Untitled

Curtis C. Bohlen

2022-12-07

Contents

Introduction

CBEP recently received a grant from NSF's CIVIC Innovation Challenge to work on developing hydrodynamic models that address community needs in Portland Harbor. As part of the project, CBEP hosted three community workshops in November of 2022.

Facilitators produced both "live" notes during the meeting – visible to all on a screen at the front of the meeting room – and detailed meeting transcripts. CBEP staff then reviewed those notes paragraph by paragraph, and coded each paragraph in terms of five characteristics:

- Potential users and uses of hydrodynamic models,
- Ideas for improving communications of model results (e.g., communications channels and user interface design),
- Data or information needs identified by community members,
- Implied extensions of the initial Casco Bay Model required to fully address those data needs, and
- Suggestions about monitoring or data collection that could improve information availability.

Of course, not all paragraphs include information related to each of the five types of information, so there is not a perfect one-to-one correspondence between categories.

In this R Notebook, I explore these data in terms of frequency with which certain ideas came up, and cross-correlations among ideas.

Load Packages

```
library(tidyverse)
#> -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
#> v ggplot2 3.4.0      v purrr   0.3.5
#> v tibble  3.1.8      v dplyr   1.0.10
#> v tidyr   1.2.1      v stringr 1.5.0
#> v readr   2.1.3      v forcats 0.5.2
#> -- Conflicts ------------------------------------------ tidyverse_conflicts() --
#> x dplyr::filter() masks stats::filter()
#> x dplyr::lag()    masks stats::lag()
library(readxl)
library(networkD3)

theme_set(theme_classic())
```

Load Data

```
the_data <- read_excel("Export_Data_Query.xlsx" ) %>%
  mutate(ID = as.integer(ID))
head(the_data)
#> # A tibble: 6 x 14
#>      ID Category   Day   Comment User_~1 Inter~2 Inter~3 Data_~4 Data_~5 Exten~6
#>   <int> <chr>      <chr> <chr>   <chr>   <chr>   <chr>   <chr>   <chr>   <chr>
#> 1     1 Live Comm~ Day ~ How ca~ Shore ~ Keep i~ Exampl~ <NA>    <NA>    Waters~
#> 2     1 Live Comm~ Day ~ How ca~ Shore ~ Keep i~ Exampl~ <NA>    <NA>    Waters~
#> 3     2 Live Comm~ Day ~ Use of~ Marine~ Make i~ General Waves   4       <NA>
#> 4     2 Live Comm~ Day ~ Use of~ Shore ~ Make i~ General Waves   4       <NA>
#> 5     3 Live Comm~ Day ~ MS4 pr~ Water ~ I Want~ Shared~ <NA>    <NA>    Waters~
#> 6     4 Live Comm~ Day ~ Consid~ Not Sp~ I Want~ Outrea~ <NA>    <NA>    <NA>
#> # ... with 4 more variables: Extension_Timing <chr>, Monitoring_Category <chr>,
#> #   Performance_Apply <chr>, Performance_Criterion <chr>, and abbreviated
#> #   variable names 1: User_Category, 2: Interface_Category, 3: Interface_Group,
#> #   4: Data_Group, 5: Data_Timing, 6: Extension_Category
```

Our coding was generated in a somewhat sloppy `Access` database, and because of the way SQL works, it is easier to replace numerical values for some groups here, in `R`, rather than before we exported the data from `Access`. I read in the dictionaries here.

```
timing_table <- read_excel("Timing Category.xlsx",
    col_types = c("numeric", "text", "text"))
```

And finally I correct the data table to all text entries.

```
the_data <- the_data %>%
  mutate(Data_Timing = timing_table$Timing[match(Data_Timing, timing_table$ID)],
         Extension_Timing = timing_table$Timing[match(Extension_Timing,
                                                      timing_table$ID)])
```

#A Warning about Uniqueness We have to be careful here, because each note or comment can be represented in this data table multiple times. Each paragraph in the meeting transcript might imply several different users, for example. But if there are multiple users and multiple data types, the records got duplicated (in part) in the SQL query. So for any analysis, we need to test for uniqueness of the data. always

We actually have over 400 records, built out of just over 200 unique comments.

```
cat("All rows in the data:\t\t")
#> All rows in the data:
nrow(the_data)
#> [1] 431

cat("Unique comments reviewed:\t")
#> Unique comments reviewed:
the_data %>%
  select(ID) %>%
  unique() %>%
  nrow()
#> [1] 207
```

Frequency Tables

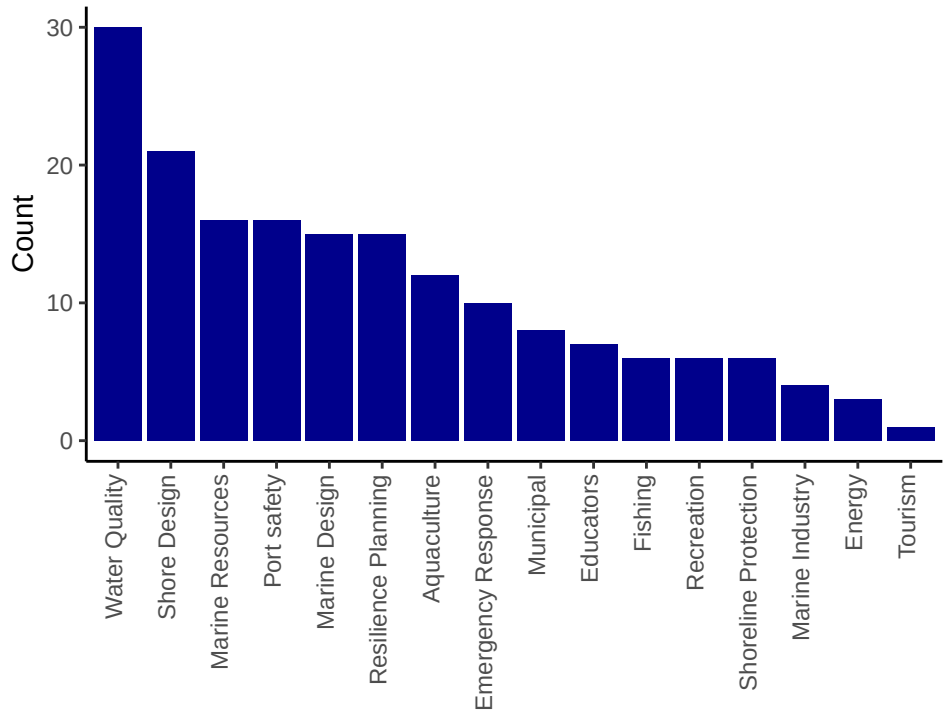
Users

```
tmp <- the_data %>%
  select(ID, User_Category) %>%
  unique()
tst <- xtabs(~User_Category, tmp) %>%
  sort(decreasing = TRUE) %>%
  as_tibble()

cat("Number of Unique User Records:\t")
#> Number of Unique User Records:
sum(tst$n)
#> [1] 244
```

```
tst %>%
  filter(User_Category != "Not Specified") %>%
  mutate(User_Category = fct_reorder(User_Category, n, .desc = TRUE)) %>%

  ggplot(aes(User_Category,n)) +
  geom_col(fill = "blue4") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1, vjust = 0.25)) +
  ylab('Count') +
  xlab("")
```



Sankay Plot

From Users to Data Requests

We need to create a network data set, which has to identify links between notes by `Source`, `Target` and `Value`.

We have a semi-hierarchical network, but we have not structured it the way the Sankay chart function wants. We need to do some data wrangling.

First, we select the user and data information, and edit so names don't don't look equivalent.

```
tmp <- the_data %>%
  select(ID, User_Category, Data_Group) %>%
  filter(! is.na(User_Category)) %>%
  filter(! is.na(Data_Group)) %>%
  filter(User_Category != "Not Specified") %>%
  mutate(User_Category = if_else(User_Category == 'Other',
                                 'Other_User', User_Category),
         Data_Group = if_else(Data_Group == 'Other',
                                 'Other_Data',Data_Group)) %>%
  mutate(User_Category = if_else(User_Category == 'Water Quality',
                                 'Water Quality User', User_Category),
         Data_Group = if_else(Data_Group == 'Water Quality',
                                 'Water Quality Data',Data_Group)) %>%
  mutate(User_Category = fct_infreq(User_Category),
         Data_Group = fct_infreq(Data_Group)) %>%
  unique()
```

We and calculate the number of links between each pair of "before" and "after" nodes.

```
tmp2 <- tmp %>%
  group_by(User_Category, Data_Group) %>%
  summarize(weight = n(),
            .groups = 'drop')
```

I do some fancy footwork to be able to control colors in the final version of the diagram.

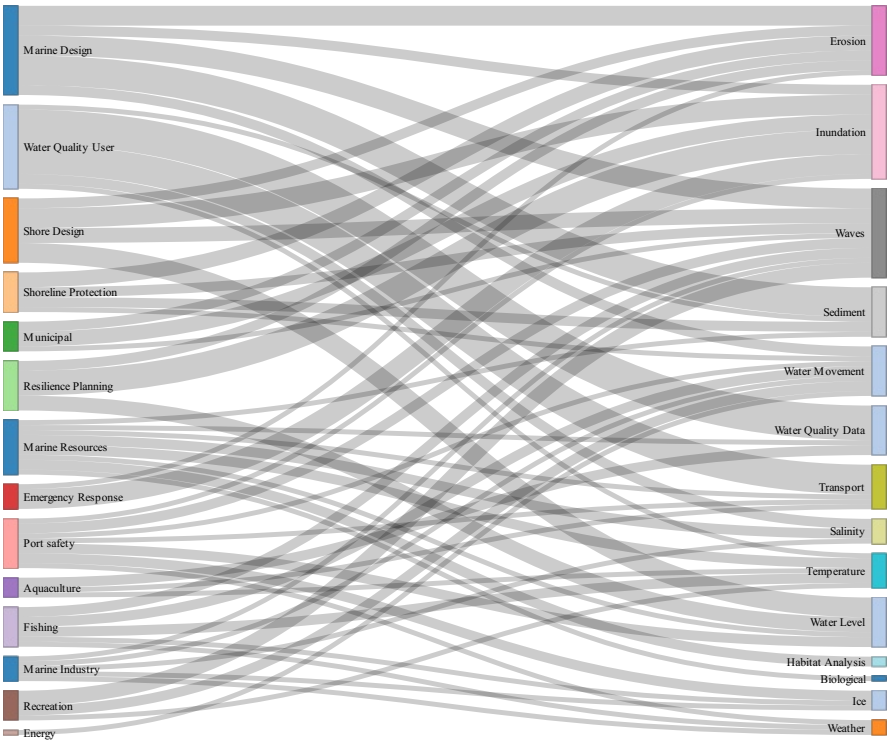
```
nodes <- tibble(node_name = c(unique(tmp2$User_Category),
                              unique(tmp2$Data_Group)),
                groups = c(rep("A", length(unique(tmp2$User_Category))),
                           rep("B", length(unique(c(tmp2$Data_Group))))))
```

Then we have to convert the character string values to zero-valued numerical IDs, as required by the `networkD3` package.

```
tmp3 <- tmp2 %>%
  mutate(link_group = User_Category,
         User_Category = match(User_Category, nodes$node_name) - 1,
         Data_Group = match(Data_Group, nodes$node_name) - 1)
```

Draft Graphic

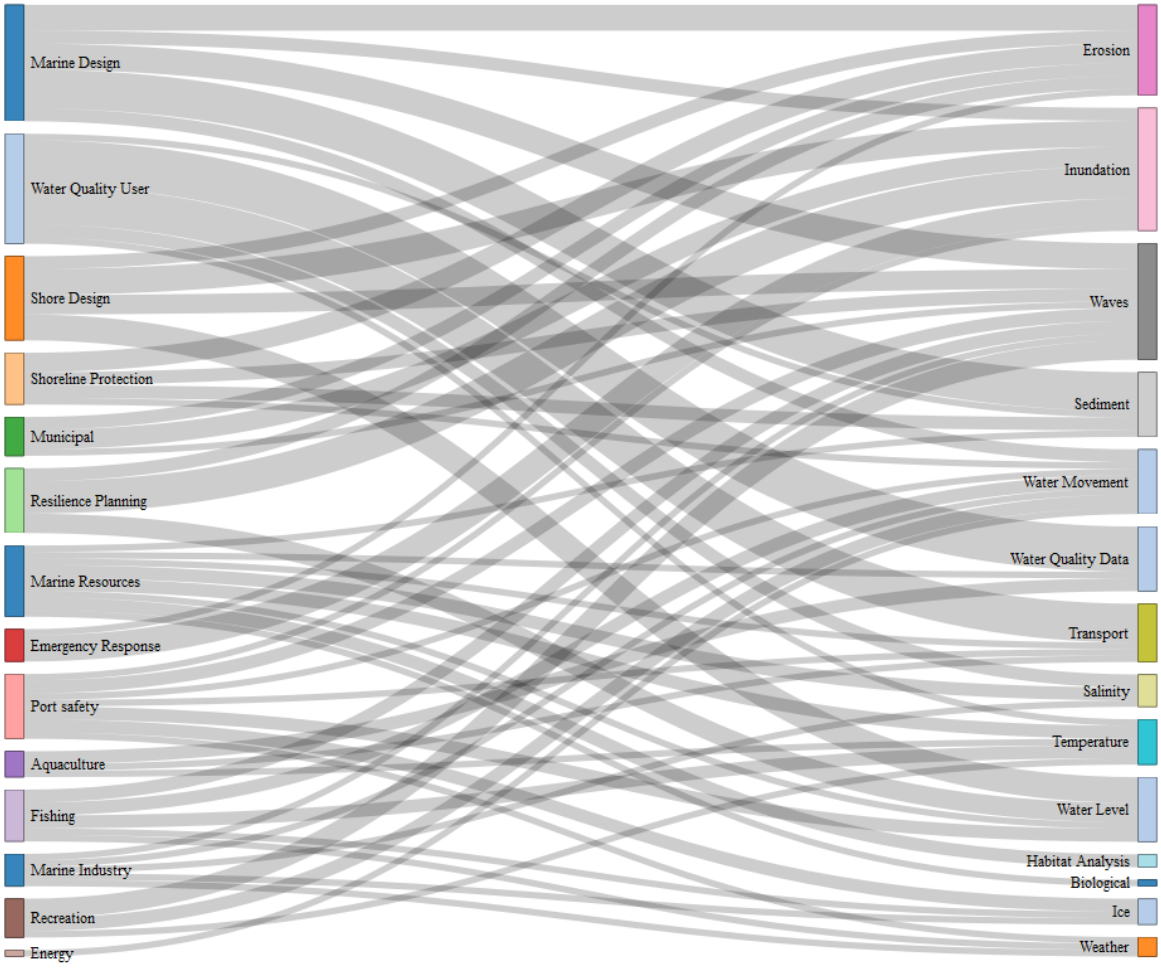
```
plt <- sankeyNetwork(Links = tmp3,
             Source = "User_Category",
             Target = "Data_Group",
             Value = "weight",
             Nodes = nodes,
             NodeID = "node_name",
             height = 800,
             fontSize = 12,
             iterations = 0)
#> Links is a tbl_df. Converting to a plain data frame.
#> Nodes is a tbl_df. Converting to a plain data frame.
plt
```




```
saveNetwork(plt, file = 'users and data.html', selfcontained = TRUE)
```

```
webshot::webshot('users and data.html', 'users and data.png')
```

¶¶¶



Custom Colors

It might be nice to color the nodes or links according to certain categories. This is surprisingly difficult, because the package uses one color scale for both notes and links.

I generate new node group names.

```
users <- as.character(levels(tmp2$User_Category))

nodes <- tibble(node_name = c(users,
                              as.character(levels(tmp2$Data_Group))),
                groups = c(users,
                           rep("B", length(unique(c(tmp2$Data_Group))))))
```

And specify matching color groups for the left hand nodes.

```
tmp3 <- tmp2 %>%
  mutate(link_group = as.character(User_Category),
         User_Category = match(User_Category, nodes$node_name) - 1,
         Data_Group = match(Data_Group, nodes$node_name) - 1)
```

Now I have to define my color palette. This is some sort of web-based color palette definition. . . . I randomize color order to minimize conflict with adjacent nodes.

```
domain = c(users, "B")
range = sample(hcl.colors(15, palette = "viridis"), 15, replace = FALSE)
```

```
cols <- tibble(d = domain, r = range)
cols
#> # A tibble: 15 x 2
#>    d                    r
#>    <chr>                <chr>
#>  1 Marine Design        #25C771
#>  2 Water Quality User   #FDE333
#>  3 Shore Design         #D4E02D
#>  4 Shoreline Protection #4B0055
#>  5 Municipal            #A6DA42
#>  6 Resilience Planning  #008A98
#>  7 Marine Resources     #3C3777
#>  8 Emergency Response   #73D25B
#>  9 Port safety          #1E4D85
#> 10 Aquaculture          #009B95
#> 11 Fishing              #00AC8E
#> 12 Marine Industry      #00BA82
#> 13 Recreation           #007796
#> 14 Energy               #006290
#> 15 B                    #471D67
```

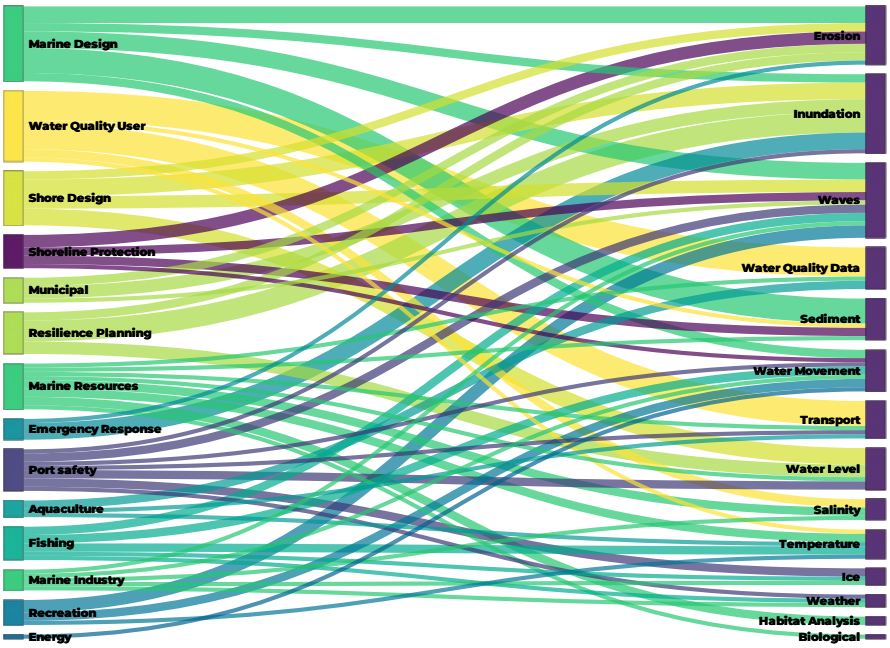
The colors are assembles as a string, which takes some fussy work to do programmatically.

```
the_colors <- paste0('d3.scaleOrdinal() .domain(["',
                     paste(cols$d, collapse='", "'),
                     '"]) .range(["',
                      paste(cols$r, collapse='", "'),
                     '"])')
cat(the_colors)
#> d3.scaleOrdinal() .domain(["Marine Design", "Water Quality User", "Shore Design", "Shoreline Protect
```

Revised Graphic

```
plt <- sankeyNetwork(Links = tmp3,
             Source = "User_Category",
             Target = "Data_Group",
             Value = "weight",
             LinkGroup = "link_group",
             NodeGroup = "groups",
             Nodes = nodes,
             NodeID = "node_name",
             colourScale = the_colors,
```

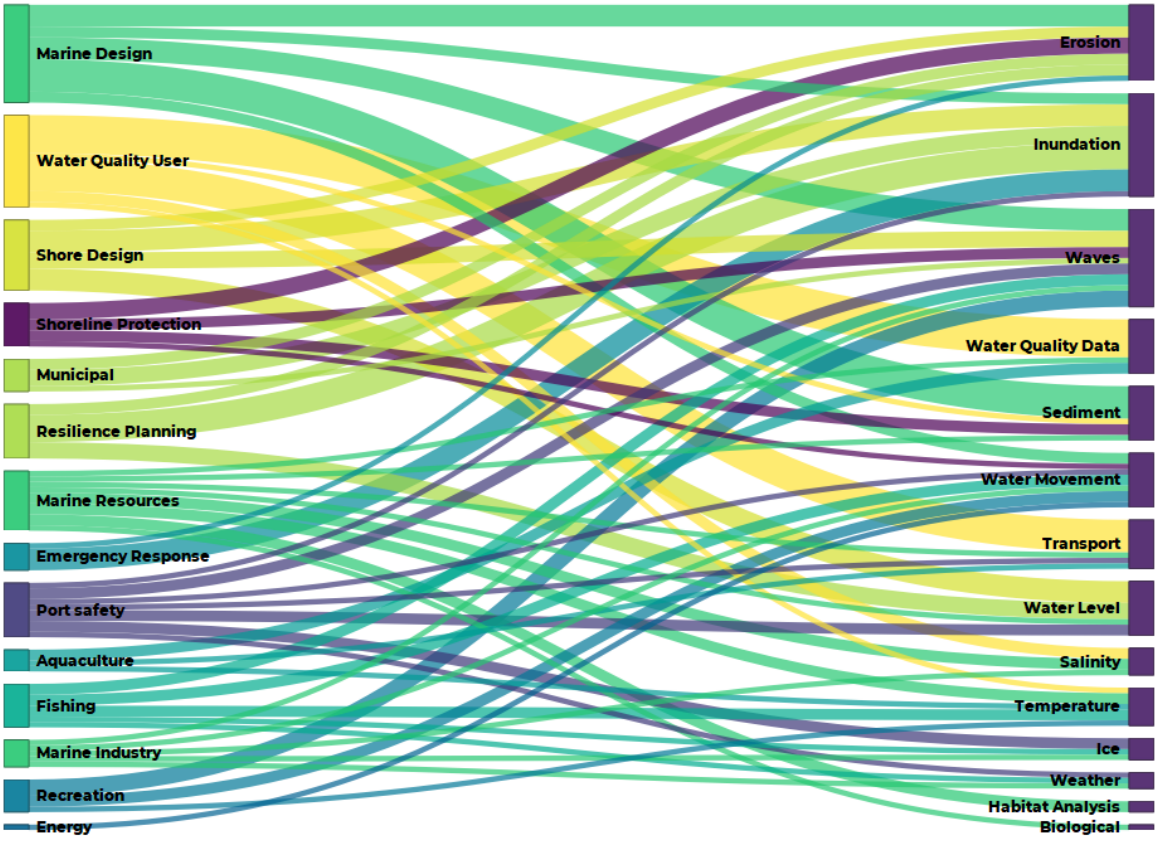
```
             nodeWidth = 20,
             nodePadding = 10,
             height = 700,
             fontSize = 12,
             fontFamily = 'Montserrat ExtraBold',
             iterations = 0)
#> Links is a tbl_df. Converting to a plain data frame.
#> Nodes is a tbl_df. Converting to a plain data frame.
plt
```




```
saveNetwork(plt, file = 'users and data better.html', selfcontained = TRUE)
```

```
webshot::webshot('users and data better.html', 'users and data better.png')
```

¶¶¶



From Data Requests to Time Domain

```
tmp <- the_data %>%
  select(ID, Data_Group, Data_Timing) %>%
  filter(! is.na(Data_Timing)) %>%
  filter(! Data_Timing == "Unclear or Unknown") %>%
  filter(! is.na(Data_Group)) %>%
  mutate(Data_Timing = if_else(Data_Timing == 'Other',
                               'Other_Timing', Data_Timing),
         Data_Group = if_else(Data_Group == 'Other',
                               'Other_Data',Data_Group)) %>%
  mutate(Data_Timing = fct_infreq(Data_Timing),
         Data_Group = fct_infreq(Data_Group)) %>%
  unique() %>%
  group_by(Data_Group, Data_Timing) %>%
  summarize(weight = n(),
            .groups = 'drop')
```

I generate new node group names.

```
types <- as.character(levels(tmp$Data_Group))
timing <- as.character(levels(tmp$Data_Timing))

nodes <- tibble(node_name = c(types, timing),
                groups = c(types,
                           rep("B", length(timing))))
```

And specify matching color groups for the left hand nodes. (Zero indexed.)

```
tmp3 <- tmp %>%
  mutate(link_group = as.character(Data_Group),
         Data_Timing = match(Data_Timing, nodes$node_name) - 1,
         Data_Group = match(Data_Group, nodes$node_name) - 1)
```

Now I have to define my color palette. This is some sort of web-based color palette definition. . . .

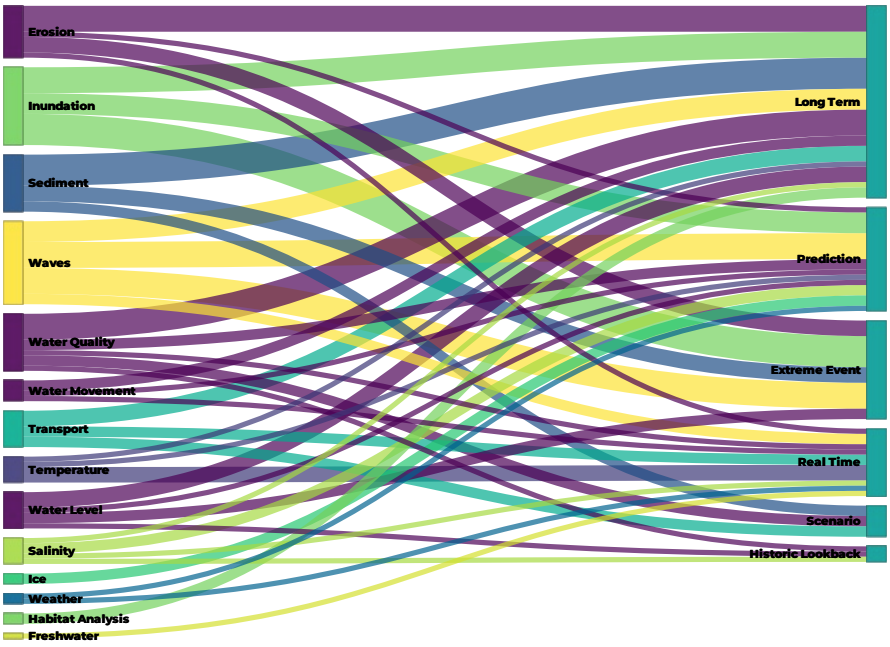
```
domain = c(types, "B")
range = sample(hcl.colors(15, palette = "viridis"), 15, replace = FALSE)
cols <- tibble(d = domain, r = range)
```

The colors are assembles as a string, which takes some fussy work to do programmatically.

```
the_colors <- paste0('d3.scaleOrdinal() .domain(["',
                     paste(cols$d, collapse='", "'),
                     '"]) .range(["',
                      paste(cols$r, collapse='", "'),
                     '"])')
cat(the_colors)
#> d3.scaleOrdinal() .domain(["Erosion", "Inundation", "Sediment", "Waves", "Water Quality", "Water Mov
```

Graphic

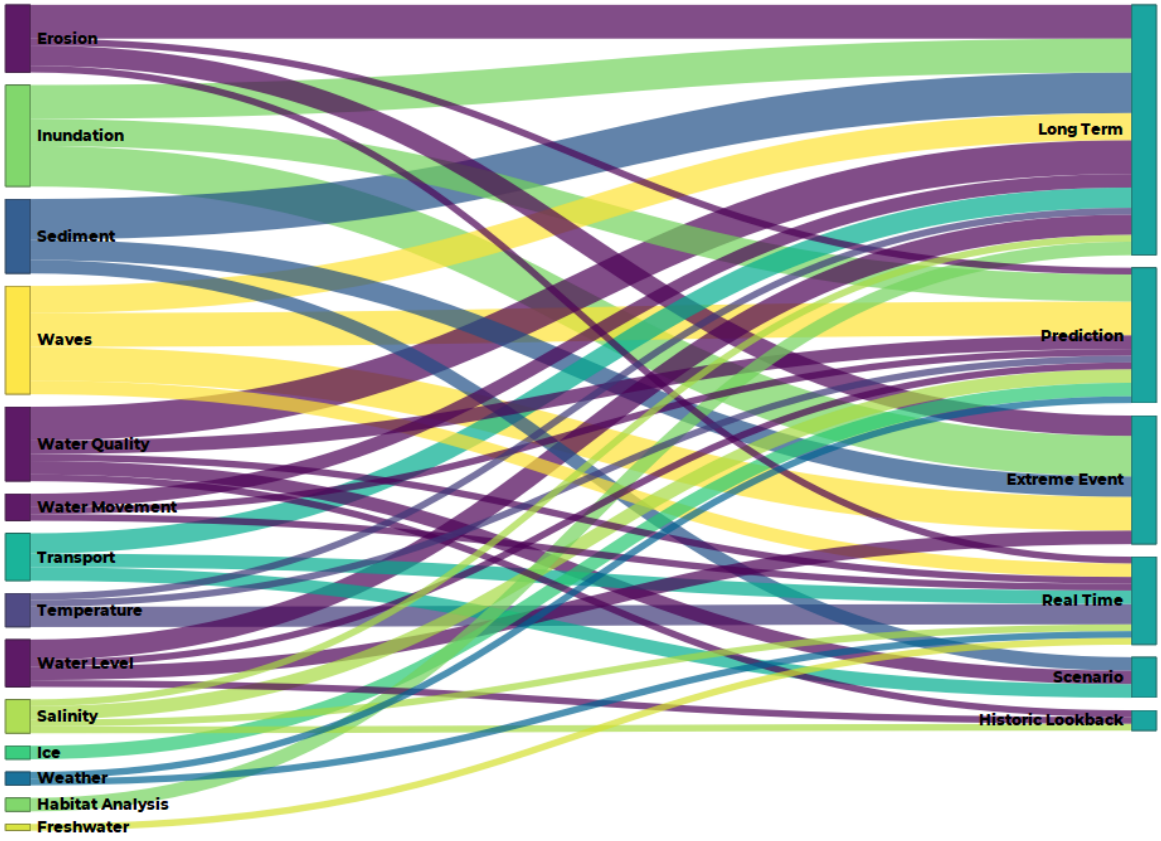
```
plt <- sankeyNetwork(Links = tmp3,
             Source = "Data_Group",
             Target = "Data_Timing",
             Value = "weight",
             LinkGroup = "link_group",
             NodeGroup = "groups",
             Nodes = nodes,
             NodeID = "node_name",
             colourScale = the_colors,
             nodeWidth = 20,
             nodePadding = 10,
             height = 700,
             fontSize = 12,
             fontFamily = 'Montserrat ExtraBold',
             iterations = 0)
#> Links is a tbl_df. Converting to a plain data frame.
#> Nodes is a tbl_df. Converting to a plain data frame.
plt
```




```
saveNetwork(plt, file = 'data to timing.html', selfcontained = TRUE)

webshot::webshot('data to timing.html', 'data to timing.png')
```

¶¶¶



From Time Domain to Data Requests

```
types <- as.character(levels(tmp$Data_Group))
timing <- as.character(levels(tmp$Data_Timing))

nodes <- tibble(node_name = c(timing, types),
                groups = c(timing,
                           rep("B", length(types))))
```

And specify matching color groups for the left hand nodes. (Zero indexed.)

```
tmp3 <- tmp %>%
  mutate(link_group = as.character(Data_Timing),
         Data_Timing = match(Data_Timing, nodes$node_name) - 1,
         Data_Group = match(Data_Group, nodes$node_name) - 1)
```

Now I have to define my color palette.

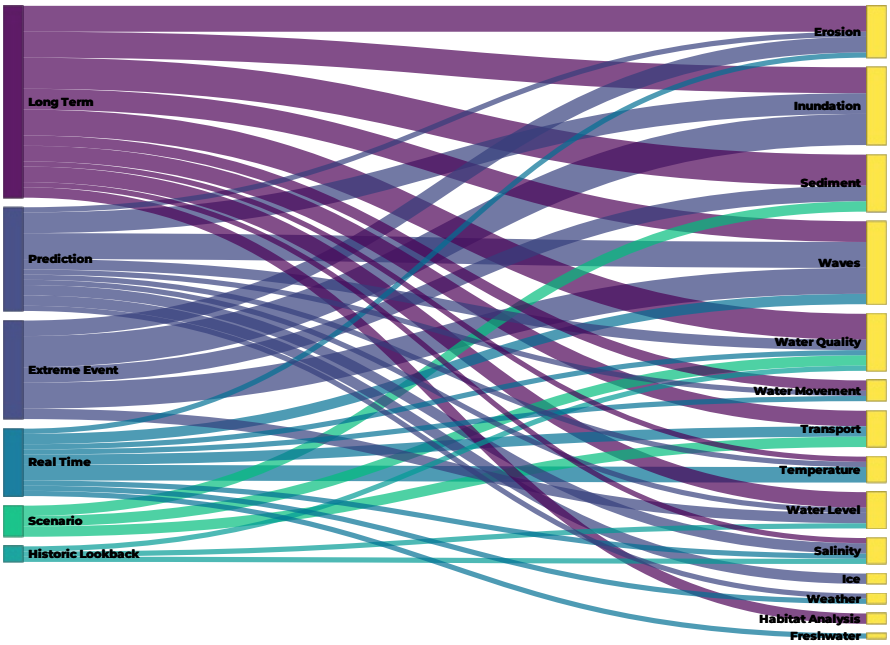
```
domain = c(timing, "B")
range = hcl.colors(7, palette = "viridis")
cols <- tibble(d = domain, r = range)
```

The colors are assembles as a string, which takes some fussy work to do programmatically.

```
the_colors <- paste0('d3.scaleOrdinal() .domain(["',
                     paste(cols$d, collapse='", "'),
                     '"]) .range(["',
                      paste(cols$r, collapse='", "'),
                     '"])')
cat(the_colors)
#> d3.scaleOrdinal() .domain(["Long Term", "Prediction", "Extreme Event", "Real Time", "Scenario", "His
```

Graphic

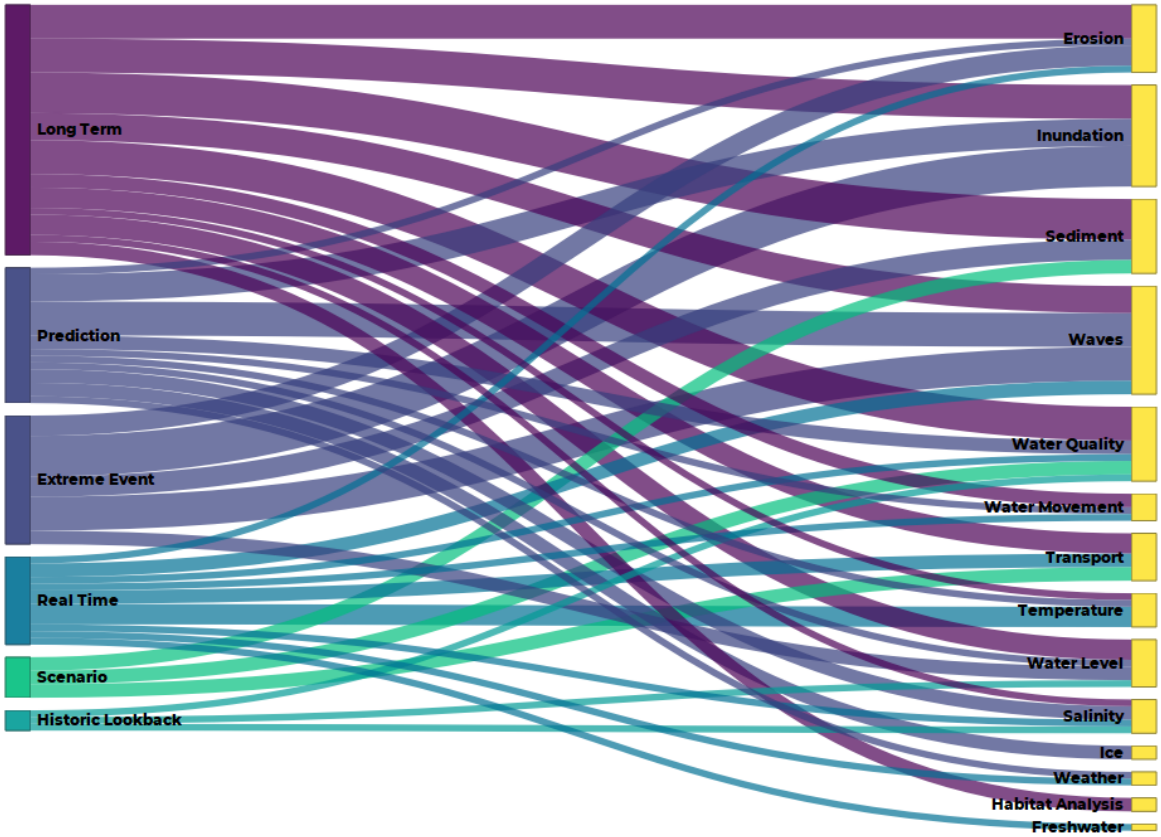
```
plt <- sankeyNetwork(Links = tmp3,
              Source = "Data_Timing",
              Target = "Data_Group",
              Value = "weight",
              LinkGroup = "link_group",
              NodeGroup = "groups",
              Nodes = nodes,
              NodeID = "node_name",
              colourScale = the_colors,
              nodeWidth = 20,
              nodePadding = 10,
              height = 700,
              fontSize = 12,
              fontFamily = 'Montserrat ExtraBold',
              iterations = 0)
#> Links is a tbl_df. Converting to a plain data frame.
#> Nodes is a tbl_df. Converting to a plain data frame.
plt
```




```
saveNetwork(plt, file = 'timing to data.html', selfcontained = TRUE)

webshot::webshot('timing to data.html', 'timing to data.png')
```

¶¶¶


Long Term
Prediction
Extreme Event
Real Time
Scenario
Historic Lookback
Erosion
Inundation
Sediment
Waves
Water Quality
Water Movement
Transport
Temperature
Water Level
Salinity
Ice
Weather
Habitat Analysis
Freshwater