Testing Protocol Companion to textNet Vignette

Elise Zufall and Tyler Scott

7 November 2024

Pre-Processing Step I: Process PDFs

```r
library(textNet)
library(stringr)
library(testthat)
URL <- "https://sgma.water.ca.gov/portal/service/gspdocument/download/2840"
download.file(URL, destfile = "old.pdf", method="curl")

URL <- "https://sgma.water.ca.gov/portal/service/gspdocument/download/9625"
download.file(URL, destfile = "new.pdf", method="curl")

pdfs <- c("old.pdf",
       "new.pdf")

old_new_text <- textNet::pdf_clean(pdfs, ocr=F, maxchar=10000,
                  export_paths=NULL, return_to_memory=T, suppressWarn = F,
                  auto_headfoot_remove = T)
names(old_new_text) <- c("old","new")

#we expect one element per pdf
expect_that(length(old_new_text), equals(length(pdfs)))
```

Pre-Processing Step II: Parse Text

```r
library(findpython)
ret_path <- find_python_cmd(required_modules = c('spacy', 'en_core_web_lg'))


water_bodies <- c("surface water", "Surface water", "groundwater", "Groundwater",
               "San Joaquin River", "Cottonwood Creek", "Chowchilla Canal Bypass",
               "Friant Dam", "Sack Dam", "Friant Canal", "Chowchilla Bypass",
               "Fresno River", "Sacramento River", "Merced River","Chowchilla River",
               "Bass Lake", "Crane Valley Dam", "Willow Creek", "Millerton Lake",
               "Mammoth Pool", "Dam 6 Lake", "Delta","Tulare Lake",
               "Madera-Chowchilla canal", "lower aquifer", "upper aquifer",
               "upper and lower aquifers", "lower and upper aquifers",
               "Lower aquifer", "Upper aquifer", "Upper and lower aquifers",
               "Lower and upper aquifers")
```

```
   old_new_parsed <- textNet::parse_text(ret_path,
                          keep_hyph_together = F,
                          phrases_to_concatenate = water_bodies,
                          concatenator = "_",
                          text_list = old_new_text,
                          parsed_filenames=c("old_parsed","new_parsed"),
                                 overwrite = T,
                          custom_entities = list(WATER = water_bodies))
```

```
## [1] "parsing complete: old"
## [1] "parsing complete: new"
```

```
   #expect all pages are preserved
   for(k in 1:length(old_new_parsed)){
     maxpage <- max(as.numeric(stringr::str_remove(old_new_parsed[[k]]$doc_id, "text")))
   expect_that(maxpage, equals(length(old_new_text[[k]])))
   }
```

Extraction Expectations

Next we call textnet_extract() to produce the network object:

```
ent_types <- c('ORG','GPE','PERSON','WATER')
extracts <- vector(mode="list",length=length(old_new_parsed))
   for(m in 1:length(old_new_parsed)){
      extracts[[m]] <- textnet_extract(old_new_parsed[[m]],concatenator="_",cl=4,
                   keep_entities = ent_types,
                   return_to_memory=T, keep_incomplete_edges=T)
   }
```

```
## [1] "crawling 802 sentences"
## [1] "crawling 1090 sentences"
```

```
#test conditions
for(m in 1:length(old_new_parsed)){
  #checking list of entities
  onp <- old_new_parsed[[m]] |> dplyr::mutate(entitynum = cumsum(str_detect(entity, "_B")))
  onp$entitynum <- ifelse(onp$entity == "", NA, onp$entitynum)
  onp <- onp |> dplyr::group_by(entitynum) |> dplyr::mutate(entityconcat = paste(token, collapse = "_")
  onp$entityconcat <- ifelse(str_detect(onp$entity,
                       paste0(ent_types, "_B", sep = "", collapse = "|")), onp$entityconcat, NA)

  #node entities should be a subset of all entities since
  #sometimes there are improper sentences that cause
  #allentities to not make it to the nodelist
  remove_nums <- ifelse("DATE" %in% ent_types | "CARDINAL" %in% ent_types |
                            "QUANTITY" %in% ent_types | "TIME" %in% ent_types |
                            "MONEY" %in% ent_types | "PERCENT" %in% ent_types, F, T)

  allentities <- onp$entityconcat[!is.na(onp$entityconcat)]
  allentities <- clean_entities(allentities, remove_nums)
  allentities <- unique(sort(allentities))
  nodentities <- unique(sort(extracts[[m]]$nodelist$entity_name))
  #sometimes appositives happen in the middle of the entity name, which textnet removes
  nodentities <- nodentities |> str_replace_all("_", "_.*_*")
  #this method accounts for the fact that the nodentity might be a substring of the
  #original entity, since it may have included an appositive
  expect_that(all(unlist(lapply(nodentities, function(j) any(str_detect(string = allentities, pattern =
}
```

Entity Consolidation Expectations

```
   old_acronyms <- find_acronyms(old_new_text[[1]])
   new_acronyms <- find_acronyms(old_new_text[[2]])

   print(head(old_acronyms))
```

```
##                                name acronym
##                              <char>  <char>
## 1:                   Central_Valley      CV
## 2:           Total_Dissolved_Solids     TDS
## 3: California_Code_of_Regulations     CCR
## 4:  Department_of_Water_Resources     DWR
## 5:        Best_Management_Practice     BMP
## 6:   Gravelly_Ford_Water_District    GFWD
```

```
tofrom <- data.table::data.table(
   from = c(as.list(old_acronyms$acronym),
            list("Sub_basin",
                 "Sub_Basin",
                 "upper_and_lower_aquifers",
                 "Upper_and_lower_aquifers",
                 "Lower_and_upper_aquifers",
                 "lower_and_upper_aquifers")),
   to = c(as.list(old_acronyms$name),
            list("Subbasin",
                 "Subbasin",
                 c("upper_aquifer","lower_aquifer"),
                 c("upper_aquifer","lower_aquifer"),
                 c("upper_aquifer","lower_aquifer"),
                 c("upper_aquifer","lower_aquifer"))))

   old_extract_clean <- disambiguate(
      textnet_extract = extracts[[1]],
      from = tofrom$from,
      to = tofrom$to,
      match_partial_entity = c(rep(F,nrow(old_acronyms)),T,T,F,F,F,F))

   #we shouldn't have changed the overall structure of the data
   expect_that(length(old_extract_clean), equals(length(extracts[[1]])))
   #we converted from acronyms to full names so should not see any acronyms
   expect_that(any(str_detect(old_extract_clean$nodelist$entity_name,
                                paste0("^", paste0(old_acronyms$acronym, collapse = "$|^"), "$"))),equal

  tofrom <- data.table::data.table(
  from = c(as.list(new_acronyms$acronym),
            list("Sub_basin",
                 "Sub_Basin",
                 "upper_and_lower_aquifers",
                 "Upper_and_lower_aquifers",
                 "Lower_and_upper_aquifers",
                 "lower_and_upper_aquifers")),
  to = c(as.list(new_acronyms$name),
            list("Subbasin",
                 "Subbasin",
                 c("upper_aquifer","lower_aquifer"),
                 c("upper_aquifer","lower_aquifer"),
                 c("upper_aquifer","lower_aquifer"),
                 c("upper_aquifer","lower_aquifer"))))

  new_extract_clean <- disambiguate(
     textnet_extract = extracts[[2]],
     from = tofrom$from,
     to = tofrom$to,
     match_partial_entity = c(rep(F,nrow(new_acronyms)),T,T,F,F,F,F))
```

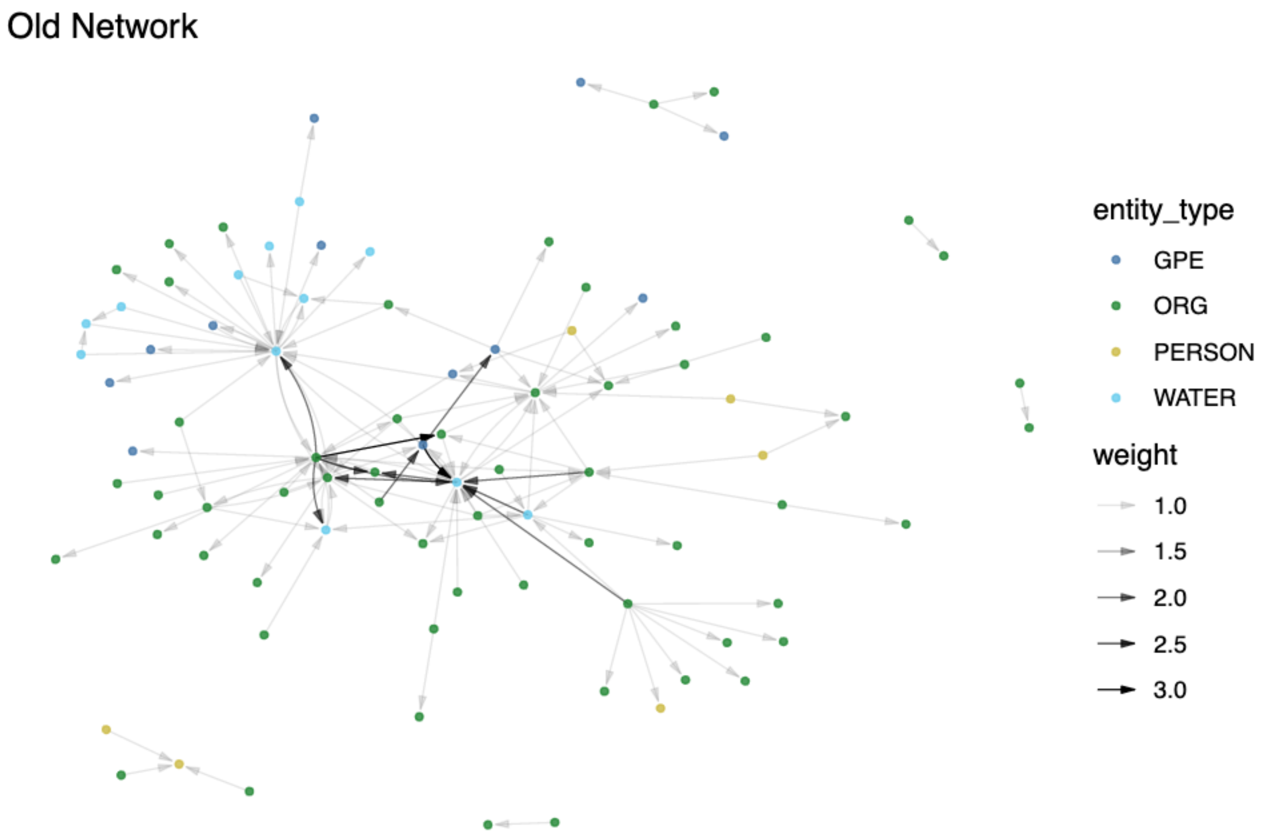
Network Attribute Expectations

```
  old_extract_net <- export_to_network(old_extract_clean, "igraph", keep_isolates = F,
                                       collapse_edges = F, self_loops = T)
  expect_that(class(old_extract_net[[1]]), equals("igraph"))
 new_extract_net <- export_to_network(new_extract_clean, "igraph", keep_isolates = F,
                                       collapse_edges = F, self_loops = T)

 expect_that(class(new_extract_net[[1]]), equals("igraph"))
  table <- t(format(rbind(old_extract_net[[2]], new_extract_net[[2]]), digits = 3,
                  scientific = F))
  colnames(table) <- c("old","new")
  print(table)
```

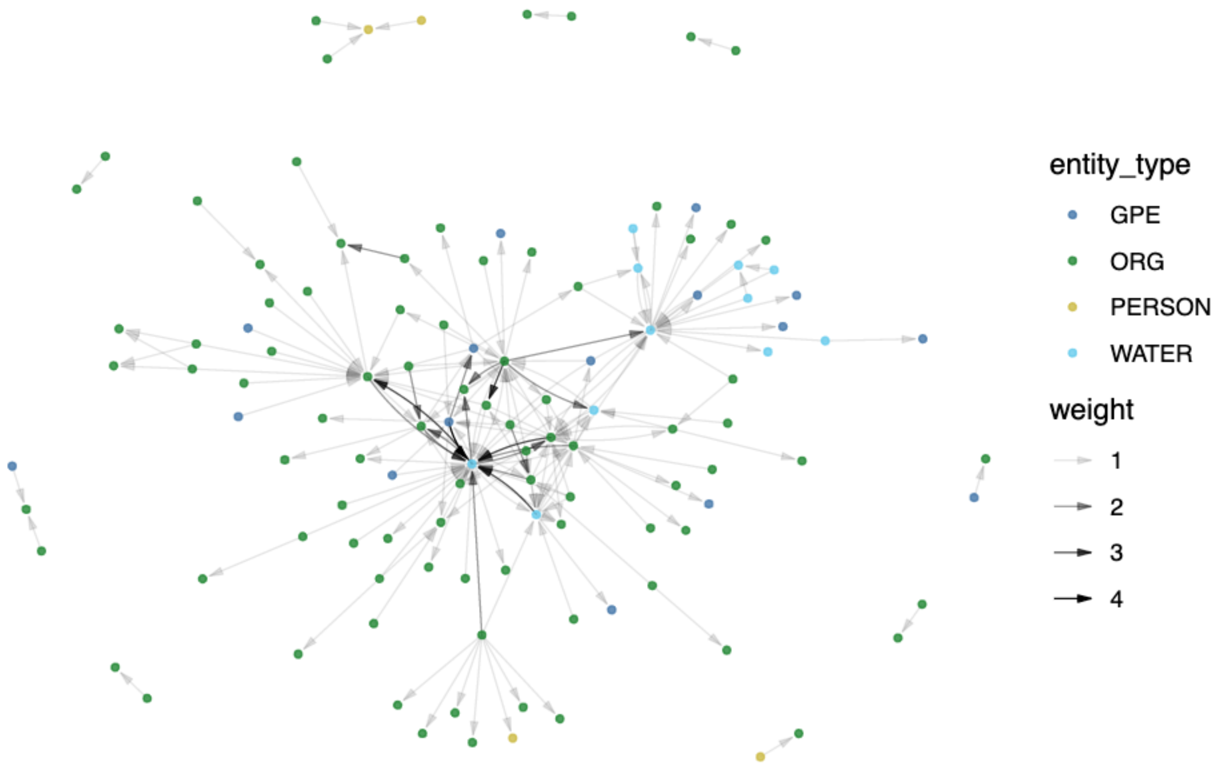
```
##                          old     new
## num_nodes                " 88"   "118"
## num_edges                "163"   "248"
## connectedness            "0.710" "0.677"
## centralization           "0.207" "0.325"
## transitivity             "0.109" "0.153"
## pct_entitytype_homophily "0.503" "0.581"
## reciprocity              "0.245" "0.306"
## mean_in_degree           "1.85"  "2.10"
## mean_out_degree          "1.85"  "2.10"
## median_in_degree         "1"     "1"
## median_out_degree        "1"     "1"
## modularity               "0.538" "0.525"
## num_communities          "12"     "16"
## percent_vbn              "0.374"  "0.423"
## percent_vbg              "0.0736" "0.0524"
## percent_vbp              "0.1288" "0.0766"
## percent_vbd              "0.0675" "0.0685"
## percent_vb               "0.135"  "0.137"
## percent_vbz              "0.221"  "0.242"
```

```
library(ggraph)
old_extract_plot <- export_to_network(old_extract_clean, "igraph", keep_isolates = F,
                                      collapse_edges = T, self_loops = T)[[1]]
new_extract_plot <- export_to_network(new_extract_clean, "igraph", keep_isolates = F,
                                      collapse_edges = T, self_loops = T)[[1]]
#order of these layers matters
ggraph(old_extract_plot, layout = 'fr')+
   geom_edge_fan(aes(alpha = weight),
                 end_cap = circle(1,"mm"),
                 color = "#000000",
                 width = 0.3,
                 arrow = arrow(angle=15,length=unit(0.07,"inches"),ends = "last",
                               type = "closed"))+
   #from Paul Tol's bright color scheme
   scale_color_manual(values = c("#4477AA","#228833","#CCBB44","#66CCEE"))+
   geom_node_point(aes(color = entity_type), size = 1,
                   alpha = 0.8)+
   labs(title= "Old Network")+
   theme_void()
```



```r
#order of these layers matters
ggraph(new_extract_plot, layout = 'fr')+
   geom_edge_fan(aes(alpha = weight),
                 end_cap = circle(1,"mm"),
                 color = "#000000",
                 width = 0.3,
                 arrow = arrow(angle=15,length=unit(0.07,"inches"),ends = "last",
                               type = "closed"))+
   #from Paul Tol's bright color scheme
   scale_color_manual(values = c("#4477AA","#228833","#CCBB44","#66CCEE"))+
   geom_node_point(aes(color = entity_type), size = 1,
                   alpha = 0.8)+
   labs(title= "New Network")+
   theme_void()
```

New Network



Edge Attribute Expectations

```
top_feats <- top_features(list(old_extract_net[[1]], new_extract_net[[1]]))
head(top_feats[[2]],10)
```

```
## # A tibble: 10 x 2
##    names   avg_fract_of_a_doc
##    <chr>                <dbl>
##  1 be                  0.104
##  2 include             0.0844
##  3 provide             0.0661
##  4 locate              0.0519
##  5 result              0.0407
##  6 base                0.0274
##  7 receive             0.0254
##  8 show                0.0224
##  9 develop             0.0212
## 10 make                0.0203
```

```
table(igraph::E(old_extract_net[[1]])$head_verb_tense)
```

```
## 
##  VB VBD VBG VBN VBP VBZ 
##  22  11  12  61  21  36
```

Composite Network Expectations

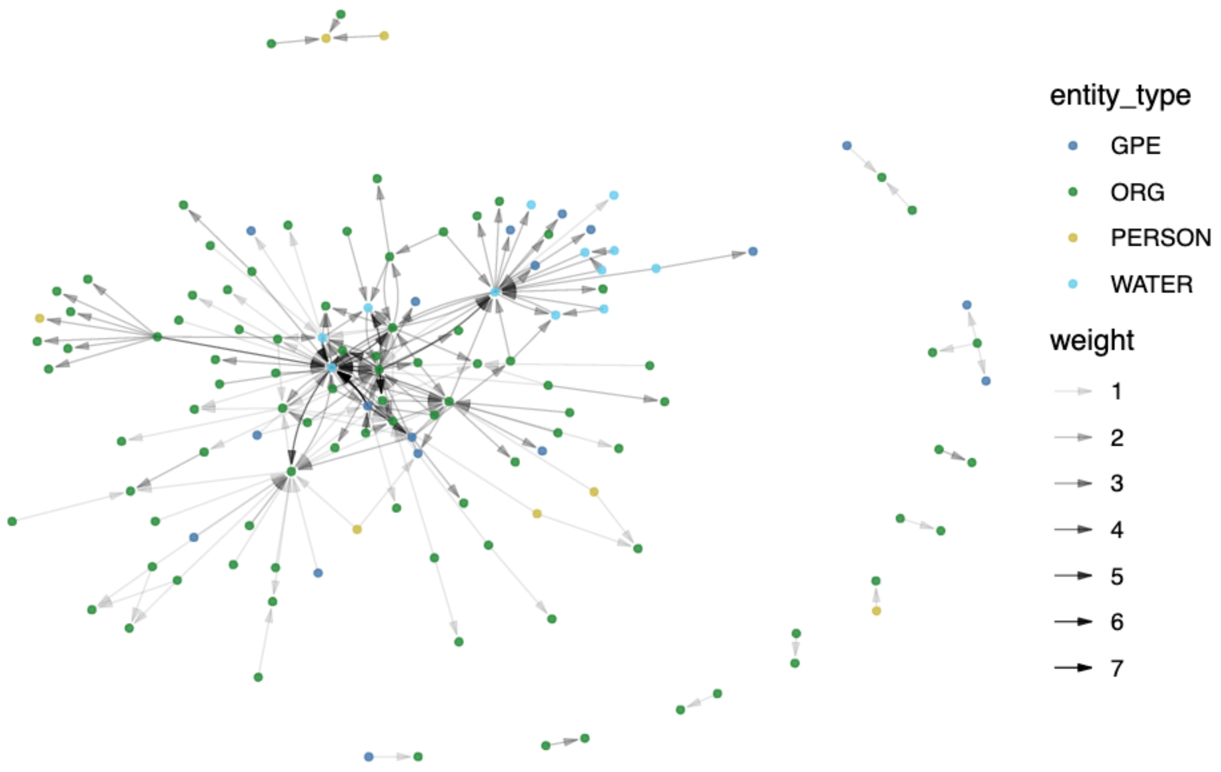
```
  composite_net <- combine_networks(list(old_extract_net[[1]], new_extract_net[[1]]),
                                    mode = "weighted")

#we expect the new nodes to be in the cleaned extracts
 expect_contains(c(old_extract_clean$nodelist$entity_name,
                  new_extract_clean$nodelist$entity_name),
                  igraph::get.vertex.attribute(composite_net, "name"))
```

```
## Warning: 'get.vertex.attribute()' was deprecated in igraph 2.0.0.
## i Please use 'vertex_attr()' instead.
## This warning is displayed once every 8 hours.
## Call 'lifecycle::last_lifecycle_warnings()' to see where this warning was
## generated.
```

```
  ggraph(composite_net, layout = 'fr')+
     geom_edge_fan(aes(alpha = weight),
                  end_cap = circle(1,"mm"),
                  color = "#000000",
                  width = 0.3,
                  arrow = arrow(angle=15,length=unit(0.07,"inches"),ends = "last",
                         type = "closed"))+
     #from Paul Tol's bright color scheme
     scale_color_manual(values = c("#4477AA","#228833","#CCBB44","#66CCEE"))+
     geom_node_point(aes(color = entity_type), size = 1,
                    alpha = 0.8)+
     labs(title= "Composite Network")+
     theme_void()
```

Composite Network



Node Attribute Expectations

```
library(network)
library(igraph)

   top_feats <- top_features(list(old_extract_net[[1]], new_extract_net[[1]]))
   print(head(top_feats[[1]],10))
```

```
## # A tibble: 10 x 2
##    names                                        avg_fract_of_a_doc
##    <chr>                                                     <dbl>
##  1 groundwater                                              0.180
##  2 gsa                                                      0.0803
##  3 san_joaquin_river                                        0.0692
##  4 gfwd_gsa                                                 0.0452
##  5 surface_water                                            0.0426
##  6 gravelly_ford_water_district                             0.0386
##  7 subbasin                                                 0.0381
##  8 gsp                                                      0.0293
##  9 madera_subbasin                                          0.0259
## 10 north_kings_groundwater_sustainability_agency            0.0254
```

```r
composite_tbl <- igraph::as_data_frame(composite_net, what = "vertices")
composite_tbl <- composite_tbl[,c("name","num_graphs_in")]

#prepare data frame version of old network, to add composite_tbl variables
old_tbl <- igraph::as_data_frame(old_extract_net[[1]], what = "both")
#this adds the num_graphs_in variable from composite_tbl
old_tbl$vertices <- dplyr::left_join(old_tbl$vertices, composite_tbl)
```

```
## Joining with `by = join_by(name)`
```

```r
#turn back into a network
old_net <- network::network(x=old_tbl$edges[,1:2], directed = T,
                            hyper = F, loops = T, multiple = T,
                            bipartiate = F, vertices = old_tbl$vertices,
                            matrix.type = "edgelist")
#we need a matrix version for some node statistics
old_mat <- as.matrix(as.matrix(export_to_network(old_extract_clean, "igraph",
                            keep_isolates = F, collapse_edges = T, self_loops = F)[[1]]))

#prepare data frame version of new network, to add composite_tbl variables
new_tbl <- igraph::as_data_frame(new_extract_net[[1]], what = "both")
#this adds the num_graphs_in variable from composite_tbl
new_tbl$vertices <- dplyr::left_join(new_tbl$vertices, composite_tbl)
```

```
## Joining with `by = join_by(name)`
```

```r
#turn back into a network
new_net <- network::network(x=new_tbl$edges[,1:2], directed = T,
                            hyper = F, loops = T, multiple = T,
                            bipartiate = F, vertices = new_tbl$vertices,
                            matrix.type = "edgelist")
#we need a matrix version for some node statistics
new_mat <- as.matrix(as.matrix(export_to_network(new_extract_clean, "igraph",
                            keep_isolates = F, collapse_edges = T, self_loops = F)[[1]]))
```

```r
paths2 <- diag(old_mat %*% old_mat)
recip <- 2*paths2 / sna::degree(old_net)
totalCC <- as.vector(unname(DirectedClustering::ClustF(old_mat, type = "directed",
                                                 isolates="zero")$totalCC))
closens <- sna::closeness(old_net, gmode = "graph", cmode="suminvundir")
between <- sna::betweenness(old_net,gmode = "graph",cmode="undirected")
deg <- sna::degree(old_net, gmode = "graph", cmode = "undirected")
old_node_df <- dplyr::tibble(name = network::get.vertex.attribute(old_net,
        "vertex.names"),
        closens,
        between,
        deg,
        recip,
        totalCC,
        entity_type = network::get.vertex.attribute(old_net,"entity_type"),
        num_graphs_in = network::get.vertex.attribute(old_net, "num_graphs_in"))
```

```
paths2 <- diag(new_mat %*% new_mat)
recip <- 2*paths2 / sna::degree(new_net)
totalCC <- as.vector(unname(DirectedClustering::ClustF(new_mat, type = "directed", isolates="zero")
closens <- sna::closeness(new_net, gmode = "graph", cmode="suminvundir")
between <- sna::betweenness(new_net,gmode = "graph",cmode="undirected")
deg <- sna::degree(new_net, gmode = "graph", cmode = "undirected")
new_node_df <- dplyr::tibble(name = network::get.vertex.attribute(new_net,
        "vertex.names"),
        closens,
        between,
        deg,
        recip,
        totalCC,
        entity_type = network::get.vertex.attribute(new_net,"entity_type"),
        num_graphs_in = network::get.vertex.attribute(new_net, "num_graphs_in"))

summary(old_node_df)
```

```
##      name              closens           between             deg
## Length:88          Min.   :0.01149   Min.   :   0.00   Min.   : 0.00
## Class :character   1st Qu.:0.25465   1st Qu.:   0.00   1st Qu.: 0.00
## Mode  :character   Median :0.30134   Median :   0.00   Median : 1.00
##                    Mean   :0.26573   Mean   :  62.41   Mean   : 1.67
##                    3rd Qu.:0.32217   3rd Qu.:  19.66   3rd Qu.: 1.00
##                    Max.   :0.51149   Max.   :1191.82   Max.   :19.00
##      recip            totalCC          entity_type         num_graphs_in
## Min.   :0.0000   Min.   :0.000000   Length:88          Min.   :1.000
## 1st Qu.:0.0000   1st Qu.:0.000000   Class :character   1st Qu.:2.000
## Median :0.0000   Median :0.000000   Mode  :character   Median :2.000
## Mean   :0.0518   Mean   :0.080564                      Mean   :1.864
## 3rd Qu.:0.0000   3rd Qu.:0.003472                      3rd Qu.:2.000
## Max.   :1.0000   Max.   :1.000000                      Max.   :2.000
```

```
summary(new_node_df)
```

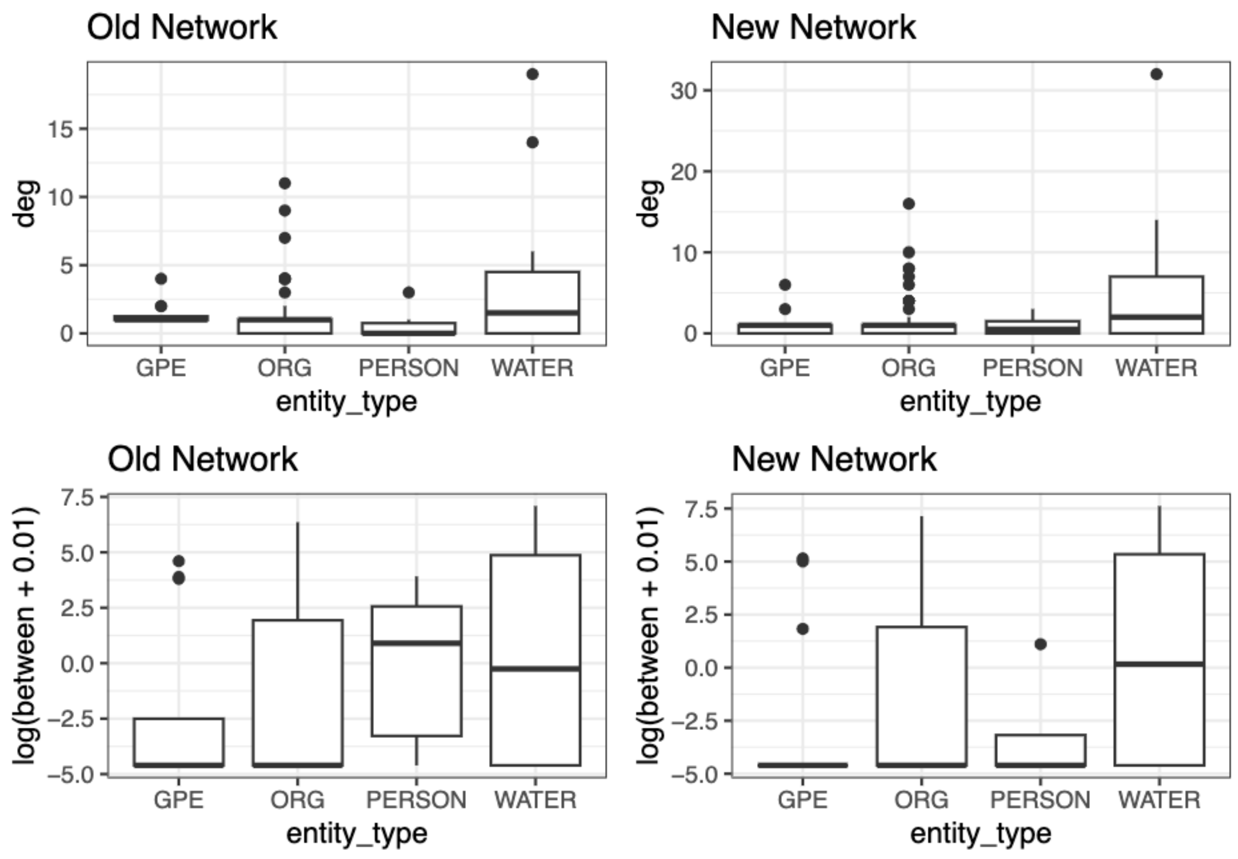
```
##      name              closens            between              deg
## Length:118         Min.   :0.008547   Min.   :   0.000   Min.   : 0.00
## Class :character   1st Qu.:0.232087   1st Qu.:   0.000   1st Qu.: 0.00
## Mode  :character   Median :0.282051   Median :   0.000   Median : 1.00
##                    Mean   :0.246142   Mean   :  82.712   Mean   : 1.78
##                    3rd Qu.:0.309829   3rd Qu.:   6.022   3rd Qu.: 1.00
##                    Max.   :0.512821   Max.   :2025.067   Max.   :32.00
##      recip             totalCC          entity_type         num_graphs_in
## Min.   :0.00000   Min.   :0.00000   Length:118         Min.   :1.000
## 1st Qu.:0.00000   1st Qu.:0.00000   Class :character   1st Qu.:1.000
## Median :0.00000   Median :0.00000   Mode  :character   Median :2.000
## Mean   :0.04173   Mean   :0.11473                      Mean   :1.644
## 3rd Qu.:0.00000   3rd Qu.:0.08808                      3rd Qu.:2.000
## Max.   :1.00000   Max.   :1.00000                      Max.   :2.000
```

```
    old_node_df$plan_version <- "old"
    new_node_df$plan_version <- "new"
    combineddf <- rbind(old_node_df, new_node_df)
    with(combineddf,table(plan_version,num_graphs_in))
```

```
##              num_graphs_in
## plan_version  1  2
##          new 42 76
##          old 12 76
```

```
    library(gridExtra)
    library(ggplot2)
    b1 <- ggplot(old_node_df, aes(x = entity_type, y = deg)) + geom_boxplot() +
      theme_bw() + labs(title="Old Network")
    b2 <- ggplot(new_node_df, aes(x = entity_type, y = deg)) + geom_boxplot() +
      theme_bw() + labs(title="New Network")
        b3 <- ggplot(old_node_df, aes(x = entity_type, y = log(between+0.01))) +
          geom_boxplot() + theme_bw() + labs(title="Old Network")
    b4 <- ggplot(new_node_df, aes(x = entity_type, y = log(between+0.01))) +
      geom_boxplot() + theme_bw() + labs(title="New Network")

    grid.arrange(b1, b2, b3, b4, ncol=2)
```


Old Network
deg
GPE
ORG
PERSON
WATER
entity_type
New Network
deg
GPE
ORG
PERSON
WATER
entity_type
Old Network
log(between + 0.01)
GPE
ORG
PERSON
WATER
entity_type
New Network
log(between + 0.01)
GPE
ORG
PERSON
WATER
entity_type