Interactive data visualisation

DATA5002 Lectorial Week 7

Assignments

Assignment 1 comments

Common problems in Assignment 1

General

- Carefully read all instructions, course announcements, and rubric.
- Every plot should be include informative title, axis labels, units (especially for an external audience).
- This class is not just about creating plots, but also understanding and interpreting plots, and distilling the message/conclusion of a plot.

Question 1

- Choose a graph appropriate for the audience.
  - For public consumption: bar chart or another simple chart.
  - For scientific audience: boxplot, violin plot, etc (better to provide more detail if you can)

Question 2 and 3

Question 2

- You should be able to look at a plot and highlight a few key insights using the visual information.

Question 3

- Don't neglect the statistics aspect: choosing appropriate variables for a model, deciding whether or not they should be included as factor or numeric.
- Diagnostic plots: even if a graph shows a small violation, it is better to mention it than to say the graph looks fine, even if you conclude that linear regression is okay in this case. This promotes honesty.
- Forest plot: if a term has been introduced in class, use examples from course materials, not from googling.

Improvements in Assignment 1

- Consider how best to use colour:
  - As an aesthetic to encode categories (but avoid colour-blind-unfriendly palette choices).
  - To highlight something important.
  - Avoid redundant rainbow colour palette for pure aesthetics (this is generally considered distracting chartjunk).
- Consider orderings besides the default (such as ordering bars from highest to lowest, rather than pizza brand letter).
- Consider whether wrangling or augmenting data would produce data that is more interpretable to the audience. Don't assume the data as given to you is the best representation.

Assignment 2 (project)

Overview

- Create a narrative infographic/scrollytelling article or interactive dashboard (or something in between).
- Suitable for a portfolio (depending how good you make it).
- Goal: unsuitable projects filtered out at proposal stage.
- Use R or another reproducible programming language (not Tableau).

Overview

Loosely, should be on the spectrum:

	narrative infographic	⟺	**interactive dashboard**
format	image or static page	⟺	application
UX	linear narrative	⟺	exploration
aesthetics	artistic	⟺	practical
audience	general	⟺	knowledgeable

Note that most of the above is negotiable to a point. Any deviations from above should be brought up in the proposal stage.

Thoughts on Shiny

- If creating a shiny project, create a minimum viable project (MVP) first.
- Especially if data is big (or even medium), test out on a small subset of data until far along in the development process.
- Then, add more graphs, complexity, or customisation one piece at a time.
- Debugging in shiny is harder than debugging a standard R script.
- Consider using a version control software like git/github (built into Rstudio).

Logistics

Proposal

- Submit by the end of Week 8.
- A wireframe, a short writeup, and any “prior art” you can find.
- Proposals may be rejected for being:
  - Too simple
  - Too similar to another project
  - Too similar to existing public materials
- It is your responsibility to ensure that your project is feasible.
- If you submit a proposal, work on project, and find it is infeasible, you can re-submit a new proposal (before end of Week 8).

Project progress

Progress

- You already have all skills needed to create a static narrative infographic.
- This week: additional skill of interactive graphics.
- Next week: shiny and dashboards.
- No matter what you want to create, you can already begin coding (once proposal is approved). - Data processing and wrangling. - Create the individual plots you want to include in your project. - Later: make them interactive, and/or arrange them into a dashboard.

. . .

Final submission

- Format depends on project type.
- Include a short report explaining how and why you did what you did.

Interaction design

Principles

Dimensions

Siang (2020)

1. Words
2. Visual representations
3. Physical objects or space
4. Time
5. Behaviour

Questions to consider

- What can a user do to directly interact with the interface?
- What about the appearance gives the user a clue about how it may function?
- Are the interface elements a reasonable size to interact with?
- Are familiar or standard formats used?

. . .

- Examples of error messages: `ggplot(mtcars, aes(wt, mpg)) %>% geom_point()`
- Examples of feedback: If recalculation takes a while, make sure the user is aware that the system is responsive.

Also,

- Are error messages informative and useful?
- What feedback does a user get once an action is performed?

Examples

- Shiny Gallery

Wireframing

Overview

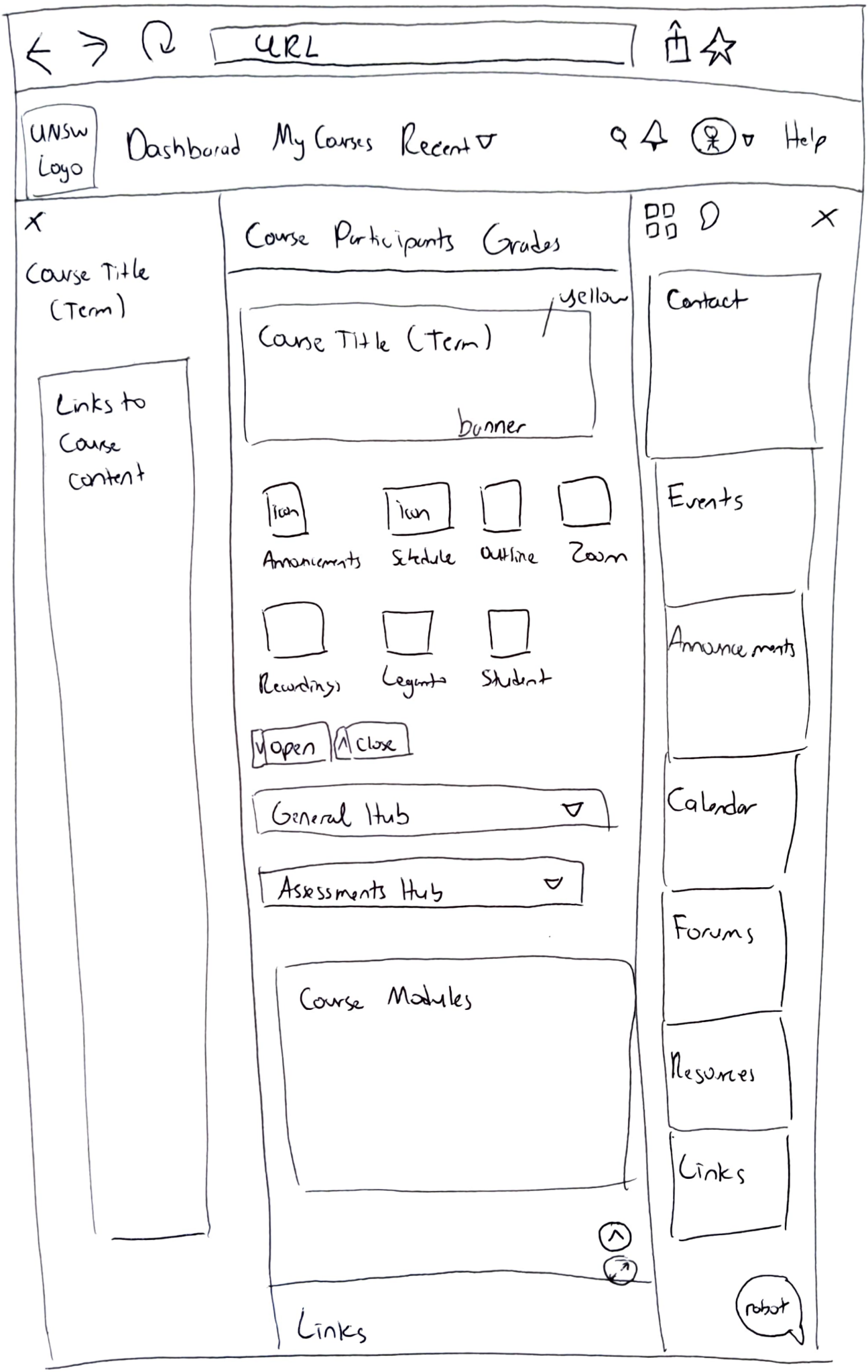
Snow (2012) | Emery (2016) | Osder (2016)

- high-level ideation
- structure and main elements
- assist in designing UI, help navigation
- general tool for UX design, useful for data vis.

. . .

- diagram of page layout
- layout of elements, hierarchy, allocation of space
- not actual design work: represent pictures with boxes
- indicate relative sizes, fonts, but not actual fonts or colours

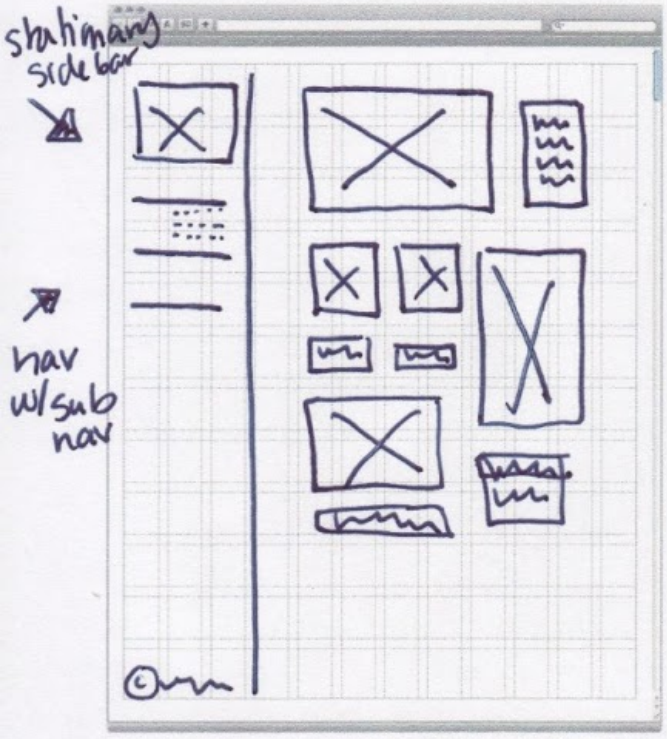
Example: DATA5002 Moodle page


URL
UNSW Logo
Dashboard
My Courses
Recent
Help
Course Participants Grades
Course Title (Term)
Links to Course content
yellow
Course Title (Term)
banner
Icon
Announcements
Icon
Schedule
Outline
Zoom
Recordings
Legends
Student
open
close
General Hub
Assessments Hub
Course Modules
Links
Contact
Events
Announcements
Calendar
Forums
Resources
Links
robot

Example: blog wireframes

Brooke

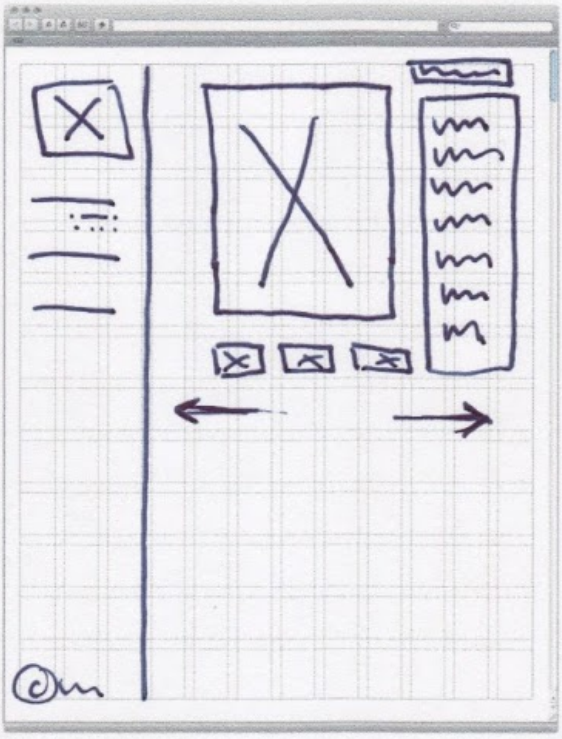
Title: IDEA #1



Sketch #: view/main page Portfolio projects & titles
Resolution:

Rating ☆ ☆ ☆ ☆ ☆

Title:

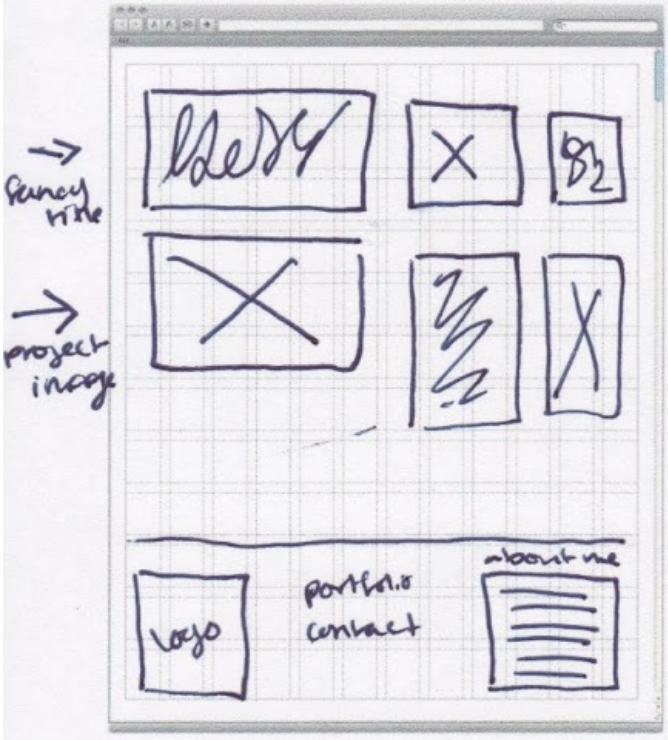


Sketch #: Project page w/ info & shuffling pics
Resolution:

Rating ☆ ☆ ☆ ☆ ☆

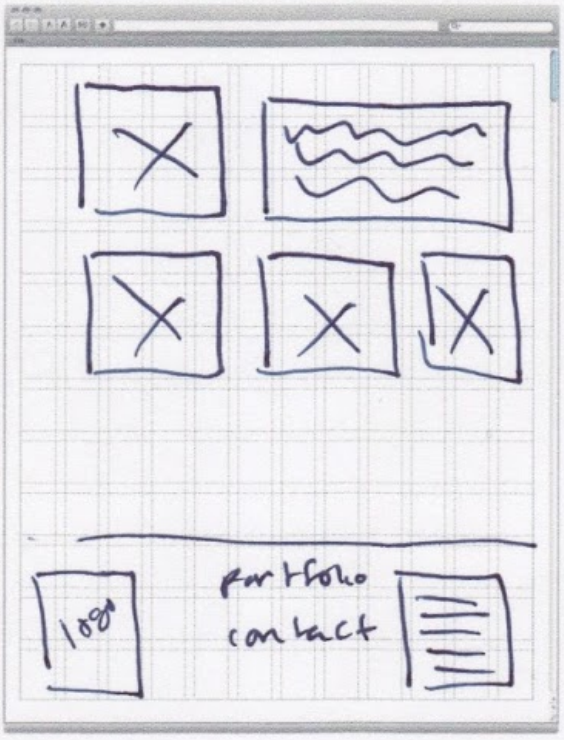
Title:

Title: IDEA #2



Sketch #: main page/ portfolio
Resolution:

Title:



Sketch #: project page
Resolution:

Title:

Visualisation heuristics

Purpose

Williams (2018)
Forsell and Johansson (2010)

- Things to pay attention to
- Predict and preempt usability issues for visualisations:
  - Evaluate and adjust plans (e.g., wireframe) before putting in the work.
  - Prevent usability issues before they emerge.

. . .

The article:

- 10 heuristics, 5 visualisations, 10 participants
- Participants evaluate visualisations according to heuristics
- "Baseline" evaluation as well

. . .

- Essentially, testing inter-rater reliability of heuristics

Heuristics (1–4)

H1 Spatial organization

- overall layout
- ease of locating info elements
- awareness of overall distribution of info elements

H2 Information coding

- mapping data elements to visual elements
- symbols/characteristics for alternative representations

H3 Orientation and Help

- user control over level of detail
- redo/undo of user actions
- breadcrumbs

H4 Data set reduction

- focusing and filtering
- clustering
- pruning

Heuristics (5–8)

H5 Recognition rather than recall

- user not required to remember
- easy access to instructions

H6 Remove the extraneous

- largest amount of data with least amount of ink

H7 Prompting

- discoverability of interactive elements
- which alternatives are available in a context
- system status information

H8 Minimal actions

- as few steps as possible to accomplish a task

Heuristics (9–10)

H9 Consistency

- similarity of design choices in similar contexts
- differences in different contexts

H10 Flexibility

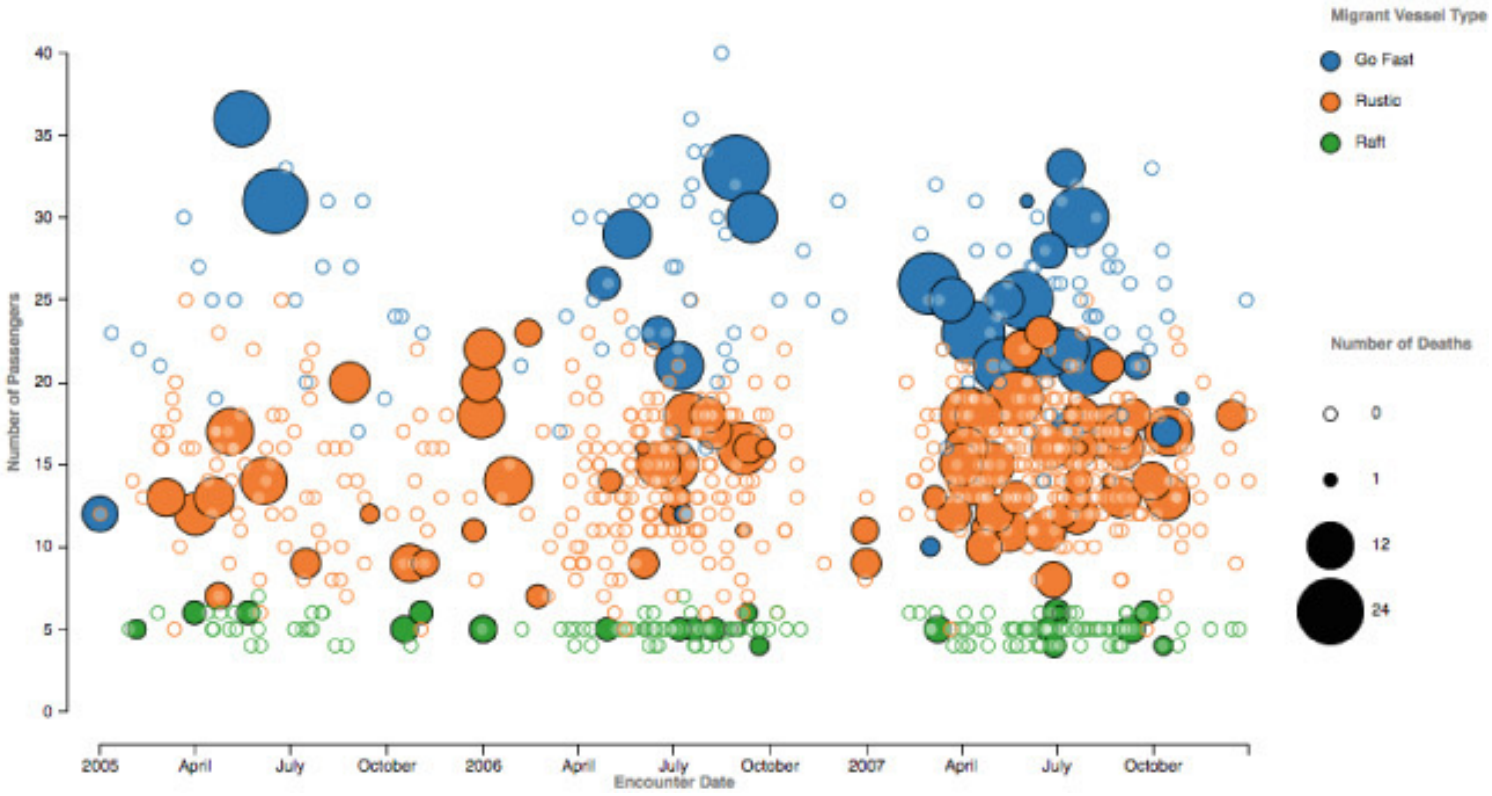
- adaptability to user's needs

Which of the above are specific to interactive visualisation?

V1: Scatterplot

Williams (2018)

Question: Which vessel type had the highest death per passenger ratio?



V1: Heuristics

Williams (2018)

H1 Occlusion: Difficult to see individual points

H2 Date axis compressed and difficult to read

H5 Difficult to compare dot size from different months

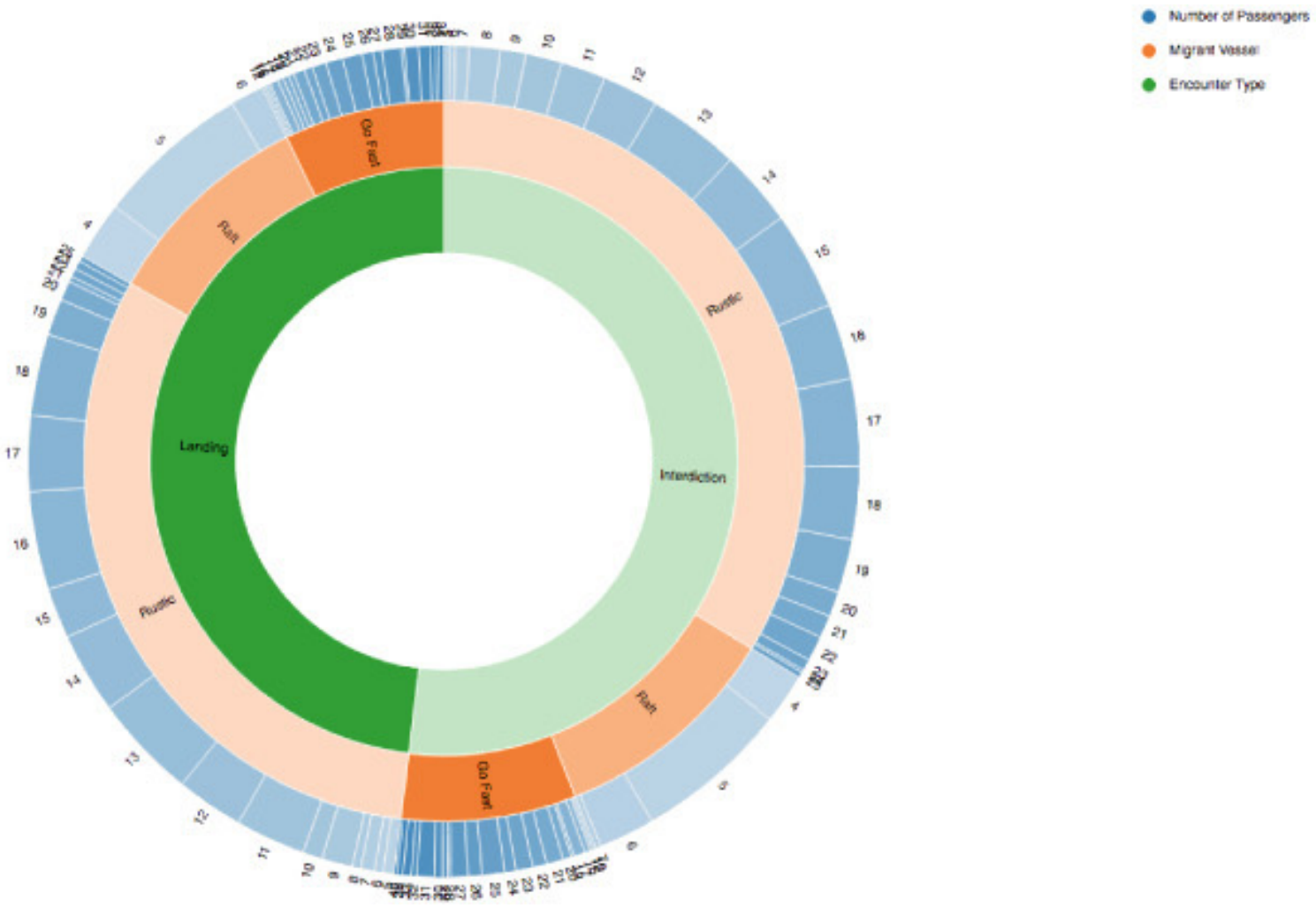
H5 Difficult to verify that 24 is the largest number of deaths? (There seem to be sizes that do not correspond to the 0, 1, 12, 24 in the legend)

H2, H8 Scales do not help answer the question directly (which vessel type has the highest death per passenger ratio), requires mental math

V2: Sunburst

Williams (2018)

Question: Which vessel type had the most cumulative passengers?



V2: Heuristics

Williams (2018)

H1 Vessel types are not contiguous which makes it difficult to count passengers (add from two parts in the display, both landing + interdiction)

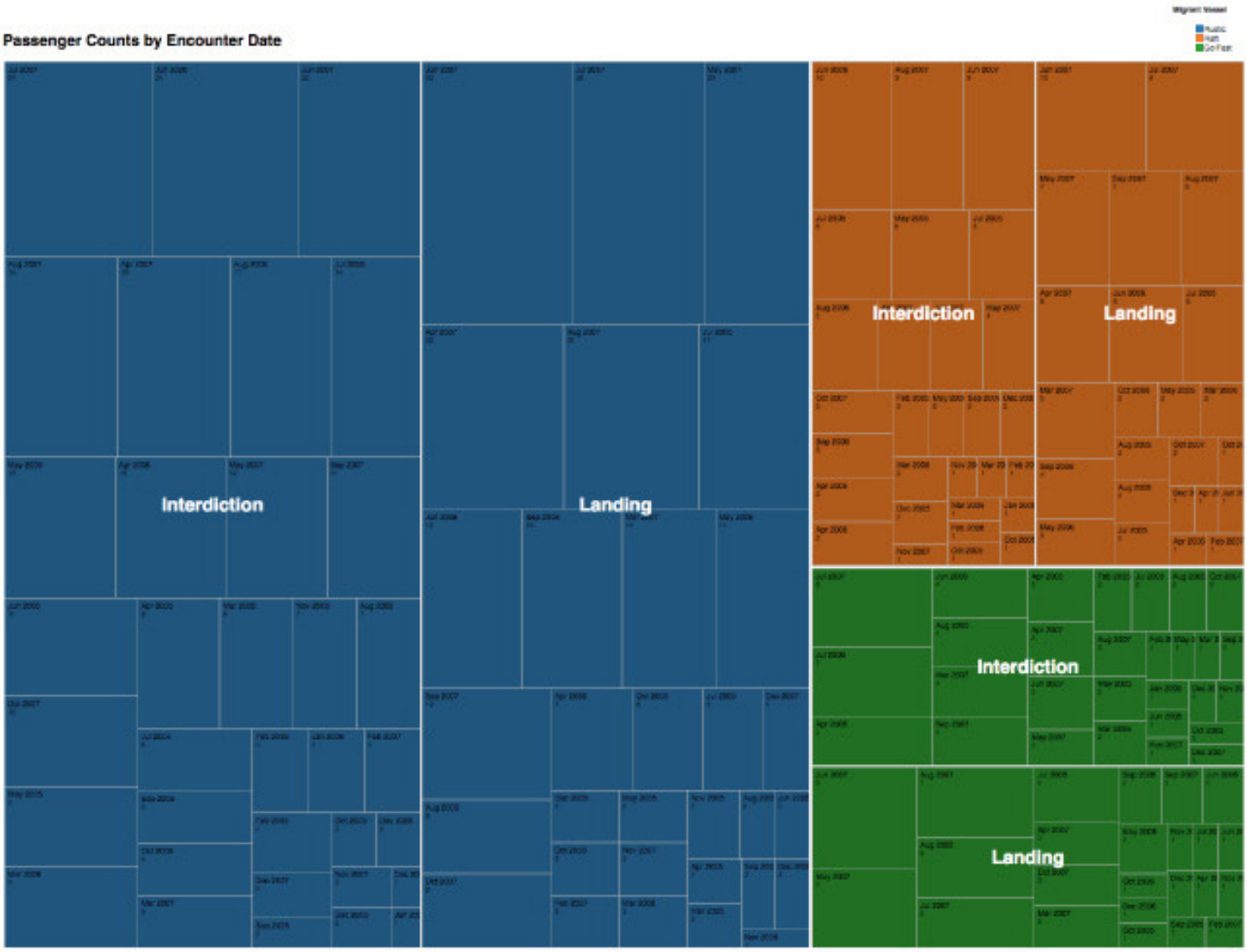
H1 Numbers occlude, are difficult to read in Go Fast vessel type

H9 Blue is inconsistent within a vessel type and the variation does not mean anything (is it that the number of passengers ~ opacity?)

V3: Tree Map

Williams (2018)

Question: Which vessel type had the most cumulative passengers?



V3: Heuristics

Williams (2018)

H1 Labels occlude each other a little bit

H3 Passenger count not labeled

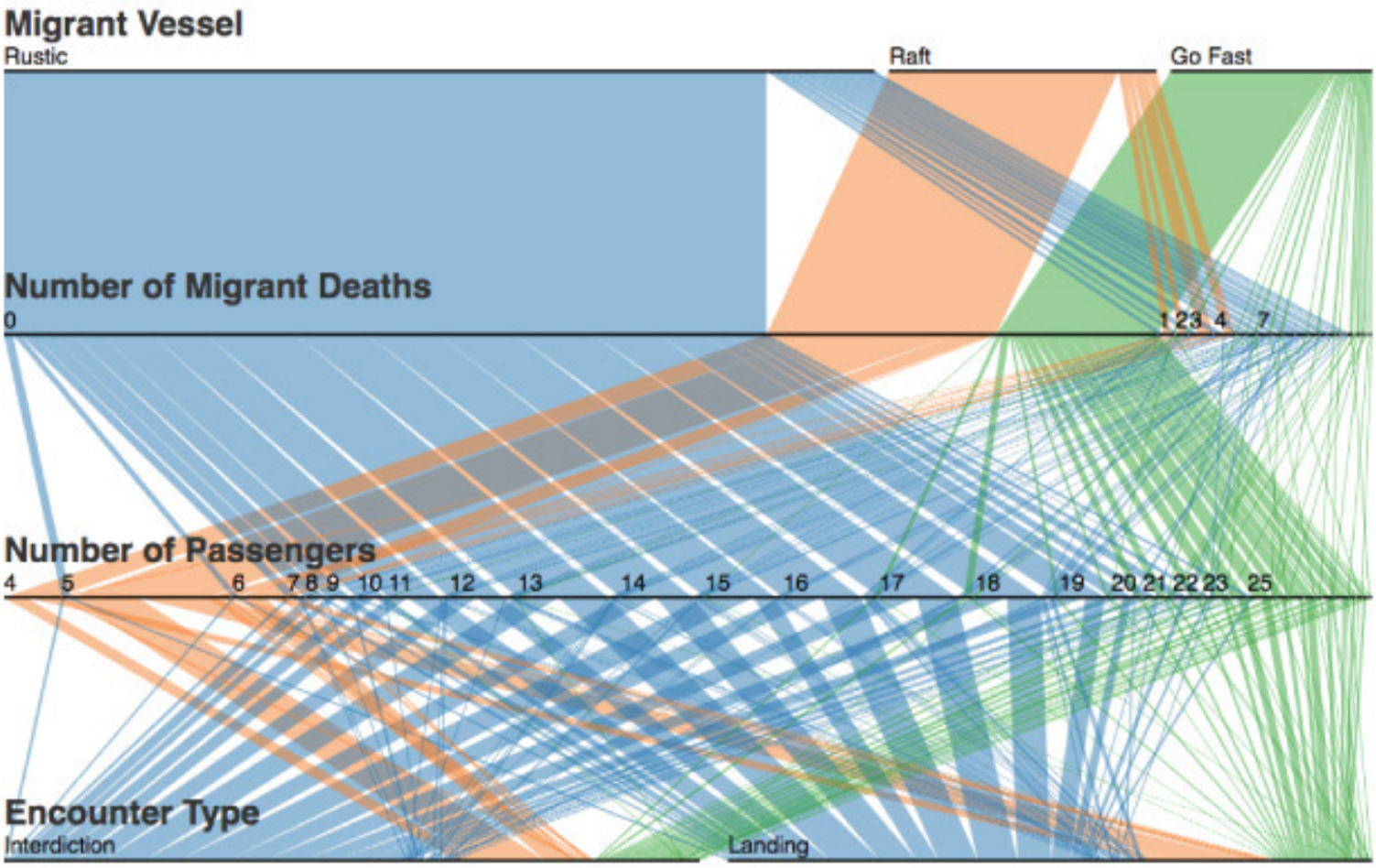
H2 Shape of vessel of same passenger count is not consistent (comparing squares and rectangles)

H8 Have to add to get total passengers

V4: Parallel Sets

Williams (2018)

Question: Which vessel type had the highest death per passenger ratio?



V4: Heuristics

Williams (2018)

H3 Number of deaths impossible to read (only labels 0, 1, 2, 3, 4, and 7, have to guess at the others)

H3 Number of passengers impossible to read (there are not enough labels at far right end)

H1 Connecting Death to Passengers requires following a thin line

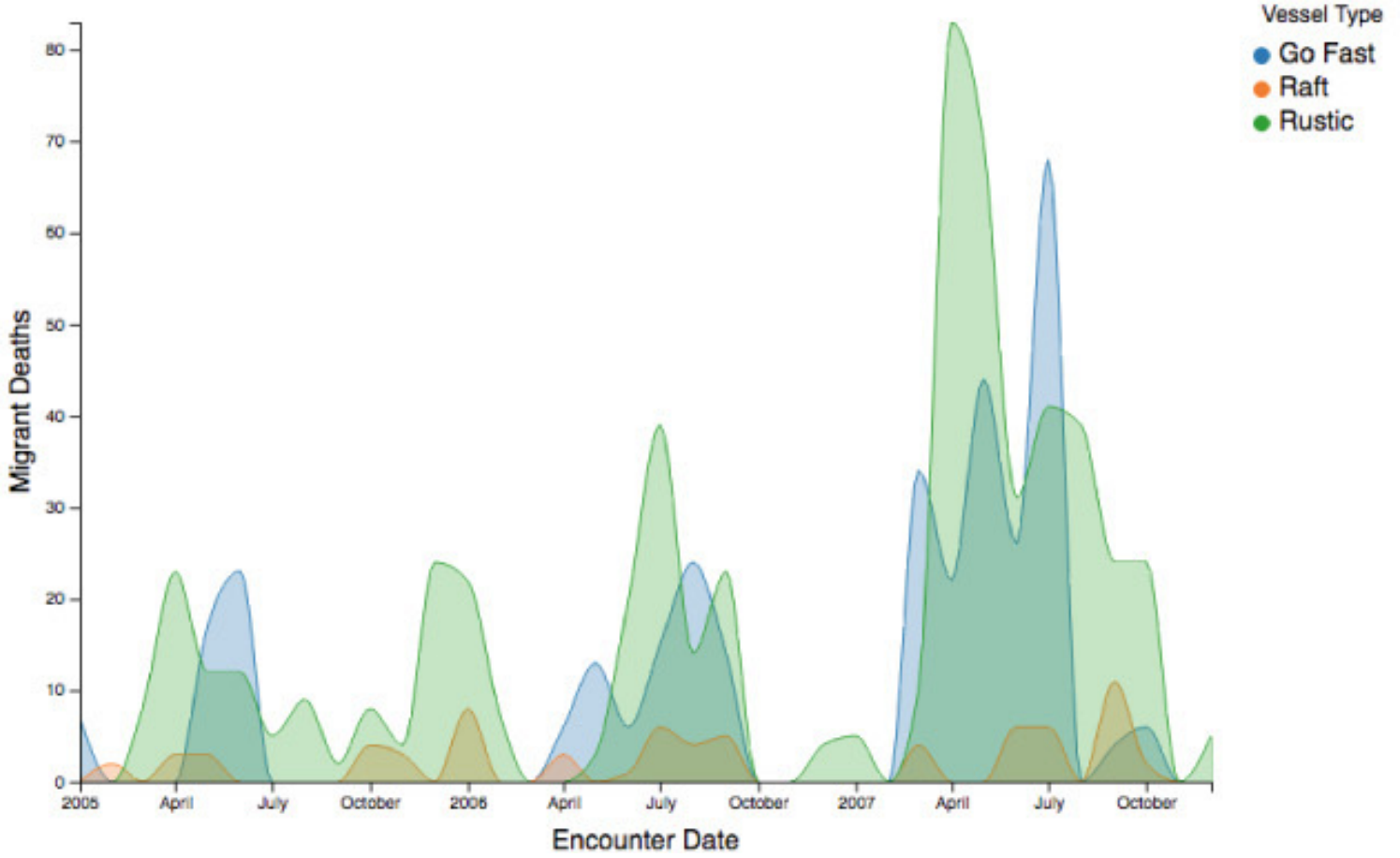
H8 Still need mental math to answer ratio question

H1 Line crossings difficult to follow

V5: Area Plot

Williams (2018)

Question: Which vessel type had the highest cumulative deaths?



V5: Heuristics

Williams (2018)

H8, H6 Requires mental math to get cumulative deaths (sum area under curve)

H6 Labels are not precise enough for exact math

H2 Continuous lines and area makes it difficult to know how to do the addition (i.e., flat portion of raft for April 2005, how many 4 s do we add?)

H2 Viz better suited for relative comparison than precise reading

Tools

In this course...

- `htmlwidgets`
- `shiny`
- Tableau

`htmlwidgets`

A family of R packages for JavaScript-based visualisations

Mechanism

1. Embed `htmlwidget` R code into an Rmarkdown document chunk.
2. Render document to HTML to insert JavaScript code into the result.
3. View the HTML in the browser; JavaScript provides interaction.

. . .

No R in the final product!

Typical functionality

- Query interesting points
- Zoom in on regions
- Rearrange elements for better visibility
- Rotate 3D plots

Advantages and disadvantages

\+ Quick interactive visualisation

\+ End-user doesn't have to run R

. . .

– Limited customisation: controls and UI limited to what's in the library

Some available widget packages

htmlwidgets.org showcase

ggiraph

`plotly`, `metricsgraphics`, `highcharter`, `rbokeh` Interactive graphics using various JS libraries each offering a variety of plots; some can wrap normal `ggplot`s to make them interactive.

`leaflet` Interactive maps and choropleths

`dygraphs` Interactive time series

`visNetwork`, `networkD3` Network data visualisation

`DT` Tabular data

`threejs`, `rglwidget` rotatable 3D

`ggiraph` Extension of htmlwidgets, adds tooltips, onclick events

`shiny`

A toolkit for creating dashboards in R

Model–View–Controller

Wikipedia

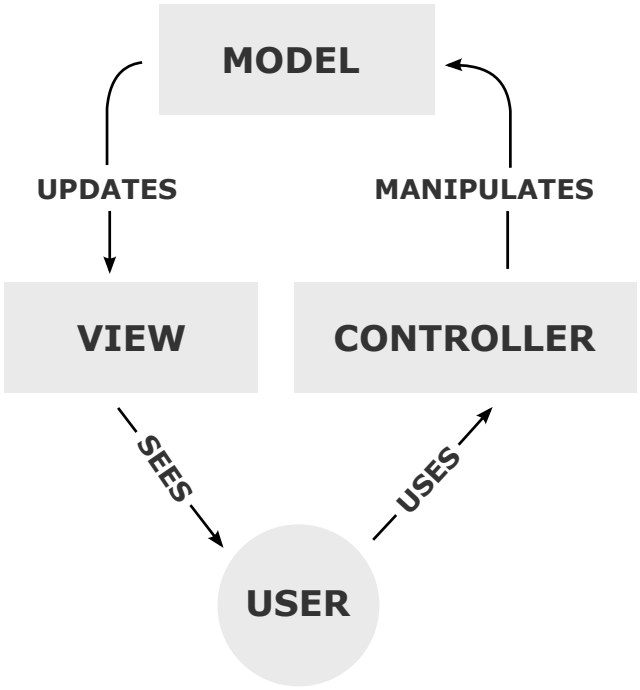
Software components

Model Data structures representing application state (independent of UI)

View Visual representation of the application state

Controller Takes inputs from user

Interactions



Mechanism

Specify:

R environment (Model) data, variables, etc.

`UI` **elements (View and Controller)** sliders, selectors, boxes, buttons, *plots*. Plots can themselves be `htmlwidgets` or other interactive elements.

`server` **code (Interactions)** declarations "reacting" to UI interactions to update the graphics

. . .

- UI delivered via HTML/JavaScript in browser
- Talks to an R session in the background
- Requires an R "server" to be running

Aside: Declarative programming

Imperative programming

Computer,

1. Do this
2. Next do this
3. Next do this

Declarative programming

Computer, here's what I want:

- Goal/constraint
- Another goal/constraint
- Another goal/constraint

. . .

Computer, **you** figure out how to best achieve them.

Declarative programming examples

Often domain-specific.

Structured Query Language (`SQL`) "I want the table rows that satisfy these criteria. Computer, figure out the quickest way to find them."

. . .

`BUGS` / `JAGS` / `STAN` "Here's a specification of conditional distributions between these variables. Computer, figure out how to sample from them."

`HTML` "Here's the text and the semantic and formatting information. Computer, figure out how to render it on the screen."

linear programs "Here's the linear function of x and the set of constraints on x. Computer, figure out how to find the optimum."

. . .

`shiny` "Here are the UI components and how I want the program to react to their inputs. Computer, figure out how to make it work."

Back to shiny

Tiny `shiny` example

```
library(shiny)

# Layout type
ui <- fluidPage(
  # Number input widget, appears in server() as input$n
  numericInput("n", "n", 1),
  # Plot output widget, server() can update it by assigning to output$plot
  plotOutput("plot")
)

server <- function(input, output)
{
  # Put the following plot into the "plot" widget:
  output$plot <- renderPlot({
    par(mar = c(0,0,0,0))
    # Plot some points
    plot(1:input$n)
    # With these dimensions
  }, width = 400, height = 400)
}

# Computer, make it all work.
shinyApp(ui, server)
```

Shiny: advantages and disadvantages

+ Can make use of all capabilities of R and its packages

+ A variety of flexible controls
+ Can embed and interact with `htmlwidgets`
+ R can run on a server or the cloud (e.g., `https://shinyapps.io`)

. . .

– Requires *someone* to run the R back-end
– Can be slow
– Requires writing code for the interaction

Tableau

A proprietary program and framework providing a GUI for creating dashboards.

Mechanism

1. Start the application.
2. Connect it to data source(s).
3. Use drag-and-drop interface to design the dash board.
4. (Optionally) Run the dashboard server (locally, on a server, and/or the cloud) and access the dashboard runs in a browser.

Examples

Tableau Desktop | Some web dashboards

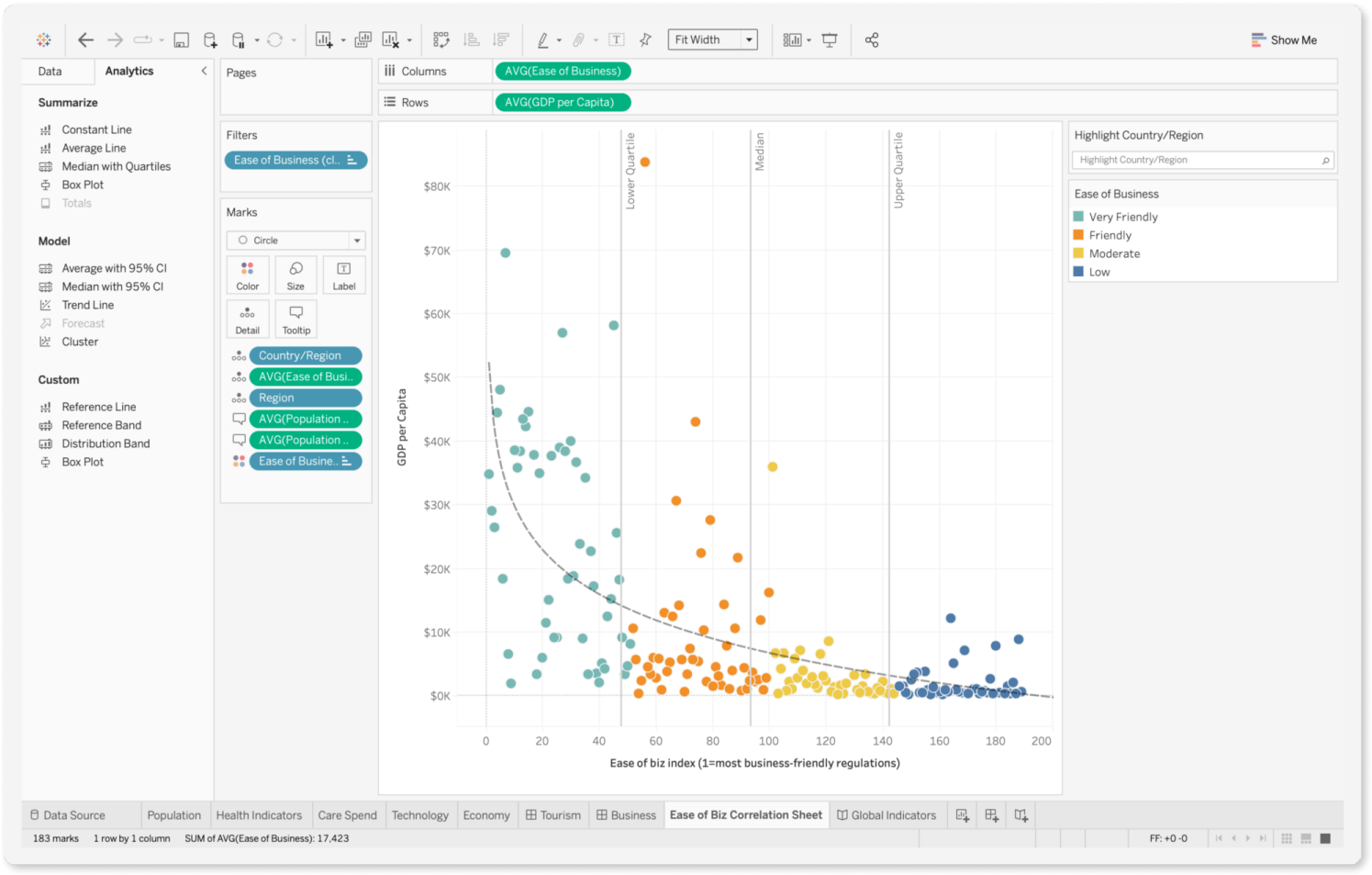


Tableau: advantages and disadvantages

+ Point-and-click interface
+ Integrates with a variety of data sources and processing systems
+ Can handle big datasets well (if back-end can)
+ Public dashboards can be hosted for free (for now)

. . .

– Point-and-click interface
– Proprietary: must pay for non-educational use
– Capabilities limited to what Tableau developers produce

Conclusion

Reminder

- Project proposal due by Friday of Week 8!

Looking forward:

- Week 8: Dashboards (last week relating to project)
- Week 9: Visualisation in machine learning
- Week 10: Graphs for specific contexts (including possibly social networks, time series, animation, text, or other topics)
- Feel free to continue suggesting specific topics for Weeks 9-10

Consultation hours

Extra consultation hours for project discussions for the remainder of the course (Weeks 7 - 10).

- Tuesday, 11 AM - 12 PM (Zoom), 1 PM - 2 PM (Zoom or H13 Anita B Lawrence Centre East 2074).
- Thursday, 1 PM - 2 PM (Zoom or H13 Anita B Lawrence Centre East 2074).
- Masks required for in-person consultation.
- Also available by appointment.
- Hours may be changed in future weeks (announcement will be made).