210905統計関連学会連合大会チュートリアル

Rによるデータ解析のためのデータ可視化

第3部 地図を描画する

国立環境研究所 生物多様性領域
瓜生真也
@u_ribo

目次

1 行政区域の地図

ggplot2での地図表現の基礎

2 地図を磨き上げる

地図表現を行う際の課題と向き合う

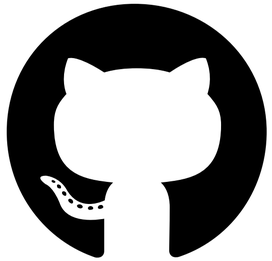
3 空間配置を考慮したグラフ

地図に限らない表現の区分

第7章 地図を描画する 247
7.1 アメリカ合衆国の州単位での地図 251
7.2 階級区分に頼らないアメリカ合衆国の地図 259
7.3 地理的な空間配置を考慮したグラフ 267
7.4 複数の地図を1枚の図にまとめる 270
7.5 そのデータは本当に空間情報を含みますか 274
7.6 次の一手 278

- 書籍ではアメリカ合衆国の事例、地図が主ですが、発表では日本を題材にします。
- 地図描画のための新たなデータ形式sfとggplot2パッケージでの描画関数geom_sf()を扱います。

資料



https://github.com/**uribo/jfssa2021_datavis**

- スライド(PDF)
- 予稿集のコード付き原稿(R Markdown)
- チュートリアルで利用するデータ

```
usethis::use_course("uribo/jfssa2021_datavis")
```

or

Gitクローンもしくは Zip ダウンロード

.Rproj ファイルを RStudio で開いてください。

チュートリアルで全般的に利用するパッケージ

```
library(dplyr) # データ操作を容易に行うパッケージ
library(ggplot2) # 可視化のためのパッケージ
library(sf) # 地理空間データを扱うパッケージ
library(rnaturalearth) # パブリックドメインで利用可能な行政地図データを提供するパッケージ
```

ggplot2テーマの設定

日本語フォント

好みのフォントを指定してOKです

```
theme_set(theme_bw(base_family = "IPAexGothic"))
```

日本語プロットの文字化けストレスを低減する - RStudio v1.4とraggパッケージを使う

https://uribo.hatenablog.com/entry/2021/03/29/202756

1. 行政区域の地図

ggplot2での地図表現の基礎

都道府県ポリゴンを用意

データ: Natural Earth

```
# Natural Earthから日本の都道府県ポリゴンを取得

ne_jpn <-
  ne_states(country = "Japan",
            returnclass = "sf") %>%
  # 使わない列を除外し、必要な列だけを選ぶようにします
  select(iso_3166_2, gn_name) %>%
  tibble::new_tibble(nrow = nrow(.), class = “sf")
class(ne_jpn)
#> [1] "sf"         "tbl_df"     "tbl"        "data.frame"

glimpse(ne_jpn)
#> Rows: 47
#> Columns: 3
#> $ iso_3166_2 <chr> "JP-46", "JP-44", "JP-40", "JP-41", "JP-42", "JP-43", "JP-4…
#> $ gn_name    <chr> "Kagoshima-ken", "Oita-ken", "Fukuoka-ken", "Saga-ken", "Na…
#> $ geometry   <MULTIPOLYGON [°]> MULTIPOLYGON (((129.7832 31..., MULTIPOLYGON (…
```

返り値のクラスを"sf"に指定

sfパッケージ

GIS規格のsimple features(sf)をRに実装

```
ne_jpn
#> Simple feature collection with 47 features and 2 fields
#> Geometry type: MULTIPOLYGON
#> Dimension:     XY
#> Bounding box:  xmin: 122.9382 ymin: 24.2121 xmax: 153.9856 ymax: 45.52041
#> CRS:           +proj=longlat +datum=WGS84 +no_defs +ellps=WGS84 +towgs84=0,0,0
#> # A tibble: 47 × 3
#>    iso_3166_2 gn_name                                                geometry
#>    <chr>      <chr>                                            <MULTIPOLYGON [°]>
#>  1 JP-46      Kagoshima-ken (((129.7832 31.79963, 129.7909 31.78441, 129.7986 3…
#>  2 JP-44      Oita-ken      (((131.2009 33.61271, 131.2199 33.60754, 131.2565 3…
#>  3 JP-40      Fukuoka-ken   (((130.0363 33.45759, 130.0402 33.46125, 130.0446 3…
#> (省略)
#> # … with 44 more rows
```

地物(feature)の情報を格納する3つのクラス（sfg,sfc,sf）

sfg (geometry) < sfc (column) < sf

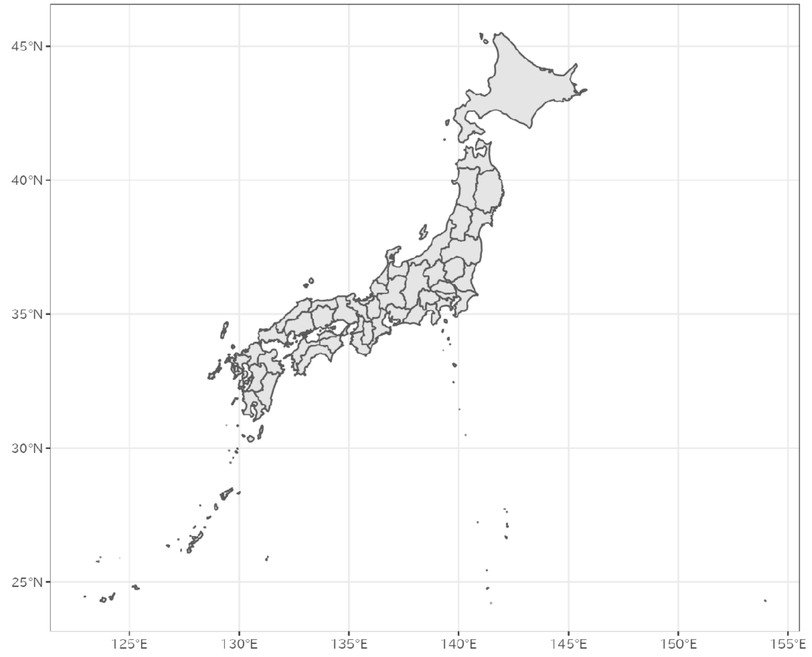
ggplot2::geom_sf()

sfオブジェクトをマッピングするgeom_*()

```
ggplot(ne_jpn) +
  geom_sf()
```

同じ出力結果

```
ggplot() +
  geom_sf(data = ne_jpn)
```



教科書では地図データの描画にgeom_polygon()を使いましたが、
現在はこちら（sf形式）を利用するのが標準的です。

ggplot2::geom_sf()

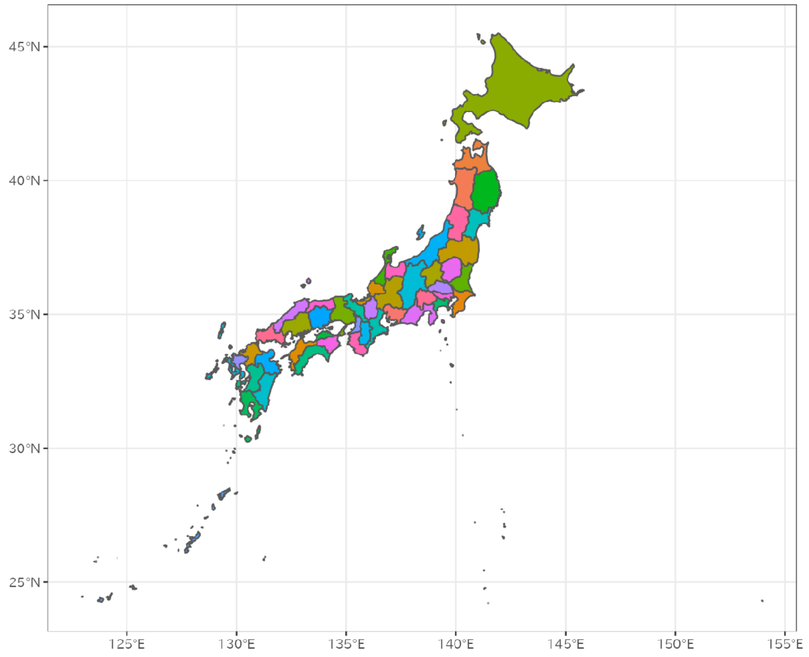
sfオブジェクトには他の変数が紐付けられるので…

都道府県ごとに塗りつぶし

```
ggplot(ne_jpn) +
  geom_sf(aes(fill = gn_name))
```

同じ出力結果

```
ggplot() +
  geom_sf(data = ne_jpn,
          aes(fill = gn_name))
```



ggplot2の審美的要素のマッピングが容易

行政区域の地図の例: 選挙結果のマッピング

データ： 第48回衆議院議員総選挙

都道府県別届出政党等別得票数（小選挙区）

https://www.soumu.go.jp/senkyo/senkyo_s/data/shugiin48/index.html

各都道府県でもっとも得票率 prop の高かった党派 party

```
glimpse(ne_jpn_shugiin48)
#> Rows: 47
#> Columns: 8
#> $ iso_3166_2 <chr> "JP-46", "JP-44", …
#> $ prefecture <chr> "鹿児島県", “大分県", …
#> $ gn_name    <chr> "Kagoshima-ken", “Oita-ken", …
#> $ party      <chr> "自由民主党", “自由民主党", …
#> $ votes      <dbl> 403187.0, 279778.0, …
#> $ is_ruling  <lgl> TRUE, TRUE, …
#> $ prop       <dbl> 53.15506, 51.12192,…
#> $ geometry   <MULTIPOLYGON [°]> MULTIPOLYGON (((129.7832 31 ...
```

行政区域の地図の例: 選挙結果のマッピング

Step1

党派を塗り分けるカラーコードを定義

```
party_colors ← c(`自由民主党` = "#41A12E",
                 `公明党` = "#F35A82",
                 `立憲民主党` = "#1B4787",
                 `希望の党` = "#136437",
                 `日本共産党` = "#D90A26",
                 `日本維新の会` = "#3EC021",
                 `社会民主党` = "#1CA9E9",
                 `諸派` = "#D3D3D3",
                 `無所属` = "#691D82")
```

Step2

1 データを定義し

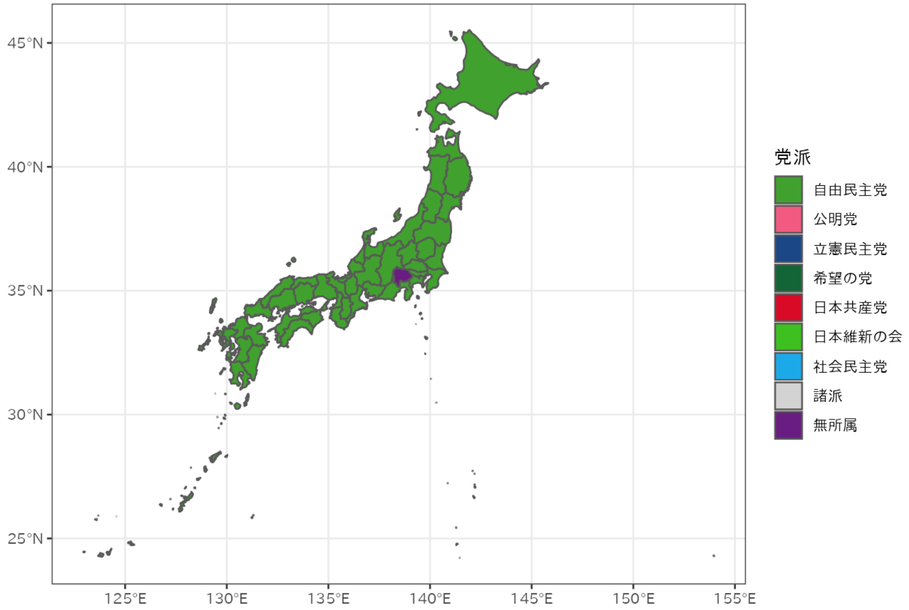
```
p ←
  ggplot(data = ne_jpn_shugiin48,
         aes(fill = party))
```

2 審美的要素をマッピング

（党派で塗り分ける）

行政区域の地図の例: 選挙結果のマッピング

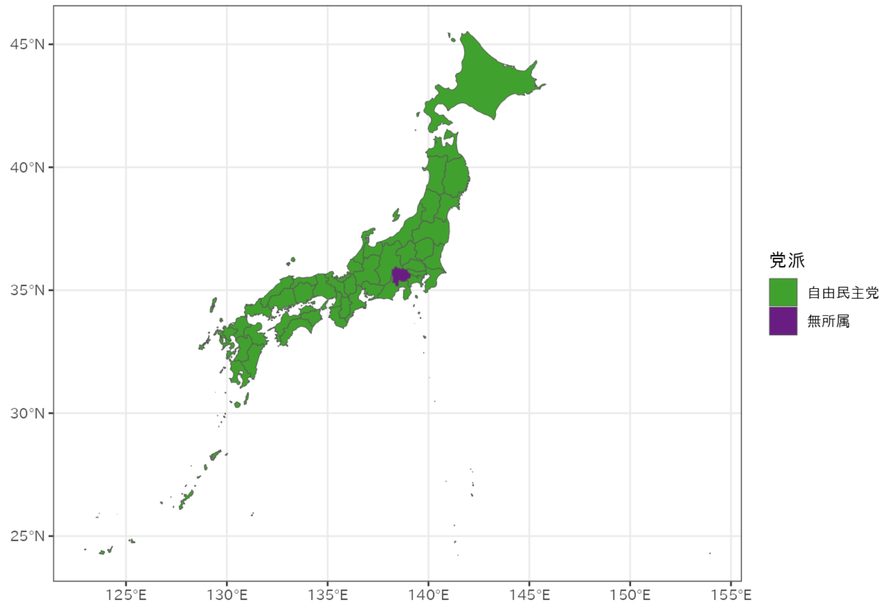
```
p1 ←
  p +
  geom_sf() +
  scale_fill_manual(
    values = party_colors,
    guide = guide_legend(title = "党派"))
p1
```



都道府県ごとの第一党

行政区域の地図の例: 選挙結果のマッピング

```
p2 ←
  p +
  # 県境を明確にするために太さを調節する
  geom_sf(size = 0.2) +
  # 使われない凡例を削除する
  scale_fill_manual(
    values = party_colors[c(1, 9)],
    guide = guide_legend(title = "党派"))
p2
```



2. 地図を磨き上げる

地図表現を行う際の課題と向き合う

地球(3次元)を平面(2次元)に表現するには?

2つの座標参照系（Coordinate Reference System: CRS）

空間データ座標が地球上のどの位置にあるか特定するための仕組み

地理座標系

緯線と経線で構成される。東西方向を経度、南北方向を緯度で表現する。

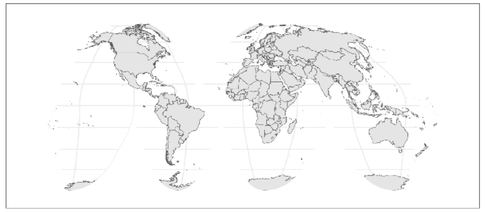
多くの場合、北半球は正の緯度を持ち、南半球は負の緯度となる。経度の原点として本初子午線が使用される。

投影座標系

2次元のXY座標で表現。面積・距離・角度のいずれかに対する歪みを補正できる。

モルワイデ図法

メルカトル図法



グード図法

日本周辺を表示する投影法

- メルカトル図法
- UTM座標系
- 平面直角座標系

地図投影法の変換

データ: Natural Earth

```
# Natural Earthから全球ポリゴンを取得
ne_world ←
  ne_countries(scale = 10,
               returnclass = "sf")
```

sfパッケージの関数st_crs()で座標参照系を確認

```
x ←
  st_crs(ne_world)
x$input
#> [1] "+proj=longlat +datum=WGS84 +no_defs +ellps=WGS84 +towgs84=0,0,0"
```

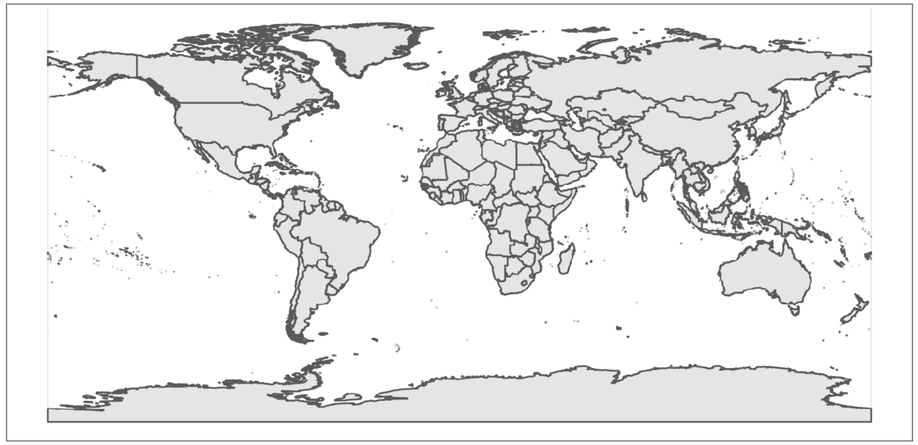
地理座標系を示す

地図投影法の変換

地理座標系のsfオブジェクトをマッピング

```
p ←
  ggplot(data = ne_world) +
  geom_sf()

p
```



1 coord_sf()でマッピング時に　2 対象オブジェクトのCRSを事前に

変換

地図投影法の変換

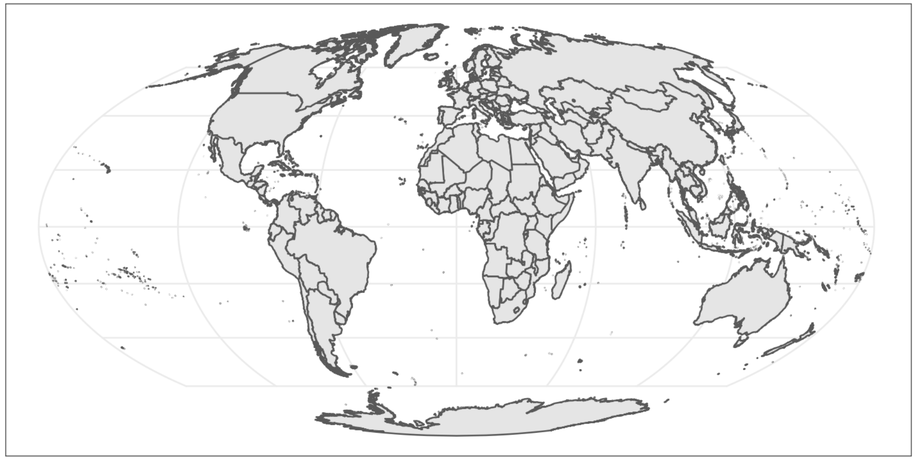
1 coord_sf()の適用

```
# モルワイデ図法による世界地図の描画
p +
  coord_sf(crs = "+proj=moll")
```

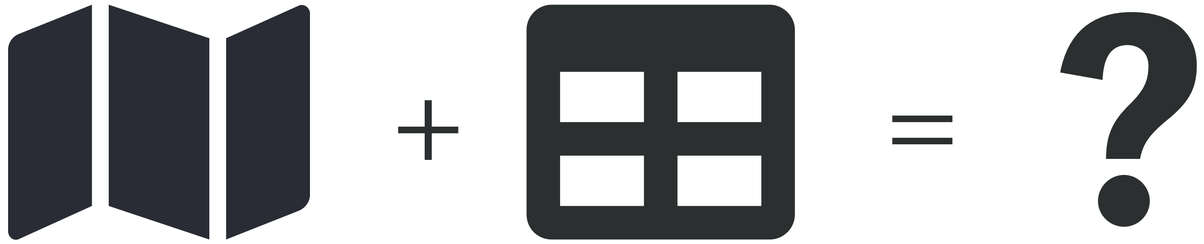
2 sfオブジェクトの座標参照系を変更

```
# st_transform()による座標参照系の変更
ne_world_moll ←
  st_transform(ne_world, crs = "+proj=moll")

ggplot(data = ne_world_moll) +
  geom_sf()
```



地図データと手持ちのデータを結合したいとき



地理空間データ（sf）

prefecture	geometry
北海道	POLYGON(…)
青森県	POLYGON(…)
…	…
沖縄県	POLYGON(…)

属性データ(data.frame)

prefecture	population
北海道	5.281
青森県	1.249
…	…
沖縄県	1.457

sfとデータ操作のためのdplyrパッケージを使って実現

地理空間データと属性データを紐づける

データ： 第48回衆議院議員総選挙

都道府県別届出政党等別得票数（小選挙区）

https://www.soumu.go.jp/senkyo/senkyo_s/data/shugiin48/index.html

```
df_shugiin48_party_votes ←
  readr::read_rds(here::here("data/shugiin48_prefecture_party_votes.rds"))

glimpse(df_shugiin48_party_votes)
#> Rows: 48
#> Columns: 31
#> $ 区分              <chr> "北海道", "青森県", "岩手県", "宮城県", "秋田県", "山形…
#> $ 自由民主党_男    <dbl> 1107667, 365462, 226455, 552240, 261709, 214176, 425155, 477…
#> $ 自由民主党_女    <dbl> 82096.0, NA, 57381.0, NA, NA, 103973.1, NA, 165437.0, NA, 20…
#> $ 自由民主党_計    <dbl> 1189763.0, 365462.0, 283836.0, 552240.0, 261709.0, 318149.1,…
#> (省略)
#> $ 合計_男           <dbl> 2250452.0, 577864.0, 543060.0, 813560.0, 508327.0, 490173…
#> $ 合計_女           <dbl> 442164.0, 19004.0, 78930.0, 188947.0, 13642.0, 103973.1, …
#> $ 合計_計           <dbl> 2692616, 596868, 621990, 1002507, 521969, 594147, 902732,…
```

地理空間データと属性データを紐づける

2つのデータを結合するための加工が必要

Step1

```
df_shugiin48_party_votes_mod ←
  df_shugiin48_party_votes %>%
  filter(区分 ≠ "計") %>%
  select(prefecture = 区分, ends_with("計")) %>%
  select(!starts_with("合計"))
```

1 都道府県の集計結果のみに

2 選択と同時に列名を変更
男女の合計値を選択

3 党派ごとの集計結果だけに

prefecture	自由民主党_計	立憲民主党_計	希望の党_計	公明党_計	日本共産党_計	日本維新の会_計	社会民主党_計	諸派_計	無所属_計
北海道	1189763.0	838165	253108.0	96795	159739.00	21643	NA	7632	125771
青森県	365462.0	NA	171280.0	NA	56011.00	NA	NA	4115	NA
岩手県	283836.0	NA	186376.0	NA	21549.00	NA	NA	NA	130229
宮城県	552240.0	78704	99141.0	NA	58795.00	10001	NA	2413	201213
秋田県	261709.0	NA	220759.6	NA	39500.39	NA	NA	NA	NA
山形県	318149.1	NA	236150.0	NA	37518.87	NA	NA	2329	NA

地理空間データと属性データを紐づける

Step2

```
df_shugiin48_party_votes_long ←
  df_shugiin48_party_votes_mod %>%
  tidyr::pivot_longer(cols = ends_with("計"),
                      names_to = "party",
                      values_to = "votes") %>%
  mutate(party = stringr::str_remove(party, "_計"),
         is_ruling = if_else(party %in% c("自由民主党", "公明党"),
                             TRUE,
                             FALSE))
```

列名と値の組み合わせを2つの列にする

prefecture	party	votes	is_ruling
北海道	自由民主党	1189763	TRUE
北海道	立憲民主党	838165	FALSE
北海道	希望の党	253108	FALSE
北海道	公明党	96795	TRUE
北海道	日本共産党	159739	FALSE
北海道	日本維新の会	21643	FALSE

地理空間データと属性データを紐づける

Step3

```
df_shugiin48_party_votes_tops ←
    df_shugiin48_party_votes_long %>%
    group_by(prefecture) %>%
    mutate(prop = votes / sum(votes, na.rm = TRUE) * 100) %>%
    top_n(n = 1, wt = prop) %>%
    ungroup()
```

グループ（都道府県）ごとに
処理をして新たな列を追加する

各都道府県でもっとも得票率が高い党派の得票数

prefecture	party	votes	is_ruling	prop
北海道	自由民主党	1189763	TRUE	44.2
青森県	自由民主党	365462	TRUE	61.2
岩手県	自由民主党	283836	TRUE	45.6
宮城県	自由民主党	552240	TRUE	55.1
…	…	…	…	…
沖縄県	自由民主党	267441	TRUE	42.0

ローマ字と漢字の結合は失敗する

データフレーム間の結合処理では、共通の値が記録された列が必要

```
ne_jpn$gn_name
#>  [1] "Kagoshima-ken" "Oita-ken" …

df_shugiin48_party_votes_tops$prefecture
#>  [1] "北海道" "青森県" …
```

prefecture	geometry
Hokkaido	POLYGON(…)
Aomori-ken	POLYGON(…)
…	…
Okinawa-ken	POLYGON(…)

prefecture	population
北海道	5.281
青森県	1.249
…	…
沖縄県	1.457

漢字と漢字の結合を行えるようにする

Step4

```
# zipanguパッケージからデータセットを利用する
jpnprefs ←
   zipangu::jpnprefs %>%
   select(prefecture_kanji, gn_name = prefecture)

ne_jpn_kanji ←
  ne_jpn %>%
  mutate(gn_name = recode(gn_name,
                          # ローマ字表記の規則を他県と合わせる
                          `Miyagi Ken` = "Miyagi-ken")) %>%
  inner_join(jpnprefs,
             by = "gn_name") %>%
  select(iso_3166_2, prefecture = prefecture_kanji, gn_name)
```

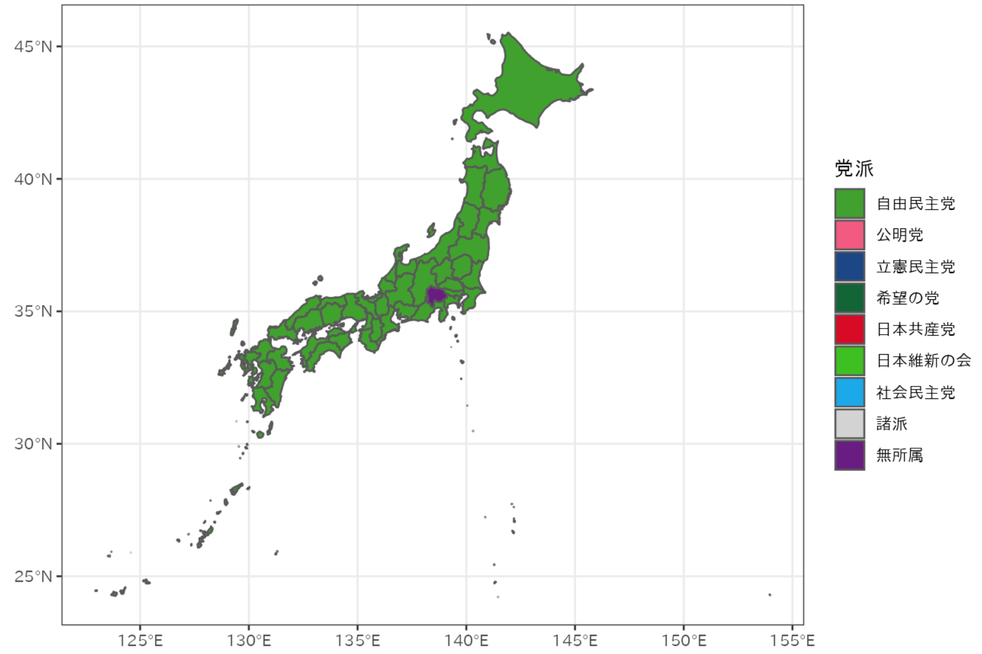
地理空間データと属性データを紐づける

sfオブジェクトに対してdata.frameの列を追加する

Step5

```
ne_jpn_shugiin48 ←
    ne_jpn_kanji %>%
    left_join(df_shugiin48_party_votes_tops, by = "prefecture") %>%
    relocate(geometry, .after = last_col())
```

選挙結果のマッピングで
利用したデータ



集計単位を変更したい

県 → 地方 = ？

県単位の集計

prefecture	population
北海道	5.281
青森県	1.249
…	…
沖縄県	1.457

地方単位での集計

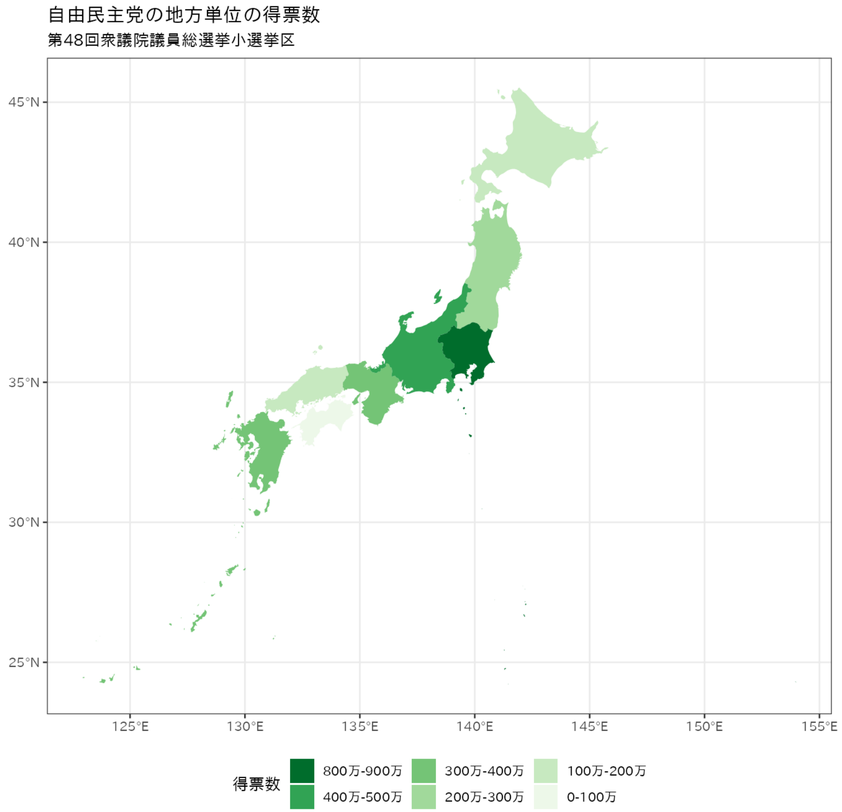
prefecture	population
北海道	5.281
東北	4.861
…	…
沖縄・九州	8.104

地図データの方も合わせて集約したい

sfのgeometryに対するdplyrの関数の適用

group_by()とsummarise()でポリゴンの集約が行われる

```
ne_jpn_region ←
  ne_jpn_kanji %>%
  left_join(zipangu::jpnprefs %>%
                  select(prefecture = prefecture_kanji,
                         region),
            by = "prefecture") %>%
  group_by(region) %>%
  summarise(.groups = "drop")
```



都道府県から地方への集約

（得票数とポリゴンデータ）

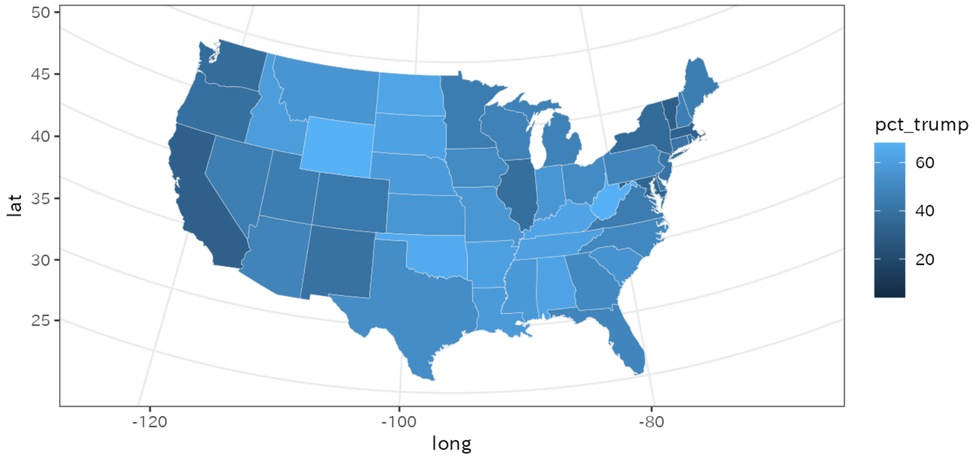
3. 空間配置を考慮したグラフ

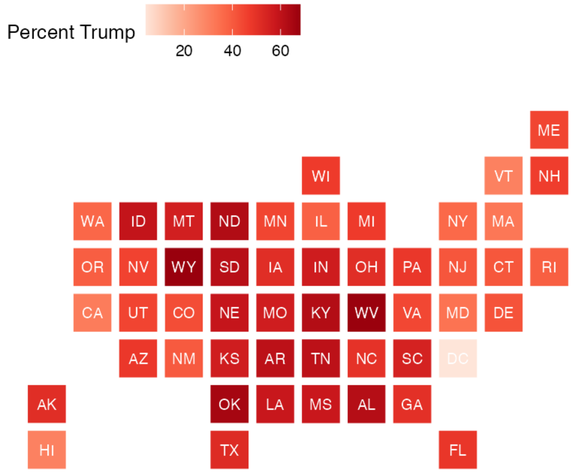
地図に限らない表現の区分

空間配置を考慮したグラフ

行政単位での地図…面積の違いによる差異が発生

面積を無視して、空間配置を簡略化





1 地理的な空間配置を
完全には再現しない

2 小さい州も大きい州も均等

3 大まかな位置関係でOK

教科書ではアメリカ合衆国を表現する"statebins"を紹介

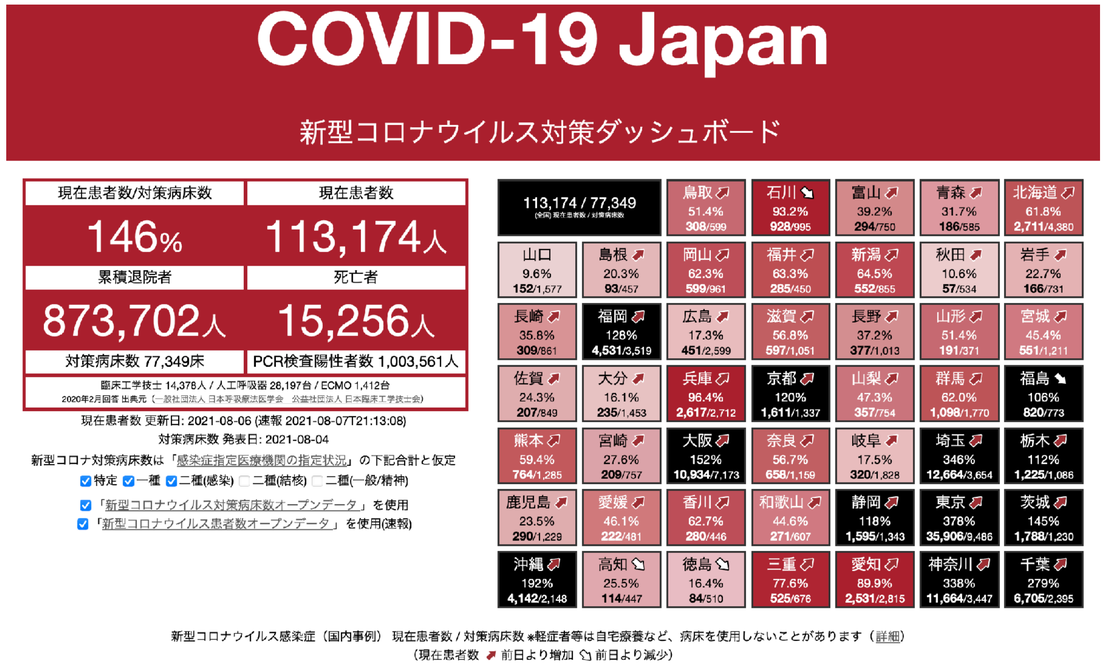
カラム地図

https://github.com/tabularmaps

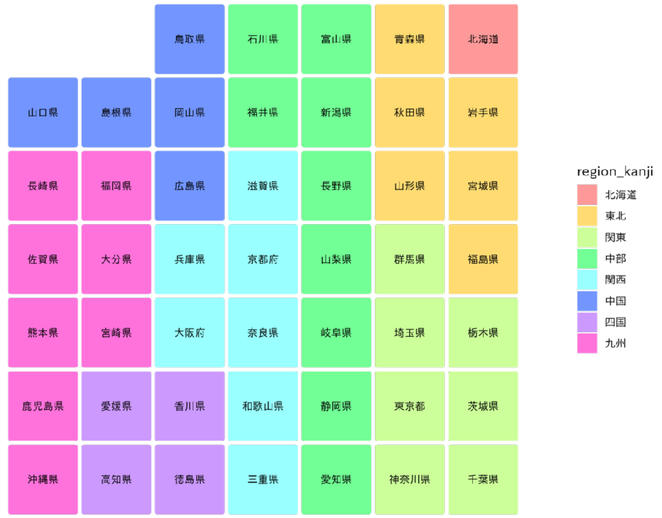
日本の都道府県、市町村を表形式に配置して表現

世界地図、県より小さな行政単位にも応用可能

https://www.stopcovid19.jp



{tabularmaps}パッケージ



タイル形式にデータを並べて表示するtabularmapsパッケージがCRANに登録されました

https://uribo.hatenablog.com/entry/2020/08/12/075316

tabularmapパッケージ

```
library(tabularmaps)

    jpn77 %>%
        select(jis_code,
               prefecture = prefecture_kanji,
               x,
               y) %>%
        left_join(df_shugiin48_party_votes_tops,
                  by = "prefecture") %>%
        tabularmap(x = x,
                   y = y,
                   group = jis_code,
                   label = prefecture,
                   fill = prop,
                   size = 2,
                   family = "IPAexGothic") +
        scale_fill_viridis_b() +
        theme_tabularmap(base_family = "IPAexGothic")
```