Transit Access Analysis in R

Justin Sherrill

Use cases:

- Measuring “usefulness” of transit for a given area
- Used for urban planning, public health, market analytics
- Can compare transit against other transport modes
- Can compare transit usefulness over time

Packages:

- Analysis & viz: {tidyverse} {sf} {tidycensus} {tigris} {mapview}
- Database tools: GTFS using {tidytransit} {r5r}
- Amenities data: OSM using {osmdata}

Our overarching question:

- How (and where) has TriMet service and usefulness changed between 2019 and 2023?

Two ingredients:

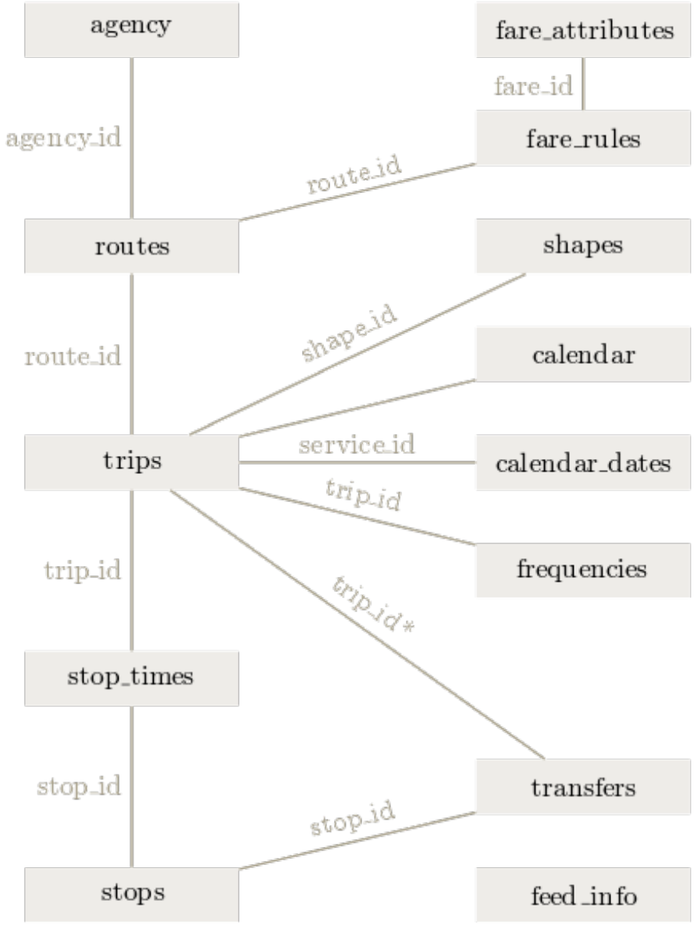
- GTFS .zip
- OSM .pbf

```
1 print(list.files("./01_data/01_raw/2023/"))
```

```
[1] "c_tran_gtfs_2023"
"c_tran_gtfs_2023.zip"
[3] "gtfs_23"                                "gtfs_23.zip"
[5] "network_settings.json"                  "network.dat"
[7] "portland_oregon.osm.pbf"
"portland_oregon.osm.pbf.mapdb"
[9] "portland_oregon.osm.pbf.mapdb.p"
```

What is it?

- **G**eneral **T**ransit **F**eed **S**pecification relational database
- Created by Google programmers and TriMet IT staff in 2005
- Since adopted by hundreds of transit agencies worldwide
- Typically packaged seasonally by larger agencies, or annually by smaller ones



- https://transitfeeds.com/ (soon to be deprecated!)
- https://database.mobilitydata.org/ (its replacement)
- https://www.transit.land
- Or directly from most agencies' websites

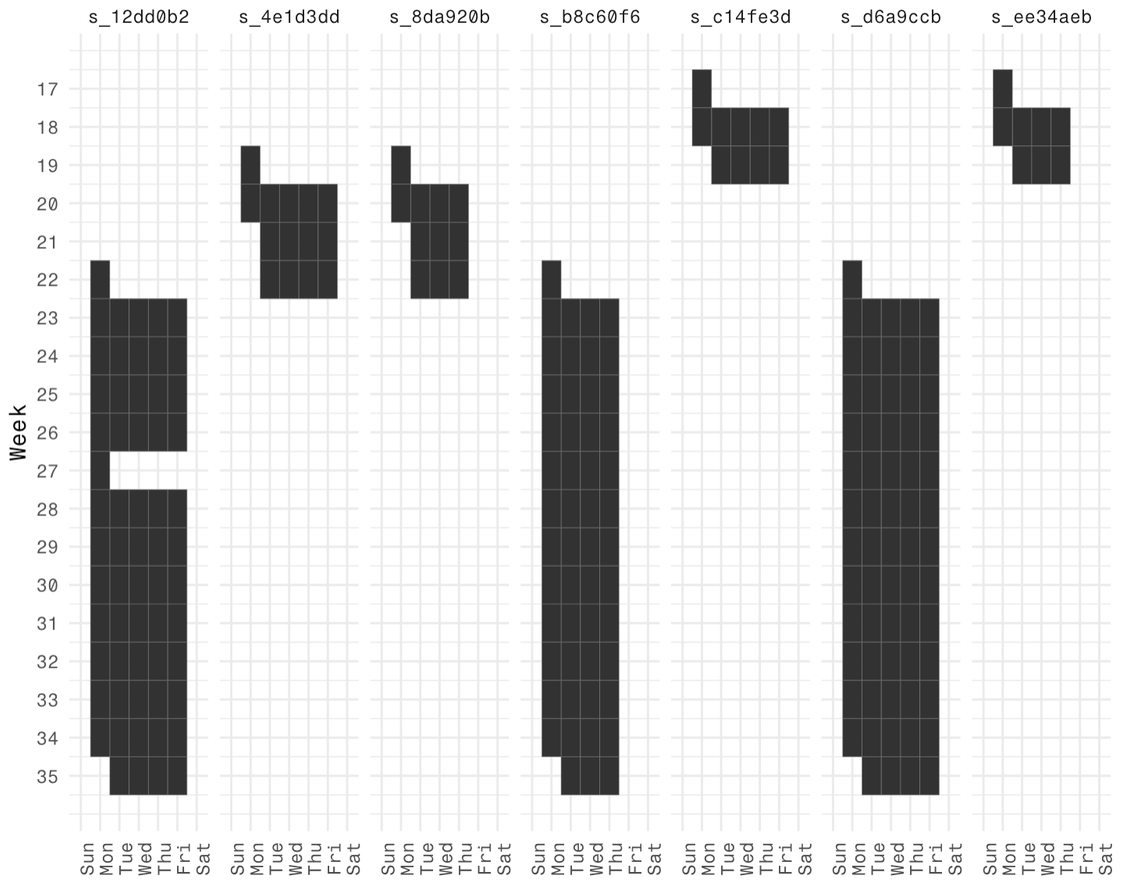
- https://www.interline.io/osm/extracts/
- https://download.geofabrik.de/

```r
blocks <- blocks("OR",
                 county = c("Multnomah",
                            "Clackamas",
                            "Washington")) %>%
  st_transform(4326) %>%
  select(geoid = GEOID20)

counties <- counties("OR") %>%
  filter(NAME %in% c("Multnomah",
                     "Clackamas",
                     "Washington")) %>%
  st_transform(4326) %>%
  select(geoid = GEOID)
```

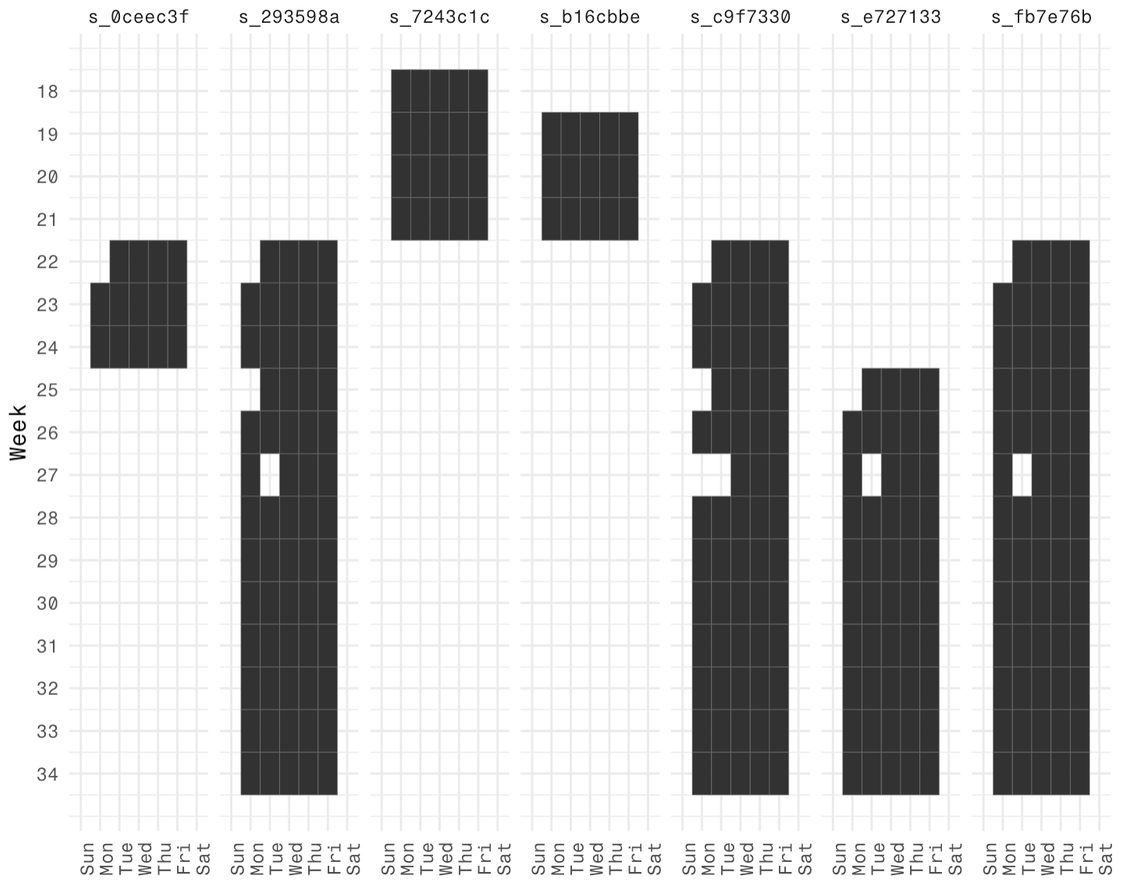
- Allows us a all-encompassing look at TriMet service in the Portland metro area

```r
gtfs_19 <- read_gtfs("./01_data/01_raw/2019/gtfs_19.zip") %>%
  gtfs_as_sf() %>%
  set_servicepattern()

gtfs_23 <- read_gtfs("./01_data/01_raw/2023/gtfs_23.zip") %>%
  gtfs_as_sf() %>%
  set_servicepattern()
```

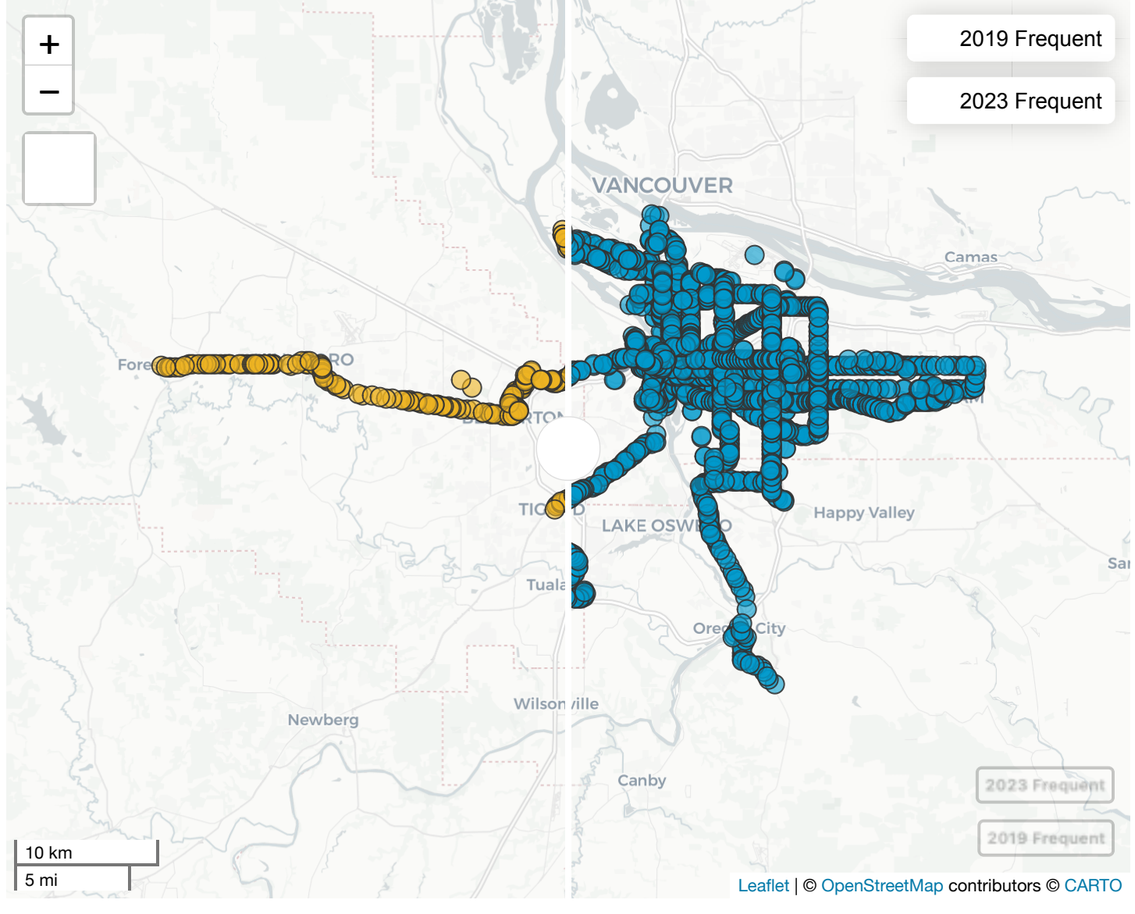
```r
service_pattern_summary_19 <- gtfs_19$trips %>%
  left_join(gtfs_19$.$servicepatterns, by="service_
  left_join(gtfs_19$stop_times, by="trip_id") %>%
  group_by(servicepattern_id) %>%
  summarise(
    trips = n(),
    routes = n_distinct(route_id),
    stops = (n_distinct(stop_id)/2))

service_calendar_19 <- gtfs_19$.$dates_servicepatte
  mutate(wday = wday(date, label = T, abbr = T, wee
         week_nr = week(date)) %>%
  group_by(week_nr) %>%
  mutate(week_first_date = min(date)) %>%
  group_by(servicepattern_id) %>%
  # We only care about weekday service patterns
  filter(any(wday == "Mon"),
         # and we don't care about special holiday
         length(date) > 7) %>%
```



```r
service_pattern_summary_23 <- gtfs_23$trips %>%
  left_join(gtfs_23$.$servicepatterns, by = "servic
  left_join(gtfs_23$stop_times, by = "trip_id") %>%
  group_by(servicepattern_id) %>%
  summarise(
    trips = n(),
    routes = n_distinct(route_id),
    stops = (n_distinct(stop_id) / 2)
  )

service_calendar_23 <- gtfs_23$.$dates_servicepatte
  mutate(wday = wday(
    date,
    label = T,
    abbr = T,
    week_start = 7
  ),
  week_nr = week(date)) %>%
```



```r
stop_freq_19 <- get_stop_frequency(
  gtfs_19,
  start_time = "09:00:00",
  end_time = "17:00:00",
  service_ids = service_ids_19
) %>%
  left_join(gtfs_19$stops %>%
              select(stop_id)) %>%
  st_as_sf() %>%
  mutate(frequent = if_else(mean_headway <= 900, "F
  distinct(stop_id, frequent, .keep_all = TRUE)

stop_freq_23 <- get_stop_frequency(
  gtfs_23,
  start_time = "09:00:00",
  end_time = "17:00:00",
  service_ids = service_ids_23
) %>%
  left_join(gtfs_23$stops %>%
```



```r
stop_freq_19 %>%
  st_drop_geometry() %>%
  count(frequent) %>%
  mutate(pct = n / sum(n)) %>%
  print()

stop_freq_23 %>%
  st_drop_geometry() %>%
  count(frequent) %>%
  mutate(pct = n / sum(n)) %>%
  print()
```

```
# A tibble: 2 × 3
  frequent          n   pct
  <chr>         <int> <dbl>
1 Frequent       2067 0.286
2 Non-frequent   5161 0.714
# A tibble: 2 × 3
  frequent          n   pct
  <chr>         <int> <dbl>
1 Frequent       1962 0.290
2 Non-frequent   4814 0.710
```

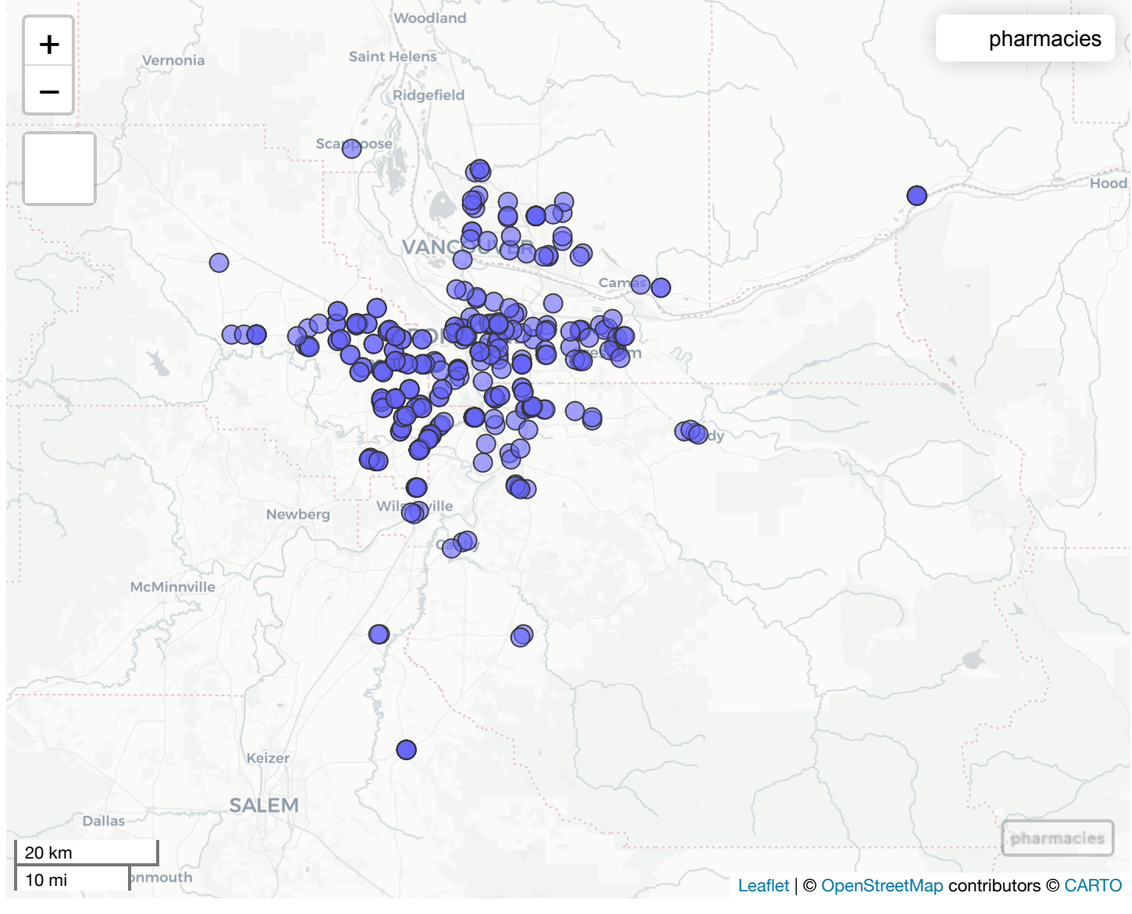
Uses:

- Building an inventory of stops
- Mapping service networks
- Summarizing feature attributes

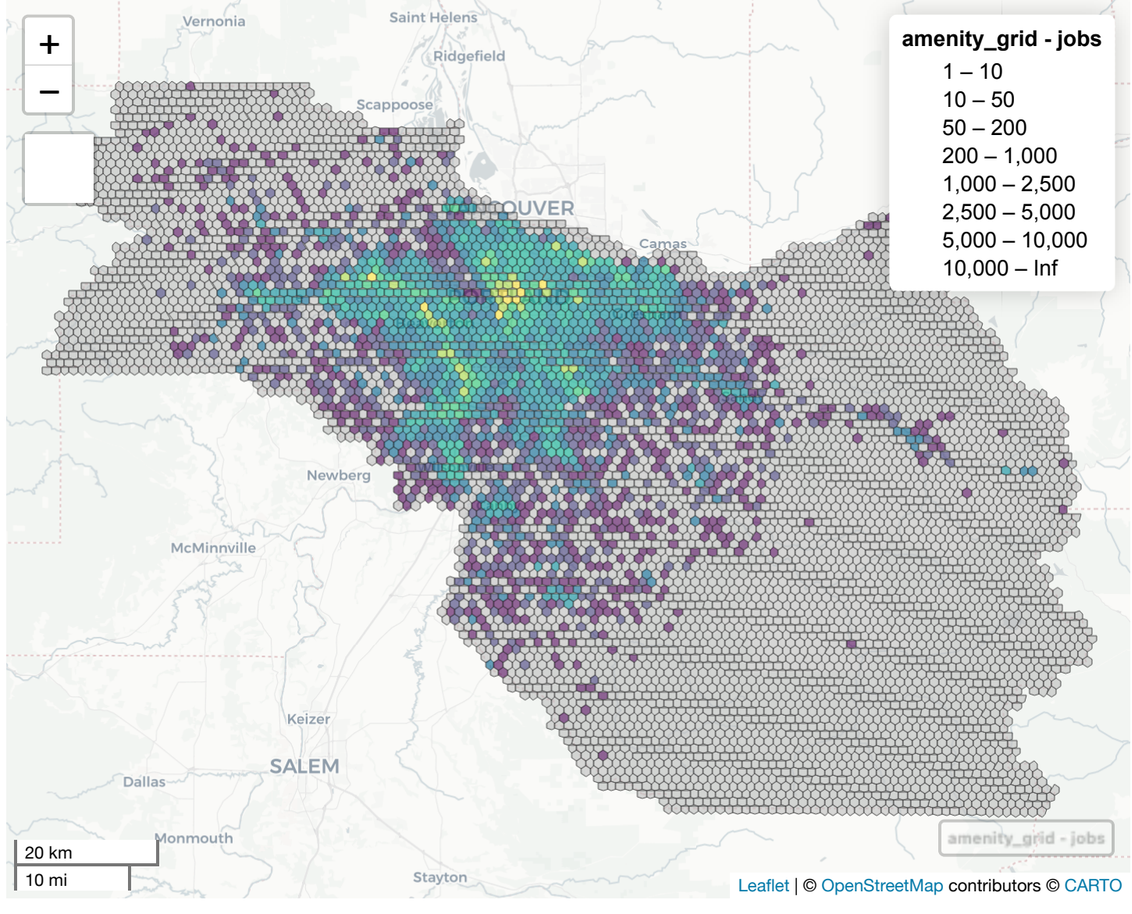
Caveats:

- GTFS is more of a template than a rigid framework
- Many exceptions from agency to agency
- Building knowledge (reaching out to staff) is essential
- Frequency depends on context

```r
# {lehdr} makes it easy to download jobs data for a
wac <- grab_lodes(
  state = "or",
  # LODES currently only goes up to 2020, and is nc
  year = 2019,
  lodes_type = "wac",
  job_type = "JT00",
  use_cache = TRUE
) %>%
  transmute(geoid = w_geocode,
            jobs = C000,
            year,
            type = "job") %>%
  inner_join(blocks) %>%
  st_as_sf() %>%
  st_centroid()

schools <-
```



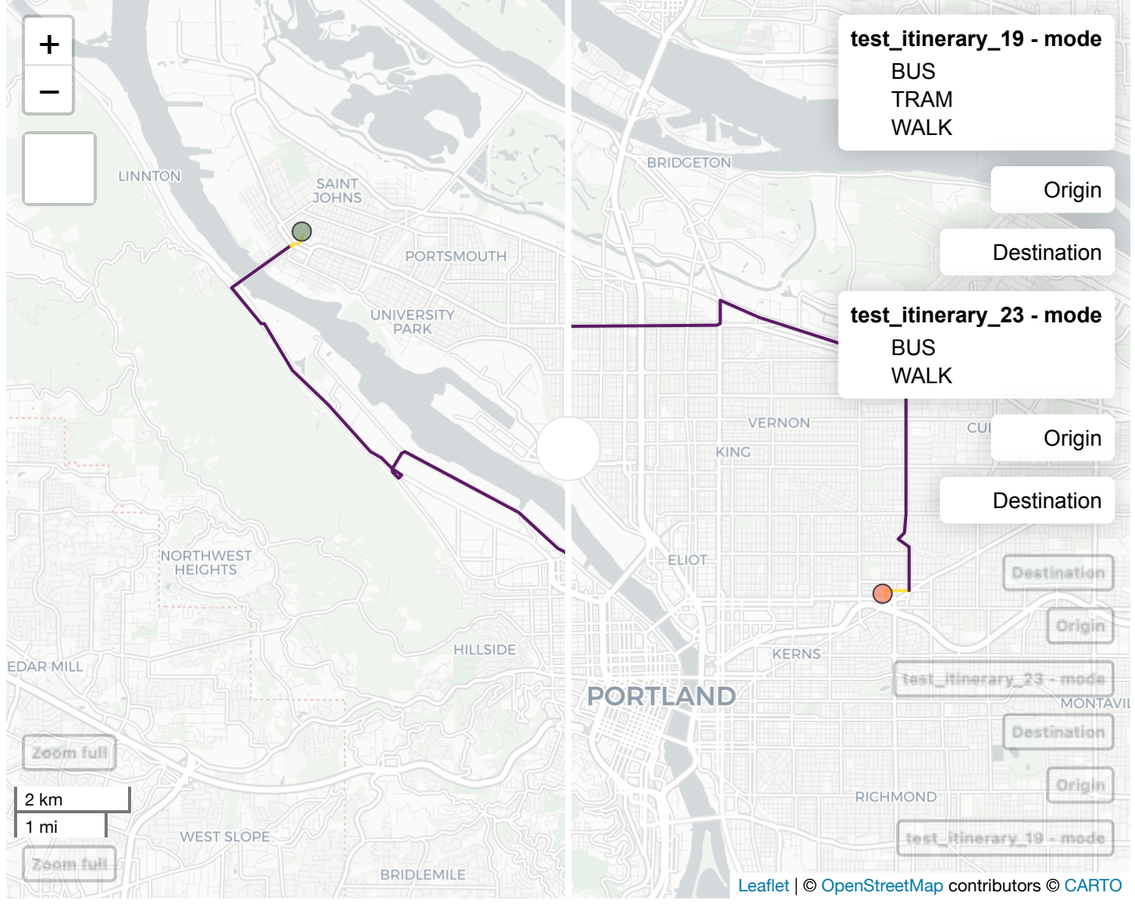
```r
grid <- st_make_grid(st_transform(counties, 6855),
                     square = FALSE,
                     cellsize = 3960) %>%
  st_sf() %>%
  st_transform(4326) %>%
  st_filter(counties) %>%
  rowid_to_column("gridid") %>%
  mutate(gridid = as.character(gridid))

jobs_grid <- aggregate(st_transform(wac, 6855)['job

pharm_grid <- aggregate(st_transform(pharmacies %>%
            by = st_transform(grid, 6855)['gridid']
            sum)

school_grid <- aggregate(st_transform(schools %>% m

amenity_grid <- grid %>%
```



What is it?:

- Rapid Realistic Routing on Real-world and Reimagined networks
- Eats GTFS and OSM PBFs, produces routing analytics and matrices
- Can model walk, bike, bus, rail, car, funicular, etc. trips
- Can model walk/traffic preferences, incorporate fare structure, and more
- Alternatives: {opentripplanner} and {otpr}

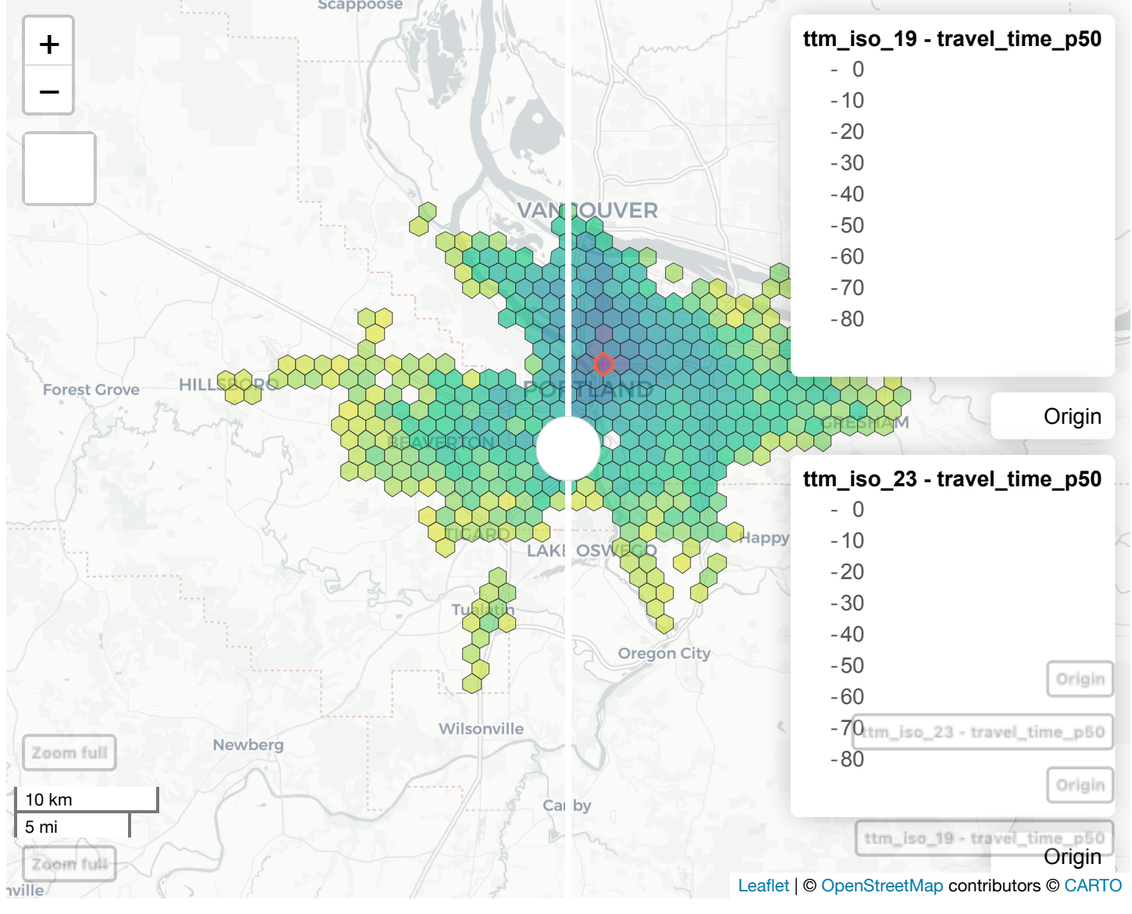
```r
r5r_core_19 <- setup_r5(data_path = "./01_data/01_r
                        verbose = TRUE,
                        overwrite = FALSE)

r5r_core_23 <- setup_r5(data_path = "./01_data/01_r
                        verbose = TRUE,
                        overwrite = FALSE)

mode <- c("WALK", "BUS", "SUBWAY", "TRAM", "RAIL")

departure_datetime_19 <- as.POSIXct("19-06-2019 08:
                                    format = "%d-%m

departure_datetime_23 <- as.POSIXct("21-06-2023 08:
                                    format = "%d-%m

o <- blocks %>%
  filter(geoid == "410510026003010") %>%
```



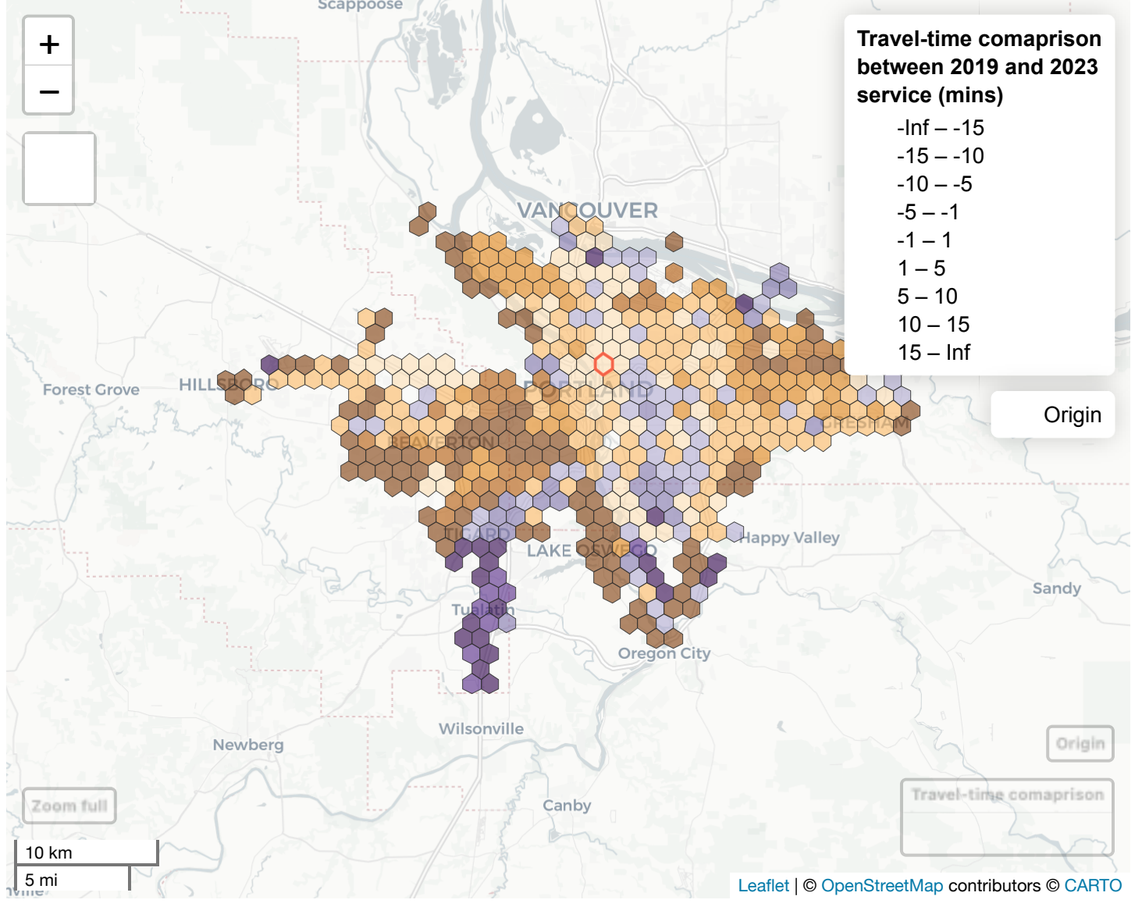
```r
grid_points <- grid %>%
  st_centroid() %>%
  transmute(id = gridid,
            lat = st_coordinates(.)[,2],
            lon = st_coordinates(.)[,1]) %>%
  st_drop_geometry() %>%
  find_snap(points = .,
            r5r_core = r5r_core_23) %>%
  filter(found != FALSE) %>%
  select(id = point_id,
         lat = snap_lat,
         lon = snap_lon)

# calculate travel time matrix
ttm_grid_19 <- travel_time_matrix(
  r5r_core_19,
  origins = grid_points,
  destinations = grid_points,
```

	from_id	to_id	travel_time_p50
1	1	1	0
2	2	2	0
3	3	3	0
4	3	6	11
5	4	4	0
6	4	12	25
7	5	1	51
8	5	5	0
9	5	9	33
10	5	13	48

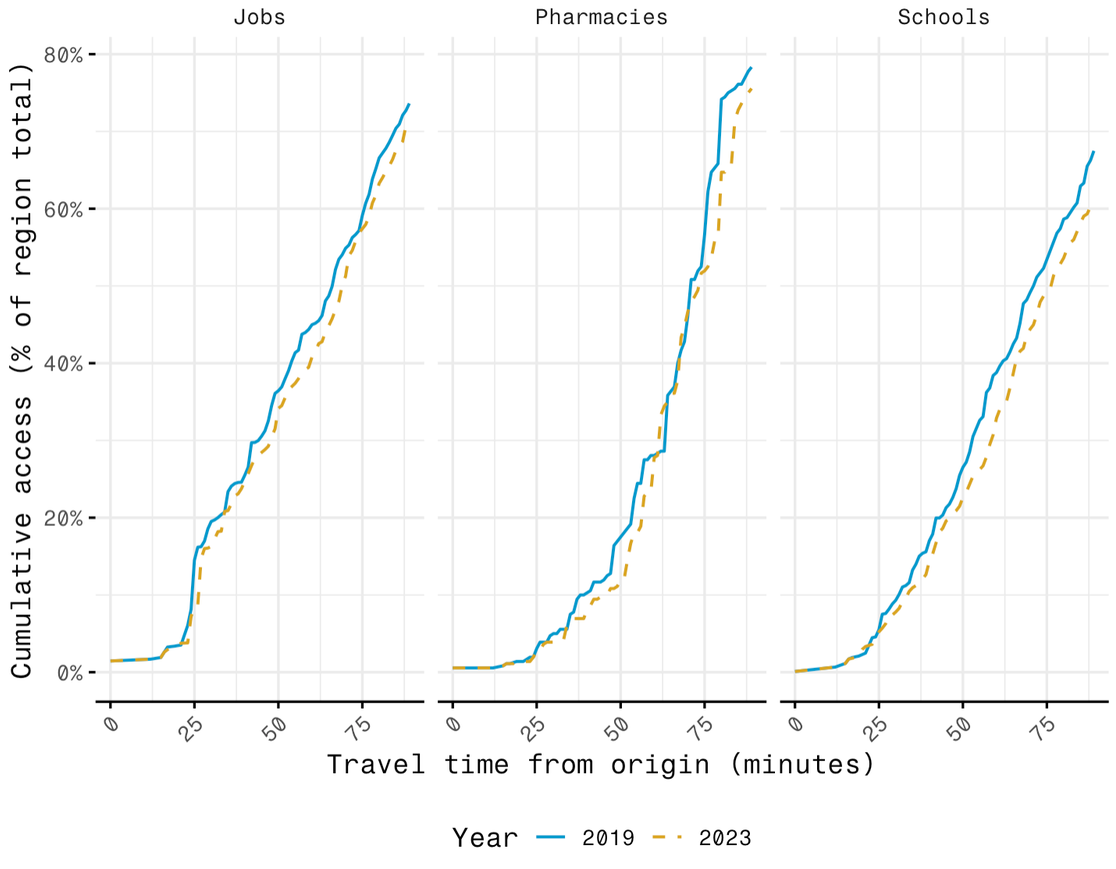
```r
me <- "2206"

me_hex <- amenity_grid %>%
  filter(gridid == me) %>%
  mapview(
    color = "tomato",
    lwd = 2,
    col.regions = "tomato",
    alpha.regions = 0,
    layer.name = "Origin"
  )

ttm_iso_19 <- ttm_grid_19 %>%
  filter(from_id == me) %>%
  left_join(grid,
            by = c("to_id" = "gridid")) %>%
  st_as_sf()

ttm_iso_23 <- ttm_grid_23 %>%
```



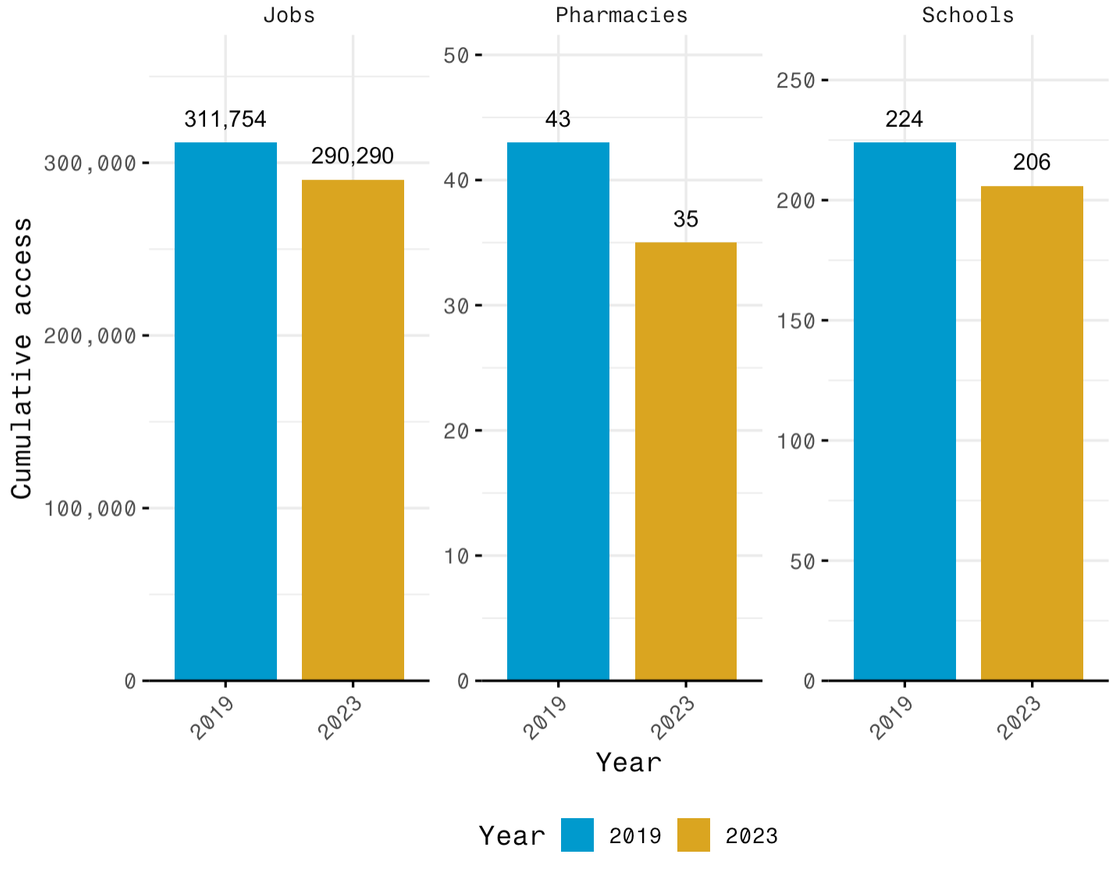
```r
comparison_iso <- ttm_iso_19 %>%
  select(gridid = to_id,
         tt_19 = travel_time_p50) %>%
  st_drop_geometry() %>%
  full_join(ttm_iso_23 %>%
              st_drop_geometry() %>%
              select(gridid = to_id,
                     tt_23 = travel_time_p50)) %>%
  mutate(
    diff = tt_23 - tt_19,
    diff = case_when(
      is.na(tt_23) & !is.na(tt_19) ~ Inf,!is.na(tt_
        is.na(tt_19) ~ -Inf,
      TRUE ~ diff
    )
  ) %>%
  ungroup() %>%
  left_join(grid) %>%
```



```r
ttm_amenities <- ttm_grid_19 %>%
  filter(from_id == me) %>%
  select(gridid = to_id,
         tt_19 = travel_time_p50) %>%
  left_join(st_drop_geometry(amenity_grid)) %>%
  mutate(tt = as.numeric(tt_19)) %>%
  group_by(tt, jobs_total, schools_total, pharmacie
  summarise_at(vars(jobs, schools, pharmacies), sum
  ungroup() %>%
  mutate_at(vars(jobs, schools, pharmacies), .funs
  mutate(year = 2019) %>%
  rbind(
    ttm_grid_23 %>%
      filter(from_id == me) %>%
      select(gridid = to_id,
             tt_23 = travel_time_p50) %>%
      left_join(st_drop_geometry(amenity_grid)) %>%
      mutate(tt = as.numeric(tt_23)) %>%
      group_by(tt, jobs_total, schools_total, pharm
```



```r
my_amenities <- ttm_amenities %>%
  filter(tt <= 45,
         !str_detect(amenity, "pct")) %>%
  group_by(year, amenity) %>%
  summarise(max = max(value)) %>%
  ungroup()

ggplot(my_amenities,
       aes(x = factor(year),
           y = max,
           fill = factor(year))) +
  geom_bar(stat = "identity",
           width = .8) +
  geom_text(aes(label = comma(max)),
            vjust = -1) +
  facet_wrap(~str_remove_all(amenity, "_cumul"),
             scales = "free_y",
             labeller = labeller(.default = str_to_
  theme_minimal(base_size = 13
```



Uses:

- Dynamic trip routing
- Measuring transit usefulness based on O/D location or time of day
- Comparing transit usefulness over time
- Creating an inventory of amenities

Caveats:

- All the GTFS quirks are backgrounded, but still there
- OSM has limits, topography errors could propogate (API rates and usability)
- No easy way to manipulate GTFS within R (yet)

Thank you!