ABiL
Applied BioInformatics Laboratory

Predictive webservers & online bioinformatics resources

Aroon Chande

Lecture 23

Thursday, April 5, 2018

Outline

- Computational genomics class
- Genome database
- Basics of webserver & database
- Predictive webservers, a case study
  - vibriocholera.com
  - GADGET – The Global Distribution of Genetic Traits webserver

Questions

- Are we doing the typing tool or browser or both?
- What are the technologies we should be looking apart from those used by previous batches?
- How can we incorporate scalability for so many genomes ?
- Will it be a simplified version of the work of 4 other groups? What parts of the previous workflow is necessary ?
- How do we incorporate(if we have to) all the assembly, gene prediction and annotation steps into the application?

Questions

- Are we doing the typing tool or browser or both?

Both

- What are the technologies we should be looking apart from those used by previous batches?

MySQL, PHP, HTML/CSS/Jquery, NodeJS

- How can we incorporate scalability for so many genomes ?

Comparative's objective

- Will it be a simplified version of the work of 4 other groups? then which part of the previous workflow is necessary ?

Which parts – the output

- How do we incorporate(if we have to) all the assembly, gene prediction and annotation steps into the application?

Downloadable files

Presentation Assumption

What do we understand:

- Sequencing and computational genomics process
- The output from different groups

What we do not understand:

- The end goal
- Database (DBMS) and web service technologies

Outline

- Computational genomics in years past
- Genome database
- Basics of webserver & database
- Predictive webservers, a case study
    - vibriocholera.com
    - GADGET – The Global Distribution of Genetic Traits webserver

Computational Genomics Class

Previous years:

- Sequencing reads from 10’s of organisms within a genus/species
- Biological questions to be answered
  typically some type of phenotype to genotype relationship
- Assembly → Prediction → Annotation → Comparative + Browser

This year:

- Sequencing reads from 100’s of organisms from multiple species
- Biological question – What makes a strain heteroresistant?
  Identifying some genotypic feature to predict heteroresistance

Computational Genomic Class

What has changed?

Numbers: 25 (2014) → 50 (2015) → 140 (2016) → 50 (2017) → 258 (2018!)

Comparative group’s scope & approach:

- Scope – phenotype-genotype correlation
- Approach – more algorithmic

Expected output from comparative:

- A predictive scheme for heteroresistance

Computational Genomics Class

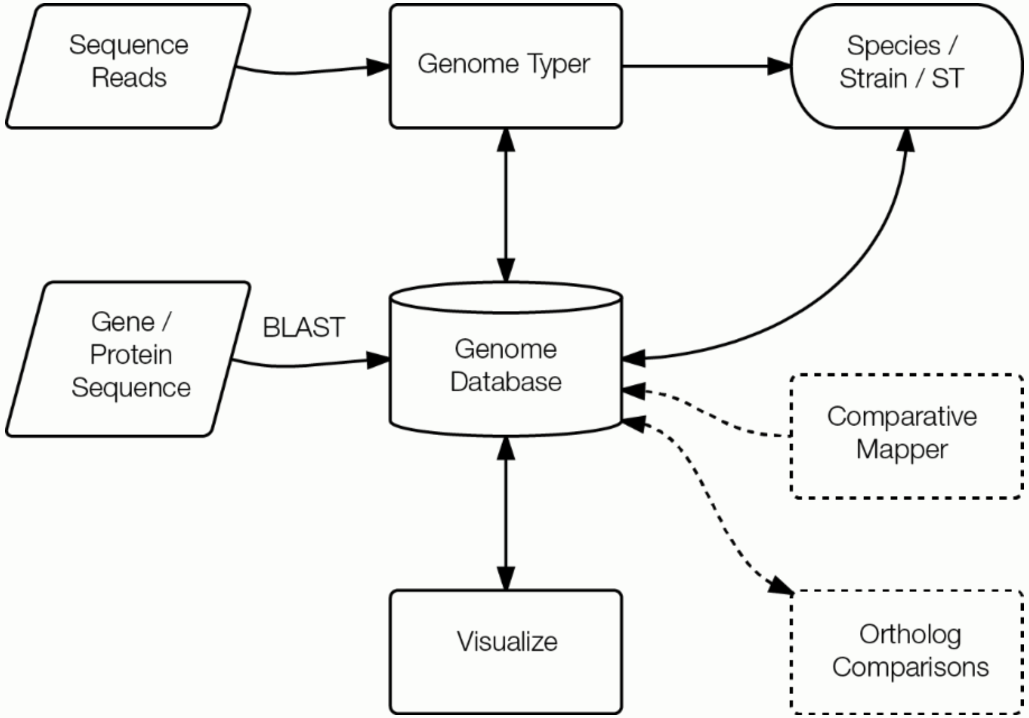
From an end user's perspective:

- Upload sequence reads & identify the organism (based on the database)
- Learn more about the organism identified (basic statistics and genomic features)

What does this all of this mean for Browser group?

- *Primary objectives*: 1) Implement the typing method, 2) Implement a genome database/browser?
- *Secondary objectives*: Add additional search tools and comparative features

Computational Genomics Class



Outline

- Computational genomics class
- Genome database
- Basics of webserver & database
- Predictive webservers, a case study
  - vibriocholera.com
  - GADGET – The Global Distribution of Genetic Traits webserver

Database

- Organized collection of data
- Can be flat files to more sophisticated systems
- What was wrong with flat files? Why did we require advanced systems?

Advantages of a Database

- Aggregates data
- Organizes data
- Ease of management
- Facilitates searches
- Facilitates data mining
- Facilitates sharing
- Provides extensibility
- Provides non-redundancy

…

Genome Database

- A database specializing in genomic information

- What is different from other databases:
    - Data content and structure
    - Users
    - Querying type – sequence and sequence ranges
    - Data hierarchy
    - Data linkage – within and link outs
    - Information presentation

Q: Okay so how do I make a database?

What do you require?

- Data (and knowledge of what that data is)
- A Database Management System (DBMS)
- A database developer/administrator
- A frontend designer

What do you require?

- Data (and knowledge of what that data is)

Output from different groups

- A Database Management System (DBMS)

MySQL hosted at compgenomics server

- A database developer/administrator

Developer – one of you. Administrator – Troy Hilley

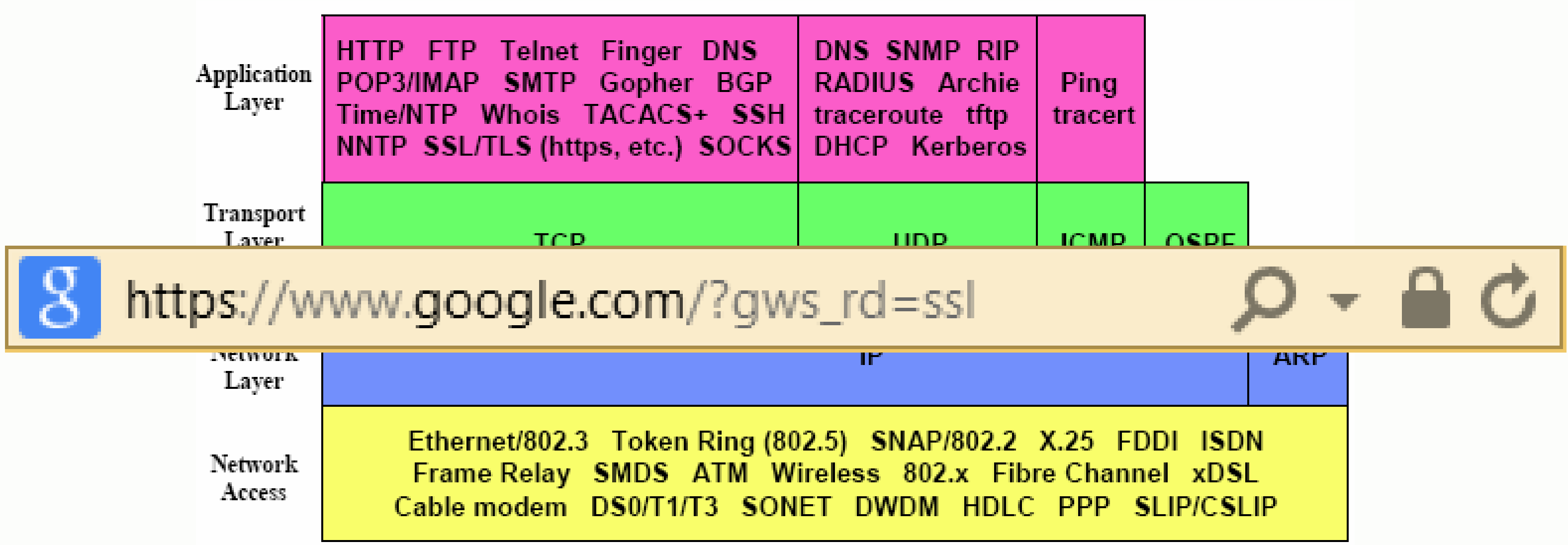
- A frontend designer

One of you

Outline

- Computational genomics class 2015
- Genome database
- Basics of webserver & database
- GMOD

Internet: Protocols


Application Layer
HTTP FTP Telnet Finger DNS
POP3/IMAP SMTP Gopher BGP
Time/NTP Whois TACACS+ SSH
NNTP SSL/TLS (https, etc.) SOCKS
DNS SNMP RIP
RADIUS Archie
traceroute tftp
DHCP Kerberos
Ping
tracert
Transport Layer
TCP
UDP
ICMP
OSPF
https://www.google.com/?gws_rd=ssl
Network Layer
IP
ARP
Network Access
Ethernet/802.3 Token Ring (802.5) SNAP/802.2 X.25 FDDI ISDN
Frame Relay SMDS ATM Wireless 802.x Fibre Channel xDSL
Cable modem DS0/T1/T3 SONET DWDM HDLC PPP SLIP/CSLIP

Internet protocol suite

HTTP (HTTPS)

- Hyper Text Transfer Protocol (Secure)
- Hyper Text = Multimedia (Images, Videos)
- Governed by HTML = Hyper Text Markup Language

Markup Language

- Different from programming languages
- HTML ≡ Microsoft® Word
- Provides structure to content
- Tags!

Markup Language

- XML = Extensible Markup Language

- HTML follows predefined tags, XML follows custom defined tags

- Widely used in bioinformatics for transferring data

- Textual, both human and machine readable

Internet: Protocols

- FTP (File Transfer Protocol) – Another important protocol
- Designed for the transfer of files over a network
- Secure variant is more commonly used (SFTP)
- Most of the big databases in bioinformatics provide FTP for file downloads

Database Management System (DBMS)

Q: What is a DBMS?
A: Collection of tools to help in creating, storing, modifying and extracting information from a database

Q: Why not something like Excel?
A: Issues with scalability, consistency, redundancy, simultaneous access and within data connectivity

Database Management System (DBMS)

- Scalability

  Spreadsheets/flat files store data in a single file. As data grows, basic operations becomes unviable.

- Consistency

  The data needs to be in the same format for pattern searching. Difficult to achieve in spreadsheets, not possible in flat files.

Database Management System (DBMS)

- Redundancy

  Redundant information uselessly increase data size and may interfere with pattern searching.

- Simultaneous access

  Limited simultaneous access in spreadsheets/flat files.

Database Management System (DBMS)

- Within data connectivity

  Difficult to maintain in spreadsheets and flat files.

- E.g. hypothetical database for outbreaks. Three types of data:
  - Strain information
  - Hospital information
  - CDC personnel information

- If any data requires update, every single one needs to be updated!

Database Management System (DBMS)

DBMS was specifically designed to resolve these issues

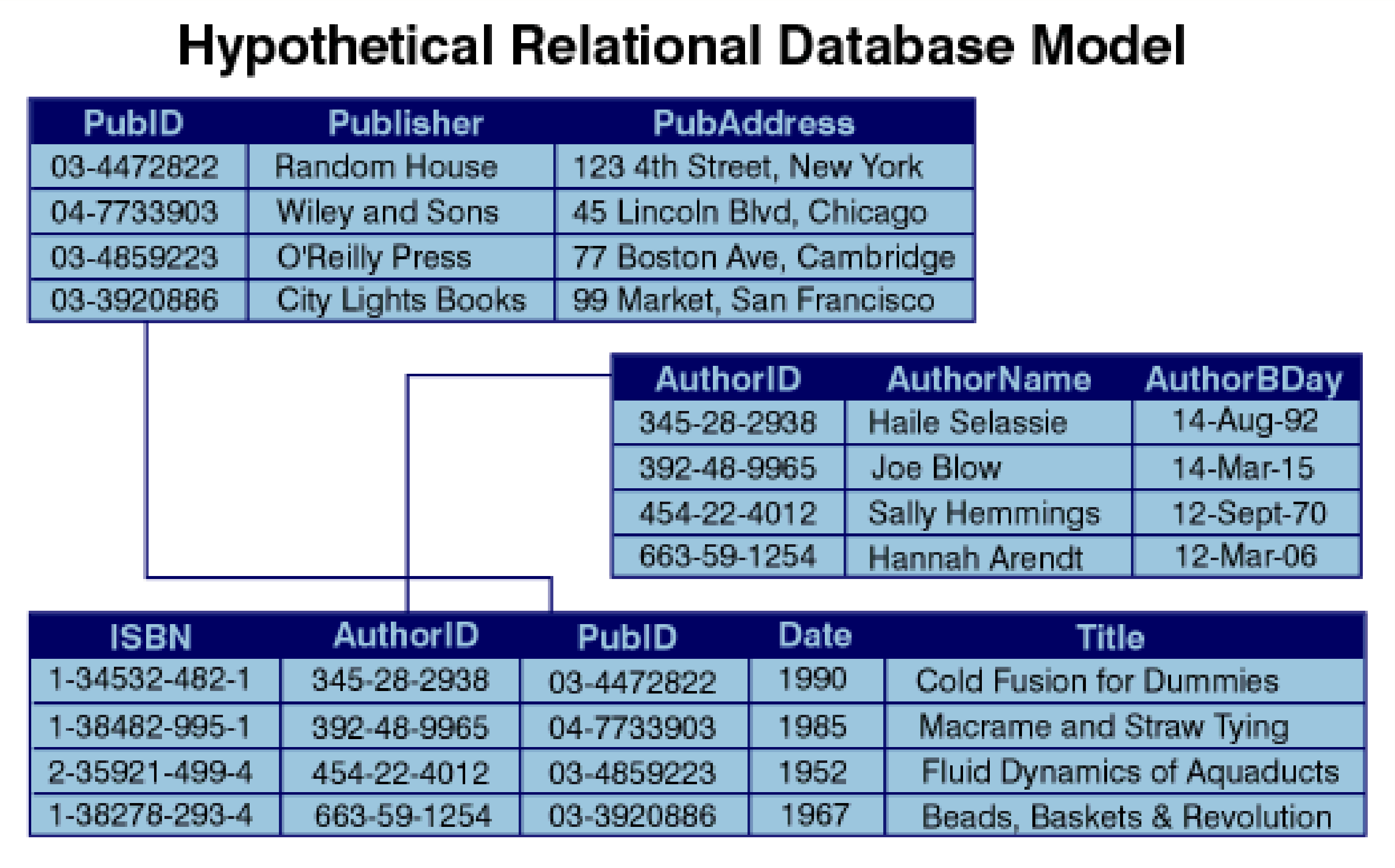
Q: What are my options for a DBMS?

A: Many!

Relational DBMS: MySQL, Oracle, Microsoft Access, Postgre SQL

Non-relational: MongoDB, CouchDB, Google Spanner

Relational DBMS



Hypothetical Relational Database Model

PubID	Publisher	PubAddress
03-4472822	Random House	123 4th Street, New York
04-7733903	Wiley and Sons	45 Lincoln Blvd, Chicago
03-4859223	O'Reilly Press	77 Boston Ave, Cambridge
03-3920886	City Lights Books	99 Market, San Francisco

AuthorID	AuthorName	AuthorBDay
345-28-2938	Haile Selassie	14-Aug-92
392-48-9965	Joe Blow	14-Mar-15
454-22-4012	Sally Hemmings	12-Sept-70
663-59-1254	Hannah Arendt	12-Mar-06

ISBN	AuthorID	PubID	Date	Title
1-34532-482-1	345-28-2938	03-4472822	1990	Cold Fusion for Dummies
1-38482-995-1	392-48-9965	04-7733903	1985	Macrame and Straw Tying
2-35921-499-4	454-22-4012	03-4859223	1952	Fluid Dynamics of Aquaducts
1-38278-293-4	663-59-1254	03-3920886	1967	Beads, Baskets & Revolution

Figure from http://www.ibm.com/

Relational DBMS

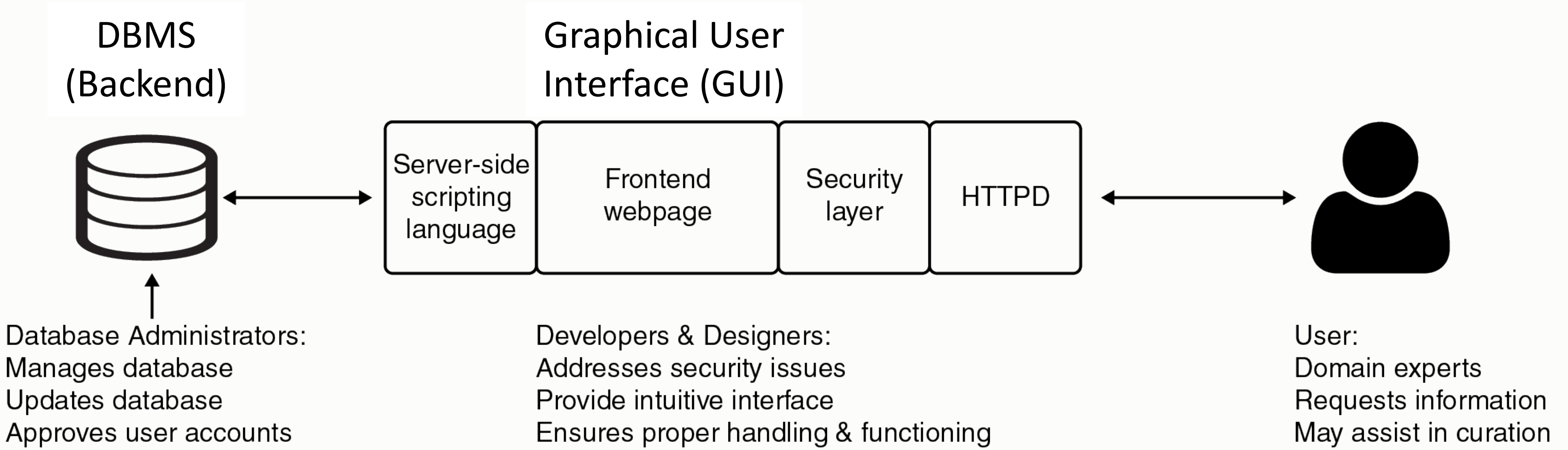
- Are defined in Structured Query Language (SQL)

- Looks like this:

```
CREATE TABLE STATION (ID INTEGER PRIMARY KEY, CITY CHAR(20), STATE
CHAR(2), LAT_N REAL, LONG_W REAL);
```

```
INSERT INTO STATION VALUES (13, 'Phoenix', 'AZ', 33, 112);
```

Database Basics



Database Administrators:
Manages database
Updates database
Approves user accounts

Developers & Designers:
Addresses security issues
Provide intuitive interface
Ensures proper handling & functioning

User:
Domain experts
Requests information
May assist in curation

Webserver Basics

- Webserver – any computer connected to the internet that provides some sort of service
- These services are provided through specific protocols. E.g. HTTP, FTP
  HTTP : Hypertext transfer protocol, FTP : File transfer protocol
- A special software on the server facilitates this communication (answers the request) – HTTPD (HTTP Daemon) i.e., the web server
- Most widely used web servers – Apache and NGINX

Security

Two levels:

- Security at access to the server

  Responsible: Sys-admin and frontend designer/developers

- Security at access to the database

  Responsible: Database administrators

Frontend

- Your normal webpage
- HTML – Hypertext markup language
- Styling – CSS (Cascading style sheets)
- More library functions – Javascript/JQuery

Server-side scripting

- Special type of programming language
- Executes on and by the webserver. Can’t run locally.
- E.g. PHP (originally: Personal homepage. Now: Hypertext preprocessor),
  JSP, Perl via CGI, R, Python
- I recommend PHP for notives– easy to pick, constructs very similar to Perl/Java/C

Task scheduling

- You webservers will need to respond to both short and easy tasks and long and difficult tasks
- Short and easy tasks can be served **synchronously**
- Long and difficult tasks should probably be done **asynchronously**

Synchronous tasks

- Synchronous tasks are blocking – the webserver can't do anything until its finished a task

- Serving web pages is (relatively) easy and can be done on first come, first served basis
  - Its unlikely you'll get so many visitors that they'll have to wait a long time for the server to handle their requests

- Short computational tasks (like generating a plot) can frequently be done synchronously, especially if you have a multi-threaded web server

Asynchronous tasks

- Asynchronous tasks are non-blocking – the webserver keeps doing other things while it **waits** for an async task to finish

- Tasks that take a noticeable amount of time to complete (seconds to minutes+), and therefore prevent a webpage from being [partially] rendered should be done asynchronously

- Async tasks use **promises**, or stand-ins for the eventual result, while your task is being run

Outline

- Computational genomics class
- Genome database
- Basics of webserver & database
- Predictive webservers, a case study
  - vibriocholera.com
  - GADGET – The Global Distribution of Genetic Traits webserver

My preferred tooling

- HTML and R Shiny based frontend

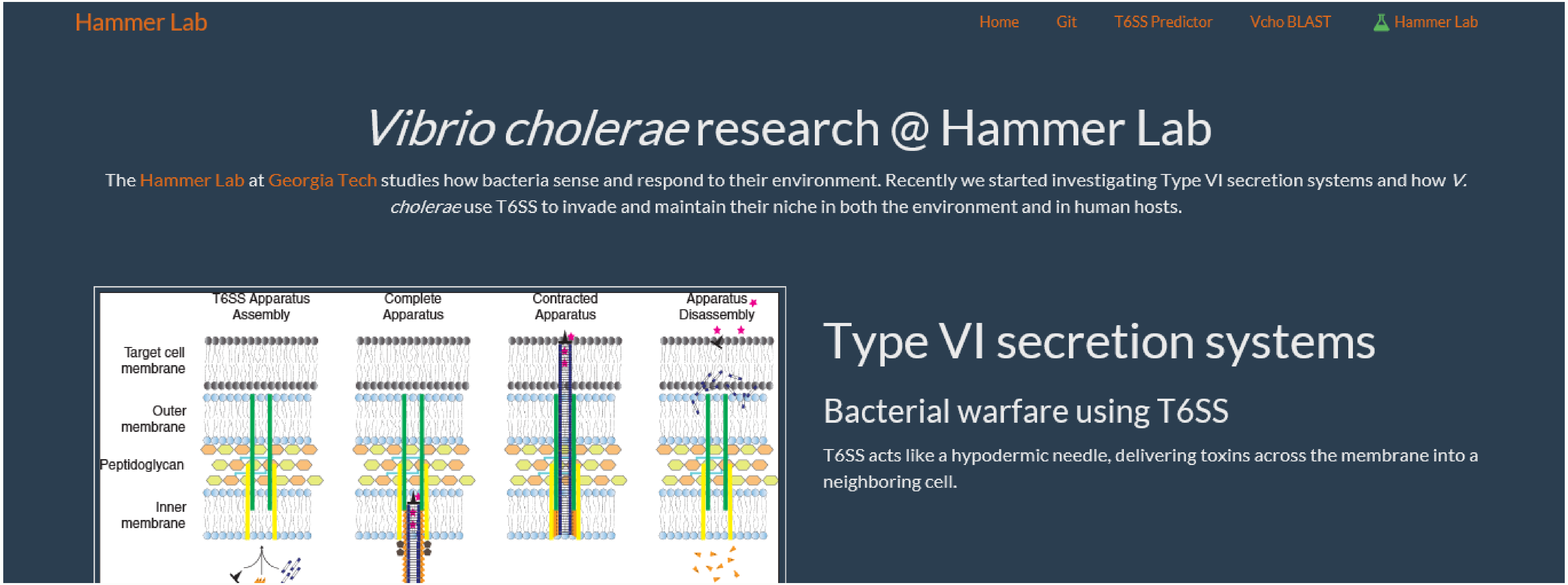
- R + Javascript for visualization and data presentation
  - ggplot2 (plots), DT (interactive tables), leaflet (maps), custom widgets, D3.js

- Perl and Python scripts that handle the bulk of the computational tasks

- SQLite and flat-file databases for the backend data store

- Use of asynchronous tasks for computational jobs

vibriocholera.com



Vibrio cholerae research @ Hammer Lab

The Hammer Lab at Georgia Tech studies how bacteria sense and respond to their environment. Recently we started investigating Type VI secretion systems and how *V. cholerae* use T6SS to invade and maintain their niche in both the environment and in human hosts.

Type VI secretion systems

Bacterial warfare using T6SS

T6SS acts like a hypodermic needle, delivering toxins across the membrane into a neighboring cell.

vibriocholera.com

- My first predictive webserver and its run on hardware I own

- Tooling:
  - R, Perl, Ruby, Go and Elixir with clever proxying behind NGINX
  - Continuous integration and automated updates

- I did a lot of things wrong…
  - Like no async tasks and the T6SS prediction can take 10+ mins
- But it was a great learning experience

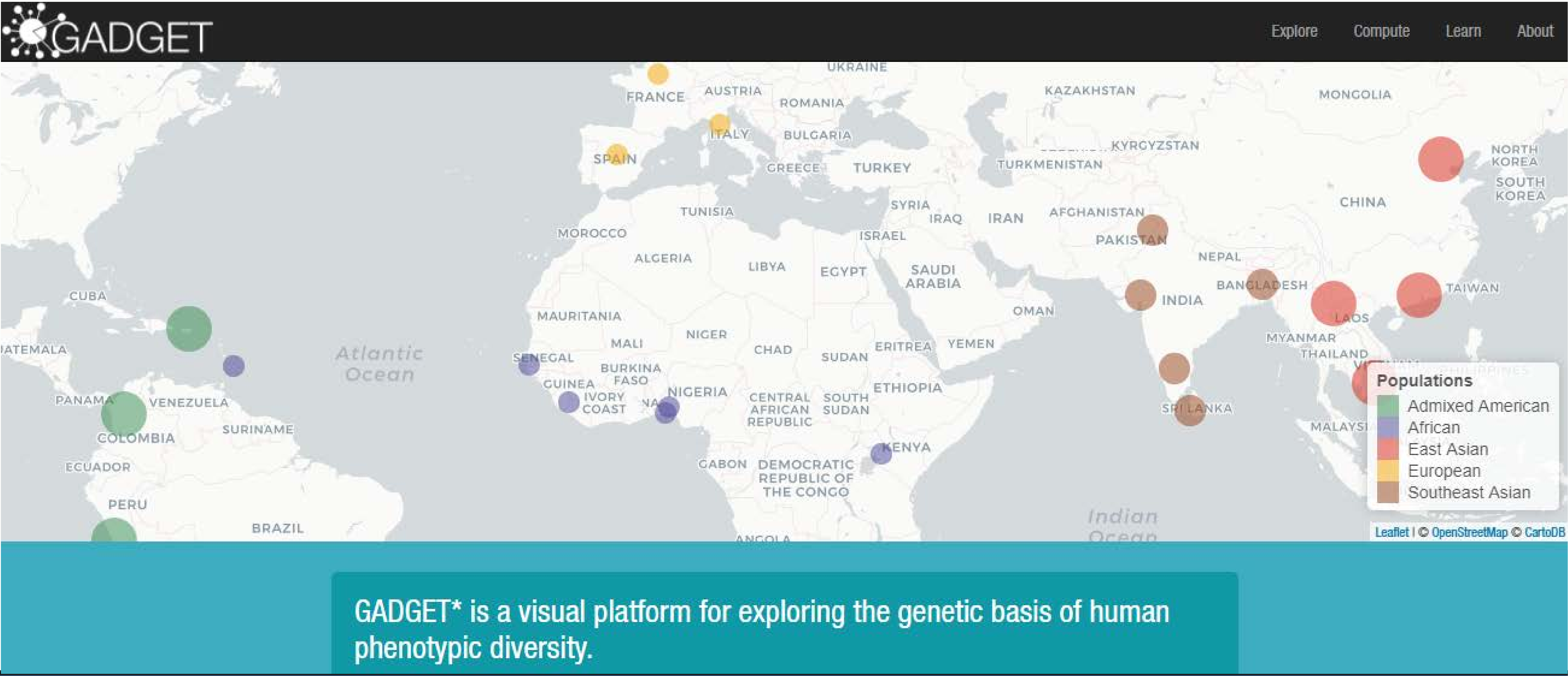
vibriocholera.com

- Hosted on two servers, one in Kansas City, Kansas and one in Paris, France

- Website code (front and backend) is hosted in a Git repository
  - https://git.vcholerae.com/arch/vibriocholera.com

- When a change is pushed to the repository, the code is automatically tested and then deployed onto the webservers
  - Supports pushing to the public production webservers and to a private development webserver

GADGET



GADGET* is a visual platform for exploring the genetic basis of human phenotypic diversity.

GADGET

- Not quite predictive, but allows you to analyze and visualize data

- Tooling:
  - R, Perl, Python behind Apache
  - SQL database of genotype and phenotype data

- Compute-heavy tasks are asynchronous
  - And tasks that take a *really* long time are done non-interactively offline