2

  )
ditch_box$geom3 <- 0
ditch_box$geom4 <- 0
ditch_box$barrels <-
ditch_box$culvert <- 0
```

c - channel (channelized stream) dimenssions

```r
channel <- xsection_dt %>%
  filter(
    is.na(shape)
    ) %>%
  mutate(
    shape = ifelse(is.na(shape), "TRAPEZOIDAL", shape)
    ) %>%
  rename(
    geom1 = width,
    geom2 = height,
    link = name
    ) %>%
  mutate(
    geom1 = ifelse(is.na(geom1), 30, geom1)
    ) %>%
  mutate(
    geom2 = ifelse(is.na(geom2), 10, geom2)
    ) %>%
  select(
     link,
    shape,
    length,
    geom1,
    geom2
    )
#Geom3 and Geom4 are side lopes, which literature indicate vary from 1/1 to 1/2.
#Ive seen side slopes perpendicular to the ground, especially where homes are built.
channel$geom3 <- 1
channel$geom4 <- 1
channel$barrels <-
channel$culvert <- 0
```

e- bind all tables

```r
xsections_df <- rbind(
  pipes,
  ditch_box,
  channel
  ) %>%
  select(
    link,
    shape,
    geom1,
    geom2,
    geom3,
    geom4,
    barrels,
    culvert,
    length
    ) %>%
  distinct(
    link,
    shape,
    geom1,
    geom2,
    .keep_all = TRUE
    )

xsections <- rbind(
  pipes,
  ditch_box,
  channel
  ) %>%
  select(
    link,
    shape,
    geom1,
    geom2,
    geom3,
    geom4,
    barrels,
    culvert
    ) %>%
  distinct(
    link,
    shape,
    geom1,
    geom2,
    .keep_all = TRUE
    )

write.csv(xsections,"inp_xsections.csv", row.names = FALSE)
```

8. Junctions and coordinates/vertices

[JUNCTIONS]
;;Name Elevation MaxDepth InitDepth SurDepth Aponded
;;———————— ———— - ———— - ———— -

```
junctions <- unique %>% filter(
    structure != "Inlet/Outlet" | is.na(structure)
    ) %>%
  select(
    node,
    elevation
    ) %>%
  rename(
    name = node
    ) %>%
  distinct(
    name,
    .keep_all = TRUE
  )
junctions$maxdepth <- 0
junctions$initdepth <- 0
junctions$surdepth <- 0
junctions$aponded <- 0

#filter for structure type in here so we can designate which junctions are outfalls
write.csv(junctions,"inp_junctions.csv", row.names = FALSE)
```

Here is a csv that will help determine what junction to route each subcatchment's outlet to.

```
route_to <- unique %>%
  select(
    node,
    elevation,
    subc
    )

route_to[
  with(
  route_to, order(
    subc,
    -elevation,
    na.last=FALSE)
  ),
  ]

route_to2 <-unique %>%
  group_by(
    subc
    ) %>%
  mutate(
    rank = row_number(subc)
    ) %>%
  filter(
    rank == 1 | is.na(rank)
    ) %>%
  select(
    subc,
    node
    )

write.csv(route_to2,"subc_outlet.csv", row.names = FALSE)
```

[OUTFALLS]
;;Name Elevation Type Stage Data Gated Route To
;;————— ———- ———- —————- ——— —————-

```
outfalls <- unique %>%
   filter(
    structure == "Inlet/Outlet"
    ) %>%
  select(
    node,
    elevation
    ) %>%
  rename(
    name = node
    ) %>%
  distinct(
    name,
    .keep_all = TRUE
  )
outfalls$type <- "FREE"
outfalls$stagedata <- NA
outfalls$gated <- "NO"
outfalls$routeto <- NA
```

```
#you will need to check these structures in SWMM.
#If these structures are at an end of a conduit, then they are outfalls,
#otherwise you can quickly turn them back into a junction in SWMM.

write.csv(outfalls,"inp_outfalls.csv", row.names = FALSE)
```

[COORDINATES]
;;Node X-Coord Y-Coord
;;————— —————— ——————

SWMM takes x and y that are in decimal degrees (lat and long).

```
coordinates <- unique %>%
  select(
    node,
    x,
    y
    ) %>%
  distinct(
    node,
    x,
    y,
    .keep_all = TRUE
  )

write.csv(coordinates,"inp_coordinates.csv", row.names = FALSE)
```

[VERTICES]
;;Link X-Coord Y-Coord
;;————— —————— ——————

```
vertices <- vertices_dt %>%
  select(
    objectid,
    point_x,
    point_y
  ) %>%
  rename(
   link = objectid,
   x = point_x,
   y = point_y
    ) %>%
  distinct(
    link,
    x,
    y,
    .keep_all = TRUE
  )
vertices$c <- "C"

#conduits
vertices$link <- apply(
  vertices,1,function(x) paste((x[4]), sep ="",(x[1]))
  )
```

```r
#remove space
vertices$link <- gsub(
  '\\s+', '', vertices$link
  )

  vertices1 <- vertices %>%
  select(
    link,
    x,
    y
  )

write.csv(vertices1,"inp_vertices.csv", row.names = FALSE)
```