Ontologies in R

Center for Peace and Security Studies (cPASS)

Thomas Brailey

May 2020

When dealing with hierarchical data, ontologies, dendrograms, and trees can be a good way of visualizing relationships between the so-called "parent" and "child" nodes of a dataset. This document provides a cursory overview of hierarchical data and some useful packages in R to deal with such data.

Prep

```
rm(list = ls())

library(webshot)
`%>%` <-
  magrittr::`%>%`
```

Load and clean data

Load

We start by loading the cleaned IISS data. This dataframe contains information about the count of military units at the country-year level. It is hierarchical in the sense that we break up military equipment into type and subtype.

```
# Load data
df <-
  readRDS(file = "../../IISS/data/01e_addrows.rds")

df <-
  data.frame(df)
```

Prep

We need the data in tree format.

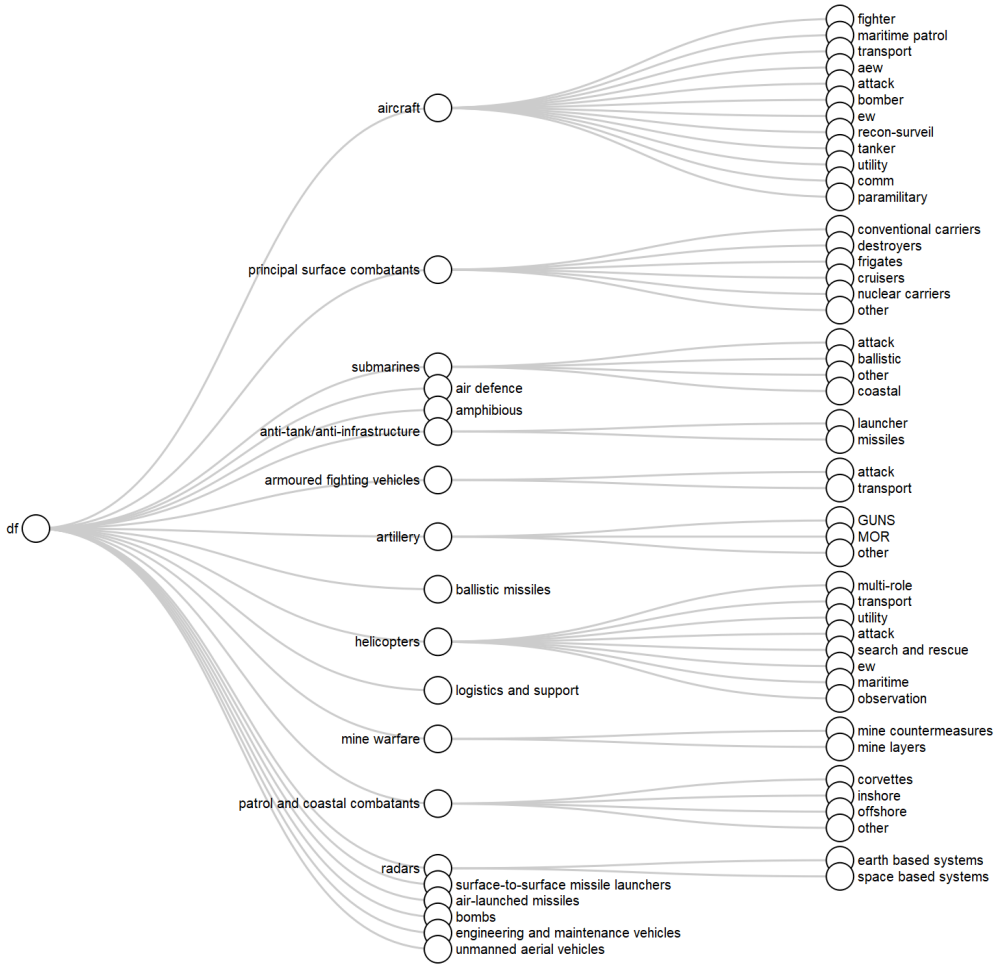
```
# Temp recode carriers
df$tek[df$tek == "principal surface combatants_aircraft carriers_nuclear"] <-
  "principal surface combatants_nuclear carriers"
```

```r
df$tek[df$tek == "principal surface combatants_aircraft carriers_non-nuclear"] <-
  "principal surface combatants_conventional carriers"

df <-
  df %>%
  dplyr::select(tek) %>%
  tidyr::separate(tek, c("parent", "child"), sep = "_", remove = TRUE) %>%
  dplyr::distinct() %>%
  dplyr::filter_all(dplyr::any_vars(!is.na(.)))

df$child[df$child == "all"] <- NA
```

Create tree

Our first example uses the package *collapsibleTree*.

```r
widgetToPng <- function(widget, file = "widget.png",  ...) {
  temp <- tempfile(fileext = ".html")
  file <- R.utils::getAbsolutePath(file)
  htmlwidgets::saveWidget(widget, temp)
  webshot(
    temp, file,
    selector = "#htmlwidget_container",
    zoom = 2,
    delay = 0.5,
    ...
  )
}

widgetToPng(
collapsibleTree::collapsibleTree(df, hierarchy = c("parent", "child"),
                                 collapse = FALSE, width = 800, height = 800),
"collapsibleTree.png"
)
```



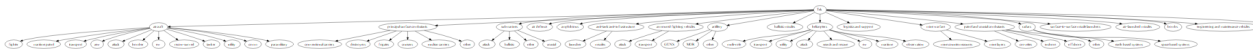
Our next example uses the package *data.tree*. Using *data.tree* requires a little bit of re-formatting, but nothing too complicated. Calling print() on the object df_tree shows us the new structure of the data frame. In order to plot the tree, simply call plot().

```
# Remove NAs
df$child[is.na(df$child)] <- ""

# Generate pathString as new column
df$pathString <-
  paste("Tek",
        df$parent,
        df$child,
        sep = "|")
```

```r
# Create list
df_tree <-
  data.tree::as.Node(df, pathDelimiter = "|")

print(df_tree, limit = 15)
```

```
##                     levelName
## 1  Tek
## 2   ¦--aircraft
## 3   ¦   ¦--fighter
## 4   ¦   ¦--maritime patrol
## 5   ¦   ¦--transport
## 6   ¦   ¦--aew
## 7   ¦   ¦--attack
## 8   ¦   ¦--bomber
## 9   ¦   ¦--ew
## 10  ¦   ¦--recon-surveil
## 11  ¦   ¦--tanker
## 12  ¦   ¦--utility
## 13  ¦   ¦--comm
## 14  ¦   °--paramilitary
## 15  °--... 18 nodes w/ 33 sub
```

```r
widgetToPng(
plot(df_tree),
"collapsibleTree2.png"
)
```

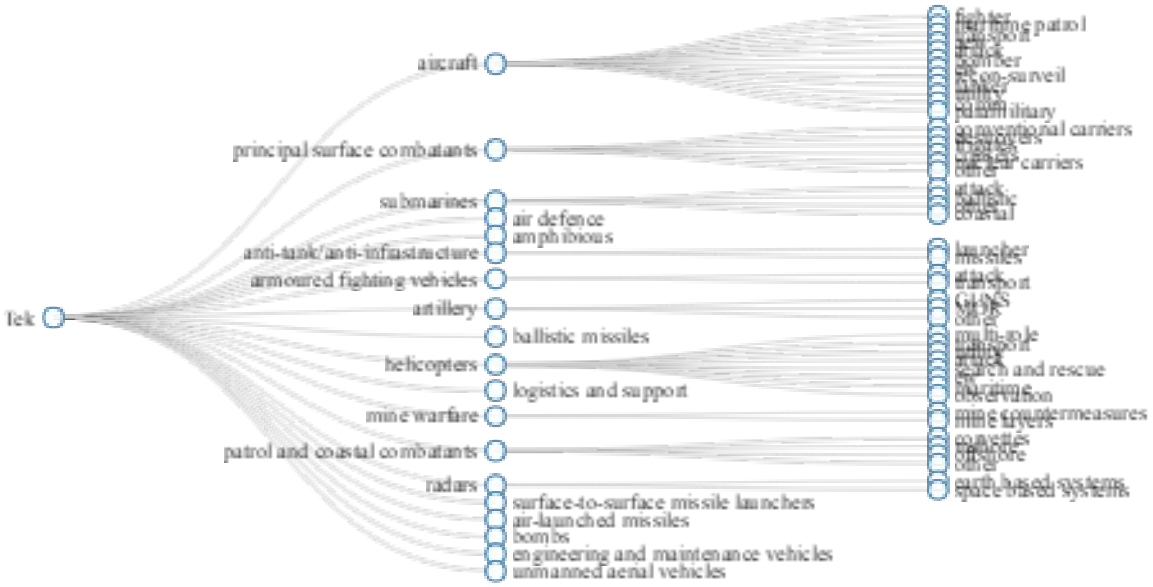


This isn't particularly helpful as a visualization, so we turn to the package *networkD3* to better visualize our data.tree object.

```
df_list <-
  data.tree::ToListExplicit(df_tree, unname = TRUE)

df_vis <-
  networkD3::radialNetwork(df_list)

df_vis
```

networkD3 has plenty of tree and dendrogram templates to choose from.

```
df_vis <-
  networkD3::diagonalNetwork(df_list)

df_vis
```



These packages allow one to easily manipulate a data.frame object to a hierarchical dataset with ease. There are more complex methods which involve edges and vertices. This information will be added to the document at a later date.

For further reading, see the vignettes for the packages discussed above:

- collapsibleTree
- data.tree
- networkD3

There are many other packages in R designed to visualize hierarchical data. They all essentially have the same capabilities as the packages we have looked at in this document. Here are a few of them:

- igraph
- ontologyX
- ggdendro