ISC-3133
Introduction to Scientific Computing

Peter Beerli
Associate professor
Department of Scientific Computing

Office: 150-T Dirac Science Library
Phone: (850) 645 1324
Email: beerli at fsu dot edu
Web: http://people.scs.fsu.edu/~beerli

Course Goals:

1. Identify the components of scientific computing;
2. Identify standard problems in scientific computing;
3. Implement basic algorithms for standard problems in computational science using the programming language Java.
4. Write, debug, and verify computer codes;
5. Output results of computer simulations on a meaningful manner.

Grading Policy:

The student's grade for the course will be based upon classwork, homework, group projects, a midterm and a final capstone project. This work is weighted as follows:

1. Biweekly Classwork/Homework - 65%
2. Midterm Exam - 15%
3. Capstone Project - 20%

Contents

Computational science [or Scientific Computing]

From Wikipedia, the free encyclopedia

Not to be confused with computer science.

Computational science (or **scientific computing**) is the field of study concerned with constructing mathematical models and quantitative analysis techniques and using computers to analyse and solve scientific problems. In practical use, it is typically the application of computer simulation and other forms of computation to problems in various scientific disciplines.

The field is distinct from computer science (the study of computation, computers and information processing). It is also different from theory and experiment which are the traditional forms of science and engineering. The scientific computing approach is to gain understanding, mainly through the analysis of mathematical models implemented on computers.

Scientists and engineers develop computer programs, application software, that model systems being studied and run these programs with various sets of input parameters. Typically, these models require massive amounts of calculations (usually floating-point) and are often executed on supercomputers or distributed computing platforms.

Numerical analysis is an important underpinning for techniques used in computational science.

Computer Science

Computer science or **computing science** (sometimes abbreviated **CS**) is the study of the theoretical foundations of information and computation, and of practical techniques for their implementation and application in computer systems.[1][2][3][4] It is frequently described as the systematic study of algorithmic processes that create, describe, and transform information.

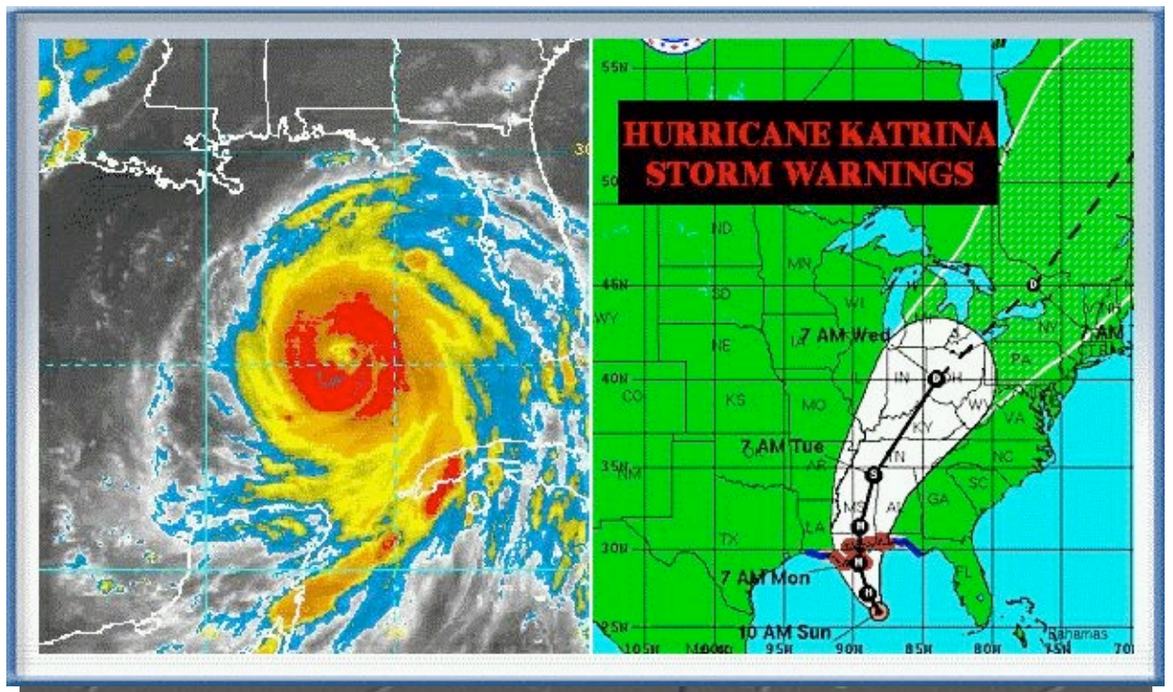
Applications of computational science

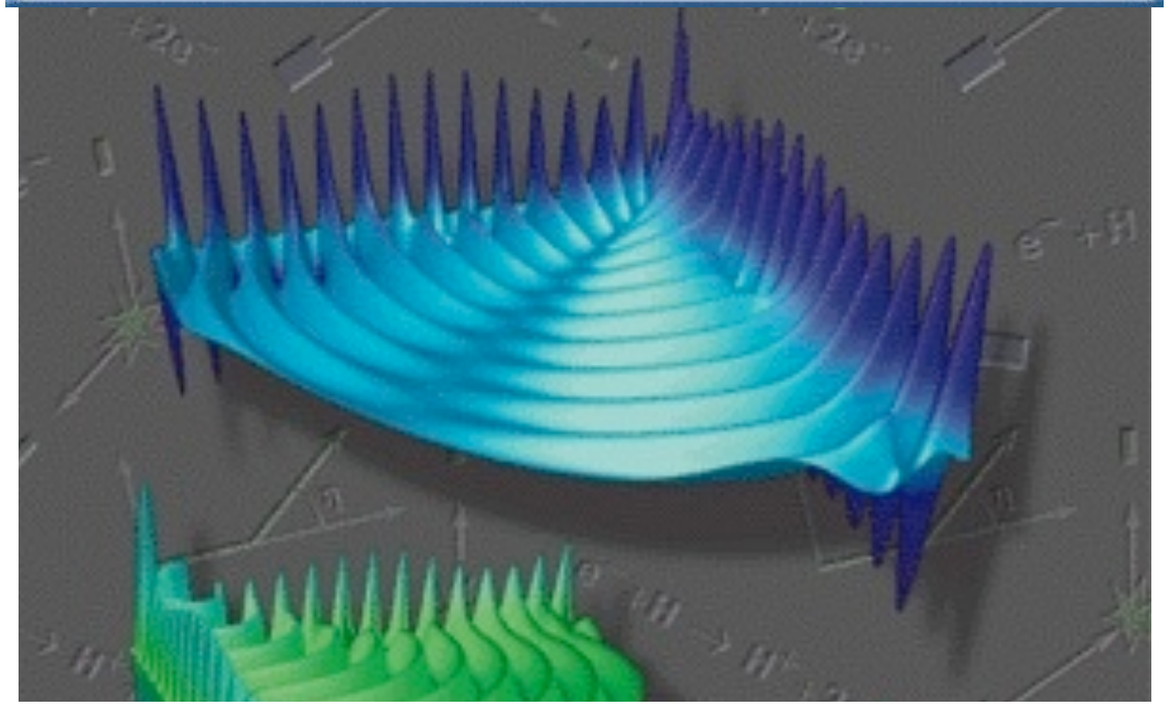
Problem domains for computational science/scientific computing include:

Numerical simulations

Numerical simulations have different objectives depending on the nature of the task being simulated:

- Reconstruct and understand known events (e.g., earthquake, tsunamis and other natural disasters).
- Predict future or unobserved situations (e.g., weather, sub-atomic particle behaviour).





Model fitting and data analysis

- Appropriately tune models or solve equations to reflect observations, subject to model constraints (e.g. oil exploration geophysics, computational linguistics)
- Use graph theory to model networks, especially those connecting individuals, organizations, and websites.

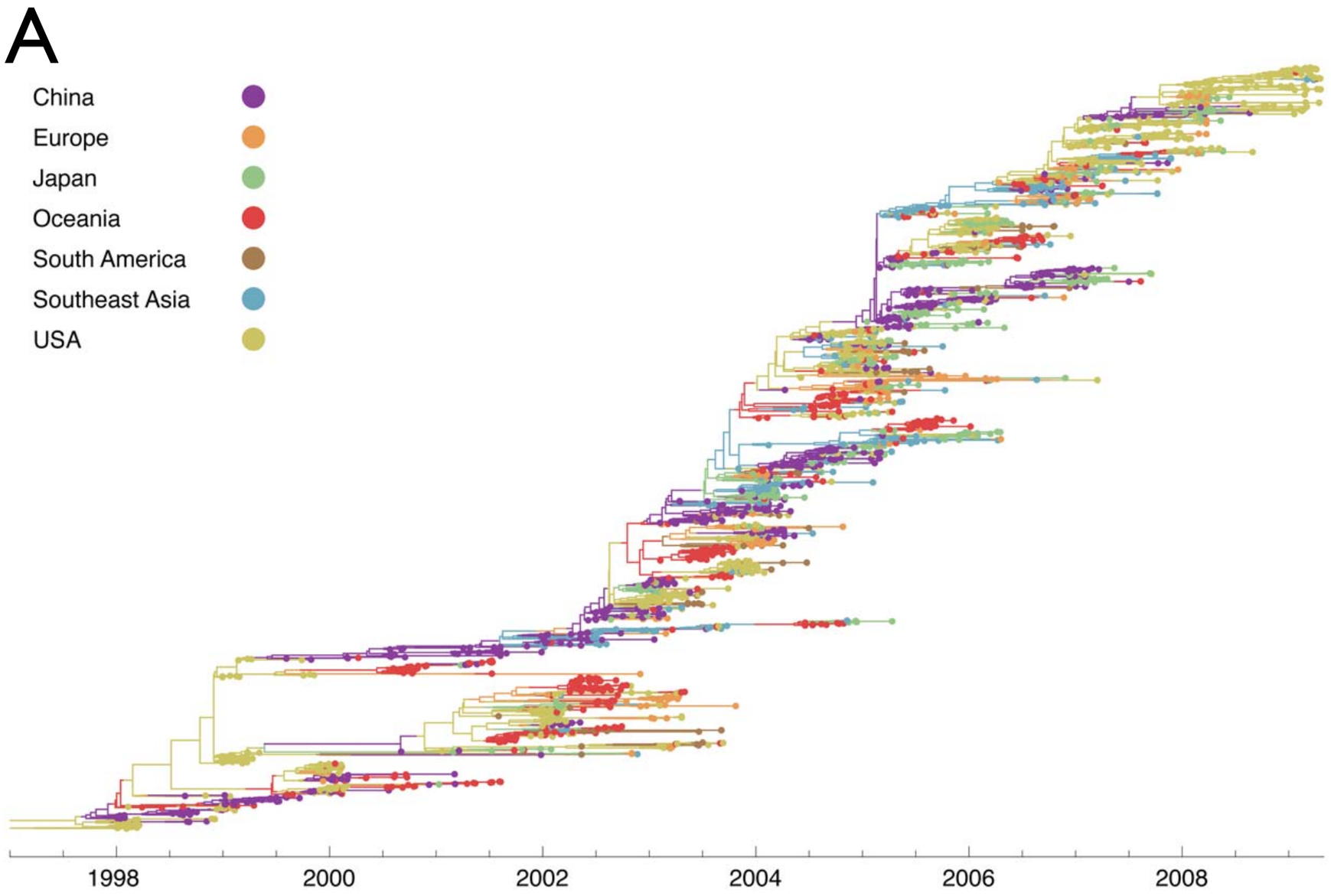
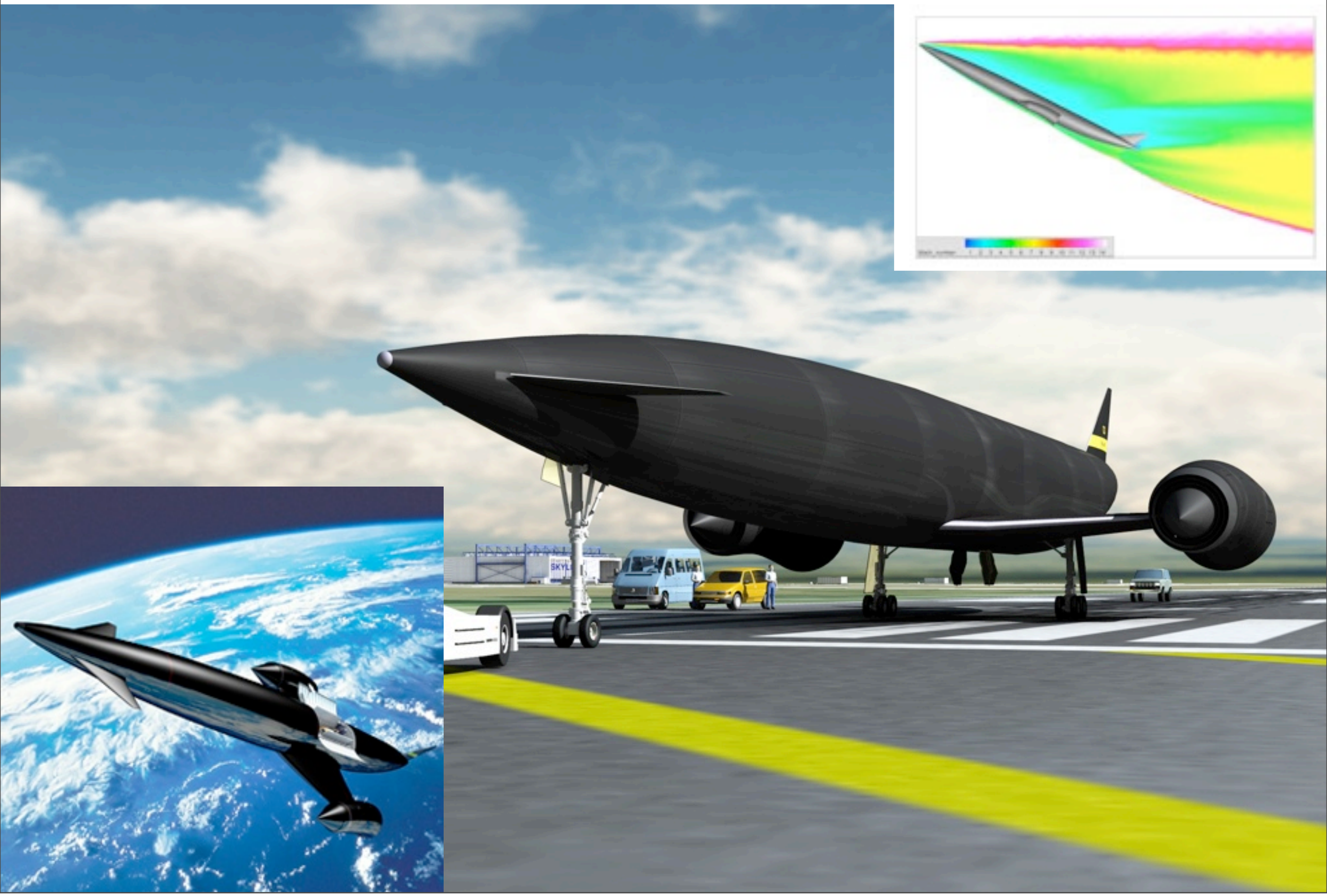


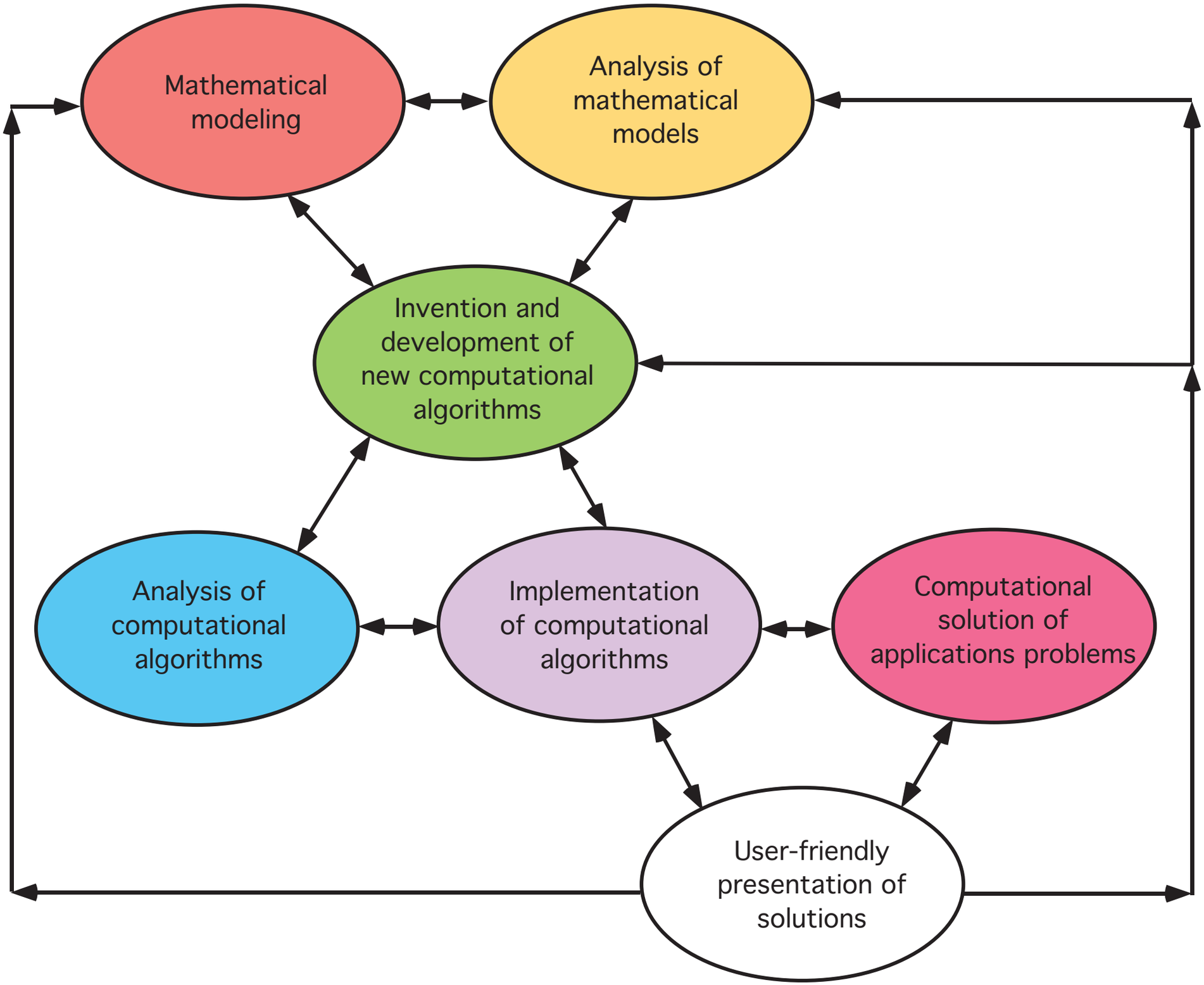
Figure 2. Genealogy of 2165 influenza A (H3N2) viruses sampled from 1998 to 2009. Each point represents a sampled virus sequence, and the color of the point shows the location where it was sampled. Samples are explicitly dated on the x-axis. Tracing a vertical line gives a contemporaneous cross-section of virus isolates. The genealogy is sorted so that lineages that leave more descendants are placed higher on the y-axis than other, less successful lineages. This sorting places the trunk along a rough diagonal, and it places lineages that are more genetically similar to the trunk higher on the y-axis than lineages that are farther away from the trunk. The tree shown is the highest posterior tree generated by the Markov chain Monte Carlo (MCMC) procedure implemented in the software program Migrate v3.0.8 [14,20].
doi:10.1371/journal.ppat.1000918.g002

Computational optimization

Main article: [Mathematical optimization]

- Optimize known scenarios (e.g., technical and manufacturing processes, front-end engineering).





The universe of scientific computing/computational science

in this course, ...

- we are mainly interested in implementing computational algorithms.
- we will use Java to implement these algorithms
- we will learn the basics of Java in the context of basic methods in scientific computing such as

  ★approximate integrals: $\int_a^b f(x)dx$

  ★solving a single nonlinear equation, e.g. find x such that $x = sin\ x$

  ★interpolating or fitting data, e.g. find a line $y = mx + b$

  ★vector and matrix operations $A\vec{x} = \vec{y}$

  ★solving simple differential equation, numerically $\frac{dy}{dt} = e^{-gt}, y(0) = y_0$

- We will visualize some of the results with either gnuplot/python or Java external routines.

JAVA HISTORY (shortened from Wikipedia)

(1991-1994) James Gosling and Patrick Naughton initiated the Java language project in June 1991 for use in one of his many set-top box projects. Java was originally designed for interactive television, but it was too advanced. The language was initially called Oak after an oak tree that stood outside Gosling's office; it went by the name Green later, and was later renamed Java, from a list of random words. Gosling aimed to implement a virtual machine and a language that had a familiar C/C++ style of notation.

(1995) Sun Microsystems released the first public implementation as Java 1.0. It promised "Write Once, Run Anywhere" (WORA), providing no-cost run-times on popular platforms. Major web browsers soon incorporated the ability to run Java applets within web pages, and Java quickly became popular.

(1997) Sun Microsystems approached the ISO/IEC JTC1 standards body and later the Ecma International to formalize Java, but it soon withdrew from the process. Java remains a de facto standard, controlled through the Java Community Process. At one time, Sun made most of its Java implementations available without charge, despite their proprietary software status. Sun generated revenue from Java through the selling of licenses for specialized products such as the Java Enterprise System.

(1998-1999) With the advent of Java 2 (released initially as J2SE 1.2), new versions had multiple configurations built for different types of platforms. For example, J2EE targeted enterprise applications and the greatly stripped-down version J2ME for mobile applications (Mobile Java). J2SE designated the Standard Edition.

(2006) For marketing purposes, Sun renamed new J2 versions as Java EE, Java ME, and Java SE, respectively.

(2006-2007) Sun released much of Java as open source software under the terms of the GNU General Public License (GPL), making all of Java's core code available under free software/open-source distribution terms, aside from a small portion of code to which Sun did not hold the copyright. Sun's vice-president Rich Green has said that Sun's ideal role with regards to Java is as an "evangelist."

(2009-2010) Following Oracle Corporation's acquisition of Sun Microsystems. Oracle has described itself as the "steward of Java technology with a relentless commitment to fostering a community of participation and transparency".

There were five primary goals in the creation of the Java language:

1. It should be "simple, object oriented, and familiar".
2. It should be "robust and secure".
3. It should be "architecture neutral and portable".
4. It should execute with "high performance".
5. It should be "interpreted, threaded, and dynamic".

UNIX shell cheat sheet

The shell allows maintenance tasks, such creating, copying, moving, renaming,... of files and directories/ Among many other things, it also allows to search for files and contents of files.

Focus Directory	`cd`	change directory to the users home directory
	`cd $HOME`	change directory to the users home directory
	`cd ..`	change directory to the directory that is outside of the current one
	`mkdir directory1`	Create directory `directory1`
Manipulating	`mv file1 file2`	Rename file1 to file2, works also with directories
	`cp file1 file2`	Copy file1 to file2
	`cat file1 > file2`	Copy file1 to file2 using pipelining
	`cat file1`	Show file1
	`less file1`	Show file1 with paging, leave this mode using `q`, Top is `g`, Bottom is `G`, paging is <spacebar>
	`nano file1`	Open text file editor (if all key-presses fail try Cntrl-G, or Cntrl-C)
finding	`find . -name file1`	find a file name starting in the current directory and all subdirectories
	`find dir1 -name file1`	find a file name starting in the directory dir1
	`find . -name '*fi*'`	find a file name containing the letters `fi` starting in the current directory
	`find . -name 'fi*'`	find a file name beginning with the letters `fi` starting in the current directory
	`grep "is this" file1`	find all lines in file1 that contain the text "is this"
	`grep "^is this" file1`	find all lines in file1 that begin with "is this"
	`grep "[tT]his" file1`	find all lines in file1 that contain the text "this" or "This"
Changing text	`tr -s '\r\n' '\n' < file1 > file2`: Change all windows end-of-line characters to UNIX end-of-line characters but piping file1 to file2	