Hackathon Intro 2024

John Novembre
Department of Human Genetics
Department of Ecology & Evolution
University of Chicago

Motivation:

- Theme this year : Interactive data visualization
- Goals:
  - Code intensively for 2 days
  - Learn new skills
  - Enjoy interacting with colleagues
  - Expose yourself to new data types
  - Learn / practice reproducible programming steps

Outline for the hackathon

- Day 1:
  - Intro to Hackathon (John, Liz, Kellen)
  - Icebreaker - (Names and Interests/ Goals for hackathon)
  - Break
  - Breakout: Form into groups - and brainstorm ideas for visualizations
  - Regroup and share design ideas
  - Tutorial on structuring a project (Sai)
  - First round of hacking…
  - Mid-afternoon: GitHub intro/ refresher (optional)
  - Status checks & Feedback at 4:30pm
- Day 2:
  - 9am :Share progress & goal setting
  - Open hacking
  - 11:30-Noon: Discussion
  - Open hacking
  - 4pm: Closing presentations & prizes

Motivation:

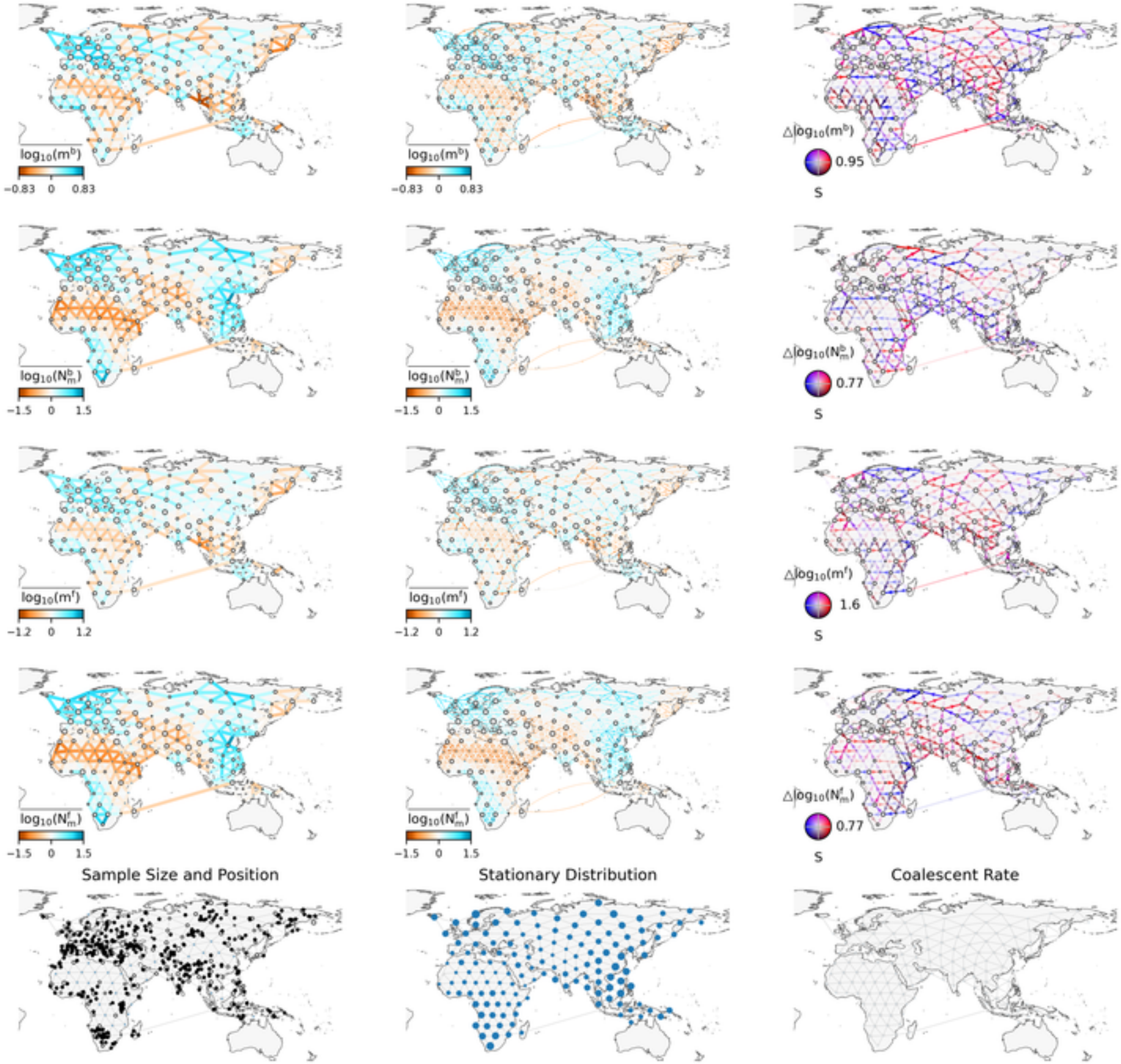
- Why interactive data visualization?
  - Plotting for publication is obviously important
  - BUT plotting for learning / understanding / exploring your questions is arguably MORE important

Motivation:

- When plotting for exploration — you are not limited by page size, by limits of static plots. There is room for much more…

Example:

- From my postdoc Hao’s work…
    - Same analysis results with 3 different outputs (3 columns) shown with 4 different methods (first 4 row)
    - Plus an additional row of information
- Obviously too much for publication - but very useful for us to automatically generate these each on many datasets to see how the method behaves



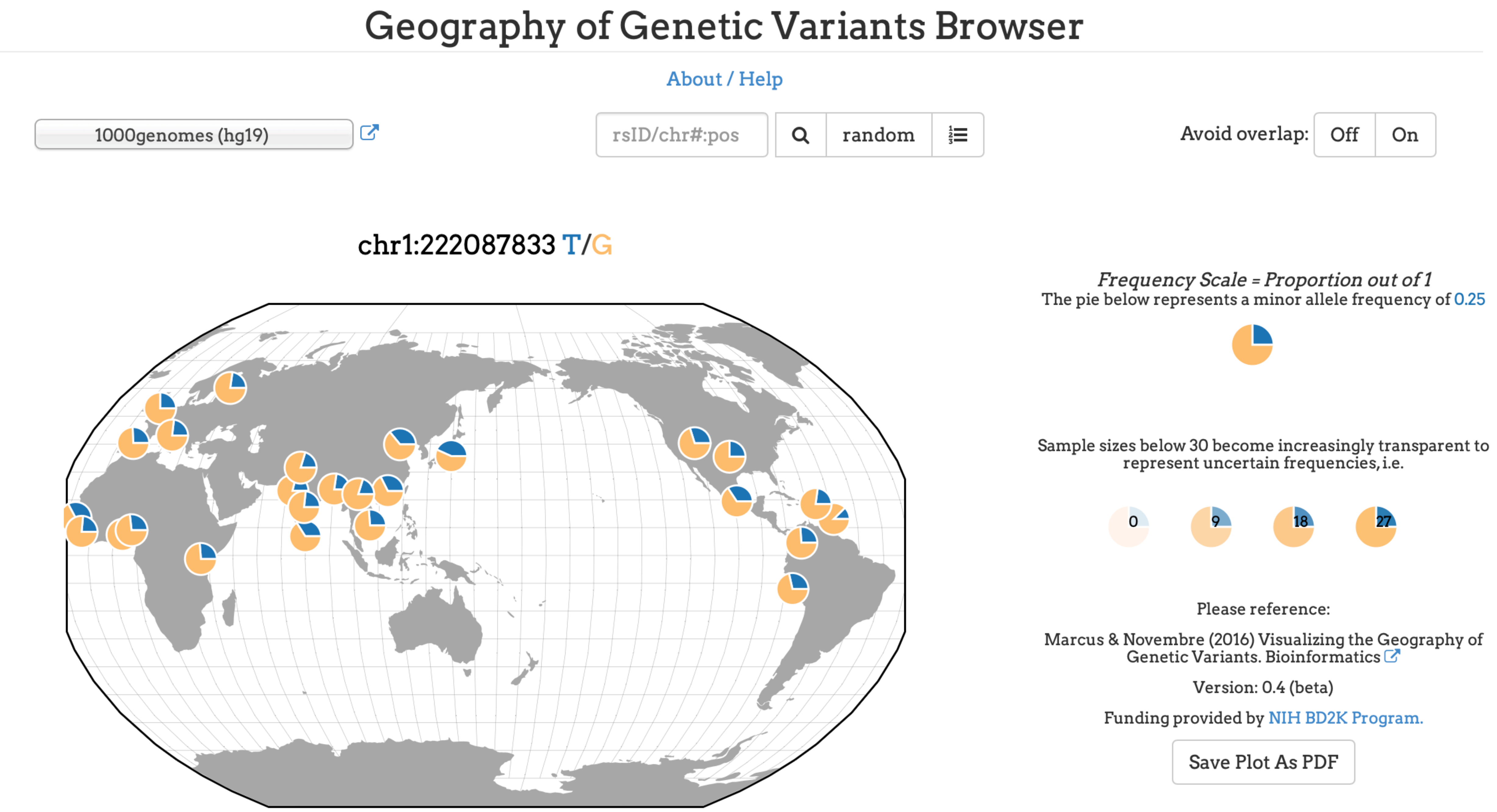
Sample Size and Position

Stationary Distribution

Coalescent Rate

Interactive approaches…

- Can allow you to interface and interrogate data…
- The most common problem — In a scatterplot - there is an outlier. “Can you tell me what gene/SNP that outlier is?” “Sure let me get back to you”
- Another problem - “But can we see the results for Gene Z?”



Geography of Genetic Variants Browser

Frequency Scale = Proportion out of 1
The pie below represents a minor allele frequency of 0.25

Sample sizes below 30 become increasingly transparent to represent uncertain frequencies, i.e.

Please reference:

Marcus & Novembre (2016) Visualizing the Geography of Genetic Variants. Bioinformatics

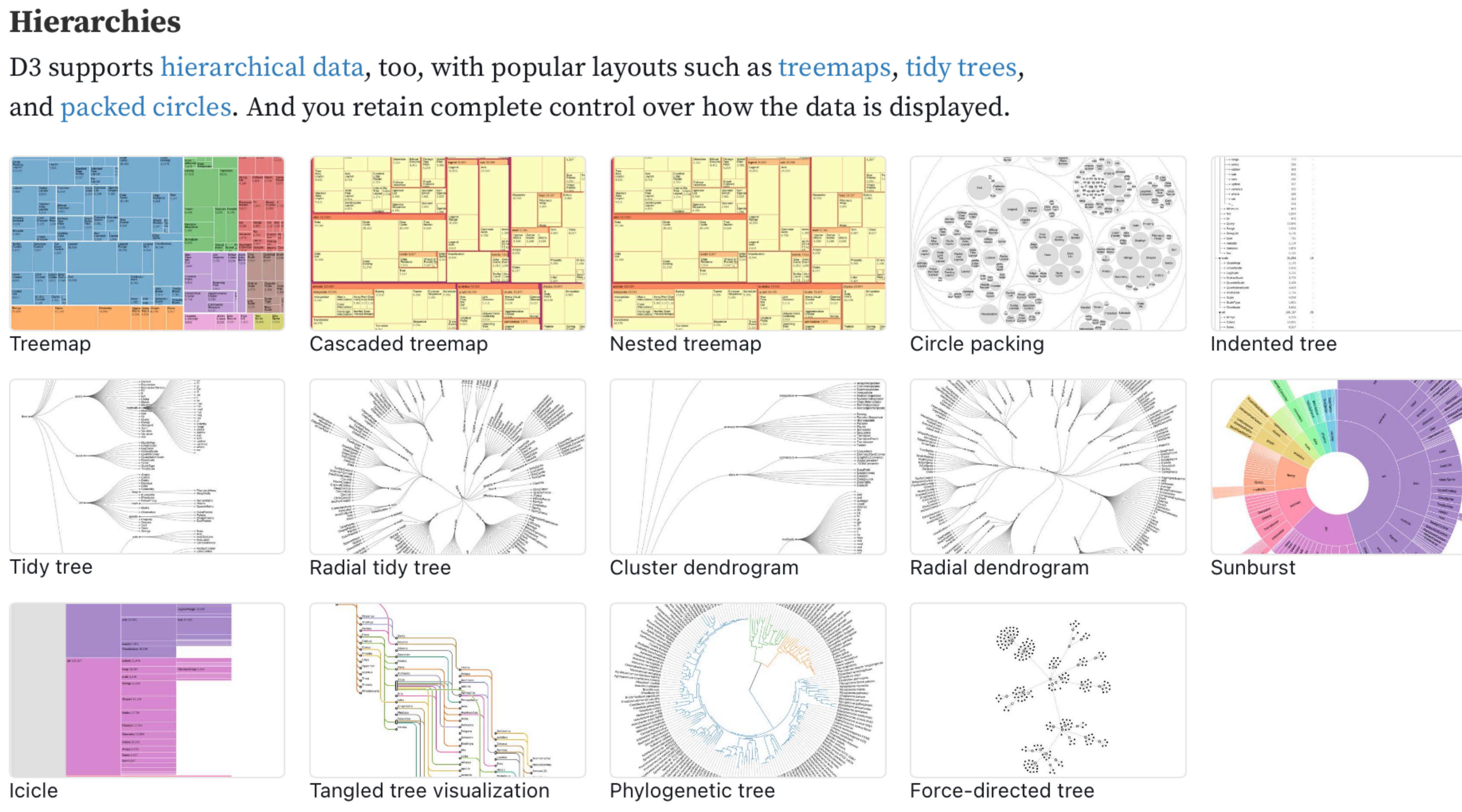
Version: 0.4 (beta)

Funding provided by NIH BD2K Program.

popgen.uchicago.edu/ggv/

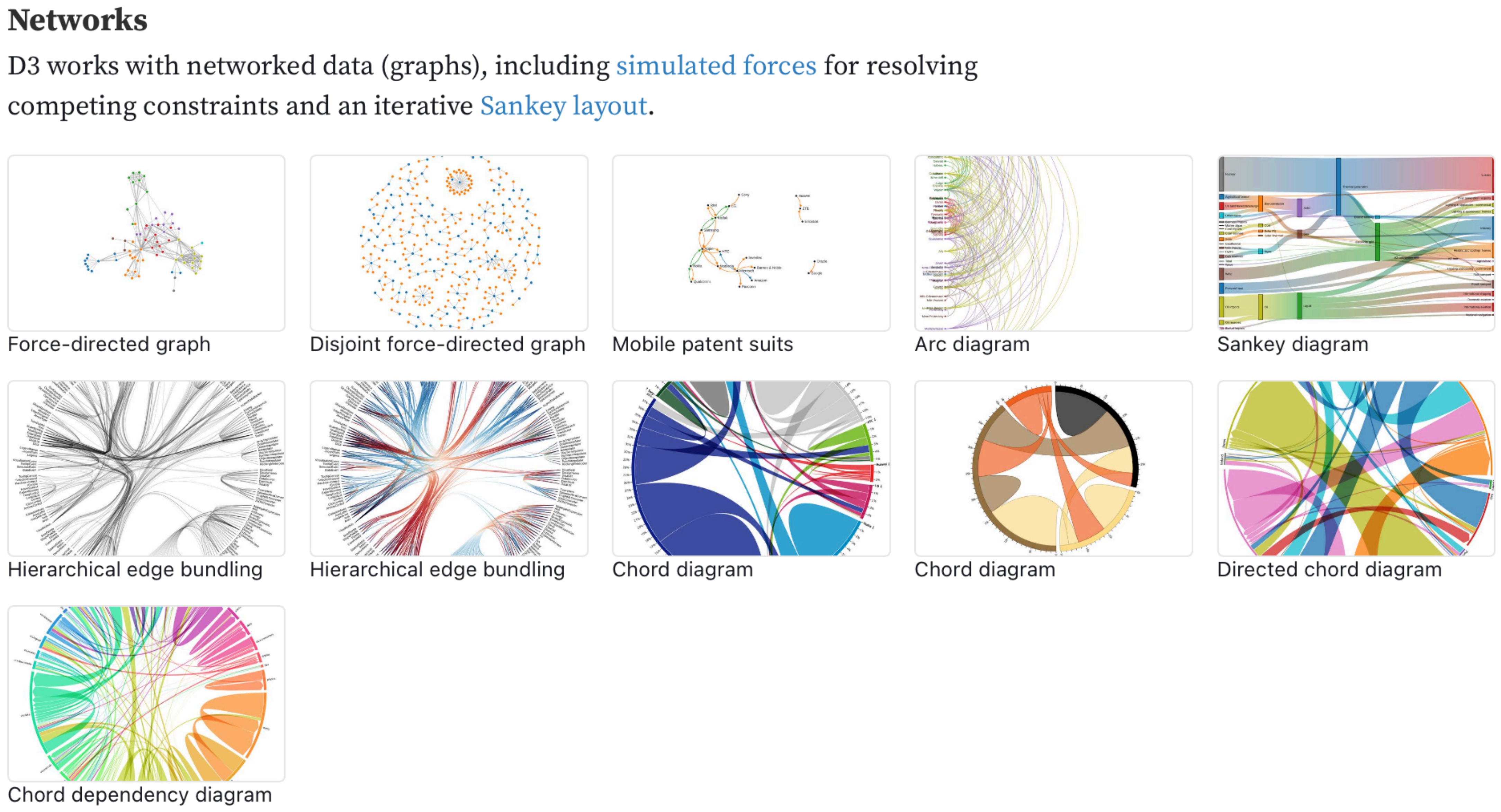
Tools for interactive plotting

- Entry-level:
  - Add-on code libraries to existing languages:
    - Shiny for R
    - Plotly for R and python
- Intermediate:
  - Dash - (dash.plotly.org)
  - D3 library for javascript (d3js.org)
- Advanced (Overkill?):
  - OpenGL library for C/C++ for 3-D programming
  - “Unreal Engine” - used in 3-D game design



Hierarchies

D3 supports hierarchical data, too, with popular layouts such as treemaps, tidy trees, and packed circles. And you retain complete control over how the data is displayed.



Networks

D3 works with networked data (graphs), including simulated forces for resolving competing constraints and an iterative Sankey layout.

Some key ideas

- To be interactive - the core programming code runs **a perpetual loop** often implicit in the code (i.e. you don’t write the code for the loop; it’s handled automatically)
    - The system checks for events (“hover over”, “click”, “button press”)
    - Events trigger “callbacks”
        - A callback function is called automatically by the system when the event happens
- The main new plotting challenge then is:
    - Learning what **events** you can have your plot respond to
    - Setting up clever **callbacks** that trigger actions that are helpful to you

Some key ideas

- To be interactive - the core programming code sets up a **server** running **a perpetual loop** (i.e. you don’t write the code for the loop; it’s handled automatically)
  - After some initialization, the loop starts
  - The system checks for events (“hover over”, “click”, “button press”)
  - Events trigger “callbacks”
    - A callback function is called automatically by the system when the event happens
- The main new plotting challenge then is:
  - Learning what **events** you can have your plot respond to
  - Setting up clever **callbacks** that trigger actions that are helpful to you

Some basic concepts of webpage design

- Many visualization tools (e.g. **Dash**) are based on concepts of html web page design
- A few key concepts are helpful:
  - A **div** object :
    - A division of content (e.g. a section of a webpage).
    - Each div can be styled differently / have size and format
  - Challenge:
    - **Define your div objects** and their styles
    - Insert code for **initializing the content within each div object** (e.g. a plot)
    - Set up your **callbacks** to **update the plots** in reaction to the **events**

Thankfully…

- Plotly and shiny have a lot of built in functionality where you do not need to write the callback functions even.
- Kellen will demo shortly…

Outro:

- This should feel like a challenge
- But hopefully everyone will leave with template code for making at least one new interactive plot that improves their research

- And to celebrate hard work - there will be prizes for:
  - Team with most compelling interactive visualization using the included dataset
  - Team with most compelling interactive visualization using any genetics-related dataset
    - Criteria:
      - **Layout**, **colors**, **informativeness**, **impact**
  - There will be honorable mentions and awards for effort/enthusiasm/helpfulness