

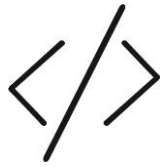
In 20 minutes


ICEPACK



TLDR;

- `sf` is a reasonably new **`R`** package that provides tidy-verse-like functionality
- It is going to replace `sp`
- You won't be wasting time learning how to wrangle with `sf`
- It's kinda fun :nerdy: because it's tidy

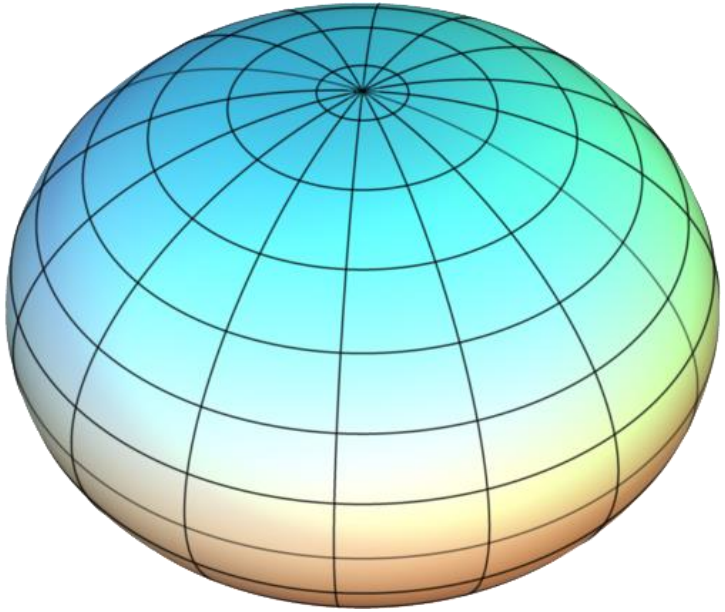


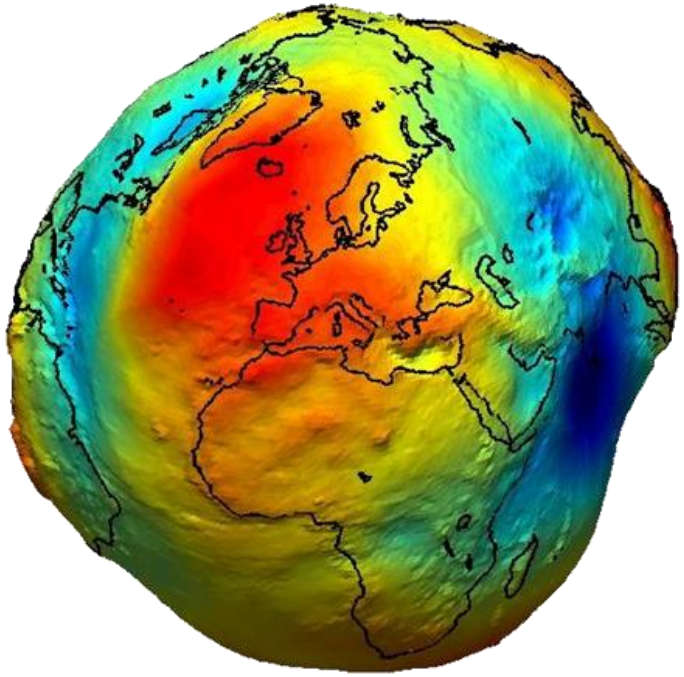
OVERVIEW

- Geometric primitives
- Geometries in **R** (using `sp`)
- `sf` basics
- Some worked examples
- Questions

</> WHY DO WE NEED A PACKAGE?







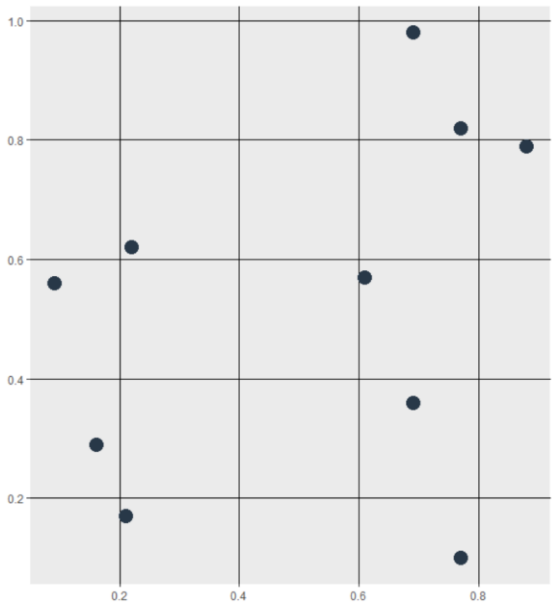
MODEL APPROXIMATIONS

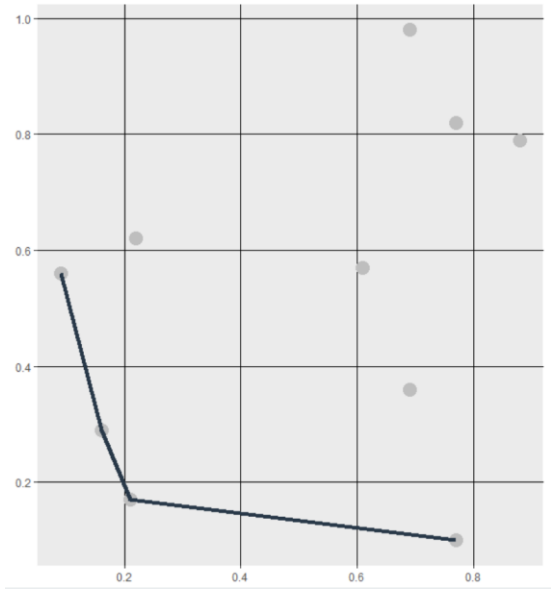
NGVD 29	Sea Level Datum 1929
OSGB36	Ordnance Survey Great Britain 1936
SK-42	Systema Koordinat 1942
ED50	European Datum 1950
SAD69	South American Datum 1969
GRS 80	Geodetic Reference System 1980
NAD 83	North American Datum 1983
WGS 84	World Geodetic System 1984
NAVD 88	N. American Vertical Datum 1988
ETRS89	European Terrestrial Reference System 1989
GCJ-02	Chinese obfuscated datum 2002
International Terrestrial Reference System	
Spatial Reference System Identifier (SRID)	
Universal Transverse Mercator (UTM)	

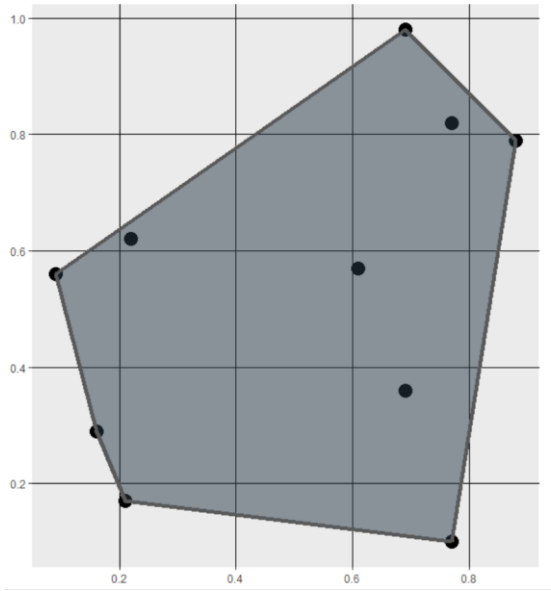
https://www.wikiwand.com/en/Figure_of_the_Earth

GEOMETRIC PRIMITIVES

- Points (and multi-points)
- Lines (and multi-lines)
- Polygons (and multi-polygons)







https://github.com/r-spatial/sf/wiki/Migrating

</> sp - CLASSIC GEOMETRIES IN R

```
library(sp)
xc = round(runif(10), 2)
yc = round(runif(10), 2)
xy = cbind(xc, yc)
xy.sp = SpatialPoints(xy)
str(xy.sp)
```

```
> str(xy.sp)
Formal class 'SpatialPoints' [package "sp"] with 3 slots
  ..@ coords     : num [1:10, 1:2] 0.94 0.6 0.52 0.06 0.84 0.54 0.75 0.44 0.88 0.93 ...
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : NULL
  .. .. ..$ : chr [1:2] "xc" "yc"
  ..@ bbox       : num [1:2, 1:2] 0.06 0.2 0.94 0.87
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : chr [1:2] "xc" "yc"
  .. .. ..$ : chr [1:2] "min" "max"
  ..@ proj4string:Formal class 'CRS' [package "sp"] with 1 slot
  .. .. ..@ projargs: chr NA
```

```
> typeof(xy.sp)
[1] "S4"
```

</> S4

- Attributes accessed through slots - @ accessor.

```
> xy.sp@coords
          xc   yc
 [1,] 0.94 0.71
 [2,] 0.60 0.52
 [3,] 0.52 0.20
 [4,] 0.06 0.33
 [5,] 0.84 0.27
 [6,] 0.54 0.77
 [7,] 0.75 0.27
 [8,] 0.44 0.87
 [9,] 0.88 0.69
[10,] 0.93 0.28
```

```
> xy.sp %>% select(coords)
Error in UseMethod("select_") :
  no applicable method for 'select_' applied to an object of class "c('SpatialPoints', 'Spatial')"
```

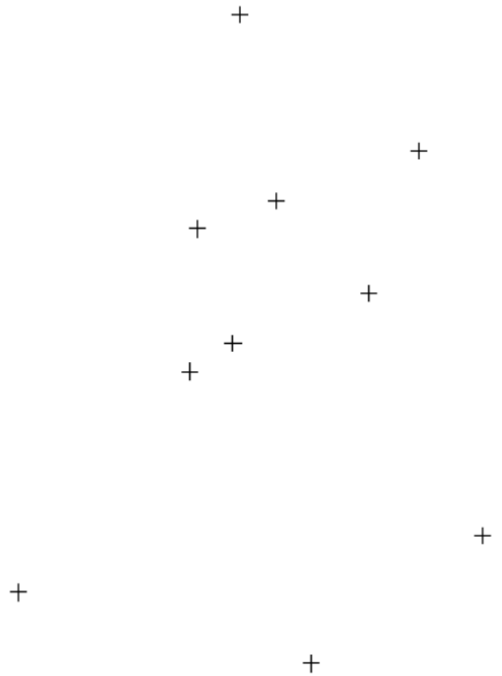
- Yuck

</> S4

```
plot(xy.sp)

xy.sp$stuff<- data.frame(morestuff = runif(10))
```

```
> xy.sp
    coordinates  morestuff
1  (0.38, 0.96) 0.89788740
2  (0.34, 0.56) 0.65803308
3  (0.84, 0.07) 0.78419931
4  (0.82, 0.62) 0.81756474
5  (0.15, 0.32) 0.77185702
6  (0.94, 0.76) 0.39817580
7  (0.71, 0.67) 0.91956325
8  (0.79, 0.76) 0.18234711
9  (0.41, 0.16) 0.08141645
10 (0.87, 0.04) 0.96167104
```



- Not half bad

</> JOINS

```
> xy.sp$col <- 1:10
> xy.sp
    coordinates  morestuff col
1   (0.71, 0.6) 0.36149498   1
2  (0.47, 0.69) 0.46381090   2
3  (0.46, 0.49) 0.07896394   3
4  (0.63, 0.08) 0.09799725   4
5  (0.53, 0.99) 0.64757856   5
6  (0.87, 0.26) 0.25846119   6
7  (0.22, 0.18) 0.43053265   7
8   (0.78, 0.8) 0.71552193   8
9  (0.58, 0.73) 0.76251542   9
10 (0.52, 0.53) 0.72598855  10
```

```
> s<- data.frame(col = sample(1:10, 5), rand = rnorm(5))
> s
  col       rand
1   4 -1.0671901
2   2 -0.7520194
3   6  0.4083095
4   1 -1.2663337
5  10  0.5058652
```

```
> s %>% left_join(xy.sp)
Error in UseMethod("tbl_vars") :
  no applicable method for 'tbl_vars' applied to an object of class "c('SpatialPointsDataFrame', 'SpatialPoints', 'Spat
```

- Yuck

</> JOINS

```
> xy.sp$col <- 1:10
> xy.sp
    coordinates  morestuff col
1   (0.71, 0.6) 0.36149498   1
2  (0.47, 0.69) 0.46381090   2
3  (0.46, 0.49) 0.07896394   3
4  (0.63, 0.08) 0.09799725   4
5  (0.53, 0.99) 0.64757856   5
6  (0.87, 0.26) 0.25846119   6
7  (0.22, 0.18) 0.43053265   7
8   (0.78, 0.8) 0.71552193   8
9  (0.58, 0.73) 0.76251542   9
10 (0.52, 0.53) 0.72598855  10
```

```
> s %>% left_join(xy.sp@data)
Joining, by = "col"
  col       rand  morestuff
1   4 -1.0671901 0.09799725
2   2 -0.7520194 0.46381090
3   6  0.4083095 0.25846119
4   1 -1.2663337 0.36149498
5  10  0.5058652 0.72598855
```

```
> s<- data.frame(col = sample(1:10, 5), rand = rnorm(5))
> s
  col       rand
1   4 -1.0671901
2   2 -0.7520194
3   6  0.4083095
4   1 -1.2663337
5  10  0.5058652
```

- More Yuck



sp

- There's a lot not to like
- There are a lot of very useful spatial functions in `sp`
- In the absence of a good alternative, we'll use it

- `sf` is the alternative



TIDY-VERSE MANIFESTO

- Reuse existing data structures.
- Compose simple functions with the pipe.
- Embrace functional programming.
- Design for humans.

https://cran.r-project.org/web/packages/tidyverse/vignettes/manifesto.html



sf

- Simple features
- Uses WKT
- Supports WKB
- Tidy compliant

columns

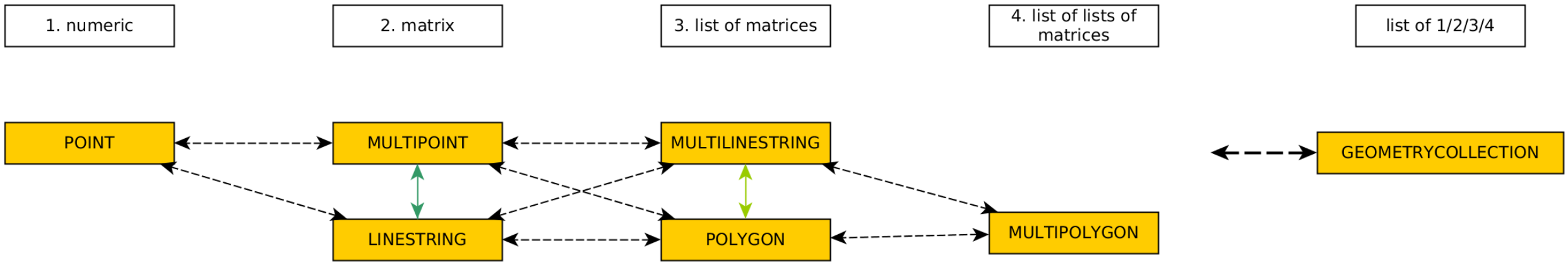
well known text

well known binary

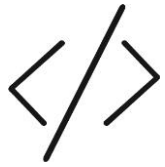
inherits data.frame

- Can coerce between different geometry types

TYPE COERCION



https://github.com/r-spatial/sf/issues/110



SOME TRIVIAL EXAMPLES

- Make some points
- Make some clusters
- Voronoi tessellations
- Polylines

</> POINTS

```
> xy = data.frame(x = round(runif(10),2),
+                 y = round(runif(10),2))
> xy$geom<-  st_geometry(st_as_sf(xy,coords = c("x", "y")))
> xy
      x    y                    geom
1  0.75 0.67 POINT (0.75 0.67)
2  0.83 0.32 POINT (0.83 0.32)
3  0.87 0.12 POINT (0.87 0.12)
4  0.18 0.22 POINT (0.18 0.22)
5  0.90 0.11  POINT (0.9 0.11)
6  0.62 0.28 POINT (0.62 0.28)
7  0.95 0.52 POINT (0.95 0.52)
8  0.05 0.35 POINT (0.05 0.35)
9  0.15 0.85 POINT (0.15 0.85)
10 0.10 0.43  POINT (0.1 0.43)
```

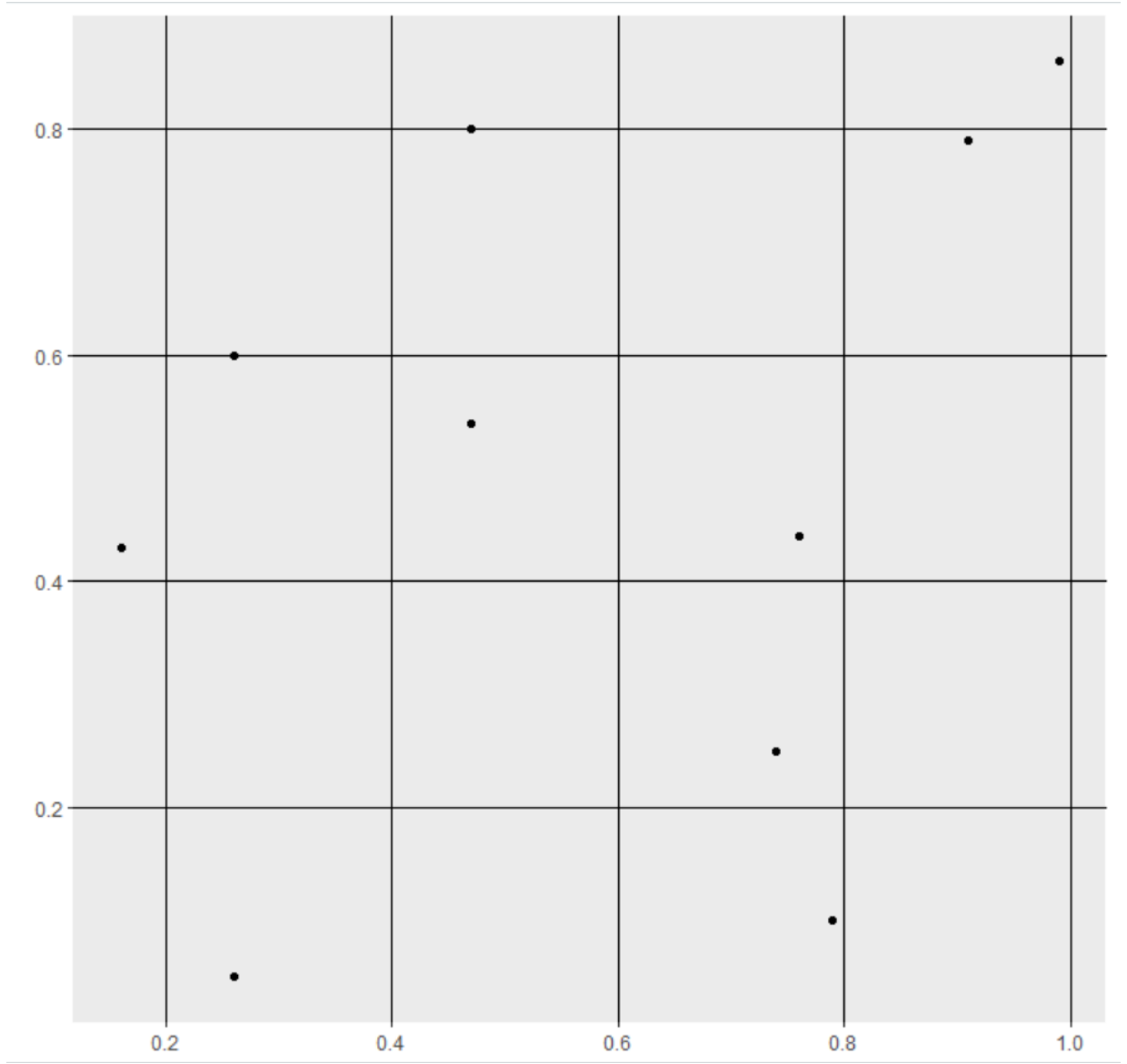
POINTS

```
> plot(sf_df$geometry, pch = 16)
```



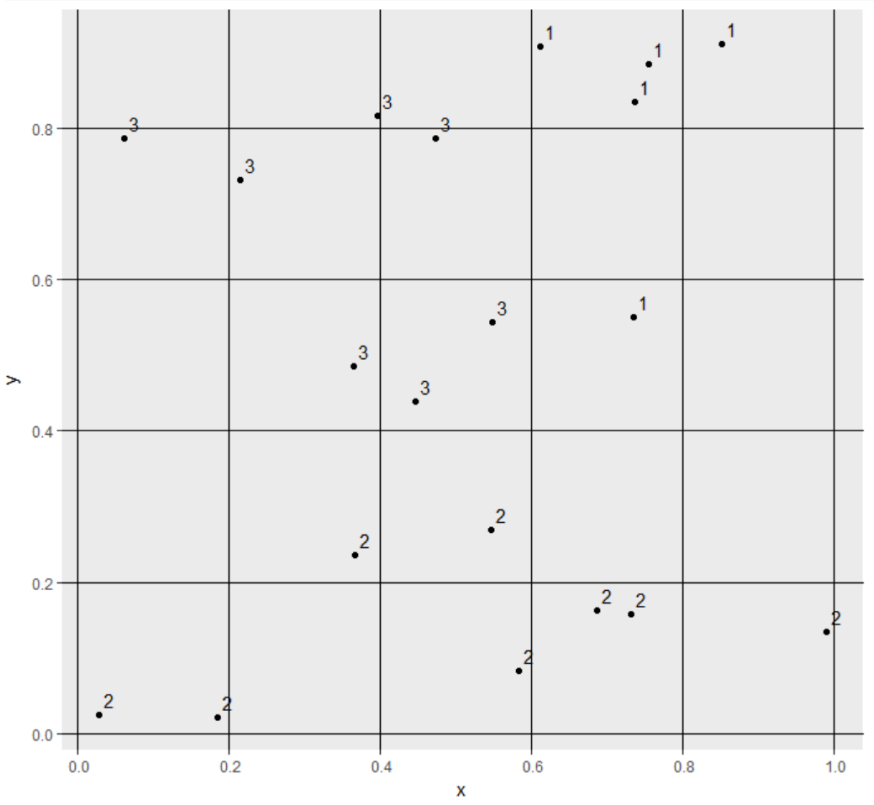
POINTS

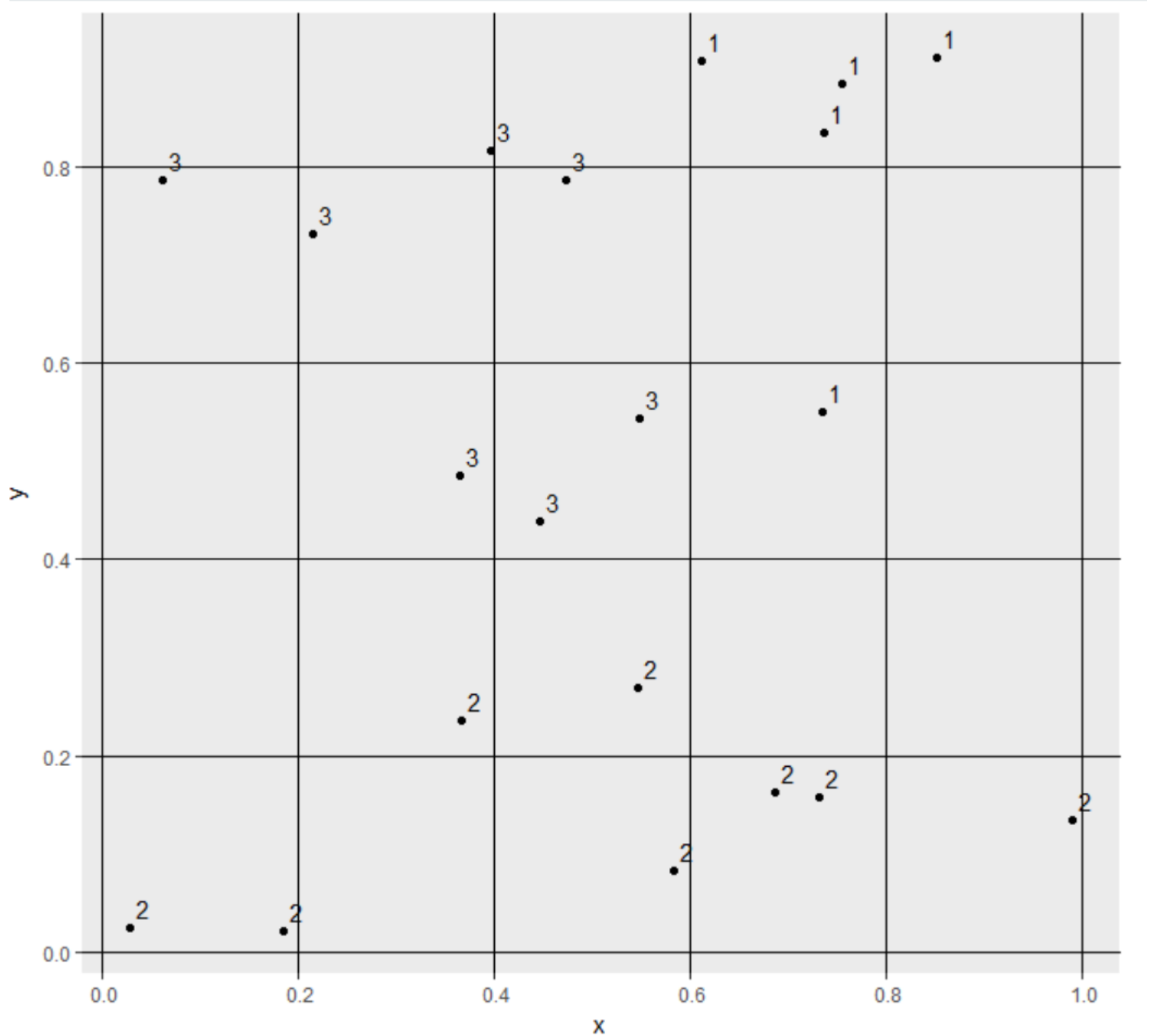
```
> ggplot(sf_df) + geom_sf()
```



CLUSTERS

```
> n= 20
> xy = data.frame(x = runif(n), y = runif(n))
> xy$id <- 1:n
> xy$geom<-  st_geometry(st_as_sf(xy,coords = c("x", "y")))
> xy$cluster<-  (xy %>% select(x, y) %>% kmeans(3))$cluster %>% as.character()
```




y
x
0.0
0.2
0.4
0.6
0.8
1.0

</> CLUSTERS

```
> xy %>% st_as_sf()
Simple feature collection with 20 features and 4 fields
geometry type:  POINT
dimension:      XY
bbox:           xmin: 0.02845615 ymin: 0.02173949 xmax: 0.9897726 ymax: 0.9126277
epsg (SRID):    NA
proj4string:    NA
First 10 features:
           x          y id cluster                       geom
1  0.36593168 0.48489014  1       3  POINT (0.3659317 0.4848901)
2  0.39639897 0.81620996  2       3     POINT (0.396399 0.81621)
3  0.73541659 0.55053874  3       1  POINT (0.7354166 0.5505387)
4  0.73630607 0.83578792  4       1  POINT (0.7363061 0.8357879)
5  0.36669736 0.23638663  5       2  POINT (0.3666974 0.2363866)
6  0.61209669 0.90848268  6       1  POINT (0.6120967 0.9084827)
7  0.21571042 0.73283567  7       3  POINT (0.2157104 0.7328357)
8  0.06146981 0.78726856  8       3 POINT (0.06146981 0.7872686)
9  0.47413536 0.78620492  9       3  POINT (0.4741354 0.7862049)
10 0.18604814 0.02173949 10       2 POINT (0.1860481 0.02173949)
```

Geometry list column
Simple Feature Column (sfc)

Feature Geometry (sfg)

Feature

</> CLUSTERS

```
> xy %>% st_as_sf()
Simple feature collection with 20 features and 4 fields
geometry type:  POINT
dimension:      XY
bbox:           xmin: 0.02845615 ymin: 0.02173949 xmax: 0.9897726 ymax: 0.9126277
epsg (SRID):    NA
proj4string:    NA
First 10 features:
           x          y id cluster                        geom
1  0.36593168 0.48489014  1       3  POINT (0.3659317 0.4848901)
2  0.39639897 0.81620996  2       3     POINT (0.396399 0.81621)
3  0.73541659 0.55053874  3       1  POINT (0.7354166 0.5505387)
4  0.73630607 0.83578792  4       1  POINT (0.7363061 0.8357879)
5  0.36669736 0.23638663  5       2  POINT (0.3666974 0.2363866)
6  0.61209669 0.90848268  6       1  POINT (0.6120967 0.9084827)
7  0.21571042 0.73283567  7       3  POINT (0.2157104 0.7328357)
8  0.06146981 0.78726856  8       3 POINT (0.06146981 0.7872686)
9  0.47413536 0.78620492  9       3  POINT (0.4741354 0.7862049)
10 0.18604814 0.02173949 10       2 POINT (0.1860481 0.02173949)
```

Geometry list column
Simple Feature Column (sfc)

Feature Geometry (sfg)

Feature

</> TRANSFORMATION

x	y	id	cluster	geom
0.76960851	0.07850595	1	1	POINT (0.7696085 0.07850595)
0.42788664	0.36055541	2	1	POINT (0.4278866 0.3605554)
0.66108101	0.06721698	3	1	POINT (0.661081 0.06721698)
0.54024998	0.20531898	4	1	POINT (0.54025 0.205319)
0.07436085	0.01957189	5	1	POINT (0.07436085 0.01957189)
0.66699819	0.23012957	9	1	POINT (0.6669982 0.2301296)
0.48819917	0.06875559	11	1	POINT (0.4881992 0.06875559)
0.34558831	0.37947484	12	1	POINT (0.3455883 0.3794748)
0.34161436	0.04351447	14	1	POINT (0.3416144 0.04351447)
0.65789121	0.34235251	18	1	POINT (0.6578912 0.3423525)
0.23500317	0.49740325	8	2	POINT (0.2350032 0.4974032)
0.14670782	0.85468580	10	2	POINT (0.1467078 0.8546858)
0.05949405	0.93424017	15	2	POINT (0.05949405 0.9342402)
0.27574650	0.55669247	19	2	POINT (0.2757465 0.5566925)
0.51016032	0.88115437	20	2	POINT (0.5101603 0.8811544)
0.72182849	0.53802197	6	3	POINT (0.7218285 0.538022)
0.97875395	0.88164621	7	3	POINT (0.9787539 0.8816462)
0.57229636	0.61702036	13	3	POINT (0.5722964 0.6170204)
0.76161126	0.35183360	16	3	POINT (0.7616113 0.3518336)
0.90815230	0.59357843	17	3	POINT (0.9081523 0.5935784)

cluster	mx	my
<chr>	<dbl>	<dbl>
1	0.497	0.180
2	0.245	0.745
3	0.789	0.596

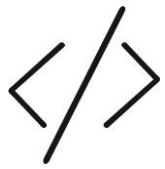
?

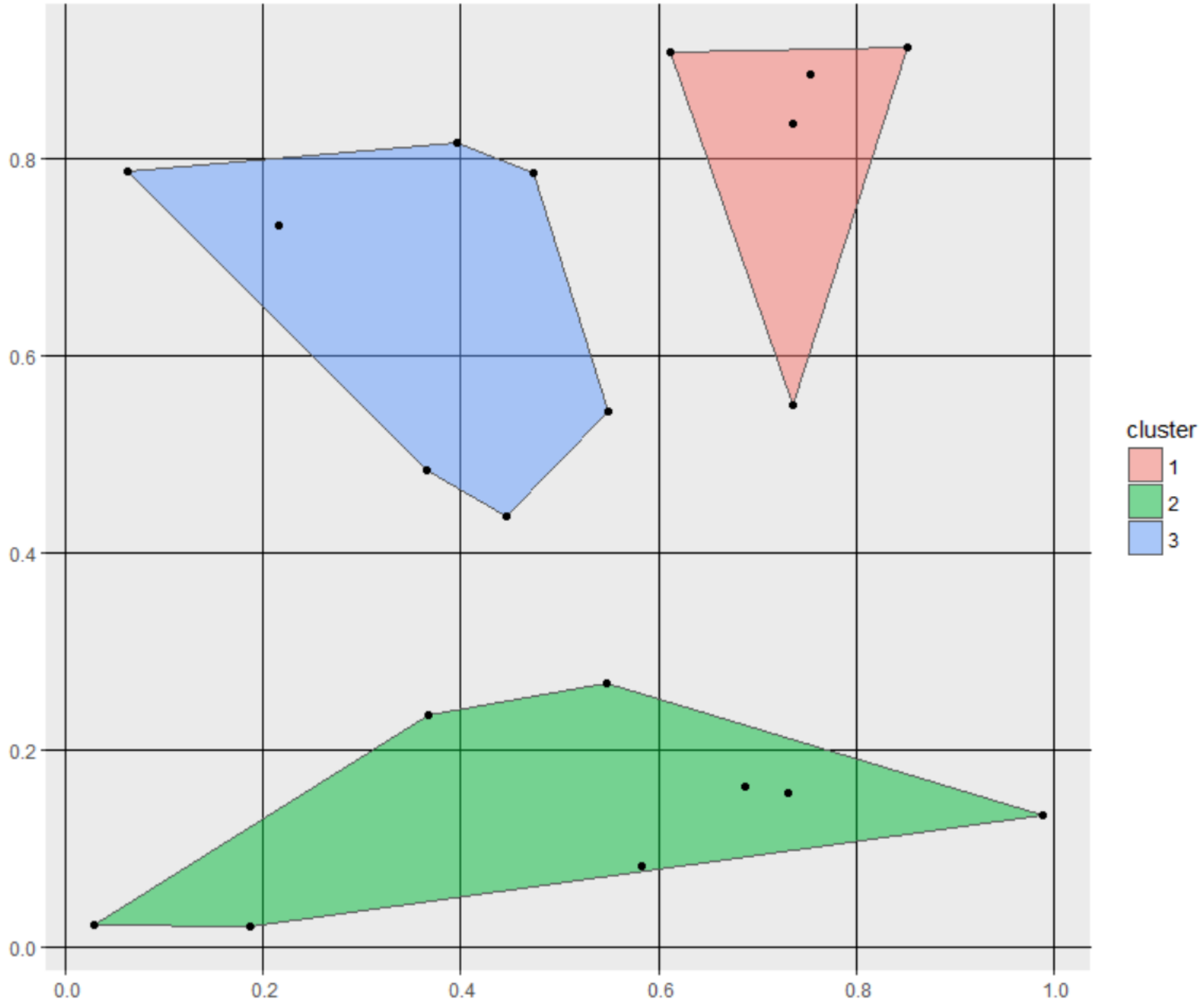
</> CLUSTERS

```
> xg<- xy %>% group_by(cluster) %>% summarise(geometry = st_union(geom))
```

```
> xg
# A tibble: 3 x 2
  cluster                       geometry
  <chr>                   <sf_geometry>
1 1       MULTIPOINT (0.6120967 0.908...
2 2       MULTIPOINT (0.02845615 0.02...
3 3       MULTIPOINT (0.06146981 0.78...
```

```
> ggplot() + geom_sf(data = xg %>% mutate(geometry = st_convex_hull(geometry)),
+                    aes(fill = cluster), alpha = 0.5) +
+            geom_sf(data = xg)
```

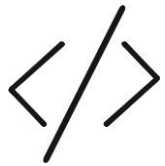



0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
cluster
1
2
3

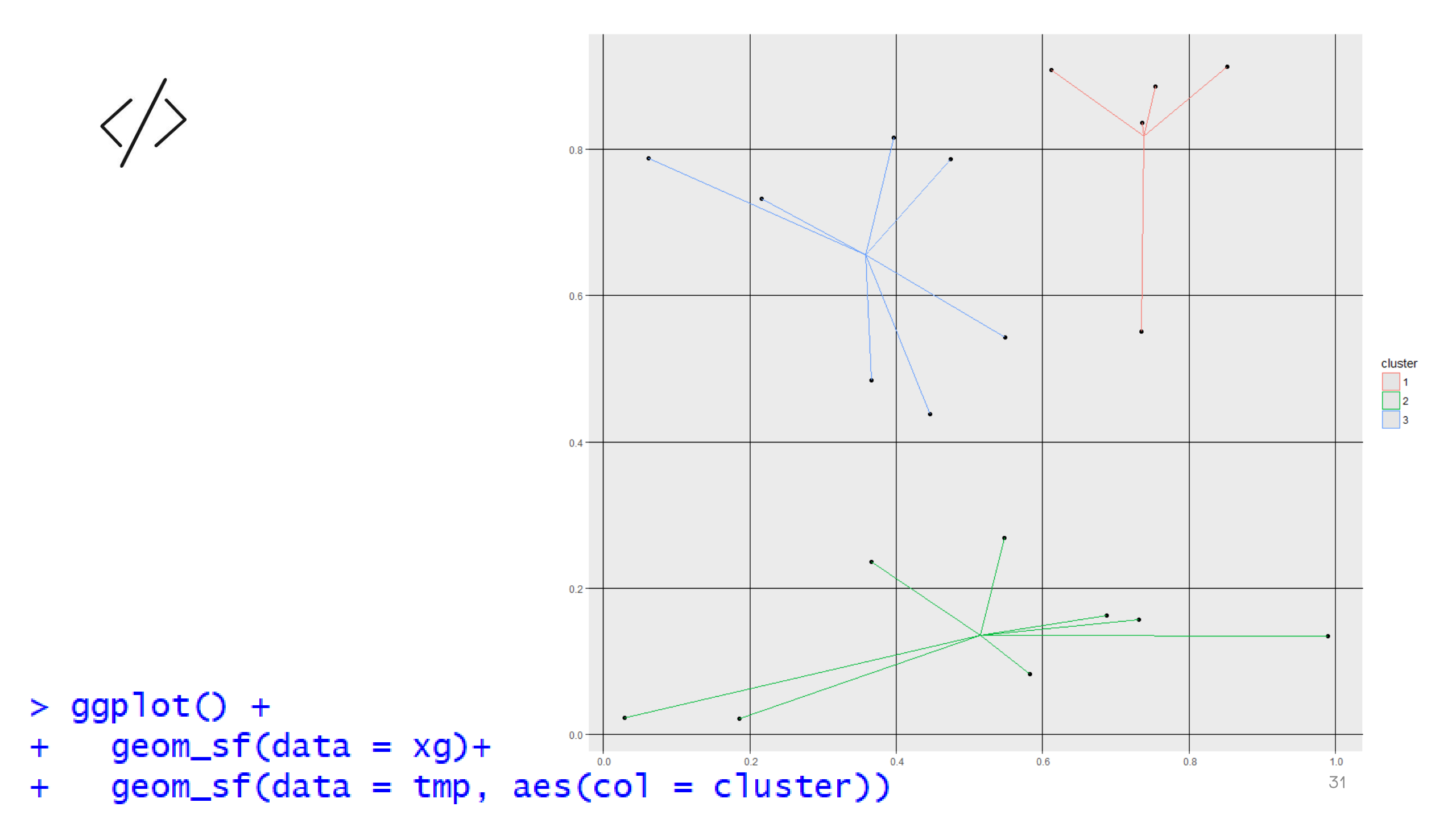
</> CLUSTERS

```
> centroids<- xy %>% group_by(cluster) %>% summarise(mx = mean(x), my = mean(y))
> centroids
# A tibble: 3 x 3
  cluster    mx    my
  <chr>   <dbl> <dbl>
1 1       0.738 0.819
2 2       0.515 0.136
3 3       0.358 0.656
```

```
> tmp<- xy %>%
+       left_join(centroids) %>%
+       mutate(geom2 = st_geometry(st_as_sf(cbind(mx, my) %>% as.data.frame(), coords = c("mx", "my")))) %>%
+       group_by(id) %>%
+       mutate(geometry = st_cast(st_union(geom, geom2), to = 'LINESTRING') ) %>%
+       select(-c(mx,my,geom,geom2))
```



```
> tmp %>% head()
# A tibble: 6 x 5
# Groups:   id [6]
      x     y    id cluster                        geometry
  <dbl> <dbl> <int> <chr>                    <sf_geometry>
1 0.366 0.485     1 3       LINESTRING (0.3659317 0.484...
2 0.396 0.816     2 3       LINESTRING (0.396399 0.8162...
3 0.735 0.551     3 1       LINESTRING (0.7354166 0.550...
4 0.736 0.836     4 1       LINESTRING (0.7363061 0.835...
5 0.367 0.236     5 2       LINESTRING (0.3666974 0.236...
6 0.612 0.908     6 1       LINESTRING (0.6120967 0.908...
```

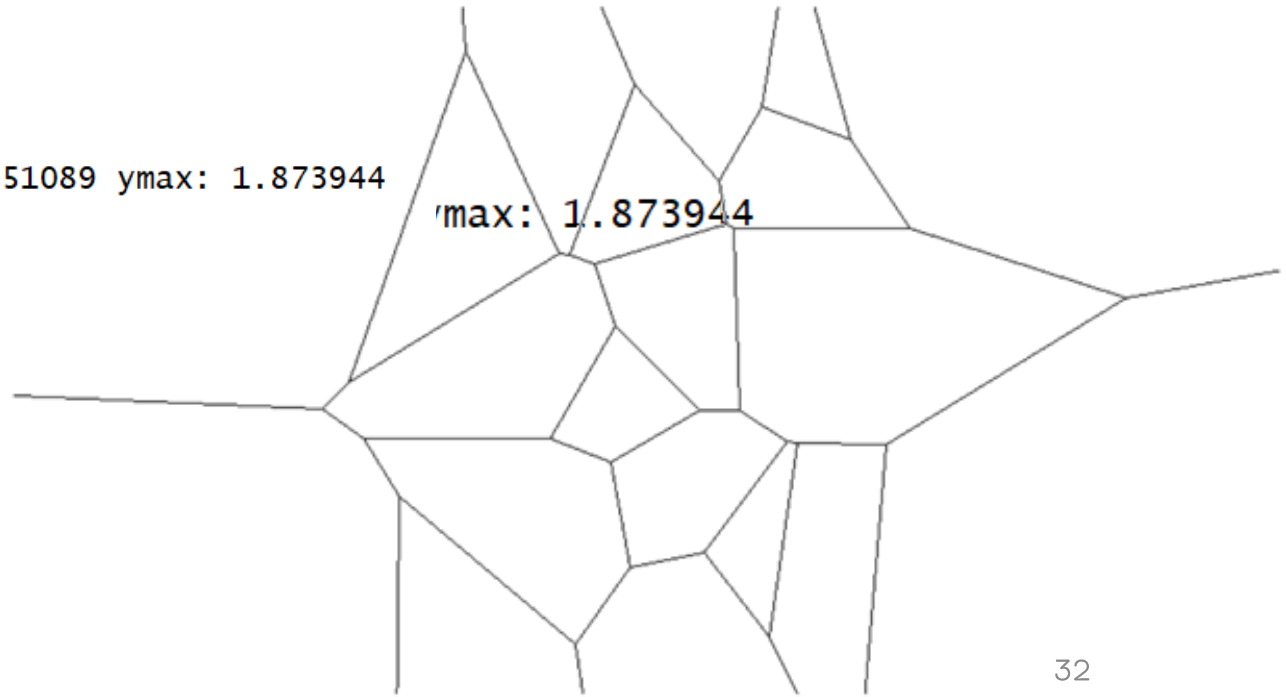



```
> ggplot() +
+    geom_sf(data = xg)+
+    geom_sf(data = tmp, aes(col = cluster))
```

VORONOI TESSELATIONS

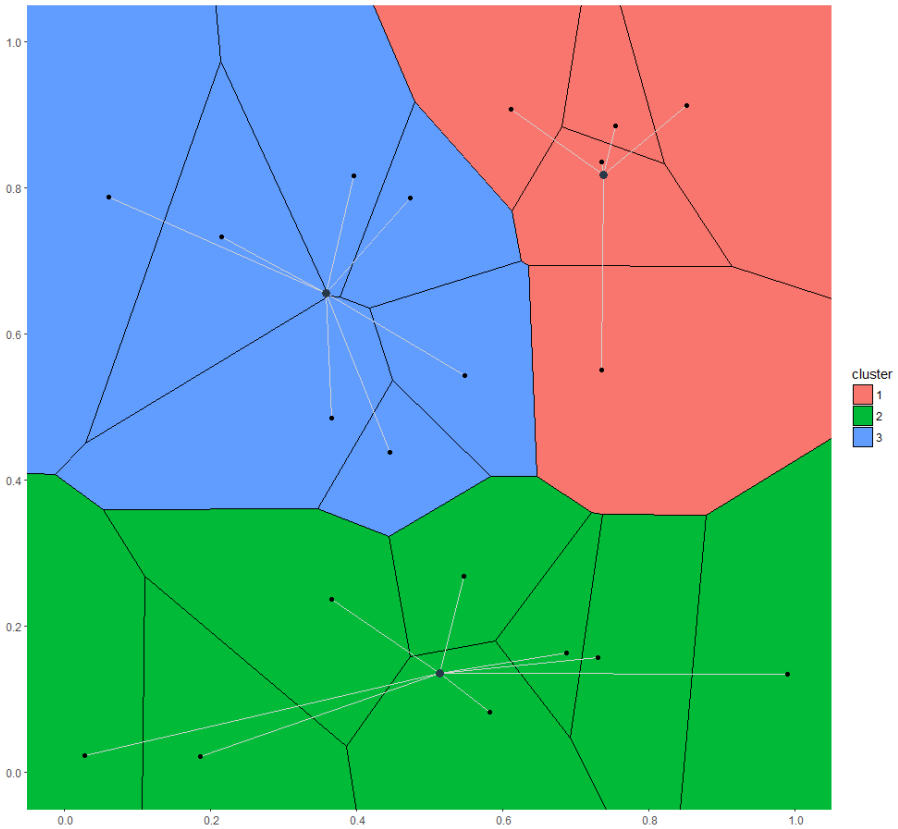
```
> v<- st_union(xy$geom) %>% st_voronoi() %>% st_cast() %>%  data.frame() %>%  st_as_sf()
```

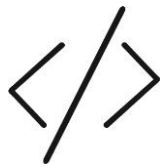
```
Simple feature collection with 20 features and 0 fields
geometry type:  POLYGON
dimension:      XY
bbox:           xmin: -0.9328603 ymin: -0.9395769 xmax: 1.951089 ymax: 1.873944
epsg (SRID):    NA
proj4string:    NA
First 10 features:
                       geometry
1  POLYGON ((-0.9328603 0.4476...
2  POLYGON ((0.09617865 -0.939...
3  POLYGON ((0.5362098 -0.9395...
4  POLYGON ((0.0530392 0.35967...
5  POLYGON ((0.2140581 0.97389...
6  POLYGON ((0.02932164 0.4504...
7  POLYGON ((0.3473502 0.36058...
8  POLYGON ((0.1100874 0.26901...
9  POLYGON ((0.9198492 -0.9395...
10 POLYGON ((0.1109062 1.87394...
```

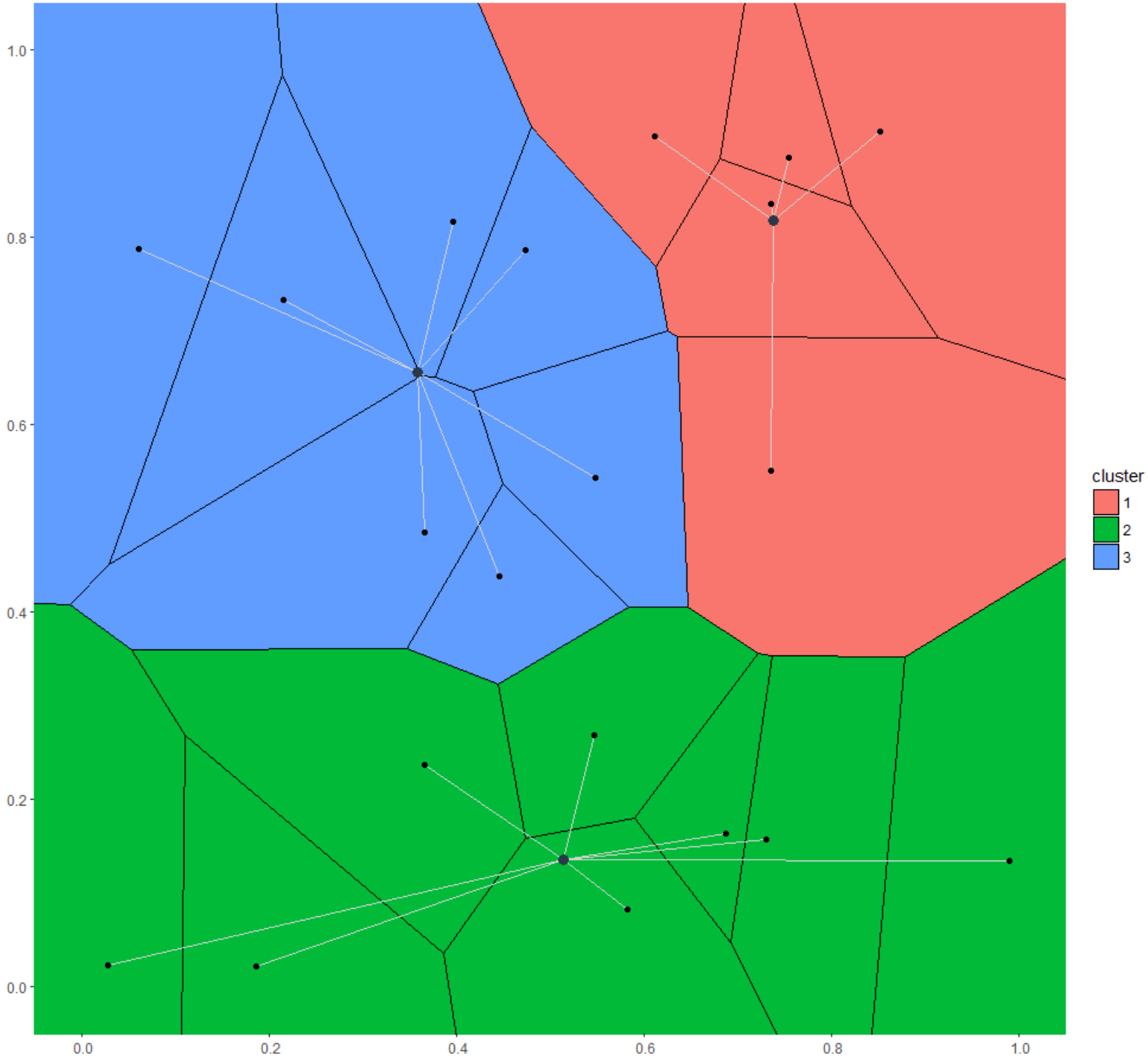


CLUSTERS + VORONOI

```
> ggplot() + geom_sf(data = xy, aes(geometry = voronoi), fill = cluster), col = 'black') +
+            geom_sf(data = tmp, col = "light gray", alpha = 0.5)+
+            geom_sf(data = centroids %>% st_as_sf(coords = c("mx", "my")), size =3, col = '#283848'
+            geom_sf(data = xy, aes(geometry = geom)) +
```





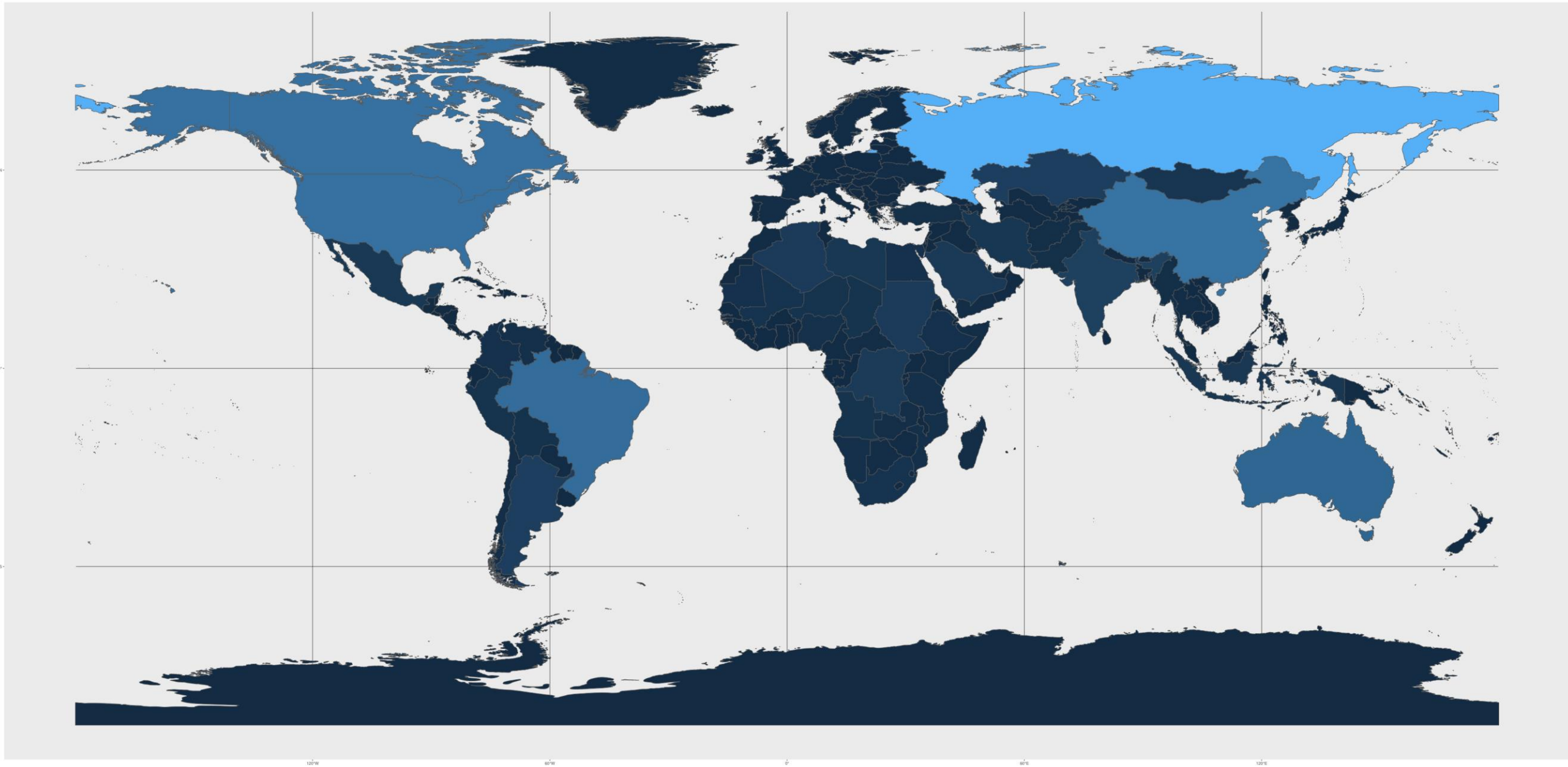

1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
cluster
1
2
3

</> SHAPE FILES (WKB)

```
> the_world<- st_read('~/TM_WORLD_BORDERS-0.3.shp')
```

```
> the_world %>% head
Simple feature collection with 6 features and 11 fields
geometry type:  MULTIPOLYGON
dimension:      XY
bbox:           xmin: -61.89111 ymin: -18.01639 xmax: 50.37499 ymax: 42.66194
epsg (SRID):    4326
proj4string:    +proj=longlat +datum=WGS84 +no_defs
  FIPS ISO2 ISO3 UN                 NAME   AREA  POP2005 REGION SUBREGION      LON      LAT                       geometry
1   AC   AG  ATG 28 Antigua and Barbuda     44    83039     19         29 -61.783   17.078 MULTIPOLYGON (((-61.68667 1...
2   AG   DZ  DZA 12              Algeria 238174 32854159      2         15   2.632   28.163 MULTIPOLYGON (((2.96361 36....
3   AJ   AZ  AZE 31           Azerbaijan   8260  8352021    142        145  47.395   40.430 MULTIPOLYGON (((45.08332 39...
4   AL   AL  ALB  8              Albania   2740  3153731    150         39  20.068   41.143 MULTIPOLYGON (((19.43621 41...
5   AM   AM  ARM 51              Armenia   2820  3017661    142        145  44.563   40.534 MULTIPOLYGON (((45.57305 40...
6   AO   AO  AGO 24               Angola 124670 16095214      2         17  17.544  -12.296 MULTIPOLYGON (((11.75083 -1...
```

```
> ggplot(the_world) + geom_sf(aes(fill = AREA))
```


50°N
0°
50°S
120°W
60°W
0°
60°E
120°E

</> PROJECTIONS

```
> world2 <- sf::st_transform(the_world, "+proj=laea +y_0=0 +lon_0=25 +lat_0=-28 +ellps=WGS84 +no_defs")
```



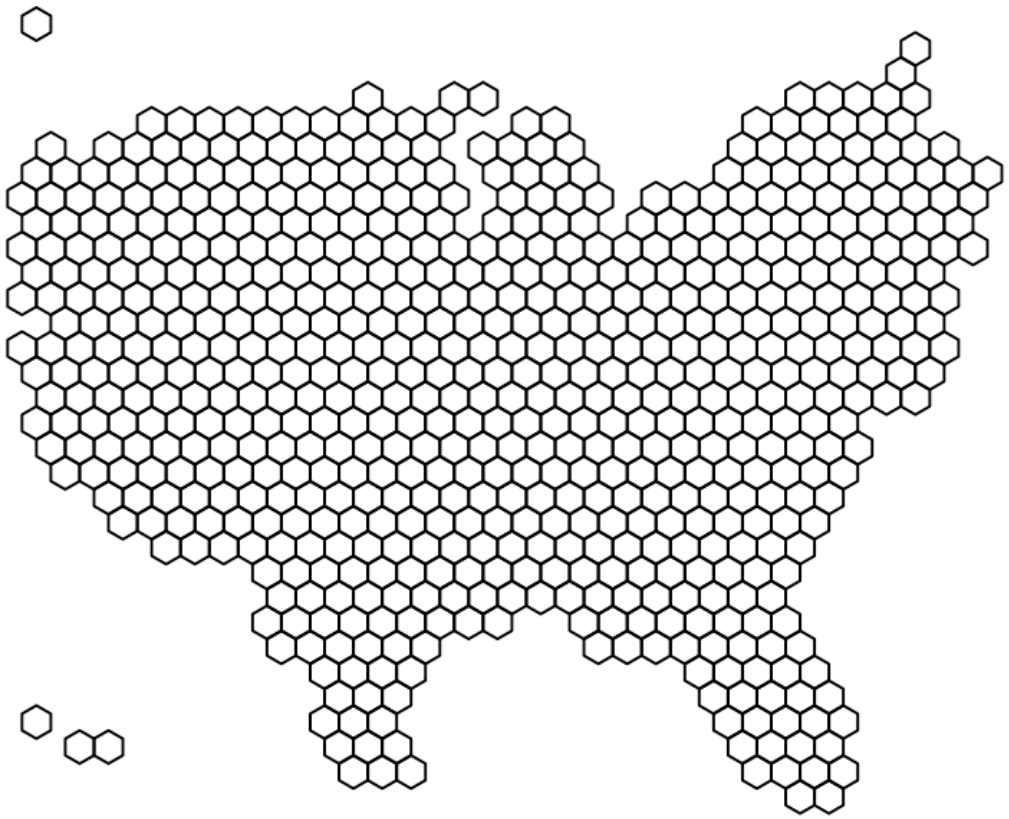
</> sf AND LEAFLET

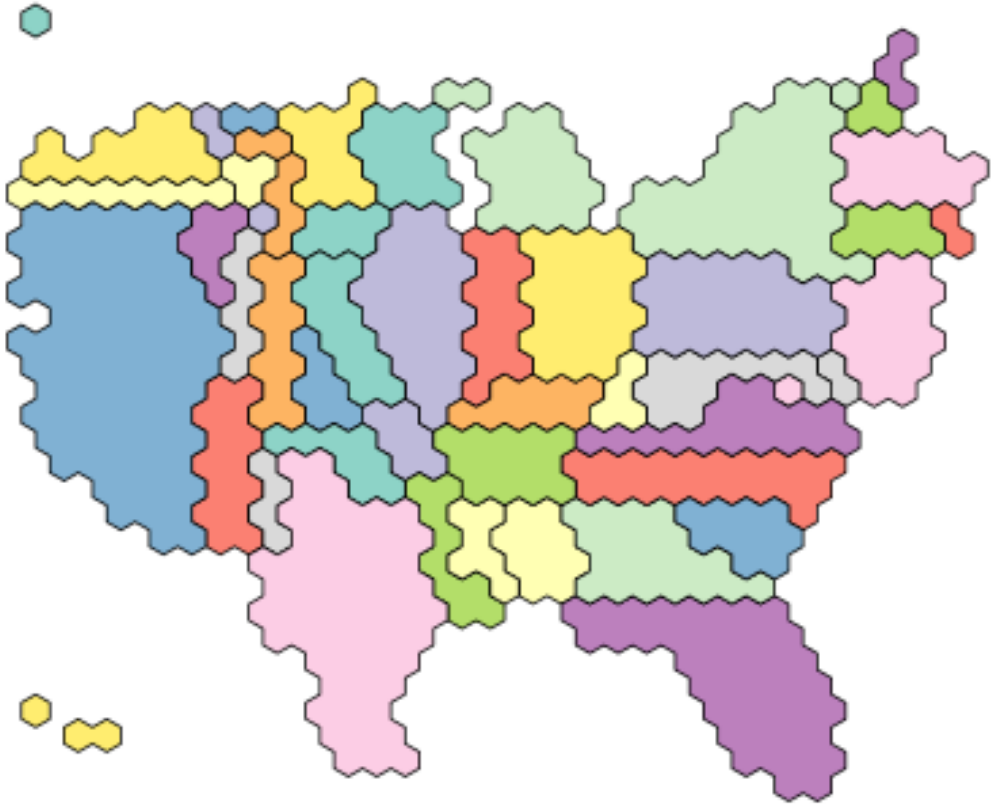
```
> hlopt<- highlightOptions(weight = 5,
+                          color = "#666",
+                          dashArray = "",
+                          fillOpacity = 0.7,
+                          bringToFront = TRUE)
> m <- leaflet() %>% addTiles() %>% addPolygons(data = the_world,
+                                               label = ~NAME,
+                                               highlight = hlopt)
```

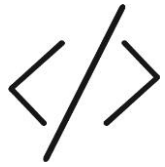

Ottawa
New York
Paris
France
España
Barcelona
Česko
Magyarország
România
București
Київ
Україна
Қазақстан
Türkiye
İstanbul
İzmir
Türkmenistan
Aşgabat
Бишкек
Кыргызстан
Toshkent
O'zbekiston
Delhi
Jaipur
Lucknow
Ahmedabad
India
Mumbai
Hyderabad
Bengaluru
Alger
Rabat
Mali
Niger
Tchad
Sénégal
Bamako
Niamey
Burkina Faso
Nigeria
Kano
Guinée
Freetown
Monrovia
Abidjan
Accra
Lagos
Cameroun
Yaoundé
Gabon
South Sudan
Somaaliya
Muqdisho
Ugand
Kenya
Nairobi
Tanzania
Dar es Salaam
République démocratique du Congo
Kinshasa
Mbuji-Mayi
Luanda
Angola
Zambia
Lusaka
Zimbabwe
Namibia
Botswana
Pretoria
Maputo
Madagasikara
Caracas
Venezuela
Medellín
Colombia
Perú
Lima
Brasil
Brasília
Bolivia
Paraguay
Asunción
Rio de Janeiro
São Paulo
Leaflet | © OpenStreetMap contributors, CC-BY-SA

</> A FINAL EXAMPLE

- https://github.com/r-spatial/sf/issues/275







The measure of an R package

If it's good enough for Hadley.
It's good enough for you.

Questions?