Towards Interactivity

HES 505 Fall 2024: Session 27

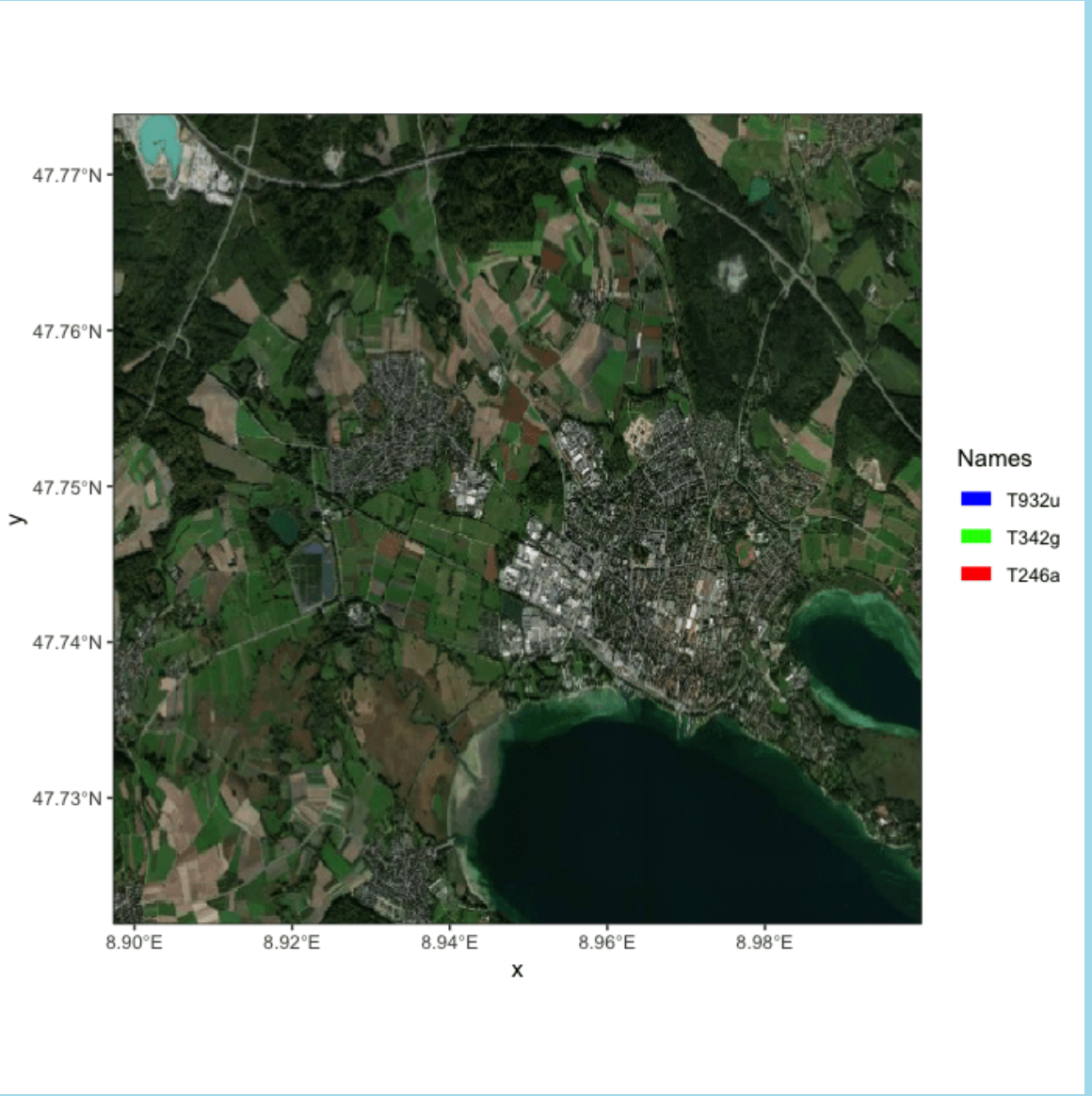
Carolyn Koehn

Objectives

- Define an API and their use in interactive visualization
- Obtain a token for common mapping APIs
- Build interactive maps using common packages
- Recognize other opportunities for interactive visuals with R

3 Categories of data visualization

- Static
- Interactive
- Dynamic



dynamic

Why Move Beyond Static Maps?

Dealing with complex datasets

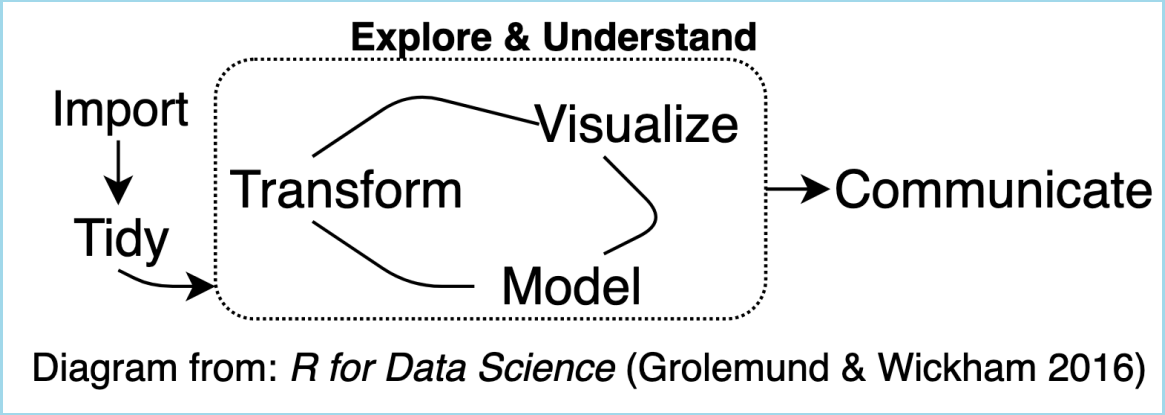


Diagram from: *R for Data Science* (Grolemund & Wickham 2016)

- Identifying structure that might otherwise be hidden
- Diagnosing models and interpreting results
- Aiding the sense-making process

Clarity in presentation

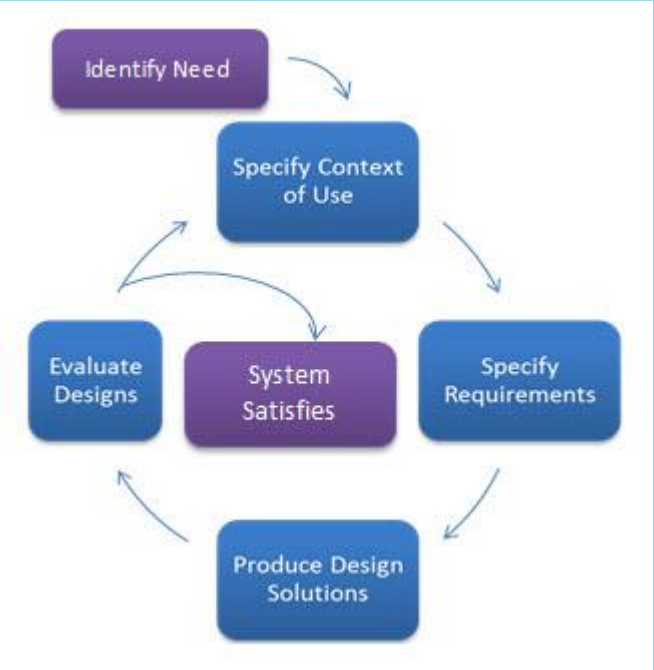
- Zooming allows the user to determine scale of presentation
- Hovering allows more information to be displayed ‘on-demand’
- Subsetting facilitates ease of interpretation

Designing for the User

Who is your audience?

- Your advisor and colleagues?
- An external collaborator?
- The general public?
  - User archetypes

Iteration



From Usability.gov

- Feedback is critical
- Ideation: What *specifically* does the user need?
- Meaning: Are the data clearly defined and explained? Are the conclusions obvious?
- Function: Given the usecases, will the application (visualization) actually perform?

Building interactive visualizations in R

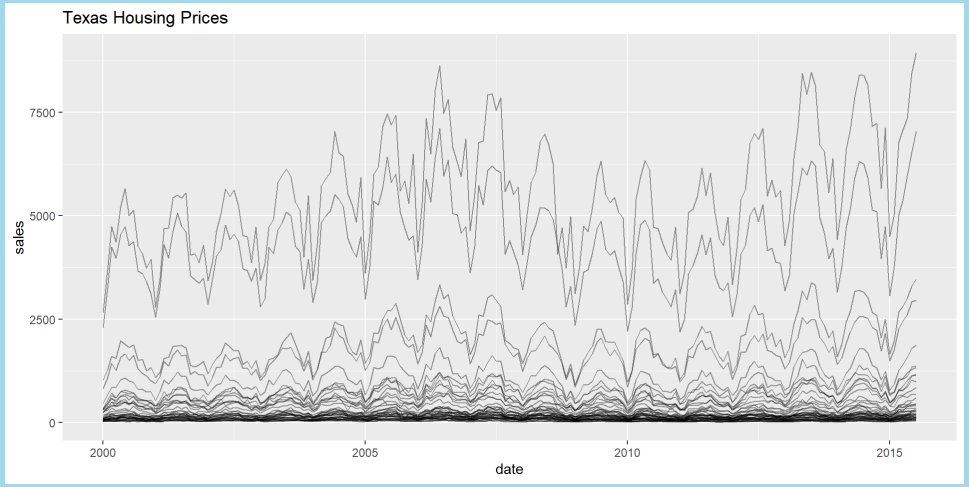
A note about APIs

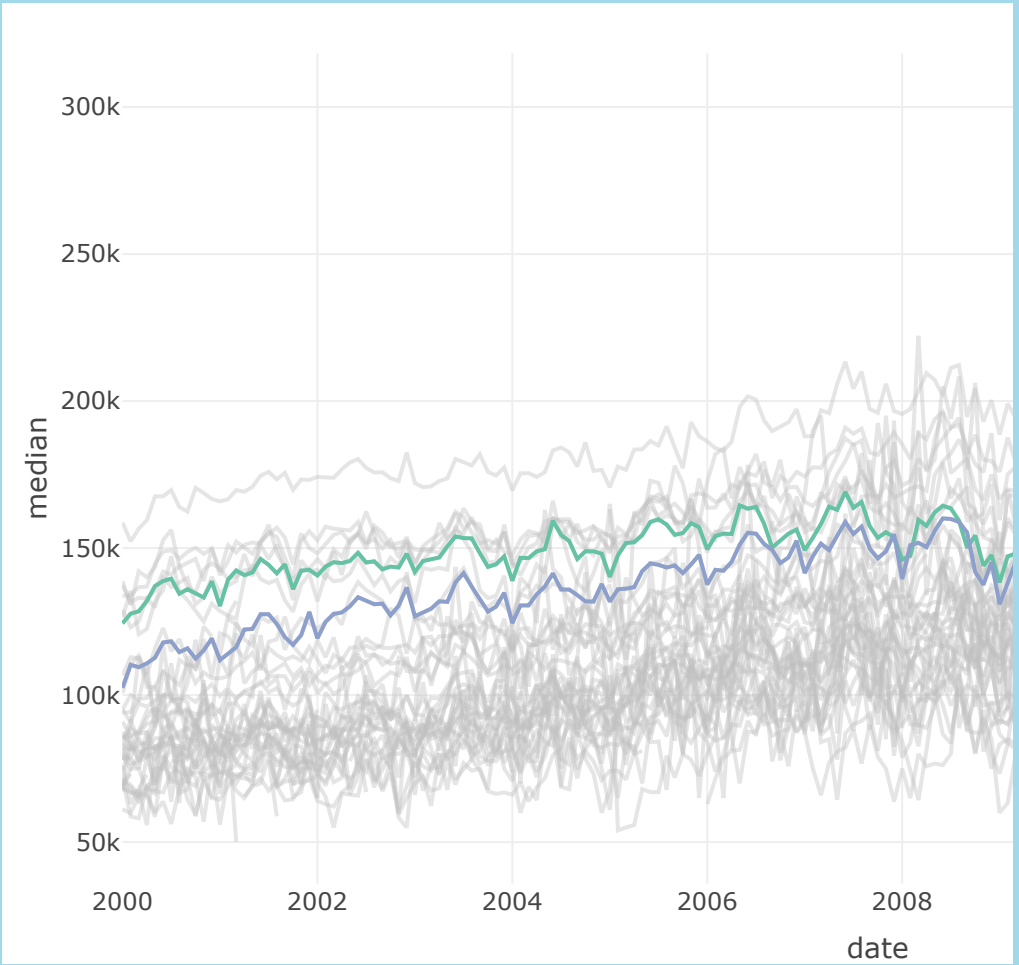
- API: Application Programming Interface
- A software intermediary that allows two applications to "communicate"
- Lots of `R` packages rely on APIs to access data on the web (e.g.,`tidycensus`)
- Facilitates reproducibility and powerful web applications built on `R` analyses
- May require "keys" and additional parsing (Mapbox and Google)

Interactive maps with `mapview` and `tmap`

- Easy extension of your existing code
- Class Demo

Clarity in presentation (revisited)





Using `plotly`

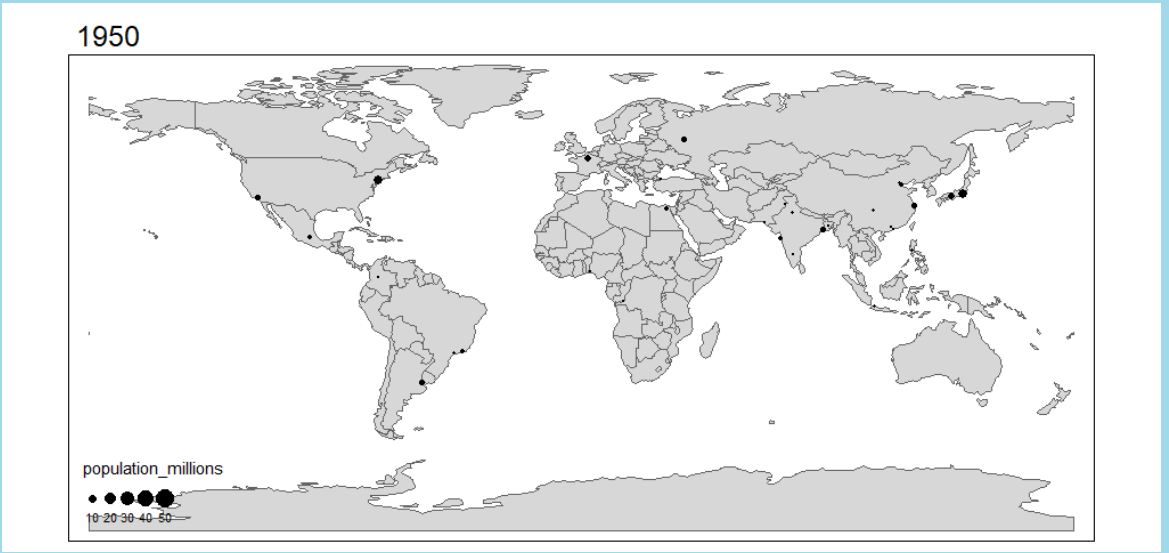
- Syntax is similar to `ggplot`
- `hoverinfo` describes which elements you'd like to make interactive
- Other plot elements available (see `?plot_ly`)

Using plotly

```
g <- txhousing %>%
  # group by city
  group_by(city) %>%
  # initiate a plotly object with date on x and median on y
   plotly::plot_ly(x = ~date, y = ~median) %>%
  # add a line plot for all texan cities
   plotly::add_lines(name = "Texan Cities", hoverinfo = "none",
                     type = "scatter", mode = "lines",
                     line = list(color = 'rgba(192,192,192,0.4)')) %>%
  # plot separate lines for Dallas and Houston
   plotly::add_lines(name = ~city,
             data = filter(txhousing,
                           city %in% c("Dallas", "Houston")),
             hoverinfo = ~city,
             color = ~city)
```

Animated maps with tmap and gganimate

```
1  urb_anim = tm_shape(world)
2    tm_shape(urban_agglomera
3    tm_facets(along = "year"
4  tmap_animation(urb_anim, f
```



dynamic