R\_AS\_GIS

Katharina Kaelin

2019

Table of Contents

Die Beispiel wurden vom Staristischen Amt des Kantons Zürich publiziert.

Die Portierung auf die RStudio-Umgebung wurde durch Peter Berger realisiert. Die ganzem Bereitstellung und Dokumentationsteile wurden aus Arbeiten von Timo Grossenbacher von SF Data übrnommen und adapiert.

## Notes

This report was generated on 2019-09-22 13:25:59. R version: 3.6.1 on i386-w64-mingw32. For this report, CRAN packages as of 2019-09-21 were used.

…

### R-Script & data

The preprocessing and analysis of the data was conducted in the [R project for statistical computing](https://www.r-project.org/). The RMarkdown script used to generate this document and all the resulting data can be downloaded [under this link](http://tgdbepe4.github.io/R_AS_GIS/rscript.zip). Through executing main.Rmd, the herein described process can be reproduced and this document can be generated. In the course of this, data from the folder ìnput will be processed and results will be written to output.

### GitHub

The code for the herein described process can also be freely downloaded from <https://github.com/tgdbepe4/R_AS_GIS>.

### License

…

### Data description of output files

#### abc.csv (Example)

|  |  |  |
| --- | --- | --- |
| Attribute | Type | Description |
| a | Numeric | … |
| b | Numeric | … |
| c | Numeric | … |

### xyz.csv

…

## Preparations

## Loading required package: knitr

## Loading required package: rstudioapi

## [1] "package package:rstudioapi detached"

## Warning: 'knitr' namespace cannot be unloaded:  
## namespace 'knitr' is imported by 'rmarkdown' so cannot be unloaded

## [1] "package package:knitr detached"

### Define packages

# from https://mran.revolutionanalytics.com/web/packages/checkpoint/vignettes/using-checkpoint-with-knitr.html  
# if you don't need a package, remove it from here (commenting is probably not sufficient)  
# tidyverse: see https://blog.rstudio.org/2016/09/15/tidyverse-1-0-0/  
cat("  
library(rstudioapi)  
library(tidyverse) # ggplot2, dplyr, tidyr, readr, purrr, tibble  
library(magrittr) # pipes  
library(readxl) # excel  
library(scales) # scales for ggplot2  
library(jsonlite) # json  
library(lintr) # code linting  
library(sf) # spatial data handling  
library(rmarkdown) # Import libraries  
library(tidyverse) # collection of R packages designed for data science  
# Vector  
library(sf) ## GIS vector library new  
library(sp) ## GIS vector library old  
# Raster  
library(raster) ## GIS raster library   
# Visualization  
library(RColorBrewer) ## ready-to-use color palettes for creating beautiful graphics  
library(rmapshaper) ## used to simplify geometries  
library(tmap) ## easy to use approach to to create theamtic maps  
library(mapview) ## interactive visualisations of spatial data  
library(classInt) ## methods for choosing univariate class intervals for mapping or other graphics purposes",  
file = "manifest.R")

### Install packages

# if checkpoint is not yet installed, install it (for people using this  
# system for the first time)  
if (!require(checkpoint)) {  
 if (!require(devtools)) {  
 install.packages("devtools", repos = "http://cran.us.r-project.org")  
 require(devtools)  
 }  
 devtools::install\_github("RevolutionAnalytics/checkpoint",  
 ref = "v0.3.2", # could be adapted later,  
 # as of now (beginning of July 2017  
 # this is the current release on CRAN)  
 repos = "http://cran.us.r-project.org")  
 require(checkpoint)  
}  
# nolint start  
if (!dir.exists("~/.checkpoint")) {  
 dir.create("~/.checkpoint")  
}  
# nolint end  
# install packages for the specified CRAN snapshot date  
checkpoint(snapshotDate = package\_date,  
 project = path\_to\_wd,  
 verbose = T,  
 scanForPackages = T,  
 use.knitr = F,  
 R.version = R\_version)  
rm(package\_date)

### Load packages

source("manifest.R")  
unlink("manifest.R")  
sessionInfo()

## R version 3.6.1 (2019-07-05)  
## Platform: i386-w64-mingw32/i386 (32-bit)  
## Running under: Windows 10 x64 (build 18362)  
##   
## Matrix products: default  
##   
## locale:  
## [1] LC\_COLLATE=German\_Switzerland.1252 LC\_CTYPE=German\_Switzerland.1252   
## [3] LC\_MONETARY=German\_Switzerland.1252 LC\_NUMERIC=C   
## [5] LC\_TIME=German\_Switzerland.1252   
##   
## attached base packages:  
## [1] stats graphics grDevices utils datasets methods base   
##   
## other attached packages:  
## [1] classInt\_0.4-1 mapview\_2.7.0 tmap\_2.3-1   
## [4] rmapshaper\_0.4.1 RColorBrewer\_1.1-2 raster\_3.0-2   
## [7] sp\_1.3-1 rmarkdown\_1.15 sf\_0.7-7   
## [10] lintr\_1.0.3 jsonlite\_1.6 scales\_1.0.0   
## [13] readxl\_1.3.1 magrittr\_1.5 forcats\_0.4.0   
## [16] stringr\_1.4.0 dplyr\_0.8.3 purrr\_0.3.2   
## [19] readr\_1.3.1 tidyr\_1.0.0 tibble\_2.1.3   
## [22] ggplot2\_3.2.1 tidyverse\_1.2.1 rstudioapi\_0.10   
## [25] checkpoint\_0.4.7   
##   
## loaded via a namespace (and not attached):  
## [1] Rcpp\_1.0.1 knitr\_1.23 xml2\_1.2.2   
## [4] units\_0.6-4 hms\_0.5.1 rvest\_0.3.4   
## [7] tidyselect\_0.2.5 viridisLite\_0.3.0 xtable\_1.8-4   
## [10] jsonvalidate\_1.1.0 colorspace\_1.4-1 lattice\_0.20-38   
## [13] R6\_2.4.0 rlang\_0.4.0 rgdal\_1.4-4   
## [16] broom\_0.5.2 xfun\_0.8 e1071\_1.7-2   
## [19] modelr\_0.1.5 withr\_2.1.2 rgeos\_0.5-1   
## [22] leafsync\_0.1.0 htmltools\_0.3.6 class\_7.3-15   
## [25] leaflet\_2.0.2 assertthat\_0.2.1 digest\_0.6.19   
## [28] httpcode\_0.2.0 lifecycle\_0.1.0 shiny\_1.3.2   
## [31] crul\_0.8.4 curl\_4.0 haven\_2.1.1   
## [34] compiler\_3.6.1 cellranger\_1.1.0 pillar\_1.4.2   
## [37] backports\_1.1.4 generics\_0.0.2 stats4\_3.6.1   
## [40] geojsonlint\_0.3.0 satellite\_1.0.1 lubridate\_1.7.4   
## [43] httpuv\_1.5.2 pkgconfig\_2.0.2 rex\_1.1.2   
## [46] munsell\_0.5.0 httr\_1.4.1 tools\_3.6.1   
## [49] webshot\_0.5.1 dichromat\_2.0-0 grid\_3.6.1   
## [52] nlme\_3.1-140 gtable\_0.3.0 png\_0.1-7   
## [55] KernSmooth\_2.23-15 DBI\_1.0.0 cli\_1.1.0   
## [58] crosstalk\_1.0.0 lazyeval\_0.2.2 yaml\_2.2.0   
## [61] lwgeom\_0.1-7 crayon\_1.3.4 later\_0.8.0   
## [64] base64enc\_0.1-3 tmaptools\_2.0-2 htmlwidgets\_1.3   
## [67] promises\_1.0.1 codetools\_0.2-16 vctrs\_0.2.0   
## [70] zeallot\_0.1.0 mime\_0.7 glue\_1.3.1   
## [73] evaluate\_0.14 V8\_2.3 stringi\_1.4.3   
## [76] XML\_3.98-1.20

### Load additional scripts

# if you want to outsource logic to other script files, see README for   
# further information  
knitr::read\_chunk("scripts/my\_script.R")  
source("scripts/my\_script.R")  
my\_function(5)

## [1] 5

##Import data ##Public transportation stops The public transportation stops dataset was downloded here: <https://data.geo.admin.ch/ch.bav.haltestellen-oev/data.zip> Release date: 06-08-2018 Format: csv

# Import csv as df  
pts\_df <- read.csv("./Data\_in/Betriebspunkt.csv", stringsAsFactors=FALSE, header= TRUE)   
  
# Select desired attributes  
pts\_df <- pts\_df %>%  
 dplyr::select(pts\_ID = "Nummer", pts\_MunNr = "GdeNummer", "y\_Koord\_Ost", "x\_Koord\_Nord")   
# !! Thre is a small mistake in this dataset: The X and Y values are swapped. Correctly, the attributes should be named as follows: x\_Koord\_Ost, y\_Koord\_Nord.   
  
# Convert df to spatial df  
pts\_sf <- sf::st\_as\_sf(x = pts\_df, coords = c("y\_Koord\_Ost", "x\_Koord\_Nord"), crs= 2056) # epsg:2056 is the ID of LV95  
  
# Check data  
class(pts\_df)

## [1] "data.frame"

## [1] "data.frame"  
pts\_df[1:5,]

## pts\_ID pts\_MunNr y\_Koord\_Ost x\_Koord\_Nord  
## 1 8508186 338 2628463 1220751  
## 2 8587698 6173 2643683 1132003  
## 3 8531188 6252 2613318 1109828  
## 4 8530051 5409 2572770 1129980  
## 5 8501442 6057 2650331 1141959

## pts\_ID pts\_MunNr y\_Koord\_Ost x\_Koord\_Nord  
## 1 8508186 338 2628463 1220751  
## 2 8587698 6173 2643683 1132003  
## 3 8531188 6252 2613318 1109828  
## 4 8530051 5409 2572770 1129980  
## 5 8501442 6057 2650331 1141959  
  
class(pts\_sf)

## [1] "sf" "data.frame"

## [1] "sf" "data.frame"  
pts\_sf[1:5,]

## Simple feature collection with 5 features and 2 fields  
## geometry type: POINT  
## dimension: XY  
## bbox: xmin: 2572770 ymin: 1109828 xmax: 2650331 ymax: 1220751  
## epsg (SRID): 2056  
## proj4string: +proj=somerc +lat\_0=46.95240555555556 +lon\_0=7.439583333333333 +k\_0=1 +x\_0=2600000 +y\_0=1200000 +ellps=bessel +towgs84=674.374,15.056,405.346,0,0,0,0 +units=m +no\_defs  
## pts\_ID pts\_MunNr geometry  
## 1 8508186 338 POINT (2628463 1220751)  
## 2 8587698 6173 POINT (2643683 1132003)  
## 3 8531188 6252 POINT (2613318 1109828)  
## 4 8530051 5409 POINT (2572770 1129980)  
## 5 8501442 6057 POINT (2650331 1141959)

## Simple feature collection with 5 features and 2 fields  
## geometry type: POINT  
## dimension: XY  
## bbox: xmin: 2572770 ymin: 1109830 xmax: 2650330 ymax: 1220750  
## epsg (SRID): 2056  
## proj4string: +proj=somerc +lat\_0=46.95240555555556 +lon\_0=7.439583333333333 +k\_0=1 +x\_0=2600000 +y\_0=1200000 +ellps=bessel +towgs84=674.374,15.056,405.346,0,0,0,0 +units=m +no\_defs  
## pts\_ID pts\_MunNr geometry  
## 1 8508186 338 POINT (2628463 1220751)  
## 2 8587698 6173 POINT (2643683 1132003)  
## 3 8531188 6252 POINT (2613318 1109828)  
## 4 8530051 5409 POINT (2572770 1129980)  
## 5 8501442 6057 POINT (2650331 1141959)

##Municipality The municipality dataset was downloded here: <https://shop.swisstopo.admin.ch/de/products/landscape/boundaries3D> Release date: 2019 Format: Shapefile

# Import shp as spatial df  
mun\_sf <- sf::st\_read("./Data\_in/swissBOUNDARIES3D\_1\_3\_TLM\_HOHEITSGEBIET.shp", stringsAsFactors = FALSE, crs=2056)

## Reading layer `swissBOUNDARIES3D\_1\_3\_TLM\_HOHEITSGEBIET' from data source `E:\R\_github\R\_AS\_GIS\analysis\Data\_in\swissBOUNDARIES3D\_1\_3\_TLM\_HOHEITSGEBIET.shp' using driver `ESRI Shapefile'  
## Simple feature collection with 2361 features and 23 fields  
## geometry type: POLYGON  
## dimension: XYZ  
## bbox: xmin: 2485410 ymin: 1075268 xmax: 2833858 ymax: 1295934  
## epsg (SRID): 2056  
## proj4string: +proj=somerc +lat\_0=46.95240555555556 +lon\_0=7.439583333333333 +k\_0=1 +x\_0=2600000 +y\_0=1200000 +ellps=bessel +towgs84=674.374,15.056,405.346,0,0,0,0 +units=m +no\_defs

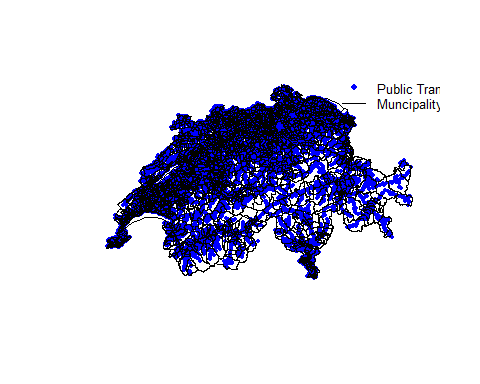
## Reading layer `swissBOUNDARIES3D\_1\_3\_TLM\_HOHEITSGEBIET' from data source `C:\gitrepos\R\_AS\_GIS\Data\_in\swissBOUNDARIES3D\_1\_3\_TLM\_HOHEITSGEBIET.shp' using driver `ESRI Shapefile'  
## Simple feature collection with 2361 features and 23 fields  
## geometry type: POLYGON  
## dimension: XYZ  
## bbox: xmin: 2485410 ymin: 1075270 xmax: 2833860 ymax: 1295930  
## epsg (SRID): 2056  
## proj4string: +proj=somerc +lat\_0=46.95240555555556 +lon\_0=7.439583333333333 +k\_0=1 +x\_0=2600000 +y\_0=1200000 +ellps=bessel +towgs84=674.374,15.056,405.346,0,0,0,0 +units=m +no\_defs  
  
# Select desired attributes  
mun\_sf <- mun\_sf %>%  
 dplyr::select(mun\_MunNr = "BFS\_NUMMER", mun\_CanNr = "KANTONSNUM") %>%  
 dplyr::group\_by(mun\_MunNr)%>%  
 dplyr::summarize(  
 mun\_CanNr = unique(mun\_CanNr)  
 ) %>%  
 st\_zm(drop = TRUE)  
  
# Check data  
class(mun\_sf)

## [1] "sf" "tbl\_df" "tbl" "data.frame"

## [1] "sf" "tbl\_df" "tbl" "data.frame"  
mun\_sf[1:5,]

## Simple feature collection with 5 features and 2 fields  
## geometry type: POLYGON  
## dimension: XY  
## bbox: xmin: 2673826 ymin: 1230185 xmax: 2686463 ymax: 1243159  
## epsg (SRID): 2056  
## proj4string: +proj=somerc +lat\_0=46.95240555555556 +lon\_0=7.439583333333333 +k\_0=1 +x\_0=2600000 +y\_0=1200000 +ellps=bessel +towgs84=674.374,15.056,405.346,0,0,0,0 +units=m +no\_defs  
## # A tibble: 5 x 3  
## mun\_MunNr mun\_CanNr geometry  
## <int> <int> <POLYGON [m]>  
## 1 1 1 ((2680492 1235035, 2680482 1235041, 2680473 1235046,~  
## 2 2 1 ((2674714 1236942, 2674699 1237008, 2674689 1237061,~  
## 3 3 1 ((2679008 1241960, 2679015 1241965, 2679022 1241971,~  
## 4 4 1 ((2686028 1230192, 2686026 1230195, 2686024 1230205,~  
## 5 5 1 ((2678523 1238540, 2678511 1238544, 2678431 1238561,~

## Simple feature collection with 5 features and 2 fields  
## geometry type: POLYGON  
## dimension: XY  
## bbox: xmin: 2673830 ymin: 1230190 xmax: 2686460 ymax: 1243160  
## epsg (SRID): 2056  
## proj4string: +proj=somerc +lat\_0=46.95240555555556 +lon\_0=7.439583333333333 +k\_0=1 +x\_0=2600000 +y\_0=1200000 +ellps=bessel +towgs84=674.374,15.056,405.346,0,0,0,0 +units=m +no\_defs  
## # A tibble: 5 x 3  
## mun\_MunNr mun\_CanNr geometry  
## <int> <int> <POLYGON [m]>  
## 1 1 1 ((2680492 1235035, 2680482 1235041, 2680473 1235046,~  
## 2 2 1 ((2674714 1236942, 2674699 1237008, 2674689 1237061,~  
## 3 3 1 ((2679008 1241960, 2679015 1241965, 2679022 1241971,~  
## 4 4 1 ((2686028 1230192, 2686026 1230195, 2686024 1230205,~  
## 5 5 1 ((2678523 1238540, 2678511 1238544, 2678431 1238561,~  
  
# Plot imported data: R base plot  
plot(st\_geometry(pts\_sf), pch = 19, col="blue", cex = 0.5)  
plot(st\_geometry(mun\_sf), add = TRUE)  
legend(x=2750000,y=1310000,  
 c("Public Transportation Stops","Muncipality"),  
 lty=c(NA,1),  
 pch=c(19,NA),  
 cex=.8,   
 col=c("blue","black"),  
 bty='n'  
 )



##Calculate density of public transportation stops per municipality

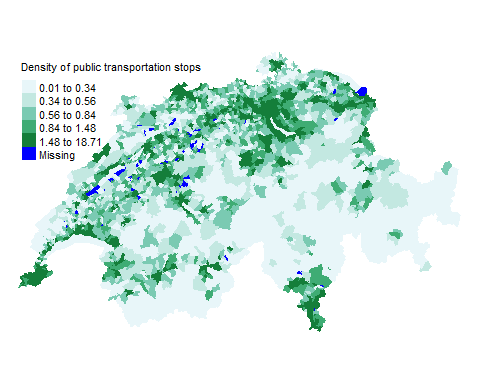
# Spatial Join: instead of joining dataframes via an equal ID we join data- frames based on an equal location.   
spjoin\_sf <- sf::st\_join(pts\_sf, mun\_sf)  
  
spjoin\_sf

## Simple feature collection with 26895 features and 4 fields  
## geometry type: POINT  
## dimension: XY  
## bbox: xmin: 2486195 ymin: 1075525 xmax: 2831996 ymax: 1294167  
## epsg (SRID): 2056  
## proj4string: +proj=somerc +lat\_0=46.95240555555556 +lon\_0=7.439583333333333 +k\_0=1 +x\_0=2600000 +y\_0=1200000 +ellps=bessel +towgs84=674.374,15.056,405.346,0,0,0,0 +units=m +no\_defs  
## First 10 features:  
## pts\_ID pts\_MunNr mun\_MunNr mun\_CanNr geometry  
## 1 8508186 338 338 2 POINT (2628463 1220751)  
## 2 8587698 6173 6173 23 POINT (2643683 1132003)  
## 3 8531188 6252 6252 23 POINT (2613318 1109828)  
## 4 8530051 5409 5409 22 POINT (2572770 1129980)  
## 5 8501442 6057 6057 23 POINT (2650331 1141959)  
## 6 8583656 6702 6702 26 POINT (2584548 1246665)  
## 7 8590028 351 351 2 POINT (2599934 1200621)  
## 8 8579143 5871 5871 22 POINT (2509845 1163460)  
## 9 8572195 2788 2788 13 POINT (2599991 1249781)  
## 10 8576536 576 576 2 POINT (2643212 1163308)

## Simple feature collection with 26895 features and 4 fields  
## geometry type: POINT  
## dimension: XY  
## bbox: xmin: 2486200 ymin: 1075520 xmax: 2832000 ymax: 1294170  
## epsg (SRID): 2056  
## proj4string: +proj=somerc +lat\_0=46.95240555555556 +lon\_0=7.439583333333333 +k\_0=1 +x\_0=2600000 +y\_0=1200000 +ellps=bessel +towgs84=674.374,15.056,405.346,0,0,0,0 +units=m +no\_defs  
## First 10 features:  
## pts\_ID pts\_MunNr mun\_MunNr mun\_CanNr geometry  
## 1 8508186 338 338 2 POINT (2628463 1220751)  
## 2 8587698 6173 6173 23 POINT (2643683 1132003)  
## 3 8531188 6252 6252 23 POINT (2613318 1109828)  
## 4 8530051 5409 5409 22 POINT (2572770 1129980)  
## 5 8501442 6057 6057 23 POINT (2650331 1141959)  
## 6 8583656 6702 6702 26 POINT (2584548 1246665)  
## 7 8590028 351 351 2 POINT (2599934 1200621)  
## 8 8579143 5871 5871 22 POINT (2509845 1163460)  
## 9 8572195 2788 2788 13 POINT (2599991 1249781)  
## 10 8576536 576 576 2 POINT (2643212 1163308)  
  
# Check: Did the Federal Office of Transport use the same municiaplity boundaries as we did in order to add the municipality inforamtion to the dataset (pts\_sf$pts\_MunNr)?  
spjoin\_sf\_check <- spjoin\_sf %>%  
 dplyr::select(mun\_MunNr,pts\_MunNr) %>%  
 dplyr::mutate (check = (spjoin\_sf$mun\_MunNr == spjoin\_sf$pts\_MunNr)) %>%  
 dplyr::arrange(check)  
  
table(spjoin\_sf\_check$check) #462/26408\*100 = 1.75%

##   
## FALSE TRUE   
## 475 26395

##   
## FALSE TRUE   
## 475 26395  
# ... no, they did not. Probably they used the same dataset with a different reference date.   
  
# Density calculation  
  
# > 1. Count points per polygon  
pts\_count <- spjoin\_sf %>%  
 dplyr::group\_by(mun\_MunNr) %>%  
 dplyr::summarise(count=n())  
  
mun\_sf <- mun\_sf %>%  
 dplyr::left\_join(pts\_count %>% st\_set\_geometry(NULL) , by = c("mun\_MunNr" ))  
  
# > 2. Calculate area of polygon  
mun\_sf <- mun\_sf %>%  
 dplyr::mutate(mun\_area\_m2 =as.vector(sf::st\_area(.)))  
  
# > 3. Calculate density: count/area  
mun\_sf$density <- mun\_sf$count / mun\_sf$mun\_area\_m2 \* 1000000  
  
# Plot result: tmap  
# > tmap static  
tmap::tmap\_mode("plot")  
#tmap\_mode("view")  
tmap::tm\_shape(mun\_sf) +  
 tmap::tm\_fill("density",  
 title="Density of public transportation stops",  
 style="quantile",   
 palette="BuGn",   
 colorNA = "blue") +  
 tmap::tm\_layout(frame = FALSE)

 ##Calculate density of public transportation stops per canton

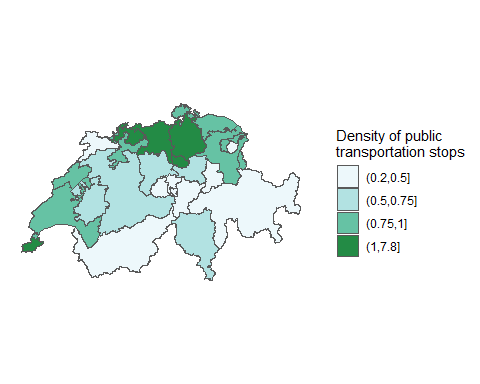
# group\_by mun\_CanNr  
canton\_sf <- mun\_sf %>%  
 dplyr::select(mun\_CanNr, count, mun\_area\_m2) %>%  
 dplyr::group\_by(mun\_CanNr)%>%  
 dplyr::summarize(  
 count = sum(count, na.rm = TRUE),  
 mun\_area\_m2 = sum(mun\_area\_m2, na.rm = TRUE)  
 ) %>%  
 dplyr::mutate(  
 density = round((count/mun\_area\_m2 \* 1000000),1)  
 )  
  
# Plot result: ggplot  
# Get quantile breaks. Add .00001 offset to catch the lowest value  
breaks\_qt <- classInt::classIntervals(c(min(canton\_sf$density) - .00001, canton\_sf$density), n = 4, style = "quantile")  
breaks\_qt$brks

## [1] 0.19999 0.50000 0.75000 1.00000 7.80000

## [1] 0.19999 0.50000 0.75000 1.00000 7.80000  
  
# Use cut to divice density into intervals and code them according to which interval they are in.  
canton\_sf <- canton\_sf %>%  
 dplyr::mutate(mycat = cut(density, breaks\_qt$brks)) %>%  
 dplyr::arrange(density)  
canton\_sf$mycat

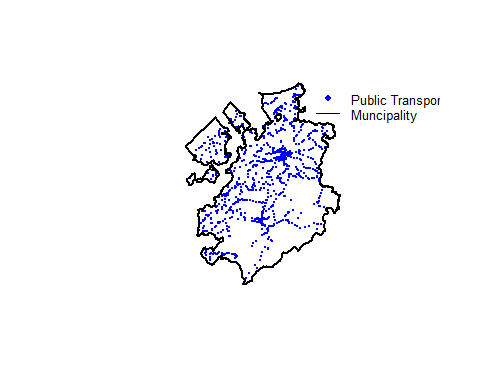
## [1] (0.2,0.5] (0.2,0.5] (0.2,0.5] (0.2,0.5] (0.2,0.5] (0.2,0.5]   
## [7] (0.2,0.5] (0.2,0.5] (0.5,0.75] (0.5,0.75] (0.5,0.75] (0.5,0.75]  
## [13] (0.5,0.75] (0.75,1] (0.75,1] (0.75,1] (0.75,1] (0.75,1]   
## [19] (0.75,1] (0.75,1] (0.75,1] (1,7.8] (1,7.8] (1,7.8]   
## [25] (1,7.8] (1,7.8] (1,7.8]   
## Levels: (0.2,0.5] (0.5,0.75] (0.75,1] (1,7.8]

## [1] (0.2,0.5] (0.2,0.5] (0.2,0.5] (0.2,0.5] (0.2,0.5] (0.2,0.5]   
## [7] (0.2,0.5] (0.2,0.5] (0.5,0.75] (0.5,0.75] (0.5,0.75] (0.5,0.75]  
## [13] (0.5,0.75] (0.75,1] (0.75,1] (0.75,1] (0.75,1] (0.75,1]   
## [19] (0.75,1] (0.75,1] (0.75,1] (1,7.8] (1,7.8] (1,7.8]   
## [25] (1,7.8] (1,7.8] (1,7.8]   
## Levels: (0.2,0.5] (0.5,0.75] (0.75,1] (1,7.8]  
  
ggplot2::ggplot(canton\_sf) +   
 geom\_sf(aes(fill=mycat)) +  
 scale\_fill\_brewer(palette = "BuGn", name = "Density of public\ntransportation stops") +   
 coord\_sf(datum=NA) + # no coordinate grid  
 theme\_bw() +# background = white  
 theme(  
 panel.border = element\_blank() # no border line around plot  
 )

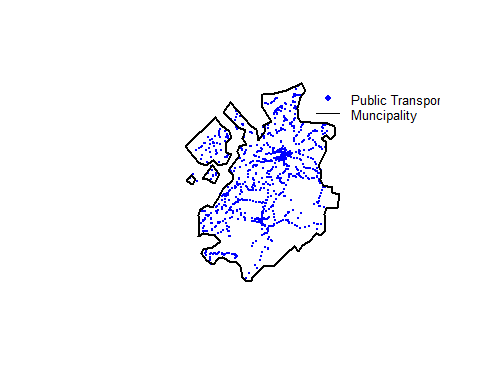


##Filter by canton: Freiburg

# Filter mun\_CanNr == 10   
pts\_freiburg<- spjoin\_sf %>%  
 dplyr::filter(mun\_CanNr == 10)  
  
canton\_freiburg <- canton\_sf %>%  
 dplyr::filter(mun\_CanNr == 10)  
  
# Plot result: R base plot  
plot(st\_geometry(pts\_freiburg), pch = 19, col="blue", cex = 0.1)  
plot(st\_geometry(canton\_freiburg), lwd = 2, add = TRUE)  
legend(x=2587000 ,y=1206350,  
 c("Public Transportation Stops","Muncipality"),  
 lty=c(NA,1),  
 pch=c(19,NA),  
 cex=.8,   
 col=c("blue","black"),  
 bty='n'  
 )



# This plot contains a lot of data as it contains a lot of coordinates. Do we really need this level of detail for our visualization? Often we don't.   
  
# Reduce level of detail  
canton\_freiburg\_gen <- rmapshaper::ms\_simplify(canton\_freiburg, keep = 0.01, keep\_shapes = TRUE)  
plot(st\_geometry(pts\_freiburg), pch = 19, col="blue", cex = 0.1)  
plot(st\_geometry(canton\_freiburg\_gen), lwd = 2, add = TRUE)  
legend(x=2587000 ,y=1206350,  
 c("Public Transportation Stops","Muncipality"),  
 lty=c(NA,1),  
 pch=c(19,NA),  
 cex=.8,   
 col=c("blue","black"),  
 bty='n'  
 )

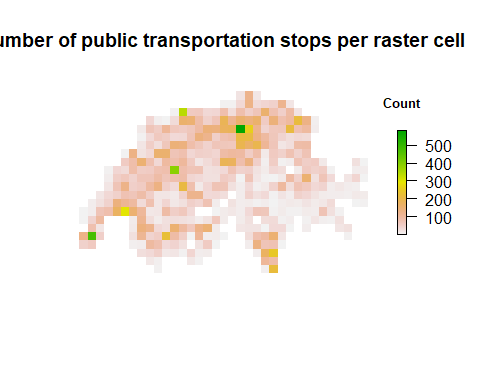


##Calculate number of public transportation stops per raster cells

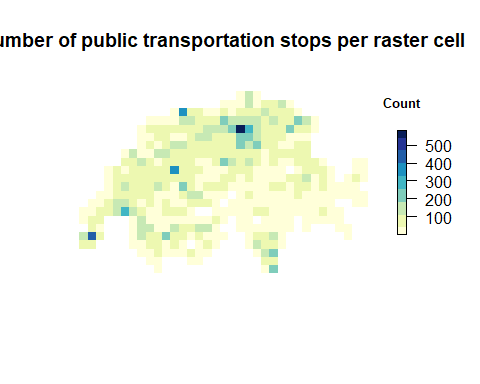
# We will use the sp package to carry out this analysis because point pattern analysis is still more established for this package.   
  
# sf to sp  
pts\_sp <- as(pts\_sf, Class = "Spatial")  
  
# Create grid with raster cell size 10'000 x 10'000m  
boundingbox\_pts\_sp = as(extent(pts\_sp), "SpatialPolygons")  
r = raster::raster(boundingbox\_pts\_sp, resolution = 10000)  
crs(r) <- CRS('+init=EPSG:2056')  
  
# Count points per raster cell   
rc = raster::rasterize(pts\_sp@coords, r, fun = "count")  
  
rc

## class : RasterLayer   
## dimensions : 22, 35, 770 (nrow, ncol, ncell)  
## resolution : 10000, 10000 (x, y)  
## extent : 2486195, 2836195, 1074167, 1294167 (xmin, xmax, ymin, ymax)  
## crs : +init=EPSG:2056 +proj=somerc +lat\_0=46.95240555555556 +lon\_0=7.439583333333333 +k\_0=1 +x\_0=2600000 +y\_0=1200000 +ellps=bessel +towgs84=674.374,15.056,405.346,0,0,0,0 +units=m +no\_defs   
## source : memory  
## names : layer   
## values : 1, 582 (min, max)

## class : RasterLayer   
## dimensions : 22, 35, 770 (nrow, ncol, ncell)  
## resolution : 10000, 10000 (x, y)  
## extent : 2486195, 2836195, 1074167, 1294167 (xmin, xmax, ymin, ymax)  
## coord. ref. : +init=EPSG:2056 +proj=somerc +lat\_0=46.95240555555556 +lon\_0=7.439583333333333 +k\_0=1 +x\_0=2600000 +y\_0=1200000 +ellps=bessel +towgs84=674.374,15.056,405.346,0,0,0,0 +units=m +no\_defs   
## data source : in memory  
## names : layer   
## values : 1, 582 (min, max)  
  
# Plot result: raster plot  
raster::plot(rc,   
 bty="n",   
 box=FALSE,   
 xaxt = "n",   
 yaxt = "n",   
 legend.args = list(text = 'Count', cex = 0.8, line = 1, font = 2),  
 main="Number of public transportation stops per raster cell")



# As many as 8 percent of men and 0.5 percent of women with Northern European ancestry have the common form of red-green color blindness. Thus, red and green colours should not be used together (http://east.nei.nih.gov:443/learn-about-eye-health/eye-conditions-and-diseases/color-blindness)  
  
my.palette = RColorBrewer::brewer.pal(n = 9, name = "YlGnBu")  
raster::plot(rc,   
 col = my.palette,   
 bty="n",   
 box=FALSE,   
 xaxt = "n", yaxt = "n",   
 legend.args = list(text = 'Count', cex = 0.8, line = 1, font = 2),   
 main="Number of public transportation stops per raster cell")

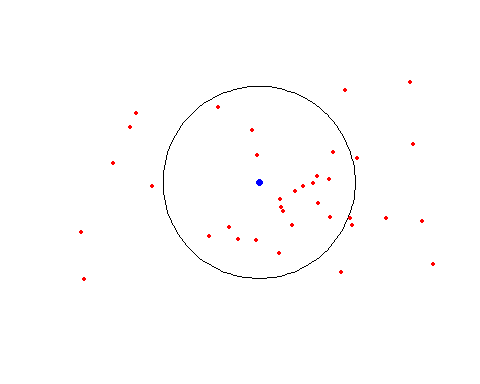


##Count number of public transportation stops in a defined area

# Somdbody asked us to calculate the number of public transportation stops 500m around this coordinate:   
N <- 47.37861  
E <- 8.53768  
  
# Create df  
mypoint\_df <- data.frame(N,E)  
  
# Convert df to spatial df  
mypoint\_sf <- sf::st\_as\_sf(x = mypoint\_df, coords = c("E", "N"), crs= 4326) %>% #epsg:4326 is the ID of WGS84  
 sf::st\_transform(2056)  
  
# Calculate area 500 m around mypoint\_sf  
myarea\_sf <- mypoint\_sf %>%  
 sf::st\_buffer(500) %>%  
 dplyr::mutate(name = "MyArea")  
  
# Count points per myarea\_sf  
spjoin\_sf <- st\_join(pts\_sf, myarea\_sf)  
  
pts\_count <- spjoin\_sf %>%   
 dplyr::group\_by(name) %>%  
 dplyr::summarise(count=n())  
  
pts\_count

## Simple feature collection with 2 features and 2 fields  
## geometry type: MULTIPOINT  
## dimension: XY  
## bbox: xmin: 2486195 ymin: 1075525 xmax: 2831996 ymax: 1294167  
## epsg (SRID): 2056  
## proj4string: +proj=somerc +lat\_0=46.95240555555556 +lon\_0=7.439583333333333 +k\_0=1 +x\_0=2600000 +y\_0=1200000 +ellps=bessel +towgs84=674.374,15.056,405.346,0,0,0,0 +units=m +no\_defs  
## # A tibble: 2 x 3  
## name count geometry  
## \* <chr> <int> <MULTIPOINT [m]>  
## 1 MyArea 20 (2682739 1247828, 2682785 1248499, 2682843 1247879, 2682890~  
## 2 <NA> 26875 (2486195 1111396, 2486530 1112028, 2486730 1111962, 2486957~

## Simple feature collection with 2 features and 2 fields  
## geometry type: MULTIPOINT  
## dimension: XY  
## bbox: xmin: 2486200 ymin: 1075520 xmax: 2832000 ymax: 1294170  
## epsg (SRID): 2056  
## proj4string: +proj=somerc +lat\_0=46.95240555555556 +lon\_0=7.439583333333333 +k\_0=1 +x\_0=2600000 +y\_0=1200000 +ellps=bessel +towgs84=674.374,15.056,405.346,0,0,0,0 +units=m +no\_defs  
## # A tibble: 2 x 3  
## name count geometry  
## \* <chr> <int> <MULTIPOINT [m]>  
## 1 MyArea 20 (2682739 1247828, 2682785 1248499, 2682843 1247879, 2682890~  
## 2 <NA> 26875 (2486195 1111396, 2486530 1112028, 2486730 1111962, 2486957~  
  
# Plot result: R base plot  
plot(st\_geometry(myarea\_sf, col="grey",border="black"))  
plot(st\_geometry(mypoint\_sf), pch = 19, col="blue", cex =1, add = TRUE)  
plot(st\_geometry(pts\_sf), pch = 19, col="red", cex =0.5, add = TRUE)



##Export Data

# shp  
sf::st\_write(canton\_freiburg\_gen, "./Data\_out/canton\_freiburg\_gen.shp", delete\_layer = TRUE)

## Deleting layer `canton\_freiburg\_gen' using driver `ESRI Shapefile'  
## Writing layer `canton\_freiburg\_gen' to data source `./Data\_out/canton\_freiburg\_gen.shp' using driver `ESRI Shapefile'  
## features: 1  
## fields: 5  
## geometry type: Multi Polygon

## Deleting layer `canton\_freiburg\_gen' using driver `ESRI Shapefile'  
## Writing layer `canton\_freiburg\_gen' to data source `./Data\_out/canton\_freiburg\_gen.shp' using driver `ESRI Shapefile'  
## features: 1  
## fields: 5  
## geometry type: Multi Polygon  
  
# json  
path\_4326 = "./Data\_out/mun\_join\_pt\_sf\_freiburg\_gen.json"  
file.remove(path\_4326)

## [1] TRUE

## [1] TRUE  
sf::st\_write(canton\_freiburg\_gen %>% st\_transform(4326), path\_4326, driver="GeoJSON")

## Writing layer `mun\_join\_pt\_sf\_freiburg\_gen' to data source `./Data\_out/mun\_join\_pt\_sf\_freiburg\_gen.json' using driver `GeoJSON'  
## features: 1  
## fields: 5  
## geometry type: Multi Polygon

## Writing layer `mun\_join\_pt\_sf\_freiburg\_gen' to data source `./Data\_out/mun\_join\_pt\_sf\_freiburg\_gen.json' using driver `GeoJSON'  
## features: 1  
## fields: 5  
## geometry type: Multi Polygon  
  
# tif  
raster::writeRaster(rc, filename= "./Data\_out/rc.tif", format="GTiff", overwrite=TRUE)

##Which canton has the longest border?

# Calculate line length  
canton\_line\_sf <- canton\_sf %>%  
 sf::st\_cast("MULTILINESTRING") %>%  
 dplyr::mutate(length\_km = round(as.vector(as.vector(sf::st\_length(.)))/1000,digits = 0))  
  
canton\_sf <-canton\_sf %>%  
 dplyr::mutate(  
 length\_km = canton\_line\_sf$length\_km  
 ) %>%  
 dplyr::arrange(desc(length\_km))  
  
# Print result  
mapview::mapview(canton\_sf,  
 map.types = c("OpenStreetMap")  
 )

