Computing Language for Open Science :Open Discussion

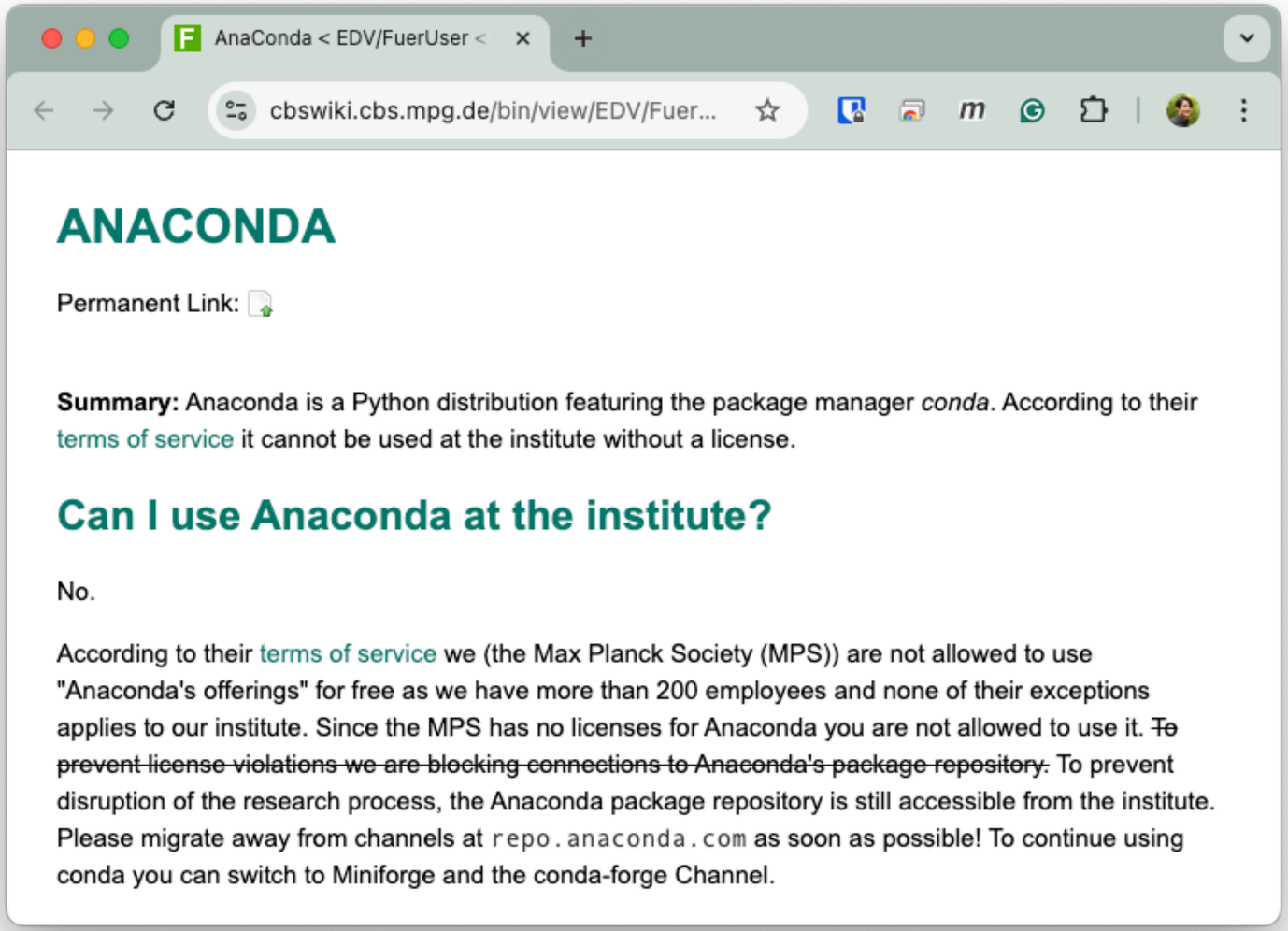
Open Science workshop @ TeaP2025
Goethe University Frankfurt, FFM

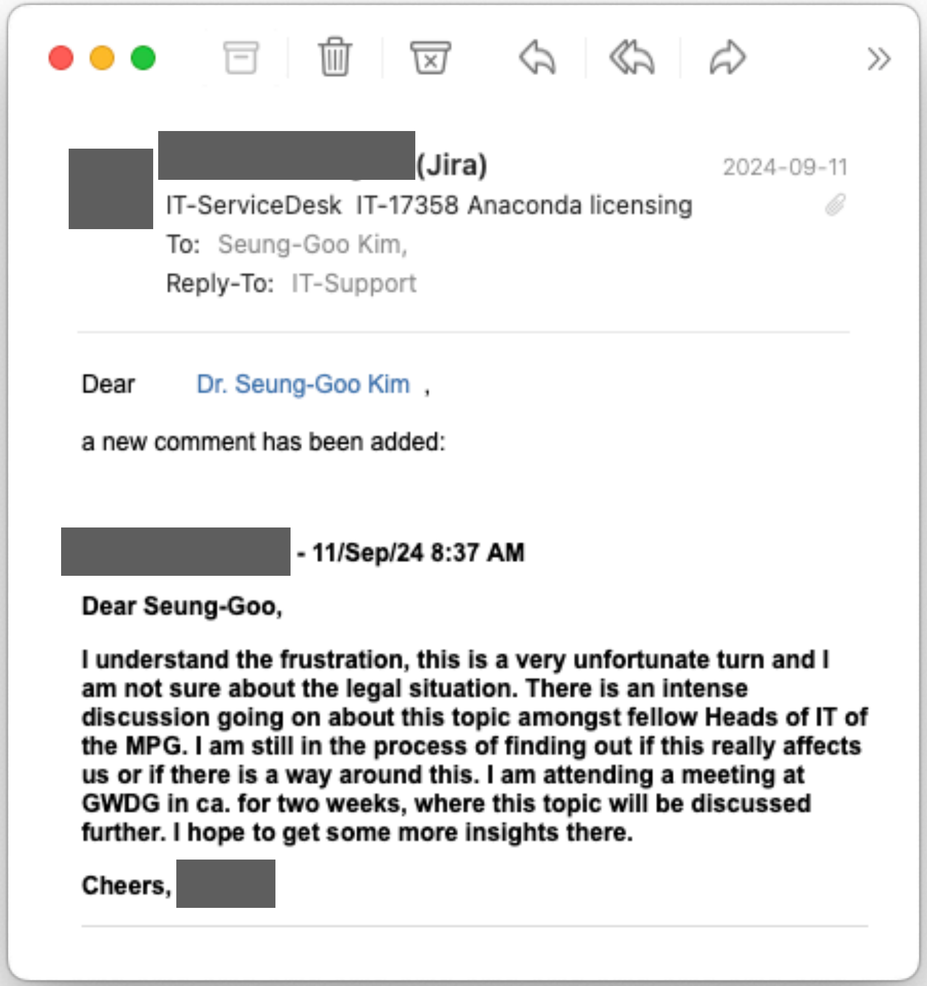
2025-03-09 seung-goo.kim@ae.mpg.de

Motivation

Anaconda drama

2024-09-03





ANACONDA. Products Solutions Resources Partners Company Free Download Sign Up Sign In

Anaconda Terms of Service FAQs

See Pricing > Contact Sales

Is Anaconda still free?

Does Anaconda require academic entities with 200 or more employees to purchase a business or enterprise license?

No. Anaconda does not require academic institutions and universities to purchase a business or enterprise license for access to our package repository, regardless of their size, when used in course curricula. We maintain a free-use policy for educational entities when Anaconda is used in course curricula, including teaching, learning, and research at accredited educational institutions worldwide. This free-use policy applies even to large universities with 200 or more employees. The 200-employee threshold for paid licenses primarily applies to commercial organizations. **However, it's important to note that paid licenses may be required for specific use cases within academic settings, such as embedding Anaconda's products, mirroring them, or providing third-party access beyond standard educational use.**

https://www.anaconda.com/pricing/terms-of-service-faqs [2024-10-01]

What computing language should we use?

What computing language should we use for open science?

Why do I associate Python with Open Science?

- Until 2000s?: We used proprietary software all the time (MATLAB, BESA, BrainVoyager, SPSS, ..., Microsoft Office, Windows, Mac{HW+SW})
- 2012: Special Section on "Replicability in Psychological Science" in Perspectives on Psychological Science, https://journals.sagepub.com/doi/10.1177/1745691612465253
- 2015: Open Science Collaboration, "Estimating the reproducibility of psychological science", Science, https://doi.org/10.1126/science.aac4716
- For **transparency**, the whole analysis needs to be replicable on others' system only using open-access software (i.e., docking everything won't violate any copyright laws)
  => "MATLAB is proprietary🤑. Thus, **MathWorks is the enemy of OpenScience👹!**"
  => "Python is free to use!😇 Everyone MUST use Python🐍 not MATLAB!"
- Now if we need to pay for **Anaconda** (or maybe Python🐍 itself), then why did we waste tons of hours for retraining? Was it for Open Science? Or for a company?

Skepticism triggered! 🤔🧐😐

A punch in the face wakening up from the naiveness

- "*If you're not paying for the product, then you are the product.*" (Tristan Harris)
- Is every open-access project👹 just a long-term investment for the market dominance?
- And are the open-science supporters the products of the market? 😱
- ...but let's get real and let's think about the problem a bit more

https://stock.adobe.com/de/images/funny-man-is-getting-punch-in-face-with-fist/114609336

Why did I start using those languages?

But first...

- **MATLAB**⛰️: because of **SPM**, **SurfStat**, **EEGLAB**, **FieldTrip**, and other popular toolboxes
- **Python**🐍: because of **MNE-python**, **TensorFlow**, **Essentia**, and other popular packages
- **R®**: because of Linear Mixed-effects Models (**lme4**)
- **Bash**🧊: because of **FSL**, **ANTs**, **FreeSurfer**
- I didn't choose those languages. I'm just using popular packages.
- The developers of the popular packages/libraries chose their languages!
- So, I'm not very happy to hear a language I know is inherently evil😡 (while I'm perfectly okay when I say that language sucks).

And why do we want to learn languages?

Why would anyone want to write a script when you have a GUI?

- *Well, at least for me...*
  - I want to be able to replicate the whole process of preprocessing, analysis, ...
  - Know a bit more about the under-the-hood.
  - I want to "hack a little" here and there.
  - Because I'm lazy and I want a single-liner for a whole paper

Open questions

- Is proprietary software evil?
- Can the transparency be implemented only by open-access software?
- What is the best language?

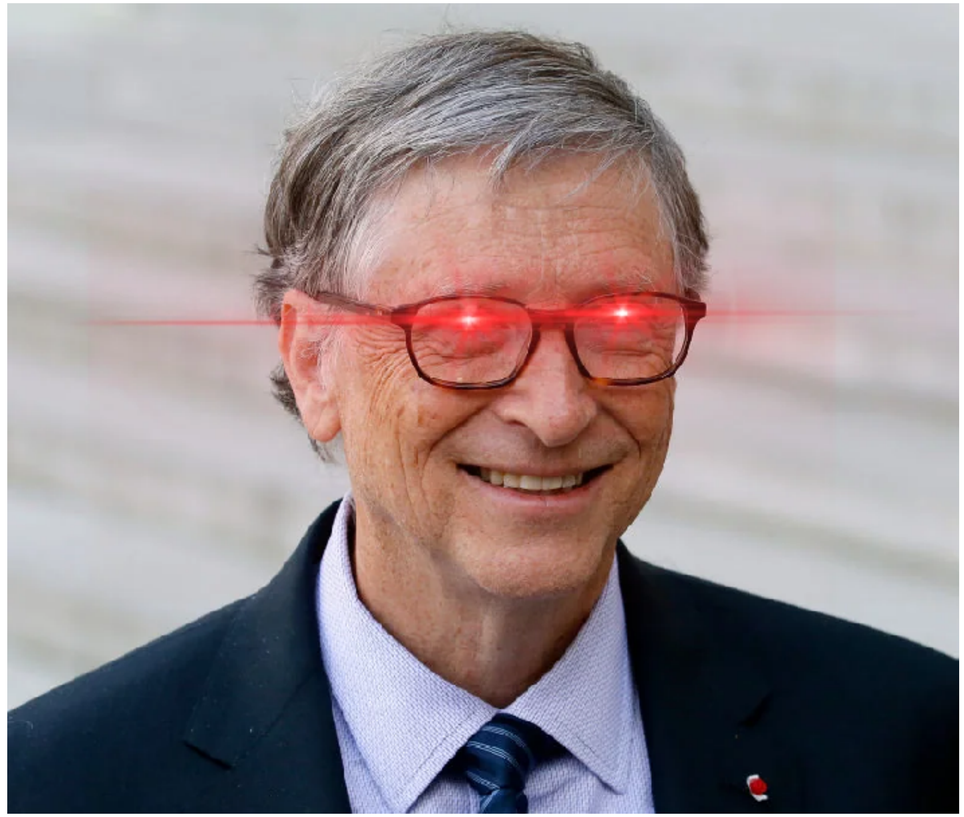
Open questions

- Is proprietary software evil?
- Can the transparency be implemented only by open-access software?
- What is the best language?

Is capitalism inherently evil?

Why some people feel software should be ethically free?

St. IGNUcius, the Church of EMACS (2012)
Free Software Foundation (1985-)



Bill Gates with red eyes, r/linuxmemes

GNU

A collection of free software

- GNU: "GNU's not Unix!" because it's Unix-like but "free" (as in freedom).
- GNU Manifesto (1985) by Richard Stallman and colleagues.
- GNU-linux: GNU software + Linux-kernel like GNU-Bash

GNU Goat (from "gnu goat")

How do they make open access software?

Are they just heavenly creatures living off their own niceness?






- **Linus Torvalds** (the creator of Linux) gets 1.7M USD in compensation as a Fellow of Linux Foundation
- **Guido van Rossum** (the creator of Python) worked at own startups, and Google[05-12], Dropbox[13-19], Microsoft[20-], 1-5M USD/yr?
- **Ross Ihaka** & **Robert Gentleman** (the co-creators of R) worked as statistic professors at University of Auckland, NZ
- **Travis Oliphant** (the creator of NumPy, SciPy and a co-founder of NumFOCUS, Anaconda[CEO:12-17]) gets compensation from as a staff at NumFOCUS.
- **Bjarne Stroustrup** (the creator of C++) worked as a programmer at Bell Labs, as a professor at Texas A&M University, and Columbia University.

When you see a nice open-access project...

How long will it be maintained and last open-access?

- A non-profit foundation?
- A for-profit start-up?
- A fixed-term research funding?
- A tenured professor's hobby?
- Small "Truck Factor" (or "Bus factor", the number of "critical" developers)? (See https://peerj.com/preprints/1233.pdf)

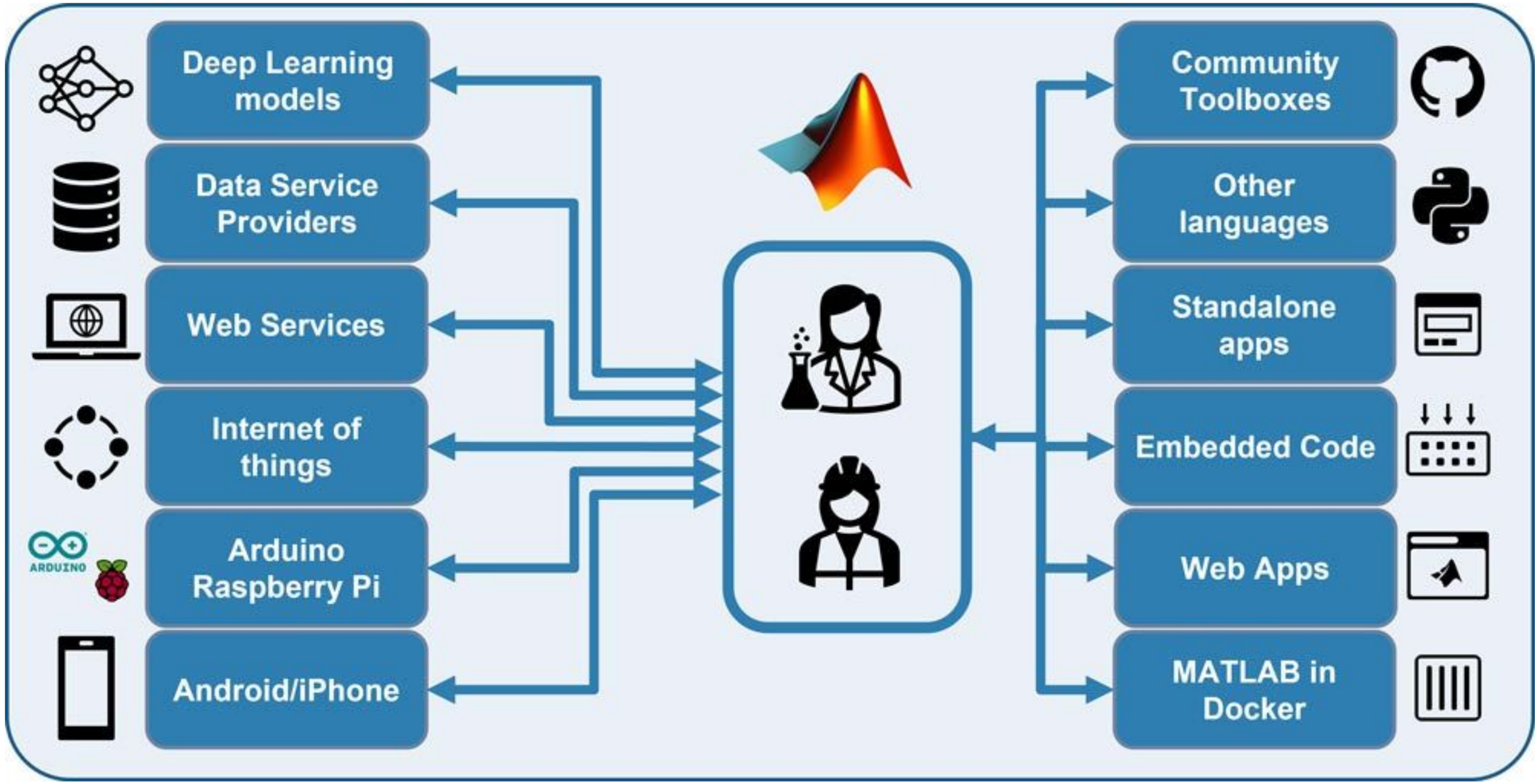
Your thoughts? 🧐💭

Open questions

- Is proprietary software evil? (should/can we be non-evil in the long run?)
- Can the transparency be implemented only by using open-access languages?
- What is the best language?

MATLAB for Open Science?

At least they also know it "sells" for now



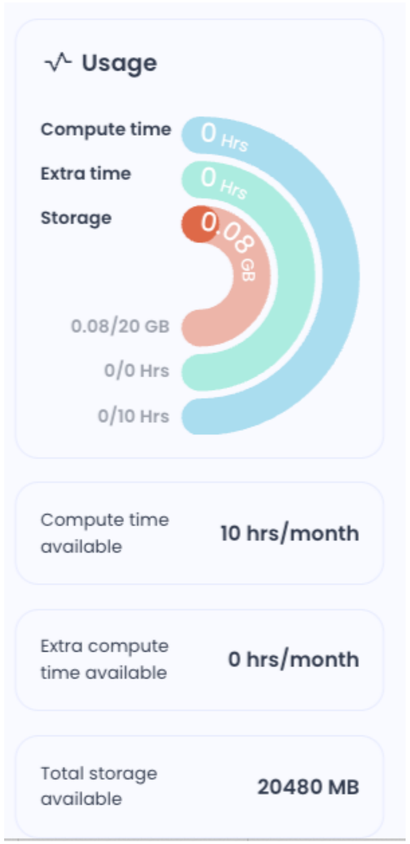
https://de.mathworks.com/discovery/open-science.html

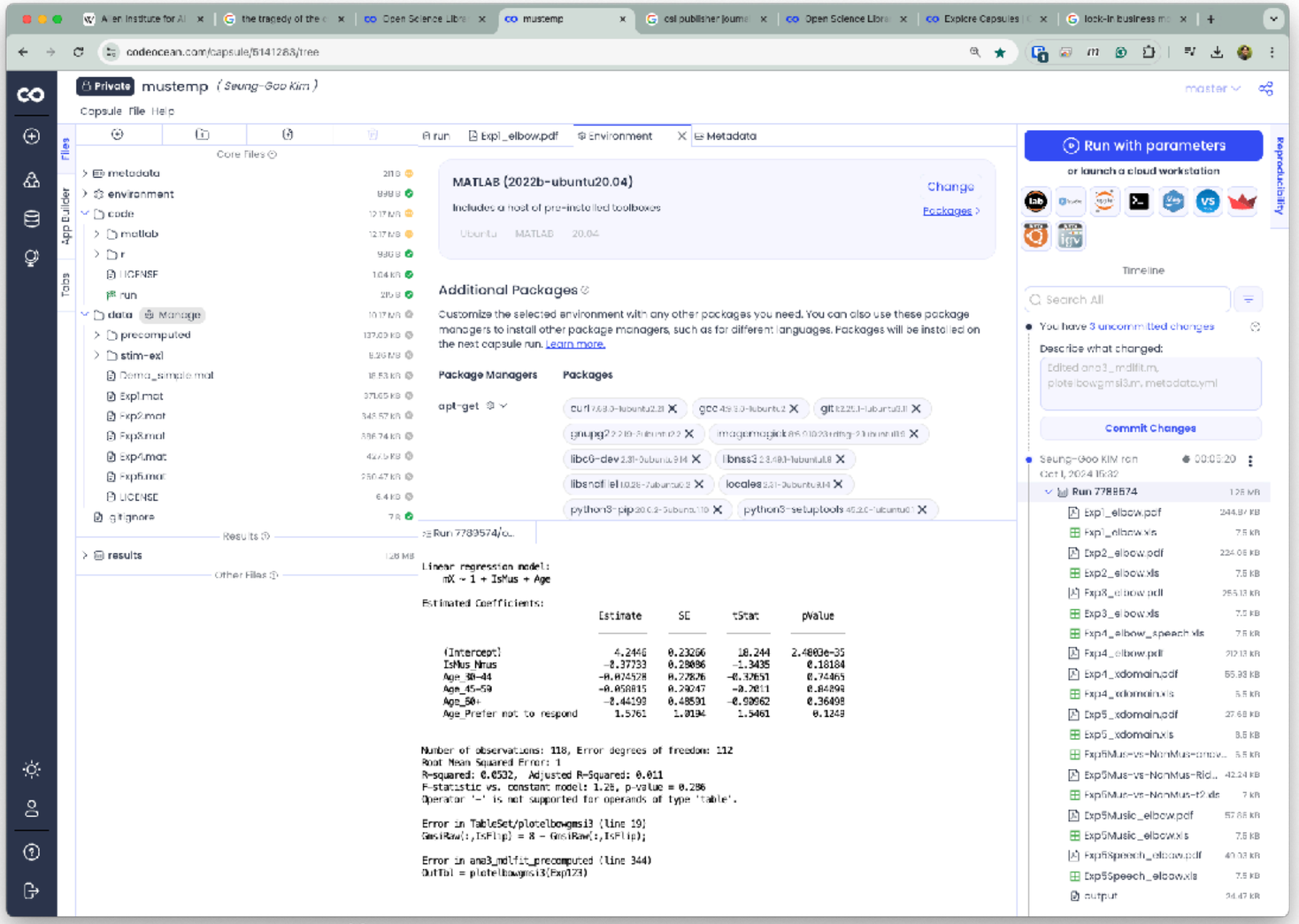
CodeOcean 

codeocean.com/capsule/5141283/

CodeOcean: Open Science Library

It's free for now... but why not institute-level license?





codeocean.com/explore?page=1&filter=all

Open Science Library

All | Search All/Author/Article/Journal/Tags/Title

All | Mathematics | Physics | Engineering | Bioinformatics | Medical Sciences | Social Sciences | Earth Sciences | Computer Science | Biology | Finance

All | Associated with article

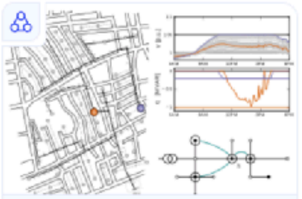
Engineering 15 | Jan | 2019

MATLAB Code---Fault diagnosis of power transformers with membershi...

A reference fault set is provided, and the fault diagnosis is implemented by calculating the membership of the DGA data to the reference fault set.

Enwen LI

Open Capsule



Engineering 14 | Oct | 2019

On the Need for Communication for Voltage Regulation of Power...

Benchmark for Volt/VAR control of power distribution networks. The benchmark illustrates the suboptimality of purely local (fully decentralized) volt/VAR strategies when

Saverio Bolognani, Ruggero Carli, Guido ...

Open Capsule

Associated article published in IEEE Transactions on Control of Network Systems

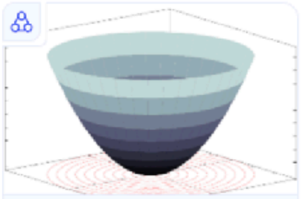
Bioinformatics 5 | Aug | 2024

"MetaModality: Enhancing Cancer Detection through Integrated...

This project places a special emphasis on early cancer detection and understanding cancer mechanisms by harnessing the power of diverse metabolomics platforms. The study

Parisa Shahnazari

Open Capsule



Mathematics 13 | Feb | 2020

"Model reduction for complex hyperbolic networks" Companion...

Numerical experiments for: C. Himpe and M. Ohlberger, "Model reduction for complex hyperbolic networks", Proceedings of the European Control Conference (ECC): 2739-

Christian Himpe

Open Capsule

Associated article published in 2014 European Control Conference (ECC)

Engineering 30 | Aug | 2023

"Multi-layer double deep Q network for active distribution network equivalen...

Multi-layer double deep Q network for active distribution network equivalent modeling with internal identification for EV loads

Jiehui Zheng & Wenhao Wang

Open Capsule

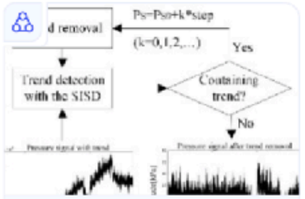
Medical Sciences 26 | Oct | 2023

"Registering the Hurt" Project: Pilot Analysis of COVID-19 Study...

The reproducibility crisis has spurred initiatives for enhanced transparency in scientific research. Among these, study preregistration, or simply registration, has

Luciana A.C. Machado

Open Capsule



Engineering 2 | Oct | 2023

"Trend Detection and Adaptive Removal in Pressure Signal for Pipelin...

The presence of a trend component in the pressure signal has a detrimental effect on the accuracy of pipeline leak detection based on the negative pressure wave (NPW). To

Lei Zhang

Open Capsule

Associated article published in IEEE Sensors Journal

Biology 17 | Sep | 2024

"Variant-to-function dissection of rare non-coding GWAS loci with high...

Input files and code to reproduce all the results figure panels of main figure 6.

Manuel Tardaguila

Open Capsule

Earth Sciences 1 | May | 2020

#shareEGU20 - impact of flood extent on population exposure

This notebook displays how output from various flood models can lead to varying estimates of people affected by a flood.

Jannis Hoch, Dirk Eilander & Hiroaki Ikeu...

Open Capsule

Associated article published in Natural Hazards and Earth System Sciences

Computer Science 9 | Jun | 2020

$H\infty$ tracking control for linear discrete-time systems: model-free Q...

$H\infty$ tracking control for linear discrete-time systems using adaptive dynamic programming (ADP)

Yunjie Yang

Open Capsule

Associated article published in IEEE Control Systems Letters

Engineering 18 | Oct | 2022

'Algorithm 1: OS-IM' code based on Matlab for radar range-spread targe...

Detection of radar range-spread targets based on order statistics Code based on Matlab for 'Algorithm 1: OS-IM'.

Shaoqiang Chang

Open Capsule

Engineering 12 | Oct | 2022

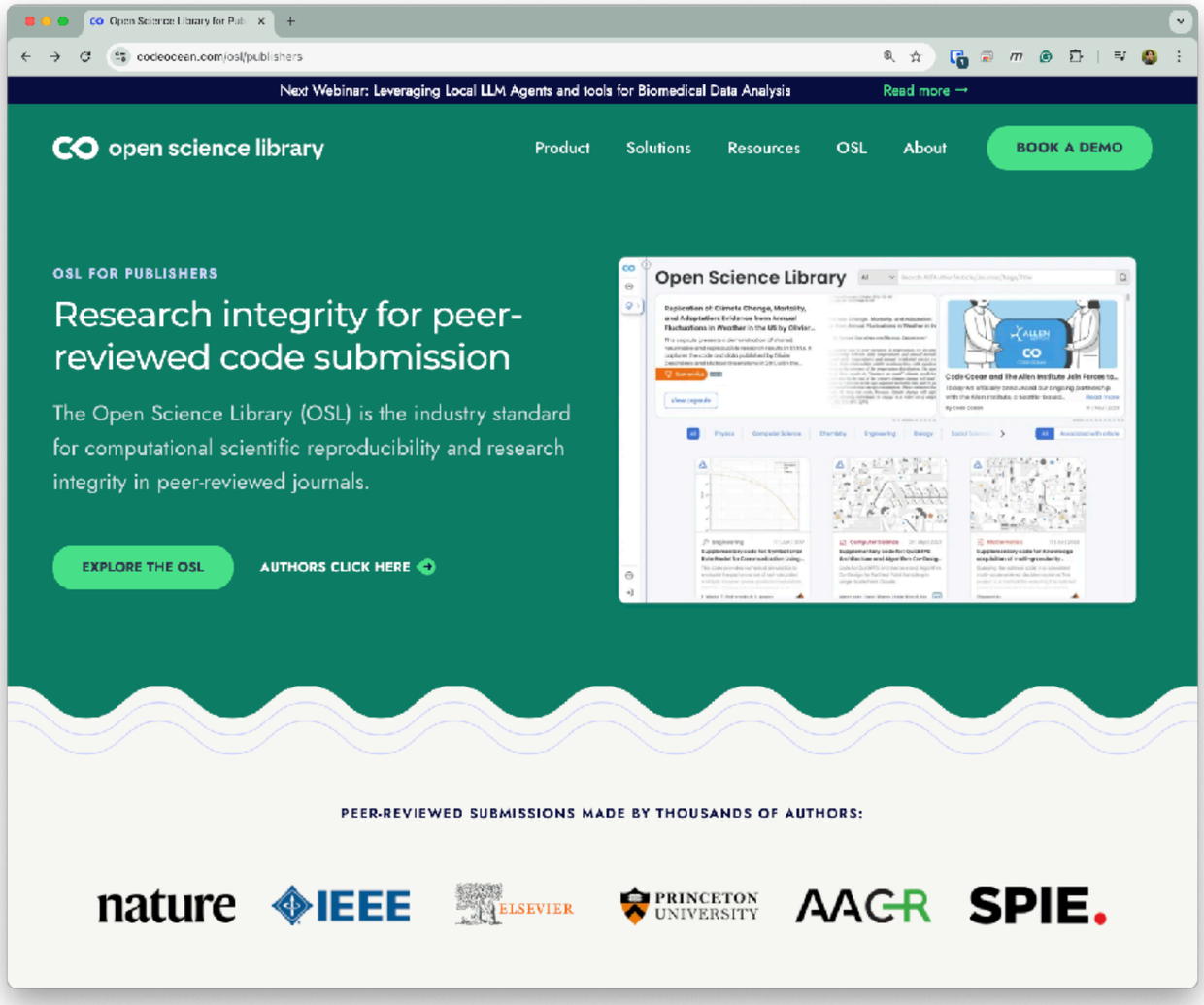
(Global Routing Scheme Code) Link Stability based Optimized Routing...

An example matlab code and the actual matlab code of the global routing scheme of the paper "Link Stability based Optimized Routing Framework for Software Defined

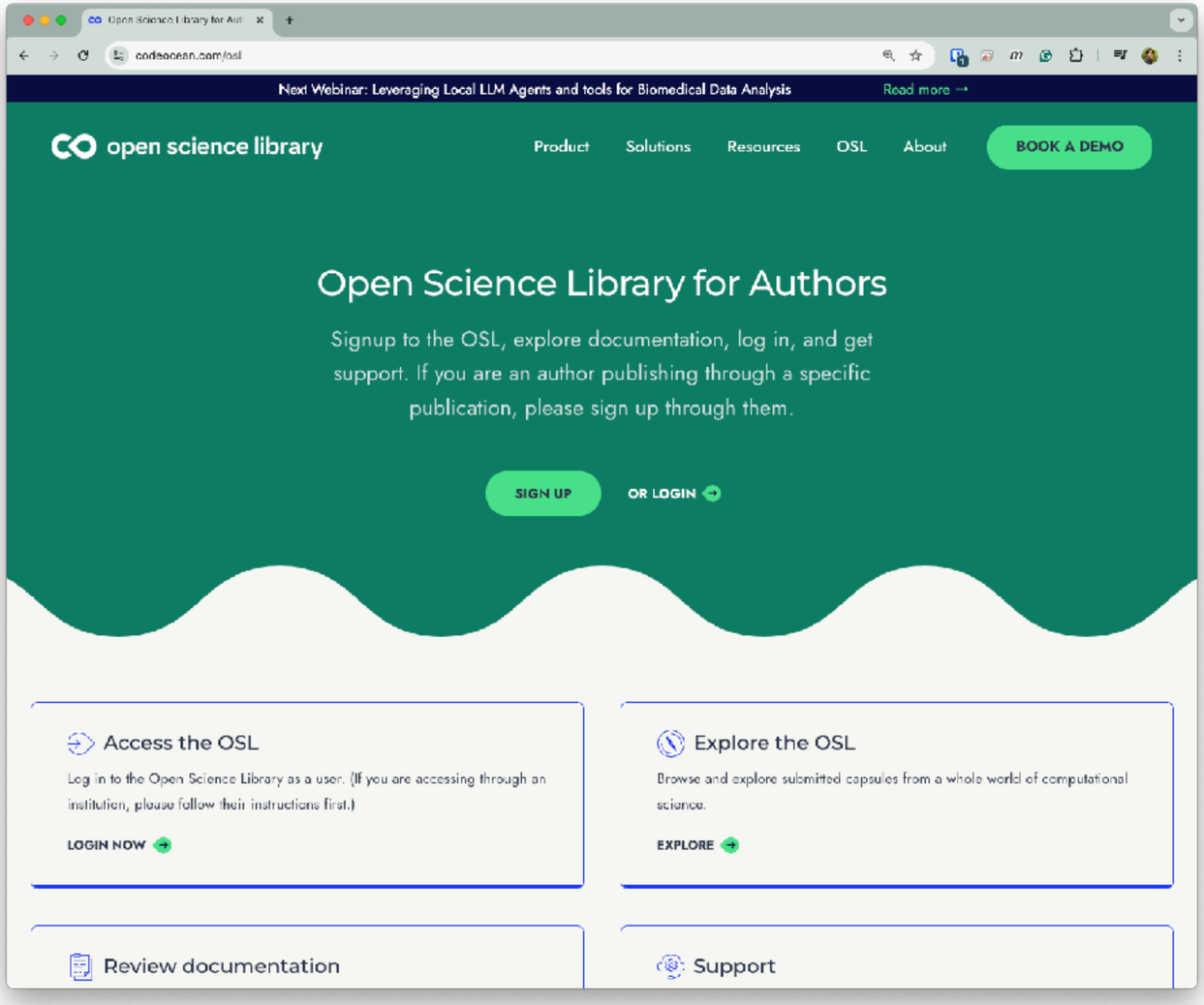
Kalupahana Liyanage Kushan Sudheera...

Open Capsule

For journals



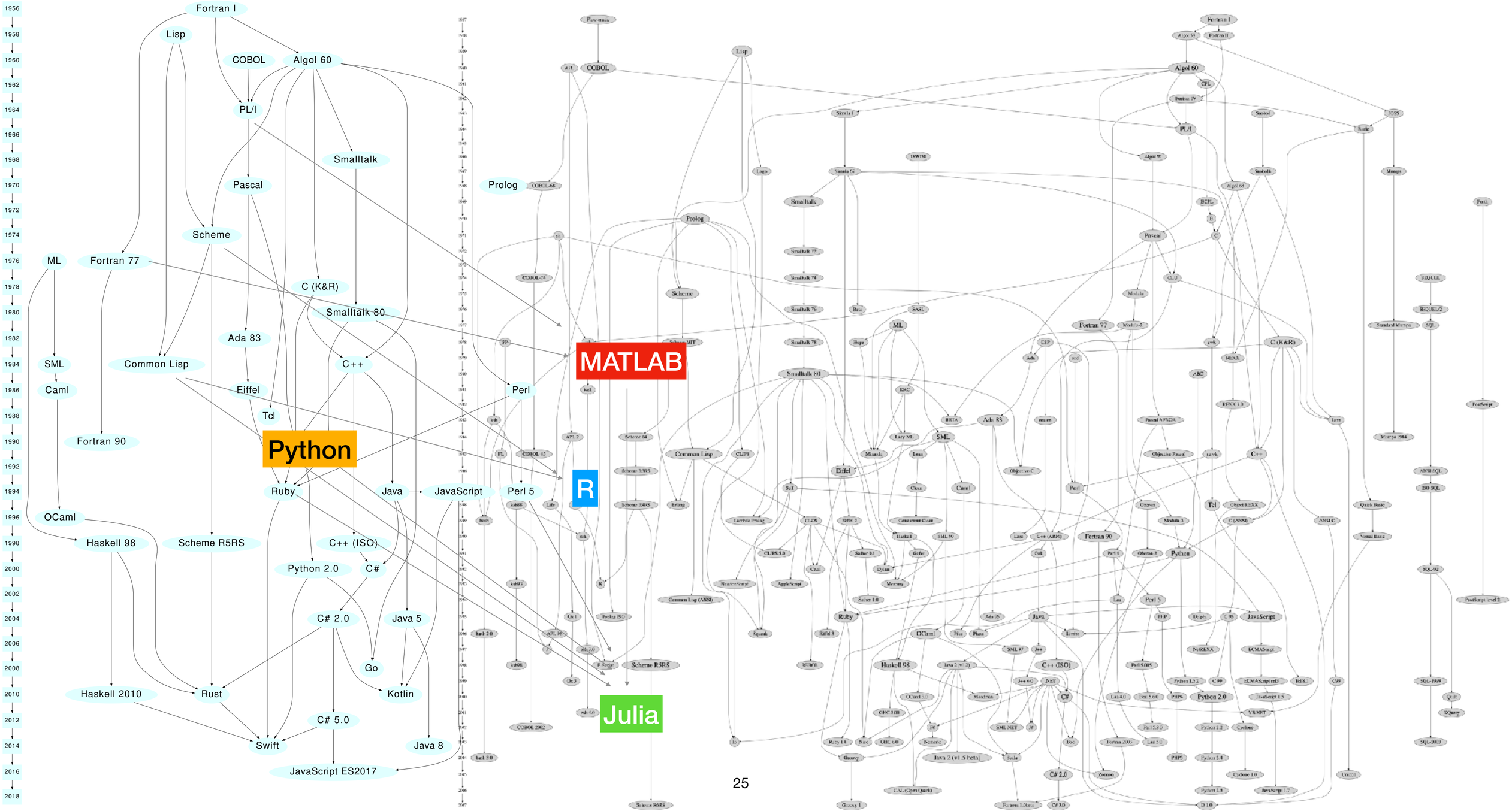
For authors



Your thoughts? 🧐💭

Open questions

- Is proprietary software evil? (should/can we be non-evil in the long run?)
- Can the transparency be implemented only by open-access software?
- What is the best language?


1956
1958
1960
1962
1964
1966
1968
1970
1972
1974
1976
1978
1980
1982
1984
1986
1988
1990
1992
1994
1996
1998
2000
2002
2004
2006
2008
2010
2012
2014
2016
2018
Fortran I
Lisp
COBOL
Algol 60
PL/I
Smalltalk
Pascal
Scheme
ML
Fortran 77
C (K&R)
Smalltalk 80
Ada 83
SML
Common Lisp
C++
Caml
Eiffel
Perl
Tcl
Fortran 90
Python
Ruby
Java
JavaScript
Perl 5
OCaml
Haskell 98
Scheme R5RS
C++ (ISO)
Python 2.0
C#
C# 2.0
Java 5
Go
Haskell 2010
Rust
Kotlin
C# 5.0
Swift
Java 8
JavaScript ES2017
Prolog
MATLAB
R
Julia

What is the best computing language?

Computing languages as meta-tools to build your own tools

- As an end-user of a language, we want tools with high...
  - **usability**: easy to learn/read/write/debug/maintain
  - **accessibility**: can be used without paid licenses
  - **sustainability**: 50+ years
  - **performance**: can handle large data fast
- ... in a good balance.

Language	Latest release	GPL/DSL	Usability	Accessibility	Sustainability	Performance
Python	2024-10-01	GPL	?	(600$/yr/user?)	?	mid
GNU-c-LISP	2023-01-13	GPL	?	0	?	?
C	2024-02-21	GPL	?	0	?	high
C++	2023-03-19	GPL	?	0	?	high
C# (Win)	2023-11-14	GPL	?	0	?	high
Swift (Mac)	2024-03-01	GPL	?	0	?	high
Julia	2024-10-01	GPL	?	0	?	mid
Java	2024-09-17	GPL	?	180$/yr/user	?	high
R	2024-06-14	DSL	?	0	?	?
GNU Octave	2024-06-07	DSL	?	0	?	mid
Fortran	2023-11-17	DSL	?	0	?	high
Mathematica	2024-07-01	DSL	?	839$/yr/user	?	?
Stata	2023-04-25	DSL	?	925$/yr/user	?	?
WolframOne	2024-07-31	DSL	?	1710$/yr/user	?	?
MATLAB	2024-09-12	DSL	?	263-3k$/yr/user	?	mid

HCI research

ChatGPT4o says...

- The **Cognitive Dimensions of Notations (CDN)** is a widely-used framework for evaluating the usability of notational systems, including DSLs (domain-specific languages). It provides a structured way to analyze how the design of a DSL affects the cognitive load on users. Some key dimensions include:
  - **Closeness of Mapping**: How well the language matches the mental model of the domain.
  - **Consistency**: How predictable and uniform the language is.
  - **Abstraction Gradient**: The complexity and flexibility in expressing concepts.
  - **Error-proneness**: How likely users are to make mistakes.
  - **Viscosity**: Resistance to change or how hard it is to make modifications.

Closeness of Mapping:
Let's invert a matrix !

DEMO

Moore-Penrose inverse (pseudo-inverse)

$$\mathbf{A}^{+}; \mathbf{A} \in \mathbb{R}^{10\times5}, a_{i,j} \sim \mathcal{U}$$

<Translated by ChatGPT4o> 

DEMO EXAMPLES $\mathbf{A}^{+}; \mathbf{A} \in \mathbb{R}^{10\times5}, a_{i,j} \sim \mathcal{U}$

<Python>

```python
import numpy as np
A = np.random.rand(10, 5)
A_pinv = np.linalg.pinv(A)
print(A_pinv)
```

<R>

```r
A <- matrix(runif(50), 10, 5)
A_pinv <- MASS::ginv(A)
A_pinv
```

<Julia>

```julia
using LinearAlgebra
🙂 = rand(10, 5)
🙃= pinv(🙂)
display(🙃)
```

<GNU-Octave/MATLAB>

```octave
A = rand(10, 5);
A_pinv = pinv(A);
disp(A_pinv);
```

<Perl>

```perl
use strict;
use warnings;
my @A = map { [ map { rand() } 1..5 ] } 1..10;
my $A_pinv = pseudoinverse(\@A);
for my $row (@$A_pinv) {
    print join("\t", @$row), "\n";
}
sub pseudoinverse {
    my ($matrix) = @_;
    my $A_T = transpose($matrix);
    my $A_TA = matrix_multiply($A_T, $matrix);
    my $A_TA_inv = invert($A_TA);
    return matrix_multiply($A_TA_inv, $A_T);
}

sub transpose {
    my ($matrix) = @_;
    my @transposed;
    for my $i (0..$#{$matrix->[0]}) {
        push @transposed, [ map { $_->[$i] } @$matrix
];
    }
    return \@transposed;
}

sub matrix_multiply {
    my ($matrix1, $matrix2) = @_;
    my @result;
    for my $i (0..$#$matrix1) {
        for my $j (0..$#{$matrix2->[0]}) {
            $result[$i][$j] = 0;
            for my $k (0..$#{$matrix2}) {
                $result[$i][$j] += $matrix1->[$i][$k]
* $matrix2->[$k][$j];
            }
        }
    }
    return \@result;
}

sub invert {
```

<Cpp>

```cpp
#include <iostream>
#include <vector>
#include <cmath>
#include <iomanip>

// Define matrix as a vector of vectors
typedef std::vector<std::vector<double> >
Matrix;

// Function prototypes
Matrix generateRandomMatrix(int rows, int cols);
void printMatrix(const Matrix &mat, const
std::string &name);
Matrix transpose(const Matrix &mat);
Matrix multiply(const Matrix &A, const Matrix &B);
Matrix pseudoinverse(const Matrix &A);

// Main function
int main() {
    // Generate a 10x5 random matrix
    Matrix A = generateRandomMatrix(10, 5);

    // Compute the pseudoinverse
    Matrix A_pinv = pseudoinverse(A);

    // Print the original matrix and its
pseudoinverse
    printMatrix(A, "Matrix A");
    printMatrix(A_pinv, "Pseudoinverse of A
(A_pinv)");

    return 0;
}

// Function to generate a random matrix of
given dimensions
Matrix generateRandomMatrix(int rows, int cols) {
    Matrix mat(rows, std::vector<double>(cols));
    for (int i = 0; i < rows; ++i) {
        for (int j = 0; j < cols; ++j) {
```

<Fortran>

```fortran
program pseudoinverse
    implicit none
    integer, parameter :: m = 10, n = 5
    real(8), dimension(m, n) :: A
    real(8), dimension(n, m) :: A_pinv
    real(8), dimension(n) :: S
    real(8), dimension(n, n) :: U, VT
    integer :: info

    ! Seed the random number generator
    call random_seed()

    ! Generate a random 10x5 matrix
    call generate_random_matrix(A, m, n)
    print *, "Original Matrix A:"
    call print_matrix(A, m, n)

    ! Perform Singular Value Decomposition
(SVD)
    call compute_svd(A, U, S, VT, m, n, info)

    ! Compute the pseudoinverse using the SVD
components
    call compute_pseudoinverse(U, S, VT, A_pinv, m,
n)

    ! Print the pseudoinverse
    print *, "Pseudoinverse of A (A_pinv):"
    call print_matrix(A_pinv, n, m)
end program pseudoinverse

! Subroutine to generate a random matrix
subroutine generate_random_matrix(mat, rows, cols)
    implicit none
    integer, intent(in) :: rows, cols
    real(8), intent(out), dimension(rows, cols) ::
mat
    integer :: i, j

    do i = 1, rows
```

<HTML+MathJS>

```html
<!DOCTYPE html>
<html>
    <head>
        <script src="https://
cdnjs.cloudflare.com/ajax/libs/mathjs/10.6.4/
math.min.js"></script>
        <script>
            // Wait for the page to load
            window.onload = function(){
                let A = math.random([10, 5]);
                let A_pinv = math.pinv(A);


document.getElementById('output').innerHTML =
                "Original Matrix (10x5):\n" +
math.format(A, {precision: 3}) +
                "\n\nPseudoinverse (5x10):\n" +
math.format(A_pinv, {precision: 3});
            }
        </script>
    </head>

    <body>
        <h1>Matrix Pseudoinverse Example</h1>
        <pre id="output"></pre> <!-- Element
to display output -->
    </body>
</html>
```

<Translated by ChatGPT4o>

DEMO EXAMPLES $\mathbf{A}^{+}; \mathbf{A} \in \mathbb{R}^{10\times5}, a_{i,j} \sim \mathcal{U}$

<Perl>

```perl
use strict;
use warnings;
my @A = map { [ map { rand() } 1..5 ] } 1..10;
my $A_pinv = pseudoinverse(\@A);
for my $row (@$A_pinv) {
    print join("\t", @$row), "\n";
}
sub pseudoinverse {
    my ($matrix) = @_;
    my $A_T = transpose($matrix);
    my $A_TA = matrix_multiply($A_T, $matrix);
    my $A_TA_inv = invert($A_TA);
    return matrix_multiply($A_TA_inv, $A_T);
}

sub transpose {
    my ($matrix) = @_;
    my @transposed;
    for my $i (0..$#{$matrix->[0]}) {
        push @transposed, [ map { $_->[$i] } @$matrix ];
    }
    return \@transposed;
}

sub matrix_multiply {
    my ($matrix1, $matrix2) = @_;
    my @result;
    for my $i (0..$#$matrix1) {
        for my $j (0..$#{$matrix2->[0]}) {
            $result[$i][$j] = 0;
            for my $k (0..$#{$matrix2}) {
                $result[$i][$j] += $matrix1->[$i][$k] * $matrix2->[$k][$j];
            }
        }
    }
    return \@result;
}

sub invert {
    my ($matrix) = @_;
    my $size = @$matrix;
    my @identity = map { [(0) x $size] } 0..$size-1;
    for my $i (0..$size-1) {
        $identity[$i][$i] = 1;
    }
    my @augmented = map { [@{$matrix->[$_]}, @{$identity[$_]}] } 0..$size-1;

    # Perform Gaussian elimination
    for my $i (0..$size-1) {
        # Make the diagonal element 1
        my $diag = $augmented[$i][$i];
        for my $j (0..$#{$augmented[0]}) {
            $augmented[$i][$j] /= $diag;
        }
        # Make other column elements 0
        for my $k (0..$size-1) {
            next if $k == $i;
            my $factor = $augmented[$k][$i];
            for my $j (0..$#{$augmented[0]}) {
                $augmented[$k][$j] -= $factor * $augmented[$i][$j];
            }
        }
    }

    # Extract the inverted matrix
    my @inverted = map { [ @$_[$size..$#$_] ] } @augmented;
    return \@inverted;
}
```

<Cpp>

```cpp
#include <iostream>
#include <vector>
#include <cmath>
#include <iomanip>

// Define matrix as a vector of vectors
typedef std::vector<std::vector<double> > Matrix;

// Function prototypes
Matrix generateRandomMatrix(int rows, int cols);
void printMatrix(const Matrix &mat, const std::string &name);
Matrix transpose(const Matrix &mat);
Matrix multiply(const Matrix &A, const Matrix &B);
Matrix pseudoinverse(const Matrix &A);

// Main function
int main() {
    // Generate a 10x5 random matrix
    Matrix A = generateRandomMatrix(10, 5);

    // Compute the pseudoinverse
    Matrix A_pinv = pseudoinverse(A);

    // Print the original matrix and its pseudoinverse
    printMatrix(A, "Matrix A");
    printMatrix(A_pinv, "Pseudoinverse of A (A_pinv)");

    return 0;
}

// Function to generate a random matrix of given dimensions
Matrix generateRandomMatrix(int rows, int cols) {
    Matrix mat(rows, std::vector<double>(cols));
    for (int i = 0; i < rows; ++i) {
        for (int j = 0; j < cols; ++j) {
            mat[i][j] = (double)rand() / RAND_MAX; // Random number between 0 and 1
        }
    }
    return mat;
}

// Function to print a matrix
void printMatrix(const Matrix &mat, const std::string &name) {
    std::cout << name << ":\n";
    for (const auto &row : mat) {
        for (const auto &val : row) {
            std::cout << std::fixed << std::setprecision(4) << val << " ";
        }
        std::cout << "\n";
    }
    std::cout << std::endl;
}

// Function to compute the transpose of a matrix
Matrix transpose(const Matrix &mat) {
    int rows = mat.size();
    int cols = mat[0].size();
    Matrix trans(cols, std::vector<double>(rows));
    for (int i = 0; i < rows; ++i) {
        for (int j = 0; j < cols; ++j) {
            trans[j][i] = mat[i][j];
        }
    }
    return trans;
}

// Function to multiply two matrices
Matrix multiply(const Matrix &A, const Matrix &B) {
    int rows = A.size();
    int cols = B[0].size();
    int common = B.size();
    Matrix result(rows, std::vector<double>(cols, 0.0));
    for (int i = 0; i < rows; ++i) {
        for (int j = 0; j < cols; ++j) {
            for (int k = 0; k < common; ++k) {
                result[i][j] += A[i][k] * B[k][j];
            }
        }
    }
    return result;
}

// Function to compute the pseudoinverse using SVD
Matrix pseudoinverse(const Matrix &A) {
    int m = A.size();
    int n = A[0].size();

    // Transpose of A
    Matrix A_T = transpose(A);

    // Multiplying A_T * A (n x n matrix)
    Matrix A_TA = multiply(A_T, A);

    // Perform SVD on A_TA (we'll approximate here by diagonalizing)
    // In real scenarios, use proper SVD decomposition
    Matrix A_TA_inv(n, std::vector<double>(n, 0.0));
    for (int i = 0; i < n; ++i) {
        A_TA_inv[i][i] = 1.0 / (A_TA[i][i] + 1e-10); // Invert the diagonal (regularization added)
    }

    // Multiply A_TA_inv * A_T to get pseudoinverse
    return multiply(A_TA_inv, A_T);
}
```

<Fortran>

```fortran
program pseudoinverse
    implicit none
    integer, parameter :: m = 10, n = 5
    real(8), dimension(m, n) :: A
    real(8), dimension(n, m) :: A_pinv
    real(8), dimension(n) :: S
    real(8), dimension(n, n) :: U, VT
    integer :: info

    ! Seed the random number generator
    call random_seed()

    ! Generate a random 10x5 matrix
    call generate_random_matrix(A, m, n)
    print *, "Original Matrix A:"
    call print_matrix(A, m, n)

    ! Perform Singular Value Decomposition (SVD)
    call compute_svd(A, U, S, VT, m, n, info)

    ! Compute the pseudoinverse using the SVD components
    call compute_pseudoinverse(U, S, VT, A_pinv, m, n)

    ! Print the pseudoinverse
    print *, "Pseudoinverse of A (A_pinv):"
    call print_matrix(A_pinv, n, m)
end program pseudoinverse

! Subroutine to generate a random matrix
subroutine generate_random_matrix(mat, rows, cols)
    implicit none
    integer, intent(in) :: rows, cols
    real(8), intent(out), dimension(rows, cols) :: mat
    integer :: i, j

    do i = 1, rows
        do j = 1, cols
            call random_number(mat(i, j))  ! Fill with random numbers [0, 1)
        end do
    end do
end subroutine generate_random_matrix

! Subroutine to compute SVD
subroutine compute_svd(A, U, S, VT, m, n, info)
    implicit none
    integer, intent(in) :: m, n
    real(8), intent(in), dimension(m, n) :: A
    real(8), intent(out), dimension(n, n) :: U
    real(8), intent(out), dimension(n) :: S
    real(8), intent(out), dimension(n, n) :: VT
    integer, intent(out) :: info

    ! Workspace variables for LAPACK SVD
    real(8), dimension(:), allocatable :: work
    integer, dimension(:), allocatable :: iwork
    integer :: lwork

    ! Allocate workspace
    lwork = 4 * n
    allocate(work(lwork), iwork(8 * n))

    ! Call LAPACK's DGESVD routine (SVD)
    call dgesvd('S', 'S', m, n, A, m, S, U, n, VT, n, work, lwork, info)

    deallocate(work, iwork)
end subroutine compute_svd

! Subroutine to compute the pseudoinverse using SVD components
subroutine compute_pseudoinverse(U, S, VT, A_pinv, m, n)
    implicit none
    integer, intent(in) :: m, n
    real(8), intent(in), dimension(n, n) :: U, VT
    real(8), intent(in), dimension(n) :: S
    real(8), intent(out), dimension(n, m) :: A_pinv
    real(8), dimension(n, n) :: S_inv
    integer :: i, j

    ! Compute the inverse of the singular values
    S_inv = 0.0
    do i = 1, n
        if (S(i) > 1e-10) then
            S_inv(i, i) = 1.0 / S(i)
        else
            S_inv(i, i) = 0.0
        end if
    end do

    ! Compute A_pinv = VT^T * S_inv * U^T
    A_pinv = matmul(matmul(VT, S_inv), transpose(U))
end subroutine compute_pseudoinverse

! Subroutine to print a matrix
subroutine print_matrix(mat, rows, cols)
    implicit none
    integer, intent(in) :: rows, cols
    real(8), intent(in), dimension(rows, cols) :: mat
    integer :: i, j

    do i = 1, rows
        do j = 1, cols
            write(*, '(F8.4)', advance="no") mat(i, j)
            if (j < cols) write(*, '(A)', advance="no") " "
        end do
        print *
    end do
end subroutine print_matrix
```

<HTML+MathJS>

```html
<!DOCTYPE html>
<html>
    <head>
        <script src="https://cdnjs.cloudflare.com/ajax/libs/mathjs/10.6.4/math.min.js"></script>
        <script>
            // Wait for the page to load
            window.onload = function(){
                let A = math.random([10, 5]);
                let A_pinv = math.pinv(A);

                document.getElementById('output').innerHTML =
                "Original Matrix (10x5):\n" + math.format(A, {precision: 3}) +
                "\n\nPseudoinverse (5x10):\n" + math.format(A_pinv, {precision: 3});
            }
        </script>
    </head>

    <body>
        <h1>Matrix Pseudoinverse Example</h1>
        <pre id="output"></pre> <!-- Element to display output -->
    </body>
</html>
```

Performance benchmarking

https://www.sas.upenn.edu/~jesusfv/Lecture_HPC_5_Scientific_Computing_Languages.pdf

A baby example

- A basic RBC model:

$$\max \mathbb{E}_0 \sum_{t=0}^{\infty} \beta^t \log c_t$$

$$\text{s.t. } c_t + k_{t+1} = e^{z_t} k_t^{\alpha} + (1-\delta)\, k_t, \ \forall\ t > 0$$

$$z_t = \rho z_{t-1} + \sigma \varepsilon_t,\ \varepsilon_t \sim \mathcal{N}(0,1)$$

- If $\delta = 1$, by "guess and verify":

$$c_t = (1 - \alpha\beta)\, e^{z_t} k_t^{\alpha}$$

$$k_{t+1} = \alpha\beta e^{z_t} k_t^{\alpha}$$

- Calibration:

Parameter	β	α	ρ	σ
Value	0.95	1/3	0.95	0.007

Table 1: Average and Relative Run Time (Seconds)

	Mac		
Language	Version/Compiler	Time	Rel. Time
C++	Clang 12.0	1.1	1.00
Fortran	GCC 11.2	1.21	1.10
Java	9	2.20	2.00
Julia	1.6	2.11	1.92
	1.6, fast	1.98	1.60
Matlab	2021a	3.82	3.47
Python	CPython 3.9.7	184.49	167.71
R	4.1.1	44.11	40.10
Mathematica	12.0.0, base	1080.91	982.65
Matlab, Mex	2021a	2.01	1.82
Rcpp	4.1.1	3.30	2.66
Python	Numba	1.99	1.81
	Cython	1.75	1.41
Mathematica	12.0.0, idiomatic	4.85	4.41

Conclusions

Computing Languages for Open Science

- Still Python (GPL) & R (DSL) seem to be good options.
- Non-CS scientists would prefer math-oriented DSLs like R, MATLAB, Julia.
- **Julia** looks very interesting (e.g., UTF-8 for any names: 🤨`(a,b) = a^b`; 파이=pi; 🤨(파이,3); built-in functions for math operations like MATLAB), but it is still also very immature yet.
- MATLAB is so expensive... so use it when you can ! (like Macs and Adobe Creative Suites).
- Transparency may be still achieved using cloud-based platforms like CodeOcean.

Take home message:

"Why switch, when you can be a multilingual?"