Introduction to Vector Data with R

Practical Examples Using sf, ggplot2 and dplyr

Andy Teucher
Ministry of Environment and Climate Change Strategy

Sam Albers
Ministry of Citizens' Services

2019-11-06

Outline

- Brief review of dplyr
- Introduction to Simple Features and the `sf` package
- Coordinate Reference Systems
- Reading various spatial data formats
- Basic manipulation, summarization, visualization
- Multi-layer plots with `ggplot2`
- Spatial Operations
- Making publication plots with `ggplot2`

What are you going to come away with?

- Review of dplyr
- Use `sf` package to:
  - Read a variety of vector data into R
  - Understand sf objects in R
  - Perform spatial operations, manipulations and joins with `dplyr`
- Plot vector data in R (`ggplot2` & `mapview`)
- Use `bcdata` package to retrieve spatial data from the B.C. data catalogue

`dplyr` verbs

Functions with English meanings that map directly to the action being taken when that function is called

Installation: `install.packages("dplyr")`

- `select()` picks variables based on their names.
- `filter()` picks cases based on their values.
- `summarise()` reduces multiple values down to a single summary.
- `arrange()` changes the ordering of the rows.
- `mutate()` adds new variables that are functions of existing variables
- `%>%` a special symbol to chain operations. Read it as "then"

For an offline tutorial:
http://swcarpentry.github.io/r-novice-gapminder/13-dplyr/index.html

Artwork by @allison_horst

Using select()

```
library(dplyr)
select(starwars, name, height, hair_color, homeworld)
#> # A tibble: 87 x 4
#>    name               height hair_color    homeworld
#>    <chr>               <int> <chr>         <chr>
#>  1 Luke Skywalker        172 blond         Tatooine
#>  2 C-3PO                 167 <NA>          Tatooine
#>  3 R2-D2                  96 <NA>          Naboo
#>  4 Darth Vader           202 none          Tatooine
#>  5 Leia Organa           150 brown         Alderaan
#>  6 Owen Lars             178 brown, grey   Tatooine
#>  7 Beru Whitesun lars    165 brown         Tatooine
#>  8 R5-D4                  97 <NA>          Tatooine
#>  9 Biggs Darklighter     183 black         Tatooine
#> 10 Obi-Wan Kenobi        182 auburn, white Stewjon
#> # … with 77 more rows
```

Using filter()

To retain only a subset of rows:

```
filter(starwars,  homeworld == "Tatooine")
#> # A tibble: 10 x 13
#>    name  height  mass hair_color skin_color eye_color birth_year gender homeworld species
#>    <chr>  <int> <dbl> <chr>      <chr>      <chr>           <dbl> <chr>  <chr>     <chr>
#>  1 Luke…    172    77 blond      fair       blue             19   male   Tatooine  Human
#>  2 C-3PO    167    75 <NA>       gold       yellow          112   <NA>   Tatooine  Droid
#>  3 Dart…    202   136 none       white      yellow           41.9 male   Tatooine  Human
#>  4 Owen…    178   120 brown, gr… light      blue             52   male   Tatooine  Human
#>  5 Beru…    165    75 brown      light      blue             47   female Tatooine  Human
#>  6 R5-D4     97    32 <NA>       white, red red              NA   <NA>   Tatooine  Droid
#>  7 Bigg…    183    84 black      light      brown            24   male   Tatooine  Human
#>  8 Anak…    188    84 blond      fair       blue             41.9 male   Tatooine  Human
#>  9 Shmi…    163    NA black      fair       brown            72   female Tatooine  Human
#> 10 Clie…    183    NA brown      fair       blue             82   male   Tatooine  Human
#> # … with 3 more variables: films <list>, vehicles <list>, starships <list>
```

The pipe (%>%): Combining two statements

```
starwars %>%
  select(name, height, hair_color, homeworld) %>%
  filter(homeworld == "Tatooine")
#> # A tibble: 10 x 4
#>    name               height hair_color  homeworld
#>    <chr>               <int> <chr>       <chr>
#>  1 Luke Skywalker        172 blond       Tatooine
#>  2 C-3PO                 167 <NA>        Tatooine
#>  3 Darth Vader           202 none        Tatooine
#>  4 Owen Lars             178 brown, grey Tatooine
#>  5 Beru Whitesun lars    165 brown       Tatooine
#>  6 R5-D4                  97 <NA>        Tatooine
#>  7 Biggs Darklighter     183 black       Tatooine
#>  8 Anakin Skywalker      188 blond       Tatooine
#>  9 Shmi Skywalker        163 black       Tatooine
#> 10 Cliegg Lars           183 brown       Tatooine
```

Using group_by()

```
starwars %>%
  select(name, height, hair_color, homeworld) %>%
  filter(homeworld == "Tatooine") %>%
  group_by(hair_color)
#> # A tibble: 10 x 4
#> # Groups:   hair_color [6]
#>    name                height hair_color  homeworld
#>    <chr>                <int> <chr>       <chr>
#>  1 Luke Skywalker         172 blond       Tatooine
#>  2 C-3PO                  167 <NA>        Tatooine
#>  3 Darth Vader            202 none        Tatooine
#>  4 Owen Lars              178 brown, grey Tatooine
#>  5 Beru Whitesun lars     165 brown       Tatooine
#>  6 R5-D4                   97 <NA>        Tatooine
#>  7 Biggs Darklighter      183 black       Tatooine
#>  8 Anakin Skywalker       188 blond       Tatooine
#>  9 Shmi Skywalker         163 black       Tatooine
#> 10 Cliegg Lars            183 brown       Tatooine
```

Using summarize()

```
starwars %>%
  select(name, height, hair_color, homeworld) %>%
  filter(homeworld == "Tatooine") %>%
  group_by(hair_color) %>%
  summarise(sd_height = sd(height, na.rm = TRUE))
#> # A tibble: 6 x 2
#>   hair_color  sd_height
#>   <chr>           <dbl>
#> 1 black            14.1
#> 2 blond            11.3
#> 3 brown            12.7
#> 4 brown, grey      NA
#> 5 none             NA
#> 6 <NA>             49.5
```

Your turn

- Using the `starwars` data set and `dplyr` syntax find the mean mass of humans grouped by homeworld and arranged by mean mass column.

Solution

```
starwars %>%
  filter(species == "Human") %>%
  group_by(homeworld) %>%
  summarise(mean_mass = mean(mass, na.rm = TRUE)) %>%
  arrange(desc(mean_mass))
#> # A tibble: 16 x 2
#>    homeworld    mean_mass
#>    <chr>            <dbl>
#>  1 Bestine IV       110
#>  2 Tatooine          96
#>  3 <NA>              89
#>  4 Haruun Kal        84
#>  5 Serenno           80
#>  6 Bespin            79
#>  7 Concord Dawn      79
#>  8 Socorro           79
#>  9 Corellia          78.5
#> 10 Kamino            78.2
#> 11 Stewjon           77
#> 12 Naboo             68.3
#> 13 Alderaan          64
#> 14 Chandrila        NaN
```

meet

GIS: it's what's for breakfast

Tails from the Field

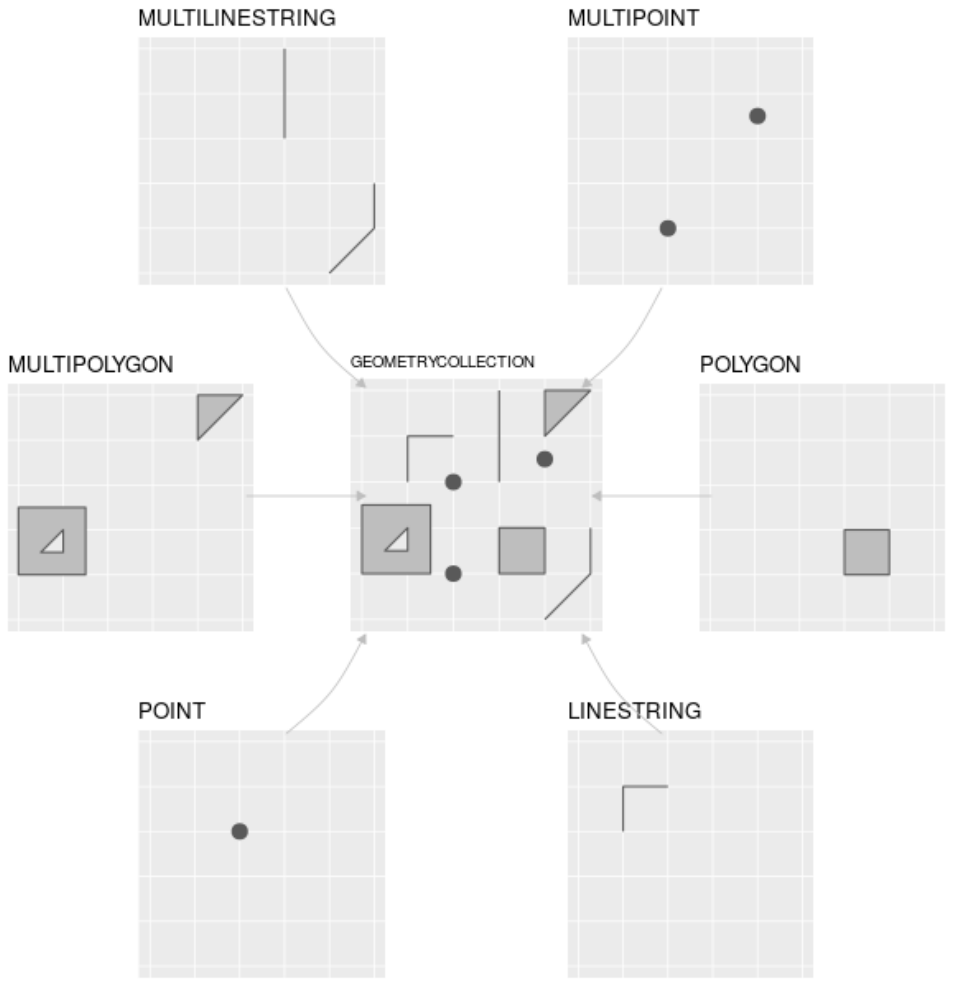
Image: http://www.tailsfromthefield.net

Simple Features

the 'sf' R package

Replaces

- sp
- rgdal
- rgeos



sf package: https://cran.r-project.org/package=sf

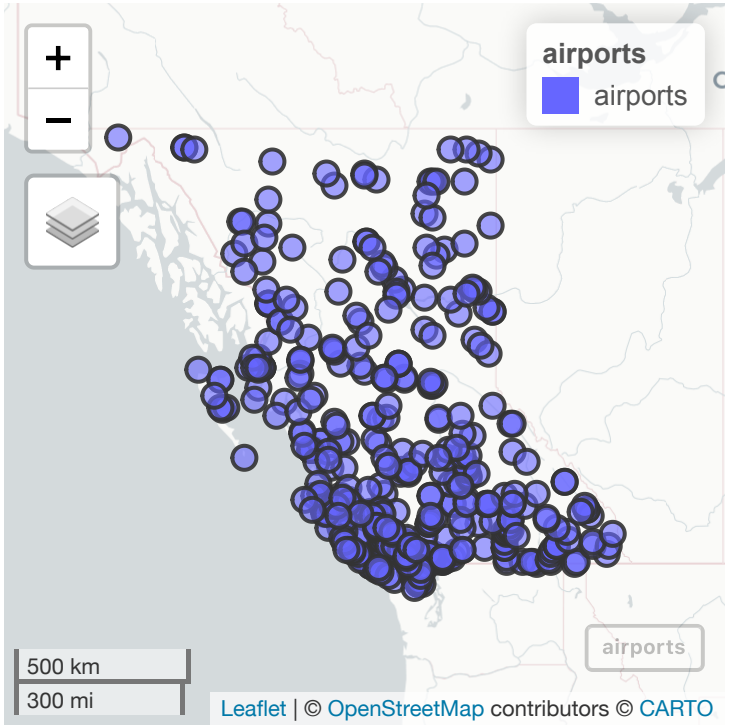
Geocomputation with R, fig 2.2: https://geocompr.robinlovelace.net

Reading and previewing spatial data

```
library(sf)
library(mapview)
```

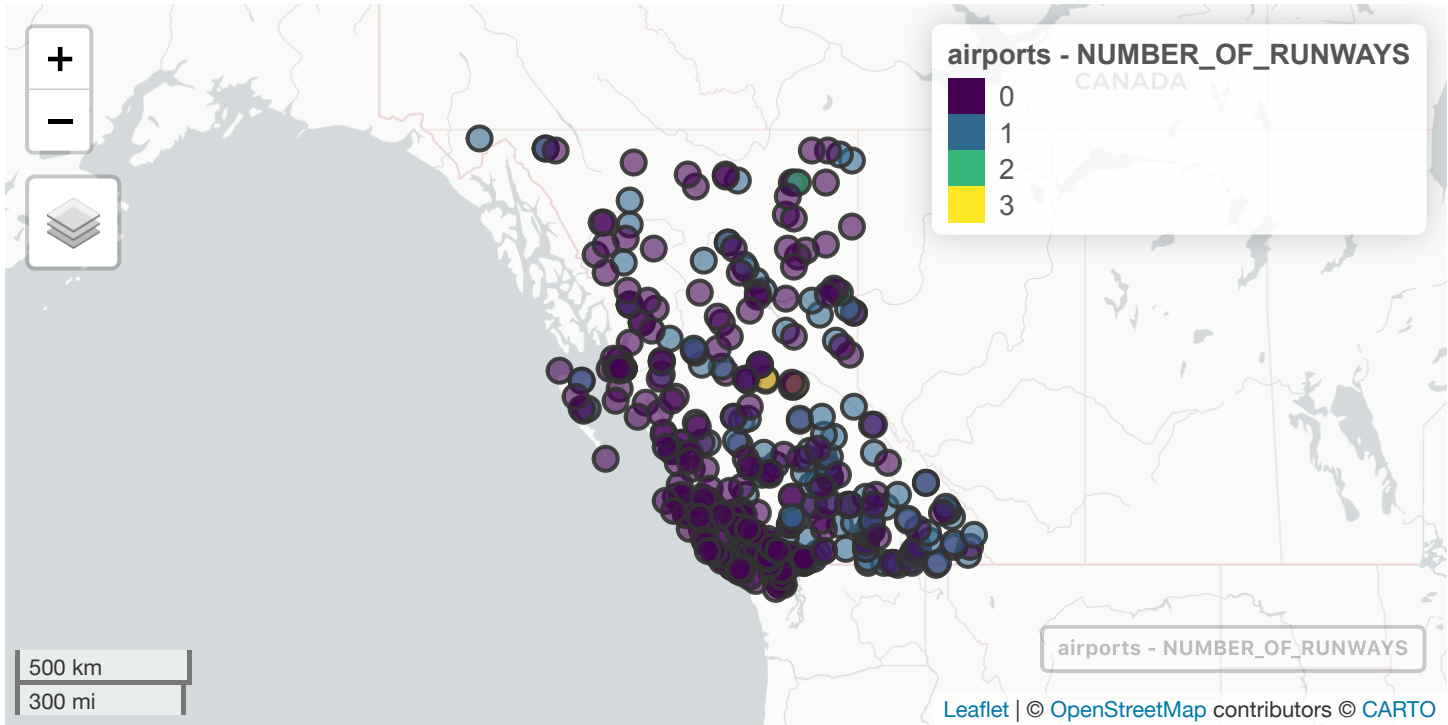
```
airports <-
st_read("data/20191106_Day_2_AM_Vector/bc_air
 quiet = TRUE)
```

```
mapview(airports)
```



mapview

```
mapview(airports, zcol = "NUMBER_OF_RUNWAYS")
```



Structure of an `sf` object

```
airports
#> Simple feature collection with 455 features and 5 fields
#> geometry type:  POINT
#> dimension:      XY
#> bbox:           xmin: 406543.7 ymin: 367957.6 xmax: 1796645 ymax: 1689146
#> epsg (SRID):    3005
#> proj4string:    +proj=aea +lat_1=50 +lat_2=58.5 +lat_0=45 +lon_0=-126 +x_0=1000000 +y_0=0
+ellps=GRS80 +towgs84=0,0,0,0,0,0,0 +units=m +no_defs
#> First 10 features:
#>                                                    AIRPORT_NAME IATA_CODE
#> 1                              Terrace (Northwest Regional) Airport      <NA>
#> 2                            Victoria Harbour (Camel Point) Heliport      <NA>
#> 3  Victoria Inner Harbour AirportÂ (Victoria Harbour Water Airport)       YWH
#> 4                            Victoria Harbour (Shoal Point) Heliport      <NA>
#> 5                         Victoria (Royal Jubilee Hospital) Heliport      <NA>
#> 6                              Victoria (General Hospital) Heliport      <NA>
#> 7                                    Victoria (BC Hydro) Heliport      <NA>
#> 8                             San Juan Point (Coast Guard) Heliport      <NA>
#> 9                                  Shawnigan Lake Water Aerodrome      <NA>
#> 10                                 Victoria International Airport       YYJ
#>          LOCALITY ELEVATION NUMBER_OF_RUNWAYS                       geom
#> 1         Terrace    217.32                2 POINT (833323.9 1054950)
```

Key features of an `sf` object

```
st_geometry(airports)
#> Geometry set for 455 features
#> geometry type:  POINT
#> dimension:      XY
#> bbox:           xmin: 406543.7 ymin: 367957.6 xmax: 1796645 ymax: 1689146
#> epsg (SRID):    3005
#> proj4string:    +proj=aea +lat_1=50 +lat_2=58.5 +lat_0=45 +lon_0=-126 +x_0=1000000 +y_0=0
+ellps=GRS80 +towgs84=0,0,0,0,0,0,0 +units=m +no_defs
#> First 5 geometries:
#> POINT (833323.9 1054950)
#> POINT (1193727 381604.1)
#> POINT (1194902 382257.7)
#> POINT (1193719 382179.3)
#> POINT (1198292 383563.6)

st_bbox(airports)
#>      xmin      ymin      xmax      ymax
#>  406543.7  367957.6 1796645.0 1689145.9

st_crs(airports)
#> Coordinate Reference System:
```

An `sf` object is a `data.frame`

```
class(airports)
#> [1] "sf"         "data.frame"
is.data.frame(airports)
#> [1] TRUE
summary(airports)
#>                                                AIRPORT_NAME   IATA_CODE
#>  Landing Strip                                        : 25   CJH    :  1
#>  Helipad                                              :  2   CXH    :  1
#>  100 Mile House Airport                               :  1   DUQ    :  1
#>  Abbotsford (Regional Hospital & Cancer Centre) Heliport:  1   WPL    :  1
#>  Abbotsford (Sumas Mountain) Heliport                 :  1   XQU    :  1
#>  Abbotsford (Teck) Heliport                           :  1   (Other): 78
#>  (Other)                                              :424   NA's   :372
#>            LOCALITY     ELEVATION      NUMBER_OF_RUNWAYS             geom
#>  Nanaimo       :  9   Min.   :   0.0   Min.   :0.0000    POINT        :455
#>  Campbell River:  7   1st Qu.:   0.0   1st Qu.:0.0000    epsg:3005    :  0
#>  Tsay Key Dene :  7   Median :   6.4   Median :0.0000    +proj=aea ...:  0
#>  Abbotsford    :  5   Mean   : 194.4   Mean   :0.3385
#>  Bute Inlet    :  5   3rd Qu.: 307.1   3rd Qu.:1.0000
#>  Fort Nelson   :  5   Max.   :1277.4   Max.   :3.0000
#>  (Other)       :417
```

We can turn it into a normal `data.frame`

```
airports_df <- st_drop_geometry(airports)
head(airports_df)
#>                                                                AIRPORT_NAME IATA_CODE   LOCALITY
#> 1                                 Terrace (Northwest Regional) Airport      <NA>    Terrace
#> 2                               Victoria Harbour (Camel Point) Heliport      <NA>   Victoria
#> 3 Victoria Inner Harbour AirportÂ (Victoria Harbour Water Airport)       YWH   Victoria
#> 4                               Victoria Harbour (Shoal Point) Heliport      <NA>   Victoria
#> 5                           Victoria (Royal Jubilee Hospital) Heliport      <NA>    Saanich
#> 6                                 Victoria (General Hospital) Heliport      <NA> View Royal
#>   ELEVATION NUMBER_OF_RUNWAYS
#> 1    217.32                2
#> 2      4.57                0
#> 3      0.00                0
#> 4      3.05                0
#> 5     15.55                0
#> 6     15.85                0
```

Coordinate Reference Systems

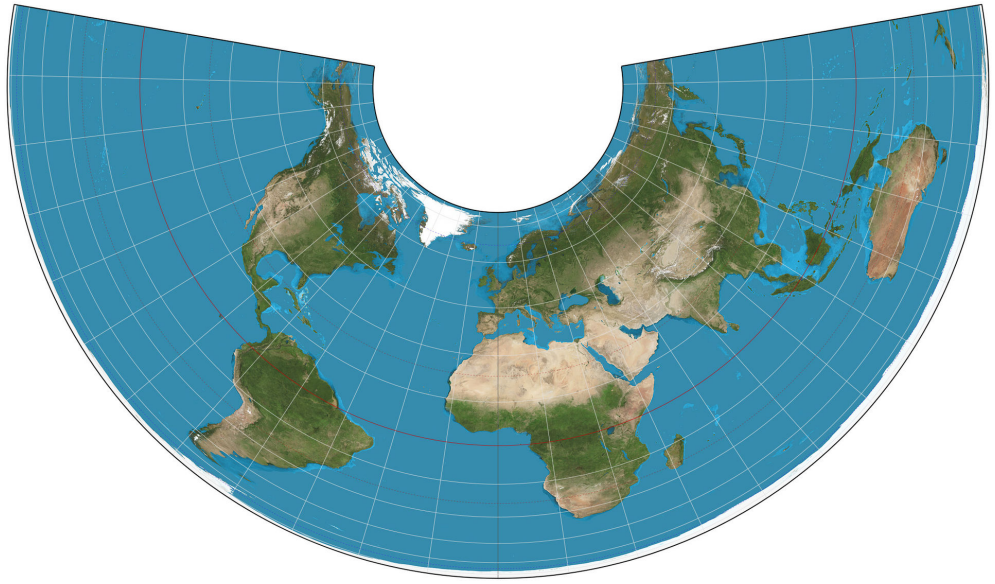
Geographic

- spherical or ellipsoidal surface; ellipsoid defined by the *datum*
- lat/long, measured in angular distances (degrees/radians)

https://en.wikipedia.org/wiki/Geographic_coordinate_system

Projected

- Cartesian coordinates on a flat plane
- origin, x and y axes, linear unit of measurement (e.g., m)



https://en.wikipedia.org/wiki/Albers_projection

CRSs in R

```
st_crs(airports)
#> Coordinate Reference System:
#>   EPSG: 3005
#>   proj4string: "+proj=aea +lat_1=50 +lat_2=58.5 +lat_0=45 +lon_0=-126 +x_0=1000000 +y_0=0
+ellps=GRS80 +towgs84=0,0,0,0,0,0,0 +units=m +no_defs"
```

BC Albers - B.C.'s standard projection

- Equal Area conic
- Centre: c(-126, 54) https://epsg.io/3005

Proj4 String

```
st_crs(3005)$proj4string
#> [1] "+proj=aea +lat_1=50 +lat_2=58.5 +lat_0=45 +lon_0=-126 +x_0=1000000 +y_0=0 +ellps=GRS80
+towgs84=0,0,0,0,0,0,0 +units=m +no_defs"
```

- projection (`proj`)
- standard parallels (`lat_1`, `lat_2`)
- origin (`lat_0`, `lon_0`, `x_0`, `y_0`)
- ellipsis (`ellps`)
- conversion to WGS84 (`towgs84`)
- `units`
- `no_defs` - no defaults are read from the defaults files
- `datum`

```
# one of "have_datum_files", "proj",
"ellps",
# "datum", "units" or "prime_meridians"
st_proj_info("proj") %>% head()
#>     name                    description
#> 1    aea             Albers Equal Area
#> 2   aeqd        Azimuthal Equidistant
#> 3   airy                         Airy
#> 4 aitoff                       Aitoff
#> 5   alsk Mod. Stereographic of Alaska
#> 6  apian              Apian Globular I
```

Your turn

Read in the electoral districts data in the `data` folder.

- What type of geometry is it?
- What is the CRS?
- What is the EPSG code?

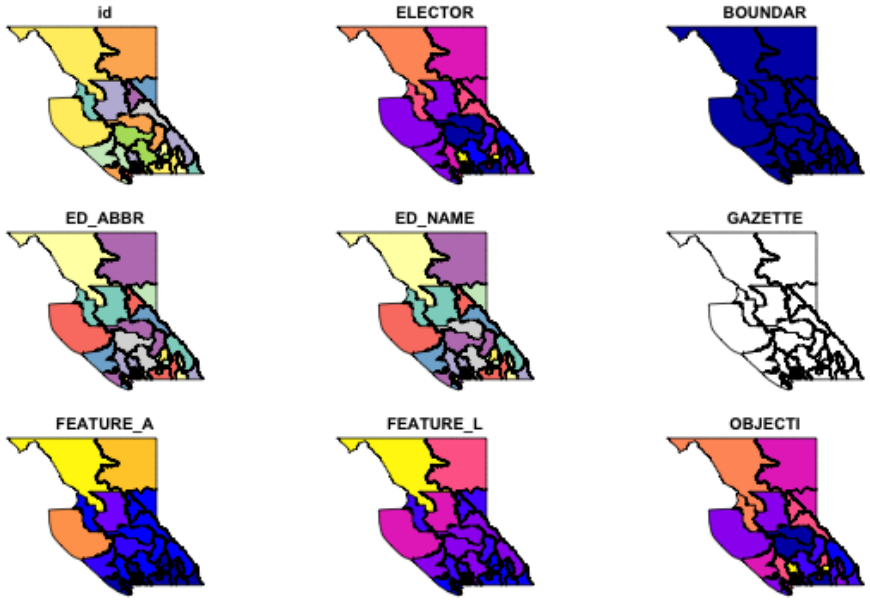
Solution

```
elec_bc <- read_sf("data/20191106_Day_2_AM_Vector/bc_electoral_districts.shp")
st_geometry(elec_bc)
#> Geometry set for 87 features
#> geometry type:  POLYGON
#> dimension:      XY
#> bbox:           xmin: -139.0624 ymin: 48.22483 xmax: -114.0535 ymax: 60
#> epsg (SRID):    4326
#> proj4string:    +proj=longlat +datum=WGS84 +no_defs
#> First 5 geometries:
#> POLYGON ((-122.1393 49.47379, -122.1453 49.4702...
#> POLYGON ((-122.0962 49.12295, -122.0803 49.1061...
#> POLYGON ((-122.3378 49.12709, -122.317 49.11858...
#> POLYGON ((-118.3136 49.81212, -118.3052 49.8085...
#> POLYGON ((-122.9676 49.24367, -122.9421 49.2363...
st_crs(elec_bc)
#> Coordinate Reference System:
#>   EPSG: 4326
#>   proj4string: "+proj=longlat +datum=WGS84 +no_defs"
```

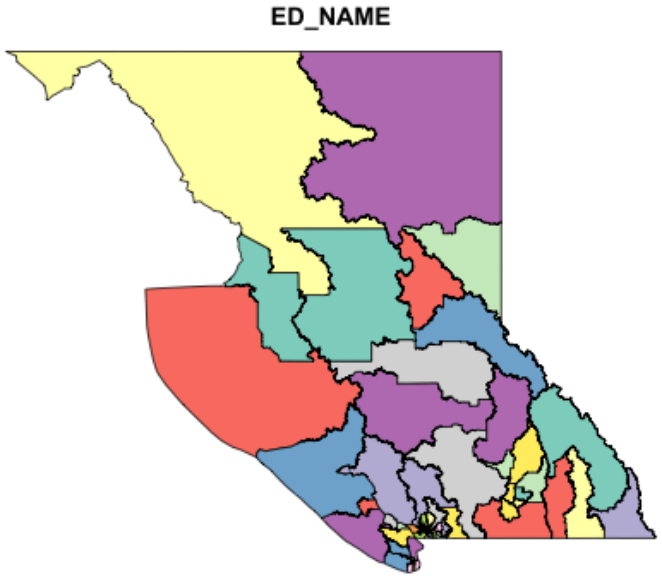
https://epsg.io/4326

Basic plotting

```
plot(elec_bc)
```



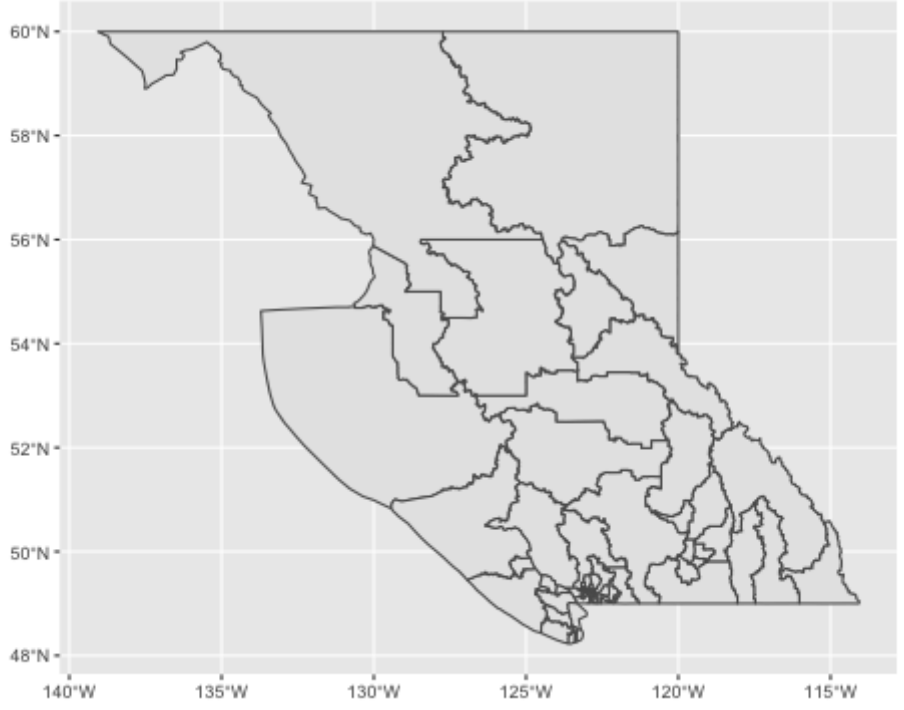

```
plot(elec_bc["ED_NAME"])
```



Use `ggplot2::geom_sf()` to make nice maps

```
library(ggplot2)
```

```
ggplot() + # leave this empty!
  geom_sf(data = elec_bc)
```

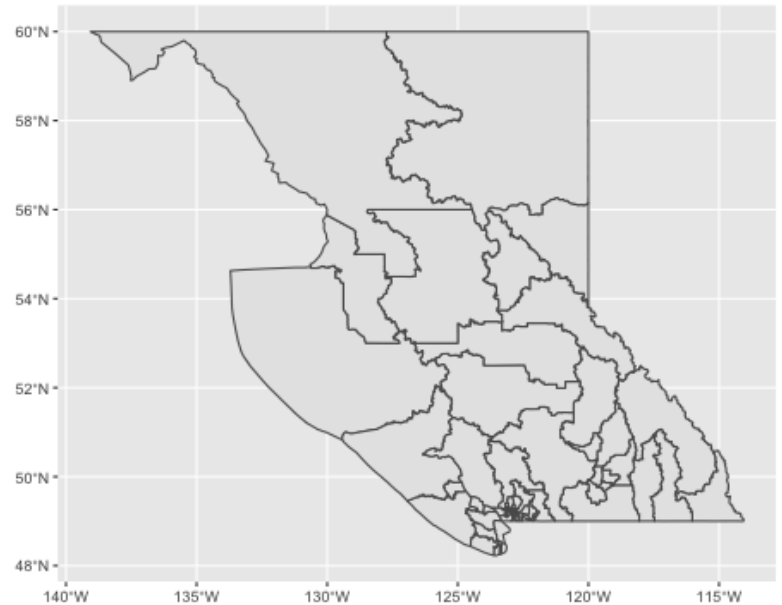


Transforming coordinate systems

```
elec_bc_albers <- st_transform(elec_bc, 3005)

# OR
albers <- "+proj=aea +lat_1=50 +lat_2=58.5 +lat_0=45 +lon_0=-126 +x_0=1000000 +y_0=0
+ellps=GRS80 +towgs84=0,0,0,0,0,0,0 +units=m +no_defs"
elec_bc_albers <- st_transform(elec_bc, albers)

# OR (very handy)

elec_bc_albers <- st_transform(elec_bc, st_crs(airports))

st_crs(elec_bc_albers)
#> Coordinate Reference System:
#>   EPSG: 3005
#>   proj4string: "+proj=aea +lat_1=50 +lat_2=58.5 +lat_0=45 +lon_0=-126 +x_0=1000000 +y_0=0
+ellps=GRS80 +towgs84=0,0,0,0,0,0,0 +units=m +no_defs"
```

WGS 84 (EPSG: 4326)

```
ggplot() +
  geom_sf(data = elec_bc)
```



BC Albers (EPSG: 3005)

```
elec_bc_albers <- st_transform(elec_bc,
3005)
ggplot() +
  geom_sf(data = elec_bc_albers)
```

Your turn

Load `"data/20191106_Day_2_AM_Vector/ski_resorts.csv"` as an `sf` object

facility_name	locality	latitude	longitude	elevation
Wapiti Ski Club	Elkford	50.02168	-114.9380	1417.36
Summit Lake Ski Area	Nakusp	50.14546	-117.6144	1082.95
Sasquatch Mountain Resort	Hemlock Valley	49.38011	-121.9354	1191.99
Apex Mountain Resort	Apex	49.39042	-119.9047	1881.92
Salmo Ski Hill	Salmo	49.18640	-117.3015	917.33
Red Mountain Resort	Rossland	49.10238	-117.8194	1376.44

Hints:

Load `"data/20191106_Day_2_AM_Vector/ski_resorts.csv"` as an `sf` object

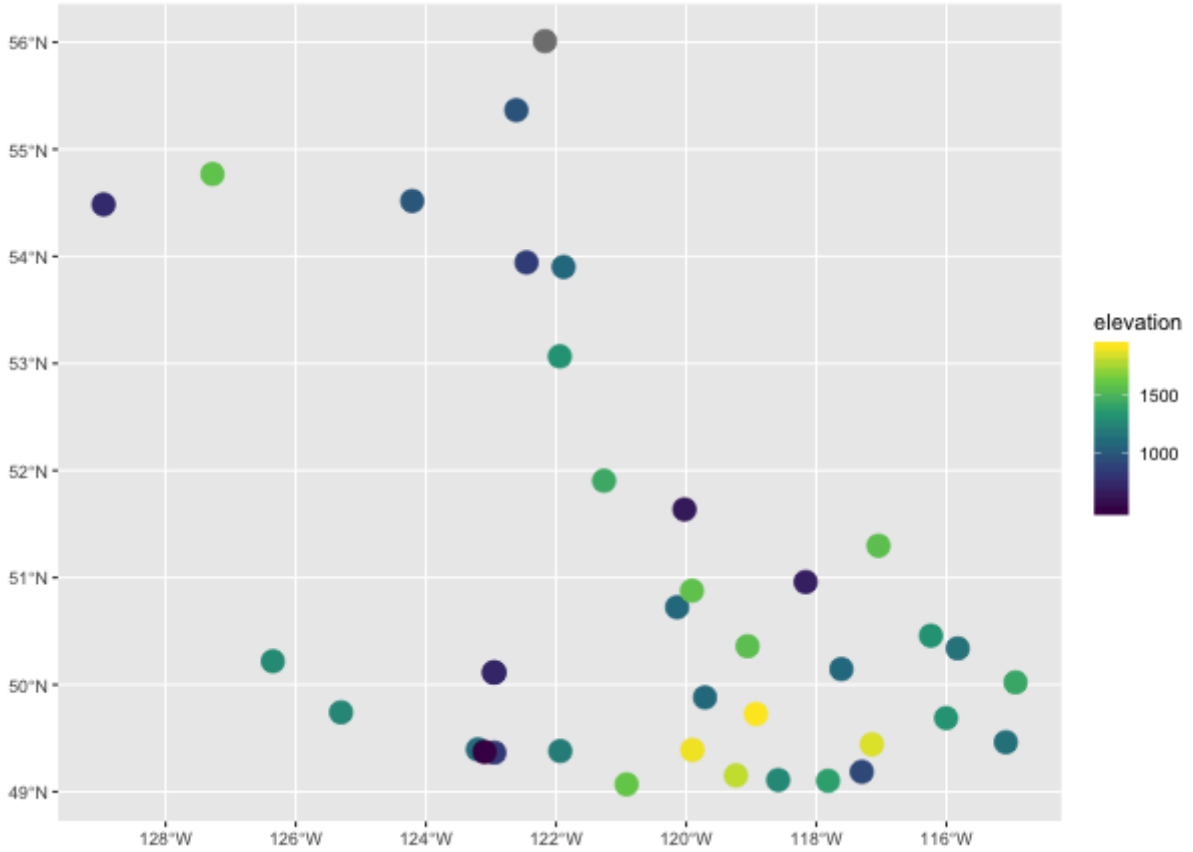
```
ski_resorts <- read.csv("data/20191106_Day_2_AM_Vector/ski_resorts.csv")
ski_resorts <- st_as_sf(ski_resorts, ...)
```

Solution

```
ski_resorts <- read.csv("data/20191106_Day_2_AM_Vector/ski_resorts.csv",
                        stringsAsFactors = FALSE)

ski_resorts <- st_as_sf(ski_resorts, coords = c("longitude", "latitude"),
                        crs = 4326)

head(ski_resorts)
#> Simple feature collection with 6 features and 3 fields
#> geometry type:  POINT
#> dimension:      XY
#> bbox:           xmin: -121.9354 ymin: 49.10238 xmax: -114.938 ymax: 50.14546
#> epsg (SRID):    4326
#> proj4string:    +proj=longlat +datum=WGS84 +no_defs
#>                facility_name       locality elevation                    geometry
#> 1           Wapiti Ski Club        Elkford   1417.36  POINT (-114.938 50.02168)
#> 2      Summit Lake Ski Area         Nakusp   1082.95 POINT (-117.6144 50.14546)
#> 3 Sasquatch Mountain Resort Hemlock Valley   1191.99 POINT (-121.9354 49.38011)
#> 4      Apex Mountain Resort           Apex   1881.92 POINT (-119.9047 49.39042)
#> 5            Salmo Ski Hill          Salmo    917.33  POINT (-117.3015 49.1864)
#> 6       Red Mountain Resort       Rossland   1376.44 POINT (-117.8194 49.10238)
```

Use `ggplot2` aesthetics to style maps

```
ggplot() +
  geom_sf(data = ski_resorts, aes(colour = elevation), size = 5)
```



- **`bcdc_browse()`**
  - Open the catalogue in your default browser
- **`bcdc_search()`**
  - Search records in the catalogue
- **`bcdc_search_facets()`**
  - List catalogue facet search options
- **`bcdc_get_record()`**
  - Print a catalogue record
- **`bcdc_get_data()`**
  - Get catalogue data
- **`bcdc_query_geodata()`**
  - Get & query catalogue geospatial data available through a Web Service

https://bcgov.github.io/bcdata

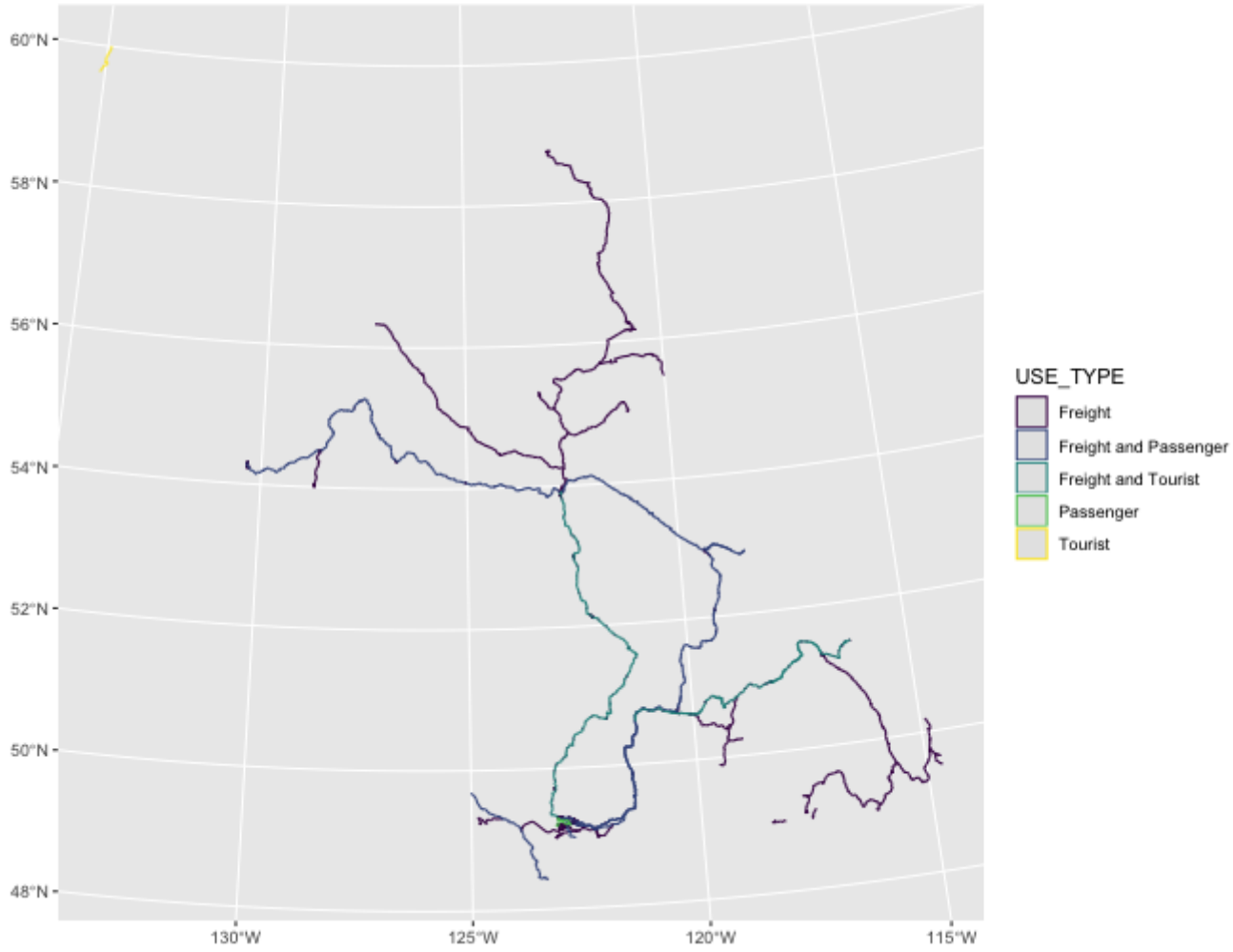
Get data with `bcdata`

https://catalogue.data.gov.bc.ca/dataset/railway-track-line

```
library(bcdata)
```

```
railways <- bcdc_get_data("railway-track-line",
                          resource = "bf30d34e-1f6b-4034-a35c-1cf7c9707ae7")
# OR: railways <- bcdc_get_data("WHSE_BASEMAPPING.GBA_RAILWAY_TRACKS_SP")
railways
#> Simple feature collection with 9502 features and 47 fields
#> geometry type:  LINESTRING
#> dimension:      XY
#> bbox:           xmin: 484759.6 ymin: 382665.4 xmax: 1816901 ymax: 1699824
#> epsg (SRID):    3005
#> proj4string:    +proj=aea +lat_1=50 +lat_2=58.5 +lat_0=45 +lon_0=-126 +x_0=1000000 +y_0=0
+ellps=GRS80 +towgs84=0,0,0,0,0,0,0 +units=m +no_defs
#> # A tibble: 9,502 x 48
#>    id    RAILWAY_TRACK_ID NID   TRACK_SEGMENT_ID TRACK_NAME TRACK_CLASSIFIC… REGULATOR
#>  * <chr>            <int> <chr> <chr>            <chr>      <chr>            <chr>
#>  1 WHSE…                1 b565… bf3a6f46cbfa456… None       Main             Unknown
#>  2 WHSE…                2 c64d… 0ded61aee01342f… None       Spur             Unknown
#>  3 WHSE…                3 42bd… 8895c9c2f0634b3… None       Yard             Unknown
#>  4 WHSE…                4 702c… d5d6d85bb4f34d0… None       Yard             Unknown
```

```
ggplot() +
  geom_sf(data = railways, aes(colour = USE_TYPE))
```



Using `dplyr` with `sf` objects

select: geometry is "sticky"

```
railways <- select(railways, TRACK_NAME, TRACK_CLASSIFICATION,
                   USE_TYPE)
railways
#> Simple feature collection with 9502 features and 3 fields
#> geometry type:  LINESTRING
#> dimension:      XY
#> bbox:           xmin: 484759.6 ymin: 382665.4 xmax: 1816901 ymax: 1699824
#> epsg (SRID):    3005
#> proj4string:    +proj=aea +lat_1=50 +lat_2=58.5 +lat_0=45 +lon_0=-126 +x_0=1000000 +y_0=0
+ellps=GRS80 +towgs84=0,0,0,0,0,0,0 +units=m +no_defs
#> # A tibble: 9,502 x 4
#>    TRACK_NAME    TRACK_CLASSIFICA… USE_TYPE                                   geometry
#>  * <chr>         <chr>             <chr>                                <LINESTRING [m]>
#>  1 None          Main              Freight an… (1628797 738603.1, 1628695 738581.5, 16285…
#>  2 None          Spur              Freight              (1216277 1002329, 1216278 1002298)
#>  3 None          Yard              Freight     (1546377 695113.9, 1546386 695108.3, 15463…
#>  4 None          Yard              Freight            (1086735 469866.2, 1086730 469841.3)
#>  5 None          Spur              Freight     (1600555 514951.1, 1600532 514951.1, 16005…
#>  6 Seymour Indu… Spur              Freight     (1217505 480938.4, 1217508 480840.9, 12175…
#>  7 None          Spur              Freight     (1234872 924754.1, 1234832 924741, 1234790…
```

Computing geometric measurements

- st_area
- st_length
- st_distance

```
st_length(railways)
#> Units: [m]
#>    [1]  1444.825965    30.677287  1026.072479    25.267932    63.413200   227.068985
#>    [7]   708.853989   152.812110    23.758005    92.789245   132.021079    91.751755
#>   [13]  2305.398646    88.262913  1480.814391   717.651348   289.681294   190.459264
#>   [19]   845.372695    51.905598  3270.962231    61.101040   501.470018    59.439442
#>   [25]    83.053247   127.900575   135.379906    24.895304   821.350608   227.444124
#>   [31]   596.857526   534.922963  1525.388725    25.901309   240.559215   639.975559
#>   [37]   241.651804   243.592625   286.004295   113.662070   162.729893  2364.854787
#>   [43]    24.718281    54.845324   444.628105   387.576207   505.555243   973.320531
#>   [49]    68.315871   234.435553   261.037718   648.467173    89.221108    39.583242
#>   [55]   864.139219  2224.377705   118.001663     8.116982  1496.220335   210.660599
#>   [61]   642.962516    27.822204   160.653241   106.115495   543.994731   486.775995
#>   [67]    57.651497   183.319516   736.334062   186.019459   204.948559    29.595415
#>   [73]    95.371738    31.514464 27132.763662    38.161366   320.282342   545.846875
#>   [79]    90.579576   129.140658    34.889519     9.859432   105.441386   247.473027
#>   [85]  2345.674919  1081.734078   143.092244   219.981287    29.077711    33.151017
#>   [91]   233.369560  2065.448386    24.485213    95.576079   212.940844   743.814772
```

Your turn

Create an `sf` object from railways containing only track segments longer than 5 km

Solution

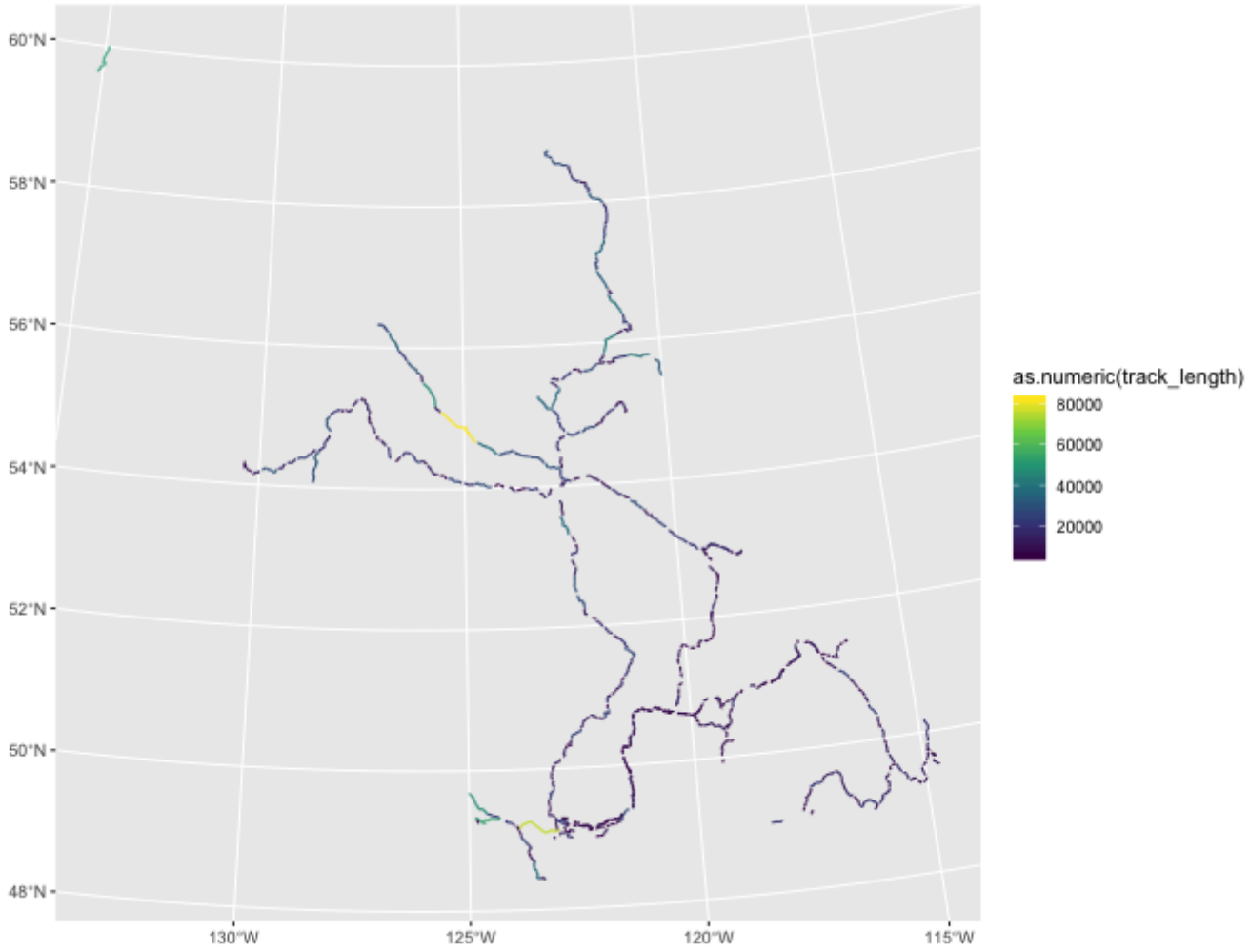
Note geometric measures return `units`

```
long_railways <- filter(railways, as.numeric(track_length) > 5000)
```

OR

```
library(units)
long_railways <- filter(railways, track_length > as_units(5000, "m"))
```

```
ggplot() +
  geom_sf(data = long_railways, aes(colour = as.numeric(track_length)))
```

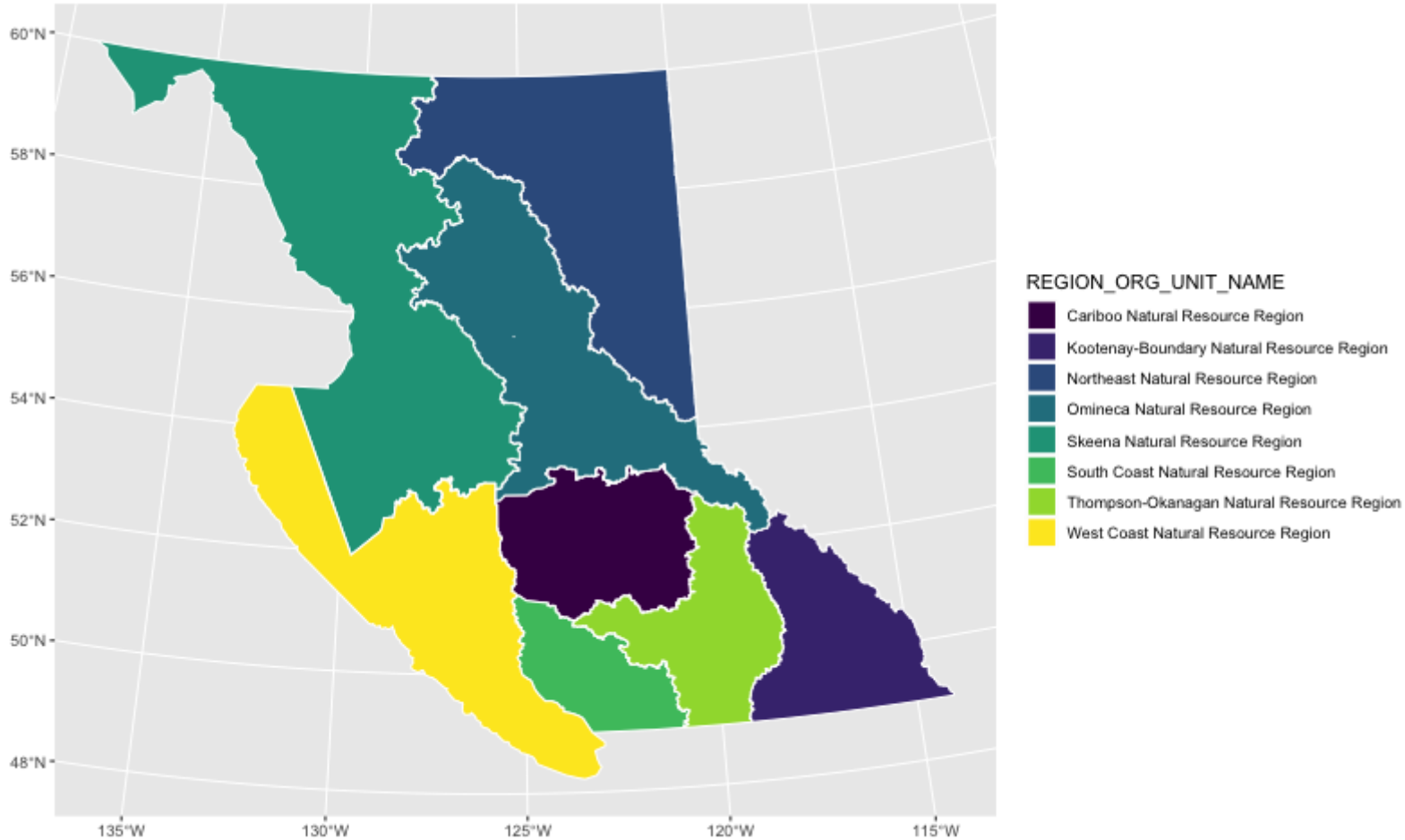


dplyr::summarise

Using summarize() - Calculate `nr_regions`

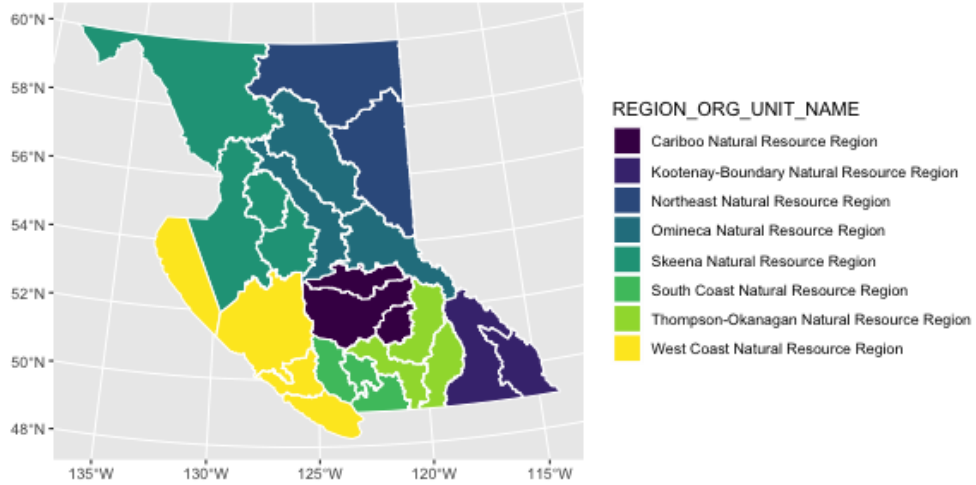
```
nr_district <- bcdc_get_data('natural-resource-nr-district')


nr_region <- nr_district %>%
  group_by(REGION_ORG_UNIT_NAME) %>%
  summarise() # << defaults to union
```

```
ggplot() +
  geom_sf(data = nr_region, colour = "white",
          aes(fill = REGION_ORG_UNIT_NAME))
```



```
nr_region_multi <- nr_district %>%
  group_by(REGION_ORG_UNIT_NAME) %>%
  summarise(do_union = FALSE) # <<--

nr_region_multi
#> Simple feature collection with 8 features
and 1 field
#> geometry type:  MULTIPOLYGON
#> dimension:      XY
#> bbox:           xmin: 273366.8 ymin:
359771.8 xmax: 1870587 ymax: 1735721
#> epsg (SRID):    3005
#> proj4string:    +proj=aea +lat_1=50
+lat_2=58.5 +lat_0=45 +lon_0=-126
+x_0=1000000 +y_0=0 +ellps=GRS80
+towgs84=0,0,0,0,0,0,0 +units=m +no_defs
#> # A tibble: 8 x 2
#>   REGION_ORG_UNIT_NAME
geometry
#> * <chr>
<MULTIPOLYGON [m]>
#> 1 Cariboo Natural Resource Regi…
(((1393203 890457.6, 1393134 890528.3,
1393088 890697.9,…
#> 2 Kootenay-Boundary Natural Res…
(((1749868 497227.4, 1749860 497306.6,
1749926 497369.4,…
```

```
ggplot() +
  geom_sf(data = nr_region_multi, colour =
"white",
          aes(fill = REGION_ORG_UNIT_NAME))
```



Your turn

Find the total area of each NR region

Solution

```
nr_region %>%
  mutate(region_area = st_area(geometry))
#> Simple feature collection with 8 features and 2 fields
#> geometry type:  POLYGON
#> dimension:      XY
#> bbox:           xmin: 273366.8 ymin: 359771.8 xmax: 1870587 ymax: 1735721
#> epsg (SRID):    3005
#> proj4string:    +proj=aea +lat_1=50 +lat_2=58.5 +lat_0=45 +lon_0=-126 +x_0=1000000 +y_0=0
+ellps=GRS80 +towgs84=0,0,0,0,0,0,0 +units=m +no_defs
#> # A tibble: 8 x 3
#>   REGION_ORG_UNIT_NAME                                              geometry region_area
#> * <chr>                                                        <POLYGON [m]>       [m^2]
#> 1 Cariboo Natural Resource … ((1381632 807771.5, 1381787 807659.5, 1381916 80…  825291483…
#> 2 Kootenay-Boundary Natural… ((1707424 724613.8, 1708109 725049.3, 1708387 72…  823399633…
#> 3 Northeast Natural Resourc… ((1352378 1480743, 1390137 1036196, 1390009 1036… 1753972117…
#> 4 Omineca Natural Resource … ((1235405 1136771, 1235426 1136708, 1235387 1136… 1582201306…
#> 5 Skeena Natural Resource R… ((973855.3 1168864, 973837.6 1168731, 973915.2 1… 2632146164…
#> 6 South Coast Natural Resou… ((1269941 574210, 1270064 574122.1, 1270195 5739…  461543065…
#> 7 Thompson-Okanagan Natural… ((1409645 458889.8, 1398207 458006.1, 1389072 45…  751265006…
#> 8 West Coast Natural Resour… ((1091120 523587.2, 1094036 519727.6, 1095742 51… 1567644712…
```


bcdata

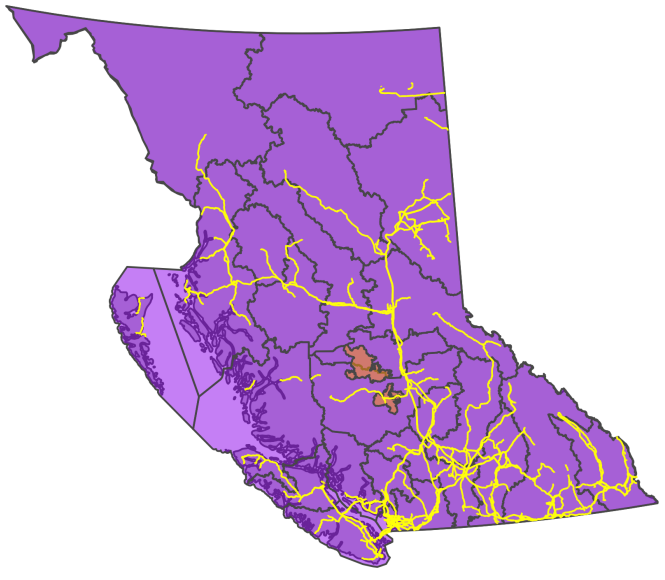
Grabbing our data

```
library(bcdata)

nr_district <- bcdc_get_data('natural-resource-nr-district')
lines <- bcdc_get_data('bc-transmission-lines')

fires_2017 <- bcdc_query_geodata('fire-perimeters-historical') %>%
  filter(FIRE_YEAR == 2017) %>%
  collect()

big_fires <- fires_2017 %>%
  filter(FIRE_NUMBER %in% c('C10784', 'C50647'))

bc <- bcdc_query_geodata('7-5m-provinces-and-states-the-atlas-of-canada-base-maps-for-bc') %>%
  filter(ENGLISH_NAME == 'British Columbia') %>%
  collect()
```

Multi layer spatial plots with `ggplot2`

Natural Resource Districts, Power Lines and Big Fires

```
library(ggplot2)

m <- ggplot() +
  geom_sf(data = bc, fill = "grey80") +
  geom_sf(data = nr_district, fill =
"purple", alpha = 0.5) +
  geom_sf(data = big_fires, fill = "orange",
alpha = 0.5) +
  geom_sf(data = lines, colour = "yellow")
```



Your turn

- Using `ggplot2` plot only *two* layers
  1. `big_fires`
  2. `fire_districts` <- only "Stuart Nechako", "Quesnel", "Cariboo-Chilcotin", and "Prince George" natural resource districts (`nr_district`)

Solution

```
fire_districts <- nr_district %>%
  filter(DISTRICT_NAME %in% c("Stuart Nechako Natural Resource District",
                              "Quesnel Natural Resource District",
                              "Cariboo-Chilcotin Natural Resource District",
                              "Prince George Natural Resource District"
                              ))
```

```
ggplot() +
  geom_sf(data = fire_districts, fill = "purple", alpha = 0.5) +
  geom_sf(data = big_fires, fill = "orange", alpha = 0.5)
```

Spatial Operations

Geometric Operations

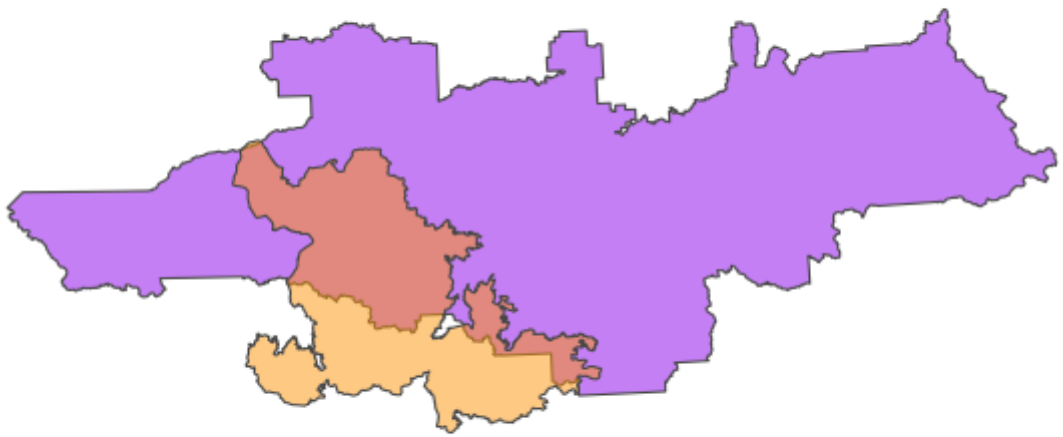
- st_union
- st_intersection
- st_difference
- st_sym_difference

Quesnel Natural Resource District and C10784

```
biggest_fire <- big_fires %>%
  filter(FIRE_NUMBER == "C10784")

quesnel_district <- nr_district %>%
  filter(DISTRICT_NAME == "Quesnel Natural Resource District")
```

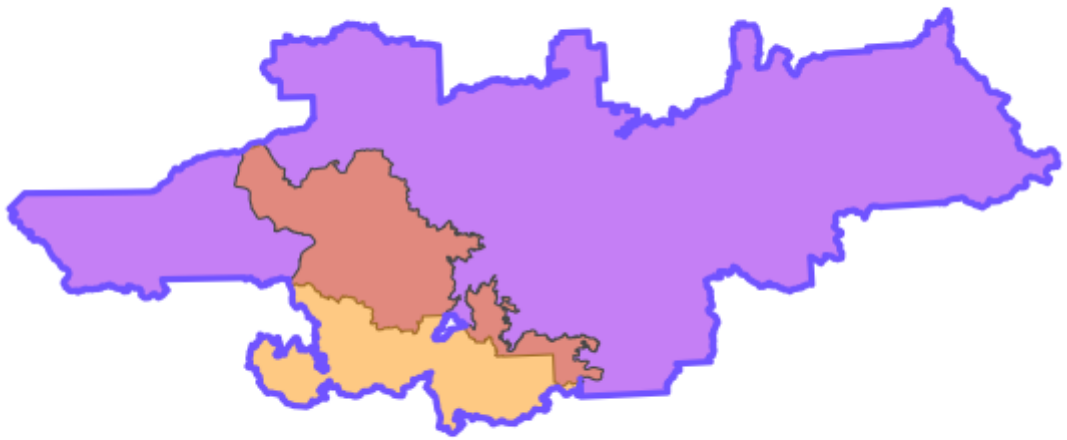
Quesnel Natural Resource District and C10784

```
p <- ggplot() +
  geom_sf(data = quesnel_district, fill = "purple", alpha = 0.5) +
  geom_sf(data = biggest_fire, fill = "orange", alpha = 0.5)
p
```



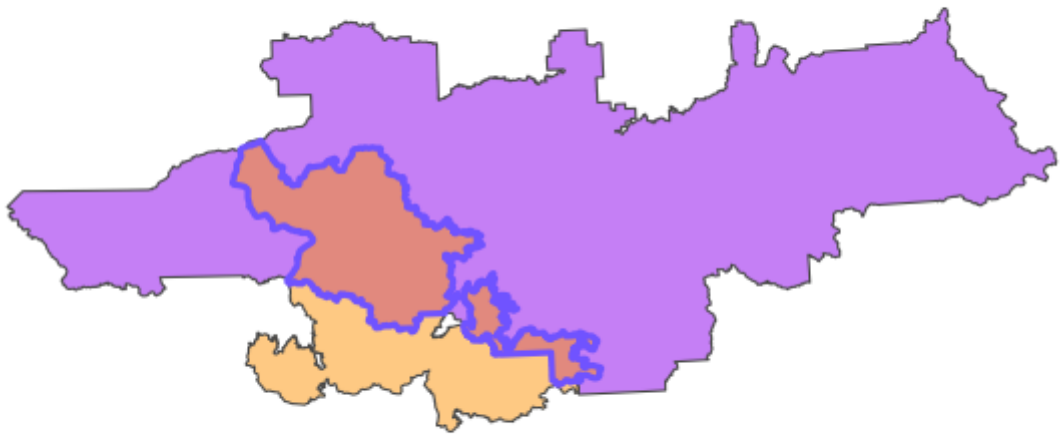
st_union

```
library(sf)
unionized <- st_union(quesnel_district, biggest_fire)
p + geom_sf(data = unionized, size = 1.5, fill = NA, colour = "lightslateblue")
```



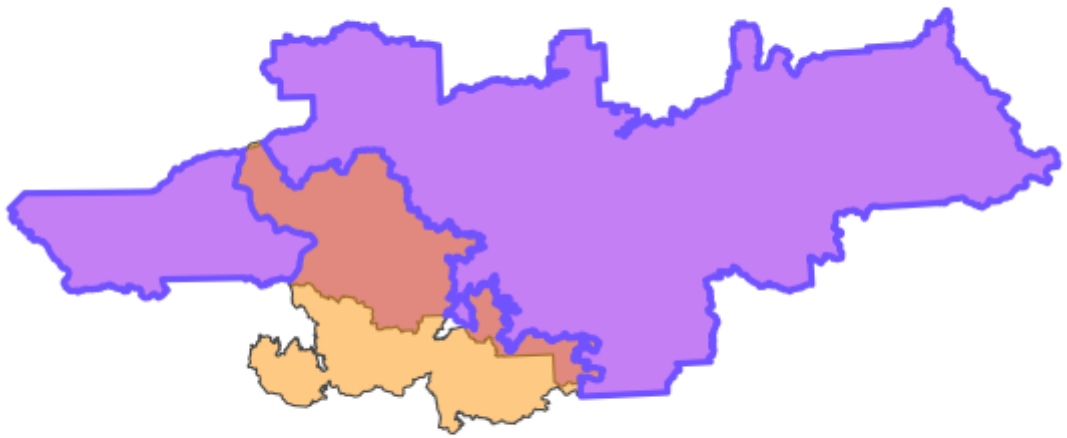
st_intersection

```
intersected <- st_intersection(quesnel_district, biggest_fire)
p + geom_sf(data = intersected, size = 1.5, fill = NA, colour = "lightslateblue")
```



st_difference

```
differenced <- st_difference(quesnel_district, biggest_fire)
p + geom_sf(data = differenced, size = 1.5, fill = NA, colour = "lightslateblue")
```



Your turn

Create a geometry of all transmission lines that intersect the four natural resource districts (`fire_districts`) and create a multi-layer plot

Solution

```
fire_lines <- lines %>%
  st_intersection(fire_districts)

ggplot() +
  geom_sf(data = fire_districts, fill = "purple", alpha = 0.5) +
  geom_sf(data = big_fires, fill = "orange", alpha = 0.5) +
  geom_sf(data = fire_lines, colour = "yellow")
```

Geometry Predicates

- `st_intersects`: touch or overlap
- `st_disjoint`: !intersects
- `st_touches`: touch
- `st_crosses`: cross (don't touch)
- `st_within`: within
- `st_contains`: contains
- `st_overlaps`: overlaps
- `st_covers`: cover
- `st_covered_by`: covered by
- `st_equals`: equals
- `st_equals_exact`: equals, with some fuzz

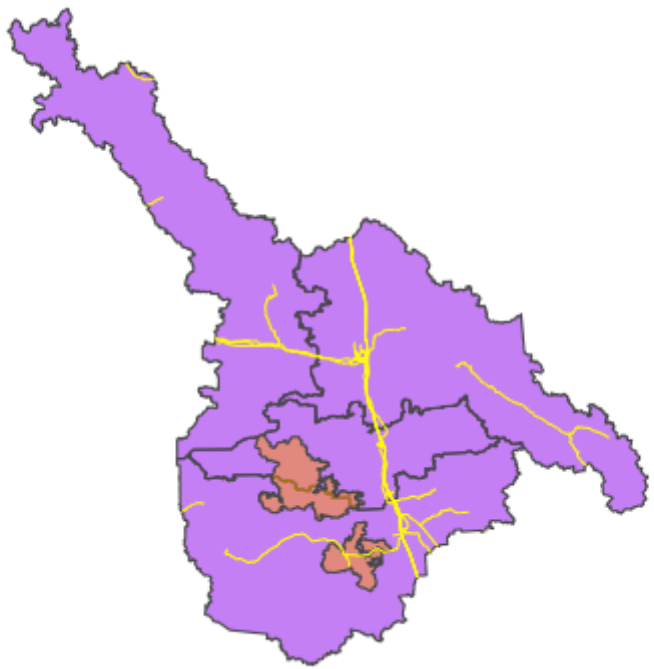
Only for subsetting

Based on PostGIS

Very similar to ESRI naming

Usage

```
p2 <- ggplot() +
  geom_sf(data = fire_districts, fill = "purple", alpha = 0.5) +
  geom_sf(data = big_fires, fill = "orange", alpha = 0.5) +
  geom_sf(data = fire_lines, colour = "yellow")
p2
```

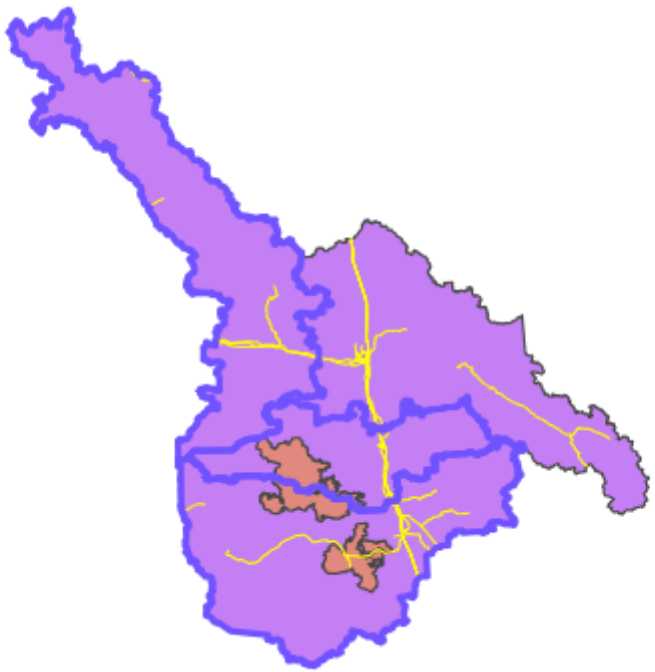


Does any district intersect this fire?

```
st_intersects(fire_districts, big_fires, sparse = FALSE)
#>       [,1]  [,2]
#> [1,]  TRUE  TRUE
#> [2,]  TRUE FALSE
#> [3,]  TRUE FALSE
#> [4,] FALSE FALSE
fire_districts[big_fires, , op = st_intersects]
#> Simple feature collection with 3 features and 12 fields
#> geometry type:  MULTIPOLYGON
#> dimension:      XY
#> bbox:           xmin: 831051 ymin: 653960.8 xmax: 1393203 ymax: 1368536
#> epsg (SRID):    3005
#> proj4string:    +proj=aea +lat_1=50 +lat_2=58.5 +lat_0=45 +lon_0=-126 +x_0=1000000 +y_0=0
+ellps=GRS80 +towgs84=0,0,0,0,0,0,0 +units=m +no_defs
#> # A tibble: 3 x 13
#>   id    DISTRICT_NAME ORG_UNIT ORG_UNIT_NAME REGION_ORG_UNIT REGION_ORG_UNIT… FEATURE_CODE
#>   <chr> <chr>         <chr>    <chr>         <chr>           <chr>             <chr>
#> 1 WHSE… Cariboo-Chil… DCC      Cariboo-Chil… RCB             Cariboo Natural… FM90000020
#> 2 WHSE… Stuart Necha… DVA      Stuart Necha… ROM             Omineca Natural… FM90000020
#> 3 WHSE… Quesnel Natu… DQU      Quesnel Natu… RCB             Cariboo Natural… FM90000020
#> # … with 6 more variables: FEATURE_NAME <chr>, OBJECTID <int>, SE_ANNO_CAD_DATA <chr>,
#> #   FEATURE_AREA_SQM <dbl>, FEATURE_LENGTH_M <dbl>, geometry <MULTIPOLYGON [m]>
```

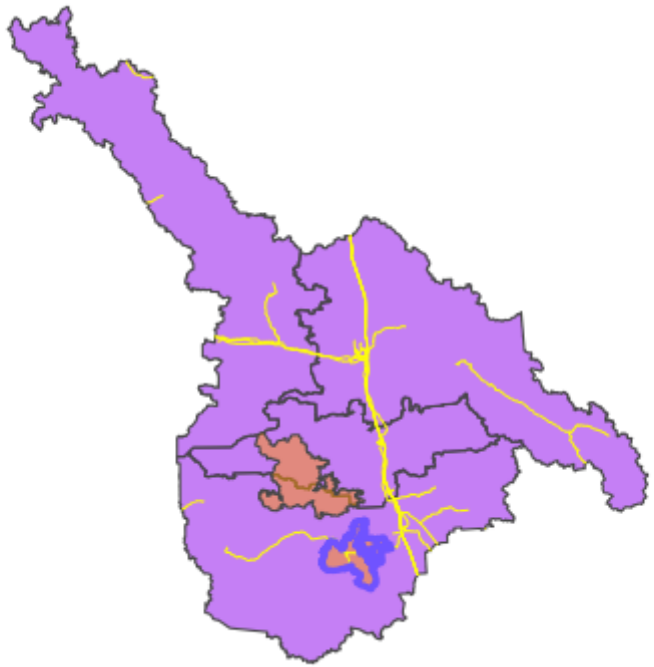
Does any district intersect this fire?

```
does_intersect <- fire_districts[big_fires, , op = st_intersects]
p2 + geom_sf(data = does_intersect, fill = NA, colour = "lightslateblue", size = 1.5)
```



Which polygon in intersected by a transmission line?

```
crosses_lines <- big_fires[fire_lines, , op = st_crosses]
p2 + geom_sf(data = crosses_lines, fill = NA, colour = "lightslateblue", size = 1.5)
```



Your turn

Which fires crossed transmission lines in 2017? Use geometry predicates to determine and ggplot2 to plot

Solution

```
all_crosses <- fires_2017[lines, , op = st_crosses]
ggplot() +
  geom_sf(data = bc, fill = "grey80") +
  geom_sf(data = nr_district, fill = "purple", alpha = 0.5) +
  geom_sf(data = fires_2017, fill = "orange", alpha = 0.5) +
  geom_sf(data = lines, colour = "yellow") +
  geom_sf(data = all_crosses, fill = "yellow", alpha = 0.5)
```

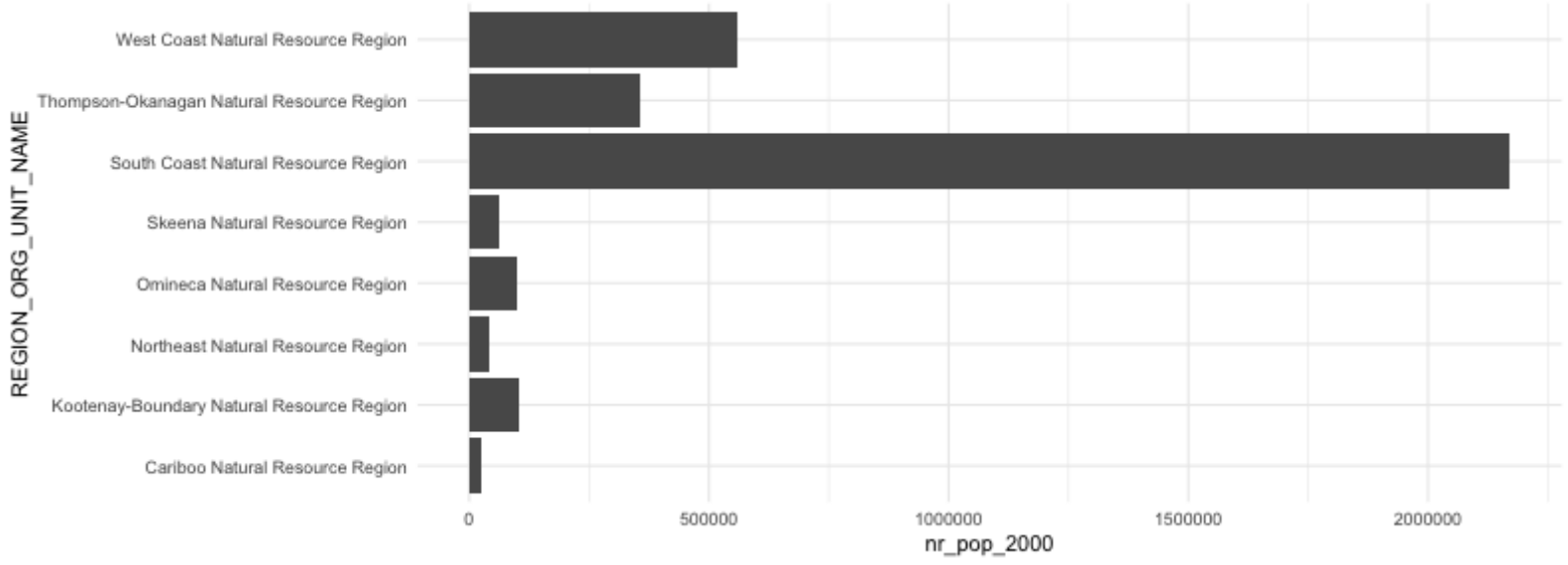
non spatial join

What is the mean population of cities that have a courthouse, grouped by municipality type?

```
courts <- bcdc_get_data('court-locations', resource = '23aa0b75-2715-4ccb-9a36-9a608450dc2d')
bc_cities <- bcdc_get_data('bc-major-cities-points-1-2-000-000-digital-baseline-mapping')

courts %>%
  left_join(bc_cities, by = c("City" = "NAME")) %>%
  group_by(LONG_TYPE) %>%
  summarise(mean_pop = mean(POP_2000))
#> # A tibble: 6 x 2
#>   LONG_TYPE              mean_pop
#>   <chr>                     <dbl>
#> 1 CITY                    108210.
#> 2 DISTRICT MUNICIPALITY     6561.
#> 3 TOWN                      5722.
#> 4 VILLAGE                   1310.
#> 5 VILLAGE (UNCHARTERED)     1352
#> 6 <NA>                        NA
```

st_join

```
cities_by_nr <- bc_cities %>%
  st_join(nr_district, join = st_intersects) %>%
  group_by(REGION_ORG_UNIT_NAME) %>%
  summarise(nr_pop_2000 = sum(POP_2000))

ggplot(cities_by_nr) +
  geom_col(aes(x = REGION_ORG_UNIT_NAME, y = nr_pop_2000)) +
  coord_flip() +
  theme_minimal()
```



Your turn

- Which natural resource district contains the greatest length of transmission lines?
- Hint - use the `lines` and `nr_district` datasets

Solution

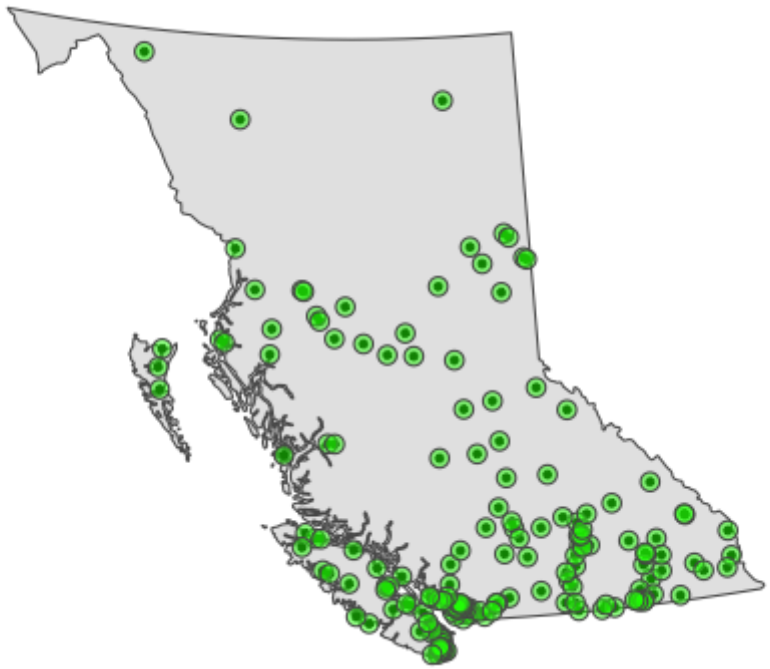
```
lines %>%
  st_join(nr_district) %>%
  mutate(length_lines = st_length(geometry)) %>%
  group_by(DISTRICT_NAME) %>%
  summarise(district_length_lines = sum(length_lines)) %>%
  arrange(desc(district_length_lines))
#> Simple feature collection with 24 features and 2 fields
#> geometry type:  MULTILINESTRING
#> dimension:      XY
#> bbox:           xmin: 591140.9 ymin: 377805.4 xmax: 1820212 ymax: 1547892
#> epsg (SRID):    3005
#> proj4string:    +proj=aea +lat_1=50 +lat_2=58.5 +lat_0=45 +lon_0=-126 +x_0=1000000 +y_0=0
+ellps=GRS80 +towgs84=0,0,0,0,0,0,0 +units=m +no_defs
#> # A tibble: 24 x 3
#>    DISTRICT_NAME           district_length_l…                                  geometry
#>    <chr>                                [m]                       <MULTILINESTRING [m]>
#>  1 Cascades Natural Resou…           3914503 ((1298455 676725.3, 1298456 676645.2, 12983…
#>  2 Chilliwack Natural Res…           3396678 ((1186980 430887.7, 1186806 430953.9), (118…
#>  3 Selkirk Natural Resour…           3223852 ((1530000 471296.8, 1529822 470871.3), (152…
#>  4 Okanagan Shuswap Natur…           2917928 ((1477591 467765.1, 1477578 467763.6, 14775…
#>  5 Prince George Natural …           2843975 ((1463955 827423.6, 1463939 827458.7, 14639…
#>  6 100 Mile House Natural…           2568603 ((1324241 660549.7, 1324274 660539.3), (132…
```

Manipulating Geometries

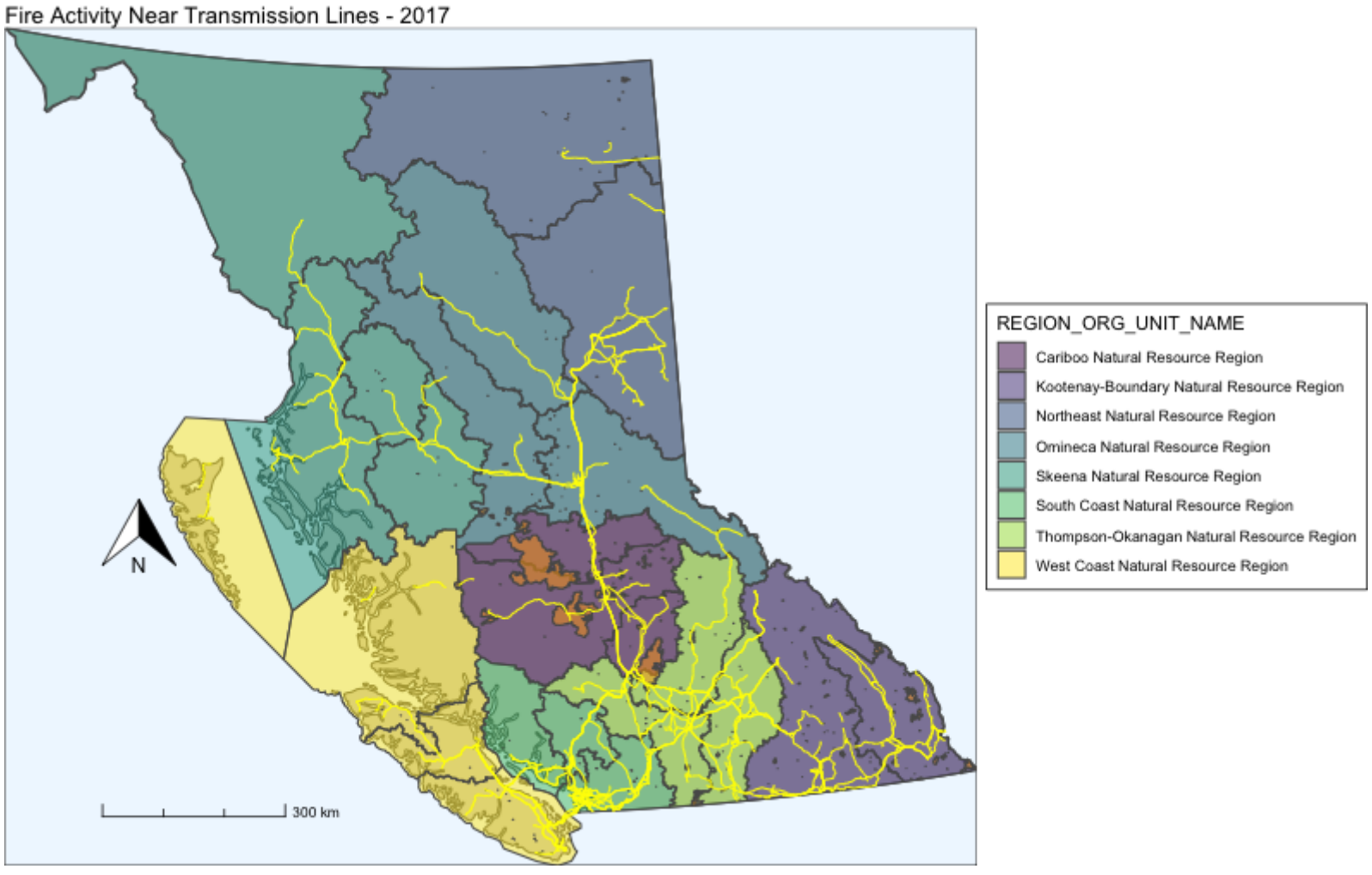
- st_line_merge
- st_segmentize
- st_voronoi
- st_centroid
- st_convex_hull
- st_triangulate
- st_polygonize
- st_simplify
- st_split
- st_buffer
- st_make_valid
- st_boundary

st_buffer

```
bc_cities_buffer <- st_buffer(bc_cities, dist = 20000) ## 20km
ggplot() +
  geom_sf(data = bc) +
  geom_sf(data = bc_cities) +
  geom_sf(data = bc_cities_buffer, fill = "green", alpha = 0.5)
```



Making nice plots



Getting Data

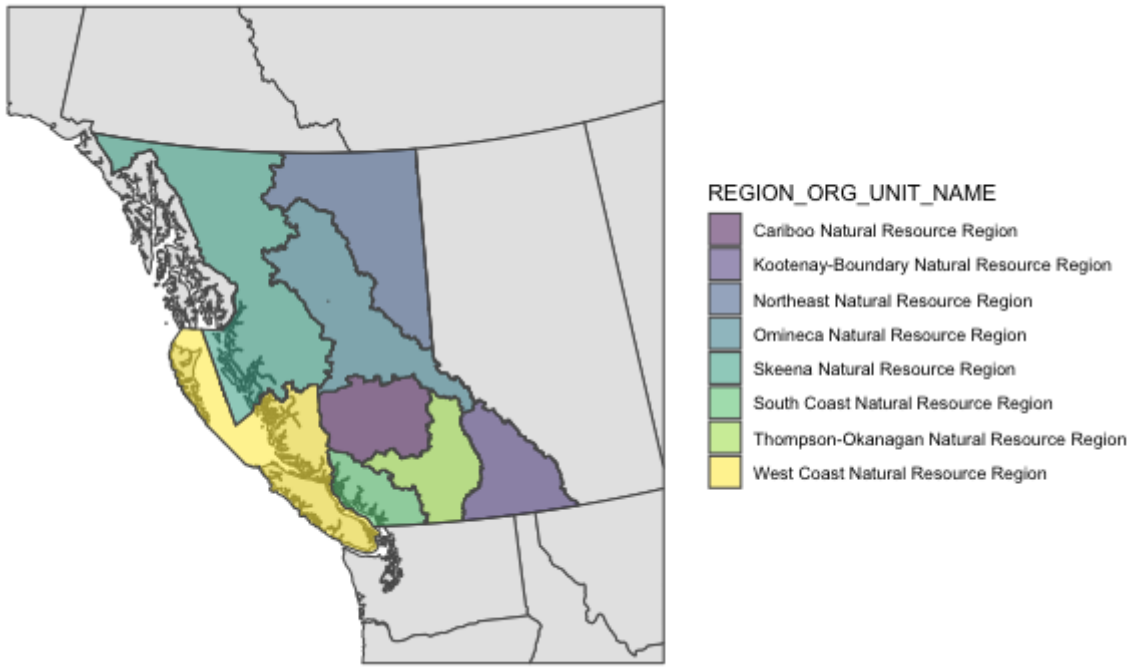
```
wna <- bcdc_query_geodata('7-5m-provinces-and-states-the-atlas-of-canada-base-maps-for-bc') %>%
  filter(!is.na(NAME)) %>%
  collect()

nr_region <- nr_district %>%
  group_by(REGION_ORG_UNIT_NAME) %>%
  summarise()
```

+ Western North American

```
fancy_plot <- ggplot() +
  geom_sf(data = wna)
fancy_plot
```

+ Natural Resource Regions

```
fancy_plot +
  geom_sf(data = nr_region, alpha = 0.5, aes(fill = REGION_ORG_UNIT_NAME))
```



+ Intersect Natural Resource Districts

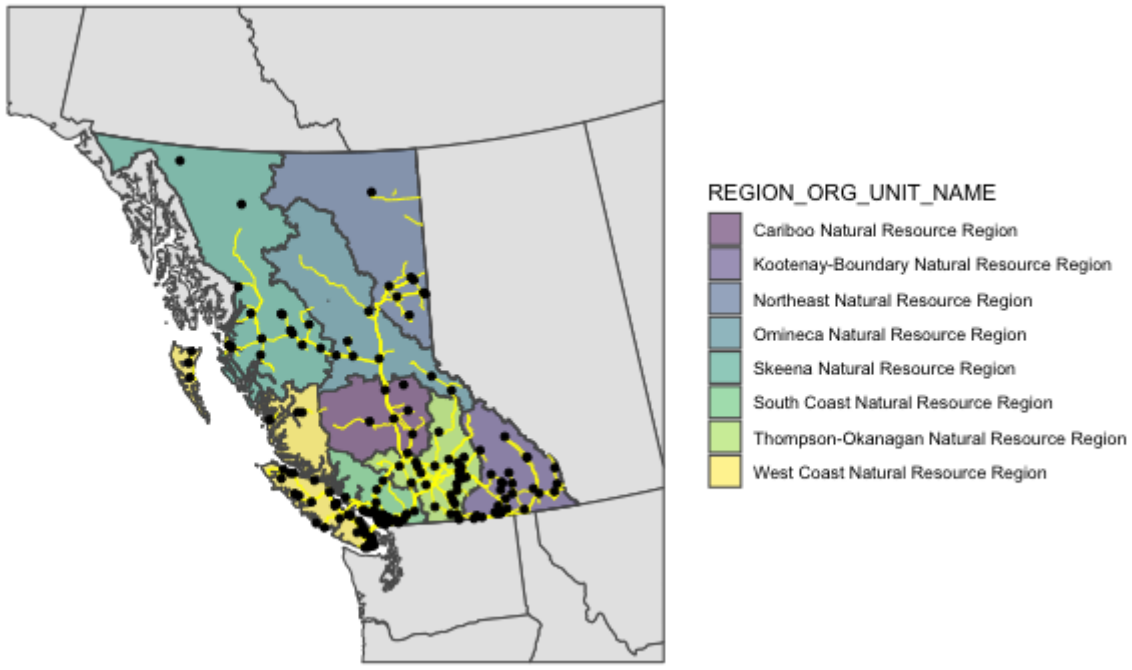
```
nr_region_int <- nr_region %>%
  st_intersection(wna)

fancy_plot <- fancy_plot + geom_sf(data = nr_region_int, alpha = 0.5, aes(fill =
REGION_ORG_UNIT_NAME))
fancy_plot
```

+ Add some lines

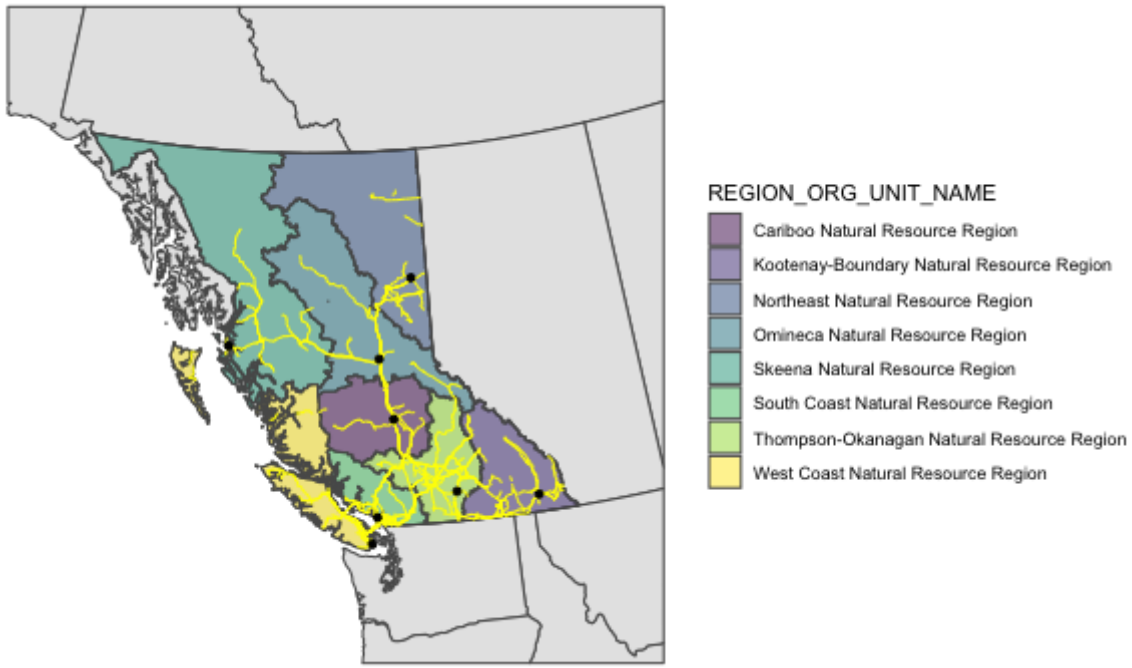
```
fancy_plot <- fancy_plot +
  geom_sf(data = lines, colour = "yellow")
```

+ Cities

```
fancy_plot + geom_sf(data = bc_cities)
```

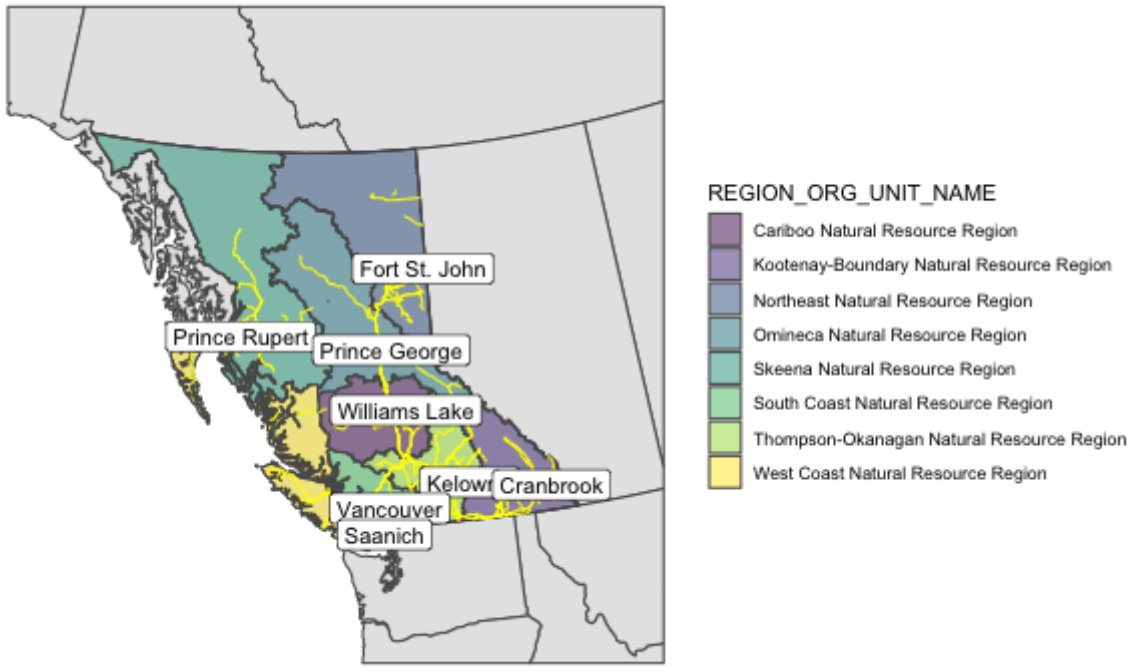


```
nrow(bc_cities)
#> [1] 166

cities_by_region <- bc_cities %>%
  st_join(nr_region, join = st_intersects) %>%
  group_by(REGION_ORG_UNIT_NAME) %>%
  filter(POP_2000 == max(POP_2000))

fancy_plot + geom_sf(data = cities_by_region)
```



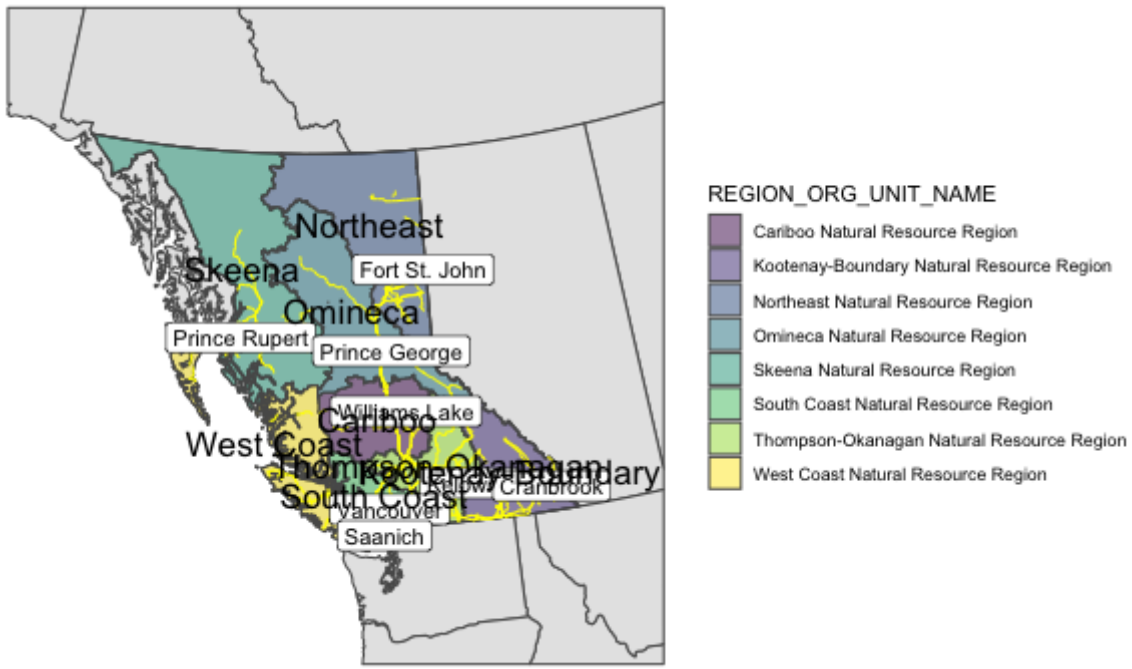
+ Cities with Name

```
fancy_plot <- fancy_plot +
  geom_sf(data = cities_by_region) +
  geom_sf_label(data = cities_by_region, aes(label = NAME), nudge_y = 3E4, nudge_x = 4E4)
fancy_plot
```



+ Labelling the Regions

```
region_names <- st_centroid(nr_region) %>%
  mutate(REGION_ORG_UNIT_NAME = gsub(" Natural Resource Region", "", REGION_ORG_UNIT_NAME))

fancy_plot +
  geom_sf_text(data = region_names, aes(label = REGION_ORG_UNIT_NAME), size = 6)
```



+ Map Components

```
library(ggspatial)
fancy_plot +
  geom_sf_text(data = region_names, aes(label = REGION_ORG_UNIT_NAME), size = 6) +
  coord_sf(datum = NA, expand = FALSE) +
  annotation_scale(pad_x = unit(2, "cm"), pad_y = unit(1, "cm"),
                   location = "bl",  style = "ticks", width_hint = 0.2) +
  annotation_north_arrow(location = "bl", which_north = "grid")
```

+ Legends and titles

```
text_box <- paste(strwrap("Some text that explains why I am plotting so things here and then
find another thing to talk about again.", width = 40), collapse = "\n")

fancy_plot +
  geom_sf_text(data = region_names, aes(label = REGION_ORG_UNIT_NAME), size = 6) +
  coord_sf(datum = NA, expand = FALSE) +
  annotation_scale(pad_x = unit(2, "cm"), pad_y = unit(1, "cm"),
                   location = "bl",  style = "ticks", width_hint = 0.2) +
  annotation_north_arrow(location = "bl", which_north = "grid") +
  labs(title = "British Columbia Natural Resource Regions",
       subtitle = "Major transmission lines displayed using yellow",
       caption = "Data retrieved from the BC Data Catalogue using the bcdata package") +
  guides(fill = FALSE) +
  annotate("label", x = 4E5, y = 4E5, label = text_box)
```

+ Themes and Colours

```
out <- fancy_plot +
  geom_sf_text(data = region_names, aes(label = REGION_ORG_UNIT_NAME), size = 6) +
  coord_sf(datum = NA, expand = FALSE) +
  annotation_scale(pad_x = unit(2, "cm"), pad_y = unit(1, "cm"),
                   location = "bl",  style = "ticks", width_hint = 0.2) +
  annotation_north_arrow(location = "bl", which_north = "grid") +
  labs(title = "British Columbia Natural Resource Regions",
       subtitle = "Major transmission lines displayed in yellow",
       caption = "Data retrieved from the BC Data Catalogue using the bcdata package") +
  guides(fill = FALSE) +
  annotate("label", x = 4E5, y = 4E5, label = text_box) +
  theme_void() +
  theme(panel.background = element_rect(fill = "aliceblue"),
        panel.border = element_rect(size = 2, fill = NA))
out
```

+ export

```
ggsave(out, file = "fancy_plot.pdf", height = 15, width = 15)
```

+ Some additional tweaking

```
out <- fancy_plot +
  geom_sf_text(data = region_names, aes(label = REGION_ORG_UNIT_NAME), size = 6) +
  coord_sf(datum = NA, expand = FALSE) +
  annotation_scale(pad_x = unit(2, "cm"), pad_y = unit(1, "cm"),
                   location = "bl",  style = "ticks", width_hint = 0.2) +
  annotation_north_arrow(location = "bl", which_north = "grid") +
  labs(title = "British Columbia Natural Resource Regions",
       subtitle = "Major transmission lines displayed in yellow",
       caption = "Data retrieved from the BC Data Catalogue using the bcdata package") +
  guides(fill = FALSE) +
  annotate("label", x = 4E5, y = 4E5, label = text_box) +
  theme_void() +
  theme(panel.background = element_rect(fill = "aliceblue"),
        panel.border = element_rect(size = 2, fill = NA),
        plot.title = element_text(size = 20),
        plot.subtitle = element_text(size = 18),
        plot.caption = element_text(size = 15))

ggsave(out, file = "fancy_plot.pdf", height = 15, width = 15)
```

Resources for R