2018-07-15 Tokyo.R#71

A guide to spatial data analysis with the tidyverse

Shinya Uryu
@u_ribo
@uribo

A guide to spatial data analysis with the tidyverse

Overview

1. About the spatial data in R
2. r-spatial + tidyverse

Case Study: Flood and rain disaster in western Japan

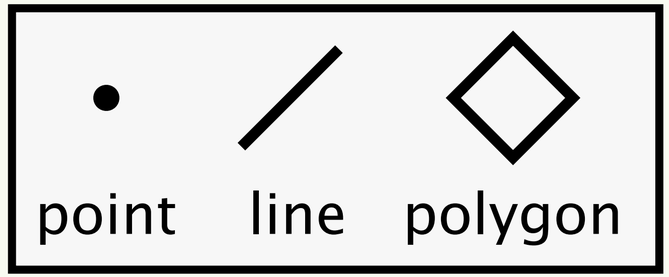
Brief history of spatial data in 

2005

rdgal

sp package

- Classes and Methods for Spatial Data
- S4 methods
- ***Spatial**** class (vector)
- lattice based graphics



2010

raster package

ggplot2::geom_raster(),
ggplot2::geom_map() etc.,

maptools

rasterVis package
leaflet package
tmap package

2016

sf package

BIG BANG

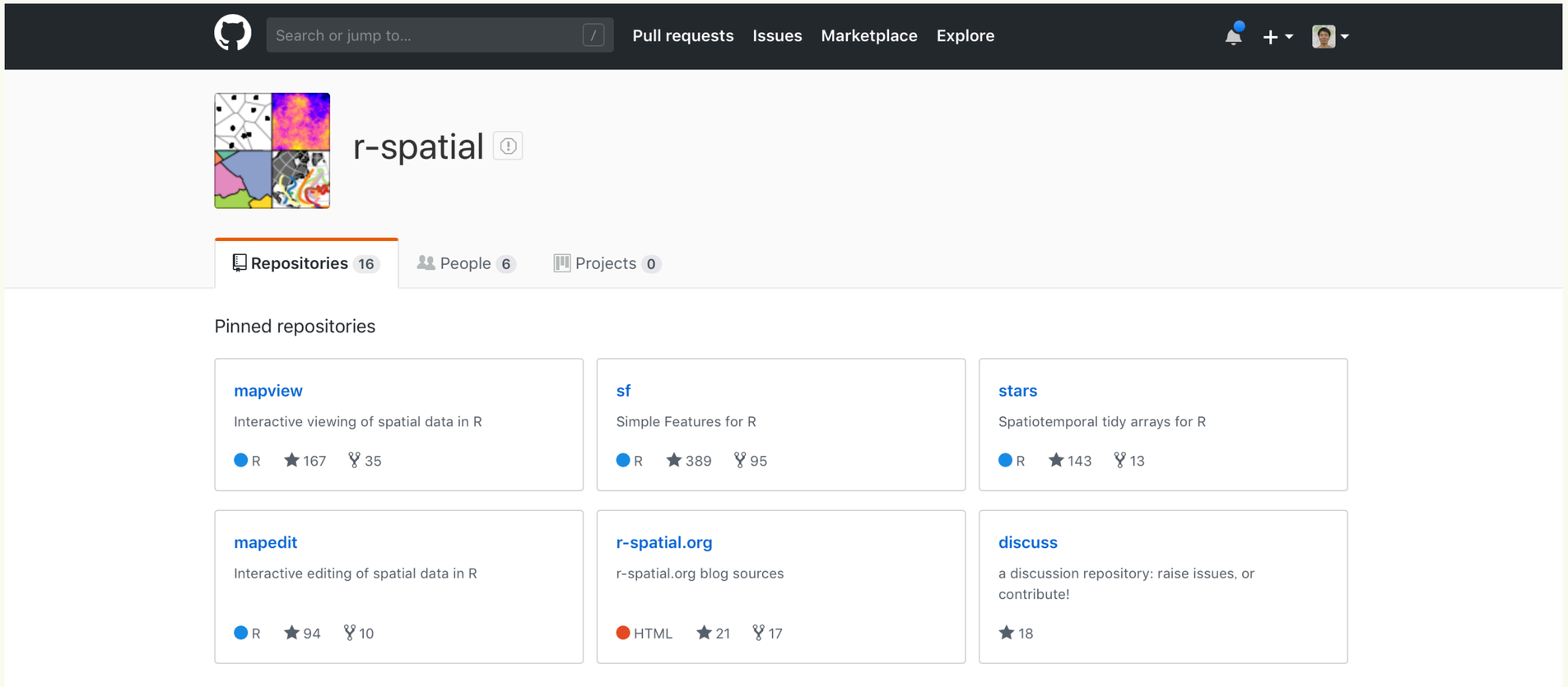
- tidy manifest
- list-column data frame
- pipe friendly APIs

mapview package
stars package
(…under development)

ggplot2::geom_sf

SEE. https://geocompr.robinlovelace.net/intro.html#the-history-of-r-spatial

r-spatial



https://github.com/r-spatial

sf

```
> library(sf) # version 0.6-3
> Linking to GEOS 3.6.1, GDAL 2.1.3,
proj.4 4.9.3
```

Simple feature

To storage and access
model of most geometry types.

all functions start with the prefix "**st_**" (spatial and temporal)

In sf, represents
simple features as R objects.

```
st_point(c(1, 2)) # X, Y Coordinates
#> POINT (1 2)
st_point(c(1, 2, 3)) # Added Z dimensions
#> POINT Z (1 2 3)


st_linestring(matrix(1:4, ncol = 2))
#> LINESTRING (1 3, 2 4)
```

```
st_polygon(list(
  rbind(
    st_point(c(1, 1)),
    st_point(c(2, 1)),
    st_point(c(2, 2)),
    st_point(c(1, 2)),
    st_point(c(1 ,1)))))
#> POLYGON ((1 1, 2 1, 2 2, 1 2, 1 1))
```

3 types of the sf object

```
library(jpndistrict)
#> Loading required package: jpmesh
#> This package provide map data is based on the Digital Map 25000
#> (Map Image) published by Geospatial Information Authority of Japan
#> (Approval No.603FY2017 information usage <http://www.gsi.go.jp>)
(sf_pref33 <- jpn_pref(33))
#> Simple feature collection with 30 features and 4 fields
#> geometry type:  GEOMETRY
#> dimens
#> bbox:
#> epsg (SRID):    4326
#> proj4st                                                    +no_defs
#> # A tib
#>    pref_code prefecture city_code city                          geometry
#>    <chr>     <chr>      <chr>     <chr>                     <GEOMETRY [°]>
#>  1 33        岡山県     33101     岡山市 北区  POLYGON ((133.9098 34.948, …¦
#>  2 33        岡山県     33102     岡山市 中区  MULTIPOLYGON (((133.9747, …¦
#>  3 33        岡山県     33103     岡山市 東区  MULTIPOLYGON (((133.9958, …¦
#>  4                                                (133.9109, …¦
#>  5 33        岡山県     33202     倉敷市       MULTIPOLYGON (((133.6414, …¦
```

sfc: Simple feature geometry **list-column**

sf: Simple feature

sfg: Simple feature **geometry**

3 types of the sf object

sf > sfc > sfg

```
class(sf_pref33)
#> [1] "sf"          "tbl_df"
    "tbl"         "data.frame"
class(sf_pref33$geometry)
#> [1] "sfc_GEOMETRY" "sfc"
class(sf_pref33$geometry[[1]])
#> [1] "XY"       "POLYGON" "sfg"
```

Seamlessly convert each sf objects.

```
(x <- st_point(c(1, 2)))
#> POINT (1 2)
x %>%
  st_sfc()
#> Geometry set for 1 feature
#> geometry type:  POINT
#> dimension:      XY
#> bbox:           xmin: 1 ymin: 2 xmax: 1 ymax: 2
#> epsg (SRID):    NA
#> proj4string:    NA
#> POINT (1 2)
x %>%
  st_sfc() %>%
  st_sf()
#> Simple feature collection with 1 feature and 0 fields
#> geometry type:  POINT
#> dimension:      XY
#> bbox:           xmin: 1 ymin: 2 xmax: 1 ymax: 2
#> epsg (SRID):    NA
#> proj4string:    NA
#>       geometry
#> 1 POINT (1 2)
```

I/O

Standard GIS file formats

- shapefile
- geojson (as strings)
- kml

```
st_read(system.file("shape/nc.shp",
        package = “sf"))

st_write(nc, "nc.shp")
```

Relational Database

- PostgreSQL

well known-text(WKT),
well known-binary (WKB) formats

Geo-processing functions

Provided like a **PostGIS** functions.

- *st_distance(), st_area(), st_make_gird()* etc.,

```
st_area(sf_pref33)
#> Units: m^2
#>  [1] 451746760  51433590 161417813 133400415 357919550 507973617 107320002
#>  [8] 136991748 245082627 212789693 548694329 794535530 260097992 127276525
#> [15] 210361376 830459441 430381734  67080736 144833382   7786478  12264588
#> [22]  91311276  67191543 420647698  54742098  70131416  58399305  79252153
#> [29] 232806322 269530873
st_bbox(sf_pref33)
#>      xmin      ymin      xmax      ymax
#> 133.26669  34.29839 134.41322  35.35286                # My Hometown (Tsukuba)
st_distance(st_sfc(st_point(c(140.10, 36.08)), crs = 4326),
            st_sfc(st_point(c(139.74, 35.68)), crs = 4326))# Here
#> Units: m
#>          [,1]
#> [1,] 55014.41
```

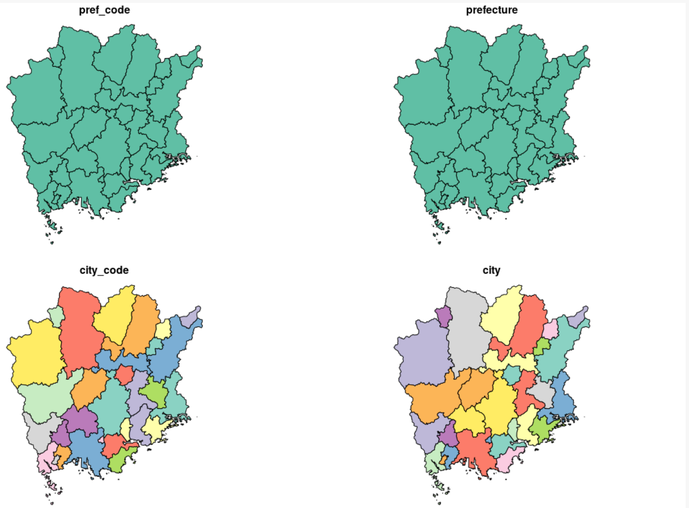
Another support functions found on lwgeom pkgs.

mapping & visualization

plot.sf*(): S3 method for sf

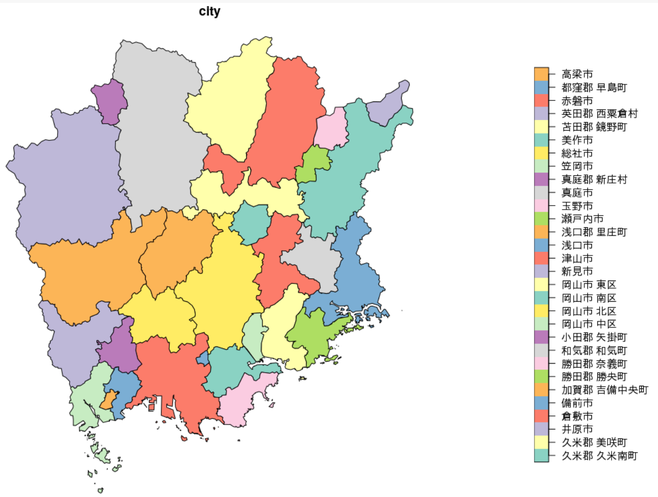
ALL

```
plot(sf_pref33)
# Drawing the first 10 out of all attributes
# Set by max.plot
```



SELECT

```
plot(sf_pref33["city"],
     key.pos = 4,
     key.width = lcm(5),
     key.length = 0.8)
```



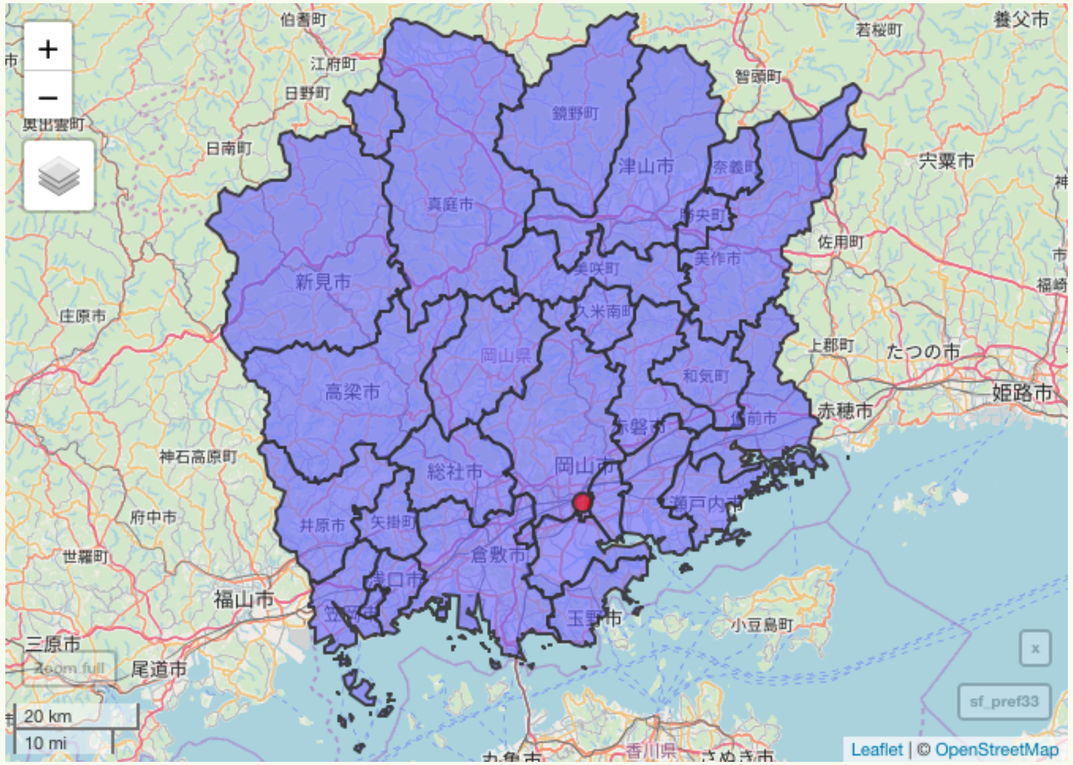
GEOMETRY

```
plot(st_geometry(sf_pref33),
     col = "white")
```

Interactive mapping with mapview

*Raster** (raster), *Spatial** (sp) and sf

```
library(mapview)
#> Okayama City Administration
  st_sfc(
    st_point(c(133.9196,
               34.65511)),
    crs = 4326)

mapview(sf_pref33) +
  mapview(x,
          col.regions = "red")
```



DEMO

r-spatial meets tidyverse

```
> library(tidyverse) # version 1.2.1
── Attaching packages ──────────────────────────────────── tidyverse 1.2.1 ──
✔ ggplot2 3.0.0     ✔ purrr   0.2.5
✔ tibble  1.4.2     ✔ dplyr   0.7.6
✔ tidyr   0.8.1     ✔ stringr 1.3.1
✔ readr   1.1.1     ✔ forcats 0.3.0
── Conflicts ─────────────────────────────────── tidyverse_conflicts() ──
✖ dplyr::filter() masks stats::filter()
✖ dplyr::lag()    masks stats::lag()
```

Case Study: Flood and rain disaster in western Japan

The Heavy Rain Event on July 2018 (平成30年7月豪雨).

A total of 11 prefectures that emergency heavy rain warnings areas.

As of 14 July, 197 were dead in various rain-related incidents.

According to the Fire and Disaster Management Agency

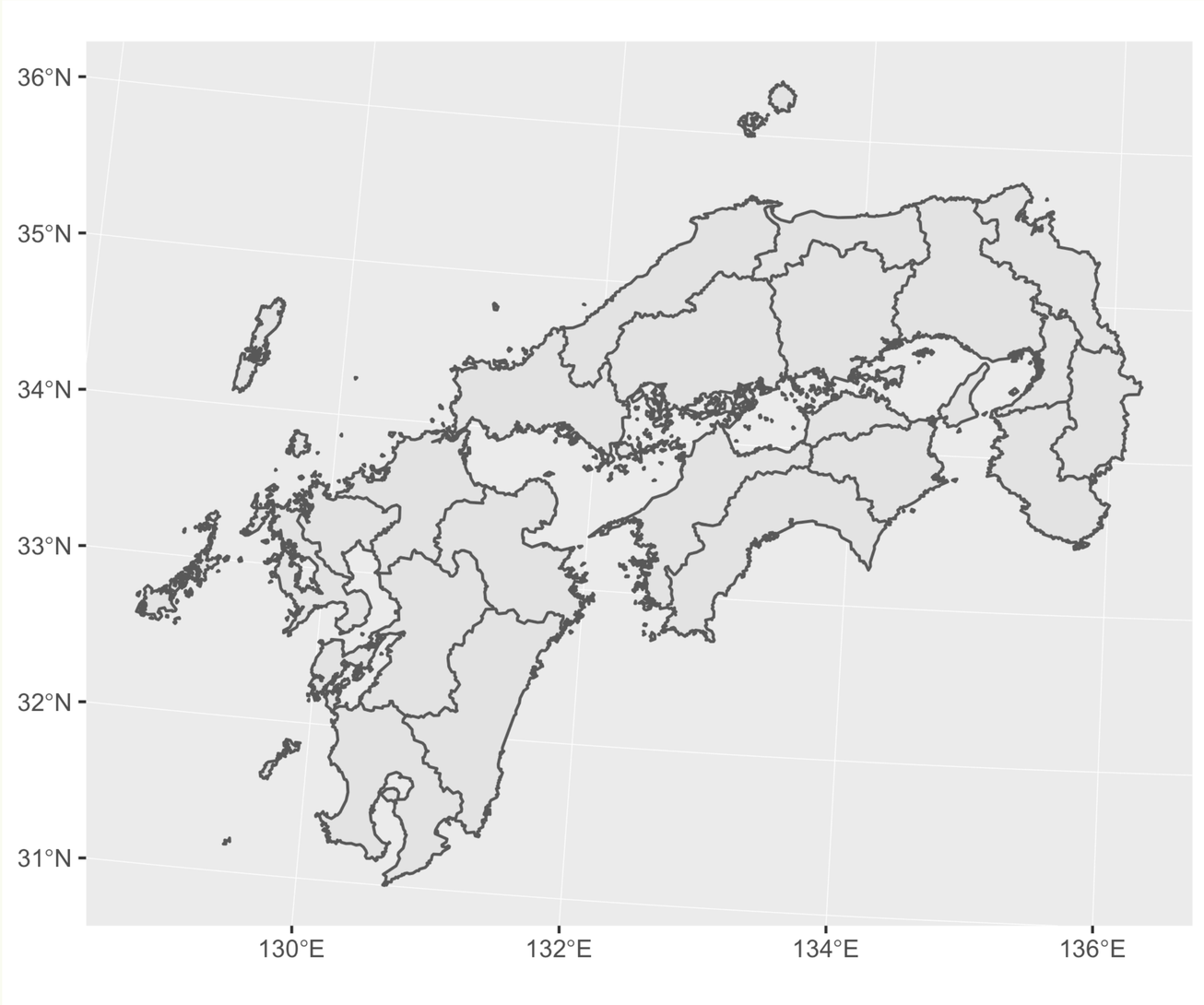
Create base map with purrr

```
# Provided by National Spatial Planning and Regional
# Policy Bureau, MILT of Japan
sf_west_japan <-
  26:46 %>%
  map(jpn_pref, district = FALSE) %>%
  # Unused map_dfr() ... bind_rows() not support sf class
  reduce(rbind)

format(object.size(sf_west_japan), units = “MB")
# [1] "18.1 Mb”

# Area cropped and simplified polygon
sf_west_japan <-
  st_crop(
    sf_west_japan,
    st_bbox(c(xmin = 128.5, ymin = 31.0, xmax = 136.5, ymax = 36.5))
  ) %>%
  # Reduce object size
  st_simplify(preserveTopology = TRUE, dTolerance = 0.005)

format(object.size(sf_west_japan), units = “MB”)
# [1] "1 Mb”
```

Mapping by ggplot2::geom_sf()

```
sf_west_japan <-
  sf_west_japan %>%
  st_transform(
    crs = “+proj=laea
           +lat_0=30
           +lon_0=140")
```

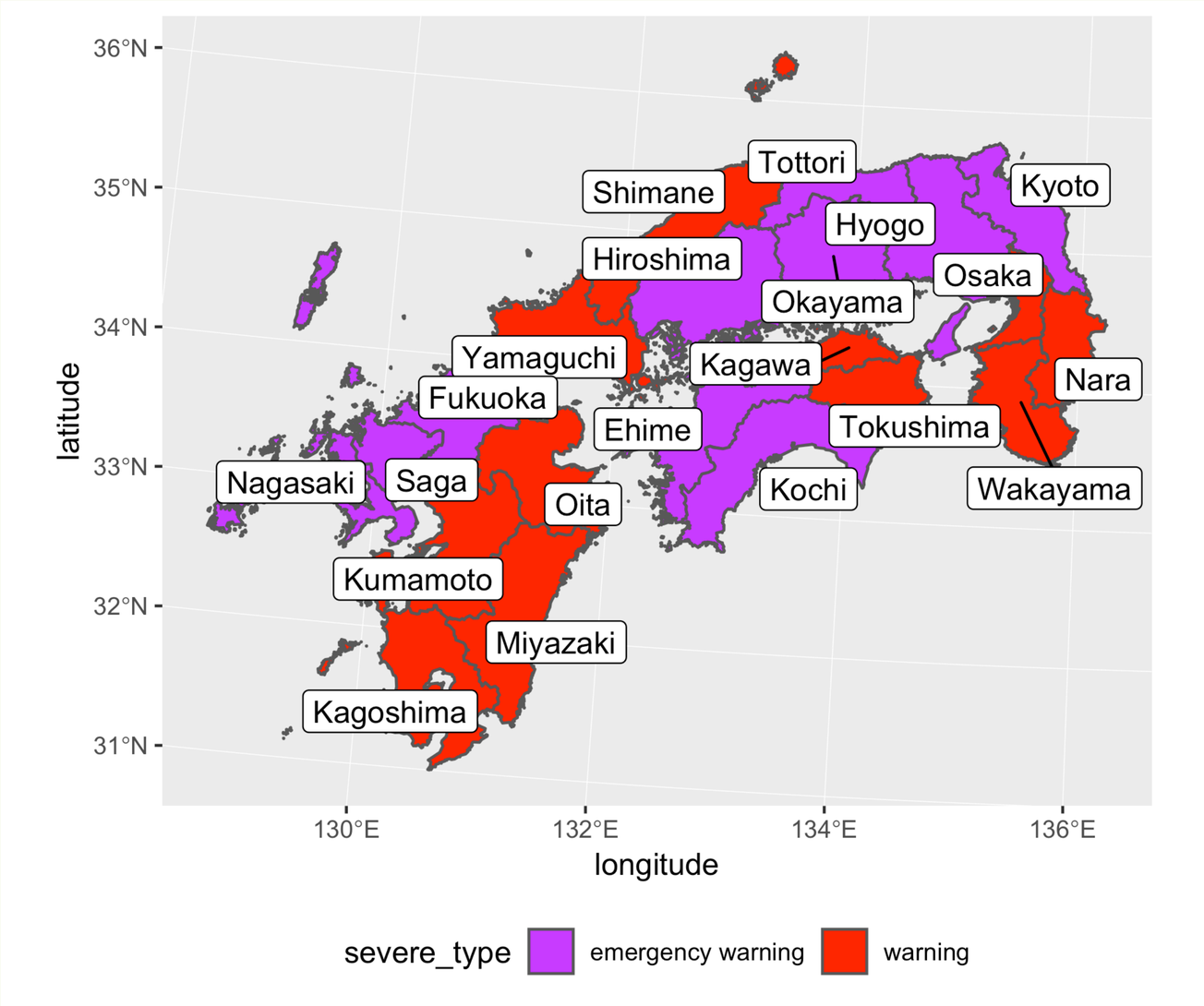
```
base_map <-
ggplot(sf_west_japan) +
  geom_sf()
```



Mapping by ggplot2::geom_sf()

Brush-up

```
df_west_japan <-
  sf_west_japan %>%
  mutate(
   longitude =
st_coordinates(st_centroid(geome
try))[, 1],
   latitude =
st_coordinates(st_centroid(geome
try))[, 2]) %>%
  st_set_geometry(NULL)

base_map +
  ggrepel::geom_label_repel(
   data = df_west_japan,
   aes(x = longitude,
       y = latitude,
   label = prefecture)) +
  coord_sf()
```



Downloading weather data from *JMA*

Japan Meteorological Agency

```
> remotes::install_git("https://gitlab.com/uribo/jmastats")
> library(jmastats) # ⚠️ Under Development 🚧
```

Access to Japan Meteorological Agency Data

Not provided API 😓

We can extract contents with rvest package.

http://www.data.jma.go.jp/obd/stats/etrn/index.php

Collect weather data

```
jma_collect("hourly",
            block_no = tgt_stations$block_no[6], # 舞鶴 (Maiduru, Kyoto)
            year = 2018,
            month = 7,
            day = 5) %>%
  parse_unit()
# A tibble: 24 x 17
    time atmosphere_land_… atmosphere_surf… precipitation_mm temperature dew_point
   <int> <S3: units>       <S3: units>      <S3: units>      <S3: units> <S3: uni>
 1     1 1003              1003.5           0                25.7        23.2
 2     2 1003.1            1003.6           0                25.4        23.5
 3     3 1003.1            1003.6           0                25.3        23.4
 4     4 1003.4            1003.9           0                25.1        23.5
 5     5 1003.9            1004.4           0                25          23.4
# ... with 19 more rows, and 11 more variables: vapor_h_pa <S3: units>,
#   humidity_percent <S3: units>, wind_speed_m_s <S3: units>,
#   wind_direction_m_s <chr>, dailight_h <S3: units>,
#   solar_irradiance_mj_m_2 <chr>, snow_fall_moment_cm <chr>,
#   snow_fall_period_cm <chr>, weather <chr>, cloud_covering <chr>,
#   visibility_km <S3: units>
```

Collect weather data

1. CRS transformed (To match polygons)
2. Convert MULTIPOINT to (single) POINT
3. Applied dplyr's verbs
    1. Check: Whether points are included in the area?

```
tgt_stations <-
  stations %>%
  st_transform(crs = "+proj=laea +lat_0=30 +lon_0=140") %>%
  st_cast("POINT") %>%
  select(area, station_type, station_name, elevation, block_no, pref_code) %>%
  distinct(block_no, .keep_all = TRUE) %>%
  filter(pref_code %in% 26:46) %>%
  mutate(in_area = purrr::pmap_lgl(., ~
                                     as.logical(sum(st_within(
                                                              ..7,
                                                              sf_west_japan,
                                                              sparse = FALSE))))) %>%
  filter(in_area == TRUE)
```

Collect weather data

```
df_weather <-
  list(block_no = rep(tgt_stations$block_no, each = 3),
       year = 2018,
       month = 7,
       day = rep(5:7, times = nrow(tgt_stations))) %>%
  pmap(~ jma_collect(item = "hourly",
                     block_no = ..1,
                     year = ..2,
                     month = ..3,
                     day = ..4) %>%
    select(time, `precipitation_(mm)`) %>%
    parse_unit() %>%
    transmute(block_no = c(..1),
              date = lubridate::make_datetime(..2, ..3, ..4, hour = time),
              precipitation_mm)) %>%
  reduce(rbind) # keep units attributes
```

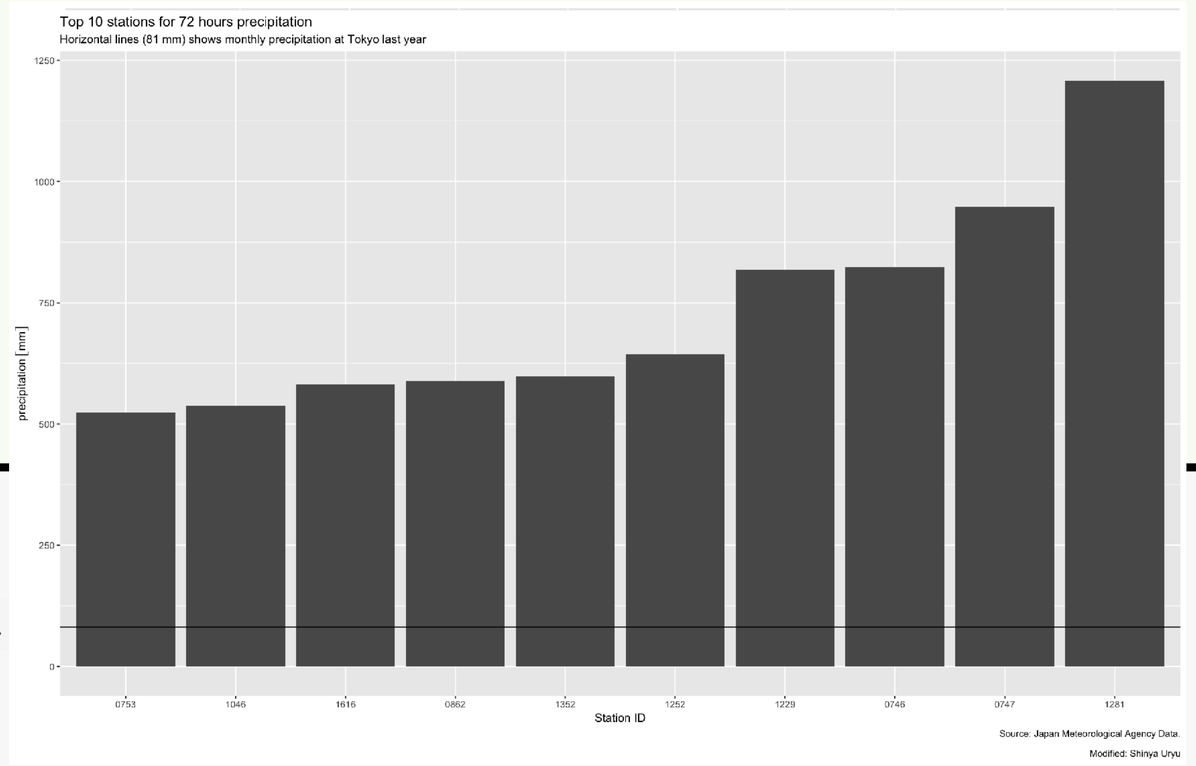
Collect weather data

```
df_weather
# A tibble: 26,136 x 3
   block_no date                precipitation_mm
   <chr>    <dttm>              <S3: units>
 1 1578     2018-07-05 01:00:00 0
 2 1578     2018-07-05 02:00:00 1.5
 3 1578     2018-07-05 03:00:00 1
# ... with 26,133 more rows
```

```
df_precipitation <-
  df_weather %>%
  group_by(block_no) %>%
  summarise(precipitation = sum(precipitation_mm))
# A tibble: 363 x 2
   block_no precipitation
   <chr>    <S3: units>
 1 0588     355
 2 0589     466
 3 0593     439.5
# ... with 360 more rows
```

Top 10 stations for 72 hours precipitation

```
df_precipitation_top10 <-
  df_precipitation %>%
  top_n(10, wt = precipitation)
```



Brush-up

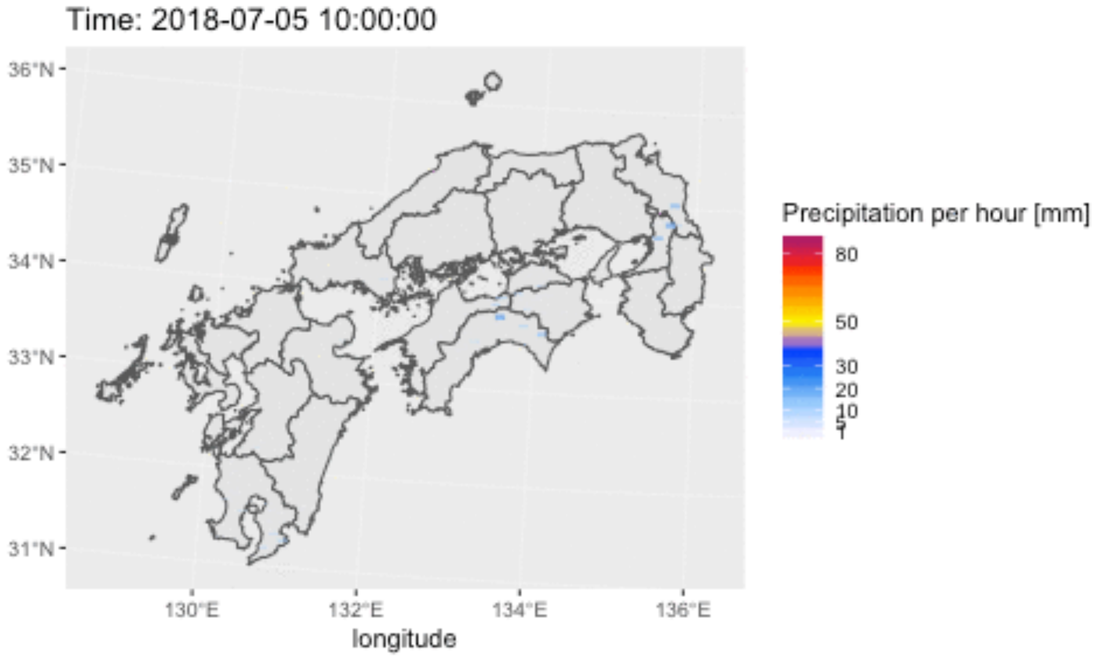
```
library(ggforce) # For display “units”
```

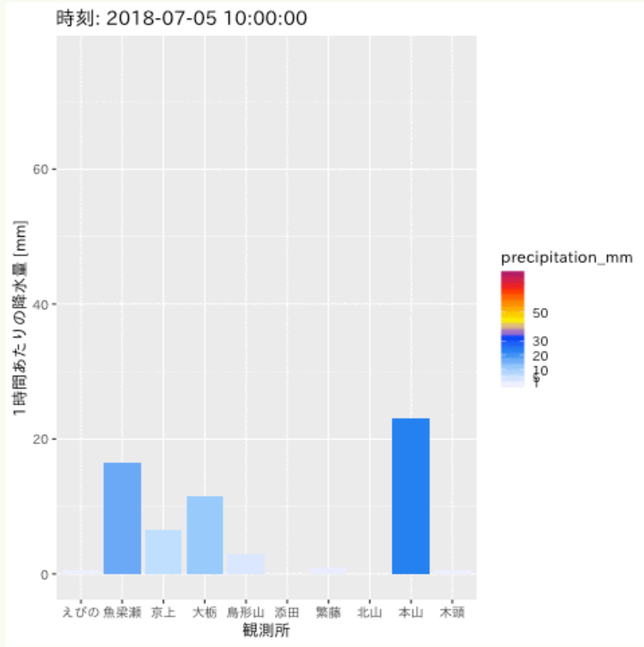
```
ggplot(df_precipitation_top10,
       aes(forcats::fct_reorder(block_no,
           units::drop_units(precipitation)),
           precipitation)) +
  geom_bar(stat = "identity") +
  geom_hline(yintercept = 81) +
  labs(x = "Station ID",
       title = "Top 10 stations for 72 hours precipitation",
       subtitle = "Horizontal lines (81 mm) shows monthly precipitation at Tokyo last year",
       caption = "Source: Japan Meteorological Agency Data.\n
       Modified: Shinya Uryu")
```

mapping plus animation

```
> remotes::install_github("thomasp85/gganimate")
> library(gganimate) # version 0.9.9.9999
```

Mapping on precipitation values

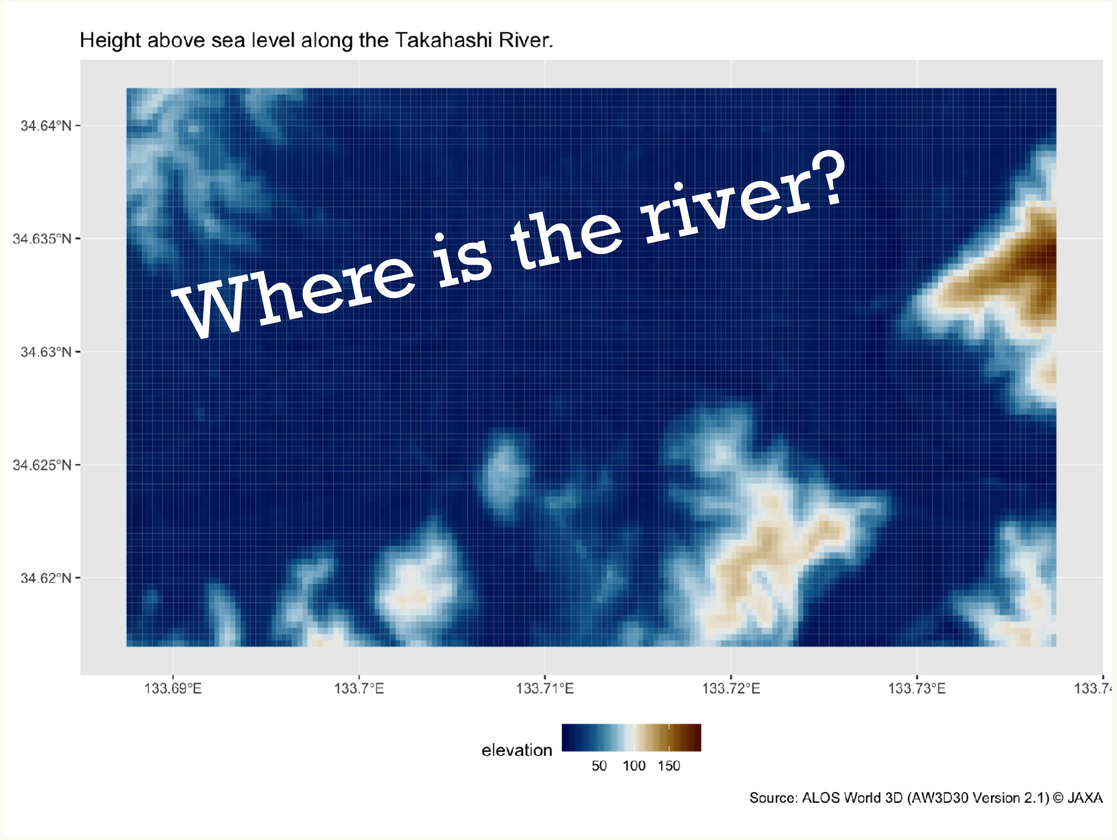




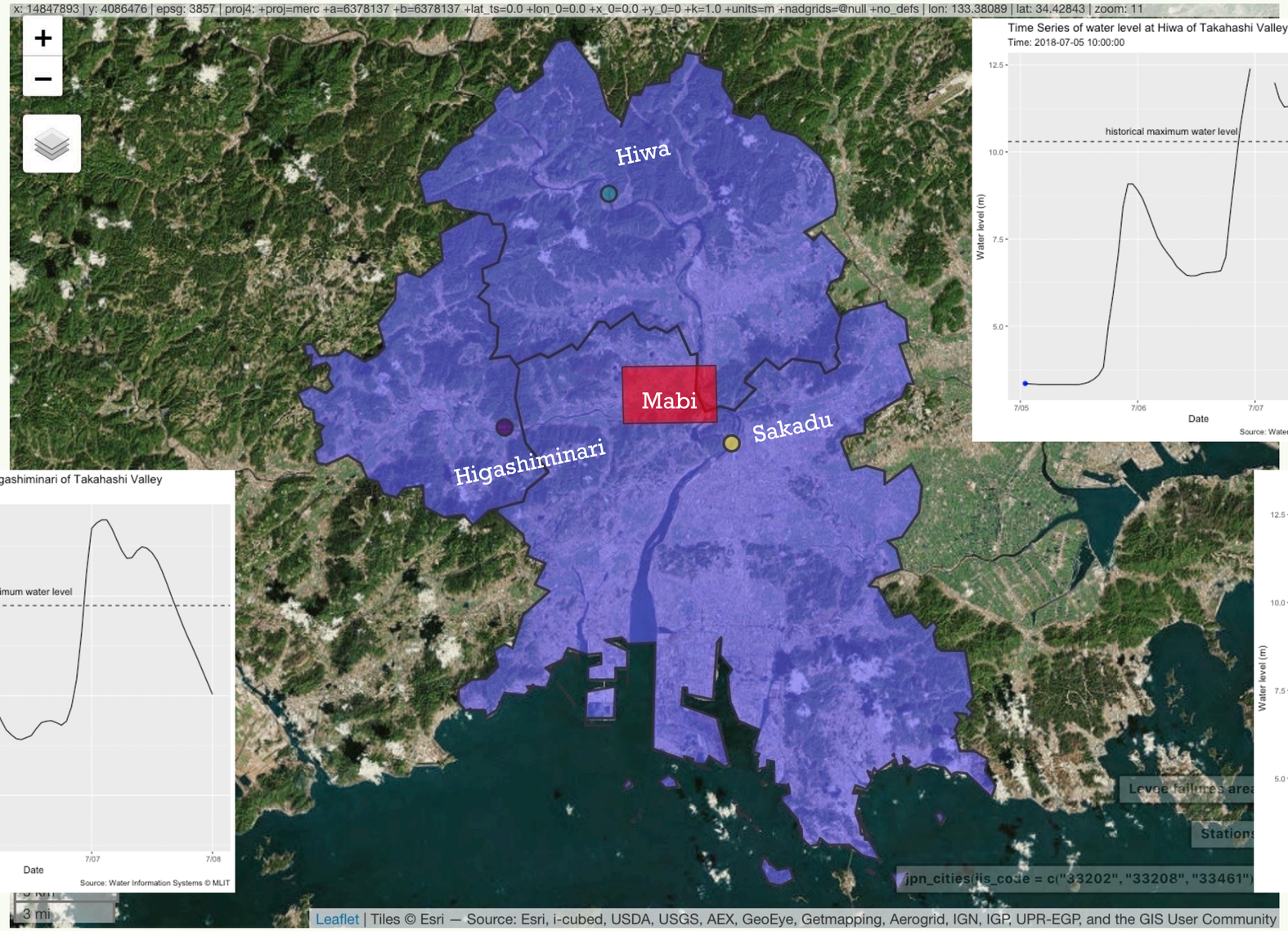
Levee failures in Mabi (真備)

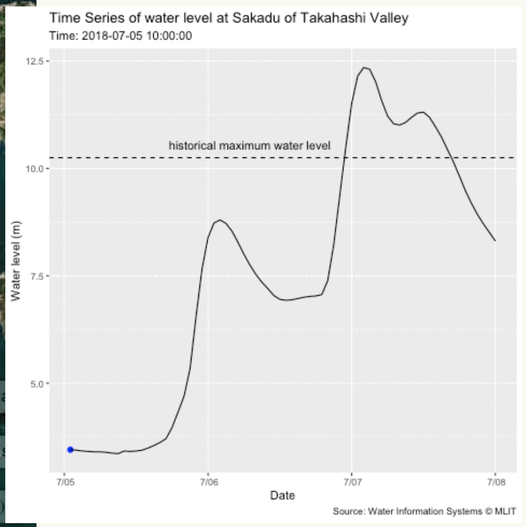
Embankment broke, flooding occurred

Water levels reaching 4.8 m



How changes the river's water level?





Session Information

	package	*	version	date	source
1	abind	*	1.4-5	2016-07-21	CRAN (R 3.5.0)
2	base	*	3.5.0	2018-04-24	local
3	bindrcpp	*	0.2.2	2018-03-29	CRAN (R 3.5.0)
4	datasets	*	3.5.0	2018-04-24	local
5	**dplyr**	*	0.7.6	2018-06-29	CRAN (R 3.5.0)
6	**forcats**	*	0.3.0	2018-02-19	CRAN (R 3.5.0)
7	**gganimate**	*	0.9.9.9999	2018-07-14	Github (thomasp85/gganimate@13a9a29)
8	**ggforce**	*	0.1.3	2018-07-07	CRAN (R 3.5.0)
9	**ggplot2**	*	3.0.0	2018-07-03	CRAN (R 3.5.0)
10	graphics	*	3.5.0	2018-04-24	local
11	grDevices	*	3.5.0	2018-04-24	local
12	**jmastats**	*	0.0.0.9000	2018-07-13	git (@a90e3a3)
13	**jpmesh**	*	1.1.1.9000	2018-06-26	local (uribo/jpmesh@5054663)
14	**jpndistrict**	*	0.3.2	2018-06-14	CRAN (R 3.5.0)
15	**mapview**	*	2.4.0	2018-04-28	CRAN (R 3.5.0)
16	methods	*	3.5.0	2018-04-24	local
17	**purrr**	*	0.2.5	2018-05-29	CRAN (R 3.5.0)
18	**raster**	*	2.6-7	2017-11-13	CRAN (R 3.5.0)
19	readr	*	1.1.1	2017-05-16	CRAN (R 3.5.0)
20	**sf**	*	0.6-3	2018-05-17	CRAN (R 3.5.0)
21	sp	*	1.3-1	2018-06-05	CRAN (R 3.5.0)
22	**stars**	*	0.1-1	2018-07-13	Github (r-spatial/stars@c9af832)
23	stats	*	3.5.0	2018-04-24	local
24	stringr	*	1.3.1	2018-05-10	CRAN (R 3.5.0)
25	**tibble**	*	1.4.2	2018-01-22	CRAN (R 3.5.0)
26	tidyr	*	0.8.1	2018-05-18	CRAN (R 3.5.0)
27	tidyverse	*	1.2.1	2017-11-14	CRAN (R 3.5.0)
28	utils	*	3.5.0	2018-04-24	local

setting	value
version	R version 3.5.0 (2018-04-23)
system	x86_64, darwin15.6.0
ui	RStudio (1.2.826)
language	En
collate	ja_JP.UTF-8
tz	Asia/Tokyo
date	2018-07-18

ENJOY 

Shinya Uryu
@u_ribo
@uribo