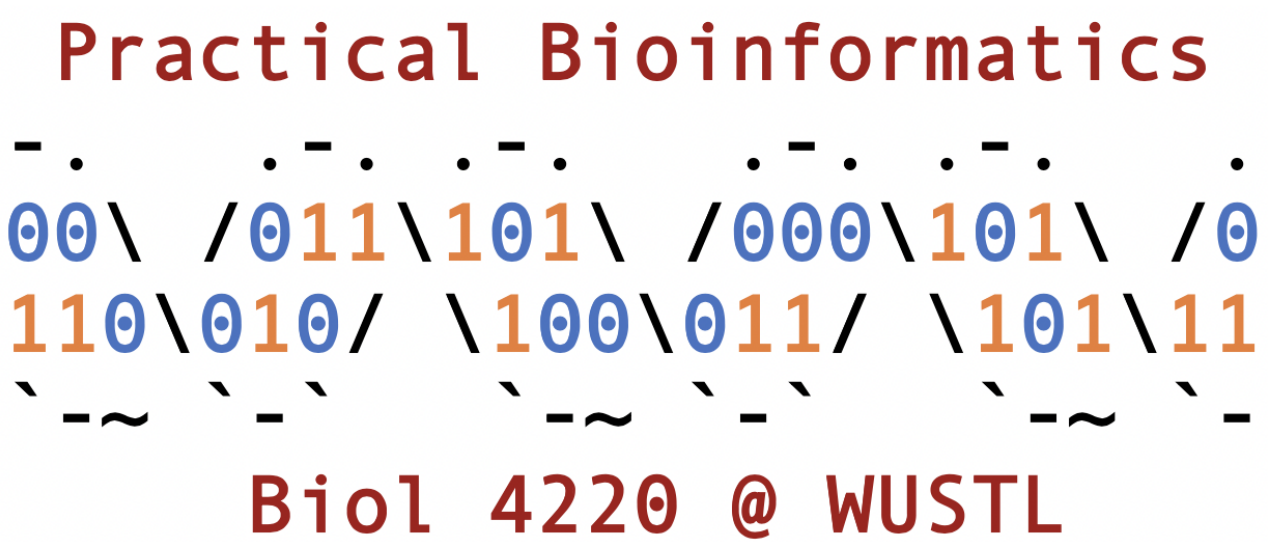
Mon, Sep 14

Lecture 1A:

Intro to practical bioinformatics



Practical Bioinformatics (Biol 4220)
Instructor: Michael Landis
Email: michael.landis@wustl.edu

Practical bioinformatics

Practice foundational computing skills
for everyday biological research

What will you learn? ...how to

- write and debug programs
- build your own analysis pipeline
- test hypotheses with pipelines
- make reproducible research
- collaborate with others

Practical bioinformatics

Everyone in the class differs in terms of background, research interests, goals, etc.

Major goals of the course

- become comfortable with computers
- translate research ideas into code
- independent problem solving
- communicate using technical language
- stay healthy

Schedule

Class

- 830am - 1200pm, Mon + Wed
- online by Zoom
- asynchronous
- synchronous activities + help available

Office hours

- 200pm - 400pm, Thu
- online by Zoom
- *or in Rebstock 210 w/ 1-day notice*

Instructor

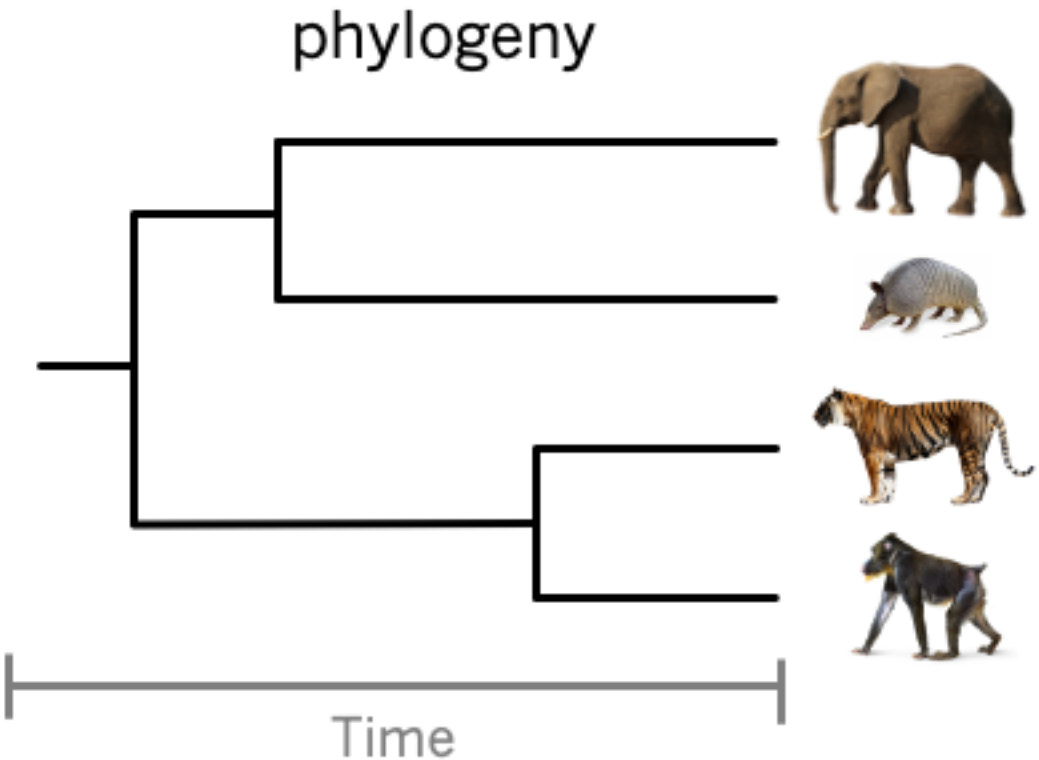
Michael Landis

- Assistant professor
- Biology @ WUSTL
- michael.landis@wustl.edu

Research: landislab.org

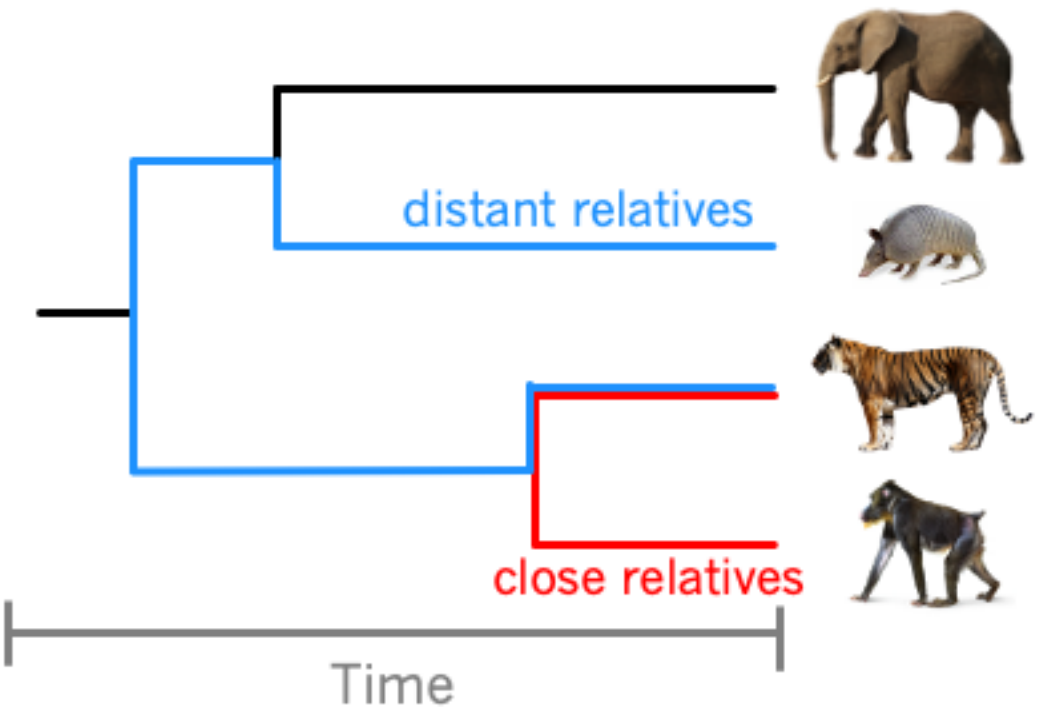
- phylogenetic models and methods
- evolutionary biology
- statistical methods

Statistical phylogenetics



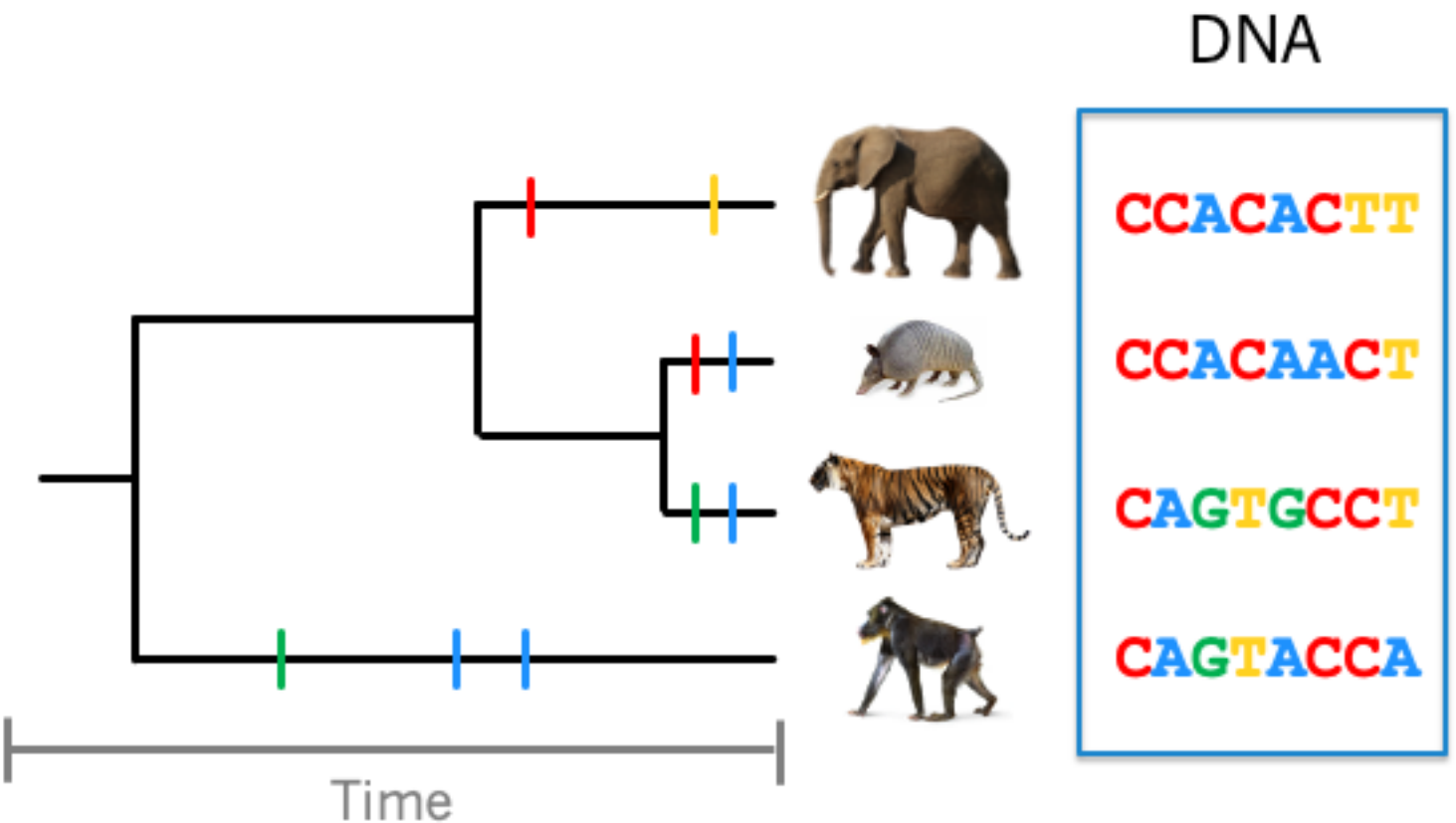
How do we assign and compute probabilities for alternative evolutionary histories?

Statistical phylogenetics



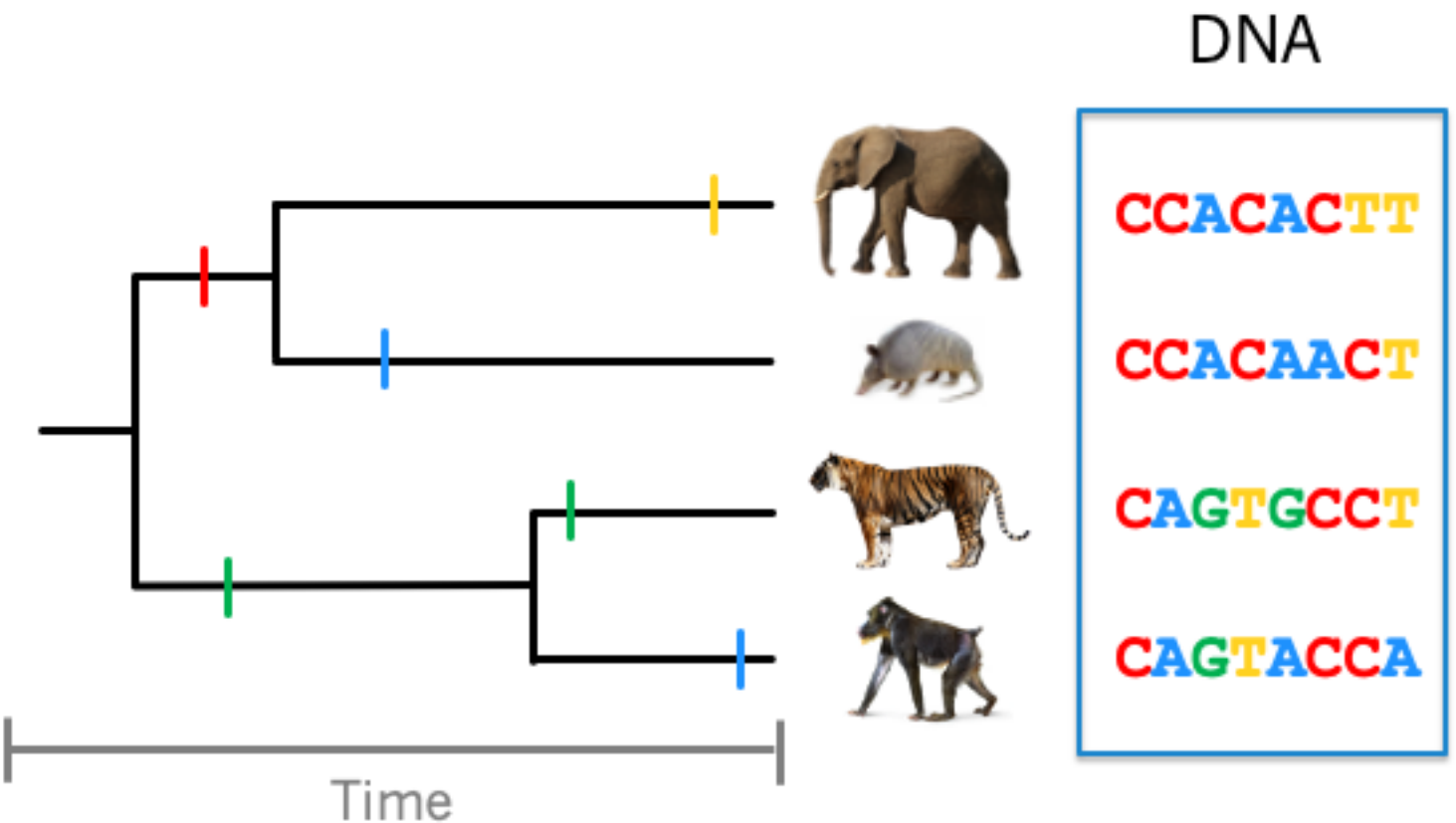
How do we assign and compute probabilities for alternative evolutionary histories?

DNA supports which phylogeny?



9+ mutations needed
(less likely)

DNA supports which phylogeny?



6+ mutations needed
(more likely)

Lecture 1A outline

1. Why bioinformatics?
2. Biol 4220 overview
3. Biol 4220 logistics
4. Anatomy of computing systems
5. Lab 1A overview

Why bioinformatics?

Most major questions in biology require a computational lens

...to name a few

- sequencing the $1K human genome
- assessing global biodiversity health
- mapping connectome of human brain
- tracking SARS-CoV-19
- identifying genetic diseases
- reconstructing tree of life

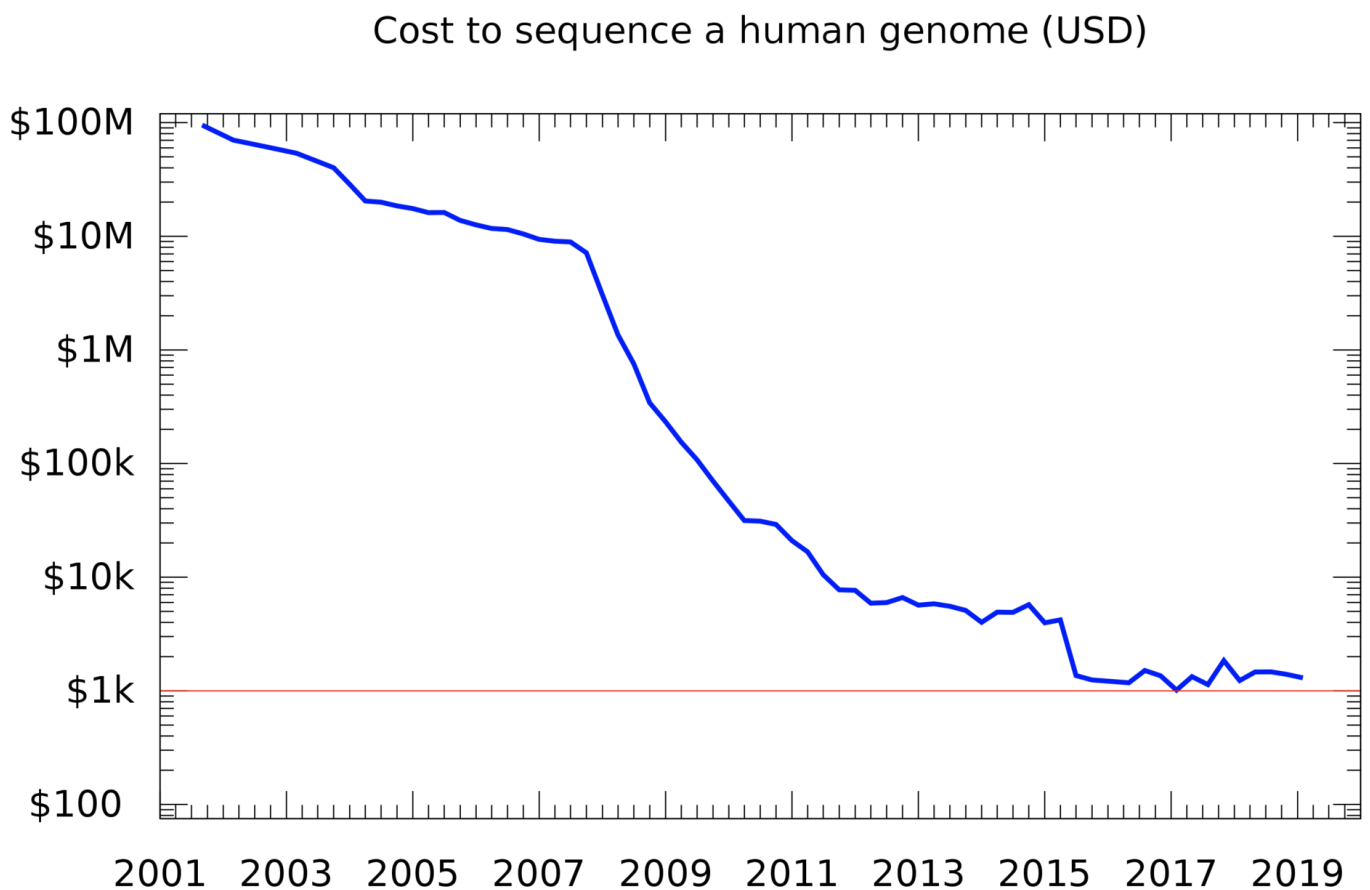
Why bioinformatics?

Different biological problems face similar computational challenges

Every year, it seems that

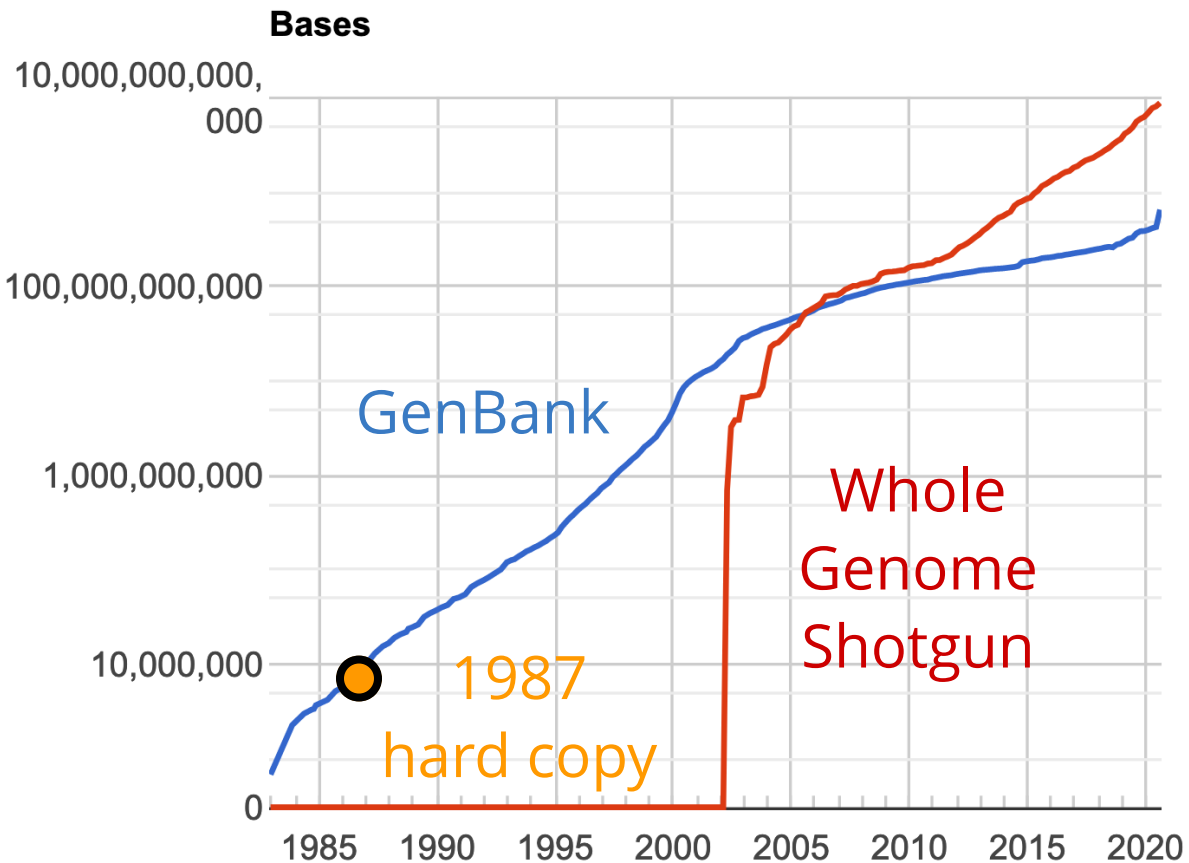
- data contain more samples
- data have higher dimension
- methods are more demanding
- methods are more scalable
- methods are more interconnected
- greater need for reproducibility

More data samples

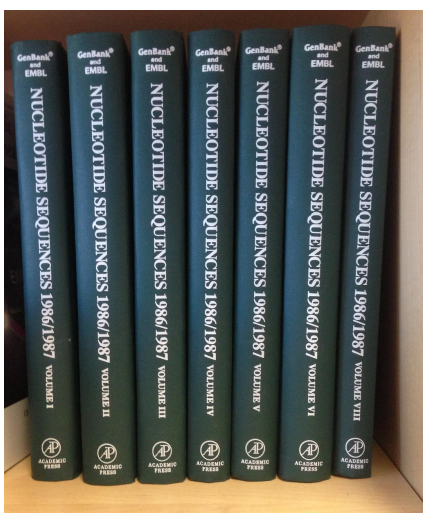


More data samples

Exponential growth in NCBI data



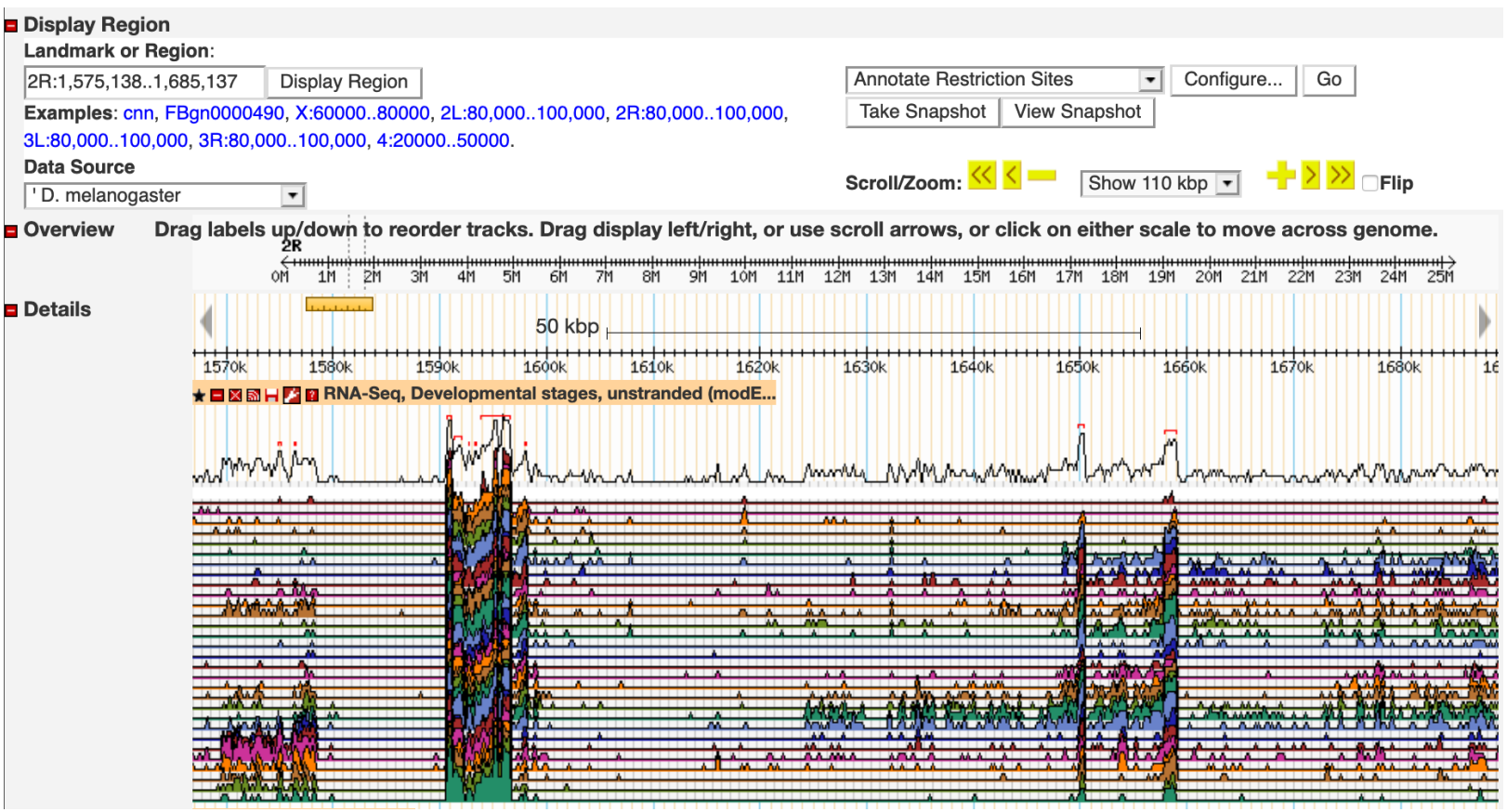
https://www.ncbi.nlm.nih.gov/genbank/statistics/



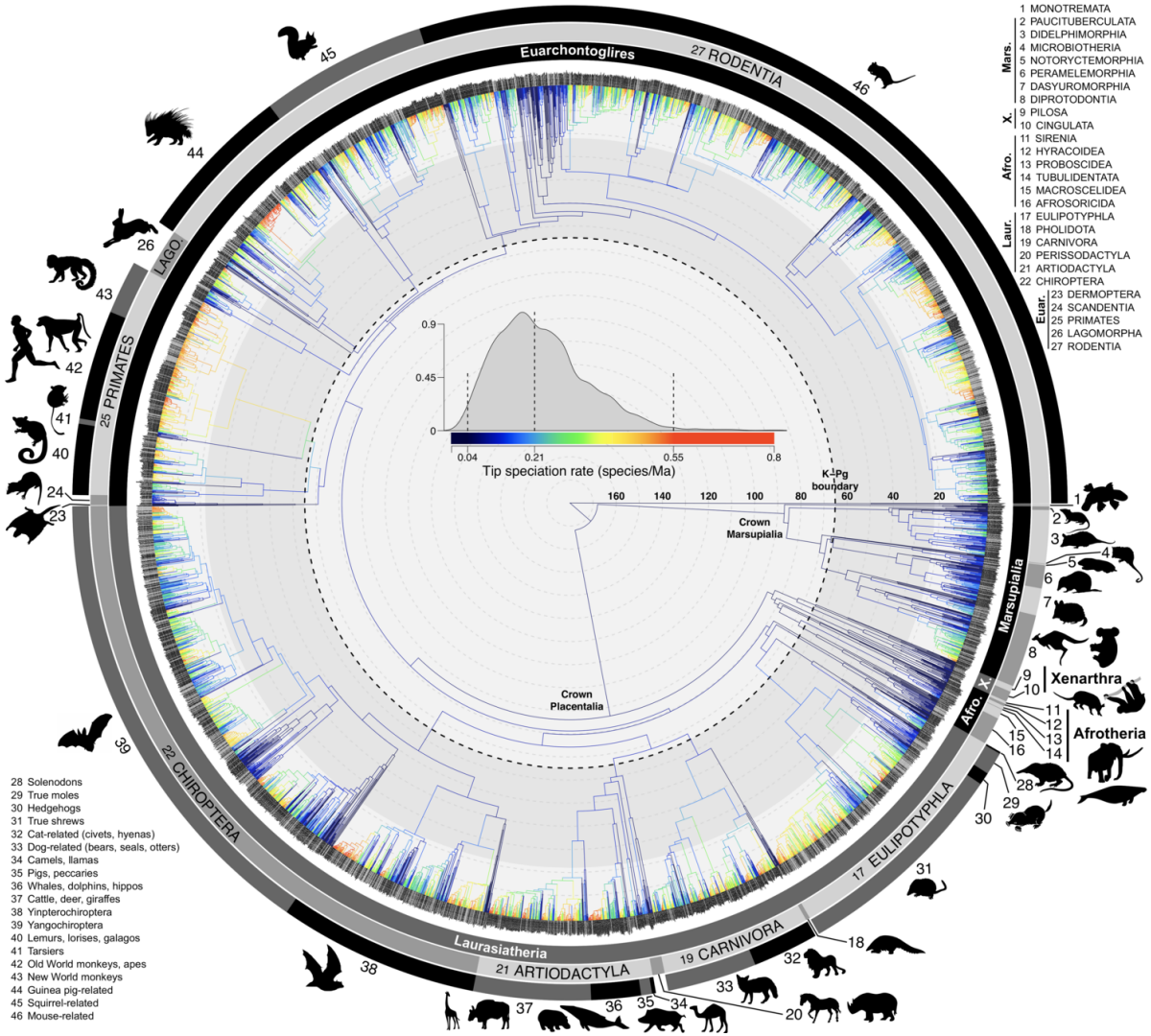
David Landsman @ NCBI

More data dimensions

Gene expression x locus x developmental stage
for *Drosophila melanogaster*



FlyBase GBrowse2



Upham et al. (2019, PLoS Biology)

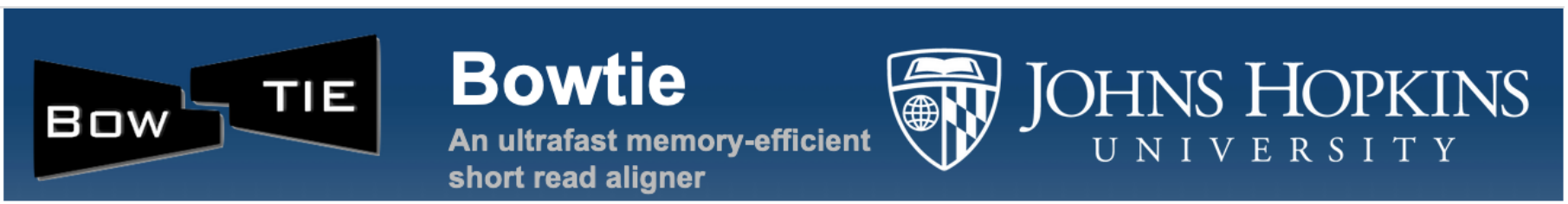
More demanding methods

120+ computer years for phylogeny with 6000+ mammal species

More scalable methods

25 million 35-bp reads per hour

3.2 Gbp in human genome



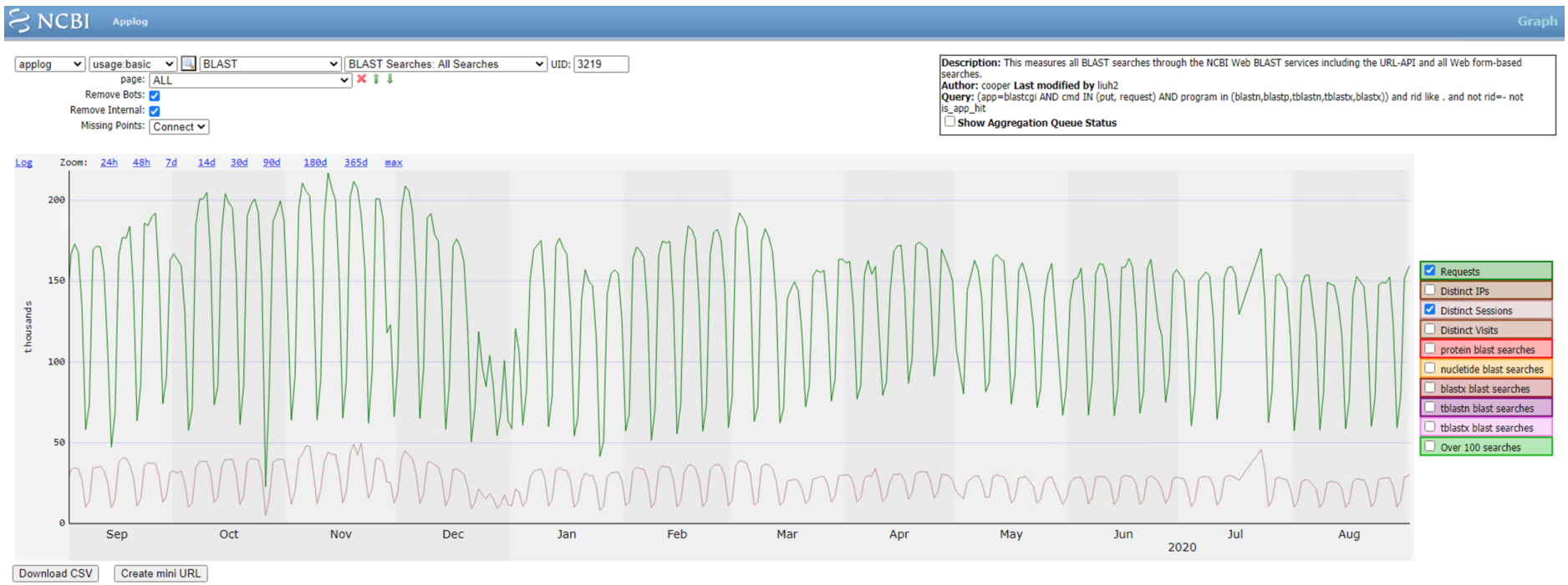
Bowtie is an ultrafast, memory-efficient short read aligner. It aligns short DNA sequences (reads) to the human genome at a rate of over 25 million 35-bp reads per hour. Bowtie indexes the genome with a Burrows-Wheeler index to keep its memory footprint small: typically about 2.2 GB for the human genome (2.9 GB for paired-end).

OSI certified

Bowtie: short-read alignment software

More interconnected systems

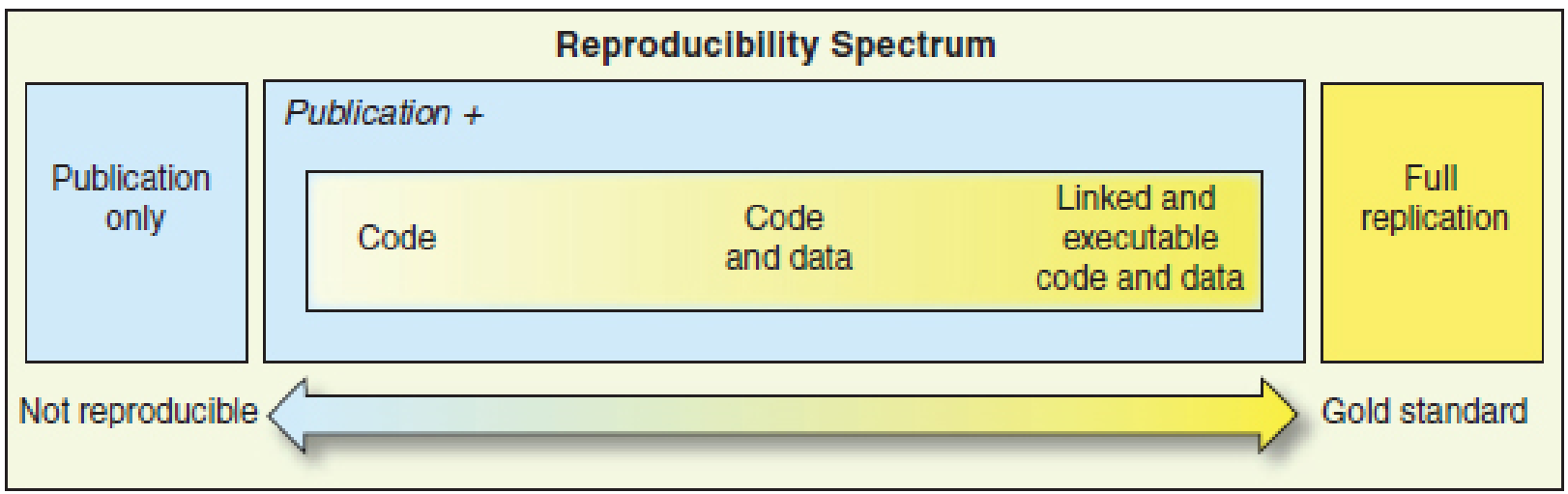
NCBI: 150k+ BLAST requests each weekday



provided by john.sullivan@nih.gov

Greater need for reproducibility

Computational methods allow exact reproduction of published results



from "Reproducible research in computational science" in Peng (Science 2011)

What interests *you* in biological research?

You wrote

- leukemia research
- primate immune system evolution
- disease genetics & epigenetics in fly
- transcription factor binding patterns in yeast
- medical genetics, anatomy & physiology

Why are *you* interested in bioinformatics?

You wrote

- process large datasets
- detect "hidden" patterns in data
- develop computational pipelines
- reduce workload
- improve programming skills
- bridge love of biology & computation

Biol 4220 topics

Computational skills

- UNIX-based operating systems
- Python and shell scripts
- scientific computing libraries
- version control software
- bioinformatics pipeline design

Biological problems

- basics in molecular phylogenetics
- hypothesis testing for Covid-19 evolution

Biol 4220 topics

The **lab exercises** will teach you to

- build a bioinformatics pipeline "from scratch"
- apply pipeline to data to test hypotheses
- identify parts of computer and code
- write, modify, optimize, and debug code
- collaborate using modern tools

The **course project** will allow you combine these ideas, and to explore ideas specific to your interests

Omitted topics

Examples of topics that we **won't** cover in class

- assemble FlyBase transcriptomes with TopHat v2.1.1
- program Java API to cross-reference GO terms against cancer database
- design neural net to infer protein structures from DNA
- derive test statistic for joint distributions of GWAS hits against polygenic traits

...but do bring these topics to office hours

Course page

All course information is centralized here

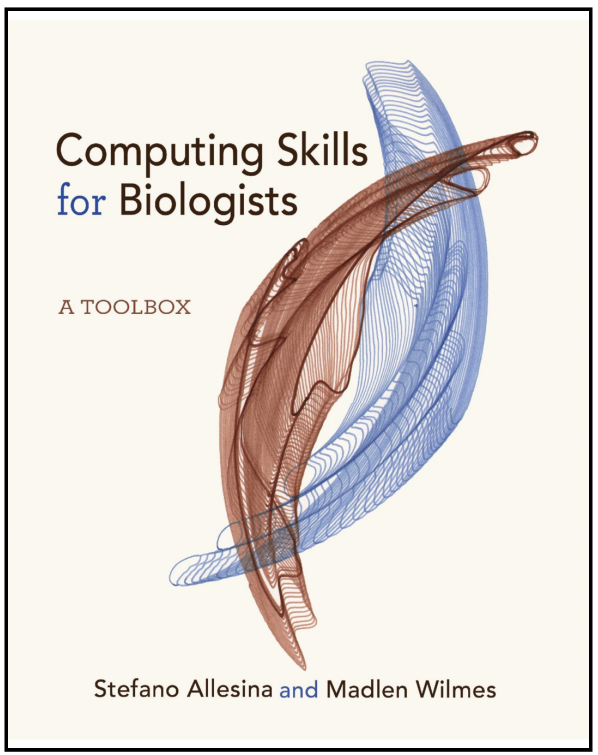
github.com/WUSTL-Biol4220/home

Contains links to

- Syllabus
- Lectures
- Labs
- Canvas
- Zoom
- GitHub Classroom (more later)

Reference texts

Assigned reading for Biol 4220

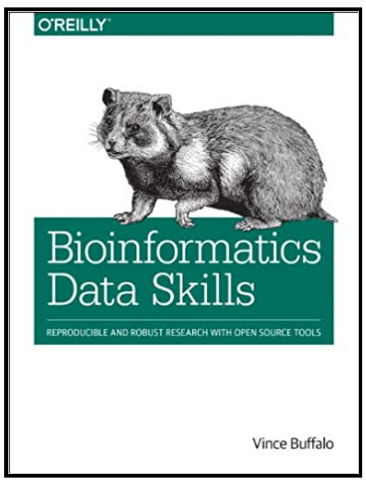


Allesina & Wilmes

ISBN: 9780691167299

~$35

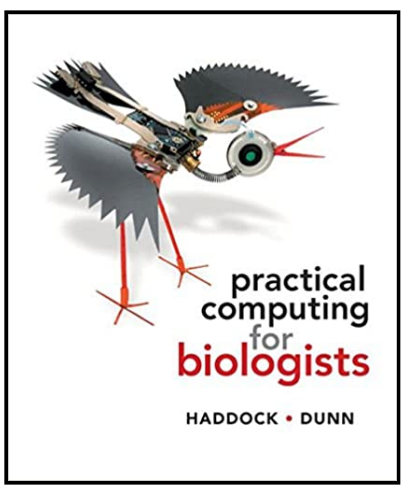
Other excellent resources



Buffalo

ISBN: 1449367372

~$35



Haddock & Dunn

ISBN: 0878933913

~$60

Assignments

Labs	500
Exam 1	150
Exam 2	150
Project	150
Participation	50
Total	**1000**

Grades will be adjusted as warranted

Labs

Each lab will focus on a new skill, and practice its use with previously learned skills

One lab assigned per course meeting,

due within 7 days (168 hours) for full credit

Labs may be completed alone or in groups,
but ***each student must turn in their own work***

Labs will be submitted using an online tool called ***GitHub Classrooms*** (introduced in *Lab 01A*)

Exams

Two open book exams (no final exam)

Exam 1 will focus on material from Weeks 1-7

Exam 2 will focus on material from Weeks 8-14

Exam problems will closely follow questions and exercises from labs and lectures

Exams will be due within 24 hours of assignment

Project

Goal: *build a bioinformatics pipeline to analyze SARS-CoV-19 genomic dataset*

Pipelines will largely be built from lab exercises:

- download and align sequences
- summarize molecular variation
- build a simple molecular phylogeny
- print and plot output
- and 2+ unique features

At the end of the semester, students will

- present their pipeline to the class
- submit their code, output, documentation at end of course

(more details later)

Participation

Students are encouraged to actively participate in the learning experience!

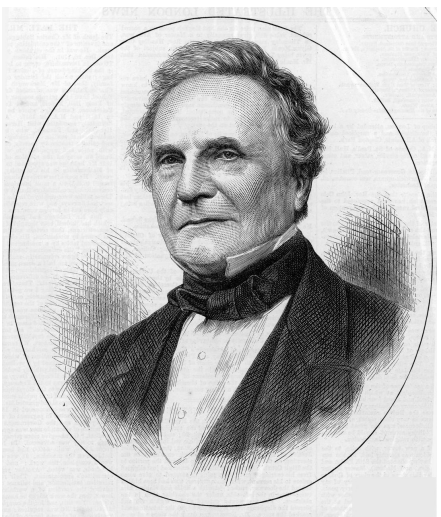
Examples of participation include

- asking and/or answering questions
- working in groups
- helping other students
- visiting office hours
- discussing research problems

Computer systems

Computers are devices that

1. accept input as data
2. process that data
3. report processed data as output



Charles Babbage
"father of the computer"

*“ On two occasions, I have been asked [by members of Parliament], 'Pray, Mr. Babbage, **if you put into the machine wrong figures, will the right answers come out?**' I am not able to rightly apprehend the kind of confusion of ideas that could provoke such a question.*

Anatomy of computing systems

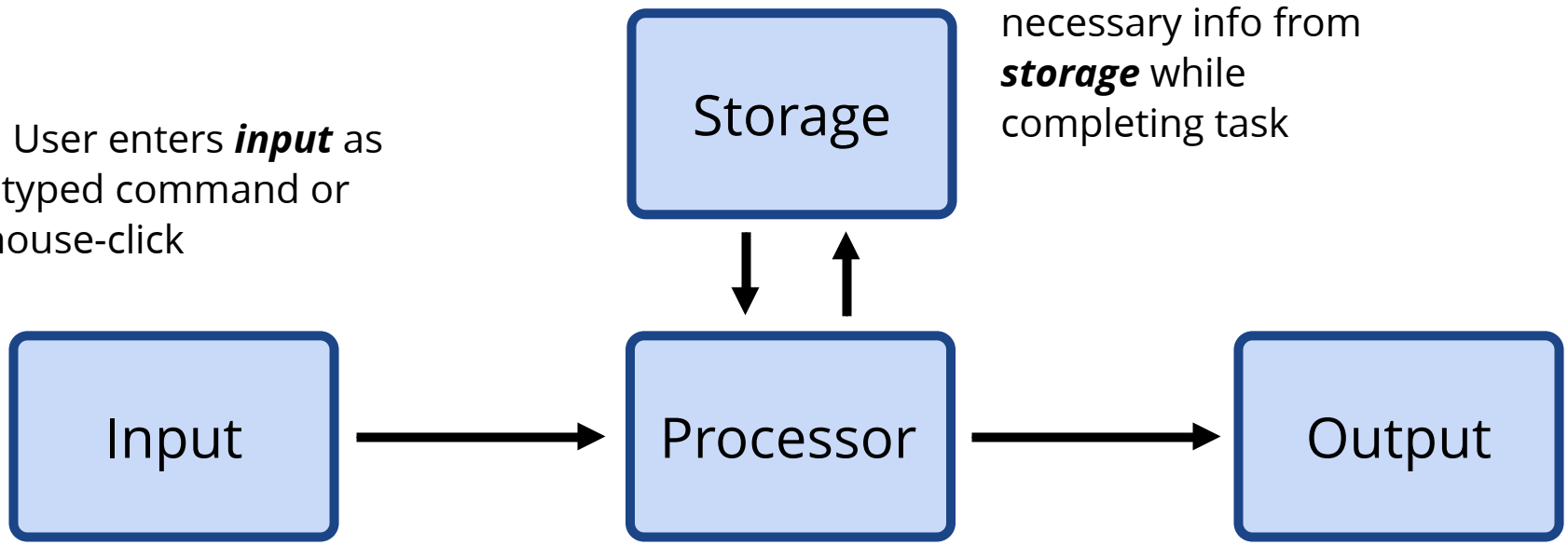
Brief overview of computing systems

- Computing hardware
- Operating systems
- Clusters
- Networks

Most bioinformatics work does not engineer these systems, but knowing how they work (even superficially) is essential for the design and troubleshooting of code.

Computing hardware

(a simple computer)



1. User enters ***input*** as a typed command or mouse-click

3. ***Processor*** reads necessary info from ***storage*** while completing task

2. ***Processor*** actively scans for new ***input*** commands, then ***processes*** command according to programming

4. ***Processor*** also writes to ***storage*** and ***output*** (monitor, printer) as it completes its task

Computing hardware

Component	Examples	Purpose	Cost	Speed (action/sec)	Bottleneck?
Input	mouse keyboard	sends user commands to Processor	$	slow	no
Processor	CPU GPU	process commands, reads to Input/Storage, writes to Output/Storage	$$$$	fast	yes
Storage	hard drive magnetic tape	large, slower, persistent (non-volatile)	$$	medium	depends
Memory	RAM cache	small, faster, temporary (volatile)	$$$	fast	yes
Output	monitor network	sends info to outside world	$	depends	no

Operating system (OS)

Operating systems coordinate user commands with computational resources & hardware

Examples:

- Windows
- Mac OS X *(Unix-based)*
- Linux *(Unix-based)*

Scientific computing mostly relies on ***Unix***-based systems;

We'll be using the Linux distribution, ***Ubuntu***, in this class

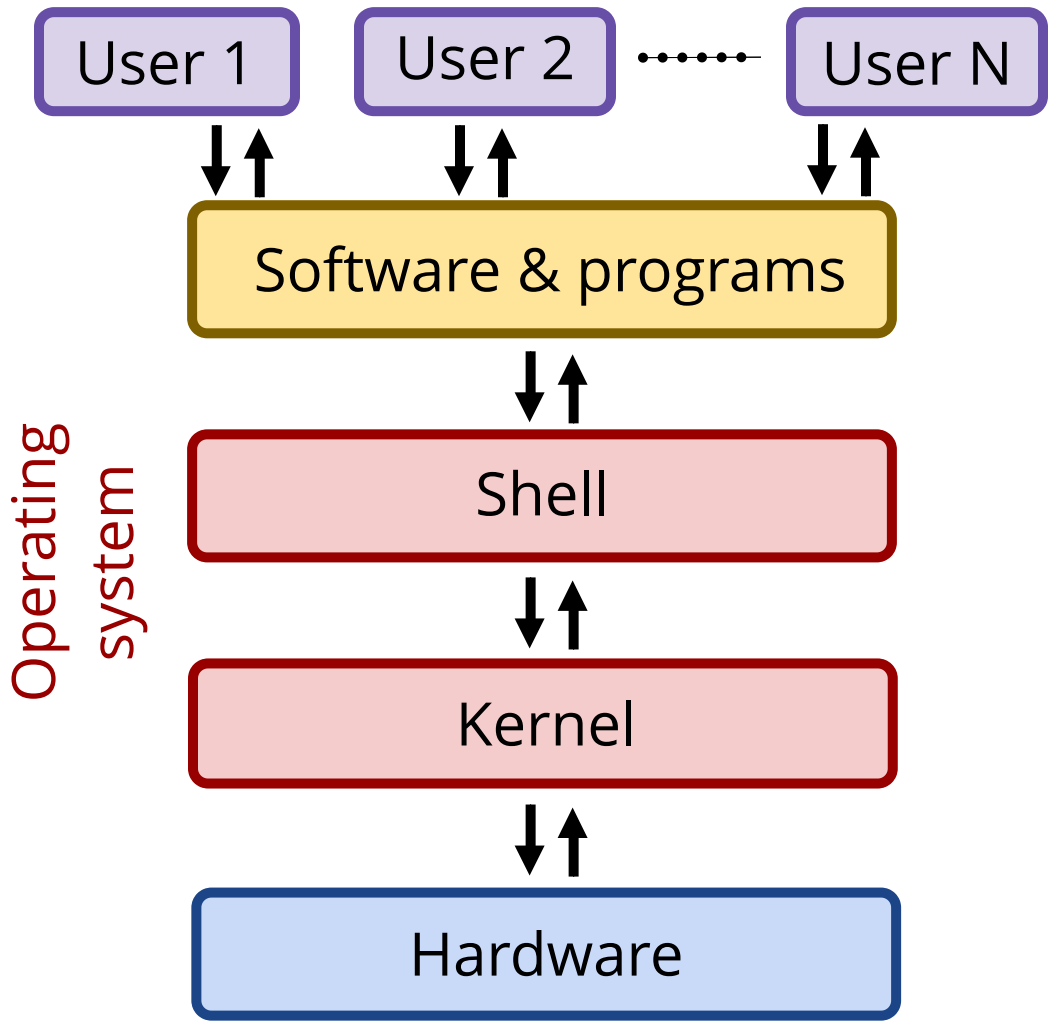
Operating system (OS)

What do operating systems do?

They manage

- ***user interface*** for computer input/output
- ***tasks*** across multiple users and resources
- ***user interruptions*** of scheduled tasks
- ***memory use*** in efficient manner
- ***filesystem*** organization of hard drive
- ***user permissions*** for resource security
- ***network communication*** with other devices
- ***custom software*** to interact with OS + hardware

Operating system (OS)



The Unix-based OS has ***two layers***

1. the ***kernel*** is the core of the OS that controls all tasks and interfaces with hardware
2. the ***shell*** is the interactive interface between the user and the kernel

Kernel vs. shell

The ***kernel*** has control over all computer resources, including processors, memory, storage, devices, task management, etc.

The ***shell*** is a command line interface and scripting language that communicates user commands to kernel for processing. *We as users interact primarily with the shell.*

Example of a Unix shell command:

```
> # connect to my workstation
> ssh mlandis@128.252.89.47
```

HPCs, servers, clusters

Scientific computing requires massive resources when

- datasets are large
- methods are sophisticated and/or inefficient
- rapid results are needed

Research centers build high-performance computing centers (***HPCs***) that provide access to farms of ***servers*** called ***clusters***

HPCs manage processing tasks, backing up data, permissions for file access, file transfer, and much more for larger user bases, often in the 1000s

HPC access

HPC administrators secure HPC resources through

1. user accounts and permissions
2. physical access
3. network access

HPC security is extremely important, because

1. HPC resources are expensive and limited
2. Users store confidential data on HPCs (HIPAA)
3. Bad actors might sabotage or commandeer HPCs, if left unprotected

Computer networks

A ***computer network*** is a group of devices that can send and receive messages to one another through the OS

Each ***network device*** typically has its own core set of hardware (processor, memory). *Examples:*

- *Laptop*
- *Mail server*
- *Wireless router*

The OS of each network device may provide any number of ***network services***. *Examples:*

- *HTTP (websites)*
- *SMTP (mail)*

IP addresses and ports

Each network device is uniquely identified by its ***IP address*** on the network. *(IP stands for Internet Protocol.)*

Each network services is uniquely identified by ***both its IP address*** on the network and its ***port on the device.***

The standard *IPv4* address is four numbers, with each number valued 0 to 255, separated by three periods. Unix ports are typically valued 0 to 65535.

128.252.160.5 : 80

WUSTL web server IP address

HTTP port

Public vs. private IPs

Devices generally "want to see others, but not be seen".

A ***public IP*** can be accessed by any connected device. *Examples:*

- Google's main site
- WUSTL HTTP server

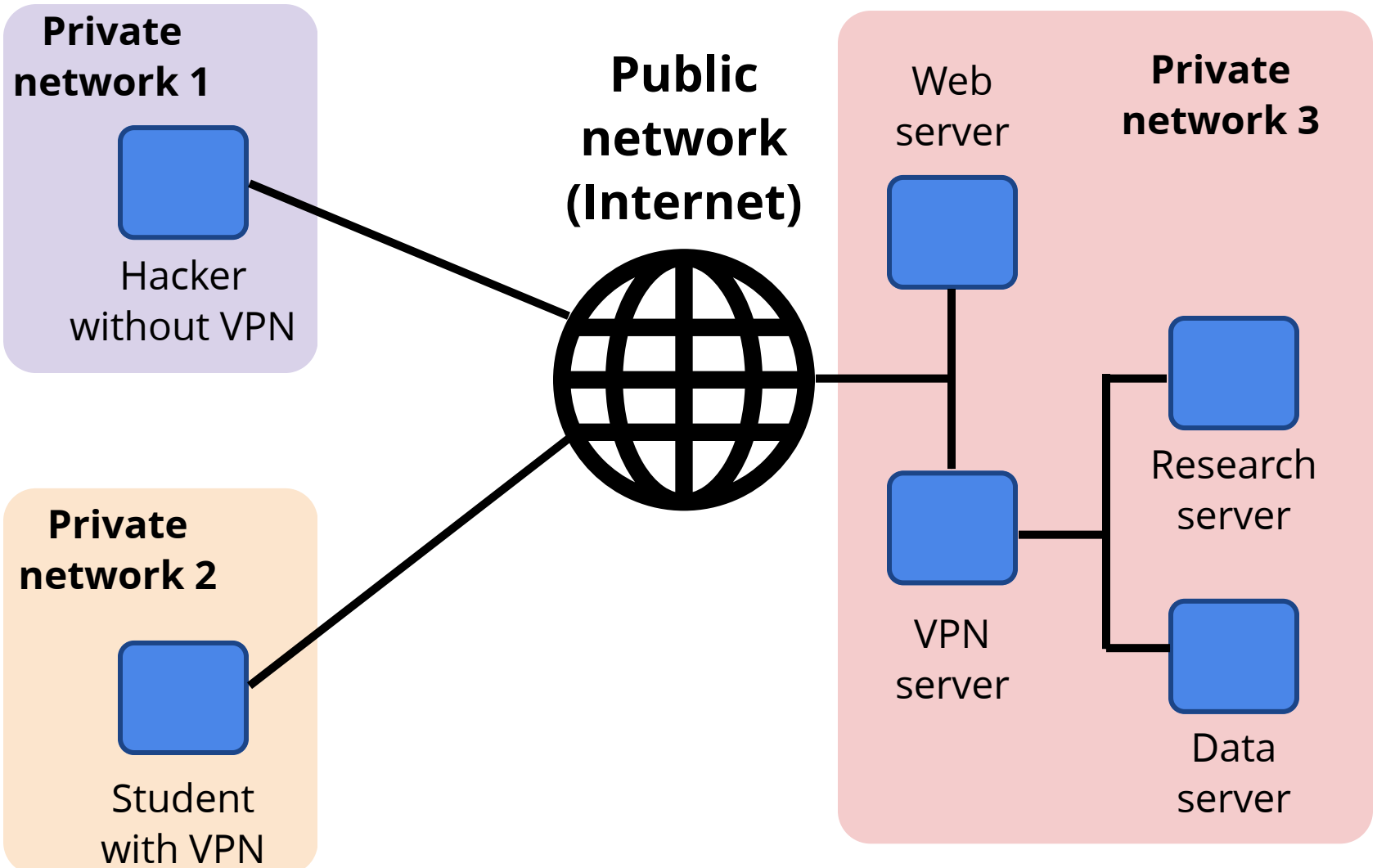
A ***private IP*** can only be accessed by devices that are currently connected (and authenticated through) that network. *Examples:*

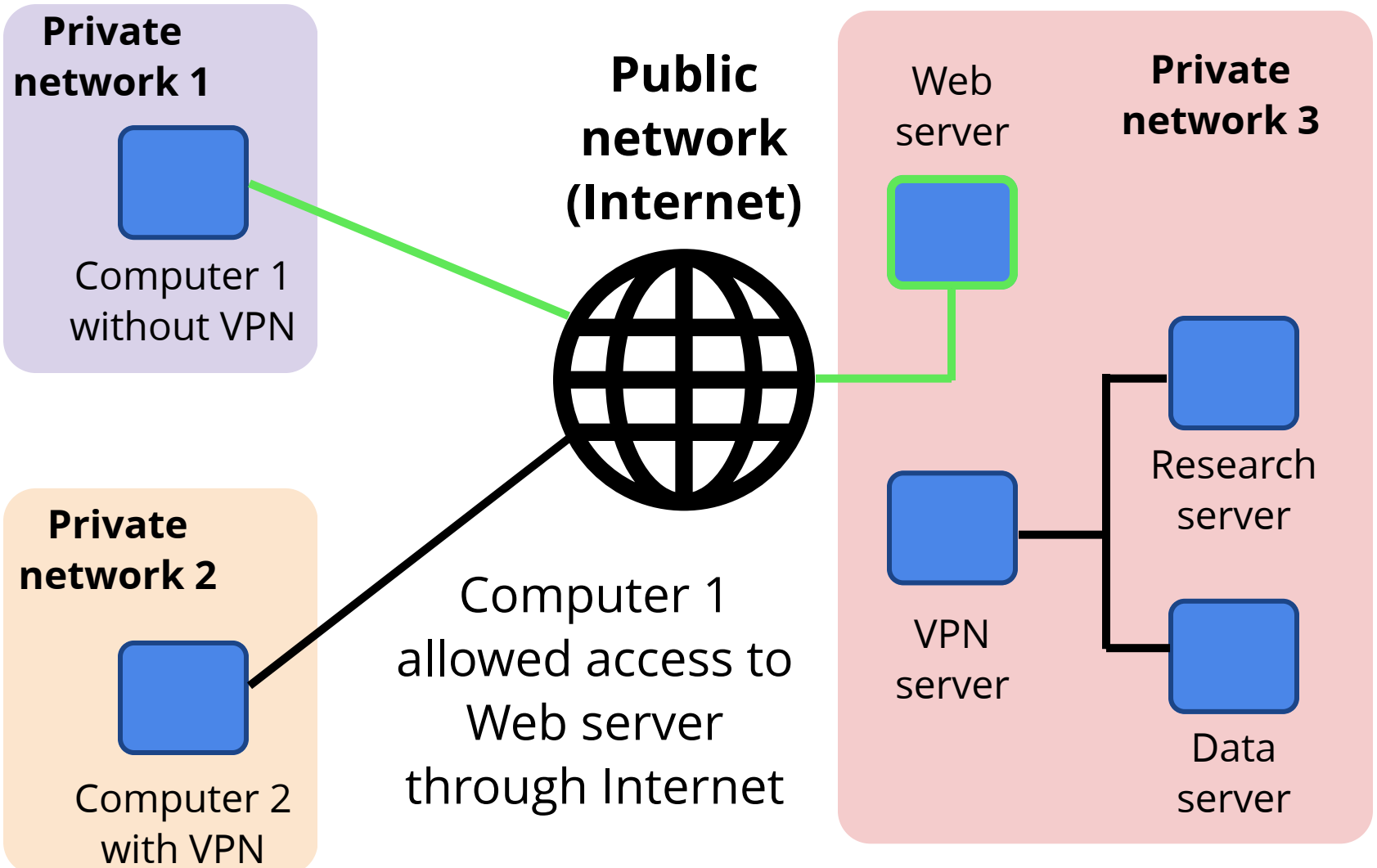
- your computer's IP at home
- WUSTL financial server
- Google's back-end storage farm

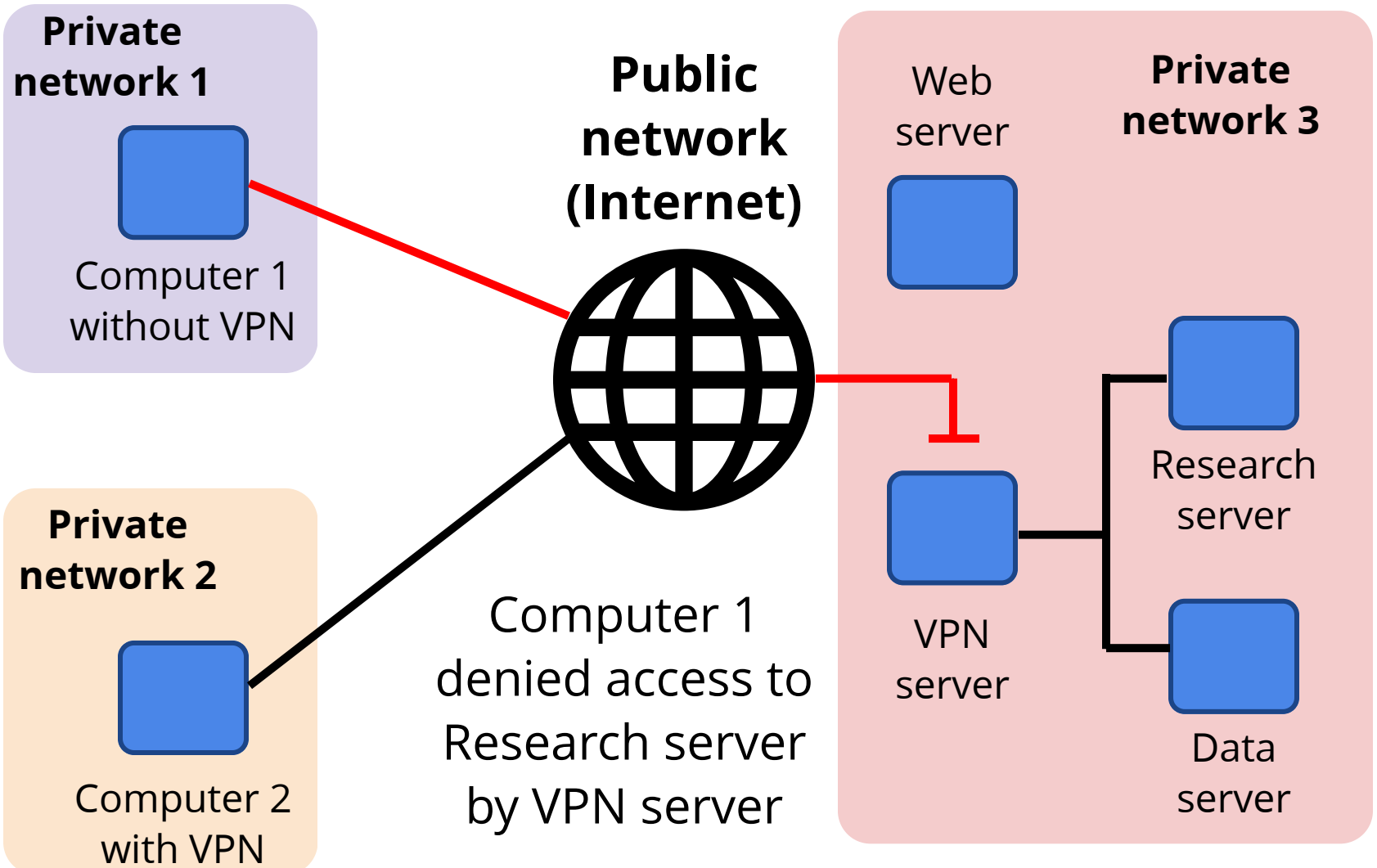
Firewalls and VPNs

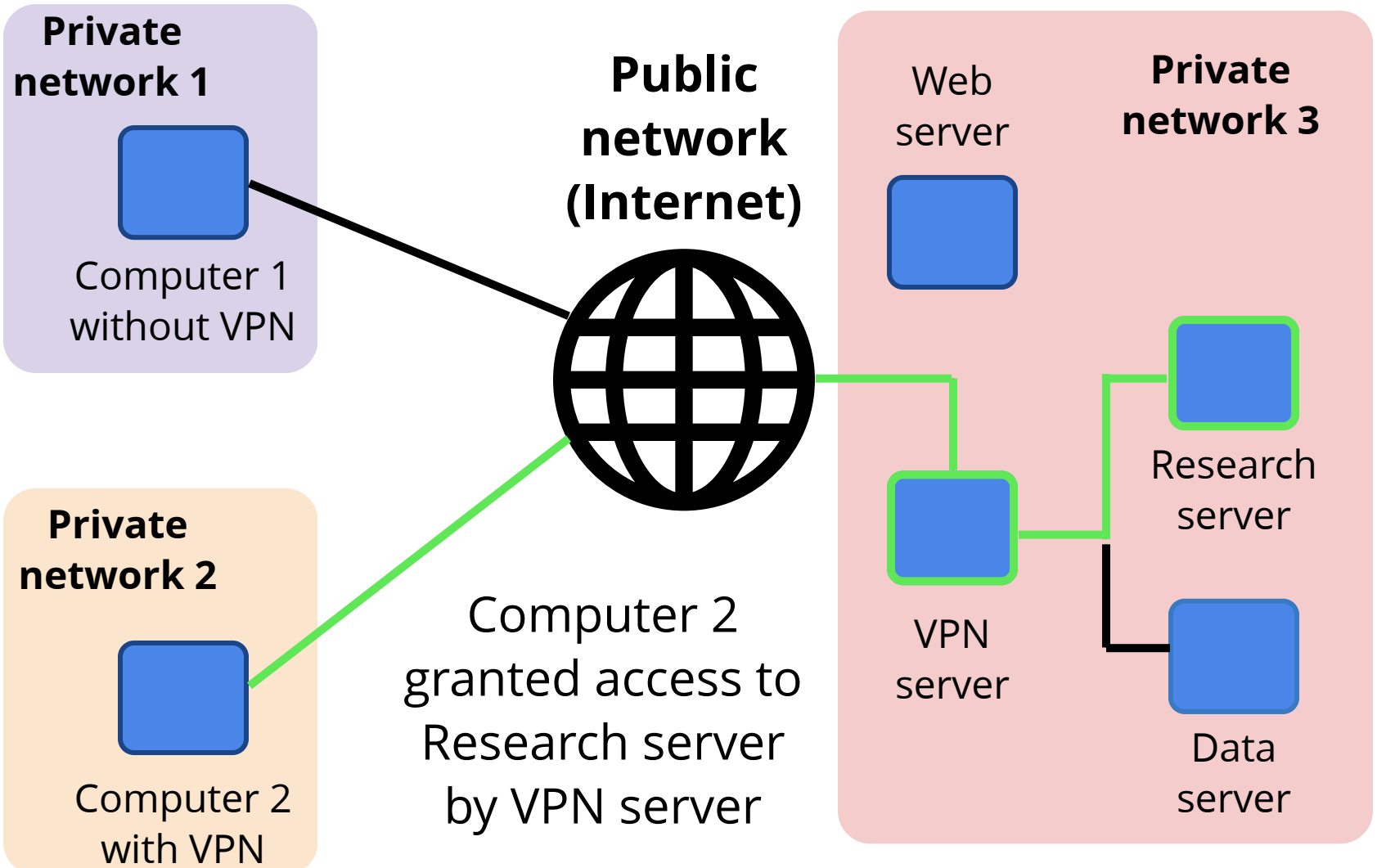
Network administrators have two primary tools for controlling access to resources with public IPs

1. ***Firewalls*** deny public access to all IPs and ports (*"blacklisting"*), but allow access for approved devices and services (called *"whitelisting"*)
2. ***Virtual private networks (VPNs)*** grant devices with special software and credentials to behave as though they also were located in the local private network


Private network 1
Hacker without VPN
Private network 2
Student with VPN
Public network (Internet)
Web server
Private network 3
Research server
VPN server
Data server


Private network 1
Computer 1 without VPN
Private network 2
Computer 2 with VPN
Public network (Internet)
Computer 1 allowed access to Web server through Internet
Web server
Private network 3
VPN server
Research server
Data server


Private network 1
Computer 1 without VPN
Private network 2
Computer 2 with VPN
Public network (Internet)
Computer 1 denied access to Research server by VPN server
Web server
Private network 3
VPN server
Research server
Data server


Private network 1
Computer 1 without VPN
Public network (Internet)
Web server
Private network 3
Research server
VPN server
Data server
Private network 2
Computer 2 with VPN
Computer 2 granted access to Research server by VPN server

Lab 1A

github.com/WUSTL-Biol4220/home/labs/lab_01A.md