Bootcamp 5: Sankey plots in plotly

Lucien Baumgartner

10/28/2018

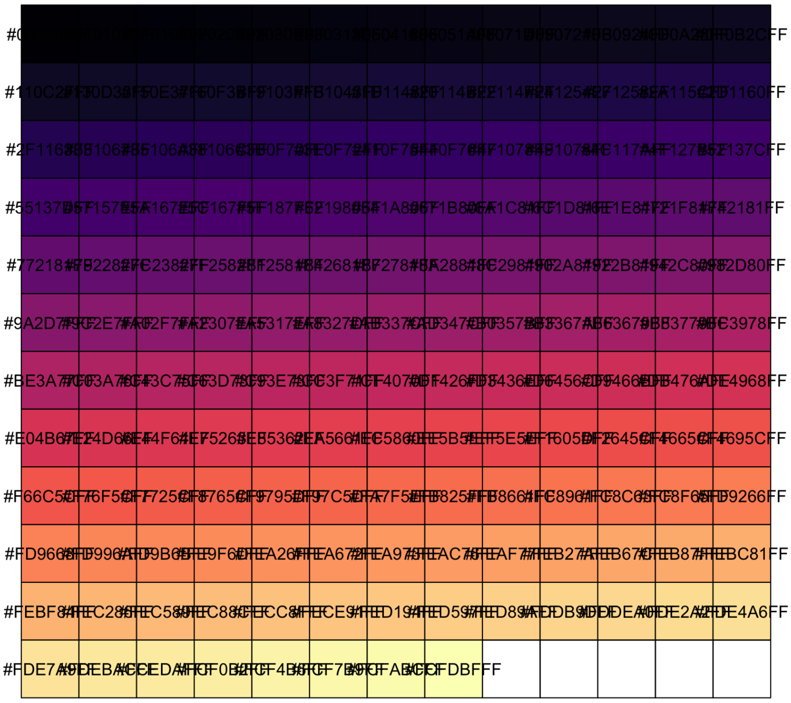
```
library(dplyr) # data crunching
library(viridis) # color scale for lazy people
library(tidyr) # some more data crunching
library(plotly) # plots
library(scales) # additional plotting stuff

# authorization tokens for the plotly API
Sys.setenv("plotly_username"="ddj18")
Sys.setenv("plotly_api_key"="lbcb2dOtKkcBj0sZN22E")

# clean workspace
rm(list=ls())

source('~/r-helpers/ggplot/ggplot-helper.R')

# set WD
setwd('~/ddj18/output/')

# load pop data
load('01-bevoelkerung-clean.RData')
bev <- df
load('01-umzug-clean.RData')
umz <- df
rm(df)
str(bev)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    9437083 obs. of  15 variables:
##  $ persnum        : int  911747 886098 347792 1073886 17361 966399 604793 797443 552749 157071 ...
##  $ anzbestwir     : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ stichtagdatjahr: int  1993 1993 1993 1993 1993 1993 1993 1993 1993 1993 ...
##  $ alterv05kurz   : chr  "40-44" "25-29" "20-24" "20-24" ...
##  $ sexcd          : int  1 1 1 2 1 1 2 2 1 1 ...
##  $ aufart2lang    : chr  "SchweizerIn" "andere" "andere" "SchweizerIn" ...
##  $ ziv2lang       : chr  "Ledig" "Ledig" "Ledig" "Ledig" ...
##  $ anzahlkinder   : int  0 0 0 0 0 0 3 0 0 1 ...
##  $ hhtyplang      : chr  NA NA NA NA ...
##  $ geblandhistlang: chr  "Asien" "Asien" "Asien" "Asien" ...
##  $ nationhistlang : chr  "Schweiz" "Asien" "Asien" "Schweiz" ...
##  $ kreislang      : chr  "Kreis 2" "Kreis 2" "Kreis 10" "Kreis 10" ...
##  $ quarlang       : chr  "Enge" "Enge" "Wipkingen" "Höngg" ...
##  $ gebnum         : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ ewid           : int  NA NA NA NA NA NA NA NA NA NA ...
```

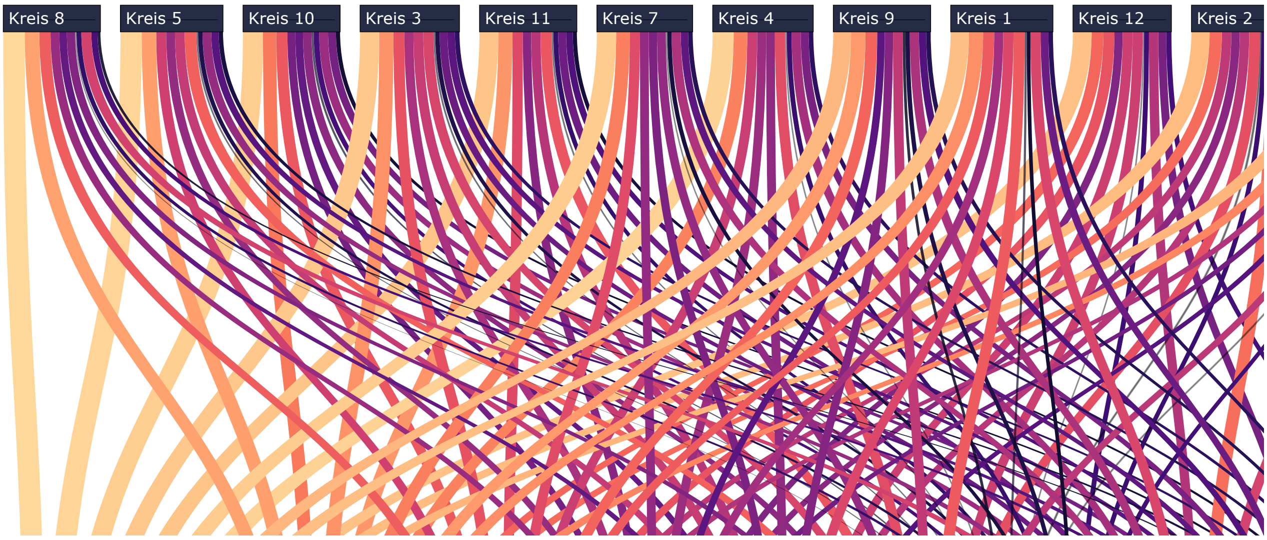
```
str(umz)
```

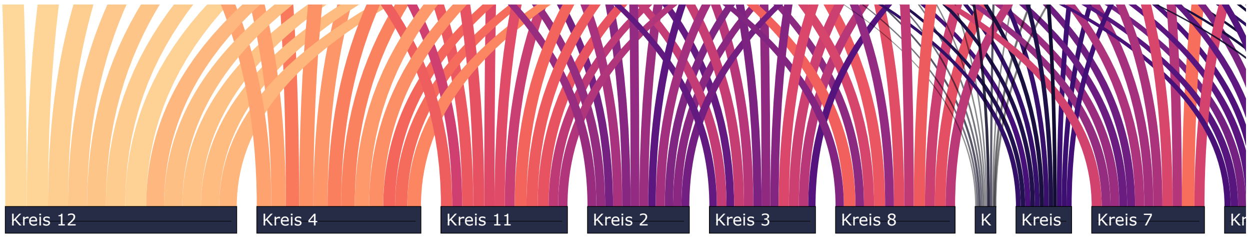
```
## Classes 'tbl_df', 'tbl' and 'data.frame':    45348 obs. of  5 variables:
##  $ persnum        : int  438840 419130 378791 388855 408979 368870 429222 348561 383728 353684 ...
##  $ anzumzuwir     : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ stichtagdatjahr: int  2015 2015 2015 2015 2015 2015 2015 2015 2015 2015 ...
##  $ kreisbisherlang: chr  "Kreis 11" "Kreis 2" "Kreis 5" "Kreis 11" ...
##  $ quarbisherlang : chr  "Seebach" "Enge" "Escher Wyss" "Oerlikon" ...
```

```
# subset bev for 2015 and persnum in umz
bev <- filter(bev, stichtagdatjahr==2015&persnum%in%umz$persnum)

# join umz with bev based on persnum
if(length(unique(bev$persnum))==length(bev$persnum)) df <- full_join(bev, umz, by=c('persnum', 'stichtagdatjahr'))

# now kreislang indicates the district they moved to, while kreisbisherlang is where the move away from
# let's count the district combinations and compute some aggregates
# I decided to go for something very simple:
# share of people that move from district X == 100% for every X
# -> now where do they move to
# the idea is simple, but it is hard to understand for the reader,
# because the aggregates on the destination side are sums of shares with different roots (so that !2%_x1==2%_2)
# nvm, i'ts just to show you

aggr <- df %>%
  # mutate(move=paste0(kreisbisherlang, '_', kreislang)) %>%
  # group_by(move) %>%
  group_by(kreisbisherlang, kreislang) %>% # group by source and target
  summarise(n_move=n()) %>% # counts
  mutate(p_move=n_move/sum(n_move)) %>% # percentages
  # separate(move, c('kreisbisherlang', 'kreislang'), '_') %>%
  # mutate(move=paste0(kreisbisherlang, '_', kreislang)) %>%
  # this is some additional stuff I did in the beginning but are only needed for some initial plots
  # I just leave it here for the sake of completeness
  left_join(., df %>%
              group_by(kreisbisherlang) %>%
              summarise(n_bisher=n()) %>%
              mutate(p_bisher=n_bisher/sum(n_bisher)),
            by='kreisbisherlang') %>%
  left_join(., df %>%
              group_by(kreislang) %>%
              summarise(n_now=n()) %>%
              mutate(p_now=n_now/sum(n_now)),
            by='kreislang') %>%
  ungroup %>%
  # factorize
  mutate(kreisbisherlang=factor(kreisbisherlang, levels = unique(kreisbisherlang)[order(table(kreisbisherlang))]),
         kreislang=factor(kreislang, levels = unique(kreislang)[order(table(kreislang))]))

# I'll work with the viridis color package to assign each share a specific color
show_col(viridis_pal(opt='A')(length(unique(aggr$p_move))))
```



```
# join in color variables
aggr <- left_join(aggr,
                  tibble(p_move=sort(unique(aggr$p_move)),
                         color=viridis_pal(opt='A')(length(unique(aggr$p_move))), # hex values
                         color_rgba=toRGB(color, alpha = 1*(p_move*100))), # rgba values; you can dynamically adjus
                  by='p_move')

# call a plotly environment
p <- plot_ly(
  type = "sankey", # sankey plot
  orientation = "v", # vertical orientation
  # panel dimensions
  domain = list(
    x =  c(0,1),
    y =  c(0,1)
  ),
  showlegend = F, # no legend
  valuesuffix = "%", # add % to the value showed in the hover element
  # this pretty much is the aes() component of ggplot for the nodes:
  node = list(
    label = c(levels(aggr$kreisbisherlang), levels(aggr$kreislang)), # bucket labels
    color = rep('#262C46', length(aggr$kreislang)), # same color for all of them
    # additional parameters
    pad = 15,
    thickness = 20,
    line = list(
      color = "black",
      width = 0.5
    )
  ),
  # this pretty much is the aes() component for the strings
  # I'll explain it
  link = list(
    source = as.numeric(aggr$kreisbisherlang),
    target = max(as.numeric(aggr$kreisbisherlang))+as.numeric(aggr$kreislang),
    value =  aggr$p_move*100,
    color = aggr$color_rgba

  )
) %>%
  # additonal stuff
  layout(
    title = "Umzüge in der Stadt Zürich",
    font = list(
      size = 12,
      color = 'white'
    ),
    margin = list(
      l = 50,
      r = 50,
      b = 100,
      t = 100,
      pad = 4
    ),
    plot_bgcolor = 'black',
    paper_bgcolor = 'black'
  )
p
```





```
# add it to the online dashboard
chart_link = api_create(p, filename="umzuege-zh-p-per-kreis")
```