Testing Protocol Companion to textNet Vignette

Elise Zufall and Tyler Scott

7 November 2024

Pre-Processing Step I: Process PDFs

```
library(textNet)
library(stringr)
library(testthat)
#these example pdfs are included with the package
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

```
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")

if(!requireNamespace("spacyr", quietly = T)){
    stop("Package 'spacyr' must be installed to use this function.",
         call.=F)
  }
```

```
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]],cl=4,
                   keep_entities = ent_types, keep_incomplete_edges=T)
   }
```

```
#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 = j))))), equals(T))
}
```

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$acronym, collapse = "$|^"),
               "$"))),equals(F))
```

```
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

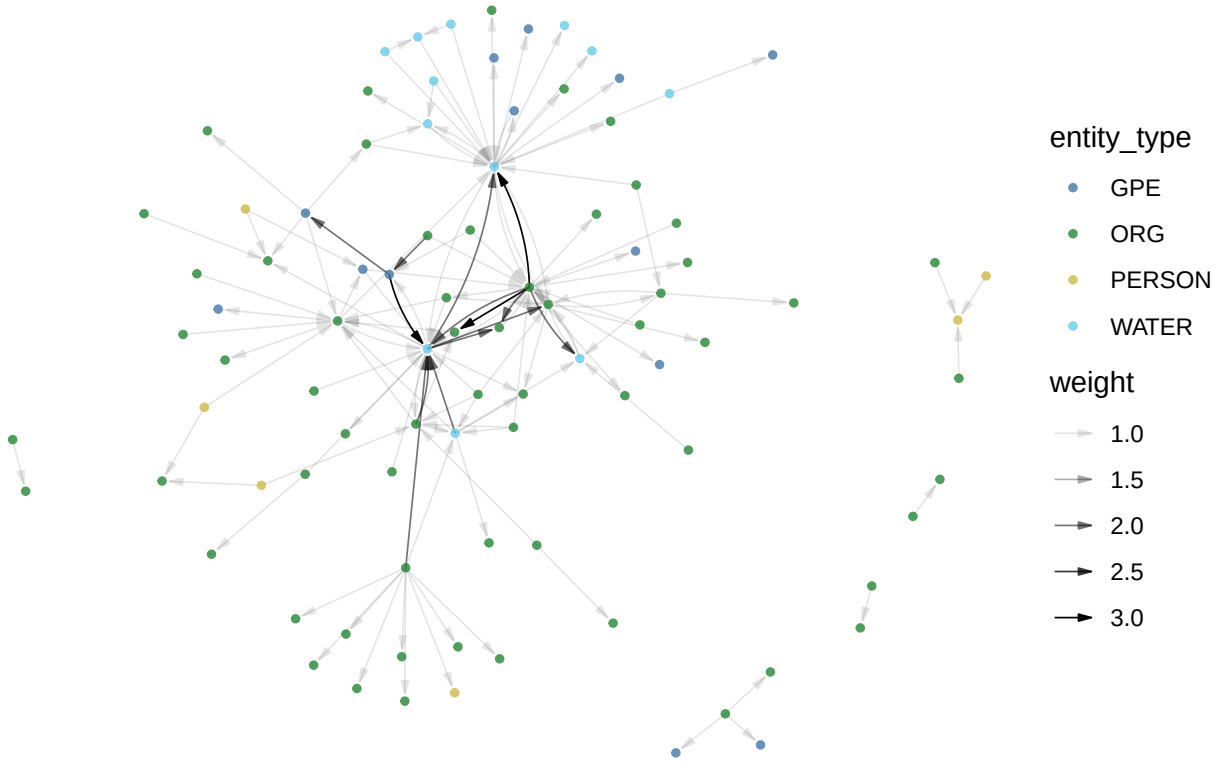
```
set.seed(50000)
old_extract_net <- export_to_network(old_extract_clean, "igraph", keep_isolates = F,
                                     collapse_edges = F, self_loops = T)
expect_that(class(old_extract_net[[1]]), equals("igraph"))
set.seed(50000)
new_extract_net <- export_to_network(new_extract_clean, "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                 " 92"    "123"
## num_edges                 "172"    "260"
## connectedness             "0.721"  "0.689"
## centralization            "0.220"  "0.332"
## transitivity              "0.111"  "0.152"
## pct_entitytype_homophily "0.506"  "0.581"
## reciprocity               "0.250"  "0.304"
## mean_in_degree            "1.87"   "2.11"
## mean_out_degree           "1.87"   "2.11"
## median_in_degree          "1"      "1"
## median_out_degree         "1"      "1"
## modularity                "0.528"  "0.519"
```

```
## num_communities          "12"     "16"
## percent_vbn              "0.355"  "0.404"
## percent_vbg              "0.0698" "0.0500"
## percent_vbp              "0.1337" "0.0846"
## percent_vbd              "0.0698" "0.0692"
## percent_vb               "0.128"  "0.131"
## percent_vbz              "0.244"  "0.262"
```

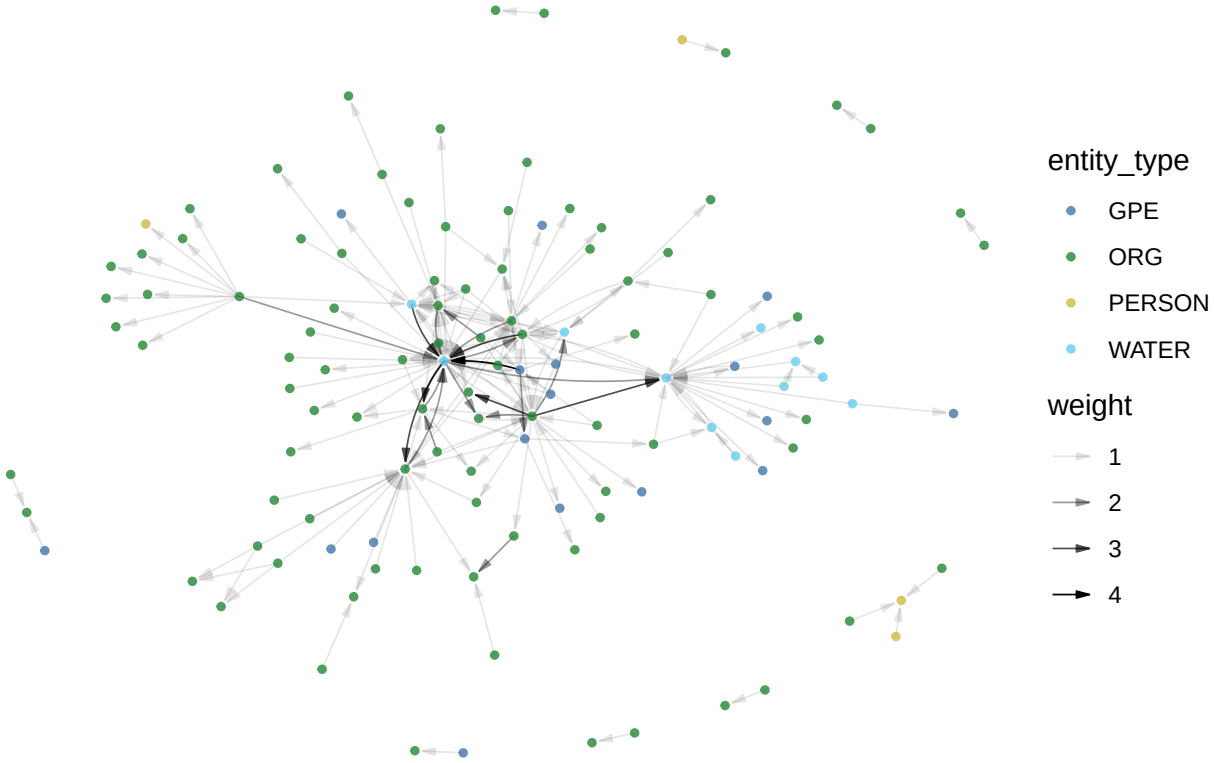
```
library(ggraph)
set.seed(50000)
old_extract_plot <- export_to_network(old_extract_clean, "igraph", keep_isolates = F,
                                      collapse_edges = T, self_loops = T)[[1]]
set.seed(50000)
new_extract_plot <- export_to_network(new_extract_clean, "igraph", keep_isolates = F,
                                      collapse_edges = T, self_loops = T)[[1]]
   #order of these layers matters
set.seed(50000)
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()
```

Old Network




```
#order of these layers matters
set.seed(50000)
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                  0.149
##  2 include             0.0802
##  3 provide             0.0628
##  4 locate              0.0493
##  5 result              0.0386
##  6 base                0.0261
##  7 receive             0.0242
##  8 show                0.0212
##  9 develop             0.0202
## 10 make                0.0193
```

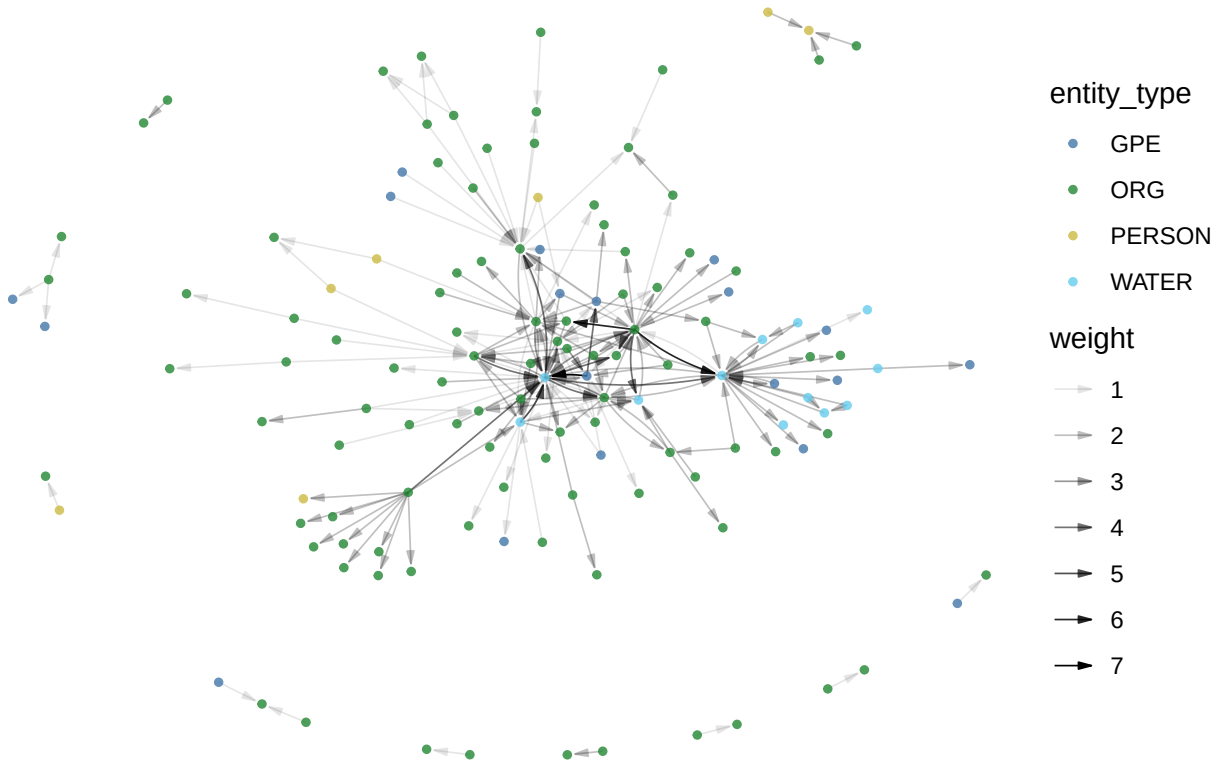
```
table(igraph::E(old_extract_net[[1]])$head_verb_tense)
```

```
##
##  VB VBD VBG VBN VBP VBZ
##  22  12  12  61  23  42
```

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"))
set.seed(50000)
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                                              0.185
##  2 gsa                                                      0.0835
##  3 san_joaquin_river                                        0.0705
##  4 gfwd_gsa                                                 0.0430
##  5 surface_water                                            0.0405
##  6 gravelly_ford_water_district                             0.0367
##  7 subbasin                                                 0.0362
##  8 north_kings_groundwater_sustainability_a~                0.0290
##  9 madera_subbasin                                          0.0280
## 10 gsp                                                      0.0279
```

```
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)`
```

```
#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
set.seed(50000)
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)`
```

```
#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
set.seed(50000)
new_mat <- as.matrix(as.matrix(export_to_network(new_extract_clean, "igraph",
                            keep_isolates = F, collapse_edges = T, self_loops = F)[[1]]))
```

```
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")$totalCC))
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
##  Length:92          Min.   :0.01099   Min.   :   0.0
##  Class :character   1st Qu.:0.26282   1st Qu.:   0.0
##  Mode  :character   Median :0.29597   Median :   0.0
##                     Mean   :0.26742   Mean   :  67.1
##                     3rd Qu.:0.32042   3rd Qu.:  20.7
```

```
##                     Max.   :0.51648   Max.   :1271.4
##       deg             recip             totalCC
##  Min.   : 0.000   Min.   :0.00000   Min.   :0.000000
##  1st Qu.: 0.000   1st Qu.:0.00000   1st Qu.:0.000000
##  Median : 1.000   Median :0.00000   Median :0.000000
##  Mean   : 1.685   Mean   :0.05151   Mean   :0.083182
##  3rd Qu.: 1.000   3rd Qu.:0.00000   3rd Qu.:0.002273
##  Max.   :20.000   Max.   :1.00000   Max.   :1.000000
##  entity_type        num_graphs_in
##  Length:92          Min.   :1.00
##  Class :character   1st Qu.:2.00
##  Mode  :character   Median :2.00
##                     Mean   :1.87
##                     3rd Qu.:2.00
##                     Max.   :2.00
```

```
summary(new_node_df)
```

```
##      name              closens            between
##  Length:123         Min.   :0.008197   Min.   :   0.000
##  Class :character   1st Qu.:0.239413   1st Qu.:   0.000
##  Mode  :character   Median :0.282104   Median :   0.000
##                     Mean   :0.248514   Mean   :  88.480
##                     3rd Qu.:0.312158   3rd Qu.:   5.404
##                     Max.   :0.516393   Max.   :2279.689
##       deg             recip             totalCC
##  Min.   : 0.000   Min.   :0.00000   Min.   :0.0000
##  1st Qu.: 0.000   1st Qu.:0.00000   1st Qu.:0.0000
##  Median : 1.000   Median :0.00000   Median :0.0000
##  Mean   : 1.797   Mean   :0.04079   Mean   :0.1209
##  3rd Qu.: 1.000   3rd Qu.:0.00000   3rd Qu.:0.1110
##  Max.   :33.000   Max.   :1.00000   Max.   :1.0000
##  entity_type        num_graphs_in
##  Length:123         Min.   :1.00
##  Class :character   1st Qu.:1.00
##  Mode  :character   Median :2.00
##                     Mean   :1.65
##                     3rd Qu.:2.00
##                     Max.   :2.00
```

```
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 43 80
##          old 12 80
```

```
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)
```