10 — Maps and Spatial Data II

Kieran Healy

March 22, 2024

Maps and Spatial Data (2)

Load our libraries

```r
library(here)       # manage file paths
library(socviz)     # data and some useful functions
library(tidyverse)  # your friend and mine
library(tidycensus) # Tidily interact with the US Census
library(maps)       # Some basic maps
```

```
Attaching package: 'maps'

The following object is masked from 'package:purrr':

    map
```

```r
library(sf)         # Make maps in ggplot
```

```
Linking to GEOS 3.11.0, GDAL 3.5.3, PROJ 9.1.0; sf_use_s2() is TRUE
```

```r
library(tigris)     # Talk to the Census's TIGER data
```

```
To enable caching of data, set `options(tigris_use_cache = TRUE)`
in your R script or .Rprofile.
```

```r
library(ggforce)    # Useful enhancements to ggplot
```

Proper Maps with Simple Features

`geom_polygon()` is limiting

It's very useful to have the intuition that, when drawing maps, we're just working with tables of `x` and `y` coordinates, and shapes represent quantities in our data, in a way that's essentially the same as any other geom. This makes it worth getting comfortable with what `geom_polygon()` and `coord_map()` are doing. But the business of having very large map tables and manually specifying projections is inefficient.

In addition, sometimes our data *really is* properly spatial, at which point we need a more rigorous and consistent way of specifying those elements. There's a whole world of Geodesic standards and methods devoted to specifying these things for GIS applications. R is not a dedicated GIS, but we can take advantage of these tools.

Enter simple features, the `sf` package, and `geom_sf()`

The Simple Features package

When we load `sf` it creates a way to use several standard GIS concepts and tools, such as the GEOS library for computational geometry, the PROJ software that transforms spatial coordinates from one reference system to another, as in map projections, and the Simple Features standard for specifying the elements of spatial attributes.

```
library(sf)
```

```
Linking to GEOS 3.11.0, GDAL 3.5.3, PROJ 9.1.0; sf_use_s2() is TRUE
```

Let's see the main upshot for us.

What’s a Feature?

A feature is a thing or object in the real world: a building, a tree, a field, a county.

Like real objects, features are often made of other objects.

“A set of features can form a single feature: A forest stand can be a feature, a forest can be a feature, a city can be a feature. A satellite image pixel can be a feature, a complete image can be a feature too.”

Features have a *geometry* describing *where* on Earth the feature is located, and they have attributes, which describe other properties.

Features have Dimensions

All geometries are composed of points. Points are coordinates in a 2-, 3- or 4-dimensional space. All points in a geometry have the same dimensionality. In addition to X and Y coordinates, there are two optional additional dimensions:

- a Z coordinate, denoting altitude
- an M coordinate (rarely used), denoting some *measure* that is associated with the point, rather than with the feature as a whole (in which case it would be a feature attribute); examples could be time of measurement, or measurement error of the coordinates

Features have Dimensions

T. Two-dimensional points refer to x and y, easting and northing, or longitude and latitude, we refer to them as `XY` 2. Three-dimensional points as `XYZ` 3. Three-dimensional points as `XYM` 4. Four-dimensional points as `XYZM` (the third axis is Z, fourth M)

The most common kinds

type	description
`POINT`	zero-dimensional geometry containing a single point
`LINESTRING`	sequence of points connected by straight, non-self intersecting line pieces; one-dimensional geometry
`POLYGON`	geometry with a positive area (two-dimensional); sequence of points form a closed, non-self intersecting ring; the first ring denotes the exterior ring, zero or more subsequent rings denote holes in this exterior ring
`MULTIPOINT`	set of points; a MULTIPOINT is simple if no

What they look like

Coordinate reference system

Coordinates can only be placed on the Earth's surface when their coordinate reference system (CRS) is known; this may be a spheroid CRS such as WGS84, a projected, two-dimensional (Cartesian) CRS such as a UTM zone or Web Mercator, or a CRS in three-dimensions, or including time.

Example: North Carolina

Conveniently, the example in the SF package is our beloved state.

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

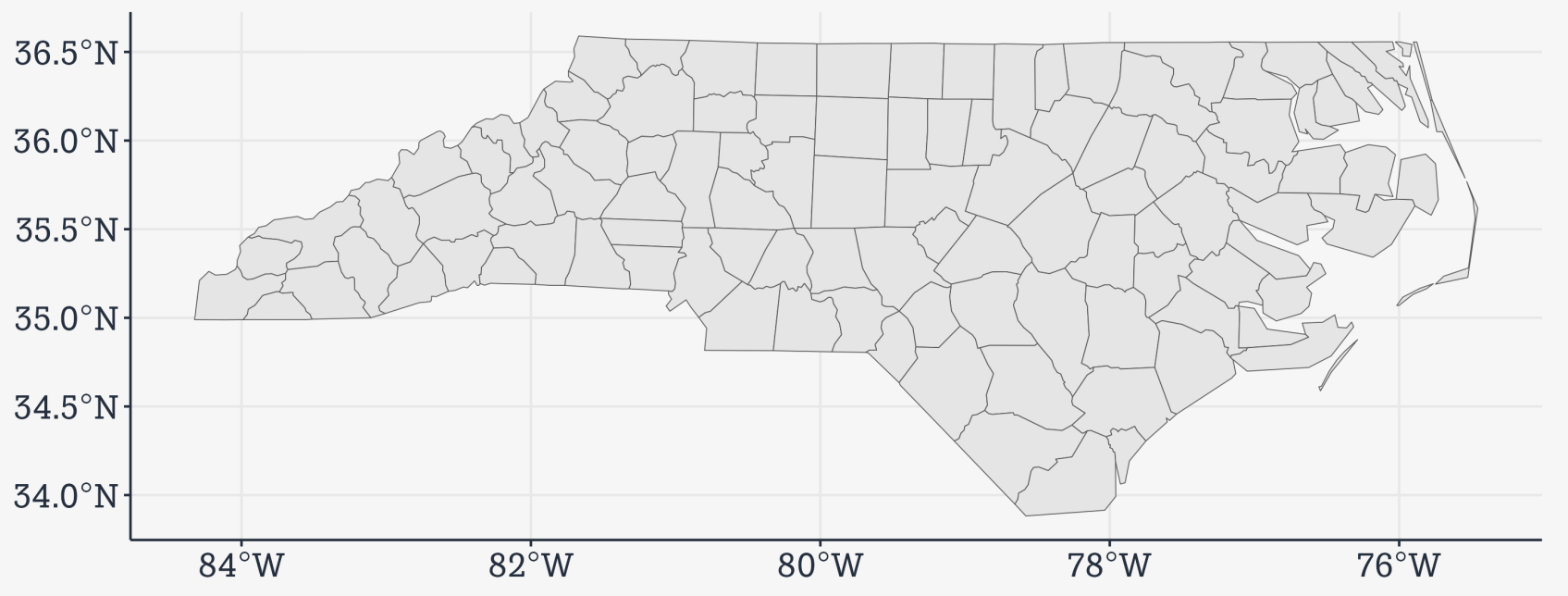
```
Reading layer `nc' from data source
  `/Users/kjhealy/Library/Caches/org.R-project.R/R/renv/cache/v5/R-4.3/aarch64-apple-darwin20/sf/1.0-
15/f432b3379fb1a47046e253468b6b6b6d/sf/shape/nc.shp'
  using driver `ESRI Shapefile'
Simple feature collection with 100 features and 14 fields
Geometry type: MULTIPOLYGON
Dimension:     XY
Bounding box:  xmin: -84.32385 ymin: 33.88199 xmax: -75.45698 ymax: 36.58965
Geodetic CRS:  NAD27
```

```
nc
```

```
Simple feature collection with 100 features and 14 fields
Geometry type: MULTIPOLYGON
Dimension:     XY
Bounding box:  xmin: -84.32385 ymin: 33.88199 xmax: -75.45698 ymax: 36.58965
Geodetic CRS:  NAD27
First 10 features:
    AREA PERIMETER CNTY_ CNTY_ID        NAME  FIPS FIPSNO CRESS_ID BIR74 SID74
1  0.114     1.442  1825    1825        Ashe 37009  37009        5  1091     1
2  0.061     1.231  1827    1827   Alleghany 37005  37005        3   487     0
3  0.143     1.630  1828    1828       Surry 37171  37171       86  3188     5
4  0.070     2.968  1831    1831   Currituck 37053  37053       27   508     1
5  0.153     2.206  1832    1832 Northampton 37131  37131       66  1421     9
6  0.097     1.670  1833    1833    Hertford 37091  37091       46  1452     7
7  0.062     1.547  1834    1834      Camden 37029  37029       15   286     0
8  0.091     1.284  1835    1835       Gates 37073  37073       37   420     0
9  0.118     1.421  1836    1836      Warren 37185  37185       93   968     4
10 0.124     1.428  1837    1837      Stokes 37169  37169       85  1612     1
   NWBIR74 BIR79 SID79 NWBIR79                       geometry
1       10  1364     0      19 MULTIPOLYGON (((-81.47276 3 ...
2       10   542     3      12 MULTIPOLYGON (((-81.23989 3 ...
```

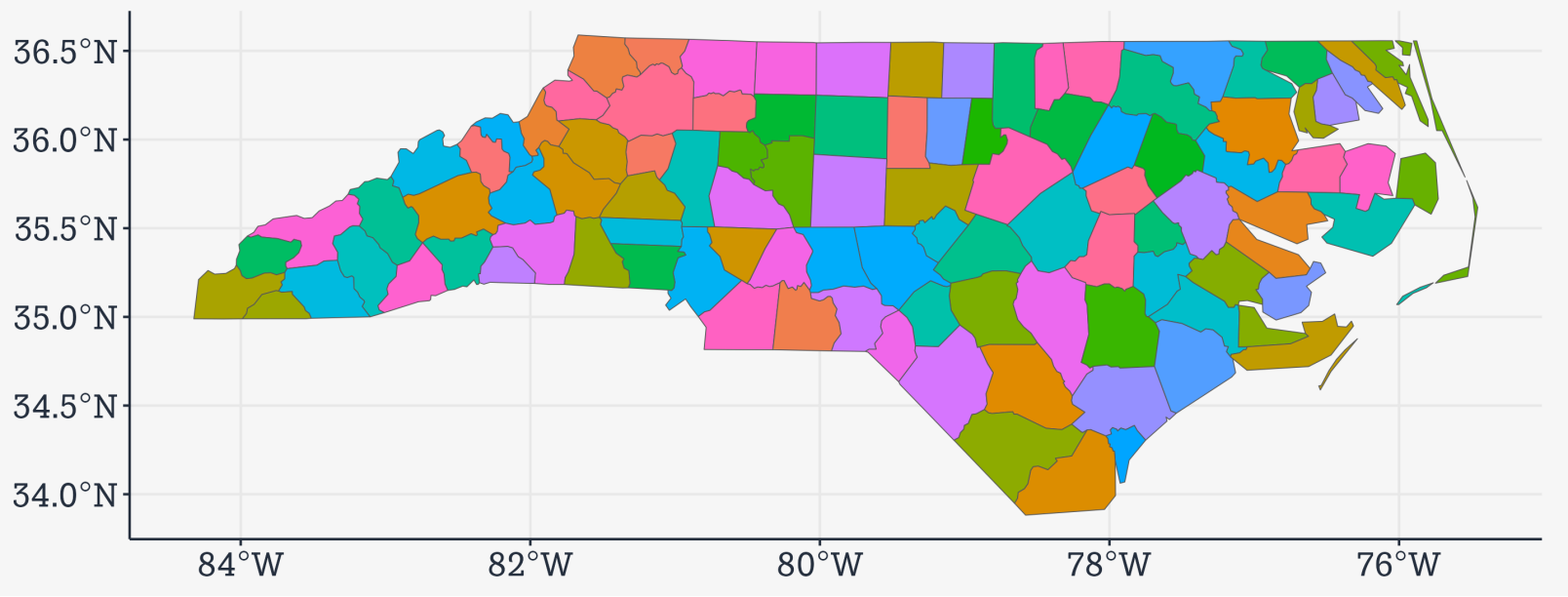
Example: North Carolina

```
nc ▷
  ggplot() +
  geom_sf()
```



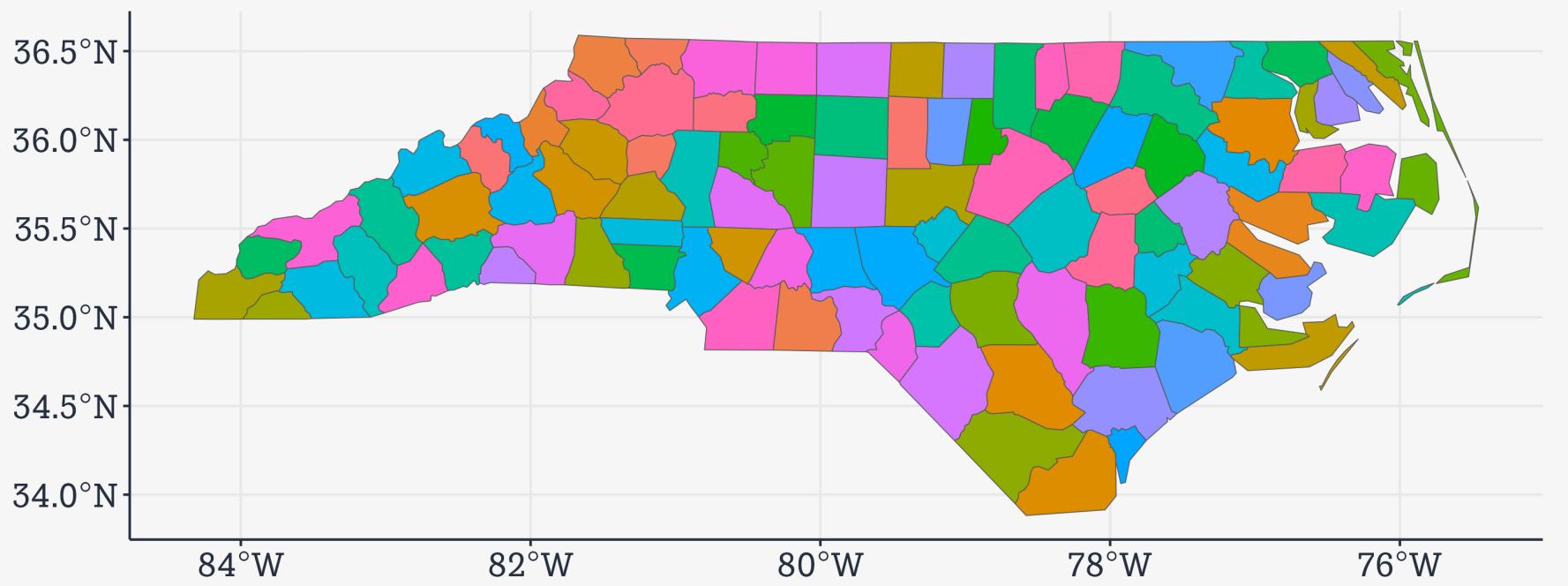
Example: North Carolina

```
nc ▷
  ggplot() +
  geom_sf(mapping = aes(fill = NAME)) +
  guides(fill = "none")
```



Example: North Carolina

```
nc ▷
  ggplot() +
  geom_sf(mapping = aes(fill = NAME)) +
  guides(fill = "none")
```



What Simple Features buy us

We can perform spatial operations much, much more easily. They're just like any other calculationg or grouping action.

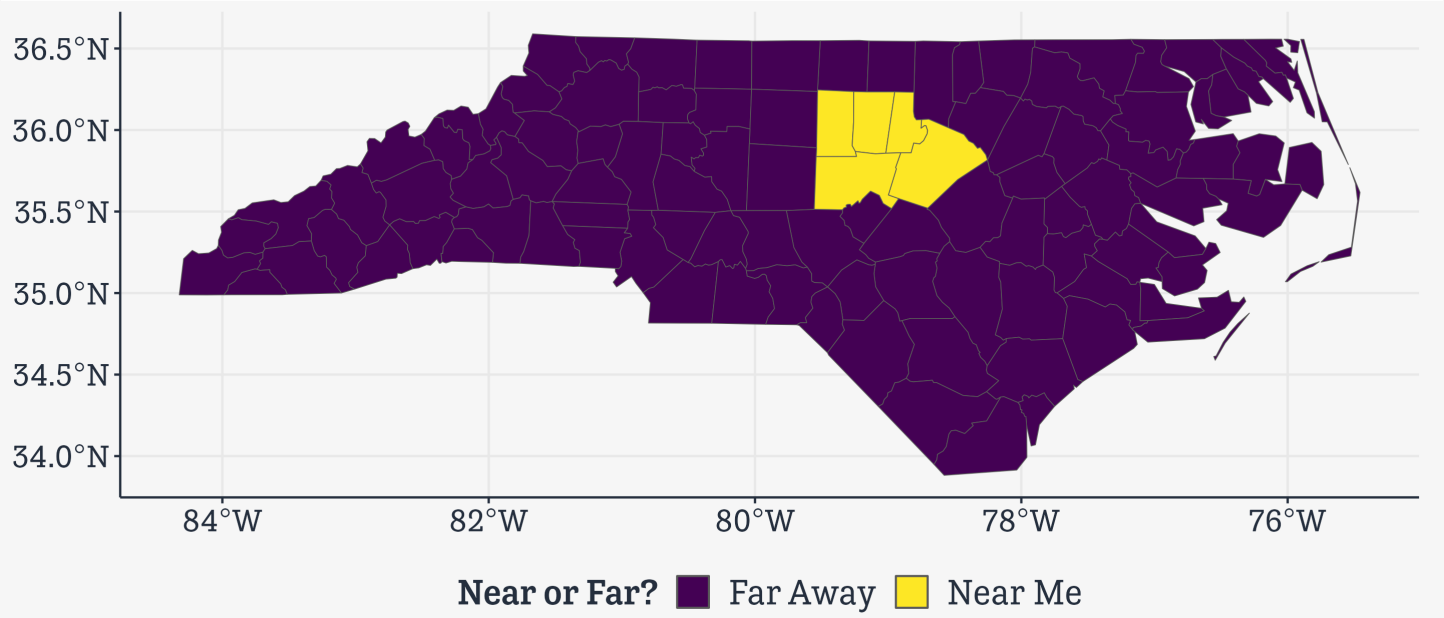
What Simple Features buy us

```r
## Make a variable picking out five counties near where I live
nc <- nc |>
  mutate(near_me = case_when(NAME %in% c("Orange", "Durham",
                                         "Wake", "Chatham",
                                         "Alamance") ~ "Near Me",
                             TRUE ~ "Far Away"))

## What we just did
nc |>
  count(near_me)
```

```
Simple feature collection with 2 features and 2 fields
Geometry type: GEOMETRY
Dimension:     XY
Bounding box:  xmin: -84.32385 ymin: 33.88199 xmax: -75.45698 ymax: 36.58965
Geodetic CRS:  NAD27
   near_me  n                       geometry
1 Far Away 95 MULTIPOLYGON (((-76.46926 3 ...
2  Near Me  5 POLYGON ((-79.54099 35.8369 ...
```

What Simple Features buy us

```
nc ▷
  ggplot() +
  geom_sf(mapping = aes(fill = near_me)) +
  scale_fill_viridis_d() +
  theme(legend.position = "bottom") +
  labs(fill = "Near or Far?")
```



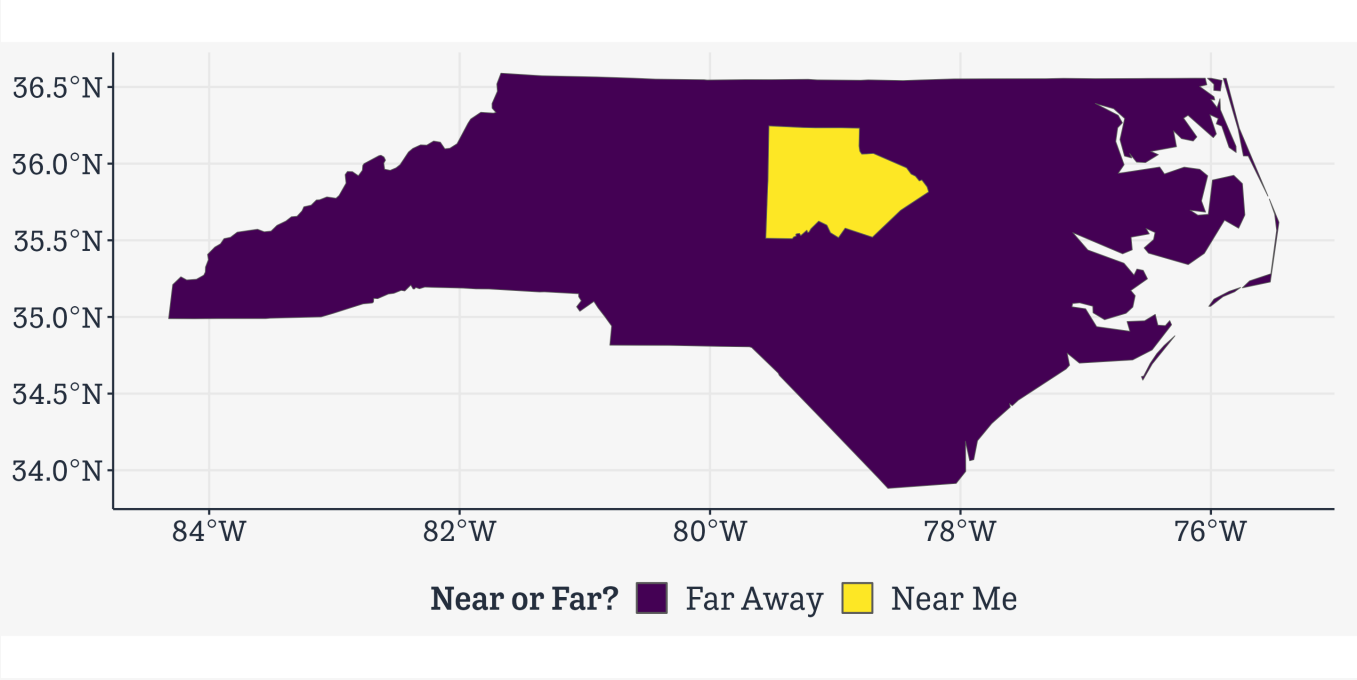
What Simple Features buy us

```r
## These are all still county polygons. But now ...
nc_merged <- nc %>%
  group_by(near_me) %>%
  summarize(mean_b = mean(BIR74),
            sum_sid = sum(SID74))

nc_merged
```

```
Simple feature collection with 2 features and 3 fields
Geometry type: GEOMETRY
Dimension:     XY
Bounding box:  xmin: -84.32385 ymin: 33.88199 xmax: -75.45698 ymax: 36.58965
Geodetic CRS:  NAD27
# A tibble: 2 × 4
  near_me  mean_b sum_sid                                                geometry
  <chr>     <dbl>   <dbl>                                            <GEOMETRY [°]>
1 Far Away  3137.     616 MULTIPOLYGON (((-76.46926 34.69328, -76.2877 34.87701…
2 Near Me   6387.      51 POLYGON ((-79.54099 35.83699, -79.55536 35.51305, -79…
```

What Simple Features buy us

```
## Now we only have two polygons
nc_merged ▷
  ggplot() +
  geom_sf(mapping = aes(fill = near_me)) +
  scale_fill_viridis_d() +
  theme(legend.position = "bottom") +
  labs(fill = "Near or Far?")
```



More sf goodies

Tree data

```
nytrees_example
```

```
Simple feature collection with 683683 features and 5 fields
Geometry type: POINT
Dimension:     XY
Bounding box:  xmin: 913349.3 ymin: 120973.8 xmax: 1067248 ymax: 271894.1
Projected CRS: NAD83 / New York Long Island (ftUS)
# A tibble: 683,683 × 6
   nta_code nta_name     spc_common latitude longitude                 geometry
   <chr>    <chr>        <chr>      <chr>    <chr>     <POINT [US_survey_foot]>
 1 QN17     Forest Hills red maple  40.7230… -73.8442…       (1027431 202756.8)
 2 QN49     Whitestone   pin oak    40.7941… -73.8186…       (1034456 228644.8)
 3 BK90     East Willia… honeylocu… 40.7175… -73.9366…       (1001823 200716.9)
 4 BK90     East Willia… honeylocu… 40.7135… -73.9344…       (1002420 199244.3)
 5 BK37     Park Slope-… American … 40.6667… -73.9759…      (990913.8 182202.4)
 6 MN14     Lincoln Squ… honeylocu… 40.7700… -73.9849…      (988418.7 219825.5)
 7 MN14     Lincoln Squ… honeylocu… 40.7702… -73.9853…      (988311.2 219885.3)
 8 MN15     Clinton      American … 40.7627… -73.9872…      (987769.1 217157.9)
 9 SI14     Grasmere-Ar… honeylocu… 40.5965… -74.0762…      (963073.2 156635.6)
10 BK26     Gravesend    London pl… 40.5863… -73.9697…      (992653.7 152903.6)
# ℹ 683,673 more rows
```

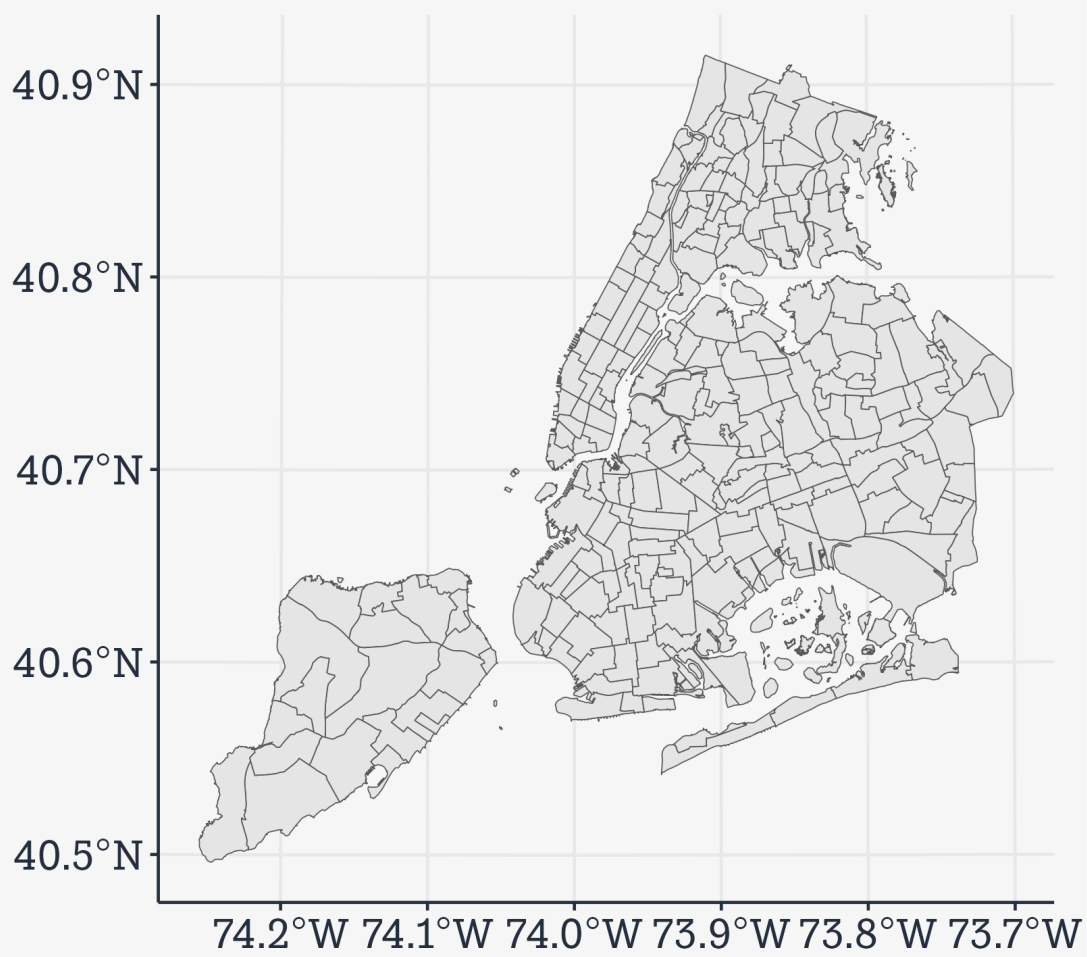
More sf goodies

Tree data

```
nta20_example
```

```
Simple feature collection with 262 features and 4 fields
Geometry type: MULTIPOLYGON
Dimension:     XY
Bounding box:  xmin: 913175.1 ymin: 120128.4 xmax: 1067383 ymax: 272844.3
Projected CRS: NAD83 / New York Long Island (ftUS)
# A tibble: 262 × 5
   nta_code20 nta_name20       cdta_code20 cdta_name20                  geometry
   <chr>      <chr>            <chr>       <chr>       <MULTIPOLYGON [US_survey>
 1 BK0101     Greenpoint       BK01        BK01 Willi… (((1003060 204572, 10029…
 2 BK0102     Williamsburg     BK01        BK01 Willi… (((994849 203499.3, 9949…
 3 BK0103     South Williamsb… BK01        BK01 Willi… (((998047.2 196303.3, 99…
 4 BK0104     East Williamsbu… BK01        BK01 Willi… (((1005302 199455.7, 100…
 5 BK0201     Brooklyn Heights BK02        BK02 Downt… (((986367.7 190549.2, 98…
 6 BK0202     Downtown Brookl… BK02        BK02 Downt… (((990056.4 196474.8, 99…
 7 BK0203     Fort Greene      BK02        BK02 Downt… (((994554.2 193593, 9945…
 8 BK0204     Clinton Hill     BK02        BK02 Downt… (((994971 187190, 994431…
 9 BK0261     Brooklyn Navy Y… BK02        BK02 Downt… (((990092 196467.1, 9900…
10 BK0301     Bedford-Stuyves… BK03        BK03 Bedfo… (((999743.6 186381.4, 99…
# ℹ 252 more rows
```

```r
nta20_example ▷
  ggplot() +
  geom_sf()
```



Problem

How to get the new NTA boundaries merged with the tree dataset, which doesn't have them?

```
nytrees_example
```

```
Simple feature collection with 683683 features and 5 fields
Geometry type: POINT
Dimension:     XY
Bounding box:  xmin: 913349.3 ymin: 120973.8 xmax: 1067248 ymax: 271894.1
Projected CRS: NAD83 / New York Long Island (ftUS)
# A tibble: 683,683 × 6
   nta_code nta_name     spc_common latitude longitude                 geometry
   <chr>    <chr>        <chr>      <chr>    <chr>     <POINT [US_survey_foot]>
 1 QN17     Forest Hills red maple  40.7230… -73.8442…       (1027431 202756.8)
 2 QN49     Whitestone   pin oak    40.7941… -73.8186…       (1034456 228644.8)
 3 BK90     East Willia… honeylocu… 40.7175… -73.9366…       (1001823 200716.9)
 4 BK90     East Willia… honeylocu… 40.7135… -73.9344…       (1002420 199244.3)
 5 BK37     Park Slope-… American … 40.6667… -73.9759…      (990913.8 182202.4)
 6 MN14     Lincoln Squ… honeylocu… 40.7700… -73.9849…      (988418.7 219825.5)
 7 MN14     Lincoln Squ… honeylocu… 40.7702… -73.9853…      (988311.2 219885.3)
 8 MN15     Clinton      American … 40.7627… -73.9872…      (987769.1 217157.9)
 9 SI14     Grasmere-Ar… honeylocu… 40.5965… -74.0762…      (963073.2 156635.6)
10 BK26     Gravesend    London pl… 40.5863… -73.9697…      (992653.7 152903.6)
# ℹ 683,673 more rows
```

Answer

```r
treepoints_sf <- nytrees_example |>
  select(nta_code, nta_name, latitude, longitude) |>
  st_as_sf(coords = c("longitude", "latitude"), crs = st_crs(nta20_example))

treepoints_sf
```

```
Simple feature collection with 683683 features and 4 fields
Geometry type: POINT
Dimension:     XY
Bounding box:  xmin: 913349.3 ymin: 120973.8 xmax: 1067248 ymax: 271894.1
Projected CRS: NAD83 / New York Long Island (ftUS)
# A tibble: 683,683 × 5
   nta_code nta_name                latitude longitude                 geometry
   <chr>    <chr>                   <chr>    <chr>     <POINT [US_survey_foot]>
 1 QN17     Forest Hills            40.7230… -73.8442…       (1027431 202756.8)
 2 QN49     Whitestone              40.7941… -73.8186…       (1034456 228644.8)
 3 BK90     East Williamsburg       40.7175… -73.9366…       (1001823 200716.9)
 4 BK90     East Williamsburg       40.7135… -73.9344…       (1002420 199244.3)
 5 BK37     Park Slope-Gowanus      40.6667… -73.9759…      (990913.8 182202.4)
 6 MN14     Lincoln Square          40.7700… -73.9849…      (988418.7 219825.5)
 7 MN14     Lincoln Square          40.7702… -73.9853…      (988311.2 219885.3)
 8 MN15     Clinton                 40.7627… -73.9872…      (987769.1 217157.9)
 9 SI14     Grasmere-Arrochar-Ft. … 40.5965… -74.0762…      (963073.2 156635.6)
10 BK26     Gravesend               40.5863… -73.9697…      (992653.7 152903.6)
# ℹ 683,673 more rows
```

Answer

```r
treepoints2020 <- treepoints_sf |>
  mutate(
    intersection = as.integer(st_intersects(geometry, nta20_example)),
    nta_code20 = if_else(is.na(intersection), '', nta20_example$nta_code20[intersection]),
    nta_name20 = if_else(is.na(intersection), '', nta20_example$nta_name20[intersection])) |>
  relocate(geometry, .after = everything())

treepoints2020 <- treepoints2020 |>
  st_drop_geometry() |>
  select(-nta_name, -nta_code) |>
  distinct()

treepoints2020
```

```
# A tibble: 683,247 × 5
   latitude    longitude    intersection nta_code20 nta_name20
   <chr>       <chr>               <int> <chr>      <chr>
 1 40.72309177 -73.84421522          185 QN0602     Forest Hills
 2 40.79411067 -73.81867946          187 QN0702     Whitestone-Beechhurst
 3 40.71758074 -73.9366077             4 BK0104     East Williamsburg
 4 40.71353749 -73.93445616            4 BK0104     East Williamsburg
 5 40.66677776 -73.97597938           22 BK0602     Park Slope
 6 40.77004563 -73.98494997          137 MN0701     Upper West Side-Lincoln Squ…
 7 40.77020969 -73.98533807          137 MN0701     Upper West Side-Lincoln Squ…
 8 40.76272385 -73.98729652          129 MN0402     Hell's Kitchen
 9 40.59657931 -74.07625483          248 SI0201     Grasmere-Arrochar-South Bea…
10 40.58635725 -73.96974394           43 BK1301     Gravesend (South)
```

Answer

Now merge back in:

```
df2020 <- nytrees_example |>
  left_join(treepoints2020, by = c("longitude", "latitude")) |>
  relocate(nta_code20, nta_name20, .after = nta_code)
df2020
```

```
Simple feature collection with 683683 features and 8 fields
Geometry type: POINT
Dimension:     XY
Bounding box:  xmin: 913349.3 ymin: 120973.8 xmax: 1067248 ymax: 271894.1
Projected CRS: NAD83 / New York Long Island (ftUS)
# A tibble: 683,683 × 9
   nta_code nta_code20 nta_name20         nta_name spc_common latitude longitude
   <chr>    <chr>      <chr>              <chr>    <chr>      <chr>    <chr>
 1 QN17     QN0602     Forest Hills       Forest … red maple  40.7230… -73.8442…
 2 QN49     QN0702     Whitestone-Beechh… Whitest… pin oak    40.7941… -73.8186…
 3 BK90     BK0104     East Williamsburg  East Wi… honeylocu… 40.7175… -73.9366…
 4 BK90     BK0104     East Williamsburg  East Wi… honeylocu… 40.7135… -73.9344…
 5 BK37     BK0602     Park Slope         Park Sl… American … 40.6667… -73.9759…
 6 MN14     MN0701     Upper West Side-L… Lincoln… honeylocu… 40.7700… -73.9849…
 7 MN14     MN0701     Upper West Side-L… Lincoln… honeylocu… 40.7702… -73.9853…
 8 MN15     MN0402     Hell's Kitchen     Clinton  American … 40.7627… -73.9872…
 9 SI14     SI0201     Grasmere-Arrochar… Grasmer… honeylocu… 40.5965… -74.0762…
10 BK26     BK1301     Gravesend (South)  Gravese… London pl… 40.5863… -73.9697…
# ℹ 683,673 more rows
```

Example 2: nycdogs again

The nycdogs package

```r
library(nycdogs)
nyc_license
```

```
# A tibble: 493,072 × 9
   animal_name animal_gender animal_birth_year breed_rc         borough zip_code
   <chr>       <chr>                     <dbl> <chr>            <chr>      <int>
 1 Paige       F                          2014 Pit Bull (or Mi… Manhat…    10035
 2 Yogi        M                          2010 Boxer            Bronx      10465
 3 Ali         M                          2014 Basenji          Manhat…    10013
 4 Queen       F                          2013 Akita Crossbreed Manhat…    10013
 5 Lola        F                          2009 Maltese          Manhat…    10028
 6 Ian         M                          2006 Unknown          Manhat…    10013
 7 Buddy       M                          2008 Unknown          Manhat…    10025
 8 Chewbacca   F                          2012 Labrador (or Cr… Manhat…    10013
 9 Heidi-Bo    F                          2007 Dachshund Smoot… Brookl…    11215
10 Massimo     M                          2009 Bull Dog, French Brookl…    11201
# ℹ 493,062 more rows
# ℹ 3 more variables: license_issued_date <date>, license_expired_date <date>,
#   extract_year <dbl>
```

The nycdogs package

The metadata tells you this is not a regular tibble.

```
nyc_zips
```

```
Simple feature collection with 262 features and 11 fields
Geometry type: POLYGON
Dimension:     XY
Bounding box:  xmin: -74.25576 ymin: 40.49584 xmax: -73.6996 ymax: 40.91517
Geodetic CRS:  WGS 84
# A tibble: 262 × 12
   objectid zip_code po_name     state borough st_fips cty_fips bld_gpostal_code
      <int>    <int> <chr>       <chr> <chr>   <chr>   <chr>               <int>
 1        1    11372 Jackson He… NY    Queens  36      081                     0
 2        2    11004 Glen Oaks   NY    Queens  36      081                     0
 3        3    11040 New Hyde P… NY    Queens  36      081                     0
 4        4    11426 Bellerose   NY    Queens  36      081                     0
 5        5    11365 Fresh Mead… NY    Queens  36      081                     0
 6        6    11373 Elmhurst    NY    Queens  36      081                     0
 7        7    11001 Floral Park NY    Queens  36      081                     0
 8        8    11375 Forest Hil… NY    Queens  36      081                     0
 9        9    11427 Queens Vil… NY    Queens  36      081                     0
10       10    11374 Rego Park   NY    Queens  36      081                     0
# ℹ 252 more rows
```

The nycdogs package

```
nyc_zips ▷
  select(objectid:borough)
```

```
Simple feature collection with 262 features and 5 fields
Geometry type: POLYGON
Dimension:     XY
Bounding box:  xmin: -74.25576 ymin: 40.49584 xmax: -73.6996 ymax: 40.91517
Geodetic CRS:  WGS 84
# A tibble: 262 × 6
   objectid zip_code po_name         state borough                     geometry
      <int>    <int> <chr>           <chr> <chr>                  <POLYGON [°]>
 1        1    11372 Jackson Heights NY    Queens  ((-73.86942 40.74916, -73.89…
 2        2    11004 Glen Oaks       NY    Queens  ((-73.71068 40.75004, -73.70…
 3        3    11040 New Hyde Park   NY    Queens  ((-73.70098 40.7389, -73.703…
 4        4    11426 Bellerose       NY    Queens  ((-73.7227 40.75373, -73.722…
 5        5    11365 Fresh Meadows   NY    Queens  ((-73.81089 40.72717, -73.81…
 6        6    11373 Elmhurst        NY    Queens  ((-73.88722 40.72753, -73.88…
 7        7    11001 Floral Park     NY    Queens  ((-73.70098 40.7389, -73.699…
 8        8    11375 Forest Hills    NY    Queens  ((-73.85625 40.73672, -73.85…
 9        9    11427 Queens Village  NY    Queens  ((-73.74169 40.73682, -73.73…
10       10    11374 Rego Park       NY    Queens  ((-73.86451 40.73407, -73.85…
# ℹ 252 more rows
```

The `polygon` column is a list of lat/lon points that, when joined, draw the outline of the zip code area. This is *much* more compact than a big table where every row is a single point.

Let’s make a summary table

```
nyc_license
```

```
# A tibble: 493,072 × 9
   animal_name animal_gender animal_birth_year
breed_rc         borough zip_code
   <chr>       <chr>                     <dbl>
<chr>            <chr>      <int>
 1 Paige       F                          2014
Pit Bull (or Mi… Manhat…    10035
 2 Yogi        M                          2010
Boxer            Bronx      10465
 3 Ali         M                          2014
Basenji          Manhat…    10013
 4 Queen       F                          2013
Akita Crossbreed Manhat…    10013
 5 Lola        F                          2009
Maltese          Manhat…    10028
 6 Ian         M                          2006
Unknown          Manhat…    10013
 7 Buddy       M                          2008
Unknown          Manhat…    10025
```

Let’s make a summary table

```
nyc_license  ▷
  filter(extract_year == 2018)
```

```
# A tibble: 117,371 × 9
   animal_name animal_gender animal_birth_year
breed_rc         borough zip_code
   <chr>       <chr>                     <dbl>
<chr>            <chr>      <int>
 1 Ali         M                          2014
Basenji          Manhat…    10013
 2 Ian         M                          2006
Unknown          Manhat…    10013
 3 Chewbacca   F                          2012
Labrador (or Cr… Manhat…    10013
 4 Lola        F                          2006
Miniature Pinsc… Manhat…    10022
 5 Lucy        F                          2014
Dachshund Smoot… Brookl…    11215
 6 June        F                          2010
Cavalier King C… Brookl…    11238
 7 Apple       M                          2013
Havanese         Manhat…    10025
```

Let's make a summary table

```
nyc_license  ▷
  filter(extract_year == 2018) ▷
  group_by(breed_rc, zip_code)
```

```
# A tibble: 117,371 × 9
# Groups:   breed_rc, zip_code [18,945]
   animal_name animal_gender animal_birth_year
breed_rc         borough zip_code
   <chr>       <chr>                     <dbl>
<chr>            <chr>      <int>
 1 Ali         M                          2014
Basenji          Manhat…    10013
 2 Ian         M                          2006
Unknown          Manhat…    10013
 3 Chewbacca   F                          2012
Labrador (or Cr… Manhat…    10013
 4 Lola        F                          2006
Miniature Pinsc… Manhat…    10022
 5 Lucy        F                          2014
Dachshund Smoot… Brookl…    11215
 6 June        F                          2010
Cavalier King C… Brookl…    11238
 7 Apple       M                          2013
```

Let's make a summary table

```r
nyc_license  ▷
  filter(extract_year == 2018) ▷
  group_by(breed_rc, zip_code) ▷
  tally()
```

```
# A tibble: 18,945 × 3
# Groups:   breed_rc [311]
   breed_rc      zip_code     n
   <chr>            <int> <int>
 1 Affenpinscher    10005     1
 2 Affenpinscher    10011     1
 3 Affenpinscher    10013     1
 4 Affenpinscher    10014     1
 5 Affenpinscher    10016     1
 6 Affenpinscher    10017     1
 7 Affenpinscher    10018     1
 8 Affenpinscher    10019     1
 9 Affenpinscher    10021     1
10 Affenpinscher    10023     1
# ℹ 18,935 more rows
```

Let's make a summary table

```r
nyc_license  ▷
  filter(extract_year == 2018) ▷
  group_by(breed_rc, zip_code) ▷
  tally() ▷
  mutate(freq = n / sum(n))
```

```
# A tibble: 18,945 × 4
# Groups:   breed_rc [311]
   breed_rc       zip_code     n   freq
   <chr>             <int> <int>  <dbl>
 1 Affenpinscher    10005     1 0.0303
 2 Affenpinscher    10011     1 0.0303
 3 Affenpinscher    10013     1 0.0303
 4 Affenpinscher    10014     1 0.0303
 5 Affenpinscher    10016     1 0.0303
 6 Affenpinscher    10017     1 0.0303
 7 Affenpinscher    10018     1 0.0303
 8 Affenpinscher    10019     1 0.0303
 9 Affenpinscher    10021     1 0.0303
10 Affenpinscher    10023     1 0.0303
# ℹ 18,935 more rows
```

Let's make a summary table

```r
nyc_license  ▷
  filter(extract_year == 2018) ▷
  group_by(breed_rc, zip_code) ▷
  tally() ▷
  mutate(freq = n / sum(n)) ▷
  filter(breed_rc == "French Bulldog")
```

```
# A tibble: 161 × 4
# Groups:   breed_rc [1]
   breed_rc       zip_code     n    freq
   <chr>             <int> <int>   <dbl>
 1 French Bulldog    10001    27 0.0167
 2 French Bulldog    10002    20 0.0123
 3 French Bulldog    10003    36 0.0222
 4 French Bulldog    10004     9 0.00555
 5 French Bulldog    10005    15 0.00925
 6 French Bulldog    10006     8 0.00494
 7 French Bulldog    10007    17 0.0105
 8 French Bulldog    10009    51 0.0315
 9 French Bulldog    10010    31 0.0191
10 French Bulldog    10011    88 0.0543
# ℹ 151 more rows
```

Let's make a summary table

```
nyc_license  ▷
  filter(extract_year == 2018) ▷
  group_by(breed_rc, zip_code) ▷
  tally() ▷
  mutate(freq = n / sum(n)) ▷
  filter(breed_rc == "French Bulldog") →
  nyc_fb
```

Let's make a summary table

```
nyc_license  ▷
  filter(extract_year == 2018) ▷
  group_by(breed_rc, zip_code) ▷
  tally() ▷
  mutate(freq = n / sum(n)) ▷
  filter(breed_rc == "French Bulldog") →
  nyc_fb
```

Now we have two tables again

```
nyc_zips ▷ select(objectid:st_fips)
```

```
Simple feature collection with 262 features and 6 fields
Geometry type: POLYGON
Dimension:     XY
Bounding box:  xmin: -74.25576 ymin: 40.49584 xmax: -73.6996 ymax: 40.91517
Geodetic CRS:  WGS 84
# A tibble: 262 × 7
   objectid zip_code po_name     state borough st_fips                  geometry
      <int>    <int> <chr>       <chr> <chr>   <chr>               <POLYGON [°]>
 1        1    11372 Jackson He… NY    Queens  36      ((-73.86942 40.74916, -7…
 2        2    11004 Glen Oaks   NY    Queens  36      ((-73.71068 40.75004, -7…
 3        3    11040 New Hyde P… NY    Queens  36      ((-73.70098 40.7389, -73…
 4        4    11426 Bellerose   NY    Queens  36      ((-73.7227 40.75373, -73…
 5        5    11365 Fresh Mead… NY    Queens  36      ((-73.81089 40.72717, -7…
 6        6    11373 Elmhurst    NY    Queens  36      ((-73.88722 40.72753, -7…
 7        7    11001 Floral Park NY    Queens  36      ((-73.70098 40.7389, -73…
 8        8    11375 Forest Hil… NY    Queens  36      ((-73.85625 40.73672, -7…
 9        9    11427 Queens Vil… NY    Queens  36      ((-73.74169 40.73682, -7…
10       10    11374 Rego Park   NY    Queens  36      ((-73.86451 40.73407, -7…
# ℹ 252 more rows
```

```
nyc_fb ▷ select(breed_rc:n)
```

```
# A tibble: 161 × 3
# Groups:   breed_rc [1]
   breed_rc       zip_code     n
   <chr>             <int> <int>
 1 French Bulldog    10001    27
 2 French Bulldog    10002    20
 3 French Bulldog    10003    36
 4 French Bulldog    10004     9
 5 French Bulldog    10005    15
 6 French Bulldog    10006     8
 7 French Bulldog    10007    17
 8 French Bulldog    10009    51
 9 French Bulldog    10010    31
10 French Bulldog    10011    88
# ℹ 151 more rows
```

Join them:

```
fb_map ← left_join(nyc_zips, nyc_fb, by = "zip_code")
```

Ready to map

```
fb_map ▷ select(zip_code, po_name, borough, breed_rc:freq, geometry)
```

```
Simple feature collection with 262 features and 6 fields
Geometry type: POLYGON
Dimension:     XY
Bounding box:  xmin: -74.25576 ymin: 40.49584 xmax: -73.6996 ymax: 40.91517
Geodetic CRS:  WGS 84
# A tibble: 262 × 7
   zip_code po_name    borough breed_rc     n     freq                  geometry
      <int> <chr>      <chr>   <chr>    <int>    <dbl>             <POLYGON [°]>
 1    11372 Jackson H… Queens  French …    13  8.02e-3 ((-73.86942 40.74916, -7…
 2    11004 Glen Oaks  Queens  French …     1  6.17e-4 ((-73.71068 40.75004, -7…
 3    11040 New Hyde … Queens  <NA>        NA NA       ((-73.70098 40.7389, -73…
 4    11426 Bellerose  Queens  French …     1  6.17e-4 ((-73.7227 40.75373, -73…
 5    11365 Fresh Mea… Queens  French …     7  4.32e-3 ((-73.81089 40.72717, -7…
 6    11373 Elmhurst   Queens  French …    14  8.64e-3 ((-73.88722 40.72753, -7…
 7    11001 Floral Pa… Queens  <NA>        NA NA       ((-73.70098 40.7389, -73…
 8    11375 Forest Hi… Queens  French …     8  4.94e-3 ((-73.85625 40.73672, -7…
 9    11427 Queens Vi… Queens  French …     2  1.23e-3 ((-73.74169 40.73682, -7…
10    11374 Rego Park  Queens  French …     6  3.70e-3 ((-73.86451 40.73407, -7…
# ℹ 252 more rows
```

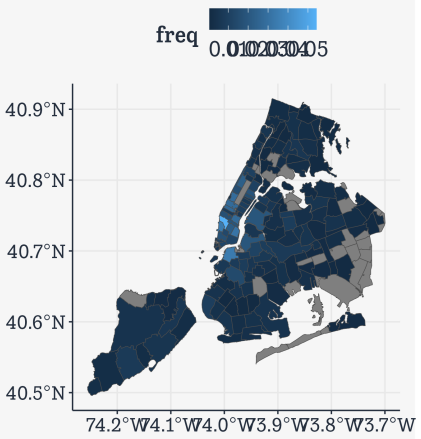
A NYC map theme

Just moving the legend, really.

```
theme_nymap <- function(base_size=9, base_family="") {
    require(grid)
    theme_bw(base_size=base_size, base_family=base_family) %+replace%
        theme(axis.line=element_blank(),
              axis.text=element_blank(),
              axis.ticks=element_blank(),
              axis.title=element_blank(),
              panel.background=element_blank(),
              panel.border=element_blank(),
              panel.grid=element_blank(),
              panel.spacing=unit(0, "lines"),
              plot.background=element_blank(),
              legend.justification = c(0,0),
              legend.position = c(0.05, 0.58),
              legend.direction = "horizontal"
        )
}
```

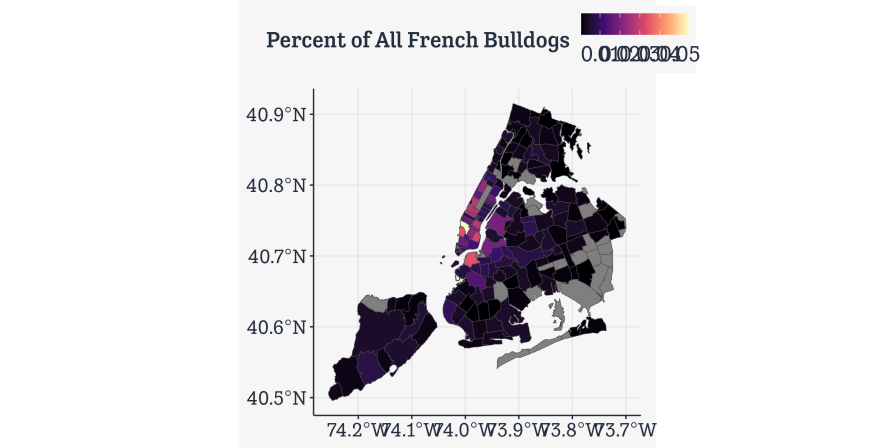
First cut at a map

```
fb_map ▷
  ggplot(mapping = aes(fill = freq)) +
  geom_sf(color = "gray30", size = 0.1)
```



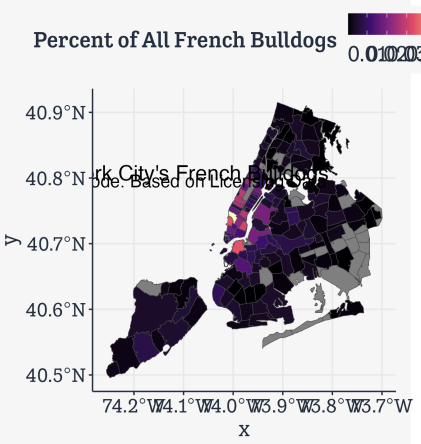
First cut at a map

```
fb_map ▷
  ggplot(mapping = aes(fill = freq)) +
  geom_sf(color = "gray30", size = 0.1) +
  scale_fill_viridis_c(option = "A") +
  labs(fill = "Percent of All French Bulldogs")
```



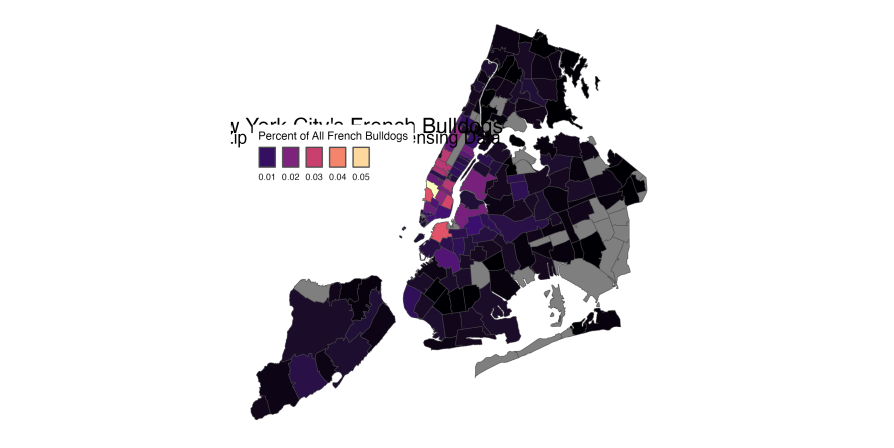
First cut at a map

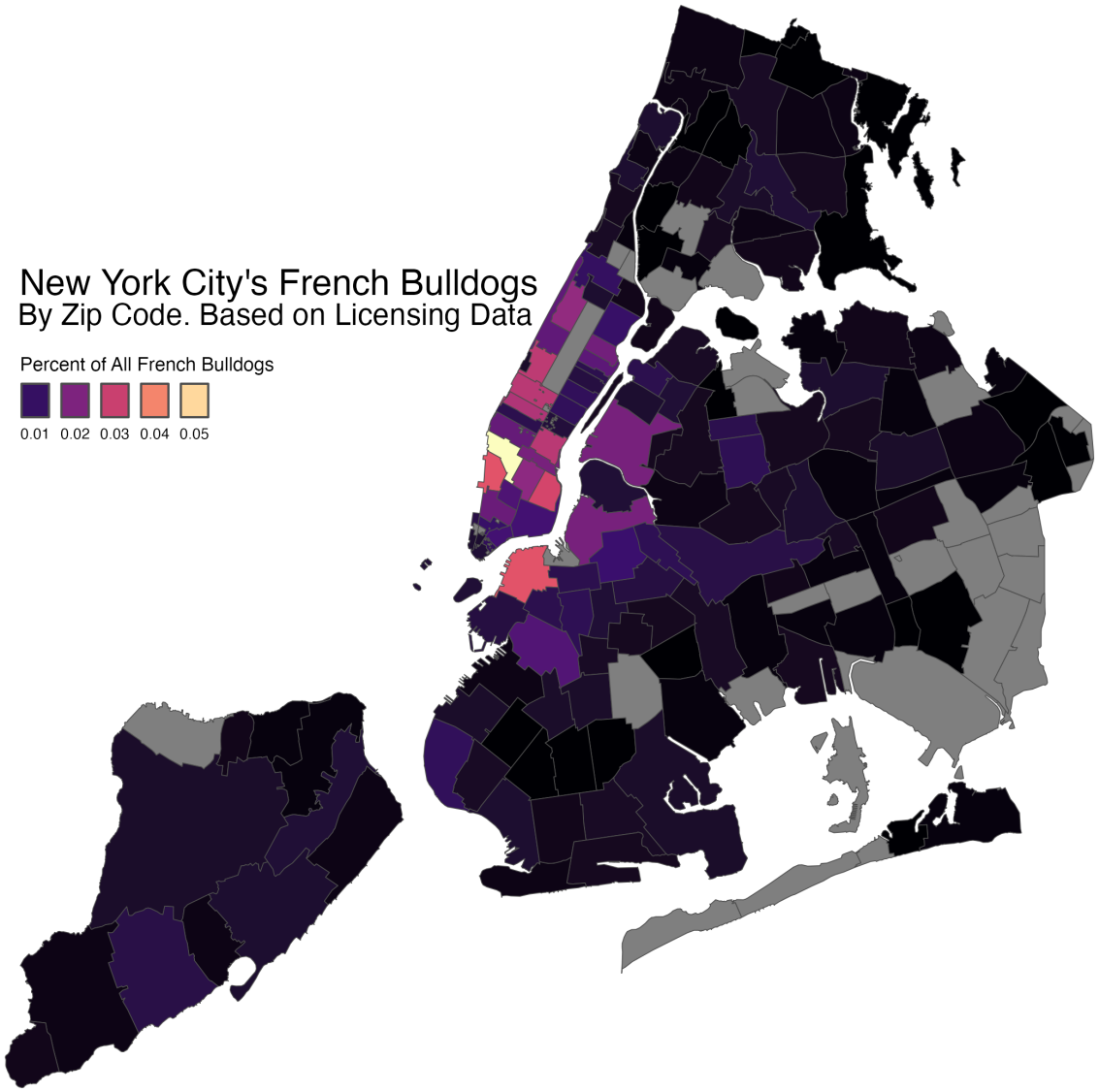
```
fb_map ▷
  ggplot(mapping = aes(fill = freq)) +
  geom_sf(color = "gray30", size = 0.1) +
  scale_fill_viridis_c(option = "A") +
  labs(fill = "Percent of All French Bulldogs")
  annotate(geom = "text",
             x = -74.145 + 0.029,
             y = 40.82-0.012,
           label = "New York City's French Bull
           size = 6) +
    annotate(geom = "text",
             x = -74.1468 + 0.029,
             y = 40.8075-0.012,
           label = "By Zip Code. Based on Licen
           size = 5)
```



First cut at a map

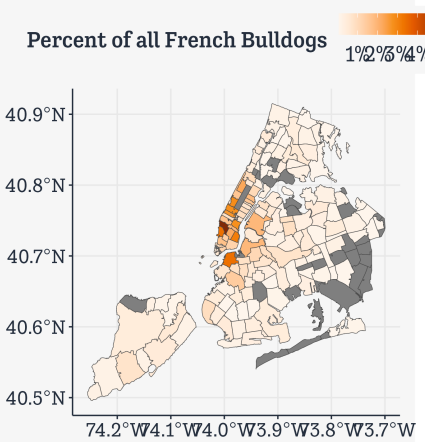
```
fb_map ▷
  ggplot(mapping = aes(fill = freq)) +
  geom_sf(color = "gray30", size = 0.1) +
  scale_fill_viridis_c(option = "A") +
  labs(fill = "Percent of All French Bulldogs")
  annotate(geom = "text",
             x = -74.145 + 0.029,
             y = 40.82-0.012,
           label = "New York City's French Bull
           size = 6) +
  annotate(geom = "text",
             x = -74.1468 + 0.029,
             y = 40.8075-0.012,
           label = "By Zip Code. Based on Licen
           size = 5) +
  kjhslides::kjh_theme_nymap() +
  guides(fill =
           guide_legend(title.position = "top",
                 label.position = "bottom",
                 keywidth = 1,
                 nrow = 1))
```




New York City's French Bulldogs
By Zip Code. Based on Licensing Data
Percent of All French Bulldogs
0.01
0.02
0.03
0.04
0.05

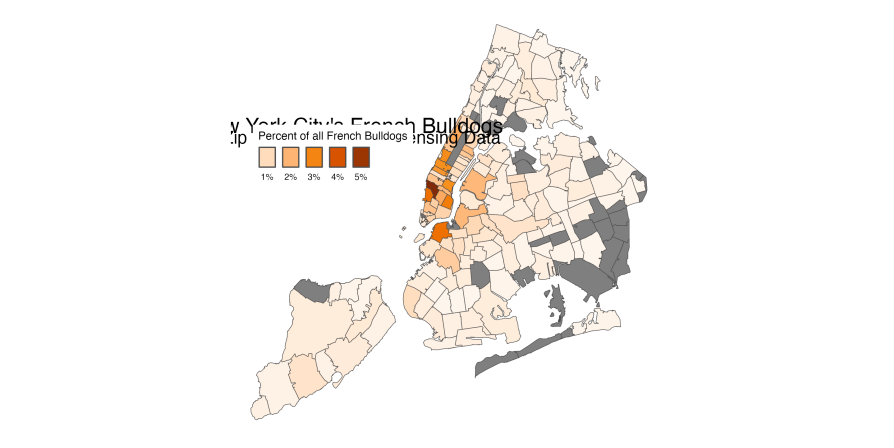
Use a different palette

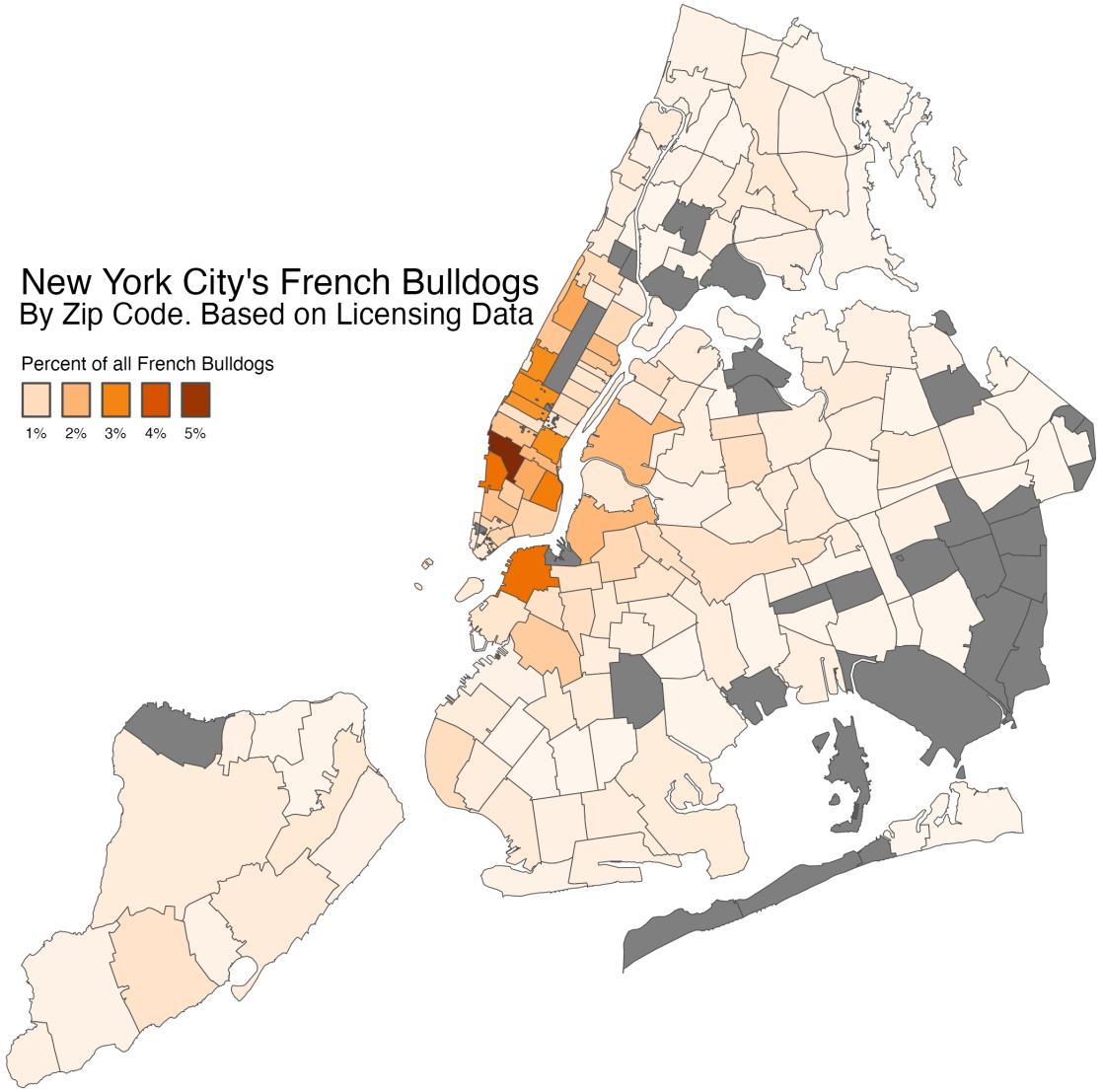
```r
library(colorspace)

fb_map ▷
  ggplot(mapping = aes(fill = freq)) +
  geom_sf(color = "gray30", size = 0.1) +
  scale_fill_continuous_sequential(
    palette = "Oranges",
    labels = scales::label_percent()) +
  labs(fill = "Percent of all French Bulldogs")
```



Use a different palette

```
fb_map ▷
  ggplot(mapping = aes(fill = freq)) +
  geom_sf(color = "gray30", size = 0.1) +
  scale_fill_continuous_sequential(
    palette = "Oranges",
    labels = scales::label_percent()) +
  labs(fill = "Percent of all French Bulldogs")
  annotate(geom = "text",
           x = -74.145 + 0.029,
           y = 40.82-0.012,
           label = "New York City's French Bull
           size = 6) +
  annotate(geom = "text",
           x = -74.1468 + 0.029,
           y = 40.7955,
           label = "By Zip Code. Based on Licen
           size = 5) +
  kjhslides::kjh_theme_nymap() +
  guides(fill =
           guide_legend(title.position = "top",
                label.position = "bottom",
                keywidth = 1,
```





NYC Dogs Map mark 2

Keep the Zero-count Zips

```r
nyc_license  ▷
  filter(extract_year == 2018) ▷
  group_by(breed_rc, zip_code) ▷
  tally() ▷
  ungroup() ▷
  complete(zip_code, breed_rc,
           fill = list(n = 0)) ▷
  # Regroup to get the right denominator
  group_by(breed_rc) ▷
  mutate(freq = n / sum(n)) ▷
  filter(breed_rc == "French Bulldog") →
  nyc_fb2

fb_map2 ← left_join(nyc_zips,
                    nyc_fb2,
                    by = "zip_code")
```

Keep the Zero-count Zips

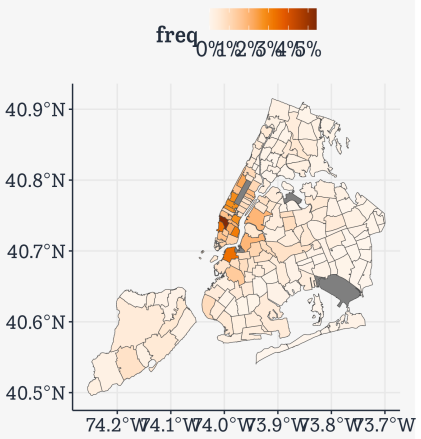
```
fb_map2 ▷ select(zip_code, po_name, borough, breed_rc:freq, geometry)
```

```
Simple feature collection with 262 features and 6 fields
Geometry type: POLYGON
Dimension:     XY
Bounding box:  xmin: -74.25576 ymin: 40.49584 xmax: -73.6996 ymax: 40.91517
Geodetic CRS:  WGS 84
# A tibble: 262 × 7
   zip_code po_name     borough breed_rc     n    freq                  geometry
      <int> <chr>       <chr>   <chr>    <int>   <dbl>             <POLYGON [°]>
 1    11372 Jackson He… Queens  French …    13 8.02e-3 ((-73.86942 40.74916, -7…
 2    11004 Glen Oaks   Queens  French …     1 6.17e-4 ((-73.71068 40.75004, -7…
 3    11040 New Hyde P… Queens  French …     0 0       ((-73.70098 40.7389, -73…
 4    11426 Bellerose   Queens  French …     1 6.17e-4 ((-73.7227 40.75373, -73…
 5    11365 Fresh Mead… Queens  French …     7 4.32e-3 ((-73.81089 40.72717, -7…
 6    11373 Elmhurst    Queens  French …    14 8.64e-3 ((-73.88722 40.72753, -7…
 7    11001 Floral Park Queens  French …     0 0       ((-73.70098 40.7389, -73…
 8    11375 Forest Hil… Queens  French …     8 4.94e-3 ((-73.85625 40.73672, -7…
 9    11427 Queens Vil… Queens  French …     2 1.23e-3 ((-73.74169 40.73682, -7…
10    11374 Rego Park   Queens  French …     6 3.70e-3 ((-73.86451 40.73407, -7…
# ℹ 252 more rows
```

This time, a number of previous NA rows are now zeroes instead.

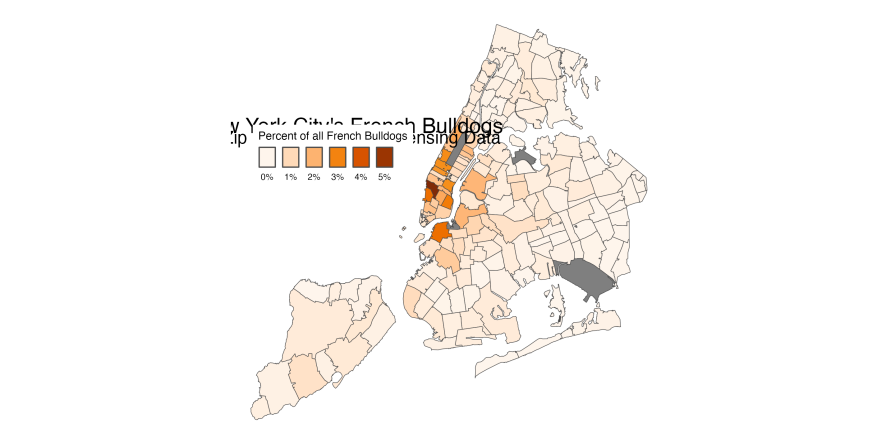
Keep the Zero-count Zips

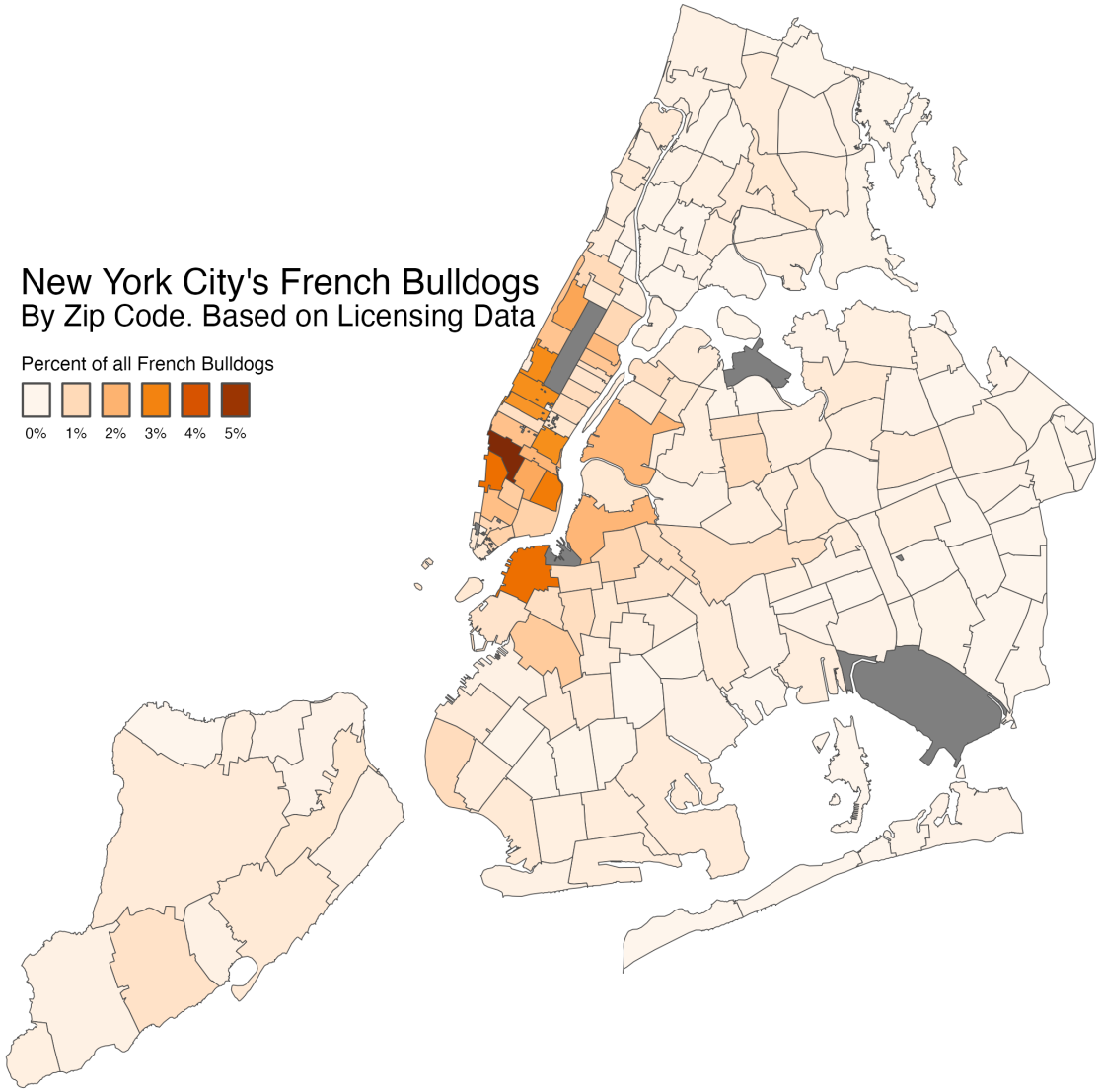
```r
fb_map2 |>
  ggplot(mapping = aes(fill = freq)) +
  geom_sf(color = "gray30", size = 0.1) +
  scale_fill_continuous_sequential(
    palette = "Oranges",
    labels = scales::label_percent())
```



Keep the Zero-count Zips

```
fb_map2 ▷
  ggplot(mapping = aes(fill = freq)) +
  geom_sf(color = "gray30", size = 0.1) +
  scale_fill_continuous_sequential(
    palette = "Oranges",
    labels = scales::label_percent()) +
  labs(fill = "Percent of all French Bulldogs")
  annotate(geom = "text",
           x = -74.145 + 0.029,
           y = 40.82-0.012,
           label = "New York City's French Bull
           size = 6) +
  annotate(geom = "text",
           x = -74.1468 + 0.029,
           y = 40.7955,
           label = "By Zip Code. Based on Licen
           size = 5) +
  kjhslides::kjh_theme_nymap() +
  guides(fill =
           guide_legend(title.position = "top",
                        label.position = "botto
                        keywidth = 1,
```



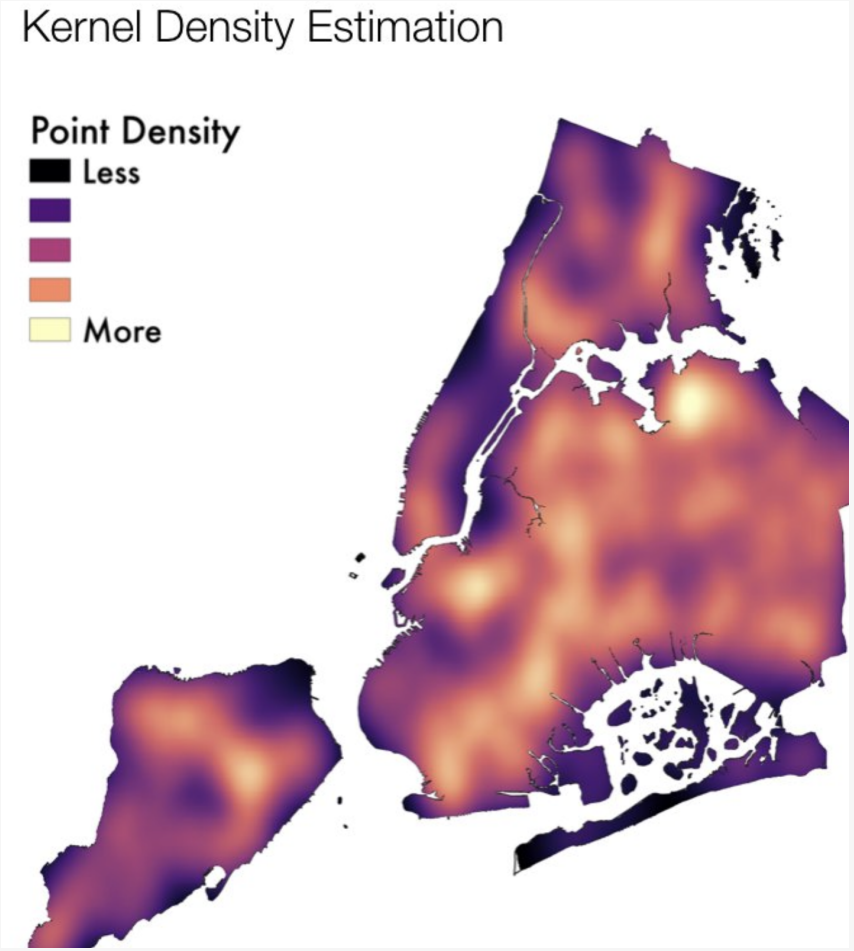


Zero areas properly zero, missing areas properly missing.

Care with Spatial Distributions

A random point-process

Care with Spatial Distributions



A heatmap derived from the random process

Care with Spatial Distributions

A formal test of significant hotspots

Example: Dorling Cartograms

Dorling Cartograms

```r
# install.packages("cartogram")
library(cartogram)
options(tigris_use_cache = TRUE)
```

Dorling Cartograms

```r
pop_names <- tribble(
    ~varname, ~clean,
    "B01003_001", "pop",
    "B01001B_001", "black",
    "B01001A_001", "white",
    "B01001H_001", "nh_white",
    "B01001I_001", "hispanic",
    "B01001D_001", "asian"
  )

pop_names
```

```
# A tibble: 6 × 2
  varname     clean
  <chr>       <chr>
1 B01003_001  pop
2 B01001B_001 black
3 B01001A_001 white
4 B01001H_001 nh_white
5 B01001I_001 hispanic
6 B01001D_001 asian
```

Dorling Cartograms

```
fips_pop <- get_acs(geography = "county",
                    variables = pop_names$varname,
                    cache_table = TRUE) |>
  left_join(pop_names, join_by(variable == varname)) |>
  mutate(variable = clean) |>
  select(-clean, -moe) |>
  pivot_wider(names_from = variable, values_from = estimate) |>
  rename(fips = GEOID, name = NAME) |>
  mutate(prop_pop = pop/sum(pop),
         prop_black = black/pop,
         prop_hisp = hispanic/pop,
         prop_white = white/pop,
         prop_nhwhite = nh_white/pop,
         prop_asian = asian/pop)

fips_map <- get_acs(geography = "county",
                    variables = "B01001_001",
                    geometry = TRUE,
                    shift_geo = FALSE,
                    cache_table = TRUE) |>
  select(GEOID, NAME, geometry) |>
  rename(fips = GEOID, name = NAME)
```

Dorling Cartograms

```
pop_cat_labels <- c("<5", as.character(seq(10, 95, 5)), "100")

counties_sf <- fips_map |>
  left_join(fips_pop, by = c("fips", "name")) |>
  mutate(black_disc = cut(prop_black*100,
                          breaks = seq(0, 100, 5),
                          labels = pop_cat_labels,
                          ordered_result = TRUE),
         hisp_disc = cut(prop_hisp*100,
                          breaks = seq(0, 100, 5),
                          labels = pop_cat_labels,
                          ordered_result = TRUE),
         nhwhite_disc = cut(prop_nhwhite*100,
                          breaks = seq(0, 100, 5),
                          labels = pop_cat_labels,
                          ordered_result = TRUE),
        asian_disc = cut(prop_asian*100,
                          breaks = seq(0, 100, 5),
                          labels = pop_cat_labels,
                          ordered_result = TRUE)) |>
  sf::st_transform(crs = 2163)
```

Dorling Cartograms

```
counties_sf
```

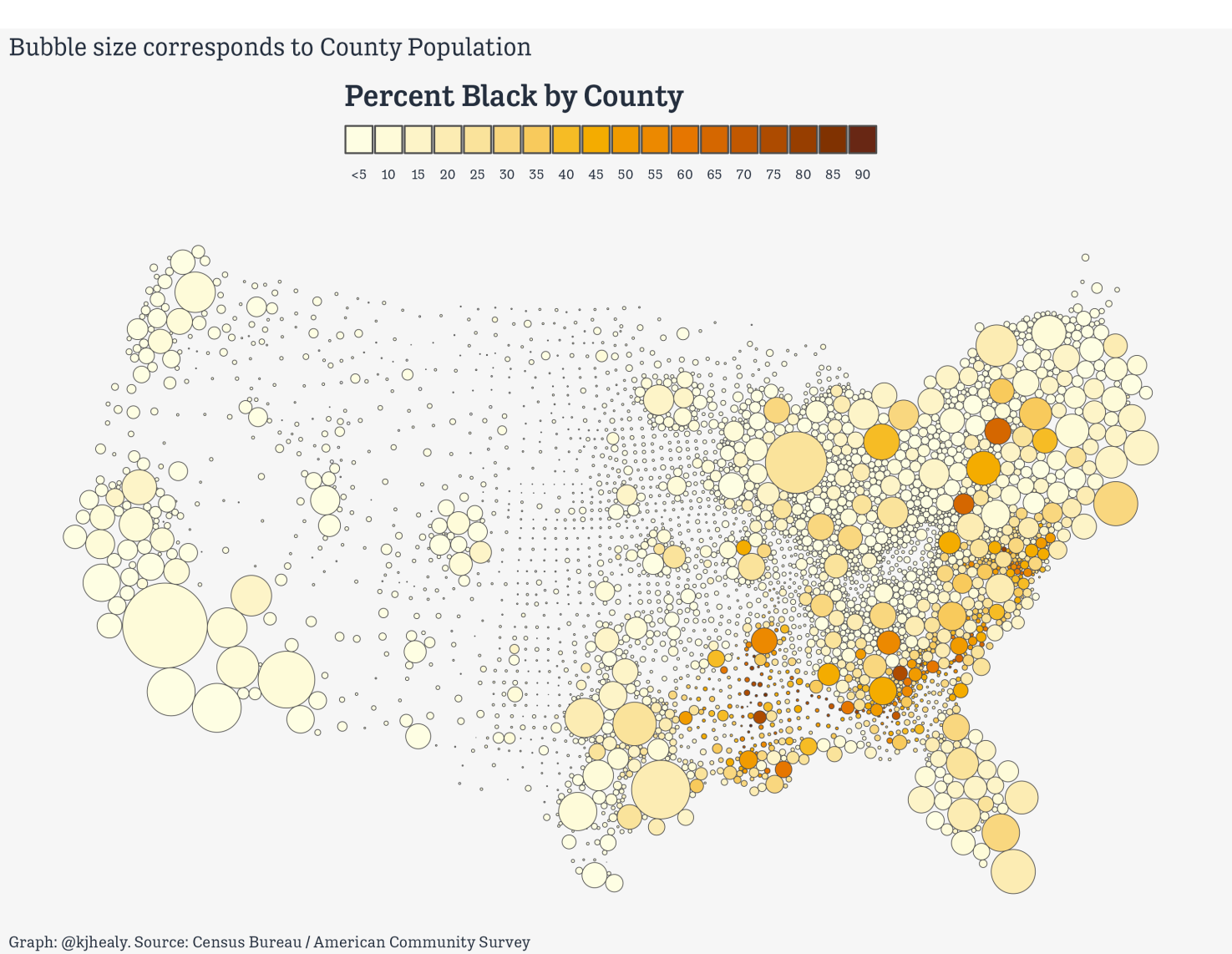
```
Simple feature collection with 3222 features and 18 fields
Geometry type: MULTIPOLYGON
Dimension:     XY
Bounding box:  xmin: -6433624 ymin: -2354597 xmax: 3667987 ymax: 3912355
Projected CRS: NAD27 / US National Atlas Equal Area
First 10 features:
    fips                           name   white  black   asian nh_white
1  01069        Houston County, Alabama   71260  29166     987    69420
2  01023        Choctaw County, Alabama    7180   5062      15     7162
3  01005        Barbour County, Alabama   11309  11668     126    11084
4  01107        Pickens County, Alabama   10880   7506      10    10141
5  01033        Colbert County, Alabama   44698   9185     214    44485
6  04012         La Paz County, Arizona   10593     94     153     9358
7  04001         Apache County, Arizona   13556    585     374    12006
8  05081  Little River County, Arkansas    8697   2238      27     8620
9  05121      Randolph County, Arkansas   16846    258      71    16796
10 06037 Los Angeles County, California 3937901 780993 1473634  2505177
   hispanic     pop     prop_pop  prop_black  prop_hisp prop_white prop_nhwhite
1      3844  107040 3.201244e-04 0.272477578 0.03591181  0.6657324    0.6485426
```

Dorling Cartograms

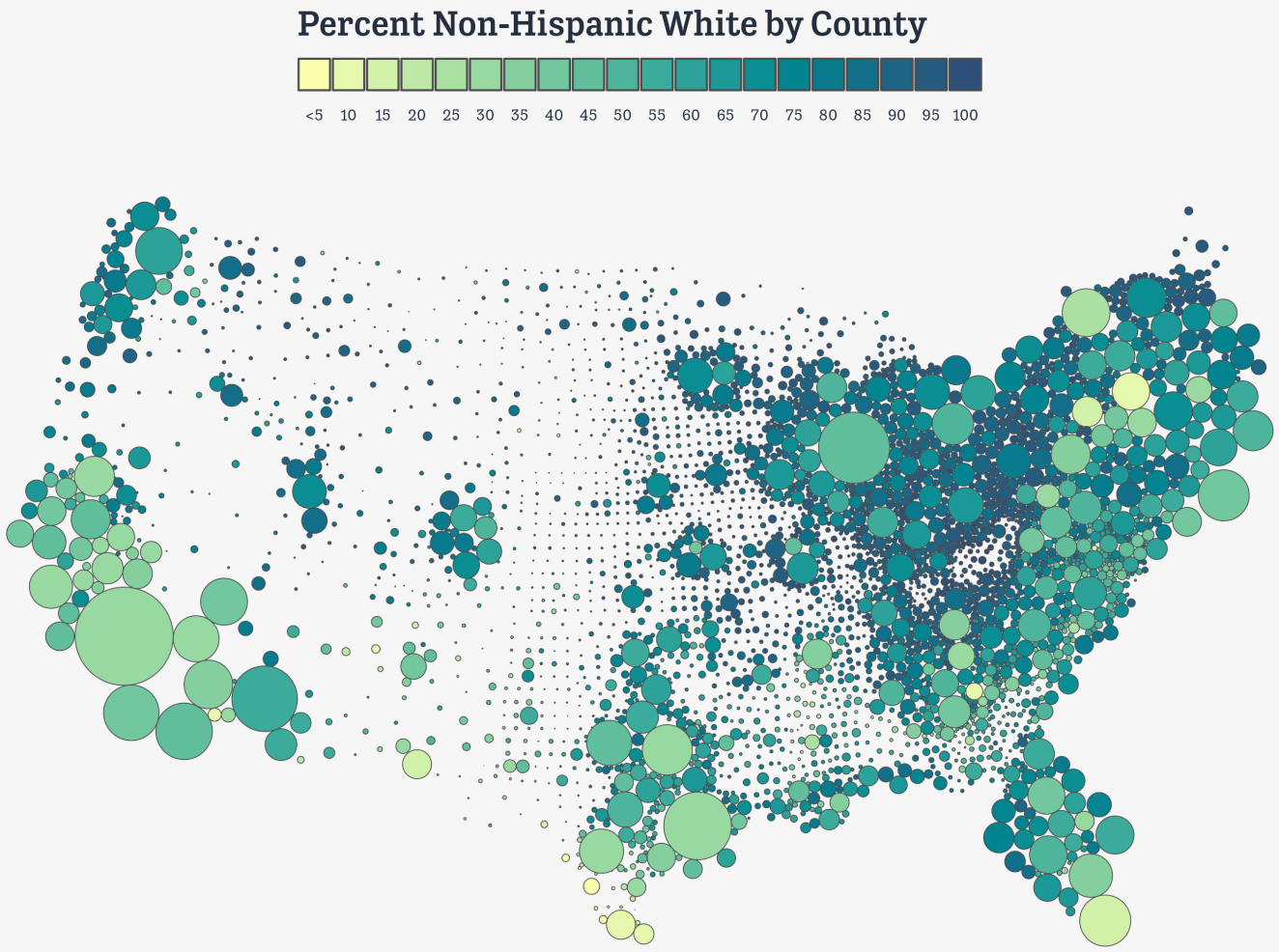
```r
## Be patient
county_dorling <- cartogram_dorling(x = counties_sf,
    weight = "prop_pop",
    k = 0.2, itermax = 100)

out_black <- county_dorling |>
  filter(!str_detect(name, "Alaska|Hawaii|Puerto|Guam")) |>
  ggplot(aes(fill = black_disc)) +
  geom_sf(color = "grey30", size = 0.1) +
  coord_sf(crs = 2163, datum = NA) +
  scale_fill_discrete_sequential(palette = "YlOrBr",
                                 na.translate=FALSE) +
  guides(fill = guide_legend(title.position = "top",
                             label.position = "bottom",
                             nrow = 1)) +
  labs(
     subtitle = "Bubble size corresponds to County Population",
     caption = "Graph: @kjhealy. Source: Census Bureau / American Community Survey",
       fill = "Percent Black by County") +
  theme(legend.position = "top",
        legend.spacing.x = unit(0, "cm"),
        legend.title = element_text(size = rel(1.5), face = "bold"),
```

Dorling Cartograms

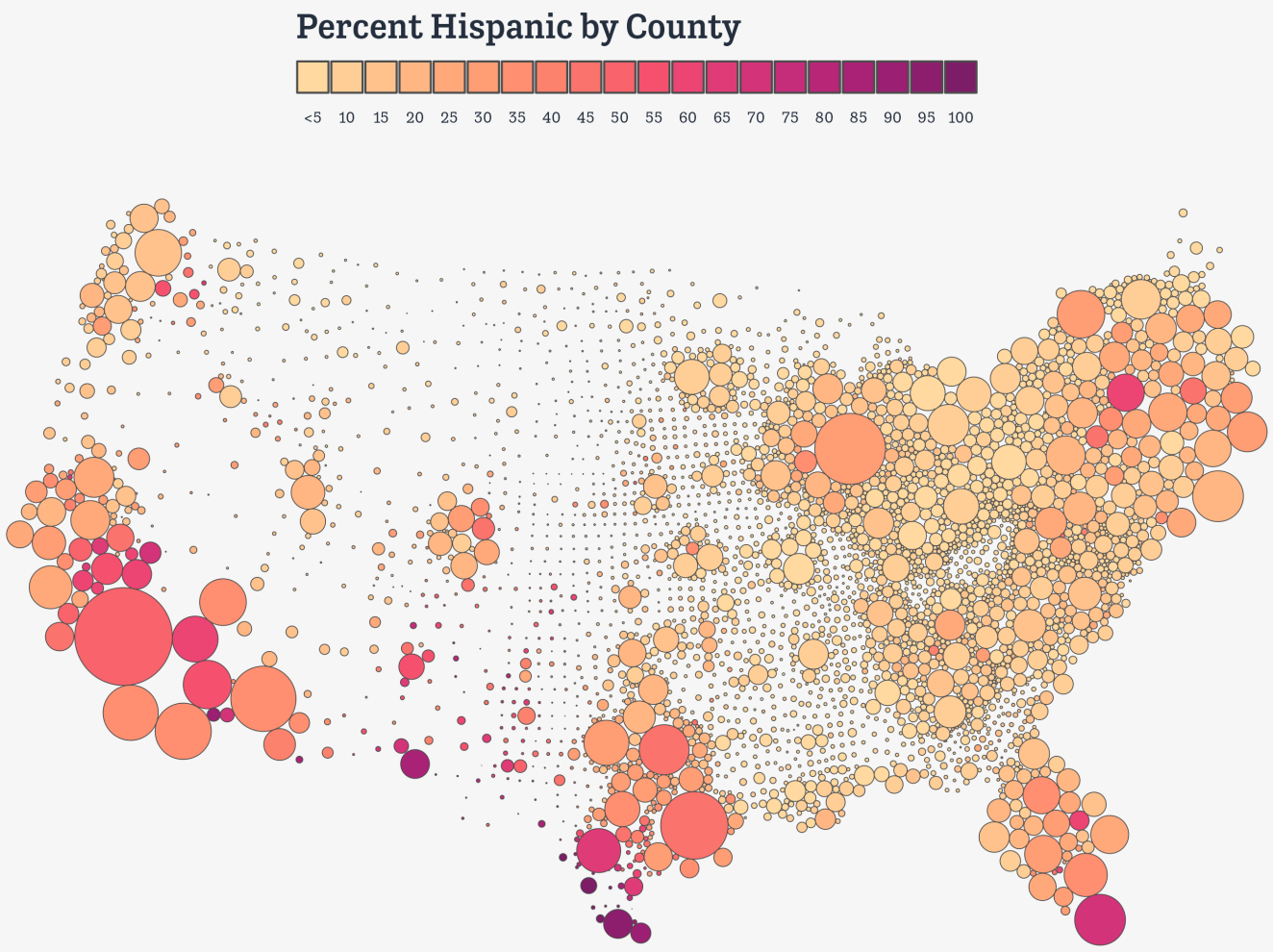
```
print(out_black)
```



```
print(out_white)
```



```
print(out_hispanic)
```



```
print(out_asian)
```