NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

Programming Languages

BS3033 Data Science for Biologists

Dr Wilson Goh
School of Biological Sciences

Learning Objectives

By the end of this topic, you should be able to:

- Describe the gamut of programming languages.
- Explain why R is well-suited for data science.
- Explain PERL's attributes for early success during the human genome project and why it lost favour.
- List the reasons for the rise of Python.
- Compare the attributes of Python and R.

NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

Types of Programming Languages

BS3033 Data Science for Biologists

Dr Wilson Goh
School of Biological Sciences

Functional Classification of Programming Languages

Here is a functional classification of programming languages:

System Languages	Architectural Languages	Application Languages	Statistical Languages

The choice of language category depends on what you want to use it for. Application and statistical programming languages are useful for quick and dirty ways of looking into data. But it may not be computationally efficient and may not be suitable for super large datasets.

System Languages

System Languages

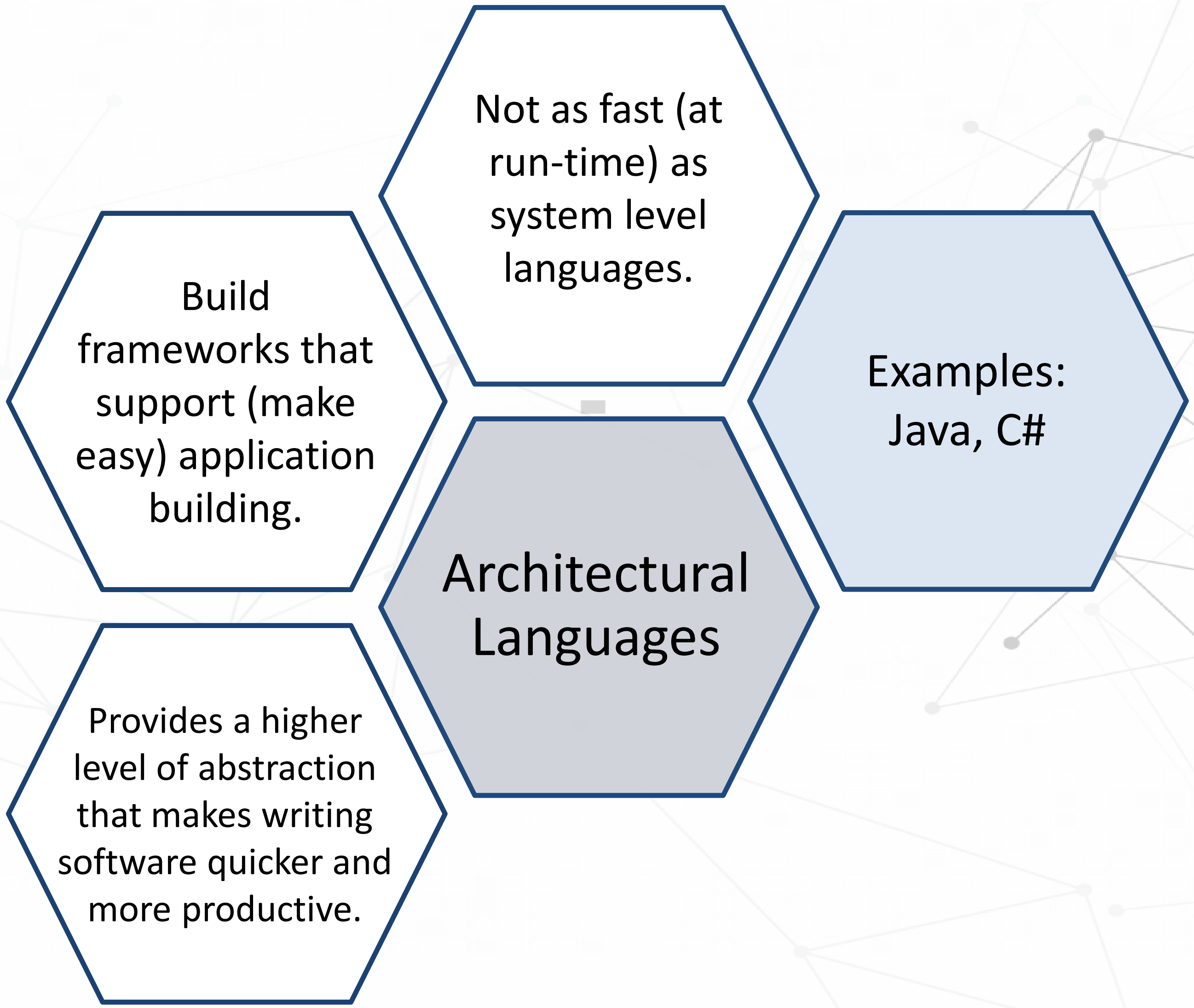
Fast and provides low level (close to the core) access to the computer.

Examples: C, C++

For building operating systems, hardware drivers, etc.

Used when speed is critical.

Architectural Languages


Not as fast (at run-time) as system level languages.

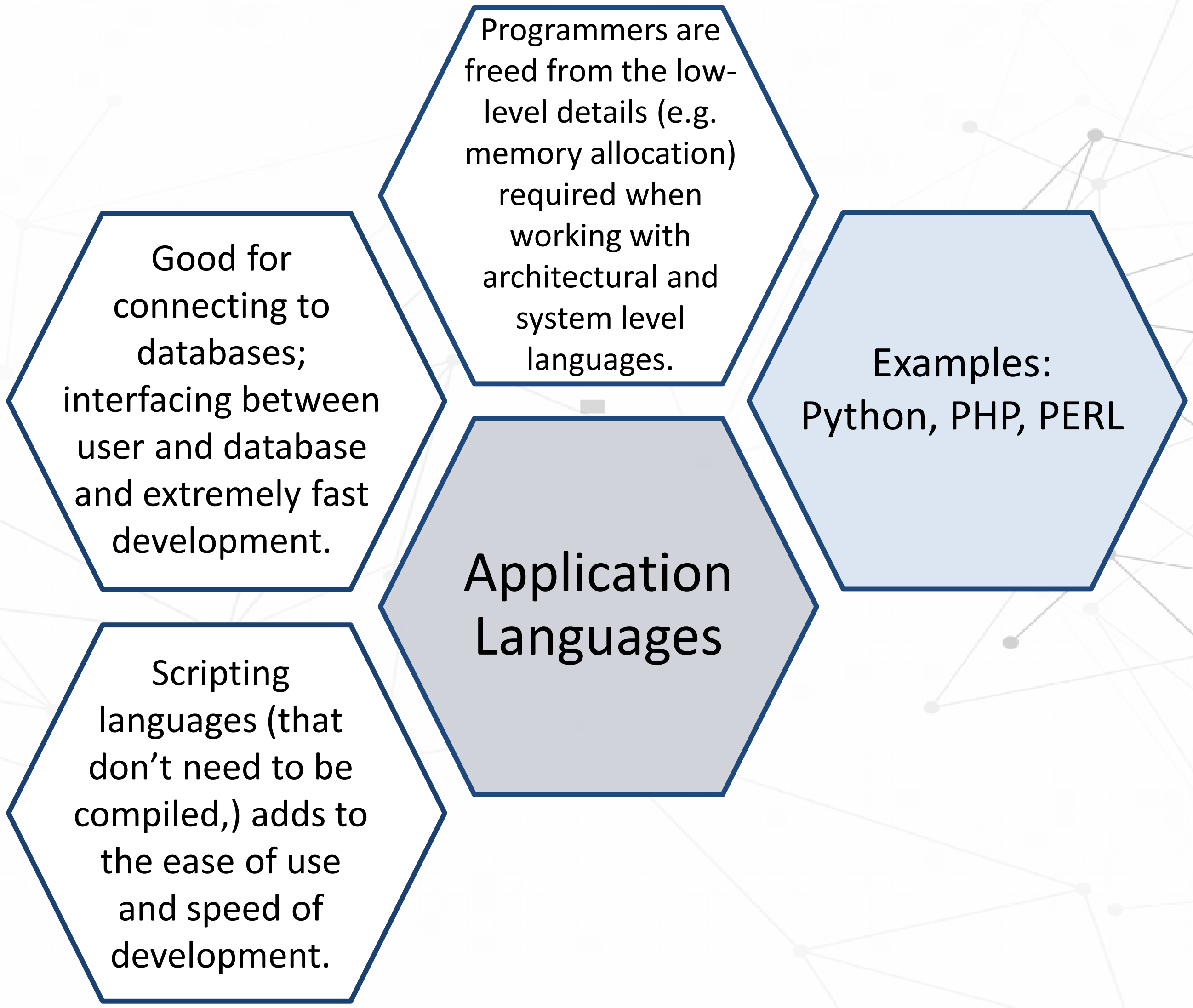
Build frameworks that support (make easy) application building.

Examples: Java, C#

Architectural Languages

Provides a higher level of abstraction that makes writing software quicker and more productive.

Application Languages



Application Languages

Programmers are freed from the low-level details (e.g. memory allocation) required when working with architectural and system level languages.

Examples: Python, PHP, PERL

Good for connecting to databases; interfacing between user and database and extremely fast development.

Scripting languages (that don't need to be compiled,) adds to the ease of use and speed of development.

Statistical Languages

Statistical Languages

Combines ease of computation with graphical output.

Not considered as serious programming languages.

Also considered scripting language as they also do not need to be compiled.

Examples:
R, Mathematica

NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

Programming Styles

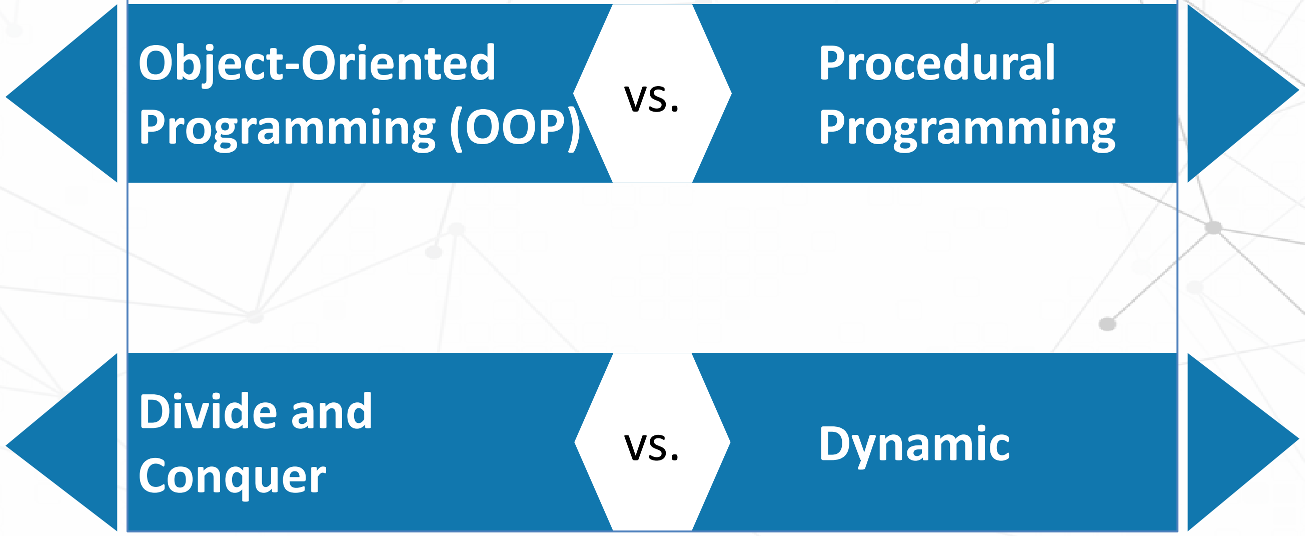
BS3033 Data Science for Biologists

Dr Wilson Goh

School of Biological Sciences

Programming Styles

Computer languages may also be classified by programming mode or style:

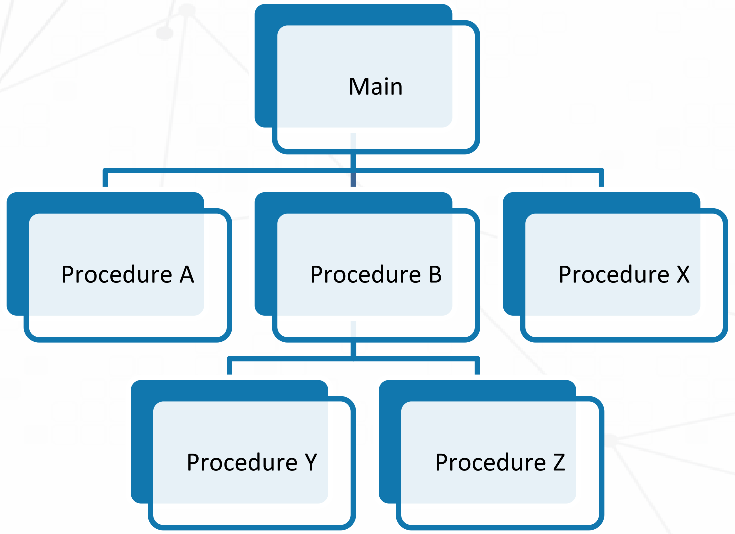


Procedural Programming

Procedural programming is a traditional mode of programming.

It involves a set of sequential procedures that takes input data, processes it, and produces output data (hence the name procedural).

It treats data as a completely passive entity.



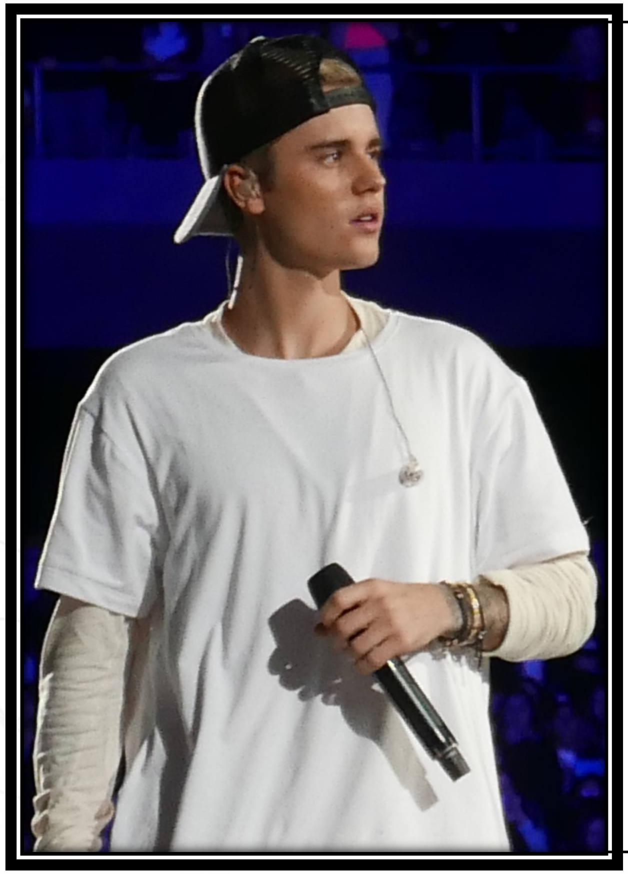
Object-Oriented Programming

OOP is a programming language model organised around objects rather than "actions" on data.

It is a “lifelike” entity, encapsulating both data and function/ methods.

In other words, an object is *active,* not passive; it *does* things. It encapsulates its own data. But it can *share* that data with other objects.

A “Justin Bieber” Object



Source: By Lou Stejskal - Flickr, CC BY 2.0, https://commons.wikimedia.org/w/index.php?curid=48546468

- Justin Bieber could be an object.
- Data:
  - Age
  - Hair Length
  - Location
- Methods:
  - Sing, dance, eat

Objects Inherit from Classes

- Every object belongs to (is an **instance** of) a **class**.
- An object may have **fields**, or **variables**. The class describes these fields.
- An object may have **methods**. The class describes these methods.
- A class is like a template, or cookie cutter. Every object is constructed from a class.

For example, we use the class "boyfriend" to create the object "Justin Bieber".

Class "Boyfriend"

Fields:
- Age
- Hair Length
- Location

Methods:
- Sing, dance, eat

Object "Justin Bieber"

Field data:
- 24
- Medium
- Canada (mostly)

Methods:
- Sing, dance, eat

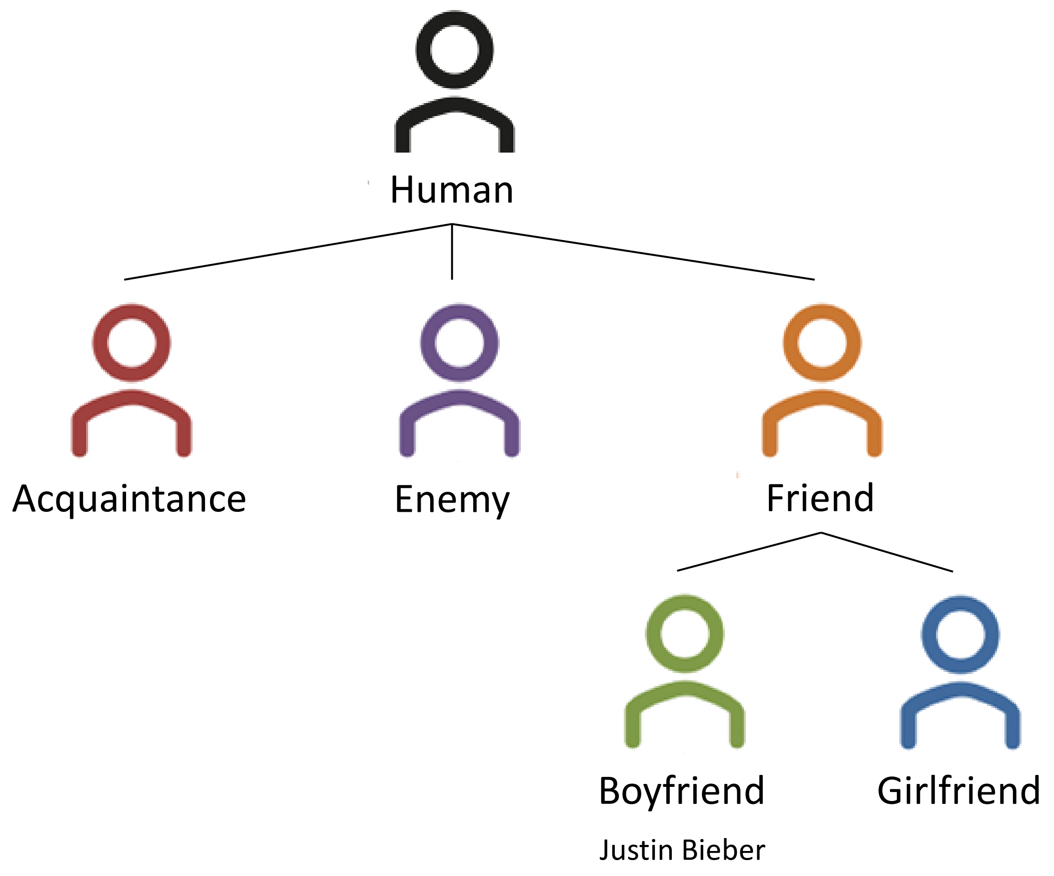
Classes Follow a Hierarchy

Classes are arranged in a treelike structure called a **hierarchy**.

Every class can be preceded by a **superclass** (Child → Parent).

Every class can be succeeded by a **subclass** (Parent → Child).

Classes Follow a Hierarchy



Justin Bieber is a boyfriend. A boyfriend inherits traits from a friend, who inherits traits from a human.

But why so complex?

OOP is meant to be a natural way of reasoning about data in terms of the attributes that describes it and the operations that can be performed on it as real world entities.

OOP with BioPerl

What is BioPerl?

- It is a collection of Perl (a scripting language) modules for processing data for the life sciences.
- A project made up of biologists, bioinformaticians and computer scientists.
- An open source toolkit of building blocks for life and sciences applications.
- Refer to the online documentation to learn about the many powerful things that BioPerl can do: http://www.bioperl.org.

History of OOP

- First work was published in 1996.
- BioPerl 1.0 was released in May 2002.
- Last stable release on 10th July 2014.
- It is a part of the open-bio.org foundation (BioJava, BioPython, BioPerl, EMBOSS, BioMoby).

OOP with BioPerl

The following piece of code prints out → **acgt**:

```
#!/usr/local/bin/perl
use Bio::Seq;
$seq_obj = Bio::Seq -> new('-seq' =>'acgt');
print $seq_obj -> seq(), "\n";
```

- A **Bio::Seq** object, or "**Sequence object**", or "**Seq object**", is found everywhere in BioPerl, it contains a single sequence and associated attributes e.g. names, identifiers, and properties.
- This generic "**Sequence object**" could be either protein or DNA, and it is not linked to a particular format, like the SwissProt, the EMBL or the GenBank ones.
- This line tells PERL to use a module on your machine called **Seq.pm** found in the directory Bio.
- Specifying the directory that contains PERL interpreter.

OOP with BioPerl

This line creates a sequence object (in memory):

```
#!/usr/local/bin/perl
use Bio::Seq;
$seq_obj = Bio::Seq -> new('-seq' =>'acgt');
print $seq_obj -> seq(), "\n";
```

- The PERL variable $seq_obj refers to an instance of the Bio::Seq class.
- new is a subroutine found in the module Bio/Seq.pm. The function call Bio::Seq-> acts as the constructor of the object. '-seq' => 'acgt' assign the value to the attribute.
- The constructor is a call to create an object from a class.

OOP with BioPerl

This line prints out what is returned by the method seq() of the object $seq_obj (which is acgt):

```
#!/usr/local/bin/perl
use Bio::Seq;
$seq_obj = Bio::Seq -> new(‘-seq’ =>‘acgt’);
print $seq_obj -> seq(), “\n”;
```

Method Calling

- The -> notation means that one specifically intends to call the subroutine seq that is attached to $seq_obj. Indeed, a different object might have a method named seq, with a possibly different implementation if it belongs to a different class (polymorphism).

- Polymorphism: Two classes can have methods of the same name, but does different things. For example, the **sing** method for the object “Adele” belonging to the class “Singer” and the **sing** method for the object “Canary” belonging to the class “Bird” are not the same things.

OOP with BioPerl

```
#!/usr/local/bin/perl
use Bio::Seq;
$seq_obj = Bio::Seq -> new(‘-seq’ =>‘acgt’);
print $seq_obj -> seq(), “\n”;
print $seq_obj -> alphabet(), “\n”;
print $seq_obj -> subseq(3,4), “\n”;
```

Prints out

```
acgt
dna
gt
```

The BioPerl documentation tells us that the Bio::Seq object has many other methods, for example seq, alphabet, subseq.

OOP with BioPerl

But all these seem rather trivial methods for dealing with strings.
Does OOP make it simpler to deal with more advanced applications?

The Bio::SeqIO object is responsible for reading/writing sequence to file. It provides support for the various database formats. The below script creates a 'sequence' object and saves it to a file named 'test.seq' under FASTA format. FASTA is a basic file format for representing biological sequence data.

```
#!/usr/local/bin/perl
use Bio::Seq;
use Bio::SeqIO;
$seq_obj = Bio::Seq -> new(‘-seq’ =>‘acgt’,
                           ‘-id’ =>‘#12345’
                           ‘-desc’ =>‘example 1’);
$seqio_obj = Bio::SeqIO->new(‘-file’=>‘>test.seq’,
                             ‘-format’=>‘fasta’);
$seqio_obj->write_seq($seq_obj);
```

Saves the file test.seq containing

```
>#12345 example 1
acgt
```

The Bio:SeqIO object has convenient methods for format conversion (format) and export (write_seq). Such operations are required often, and can be error prone if we write a new process each time.

OOP with BioPerl

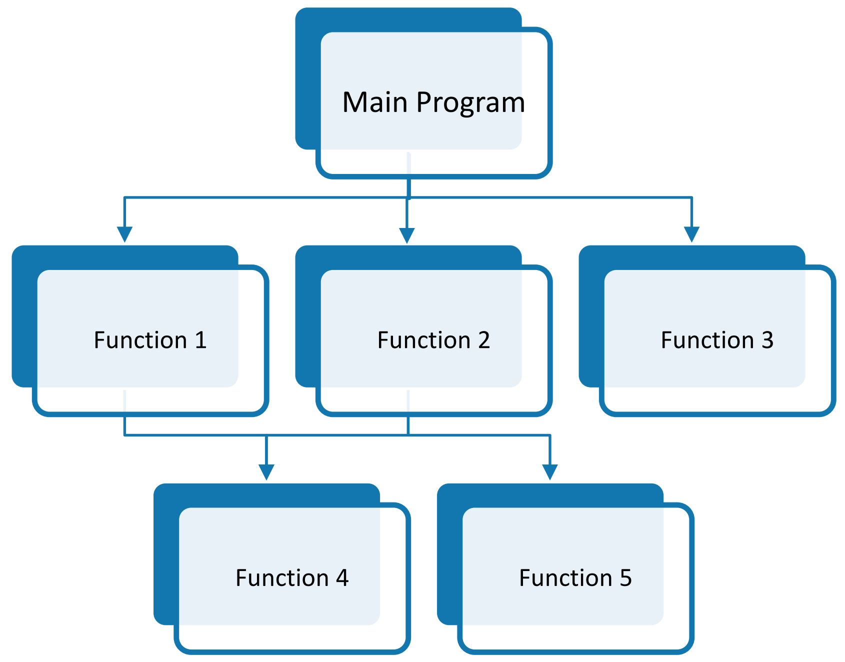
Fasta is a very simple format and easy to write with code. How about a more complex format? If we replace '-format' => 'fasta' with '-format' => 'embl' in the Bio::SeqIO constructor, we get:

```
ID   #12345 standard; DNA; UNK; 4BP.
XX
AC   unknown;
XX
DE   example 1
XX
FH   Key Location/Qualifiers
FH
XX
SQ   Sequence 4 BP; 1 A; 1 C; 1 G; 1 T; 0 other;
     acgt 4
//
```

This is a rather extensive flatfile format. It is not so simple to write programmatically since you need to know all the fields and their order. It is rather easy to make mistakes.

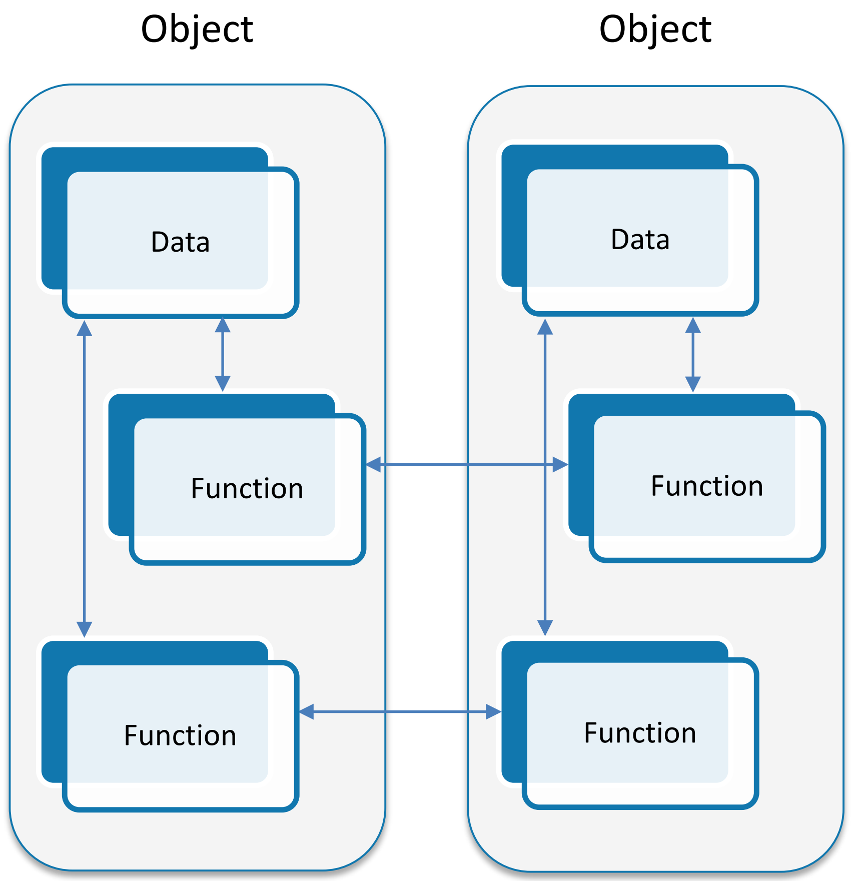
OOP vs. Procedural

Procedural Programming



V S.

OOP



OOP vs. Procedural

Procedural Programming

- Top-down Design
- Limited Code Reuse
- Complex Code
- Global Data Focused

V S.

OOP

- Object Focused Design
- Code Reuse
- Complex Design
- Protected Data

Divide and Conquer Programming

The **divide and conquer** approach takes a complex problem and breaks it down into smaller non-overlapping components. It usually comprises 3 steps:

1

Divide: Break the given problem into sub-problems (top-down).

2

Conquer: Recursively solve these sub-problems.

3

Combine: Combine the answers.

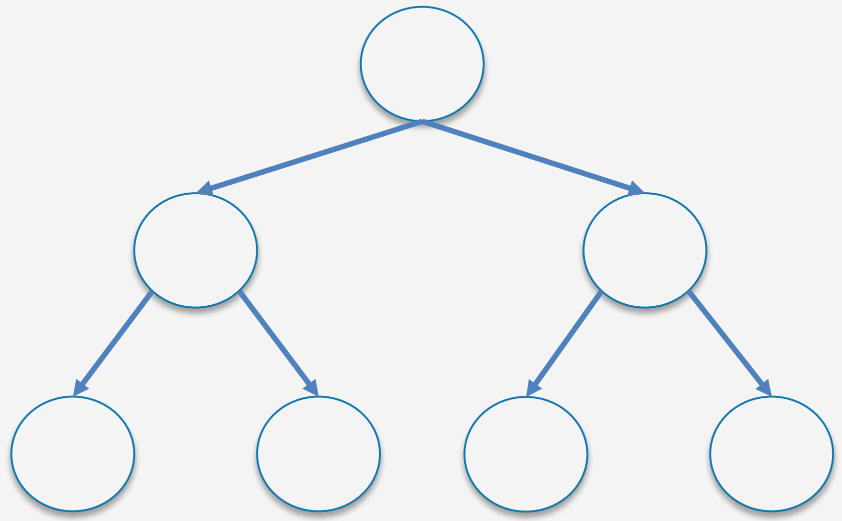
Dynamic Programming

Dynamic programming is an approach for solving a complex problem by breaking it down into a collection of simpler but possibly overlapping subproblems. It involves the following steps:

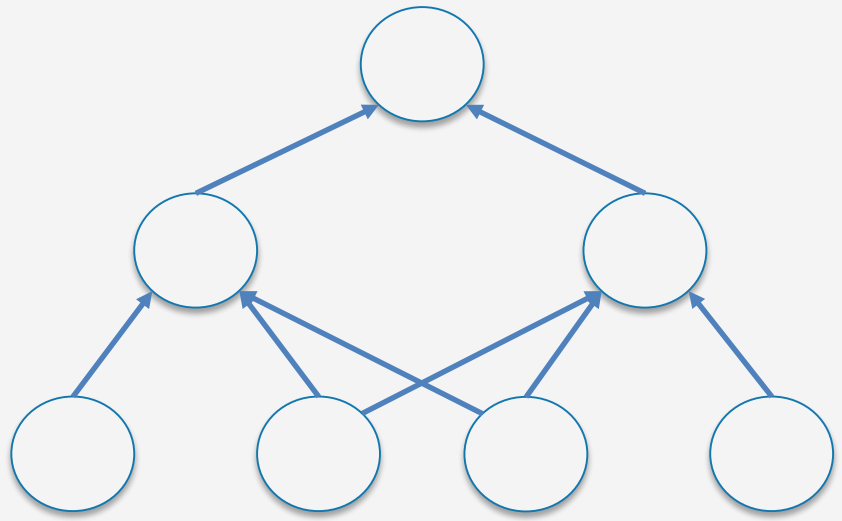
1. Analyse the problem and see how the sub-problems can be solved upwards (bottom up).
2. Break the given problem into sub-problems. Solving each once and storing the solution (bottom up).
3. Ensure that the sub-problems are solved before solving the problem.

Divide and Conquer vs. Dynamic Programming

Divide and Conquer



Dynamic



Divide and Conquer vs. Dynamic Programming

Divide and Conquer		
Partitions a problem into independent smaller sub-problems.	Doesn’t store solutions of sub-problems. Identical sub-problems may arise, which results in the same computations to be performed repeatedly.	Top-down algorithms which logically progress from the initial instance down to the smallest sub-instances via intermediate sub-instances.
Partitions a problem into overlapping sub-problems.	Stores solutions of sub-problems, which avoids repeated calculations for same problems.	Bottom-up algorithms in which the smallest sub-problems are explicitly solved first and the results of these is used to construct solutions for progressively larger sub-instances.
Dynamic Programming		

R

BS3033 Data Science for Biologists

Dr Wilson Goh
School of Biological Sciences

What is R?

R is a free software environment for statistical computing and graphics.

Cross-platform. Available on UNIX, Windows and Mac OS X

R is regarded as an implementation of the S language which was developed at Bell Laboratories by Rick Becker, John Chambers and Allan Wilks.

Source for R Logo: By Hadley Wickham and others at RStudio - https://www.r-project.org/logo/, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=35599651

R for Data Science

Why R is well-suited for data science?

R has an integrated suite of software facilities for data manipulation, calculation and graphical display.

It has an effective data handling and storage facility.

It has a suite of operators for calculations on arrays, in particular **matrices**.

It is based on a well-developed, simple and effective programming language (called 'S') which includes conditionals, loops, user defined recursive functions and input and output facilities. (Indeed most of the system supplied functions are themselves written in the S language.)

It has a large, coherent, integrated collection of intermediate tools for data analysis.

It has graphical facilities for data analysis and display either directly at the computer or on hardcopy.

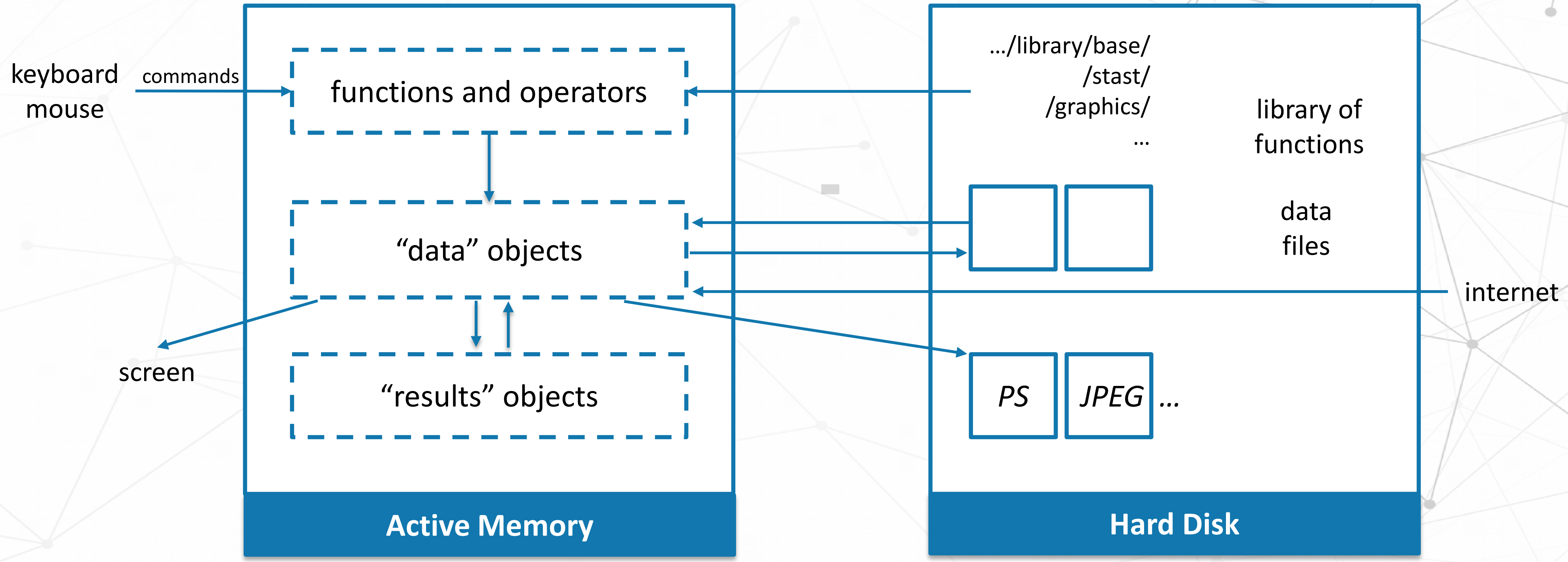
It is more than just a programming language. It is a chimera. Sometimes, it is also referred to as a statistical language.

Refer to https://cran.r-project.org for further information on R.

It does well in **Data handling, Calculations, Tool Development, Graphics and Programming**.

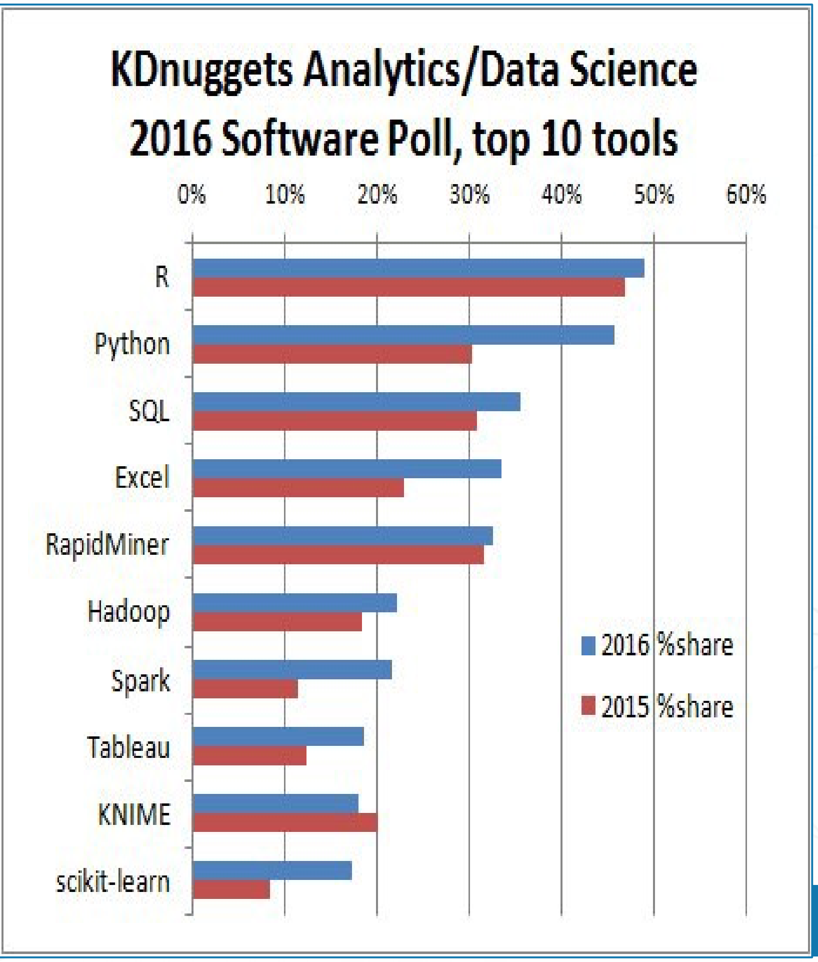
How R works?

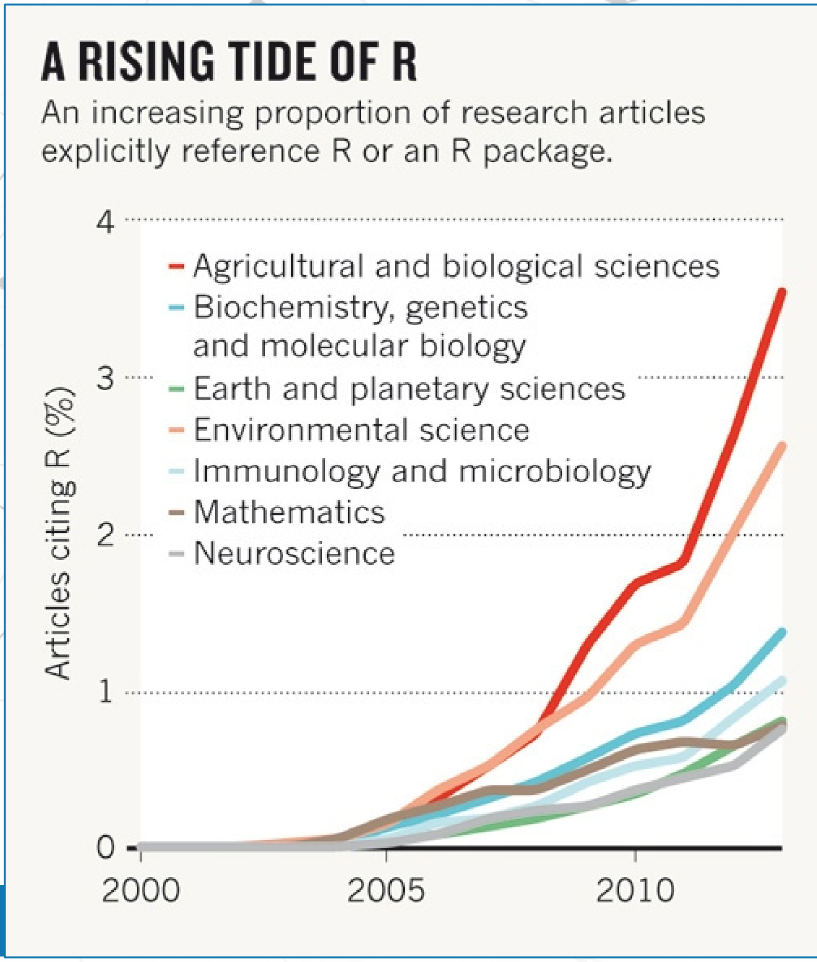
An overview of how R works:



Source: Paradis, Emmanuel. "R for Beginners" (2002)

Popularity of R





Sylvia Tippmann/Source: Elsevier Scopus Database

R in the Biosciences

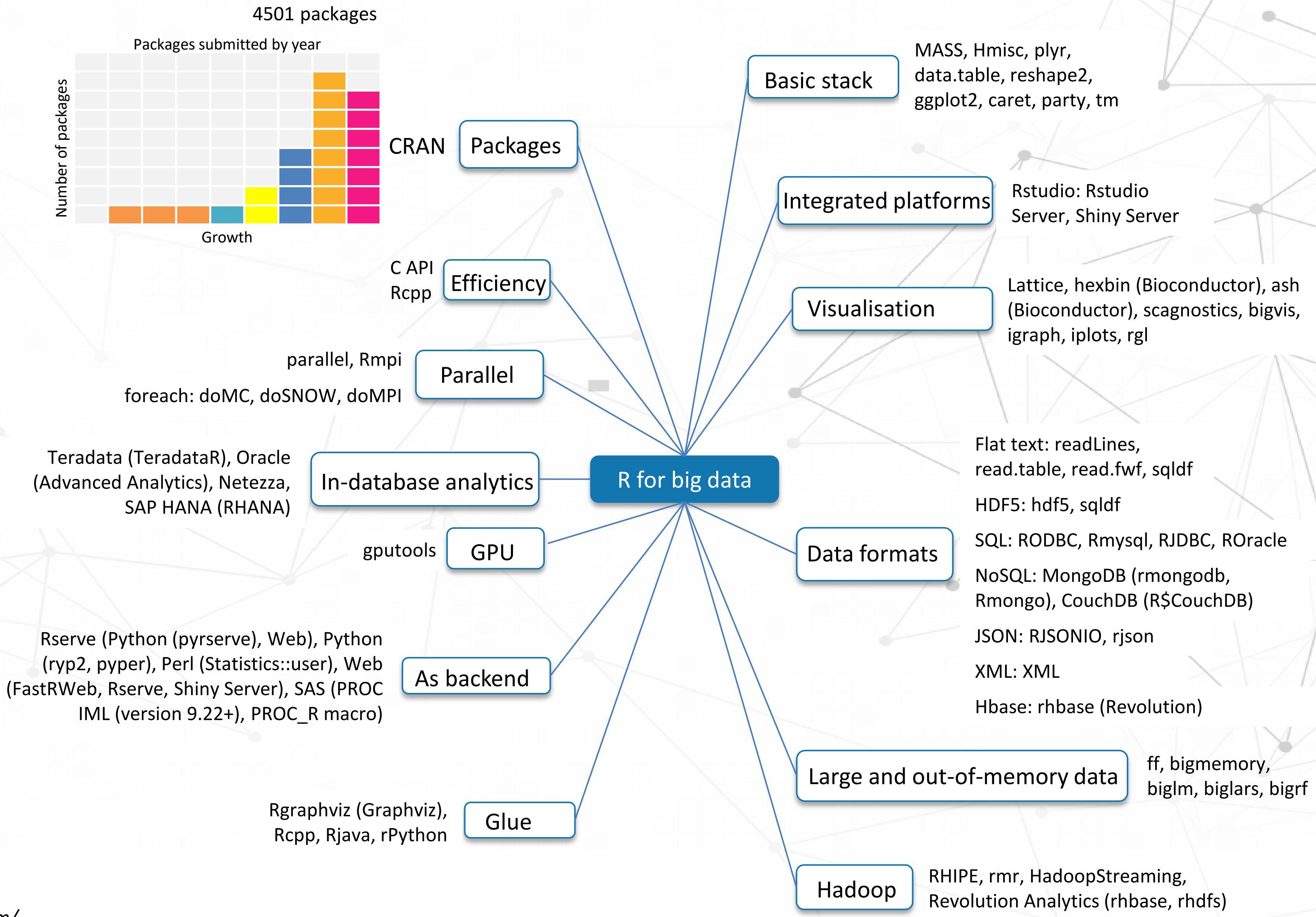
R allows scientists to compare a human and a Neanderthal genome (using **Bioconductor**); to model population growth (**IPMpack**); predict equity prices (**quantmod**); and visualise the results in polished graphics (**ggplot2**) in a few lines of code.

Experts can use R to write up manuscripts, embedding raw code in them to be run by the reader (**knitr**).

Nearly 1 in 100 scholarly articles indexed in Elsevier's Scopus database last year cites R or one of its packages – and in agricultural and environmental sciences, the share is even higher.

Source: http://www.nature.com/news/programming-tools-adventures-with-r-1.16609

R is Powerful



4501 packages
Packages submitted by year
Number of packages
Growth
CRAN
Packages
R for big data
Basic stack
MASS, Hmisc, plyr, data.table, reshape2, ggplot2, caret, party, tm
Integrated platforms
Rstudio: Rstudio Server, Shiny Server
Visualisation
Lattice, hexbin (Bioconductor), ash (Bioconductor), scagnostics, bigvis, igraph, iplots, rgl
Data formats
Flat text: readLines, read.table, read.fwf, sqldf
HDF5: hdf5, sqldf
SQL: RODBC, Rmysql, RJDBC, ROracle
NoSQL: MongoDB (rmongodb, Rmongo), CouchDB (R$CouchDB)
JSON: RJSONIO, rjson
XML: XML
Hbase: rhbase (Revolution)
Large and out-of-memory data
ff, bigmemory, biglm, biglars, bigrf
Hadoop
RHIPE, rmr, HadoopStreaming, Revolution Analytics (rhbase, rhdfs)
Efficiency
C API
Rcpp
Parallel
parallel, Rmpi
foreach: doMC, doSNOW, doMPI
In-database analytics
Teradata (TeradataR), Oracle (Advanced Analytics), Netezza, SAP HANA (RHANA)
GPU
gputools
As backend
Rserve (Python (pyrserve), Web), Python (ryp2, pyper), Perl (Statistics::user), Web (FastRWeb, Rserve, Shiny Server), SAS (PROC IML (version 9.22+), PROC_R macro)
Glue
Rgraphviz (Graphviz), Rcpp, Rjava, rPython

Source: http://www.datasciencecentral.com/

NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

PERL

BS3033 Data Science for Biologists

Dr Wilson Goh
School of Biological Sciences

What is PERL?

PERL stands for “**P**ractical **E**xtraction and **R**eporting **L**anguage”. It is a general-purpose programming language originally developed for text manipulation.

It is now used for a wide range of tasks including system administration, web/ GUI development (under the LAMP framework), network programming, and more.

It is intended to be practical (easy to use, efficient, complete) rather than beautiful (tiny, elegant, minimal).

It supports both procedural and object-oriented (OO) programming.

Its strength is in dealing with text-based data via its regular expression (**regex**) and text operators.

PERL Extensions

PERL modules provide a range of features to help you avoid writing everything yourself. You can download the modules from CPAN http://www.cpan.org/.

Biology-specific modules are hosted on BioPerl at http://bioperl.org/.

How did PERL save the human genome project?

Location

Cambridge, England, in the conference room of the largest DNA sequencing center in Europe.

Problem

Although the two centers use almost identical laboratory techniques, almost identical databases, and almost identical data analysis tools, they still can't interchange data or meaningfully compare results.

Date

Early February, 1996

Occasion

A high level meeting between the computer scientists of the center and the largest DNA sequencing center in the United States.

Solution

Practical **E**xtraction and **R**eporting **L**anguage

Source: http://www.tpj.com

How did PERL save the human genome project?

Flexible

Can deal with unstructured data e.g. text. During early days of HGP, most primary data is text (DNA sequences). Interconverting different data formats is simply a matter of extracting the useful bits from text information. PERL has powerful regex and string manipulation operators.

Forgiving

Biological data is often incomplete and messy (missing data, duplicates, human mistakes). Regex can be written to pick up and correct a variety of common errors in data entry. Of course this flexibility can be also be a curse.

Modular

PERL encourages people to write their software in small packages of code called modules. Modules can be chained together using a master program. Chaining also means PERL modules can be used with other language modules e.g. C. It also makes it easy for people to collaborate.

Source: https://web.stanford.edu/class/gene211/handouts/How_Perl_HGP.html

How did PERL save the human genome project?

Simple

PERL doesn't require you to declare all your function prototypes and data types in advance, new variables spring into existence as needed, calls to undefined functions only cause an error when the function is needed.

Fast Prototyping

Because PERL is quick and dirty, it often makes sense to prototype new algorithms in PERL before moving them to a faster compiled language. Sometimes it turns out that PERL is fast enough so that the algorithm doesn't have to be ported; more frequently one can write a small core of the algorithm in C, compile it as a dynamically loaded module or external executable, and leave the rest of the application in PERL.

Web/Data-based Capabilities

PERL is a good language for Web CGI scripting, and is growing in importance as more labs turn to the Web for publishing their data.

Source: https://web.stanford.edu/class/gene211/handouts/How_Perl_HGP.html

The Fall of PERL

In bioinformatics, where PERL’s position as the most popular scripting language powered many 1990s breakthroughs like genetic sequencing, PERL has been supplanted by Python and the statistical language R (a variant of S-plus and descendent of S, also developed in the 1980s).

Other programming languages (Python, Ruby and even PHP) have matured quite a bit in recent years, and have lots of libraries (PERL's biggest asset during its "boom period" was CPAN,) and large followings. It has now lost this advantage.

Too much flexibility can be confusing. PERL’s mantra is “*There is more than one way to do it*”. PYTHON’s is “*There is one obvious way to do it*”. In a day and age where collaboration is essential, which language has more relevance?

Shift in focus from text data to numeric data (PERL’s regex advantage is less needed).

Source: https://www.fastcompany.com/3026446/the-fall-of-perl-the-webs-most-promising-language

NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

Python

BS3033 Data Science for Biologists

Dr Wilson Goh
School of Biological Sciences

What is Python?

It is free, powerful, widely used and has an elegant syntax (uses indents instead of brackets for loops).

Extensions to Python made it data science compatible. Chief amongst these include pandas, SciPy, NumPy, MatplotLib, Seaborn.

Google, NASA, Yahoo, Electronic Arts, some UNIX scripts etc. use Python.

Python is an interpreted high-level programming language for general-purpose programming.

It is named after a British comedy “Monty Python’s Flying Circus”.

Biopython is a set of freely available tools for biological computation written in Python.

Refer to: http://www.python.org

pandas

pandas is an open source library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language.

It adds data structures and tools designed to work with table-like data (similar to series and data frames in R).

It allows handling of missing data.

Refer to: http://pandas.pydata.org/

NumPy

NumPy introduces objects for multidimensional arrays and matrices, as well as functions that allow to easily perform advanced mathematical and statistical operations on those objects.

Many python libraries are built on NumPy.

It provides vectorisation of mathematical operations on arrays and matrices which significantly improves the performance.

Refer to: http://www.numpy.org/

SciPy

SciPy (pronounced “Sigh Pie”) is a Python-based ecosystem of open-source software for mathematics, science, and engineering.

It is a collection of algorithms for linear algebra, differential equations, numerical integration, optimisation, statistics and more.

It is built on NumPy.

Refer to: https://www.scipy.org/scipylib/

Matplotlib

Matplotlib is a Python 2D plotting library, which produces publication quality figures in a variety of hardcopy formats.

It offers a set of functionalities similar to those of MATLAB.

Examples: Line plots, Scatter plots, Bar charts, Histograms, Pie charts etc.

It is relatively low-level and requires effort to create advanced visualisation.

Refer to: https://matplotlib.org/

Seaborn

Seaborn is based on Matplotlib.

It provides high level interface for drawing attractive statistical graphics.

It is similar (in style) to the popular ggplot2 library in R.

Refer to: https://seaborn.pydata.org/

Jupyter

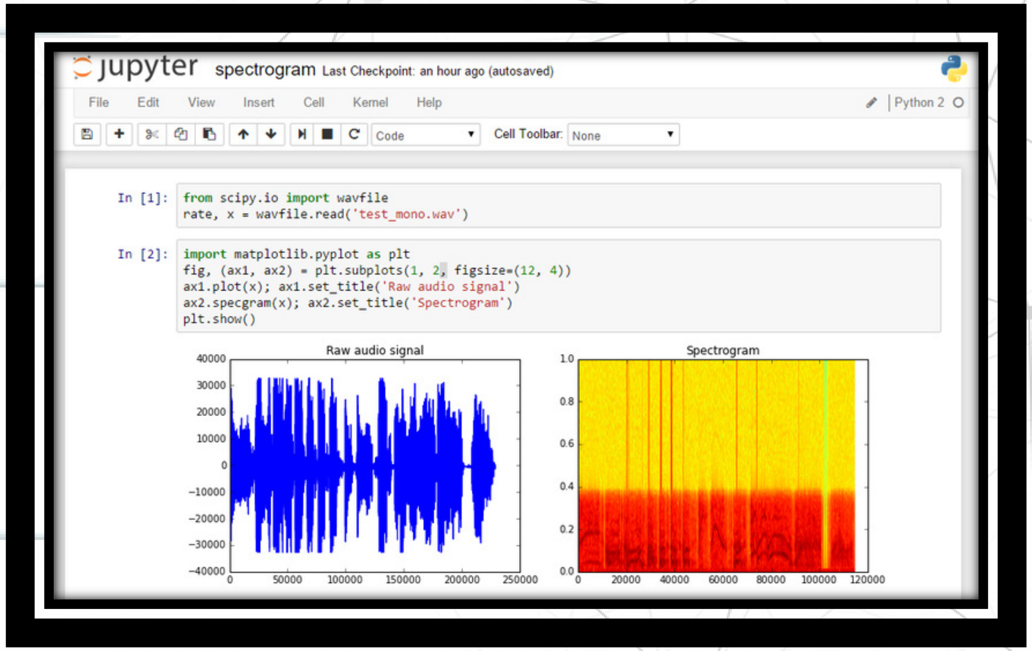
Jupyter is built on top of iPython (an interactive shell similar to R's).

It was born in 2014.

It provides interactive data science and scientific computing across all programming languages (not just Python).

It is an IDE and combines easy interface between graphics and code. It can be used to produce presentations.

Refer to: http://jupyter.org/



jupyter
spectrogram Last Checkpoint: an hour ago (autosaved)
File Edit View Insert Cell Kernel Help
Python 2
Code
Cell Toolbar: None
Raw audio signal
Spectrogram

```
In [1]: from scipy.io import wavfile
        rate, x = wavfile.read('test_mono.wav')

In [2]: import matplotlib.pyplot as plt
        fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 4))
        ax1.plot(x); ax1.set_title('Raw audio signal')
        ax2.specgram(x); ax2.set_title('Spectrogram')
        plt.show()
```

BioPython

Biopython is a set of freely available tools for biological computation.

It is similar to BioPerl and provides many mechanisms for dealing with biological sequences and format conversions.

It also allows easy connection to databases, e.g. ExPASy, Entrez, Pubmed, etc.

Refer to: http://biopython.org/

NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

Comparing R and Python

BS3033 Data Science for Biologists

Dr Wilson Goh
School of Biological Sciences

Python vs. R

Python vs. R for Predictive Modelling

Feature	Which is better?
Model Building	Both are Similar
Model Interpretability	R
Production	Python
Community Support	R
Libraries	Both are Similar
Visualisations	R
Learning Curve	Python

Python vs. R

Feature	Python	R
Model Building	Both are similar	Both are similar
Model Interpretability	Not better than R	R is better
Production	Python is better	Not better than Python
Community Support	Not better than R	R has good community support over Python
Data Science Libraries	Both are similar	Both are similar
Data Visualisations	Not better than R	R has good data visualisation libraries and tools
Learning Curve	Learning Python is easier than learning R	R has a steep learning curve

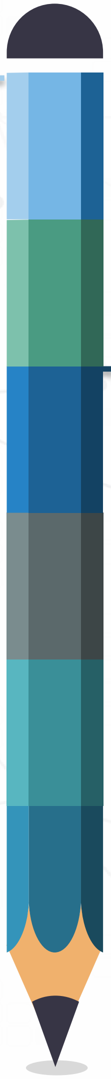
NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

Summary

BS3033 Data Science for Biologists

Dr Wilson Goh
School of Biological Sciences

Key Takeaways from this Topic



1. System, Architectural, Application, and Statistical are the four classifications of programming languages. The choice of language category depends on what you want to use it for.

2. R is an integrated suite of software facilities for data manipulation, calculation and graphical display. It does well in Data handling, Calculations, Tools, Graphics and Programming which makes it well-suited for data science.

3. In bioinformatics, PERL powered many 1990s breakthroughs like genetic sequencing, it has been supplanted by Python and the statistical language R. This along with shift in focus from text data to numeric data led to the fall of PERL.

4. Python is a general-purpose language, which is designed to be simple to read and write. The designers placed less of an emphasis on conventional syntax, which makes it easier to work with, even for non-programmers or developers and makes it popular.

5. R and Python are both open-source languages used in a wide range of data analysis fields. Their main difference is that R has traditionally been geared towards statistical analysis, while Python is more generalist. Both comprise a large collection of packages for specific tasks and have a growing community that offers support and tutorials online.

Readings

The R Project for Statistical Computing

The R Journal

An introduction to R

Reading Resources

Little Book of R for Bioinformatics

Going Further

Although we will not teach compiled languages. You may be interested to find out more what it is.

Python is seen as a potential successor to R in data science. Why is that so?

PERLs particular strength is in text-parsing (due regex), something R is weak in. Find out how regex works, and why it is useful in biological data analysis. If you like, you can try writing a regex to parse a GeneBank flatfile.

What is LAMP? Does what it stand for? Who still uses it for web development these days?