Analysis of Data and Model Timing

Curtis C. Bohlen

2022-12-09

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 six characteristics:

- Potential users and uses of hydrodynamic models,

- Data or information needs identified by community members,
- Implied extensions of the initial Casco Bay Model required to fully address those data needs, and
- Ideas for improving communications of model results (e.g., communications channels and user interface design),
- Specifications for model performance or capabilities such as resolution, geographic coverage or ability to conduct simulations.
- Suggestions about monitoring or data collection that could improve information availability.

If a paragraph or live note included something relevant to one or more of these categories, we summarized the related idea, and then assigned each paragraph or comment to categories. In this way we cn look at what ideas were expressed most commonly during the workshops.

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 the time domains related to addressing the problems identified in workshop notes.

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())
```

Create Figures Folder

```
dir.create(file.path(getwd(), 'figures'), showWarnings = FALSE)
```

Load Data

```
the_data <- read_excel("Data_Export_Query.xlsx" ) %>%
  mutate(ID = as.integer(ID)) %>%
  rename_with(function(x) sub(" Category_Category", '_Category', x)) %>%
```

```
  rename_with(function(x) sub(" ", '_', x))
head(the_data)
#> # A tibble: 6 x 16
#>      ID Category   Day   Comment User_~1 Inter~2 Inter~3 Data_~4 Data_~5 Exten~6
#>   <int> <chr>      <chr> <chr>   <chr>   <chr>   <chr>   <chr>     <dbl> <chr>
#> 1     1 Live Comm~ Day ~ How ca~ Urban ~ Simple  Exampl~ <NA>         NA Draina~
#> 2     2 Live Comm~ Day ~ Use of~ Harbor~ <NA>    <NA>    Waves         4 <NA>
#> 3     2 Live Comm~ Day ~ Use of~ Marine~ <NA>    <NA>    Waves         4 <NA>
#> 4     2 Live Comm~ Day ~ Use of~ Urban ~ <NA>    <NA>    Waves         4 <NA>
#> 5     3 Live Comm~ Day ~ MS4 pr~ Water ~ Inform~ Shared~ <NA>         NA Discha~
#> 6     3 Live Comm~ Day ~ MS4 pr~ Water ~ Inform~ Shared~ <NA>         NA Draina~
#> # ... with 6 more variables: Extension_Timing <dbl>, Monitoring_Category <chr>,
#> #   Monitoring_Data_Group <dbl>, Data_Quality_Category <chr>,
#> #   Data_Quality_Type <dbl>, Data_Quality_Timing <dbl>, 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"))
data_types_table <- read_excel("Data Type.xlsx",
    col_types = c("numeric", "text", "text"))
```

Numerical Values to Strings

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)],
         Data_Quality_Timing = timing_table$Timing[match(Data_Quality_Timing,
                                                      timing_table$ID)]) %>%
  mutate(Monitoring_Data_Group = data_types_table$Group[match(Monitoring_Data_Group,
                                                    data_types_table$ID)],
         Data_Quality_Type = data_types_table$Group[match(Data_Quality_Type,
                                                    data_types_table$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.

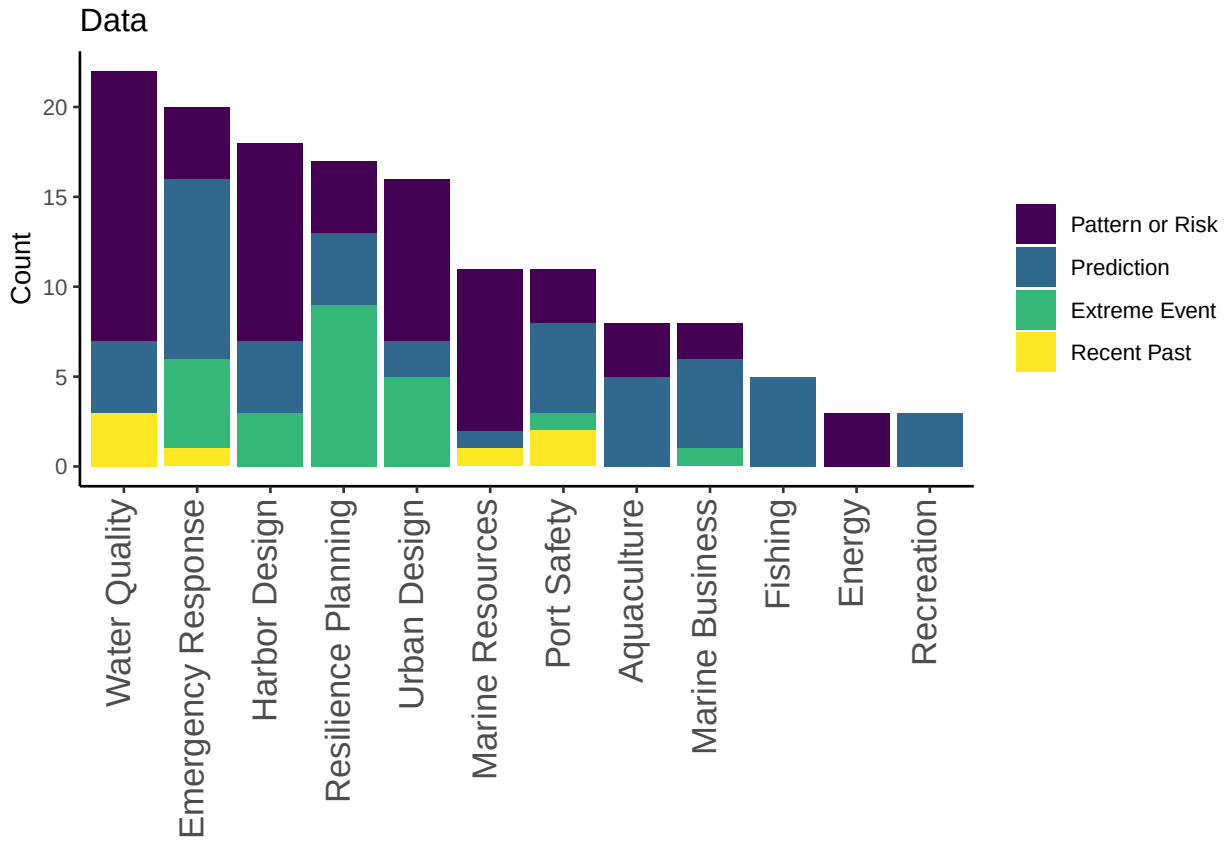
```
cat("All rows in the data:\t\t")
#> All rows in the data:
nrow(the_data)
```

```
#> [1] 376

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

Users and Data Timing

```
tmp <- the_data %>%
  select(ID, User_Category, Data_Timing) %>%
  unique() %>%
  mutate(User_Category = fct_infreq(User_Category),
         Data_Timing = fct_infreq(Data_Timing))
xtabs(~ User_Category + Data_Timing, tmp)
#>                      Data_Timing
#> User_Category         Pattern or Risk Prediction Extreme Event Unclear
#>   Water Quality                    15          4             0       6
#>   Urban Design                      9          2             5       1
#>   Resilience Planning               4          4             9       0
#>   Emergency Response                4         10             5       1
#>   Harbor Design                    11          4             3       0
#>   Port Safety                       3          5             1       2
#>   Marine Resources                  9          1             0       1
#>   Aquaculture                       3          5             0       0
#>   Marine Business                   2          5             1       0
#>   Educators                         0          0             0       0
#>   Fishing                           0          5             0       1
#>   Recreation                        0          3             0       0
#>   Energy                            3          0             0       0
#>   Other                             0          0             0       0
#>                      Data_Timing
#> User_Category         Recent Past
#>   Water Quality                 3
#>   Urban Design                  0
#>   Resilience Planning           0
#>   Emergency Response            1
#>   Harbor Design                 0
#>   Port Safety                   2
#>   Marine Resources              1
#>   Aquaculture                   0
#>   Marine Business               0
#>   Educators                     0
#>   Fishing                       0
#>   Recreation                    0
#>   Energy                        0
#>   Other                         0
```

```
reduced_data <- the_data %>%
  select(ID, User_Category, Data_Timing) %>%
  unique() %>%
  filter(! is.na(User_Category),
         ! is.na(Data_Timing),
         User_Category != "Other",       # Dropped because none have timing
         Data_Timing != 'Unclear') %>%  # Dropped because uninformative.
  mutate(User_Category = fct_infreq(User_Category),
         Data_Timing = fct_infreq(Data_Timing))

  ggplot(reduced_data, aes(User_Category)) +
  geom_bar(aes(fill = Data_Timing)) +
  theme(axis.text.x = element_text(angle = 90, size = 14,
                                   hjust = 1, vjust = 0.25)) +
  scale_fill_viridis_d(name = '') +
  ylab('Count') +
  xlab("") +
  ggtitle("Data")
```




```
ggsave('figures/user_timing.png', type='cairo',
         width = 7, height = 5)
```

Functions for Generating the Graphics

I developed three functions in the "Sankey Plots.Rmd" notebook. Here I just recreate them locally for convenience. The three functions are:

1. A function to assemble the `D3` color function call
2. A function to convert our source data frames, with a column of source and target names, into the format required by `sankeyNetwork()`.
3. A function that actually generates the Sankey Plot (and calls the other two functions).

Define Color String

```
d3_colors <- function(left_names, right_name, final_color = NULL) {
  #left_names is a vector of string node labels used for the source nodes.
  #right.name is the name for the (single) target node group.
  #final_color is a string defining a color for the right (target) nodes.

  domain = c(left_names, right_name)
  grps <- length(left_names) + 1
  #browser()
  if(! is.null(final_color)) {
    range <- sample(hcl.colors(grps-1, palette = "viridis"), grps - 1,
                    replace = FALSE)
    range <- c(range,final_color)
  }
  else
    #print(grps)
    range <- sample(hcl.colors(grps, palette = "viridis"), grps,
                    replace = FALSE)
  cols <- tibble(d = domain, r = range)
  the_colors <- paste0('d3.scaleOrdinal() .domain(["',
                       paste(cols$d, collapse='", "'),
                       '"]) .range(["',
                       paste(cols$r, collapse='", "'),
                       '"])')
  return(the_colors)
}
```

Assemble the Nodes and Links Data Frames

```
assemble_frames <- function(.dat, .left, .right,
                            right_name = 'right') {
  # .data is the RAW data with left-right string pairs (in two variables) that
  #  show a link between source (.left) and target (.right).

  grouped <- .dat %>%
    group_by({{ .left}}, {{.right}}) %>%
    summarize(weight = n(),
```

```
              .groups = 'drop')

  # browser()
  # there is probably a more efficient way to extract labels, that avoids
  # building an unnecessary data frame, but this works, and lets me use
  # tidyverse indirection....

  labs <- grouped %>%
    mutate(left = factor({{ .left}}),
           right = factor({{.right}}))

  #I can ignore order here because the Sankey function does....
  # By calling `factor()` here, it re-levels and drops empty categories.
  left = levels(factor(labs$left))
  right = levels(factor(labs$right))
  rm(labs)

  nodes <- tibble(node_name = c(left, right),
                  groups = c(left,
                             rep(right_name, length(right))))
  links <- grouped %>%
    mutate(link_group = as.character({{.left}}),
           "{{.left}}" := match({{.left}}, nodes$node_name) - 1,
           "{{.right}}":= match({{.right}}, nodes$node_name) - 1)

  return(list(Nodes = as.data.frame(nodes), Links = as.data.frame(links)))
}
```

Draw Sankey Plot

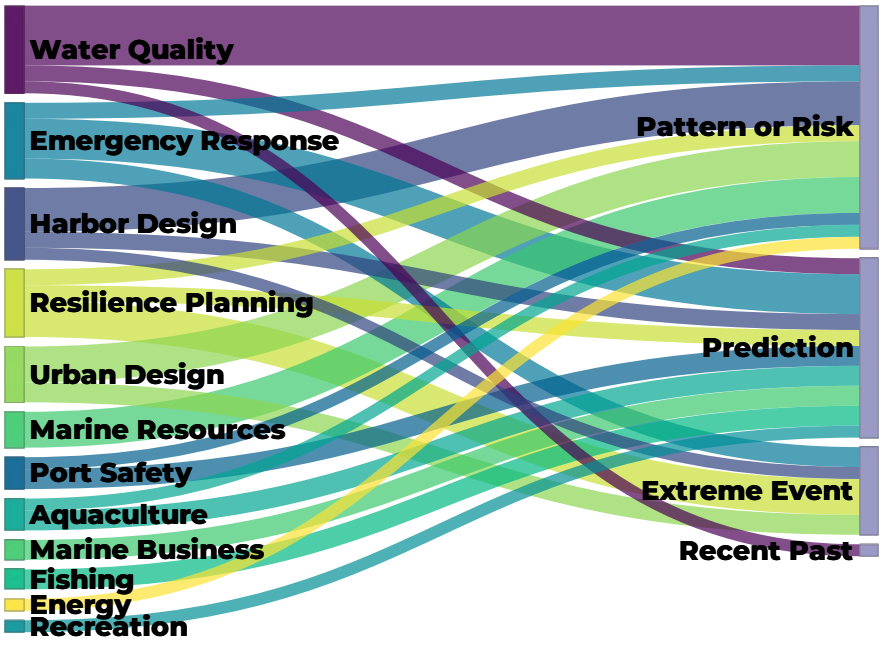
This encapsulates data frame preparation, color assignment and my selected plot characteristics.

```
my_sankey <- function(.dat, .left, .right,
                      final_color = NULL, drop_below = NULL) {

  right_name <- 'right'
  left_str <- as.character(ensym(.left))
  right_str <- as.character(ensym(.right))

  if (! is.null(drop_below)) {
    .dat <- .dat %>%
      group_by({{.left}}, {{.right}}) %>%
      mutate(links = n()) %>%
      filter(links >= drop_below)%>%
      select(-links)
  }

  my_data <- assemble_frames(.dat, {{.left}}, {{.right}},
                             right_name = right_name)

  left <- my_data$Links %>%
    mutate(left = as.character(link_group)) %>%
```

```
    pull(left)
  left <- unique(as.character(left))

  the_colors <- d3_colors(left, right_name = right_name,
                          final_color = final_color)
  #browser()
  the_graphic <- sankeyNetwork(Links = my_data$Links,
              Source = left_str,
              Target = right_str,
              Value = "weight",
              LinkGroup = "link_group",
              NodeGroup = "groups",
              Nodes = my_data$Nodes,
              NodeID = "node_name",
              colourScale = the_colors,
              nodeWidth = 20,
              nodePadding = 10,
              height = 700,
              fontSize = 28,
              fontFamily = 'Montserrat ExtraBold',
              iterations = 0)

 return(the_graphic)
}
```

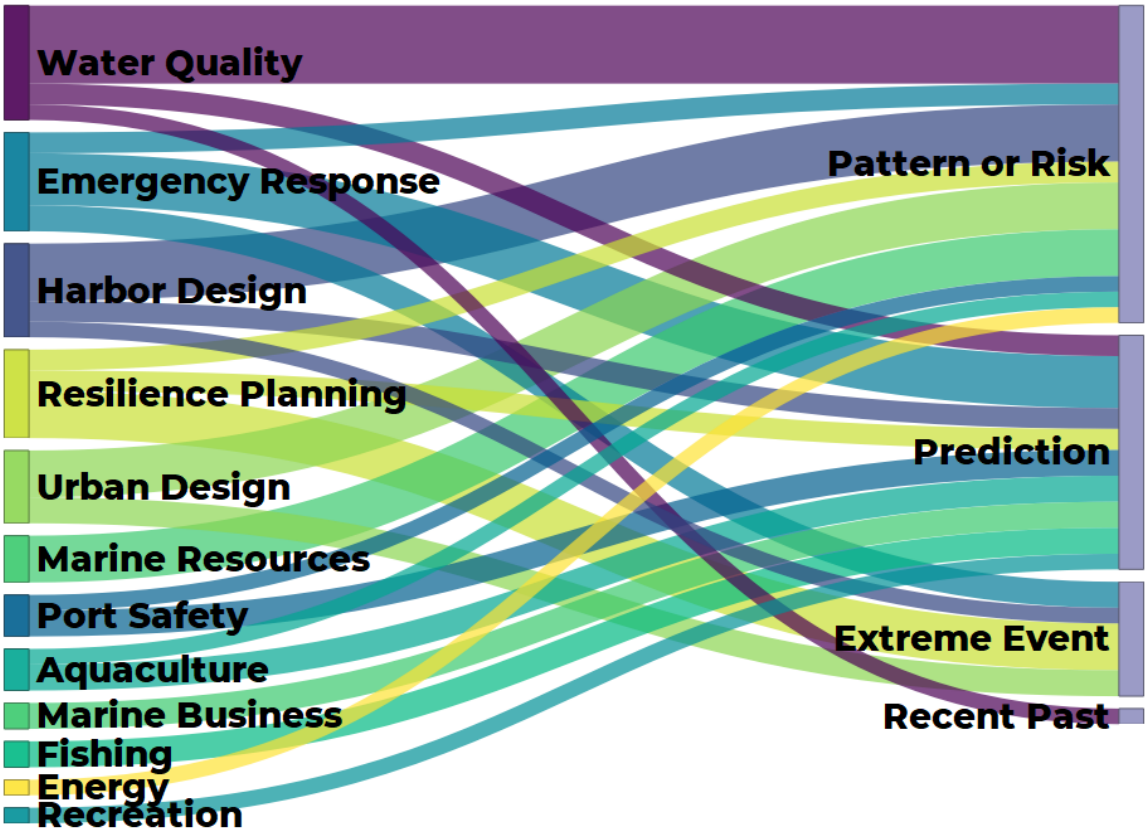
Generate Sankay Graphics

```
plt <- reduced_data %>%
  my_sankey(User_Category, Data_Timing, final_color = "#9090c0", drop_below = 3)
plt
```

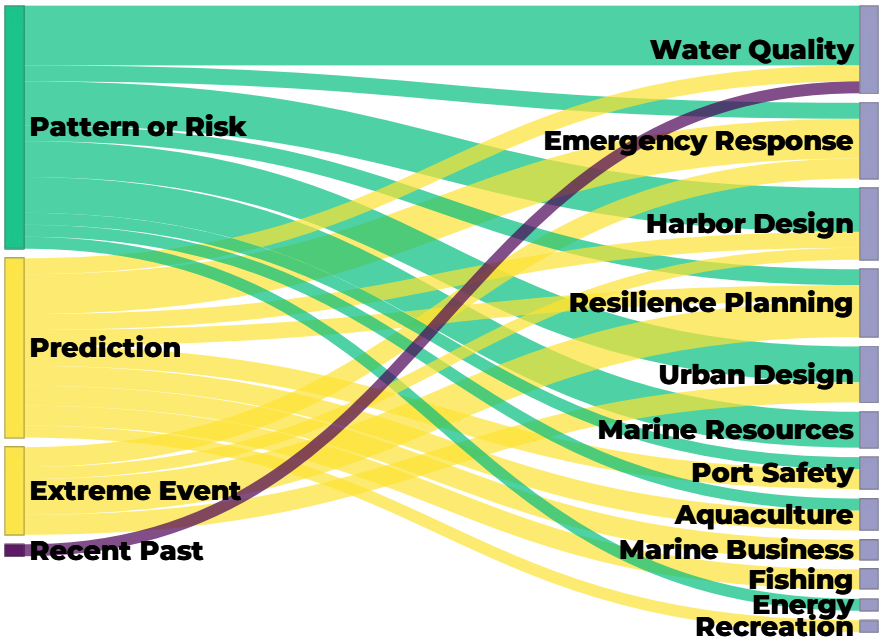



```
saveNetwork(plt, file = 'figures/user timing.html', selfcontained = TRUE)
webshot::webshot('figures/user timing.html', 'figures/data timing.png')
```

¶¶¶




```
reduced_data %>%
  my_sankey(Data_Timing, User_Category, final_color = "#9090c0", drop_below = 3)
```

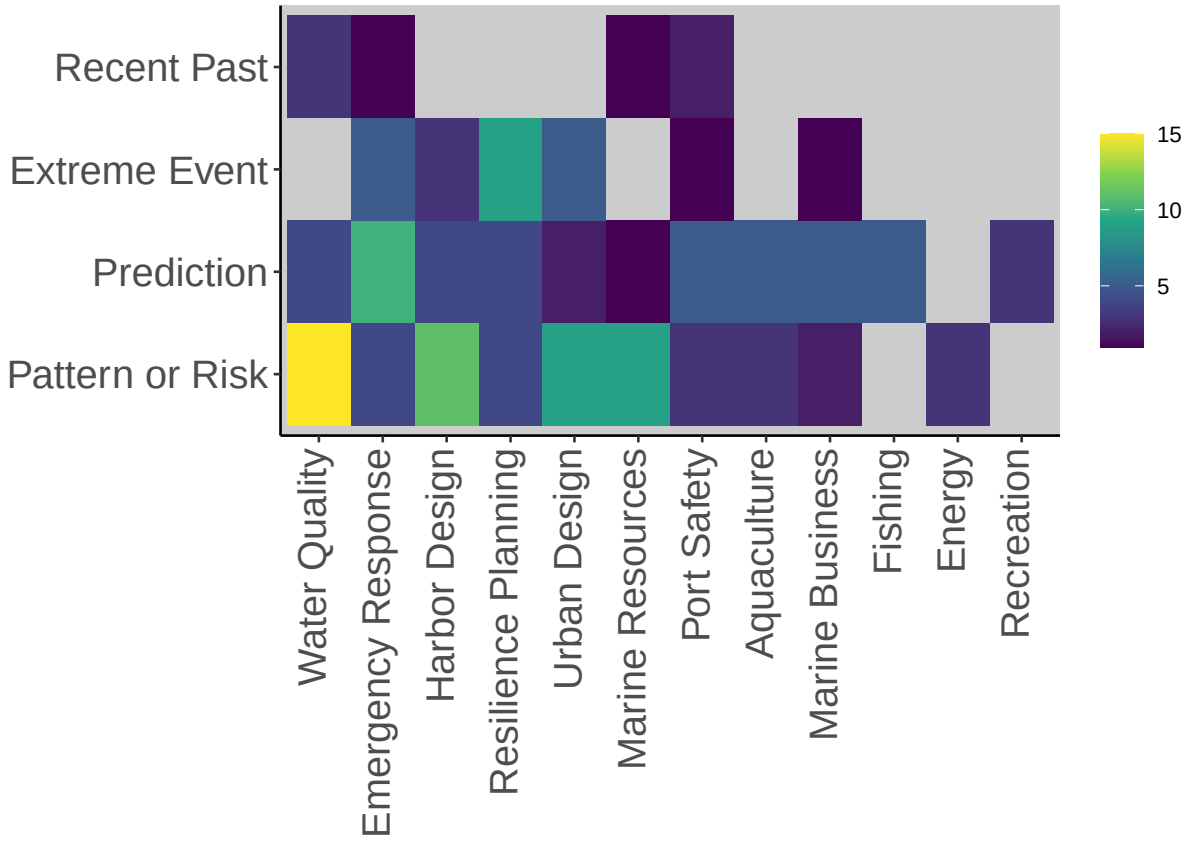


Matrix Plot

The idea here is to make a visual matrix, tiled with colors

```
tmp <- reduced_data %>%
  group_by(User_Category, Data_Timing) %>%
  summarize(count = n(), .groups = 'drop')
```

```
tmp %>%
 # filter(count > 2) %>%
  ggplot(aes(User_Category, Data_Timing, fill = count)) +
  geom_tile() +
  scale_fill_viridis_c(name = '', option = "viridis") +
  theme(axis.text.x = element_text(angle = 90, size = 16, hjust = 1, vjust = 0.25),
        axis.text.y = element_text(size = 16),
        panel.background = element_rect(fill = 'grey80')) +
  xlab('') +
  ylab('')
```




```
ggsave('figures/user_timing_tiles.png', type='cairo',
         width = 7, height = 5)
```

Which Types of Users want Long Term Data?

```
reduced_data %>%
  filter(Data_Timing == 'Pattern or Risk') %>%
  group_by(User_Category) %>%
  summarise(count = n()) %>%
  filter(count>0) %>%
  pull(User_Category)
#>  [1] Water Quality       Emergency Response  Harbor Design
#>  [4] Resilience Planning Urban Design        Marine Resources
#>  [7] Port Safety         Aquaculture         Marine Business
#> [10] Energy
#> 12 Levels: Water Quality Emergency Response ... Recreation
```