

Rを使った 地理空間情報データ操作と可視化

FOSS4G北海道 ハンズオンデイ:
モダンな方法で学ぶ、Rによる地理空間情報データの処理

瓜生 真也 @u_ribo

2017年6月30日

概要

パッケージを利用したデータ操作と可視化の方法を学びます。はじめに、用意したデータで簡単な説明をします。その後、北海道のオープンデータを用いて、応用的な処理を実践します。

パッケージを呼び出そう

```
# library(tidyverse)
(dplyr) # データ操作一般
(sf) # 地理空間情報データ処理
(leaflet) # 地図描画
```

dplyrパッケージ

データ操作を行う上での重要な機能関数として提供

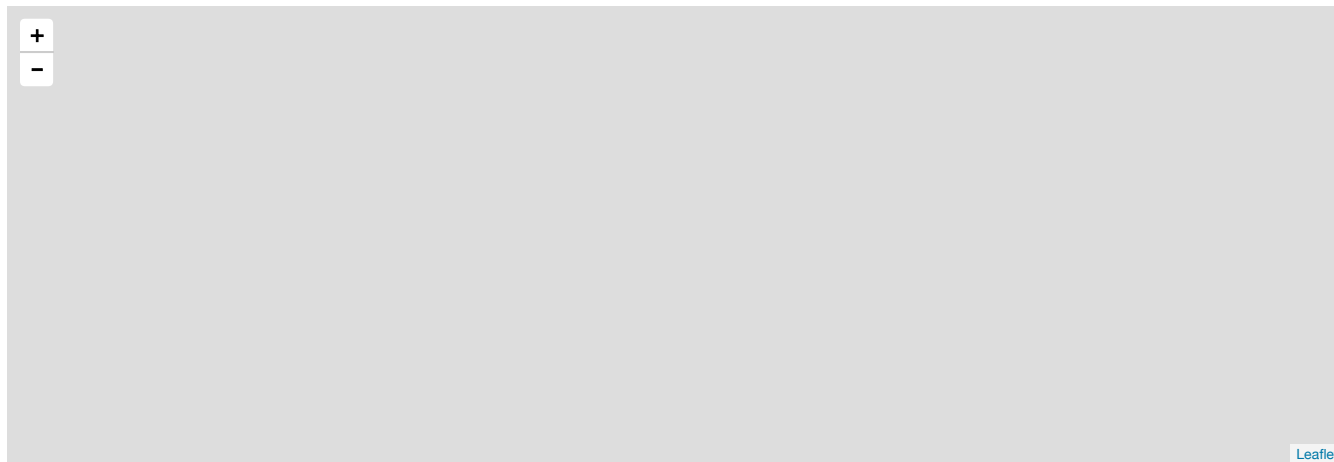
- データに対する主要な操作
 - 選択
 - 抽出
 - 加工
 - 集計
 - ...
- データベースからのデータ取得

sfパッケージ

- Simple Featuresをはじめとした多様な地理空間データソースを扱う
 - shapefile
 - kml, geojson, WKT/WKB
 - PostgreSQL
- 同名の関数による地理空間データの操作
- PostGISの関数に近い空間操作・解析が可能

leafletパッケージ

- JavaScriptで書かれたオープンソースのライブラリ
- インタラクティブな地図操作が可能



データの用意

- 平成27年国勢調査 人口等基本集計 総務省統計局 <http://www.stat.go.jp/data/kokusei/2015/>
 -

```
df.pops <- readr::read_rds("pref01_population2015.rds")
```

```
head(df.pops, 3)
```

```
# # A tibble: 3 x 3
#   city_code city_name  value
#   <chr>      <chr>    <dbl>
# 1    01000      北海道 5381733
# 2    01001 北海道市部 4395172
# 3    01002 北海道郡部  986561
```

データの用意

- 平成27年国勢調査 人口等基本集計 総務省統計局 <http://www.stat.go.jp/data/kokusei/2015/>
 -

```
df.pops <- readr::read_rds("pref01_population2015.rds")
```

```
head(df.pops, 3)
```

```
# # A tibble: 3 x 3
#   city_code city_name  value
#   <chr>      <chr>    <dbl>
# 1    01000      北海道 5381733
# 2    01001 北海道市部 4395172
# 3    01002 北海道郡部  986561
```

- 国土数値情報 行政区域データ (北海道) <http://nlftp.mlit.go.jp/ksj/gml/datalist/KsjTmplt-N03.html>

```
df.pref01 <- readr::read_rds("ksj_n0301.rds")
```


sfオブジェクト

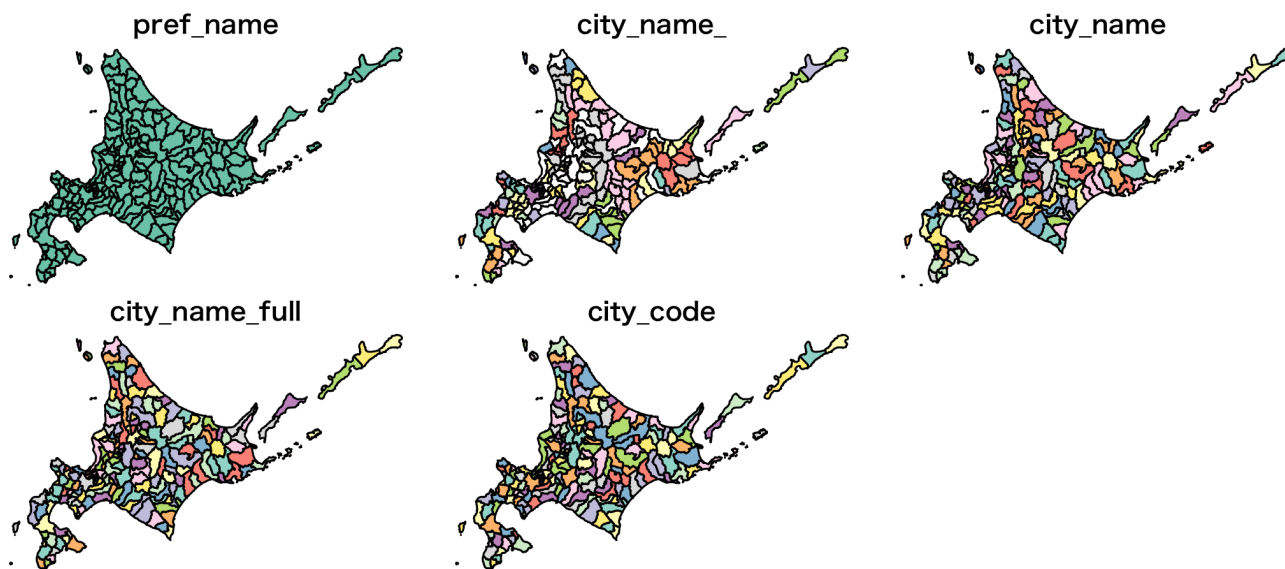
- Simple feature collection
 - 5変数 + geometry
- 地物タイプ: ポリゴン
- 次元: XY
- 範囲
- 空間参照システム(epsg)
- 投影法

```
head(df.pref01, 3)
```

```
# Simple feature collection with 3 features and 5 fie
# geometry type: POLYGON
# dimension: XY
# bbox: xmin: 141.202 ymin: 42.99747 xmax:
# epsg (SRID): 4326
# proj4string: +proj=longlat +datum=WGS84 +no_defs
# pref_name city_name_ city_name city_name_full cit
# 1 北海道 札幌市 中央区 札幌市 中央区 0110
# 2 北海道 札幌市 北区 札幌市 北区 0110
# 3 北海道 札幌市 東区 札幌市 東区 0110
#
# geometry
# 1 POLYGON((141.35520083 43.06...
# 2 POLYGON((141.438800272 43.1...
# 3 POLYGON((141.457257497 43.0...
```

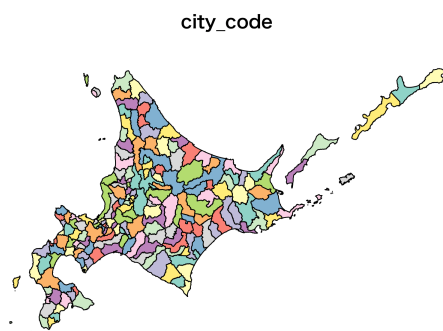
描画してみる

```
plot(df.pref01)
```



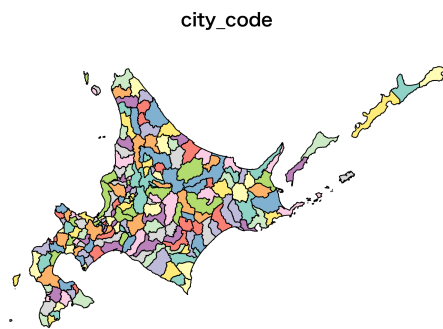
描画してみる

```
plot(df.pref01["city_code"])
```



描画してみる

```
plot(df.pref01["city_code"])
```



```
# 結果は同じ  
df.pref01 %>%  
  select(city_code) %>%  
  plot()
```

```
plot(st_geometry(df.pref01))
```

select()による変数の選択

```
# 変数名を引数で指定する  
df.mod <- df.pref01 %>% select(city_name_full  
names(df.mod))
```

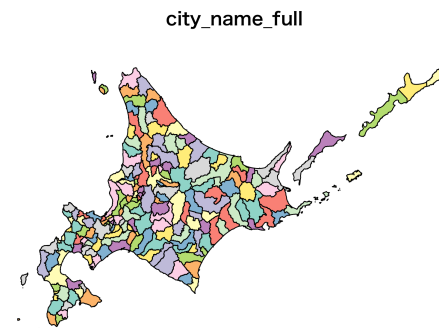
```
# [1] "city_name_full" "geometry"
```

select()による変数の選択

```
# 変数名を引数で指定する  
df.mod <- df.pref01 %>% select(city_name_full  
names(df.mod)
```

```
# [1] "city_name_full" "geometry"
```

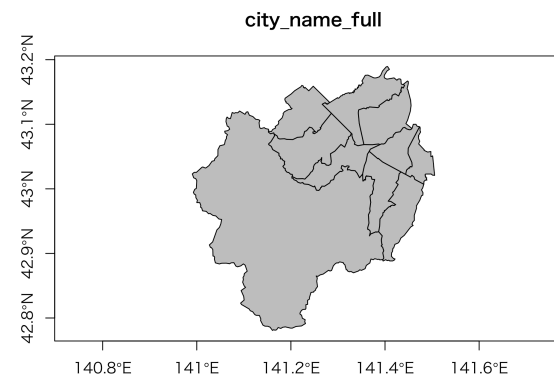
```
plot(df.mod)
```



filter()によるデータ抽出

```
# 条件に従うデータを抽出する
df.mod <- df.mod %>%
  # 「札幌市」を含んだ行を取り出す
  filter(grepl("札幌市", city_name_full))
```

```
plot(df.mod,
      col = "gray",
      axes = TRUE)
```



mutate()によるデータの加工

```
df.pref01$city_code[1] %>% class()
```

```
# [1] "factor"
```

```
# 文字列型に変換  
df.pref01$city_code[1] %>% as.character() %>%  
  class()
```

```
# [1] "character"
```

```
# データフレームの列に適用  
df.mod <- df.pref01 %>%  
  mutate(city_code = as.character(city_code))  
  
df.mod$city_code[1] %>% class()
```

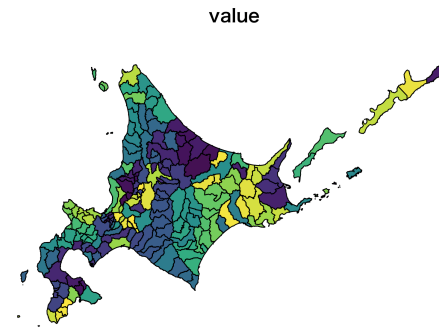
```
# [1] "character"
```


人口データと結合

2つのデータフレーム間で共通する変数を紐付ける

```
df.mod <- df.mod %>%  
  left_join(df.pops, by = "city_code")  
  
# 結合時の変数名とデータ型に注意  
# df %>%  
#   left_join(df.pops, by = "city_code")  
# # Warning message:  
# # Column `city_code` joining factor and cha
```

```
plot(df.mod["value"],  
     col = colormap::colormap("viridis"))
```



やってみよう

ジオメトリ操作

sfパッケージの関数を利用

- st_union: ジオメトリの結合
- st_buffer: 緩衝帯の付与
- st_centroid: 重心点を求める
- ...

```
st_union(df.pref01) %>%  
  plot()  
  
df.pref01["city_code"] %>%  
  st_buffer(dist = 0.05) %>%  
  plot()  
  
st_centroid(df.pref01["city_code"]) %>%  
  plot()
```

leaflet

```
base.map <- leaflet() %>%  
  addTiles()
```

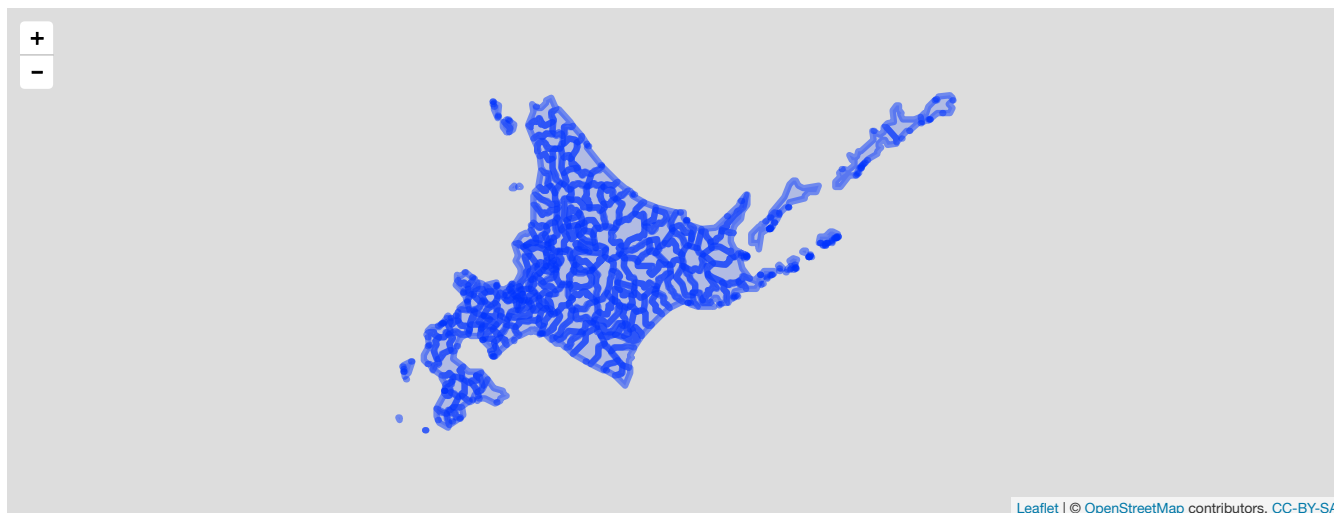
leaflet

```
base.map <- leaflet() %>%  
  addTiles()
```

- はじめに`leaflet()`を使って、必要な要素を足していく
- `addTiles()`はOpenStreetMapのタイルを呼び出す関数
 - `addProviderTiles()`の引数に`names(providers)`で表示されるサードパーティタイルを変更
 - `addTiles()`任意のタイルを利用。
 - 国土地理院タイルを利用する方法 <http://rpubs.com/yutannihilation/121912>

leafletにsfオブジェクトを描画させる

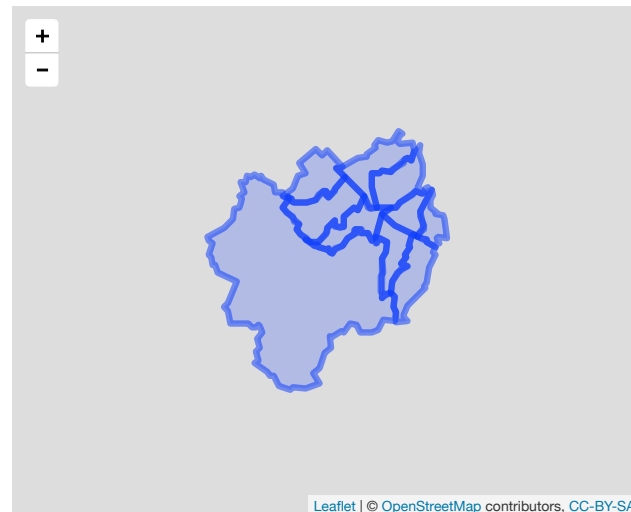
```
base.map %>%  
  addPolygons(data = df.pref01)
```



leafletにsfオブジェクトを描画させる

```
df.mod <- df.pref01 %>%  
  filter(city_name_ == "札幌市")
```

```
base.map %>%  
  addPolygons(data = df.mod) %>%  
  addMarkers(data = st_centroid(df.mod),  
            popup = ~city_name)
```



オープンデータを利用しよう

北海道オープンデータカタログ

<http://www.pref.hokkaido.lg.jp/ss/jsk/opendata/opendata.htm>

- 北海道はオープンデータが多い
- 北海道オープンデータカタログでは、基本的に
可能

に従うことで二次利用

どのデータを使う？

- ウェブスクレイピングにより一覧を確認
 - スクレイピング... ウェブページ上のテキストやデータを取得

```
(rvest)
```

```
x <- read_html("http://www.pref.hokkaido.lg.jp/ss/jsk/opendata/opendata.htm")
df.opd <- x %>% html_table(fill = TRUE) %>%
  .[[3]]
```

```
df.opd %>% view()
```

詳細な要素（リンク先のURL）を取得

```
x %>% html_nodes(css = '#rs_contents > p:nth-child(4) > span > strong')  
# {xml_nodeset (1)}  
# [1] <strong>北海道オープンデータカタログ</strong>
```

詳細な要素（リンク先のURL）を取得

```
x %>% html_nodes(css = '#rs_contents > p:nth-child(4) > span > strong')  
# {xml_nodeset (1)}  
# [1] <strong>北海道オープンデータカタログ</strong>
```

```
x %>% html_nodes(css = '#rs_contents > p:nth-child(4) > span > strong') %>%  
  html_text()  
# 北海道オープンデータカタログ
```

```
x %>% html_nodes(css = '#open_data > tbody > tr:nth-child(222) > td:nth-child(3) > a') %>%  
  html_text()  
# [1] "森林計画関係資料(GIS用データ)"  
(link.url <- x %>% html_nodes(css = '#open_data > tbody > tr:nth-child(222) > td:nth-child(3) >  
  html_attr(name = "href"))  
# [1] "http://www.pref.hokkaido.lg.jp/sr/srk/OPD.htm"
```

ブラウザの開発モードを使ってselector、XPathを取得すると楽

森林計画関係資料のダウンロード

```
x <- read_html(link.url)

x %>% html_nodes(css= '#rs_contents > div > table > tbody > tr > td:nth-child(2) > p:nth-child(1)' ) %>%
  html_text()
# [1] "留萌 " "留萌"

x %>% html_nodes(css= '#rs_contents > div > table > tbody > tr > td:nth-child(2) > p:nth-child(1)' ) %>%
  html_attr("href")
# [1] "https://www.fics.pref.hokkaido.lg.jp/FILE/2015/KMZ/09rumoi.zip"
# [2] "https://www.fics.pref.hokkaido.lg.jp/FILE/2015/GIS/09rumoi.zip"

download.file("https://www.fics.pref.hokkaido.lg.jp/FILE/2015/GIS/09rumoi.zip",
              destfile = "inst/09rumoi.zip")
unzip(
  'inst/rumoi.zip',
  exdir = "inst/"
)
```

やってみよう

