# Europe COVID-19 deaths animated map

# data European Centre for Disease Prevention and Control

# packages

library(tidyverse) library(httr) library(fs) library(sf) library(readxl) library(janitor) library(glue) library(tmap) library(grid) library(classInt) library(magick)

# + btb, raster, fasterize, plyr

# sources

# https://data.europa.eu/euodp/en/data/dataset/covid-19-coronavirus-data

covid\_file <- "covid\_eu.csv"

covid\_url <- "https://opendata.ecdc.europa.eu/covid19/casedistribution/csv"

countries\_file <- "ne\_50m\_admin\_0\_countries.shp"

countries\_url <- "https://[www.naturalearthdata.com/http//www.naturalearthdata.com/download/](http://www.naturalearthdata.com/http//www.naturalearthdata.com/download/) 10m/cultural/ne\_10m\_admin\_0\_countries.zip"

# config

radius <- 600000 # smoothing radius (m) pixel <- 100000 # grid resolution (m)

force\_download <- FALSE # download even if already downloaded today ?

#' Kernel weighted smoothing with arbitrary bounding area #'

#' @param df sf object (points)

#' @param field weight field in the df

#' @param bandwidth kernel bandwidth (map units)

#' @param resolution output grid resolution (map units) #' @param zone sf study zone (polygon)

#' @param out\_crs EPSG (should be an equal-area projection) #'

#' @return a raster object

#' @import btb, raster, fasterize, dplyr, plyr, sf

lissage <- function(df, field, bandwidth, resolution, zone, out\_crs = 3035) { if (st\_crs(zone)$epsg != out\_crs) {

message("reprojecting data...") zone <- st\_transform(zone, out\_crs)

}

if (st\_crs(df)$epsg != out\_crs) { message("reprojecting study zone...") df <- st\_transform(df, out\_crs)

}

zone\_bbox <- st\_bbox(zone) # grid generation

message("generating reference grid...") zone\_xy <- zone %>%

dplyr::select(geometry) %>% st\_make\_grid(

cellsize = resolution,

offset = c(plyr::round\_any(zone\_bbox[1] - bandwidth, resolution, f = floor),

plyr::round\_any(zone\_bbox[2] - bandwidth, resolution, f =

floor)),

what = "centers") %>% st\_sf() %>%

st\_join(zone, join = st\_intersects, left = FALSE) %>% st\_coordinates() %>%

as\_tibble() %>% dplyr::select(x = X, y = Y)

# kernel

message("computing kernel...") kernel <- df %>%

cbind(., st\_coordinates(.)) %>% st\_set\_geometry(NULL) %>% dplyr::select(x = X, y = Y, field) %>% btb::kernelSmoothing(

dfObservations = ., sEPSG = out\_crs, iCellSize = resolution, iBandwidth = bandwidth, vQuantiles = NULL, dfCentroids = zone\_xy

)

# rasterization message("\nrasterizing...") raster::raster(

xmn = plyr::round\_any(zone\_bbox[1] - bandwidth, resolution, f = floor), ymn = plyr::round\_any(zone\_bbox[2] - bandwidth, resolution, f = floor), xmx = plyr::round\_any(zone\_bbox[3] + bandwidth, resolution, f = ceiling), ymx = plyr::round\_any(zone\_bbox[4] + bandwidth, resolution, f = ceiling), resolution = resolution

) %>%

fasterize::fasterize(kernel, ., field = field)

}

# download data

if (!dir\_exists("data")) dir\_create("data")

if (!dir\_exists("results")) dir\_create("results")

if (!dir\_exists("results/animation\_eu")) dir\_create("results/animation\_eu")

if (!file\_exists(path("data", covid\_file)) |

file\_info(path("data", covid\_file))$modification\_time < Sys.Date() | force\_download) {

GET(covid\_url, progress(),

write\_disk(path("data", covid\_file), overwrite = TRUE)) %>% stop\_for\_status()

}

if (!file\_exists(path("data", countries\_file))) { dl <- file\_temp()

GET(countries\_url, progress(), write\_disk(dl)) %>%

stop\_for\_status()

unzip(dl, exdir = "data")

}

# data

# some countries doesn't have data for the latest days ; we fill with latest # data

covid <- read\_csv(path("data", covid\_file),

col\_types = cols(dateRep = col\_date(format = "%d/%m/%Y"))) %>% clean\_names() %>%

complete(countryterritory\_code, date\_rep) %>% replace\_na(list(deaths = 0)) %>% group\_by(countryterritory\_code) %>% arrange(date\_rep) %>%

mutate(deaths\_cum = cumsum(deaths))

# keep only europen countries minus Russia and adding TUR and CYP # remove overseas territories, reproject in LAEA

countries <- read\_sf(path("data", countries\_file)) %>% clean\_names() %>%

filter(continent == "Europe" & iso\_a3\_eh != "RUS" | iso\_a3\_eh %in% c("TUR", "CYP")) %>%

st\_cast("POLYGON") %>% st\_set\_crs(4326) %>%

st\_join(c(xmin = -20, xmax = 35, ymin = 35, ymax = 70) %>% st\_bbox() %>%

st\_as\_sfc() %>% st\_as\_sf() %>% st\_set\_crs(4326),

left = FALSE) %>%

group\_by(iso\_a3\_eh) %>%

summarise(geometry = st\_combine(geometry)) %>% st\_transform(3035)

# pretreatment

# mask to generate grid : union all countries unioned\_countries\_file <- "data/eu.rds"

if (!file\_exists(unioned\_countries\_file)) { unioned\_countries <- countries %>%

st\_union() %>% st\_sf() %>%

write\_rds(unioned\_countries\_file)

} else {

unioned\_countries <- read\_rds(unioned\_countries\_file)

}

# join countries/data for a specific date create\_df <- function(territory, date = NULL) {

covid %>%

filter(date\_rep == if\_else(is.null(date), max(date\_rep), date)) %>% right\_join(countries,

by = c("countryterritory\_code" = "iso\_a3\_eh")) %>% st\_as\_sf() %>%

st\_point\_on\_surface() %>% drop\_na(deaths\_cum) %>% st\_as\_sf()

}

covid\_geo <- create\_df(countries)

# smoothing for last date

# deaths

d <- covid\_geo %>%

lissage("deaths\_cum", radius, pixel, unioned\_countries)

# population

p <- covid\_geo %>%

lissage("pop\_data2018", radius, pixel, unioned\_countries)

# grid per 100000 inhab death\_pop <- d \* 100000 / p

# carto

# classification for last date to be reused in animation set.seed(1234)

classes <- classIntervals(raster::values(death\_pop), n = 6, style = "kmeans", dataPrecision = 0)$brks

# animation

image\_animation <- function(date) { message(glue("\n\n{date}\n==========\n"))

m <- create\_df(countries, date) %>%

lissage("deaths\_cum", radius, pixel, unioned\_countries) %>% magrittr::divide\_by(p) %>%

magrittr::multiply\_by(100000) %>% tm\_shape() +

tm\_raster(title = glue("deaths

per 100 000 inhab."), style = "fixed",

breaks = classes, palette = "viridis",

legend.format = list(text.separator = "to less than", digits = 0),

legend.reverse = TRUE) +

tm\_layout(title = glue("COVID-19 - Europe\ncumulative as of {date}"), legend.position = c("right", "top"),

frame = FALSE) + #tm\_shape(countries, bbox = death\_pop) + #tm\_borders() + tm\_credits(glue("<http://r.iresmi.net/>

bisquare kernel smoothing {radius / 1000} km on {pixel / 1000}

km grid

Naturalearth"),

classif. kmeans, LAEA Europe projection

data European Centre for Disease Prevention and Control / map

size = .5,

position = c(.5, .025))

message("saving map...")

tmap\_save(m, glue("results/animation\_eu/covid\_eu\_{date}.png"), width = 800, height = 800, scale = .4,)

}

covid %>%

filter(date\_rep >= "2020-03-15") %>% pull(date\_rep) %>%

unique() %>% walk(image\_animation)

animation <- glue("results/deaths\_covid19\_eu\_{max(covid$date\_rep)}.gif") dir\_ls("results/animation\_eu") %>%

map(image\_read) %>% image\_join() %>% #image\_scale("500x500") %>% image\_morph(frames = 1) %>%

image\_animate(fps = 2, optimize = TRUE) %>% image\_write(animation)…