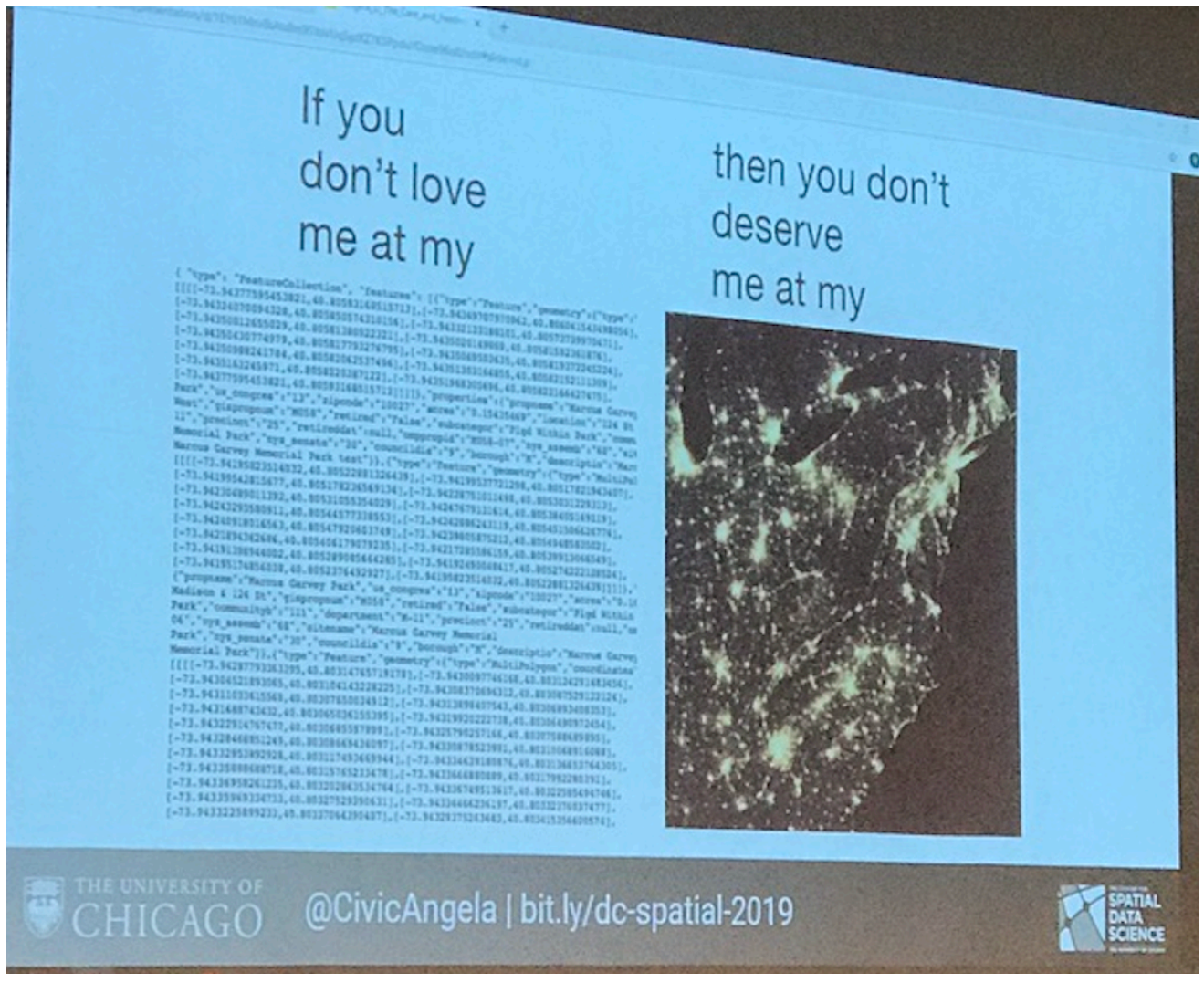
DC R Recap

Yiyuan Wu, MS

Dec 3, 2019

FUN PIC!


If you
don't love
me at my
then you don't
deserve
me at my
THE UNIVERSITY OF
CHICAGO
@CivicAngela | bit.ly/dc-spatial-2019
SPATIAL
DATA
SCIENCE

What are some highlights for me?

- 😄 **Food** breakfast & lunch! 🦑 🍵
- Many beautiful plots
- The opening song is **R song (1:18)** (https://www.youtube.com/watch?v=uDSTKFeZ8jM)
- Speakers will present their package or project by doing a **20 mins talks** , this R conference co wide range of topics in R
- **TidyTuesday** : A weekly data project in R from the R4DS online learning community
- People twitts! 👍

Overview of packages

1. `parsnip` : part of tidymodels. Contains wrappers for a number of models. easy syntax. githu (https://tidymodels.github.io/parsnip/articles/articles/Models.html) tidyverse.org/blog (https://www.tidyverse.org/blog/2018/11/parsnip-0-0-1/)
2. R packages for working with APIs `httr` (https://github.com/r-lib/httr), `curl`, `jsonlite`
3. `flexdashboard` package: easy interactive dashboards for R rstudio-flexdashboard (https://rmarkdown.rstudio.com/flexdashboard/)
4. `plotly` package : To make ggplot become interactive.
5. Geospatial packages `sf` `tmap` `spdep` `sp`
6. `gganimate` package: how to create plots with animation article (https://www.datanovia.com/en/blog/gganimate-how-to-create-plots-with-beautiful-animati
7. `textmineR` for textmining and NLP using R link (https://cran.rstudio.com/web/packages/textmineR/vignettes/c_topic_modeling.html)
8. CHEAT::SHEETS!

Our focus Today

1. `funneljoin` package

   Speaker: Emily Robinson, Data Scientist at DataCamp

   Developed by Anthony Baker, David Robinson, and Emily Robinson

2. `tmap` package

   Speaker: Angela Li, Research Specialist at the Center for spatial data science Univerisy of Chicago

3. `networkD3` package: Create D3 network graphs *Speaker: Ami Gates at Georgetown University*

4. `tidyr`

   Speaker: David Robinson at Flatiron Health

What is `funneljoin`?

- Join tables based on events occuring in sequence
- To answer “first this than that” questions

  -> Who signed up after first clicking the ads?

  -> What movies did people watch in the last month before watching movies XYZ?

What is `funneljoin`?

Examples datasets

```
landed
## # A tibble: 9 x 2
##   user_id timestamp
##     <dbl> <date>
## 1       1 2018-07-01
## 2       2 2018-07-01
## 3       3 2018-07-02
## 4       4 2018-07-01
## 5       4 2018-07-04
## 6       5 2018-07-10
## 7       5 2018-07-12
## 8       6 2018-07-07
## 9       6 2018-07-08
```

```
registered
## # A tibble: 8 x 2
##   user_id timestamp
##     <dbl> <date>
## 1       1 2018-07-02
## 2       3 2018-07-02
## 3       4 2018-06-10
## 4       4 2018-07-02
## 5       5 2018-07-11
## 6       6 2018-07-10
## 7       6 2018-07-11
## 8       7 2018-07-07
```

What is `funneljoin`?

Examples code

- If we want to find out for a website, when was the user's first landing and their first registrat afterward?
- we would type codes as below:

```
landed %>%
arrange(timestamp)%>%
distinct(user_id, .keep_all = TRUE)%>%
left_join(registered, by="user_id") %>%
 filter(timestamp.y>=timestamp.x ) %>%
 arrange(timestamp.y) %>%
 distinct(user_id, .keep_all = TRUE)
## # A tibble: 5 x 3
##   user_id timestamp.x timestamp.y
##     <dbl> <date>      <date>
## 1       1 2018-07-01  2018-07-02
## 2       4 2018-07-01  2018-07-02
## 3       3 2018-07-02  2018-07-02
## 4       6 2018-07-07  2018-07-10
## 5       5 2018-07-10  2018-07-11
```

1. Filter `landed` dataset for first row per user
2. Left join with registered on user id
3. Filter for timestamp.y >= timestamp.x
4. Filter for first row of timestamp.y

What is `funneljoin`?

Examples code

- If we want to find out an website, when is the user's first click and their first registration afte
    - We wanted to get the first time people clicked website and the first time registration aft
      Choose the `after_inner_join()` with a `first-firstafter` type:

What is `funneljoin`?

Examples code

- We would type codes as below:

```
landed %>%
  after_inner_join(registered,
                   by_user = "user_id",
                   by_time = "timestamp",
                   type = "first-firstafter",
                   suffix = c("_landed", "_registered"))
## # A tibble: 5 x 3
##   user_id timestamp_landed timestamp_registered
##     <dbl> <date>           <date>
## 1       1 2018-07-01       2018-07-02
## 2       4 2018-07-01       2018-07-02
## 3       3 2018-07-02       2018-07-02
## 4       6 2018-07-07       2018-07-10
## 5       5 2018-07-10       2018-07-11
```

what about `after_left_join()`?

- Notice in result, it includes the user_id 2 because it is a left join

```
landed %>%
  after_left_join(registered,
                  by_user = "user_id",
                  by_time = "timestamp",
                  type = "first-firstafter",
                  suffix = c("_landed", "_registered"))
## # A tibble: 6 x 3
##   user_id timestamp_landed timestamp_registered
##     <dbl> <date>           <date>
## 1       1 2018-07-01       2018-07-02
## 2       2 2018-07-01       NA
## 3       4 2018-07-01       2018-07-02
## 4       3 2018-07-02       2018-07-02
## 5       6 2018-07-07       2018-07-10
## 6       5 2018-07-10       2018-07-11
```

`funneljoin` different join types

1. *first-firstafter*: Take the first x, then the first y after that
2. *first-first*: take the first x and first y by user
3. *lastbefore-firstafter*: First x that's followed by a y before the next x
4. *any-firstafter*: Take all Xs followed by the first Y after it
5. *any-any*: Take all Xs followed by all Ys

Next

Geo Package

Hello `tmap` !

- thematic maps
- can display both interactive and static
- most common data formats:

```
* .shp (shapefile)
* .geojson
* .csv
* .tiff (raster data)
```

tmap

Geocode

- What is `geocode_OSM` ?
    - OSM stands for OpenStreetMap
    - Nominatim: a search engine for OpenStreetMap data

```
 library(tmaptools)
 tmaptools::geocode_OSM("Weill Cornell Medical College")
## $query
## [1] "Weill Cornell Medical College"
##
## $coords
##         x         y
## -73.95491  40.76475
##
## $bbox
##      xmin      ymin      xmax      ymax
## -73.95496  40.76470 -73.95486  40.76480
```

`tmap`

basic syntax

- `tm_shape()`: specify shape object
- `tm_fill()`: fills the polygon
- `tm_borders` : draw borders of the polygons
- `qtm()`: quick thematic map plot
- `tmap_mode()`: plot for static or view for interactive

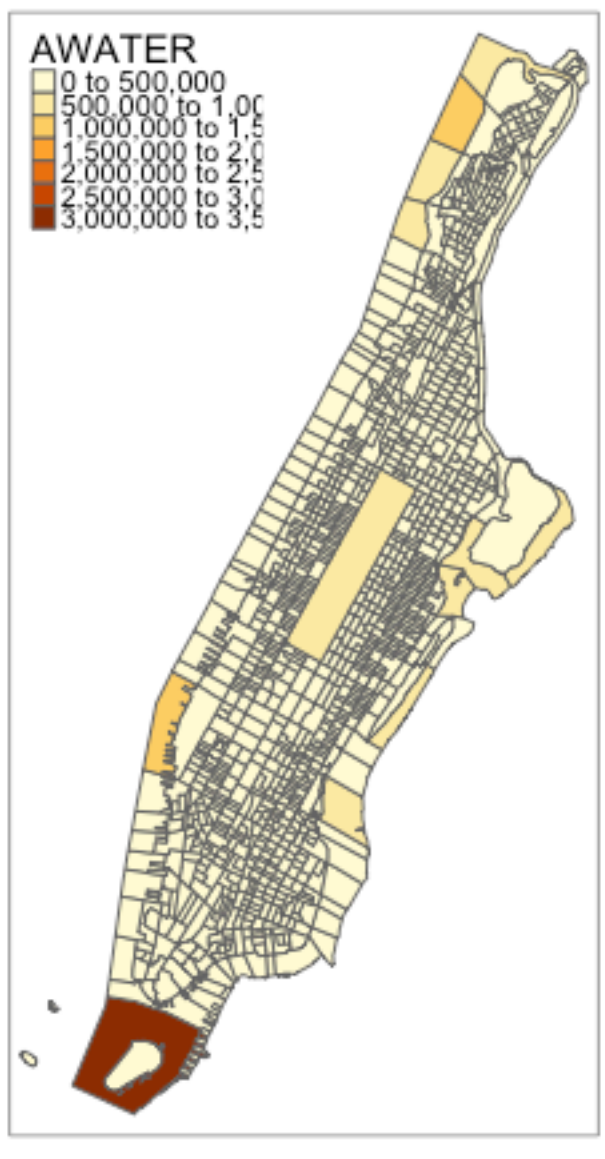
tmap

- static

```r
#nyc=st_read("Box/yi/2019 R conference/new york city.shp")
#tmap_mode("plot")
#manh=nyc[nyc$COUNTYFP=="061",]
#tm_shape(manh) +
    #   tm_fill(col="AWATER") +
     #  tm_borders()

```

tmap


AWATER
0 to 500,000
500,000 to 1,0
1,000,000 to 1,5
1,500,000 to 2,0
2,000,000 to 2,5
2,500,000 to 3,0
3,000,000 to 3,5

map

tmap

- Interactive

```
# tmap_mode("view")
 #   tm_shape(manh) +
  #   tm_fill(col="AWATER") +
  # tm_borders()
```

tmap

map (file:///Users/ywu/Box/yi/2019%20R%20conference/Manh_inter_map.html)

tmap

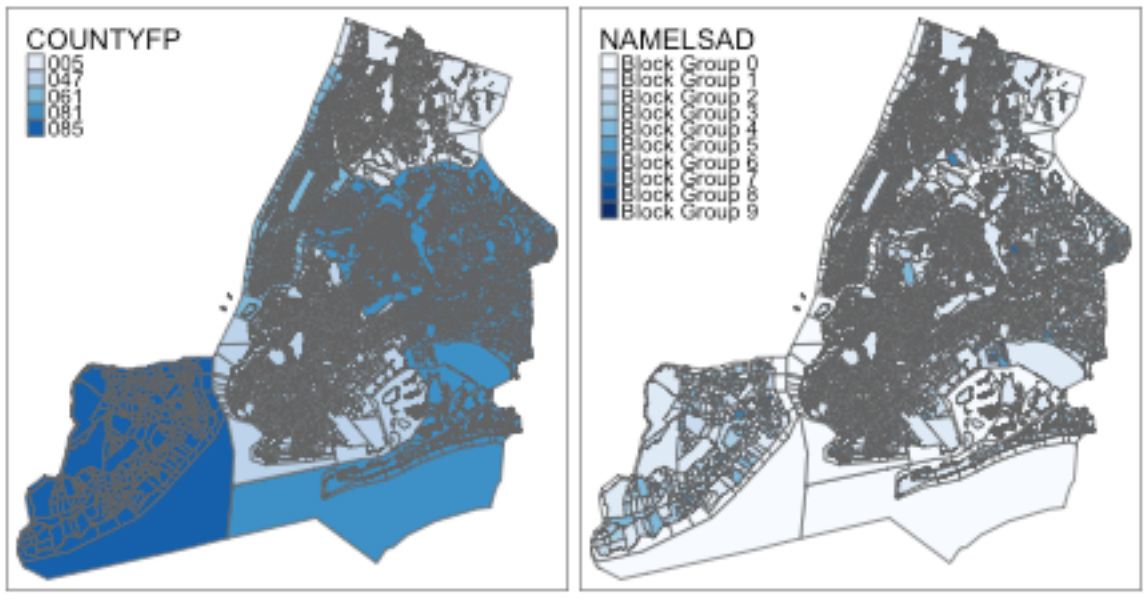
code

```r

      ## qtm(shp = nyc, fill = "NAMELSAD", fill.palette = "-Blues") # not shown
      ## qtm(shp = nyc,  fill=c("COUNTYFP", "NAMELSAD"), fill.palette = "Blues", ncol = 2) # n

```

tmap


COUNTYFP
005
047
061
081
085
NAMELSAD
Block Group 0
Block Group 1
Block Group 2
Block Group 3
Block Group 4
Block Group 5
Block Group 6
Block Group 7
Block Group 8
Block Group 9

result

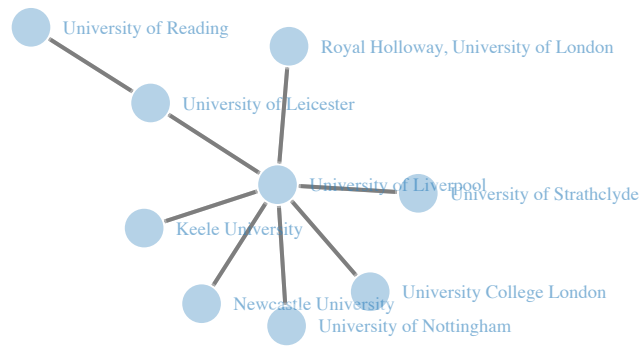
(https://rmarkdown.rstudio.com/flexdashboard/)

networkD3

networkD3

- dataset `SchoolsJournals` : edge list of REF(2014) journal submisisons for politics and interr relaions

```
data=networkD3::SchoolsJournals
## Convert to lis
# Use subset of data for more readable diagram
sub_data = data%>%filter(journal=="West European Politics")
sn=simpleNetwork(sub_data)
sn
```



networkD3

sankeyNetwork

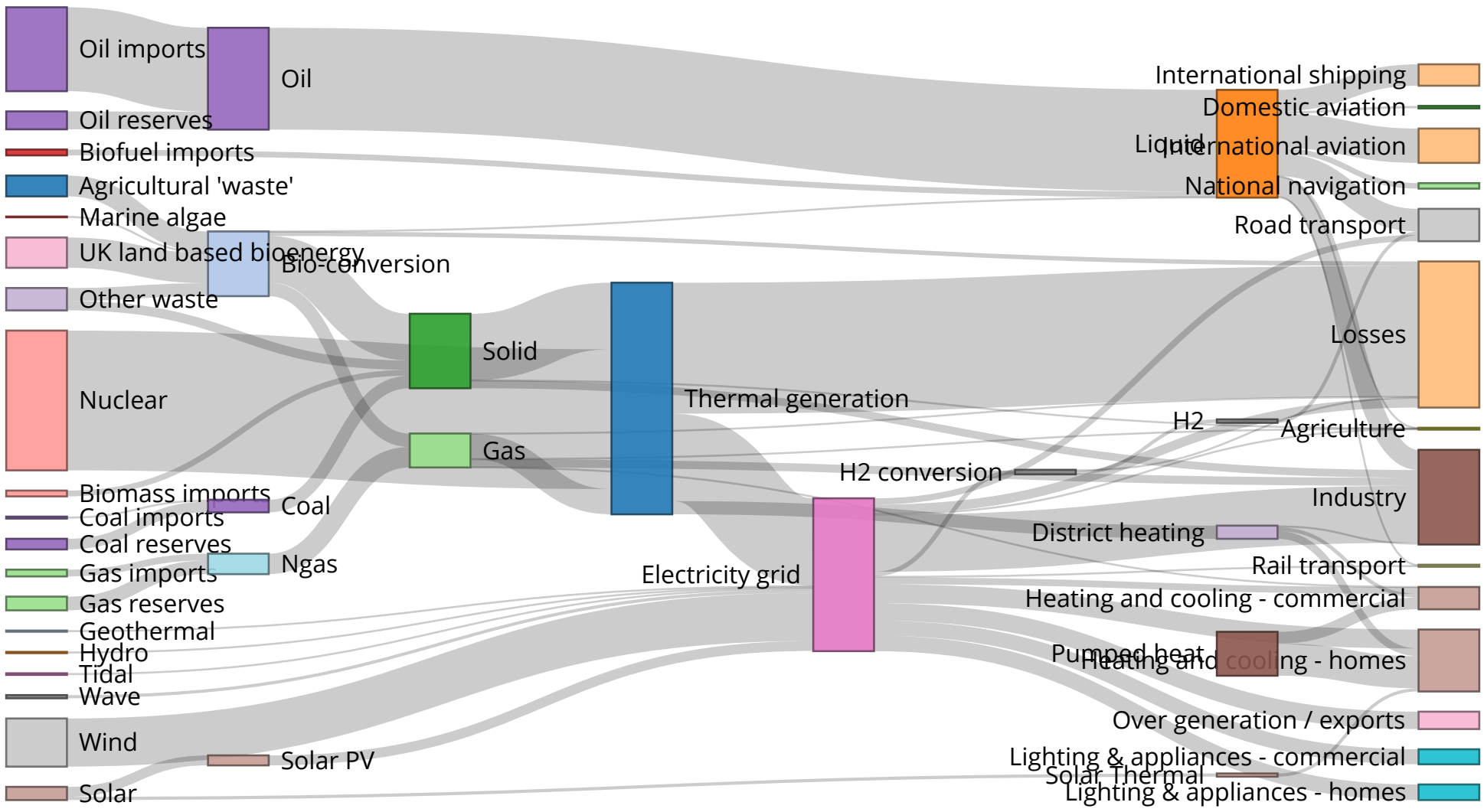
- the width of the arrows is proportional to the flow rate

```
library(rjson)
URL <- paste0(
        "https://cdn.rawgit.com/christophergandrud/networkD3/",
        "master/JSONdata/energy.json")
Energy <- jsonlite::fromJSON(URL)
Energy$links=Energy$links%>%arrange(desc(Energy$links$value))
head(Energy$nodes)
```

```
##                    name
## 1 Agricultural 'waste'
## 2       Bio-conversion
## 3               Liquid
## 4               Losses
## 5                Solid
## 6                  Gas
```

```
head(Energy$links)
##   source target   value
## 1     35     26 839.978
## 2     26      3 787.129
## 3     37      2 611.990
## 4     26     15 525.531
## 5     36     37 504.287
## 6      4     26 400.120
```

```
# Plot
sankeyNetwork(Links = Energy$links, Nodes = Energy$nodes, Source = "source",
              Target = "target", Value = "value", NodeID = "name",
              units = "TWh", fontSize = 12, nodeWidth = 30)
```



tidyr

separate()

```
separate(landed, timestamp, sep="-", into=c("year", "month", "day"))
## # A tibble: 9 x 4
##   user_id year  month day
##     <dbl> <chr> <chr> <chr>
## 1       1 2018  07    01
## 2       2 2018  07    01
## 3       3 2018  07    02
## 4       4 2018  07    01
## 5       4 2018  07    04
## 6       5 2018  07    10
## 7       5 2018  07    12
## 8       6 2018  07    07
## 9       6 2018  07    08
```

`crossing()`

- similar to `expand.grid()`
- example: *iris* dataset

```
library(dplyr)
head(iris)
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.7         0.4  setosa
```

example code

```
formulas <- list(
  formula1 = Sepal.Length ~ Sepal.Width,
  formula2 = Sepal.Length ~ Sepal.Width + Petal.Width,
  formula3 = Sepal.Length ~ Sepal.Width + Petal.Width + Petal.Length,
  formula4= Sepal.Length ~ Sepal.Width * Petal.Width + Petal.Length,
  formula5= Sepal.Length ~ Sepal.Width *Petal.Width * Petal.Length
)

data <- split(iris, iris$Species)
all=crossing(formula = formulas, data)
col=names(iris)
```

Links

Journal of Statistics Software Article about tmap (https://www.jstatsoft.org/article/view/v084i06

Rstats DC Twitter (https://twitter.com/rstatsdc)

Videos from NYR (https://dc.rstats.ai/2019/nyr/)

Youtube Videos Posted by Sponsor Lander Analytics (https://www.youtube.com/channel/UC2-hKemnrmVCH_29duyJ26A)


DCR
The 2018 Washington DC R Conference by Lander Analytics

Thank You!