EMORY
UNIVERSITY
SCHOOL OF
MEDICINE

Department of
Biomedical Informatics

BMI 500:

Introduction to Biomedical Informatics

Lecture 2. Coding, documentation, security & data management basics

https://tinyurl.com/bmi500

31st Aug 2022

Tony Pan & Gari D. Clifford
Department of Biomedical Informatics, Emory University, Atlanta, GA USA
Department of Biomedical Engineering, Georgia Institute of Technology, Atlanta, GA, USA

Syllabus: Timetable

GILMORE & PITTSBURGH RAILROAD CO., LTD.

W. N. BICHLER, President and Gen. Manager, Armstead, Mont.
F W. SWENEY, Comptroller, St. Paul, Minn.
P. B. LACY, Treasurer,

OFFICES—Armstead, Mont.

	Motor.	Mls.	*February*, 1935.	Mls.	Motor.	Motor.
	†1 25 P M	0	lve... **Armstead**[1]..arr.	100.3	†1245 P M	
	1 55 "	10.9	.....Grant	89.4	12 15 P M	
	2 30 "	20.6	..... Brenner.....	79.7	11 45 A M	
	2 50 "	27.4	..... Donovan	72.9	11 25 "	
	3 30 "	37.4	..... Tunnel.....	62.9	10 45 "	
	3 57 "	44.7	.....Cruik.....	55.6	10 15 "	
	†4 25 P M	55.0	arr.... **Leadore** ...lve.	45.3	†9 30 A M	12 20 P M
		74.5	arr.... **Gilmore** ...lve.	64.8		δ1105 A M
	δ1105 A M	74.5	lve.... **Gilmore** ...arr.	64.8	δ1050 A M	
	†4 25 P M	55.0	lve....**Leadore**....arr.	45.3	†9 30 A M	
	5 25 "	72.7	..... Lemhi	27.6	8 30 "	
	5 50 "	80.7	.....Tendoy.....	19.6	8 05 "	
	6 20 "	91.3	.....Baker.....	9.0	7 30 "	
	†6 50 P M	100.3	arr.....**Salmon**....lve.	0	†7 00 A M	

EXPLANATION OF SIGNS.

† Daily, except Sunday.
δ Tuesday, Thursday and Saturday.
STANDARD—*Mountain time.*

CONNECTION.

[1] With Oregon Short Line R.R.

https://upload.wikimedia.org/wikipedia/commons/3/3b/G-and-P-timetable.jpg

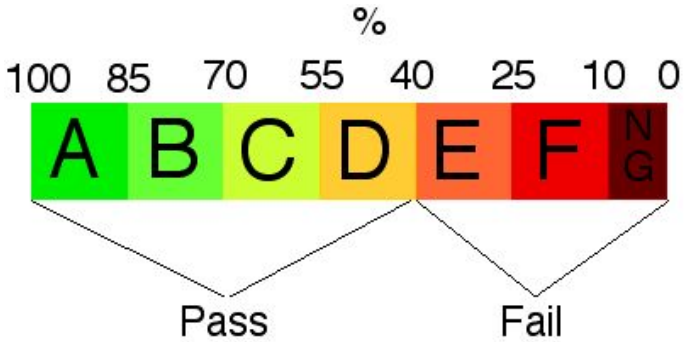
See link here

Week 1, Aug 25	**Orientation - Expectations, Ethics, HIPAA & Communication**	**Instructor: Clifford**
Week 2, Sep 1	**Coding, documentation, security, data wrangling & the cluster**	**Instructor: Clifford**
Week 3, Sept 8	**Python Bootcamp**	**Instructor: Kamaleswaran**
Week 4, Sep 15	**Better Data Treatment**	**Instructor: Reyna**
Week 5, Sep 22	**Natural Language Processing**	**Instructor: Sarker**
Week 6, Sep 29	**Text Representations, Ontologies and Knowledge bases**	**Instructor: Sarker**
Week 7, Oct 6	**Clinical Informatics**	**Instructor: Kamaleswaran**
Week 8, Oct 13	**Ethics, De-identification & HIPAA revisited**	**Instructor: Clifford**
Week 9, Oct 20	**Cancer Informatics**	**Instructor: Bhasin**
Week 10, Oct 27	**Time series analytics - Python vs Matlab?**	**Instructor: Mahmoudi**
Week 11, Oct 30	**Blind Source Separation**	**Instructor: Sameni**
Week 12, Nov 3	**Motion Analysis 1: Clinical 3-D Motion Capture**	**Instructor: McKay**
Week 13, Nov 10	**Motion Analysis 2: Markerless Motion Capture in Neurology**	**Instructor: McKay**
Week 14, Nov 17	**Class wrap-up: Review of review articles**	**Instructor: Clifford**
Week 15, Nov 24	***NO CLASS – Thanksgiving recess -- and end of lectures!***	***Instructor: Food!***
Week 16, Dec 1	**Final Project Deadline**	**Instructor: Clifford**

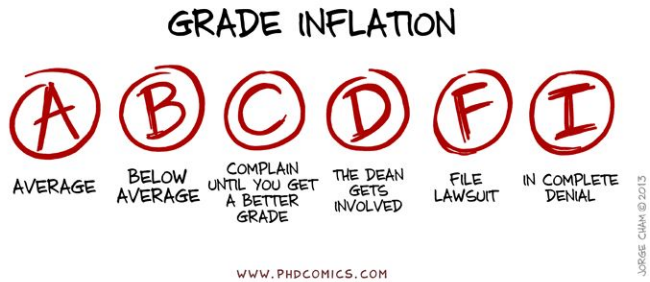
https://tinyurl.com/bmi500

Expectations: grading

- Filled in by the week's instructor and averaged.
- Graded on a curve



https://upload.wikimedia.org/wikipedia/commons/e/e0/Junior_certificate_grading.png



http://www.phdcomics.com/comics/archive/phd012014s.gif

BMI 500 Class Grades Fall 2018

File Edit View Insert Format Data Tools Add-ons Help All changes saved in Drive

100% $ % .0 .00 123 Arial 10 B I S A

=(C2*0.3)+(D2*0.5)+(E2*0.2)

A	B	C	D	E	F
Student Name	Student Email	Punctuality, attendance & class participation /100 (30% of grade)	Project performance /100 (50%)	Presentation and documentation /100 (20%)	Overall Grade
					0

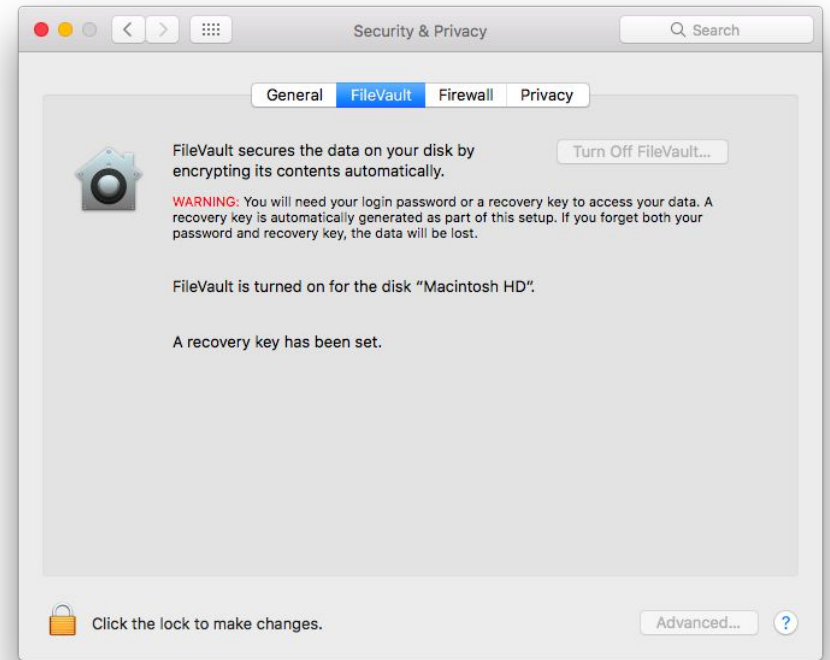
Quick survey ...

1. Has anyone coded with others on a project?
2. What do you use to backup / version code?
3. What do you use to backup data?
4. What do you use to back up documents?
5. Did anyone encrypt their hard disk?
6. What languages do you code in?

Encrypt your hard disk tonight …

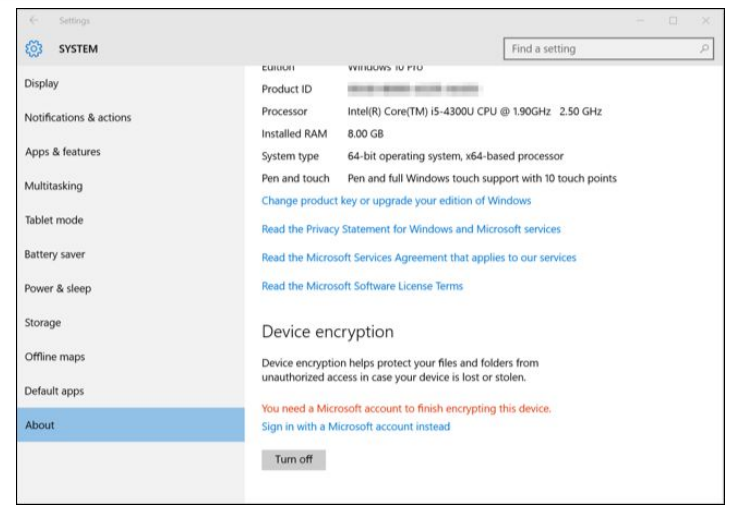
Mac:

1. Click on the Apple menu and select System Preferences.
2. Select Privacy & Security.
3. Click on the FileVault tab, then click the lock in the bottom left corner of the window.
4. Enter your administrator name and password and click Unlock.
5. Click Turn On FileVault.
6. Choose whether you want to link your iCloud account to FileVault to unlock the disk and reset your password or create a recovery key and click Continue.
7. Click Restart to reboot your Mac and begin the encryption process.



Windows 10

1. Locate the hard drive you want to encrypt under "This PC" in Windows Explorer.
2. Right-click the target drive and choose "Turn on BitLocker."
3. Choose "Enter a Password."
4. Enter a secure password.
5. Choose "How to Enable Your Recovery Key" which you'll use to access your drive if you lose your password. You can print it, save it as a file to your hard drive, save it as a file to a USB drive, or save the key to your Microsoft account.
6. Choose "Encrypt Entire Drive." This option is more secure and encrypts files you marked for deletion.
7. Unless you need your drive to be compatible with older Windows machines, choose "New Encryption Mode."
8. Click "Start Encrypting" to begin the encryption process. Note that this will require a computer restart if you're encrypting your boot drive. The encryption will take some time, but it will run in the background, and you'll still be able to use your computer while it runs.



Note: BitLocker is not available on Windows 10 Home edition, but there is a similar feature for device encryption.

Encrypting Linux

1. **Tomb**. Tomb is a free and open source tool for easily **encrypting** and backing up files on GNU/**Linux** systems. ...
2. Cryptmount. Cryptmount is an open source utility created for GNU/**Linux** Operating Systems to enable users to mount **encrypted** files without root privileges. ...
3. CryFS. ...
4. GnuPG. ...
5. VeraCrypt. ...
6. EncFS. ...
7. 7-zip. ...
8. dm-crypt.

If you do download (by mistake or otherwise) - Shred sensitive files

Windows:
Eraser, axcrypt, ….
https://www.techrepublic.com/article/how-to-completely-and-securely-delete-files-in-windows/

Mac:
gshred:
https://superuser.com/questions/617515/using-shred-from-the-command-line
rm -P

Linux:
srm (built in)

Second Tip: Work remotely - don't download

I know it's really easy to just download your data and start hacking away at it, but this causes many problems:

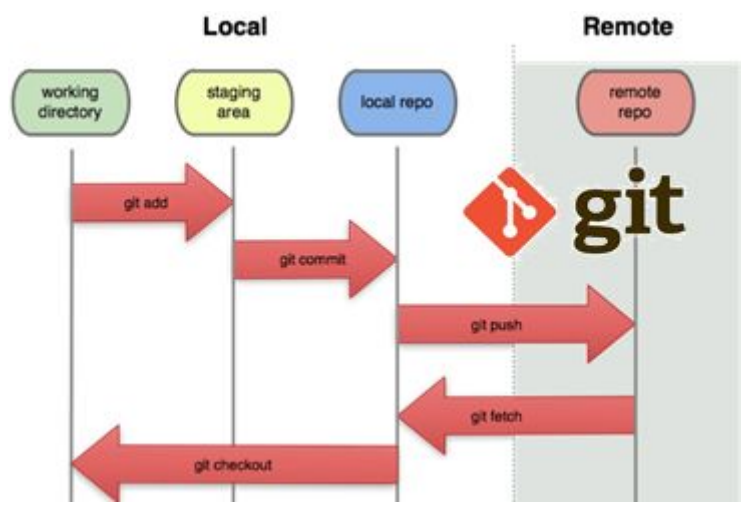
1. What if the data has PHI in it and the IRB said this would never be put on removable media. You just put the whole project at risk of being shut down,
2. Your laptop or desktop is configured in a very specific way. When someone else checks out your code to try to repeat or reuse your work, it has a low probability of actually working. Libraries will be incompatible, environment variables will be different, and some files will just be missing. Even the processor architecture can lead to different answers (e..g because of different random number generators - and no - the seed won't always help)
3. What if you forget to check in the code and your laptop is stolen/dies/you are hit by a bus … all that work is wasted and your supervisor may have to cut her losses.
4. Remote machines are generally more powerful than local machines, and the data and code are backed up more effectively.

Third Tip: Use Source Code Control …

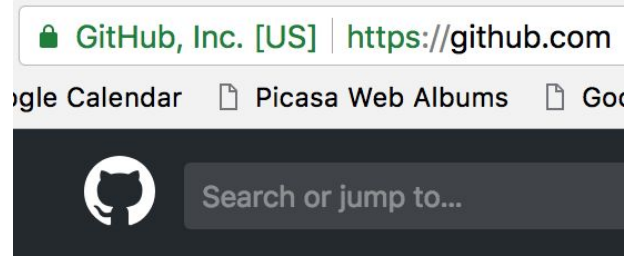
Even if you never collaborate or intend to share your code

When you write code, you generally find, one day, that this week's code doesn't work quite as well as last week's code. How are you going to roll back to that magic version?

Use for labels, annotations and other expert generated information that isn't raw data - you need to version control your labels - they will evolve!



Source Code Control



- http://github.com (or SVN)
- **Do not make it public**
- Use two-factor authentication
- Do not commit data, passwords, or local drive settings
- Ask a group member to check out your code and try to run it. I bet they can't … why?
- Avoid local configurations, paths
- Operate on the source data, not intermediate data

If an organization owner has restricted the ability to change repository visibility to organization owners only, members with admin permissions to a private repository cannot make it public. For more information, see "Repository permission levels for an organization."

Tip: If you're converting your private repository to a public repository as part of a move toward creating an open source project, see the Open Source Guides for helpful tips and guidelines. You can also take a free course on managing an open source project with GitHub Learning Lab. Once your repository is public, you can also view your repository's community profile to see whether your project meets best practices for supporting contributors.

Warning: When a private repository is made public, its private forks are detached and billed separately. For more information, see "What happens to forks when a repository is deleted or changes visibility?"

1. On GitHub, navigate to the main page of the repository.
2. Under your repository name, click ⚙ **Settings**.

3. Click **Make public**.

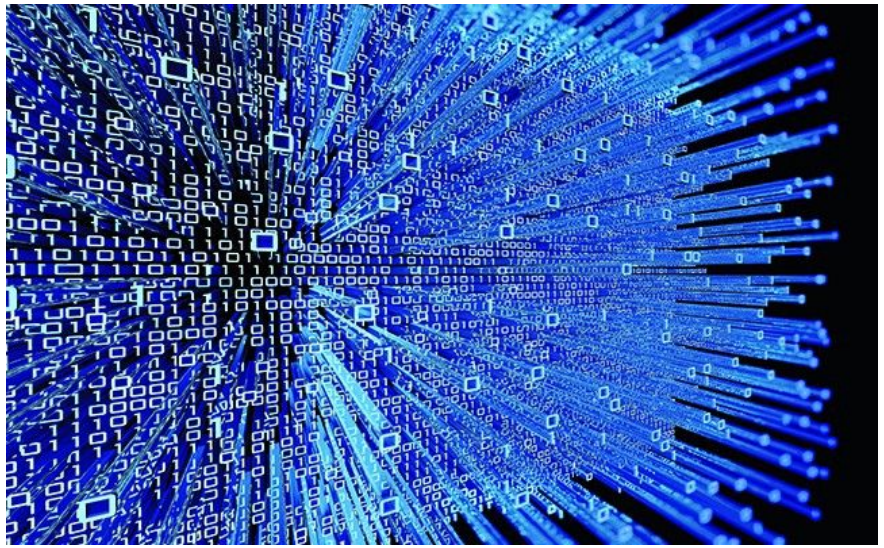
4. Read the warnings.
5. Type the name of the repository that you want to make public.
6. Click **I understand, make this repository public**.

On-the-fly Backups

You will still want to keep dynamic backups of your laptop/desktop in case you forgot to back something up

- **Don't** check in data or docs to Github
- Use GDrive for rough versions of documents
- HIPAA-compliant cloud storage for data that *might* be sensitive (this includes proprietary data as well as PHI)
  - OneDrive
  - aws.emory.edu
- Sync *everything* on your laptop to OneDrive, *including* data: https://it.emory.edu/office365/onedrive.html
- Treat your computer as disposable
- Have one at home and one at the office and **mirror them**
- Encrypt local hard drive
- Encrypt your phone
- Use a VPN on public WiFi or abroad (e.g. NordVPN not just vpn.emory.edu, which only encrypts traffic to Emory)

Managing your data



Ask yourself: What data do I have?

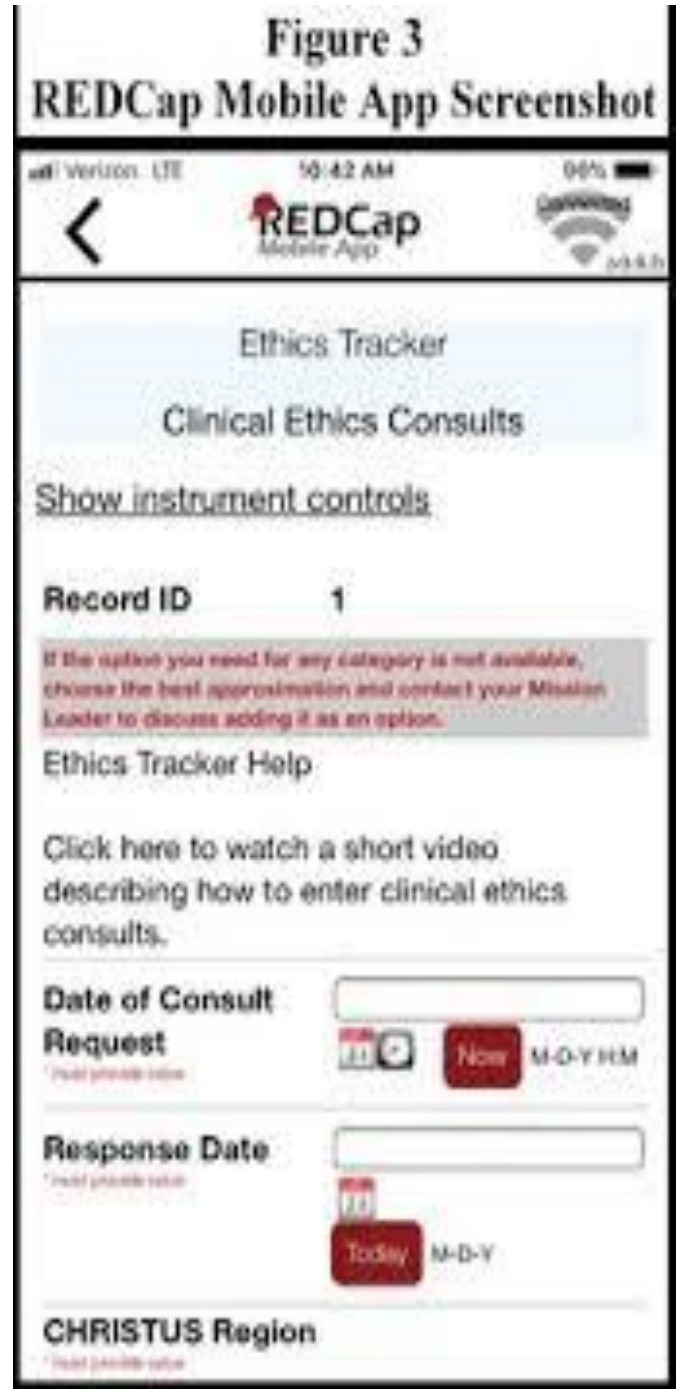
- Research data (raw source data)
- Research data (derived variables)
- Results
- Documentation (lab book, Readme, wiki)
- Publications
- Reporting

Each one will require a different mode of management, but version control is vital!

Research **E**lectronic **D**ata **Cap**ture

- **REDCap** - a secure online Database
- Designed for online questionnaires and data collection forms from patients - nothing else.
- Uses a single table within a MySQL database
- It's good for small projects and simple GUI-driven charts.
- For complex analysis, export all the data and use a scripting language like R or Python.

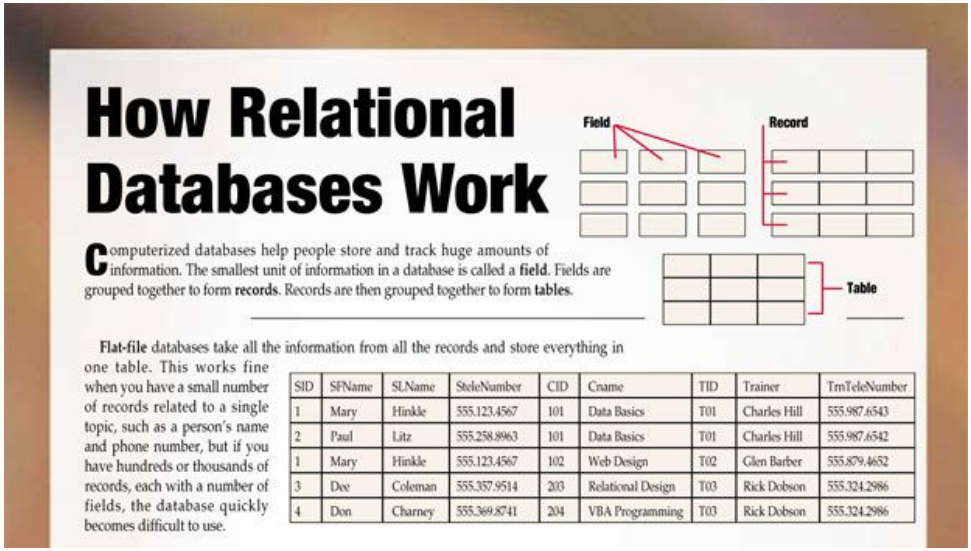
https://chrisbotte.wordpress.com/2014/10/29/major-redcap-disadvantage-explained/



When should I build a relational database?

E.g. Oracle (Cerner), Postgres,

MS Access, Caché (Epic)



How Relational Databases Work

Computerized databases help people store and track huge amounts of information. The smallest unit of information in a database is called a **field**. Fields are grouped together to form **records**. Records are then grouped together to form **tables**.

Flat-file databases take all the information from all the records and store everything in one table. This works fine when you have a small number of records related to a single topic, such as a person's name and phone number, but if you have hundreds or thousands of records, each with a number of fields, the database quickly becomes difficult to use.

SID	SFName	SLName	SteleNumber	CID	Cname	TID	Trainer	TrnTeleNumber
1	Mary	Hinkle	555.123.4567	101	Data Basics	T01	Charles Hill	555.987.6543
2	Paul	Litz	555.258.8963	101	Data Basics	T01	Charles Hill	555.987.6542
1	Mary	Hinkle	555.123.4567	102	Web Design	T02	Glen Barber	555.879.4652
3	Dee	Coleman	555.357.9514	203	Relational Design	T03	Rick Dobson	555.324.2986
4	Don	Charney	555.369.8741	204	VBA Programming	T03	Rick Dobson	555.324.2986

https://www.ibm.com/ibm/history/ibm100/us/en/icons/reldb/

[Six advantages and disadvantages]:

Speed	Cost
Security	Performance
Simplicity	Storage
Accessibility	Complexity
Accuracy	Information Loss
Multi-user	Structure Limitations

NoSQL / non-relational DBs?

Examples:

MongoDB, Mumps (Epic),

Apache Cassandra,

Redis,

Couchbase and

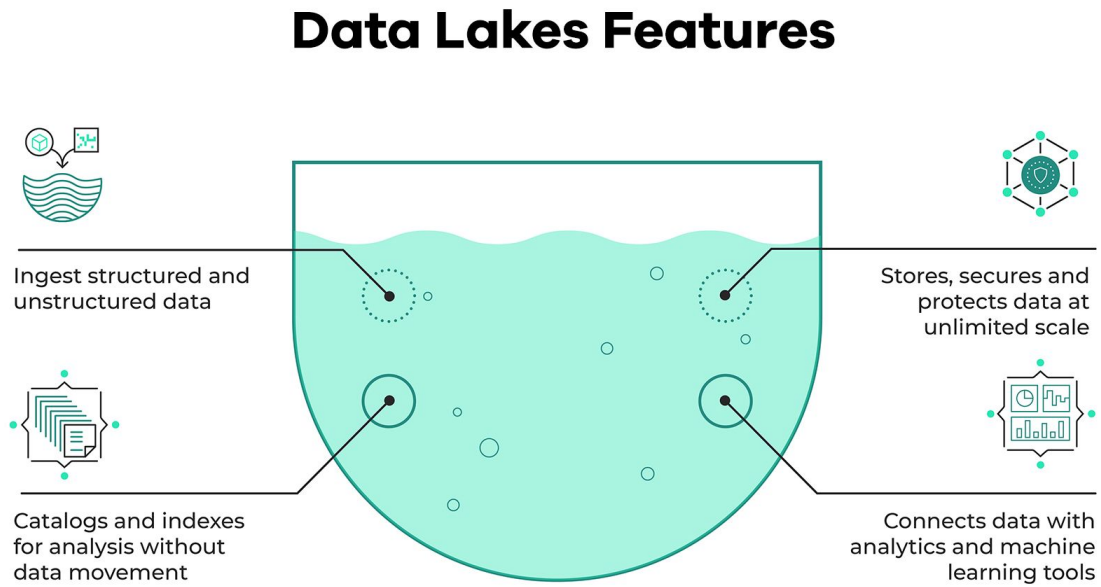
Apache HBase.

They are best for Rapid Application Development.
NoSQL is the best selection for flexible data storage with little to no structure limitations.

Managing raw data

- It depends on the type of data. Imaging, time series, spatial, relational?
- Use standard lossless formats. (DICOM, EDF, ...)
- Don't use bloated text formats like JSON & CSV unless the data are small. They are hard to index and search.
- Don’t just throw it into a relational database or make a ‘data lake’ without thinking about design.
- Don't rely on your employer to back it up. Create a live mirror (e.g. on AWS)

Data lake or data swamp?



Data lakes and data warehouses are both widely used for storing **big data**, but they are not interchangeable terms. A data lake is a vast pool of raw data, the purpose for which is not yet defined. A data warehouse is a repository for structured, filtered data that has already been processed for a specific purpose.

Data Lake example: MIMIC III on AWS

Data Swamp example: Your laptop. See:

https://lakefs.io/data-lakes/#How_to_Avoid_a_Data_Swamp

Data Lakehouse: Hybrid Data lake/warehouse:

https://databricks.com/blog/2020/01/30/what-is-a-data-lakehouse.html

So how do I structure a data lake?

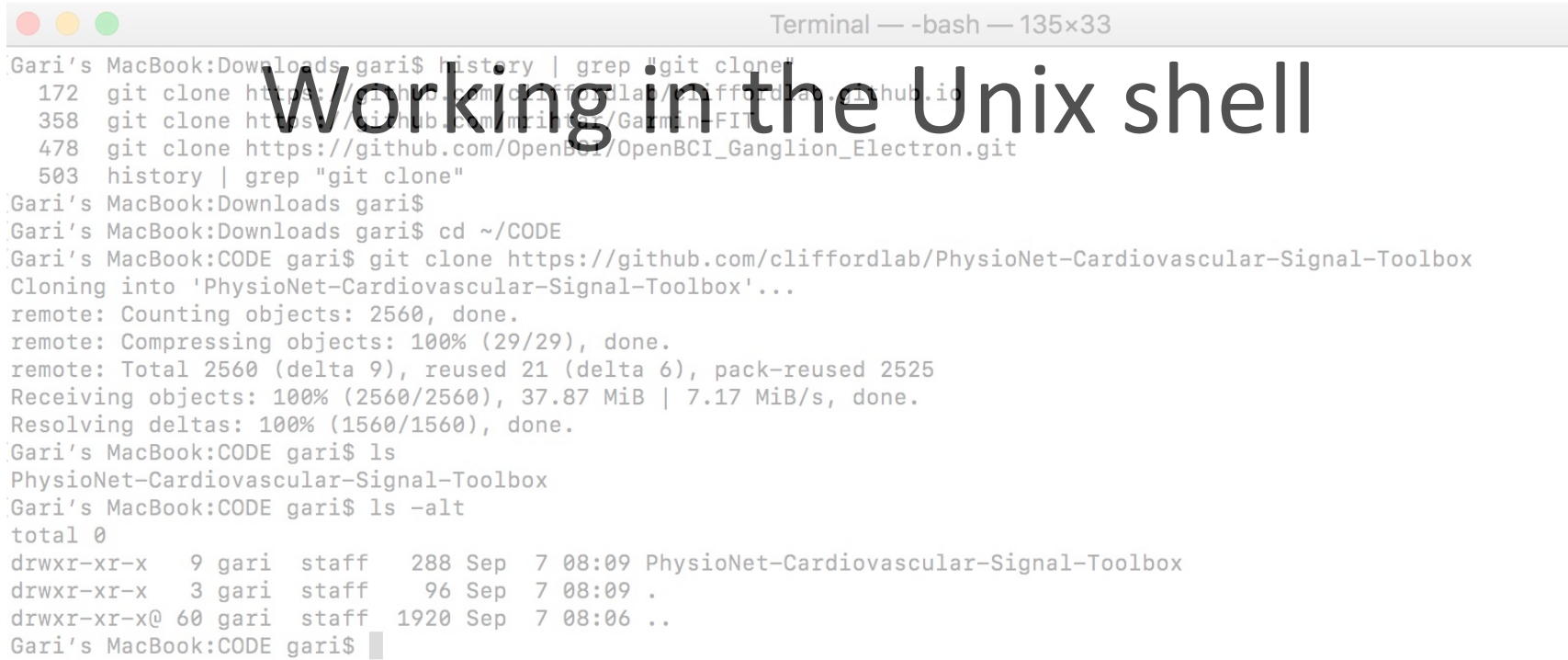
- Choose naming conventions and folder structures to reflect the project.
- All file systems are indexed, so think carefully.
- /data/project/cohortID/patient_xxx/year/month/week/...
- With > 5-10,000 files in a folder you'll see a performance hit, even with ls, so think about how deep you should go.
- Build a database to tie all the databases together.
- Build a map of all the component databases.
- Document it!

Learn a few command line tools to deal with data lakes (like your laptop)

locate	sed
find	awk
grep	|
cut	>
head	<
tail	$
more	

```
[Gari's MacBook:Downloads gari$ locate garid | grep 'misc' > ans.txt
[Gari's MacBook:Downloads gari$ more ans.txt
/usr/local/texlive/2018/texmf-dist/fonts/misc/xetex/fontmapping/polyglossia/deva
nagaridigits.map
/usr/local/texlive/2018/texmf-dist/fonts/misc/xetex/fontmapping/polyglossia/deva
nagaridigits.tec
```

Working in the Unix shell



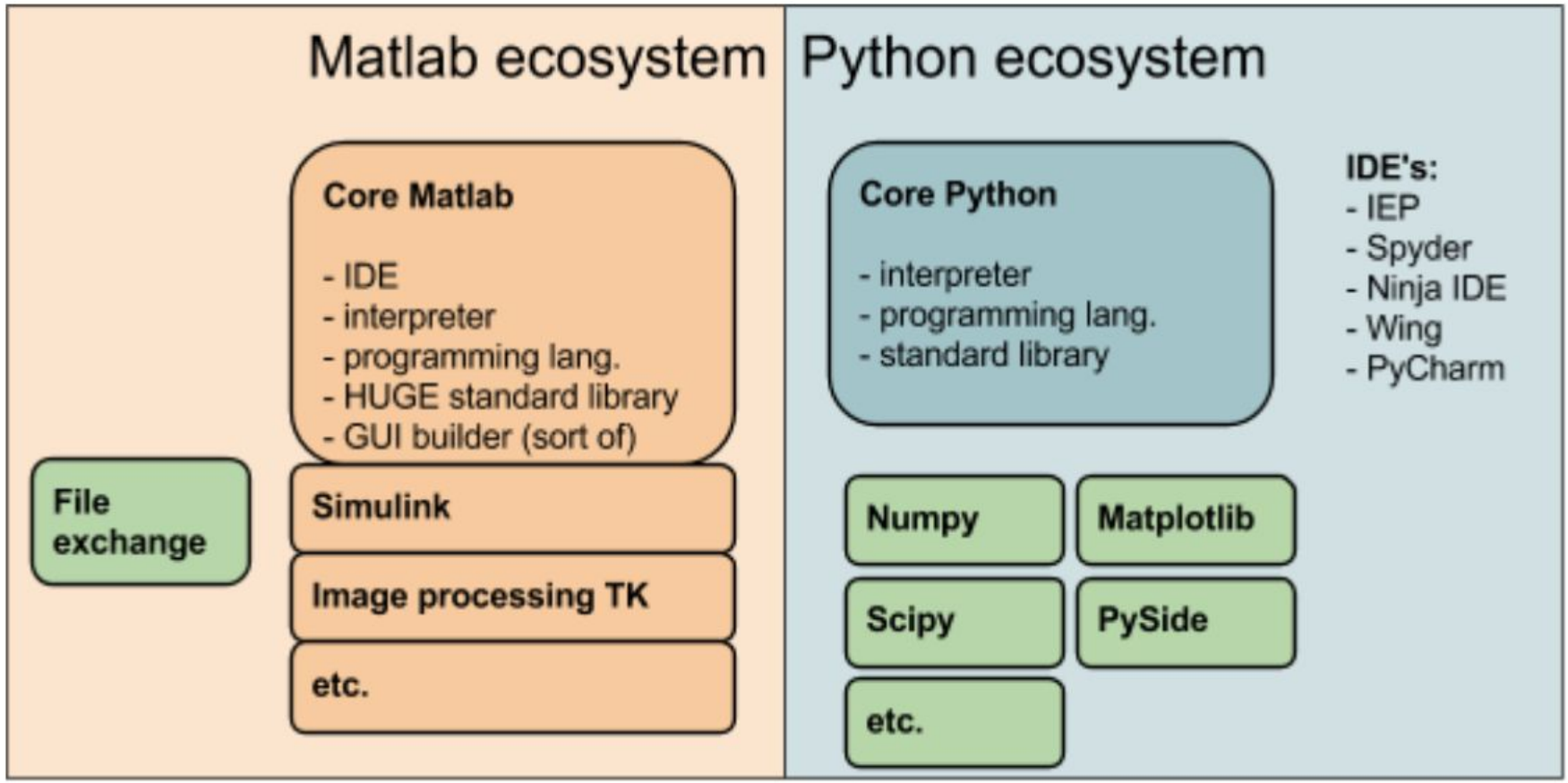
The Unix Shell: Summary of Basic Commands:

https://swcarpentry.github.io/shell-novice/reference/

Opening a terminal

- How to Use Terminal on a Mac
- Git for Windows
- How to Install Bash shell command-line tool on Windows 10
- Install and Use the Linux Bash Shell on Windows 10
- Using the Windows 10 Bash Shell
- Using a UNIX/Linux emulator (Cygwin) or Secure Shell (SSH) client (Putty)

Which coding language? Python, R, Matlab, C?



Ozgur *et al.* MatLab vs. Python vs. R Journal of Data Science 15(2017), 355-372

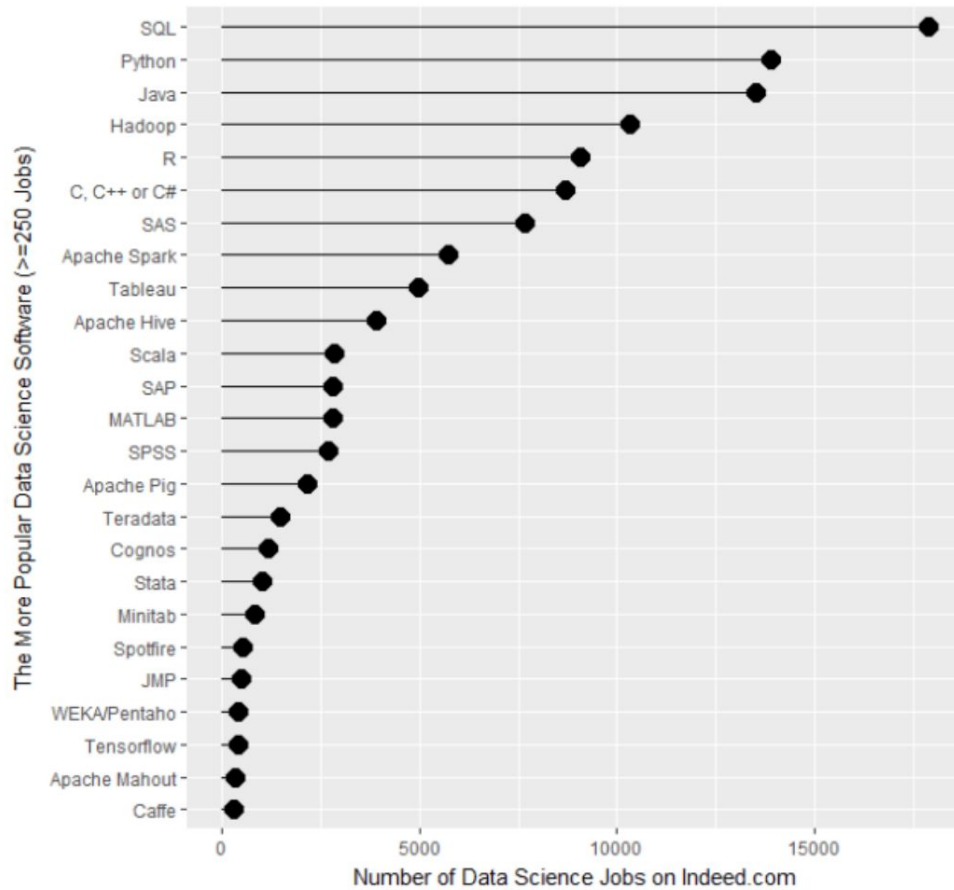
Which data science coding language? Python, R, Matlab or C?

- Matlab -> Use for time series analysis. Python is not mature and bug free in this respect.
- **R** -> Use for statistics. It's probably the most powerful in this domain, but the syntax is awkward
- **Python** -> Use for machine learning, especially scripting on cluster
- **C** -> Use for *really* large jobs, where speed and efficiency is needed
- **Julia** -> Smaller but growing community. Generally faster and easier to use, but less flexible and less support.

(**Python** can be made faster through external libraries, third-party JIT compilers (PyPy), and optimizations with tools like Cython, but **Julia** is designed to be faster - closer to C.)

Which coding language for jobs?

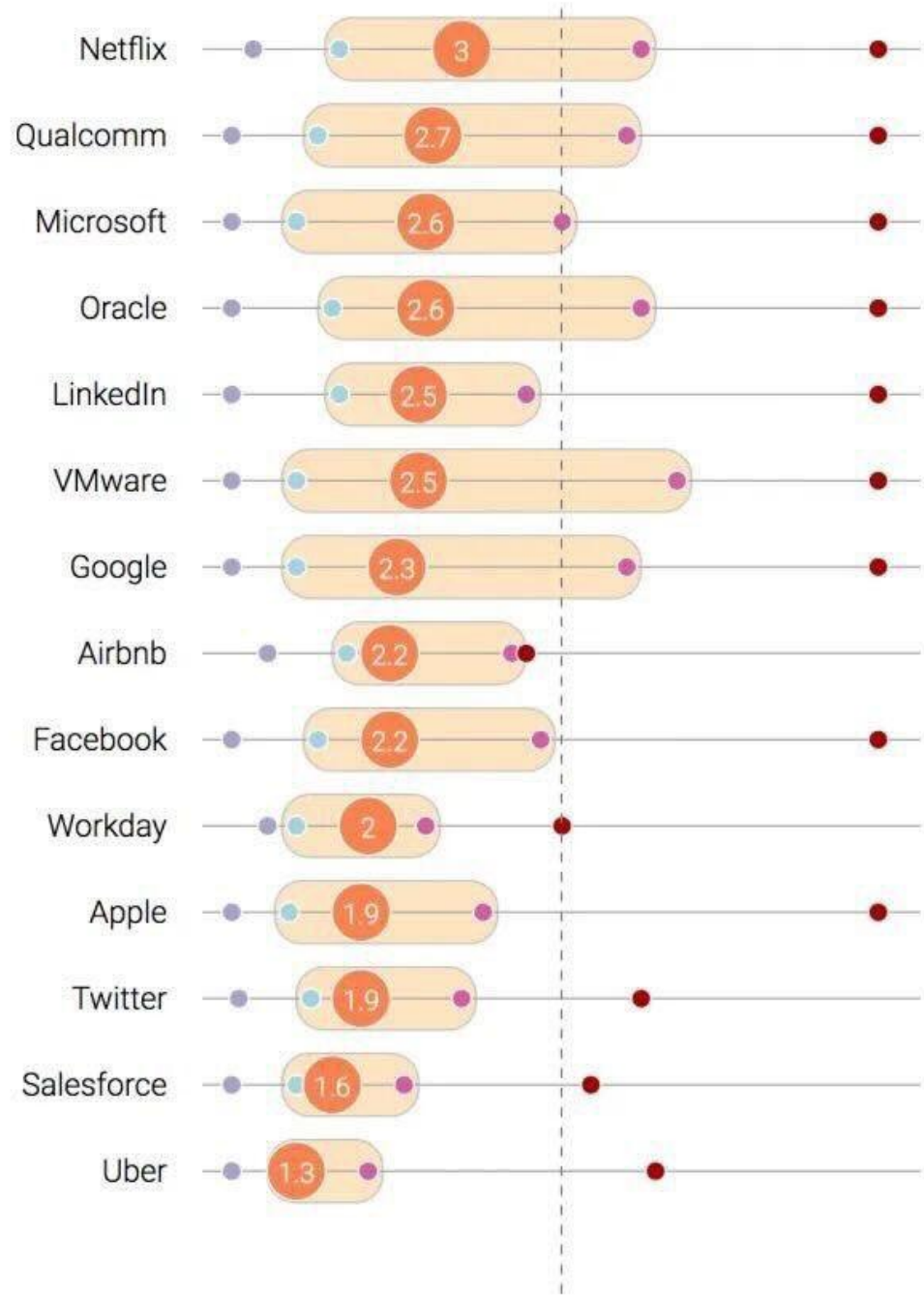
R expanded from being required in 7% of the jobs posted in 2014 to 11% in 2017. While a four-percent shift may not seem as though it is worth noting, the fact that this was accomplished within just three years suggests that this is a trend which not only may continue, but may do so at a highly accelerated rate.



The changing landscape of the job postings available for different types of data analytics software. (B. Muenchen, Data Science Job Report 2017: R Passes SAS, But Python Leaves Them Both Behind, March 1 2017, Online:https://www.r-bloggers.com/data-science-job-report-2017-r-passes-sas-but-python-leaves-them-both-behind/ Accessed Aug 25 2020.)

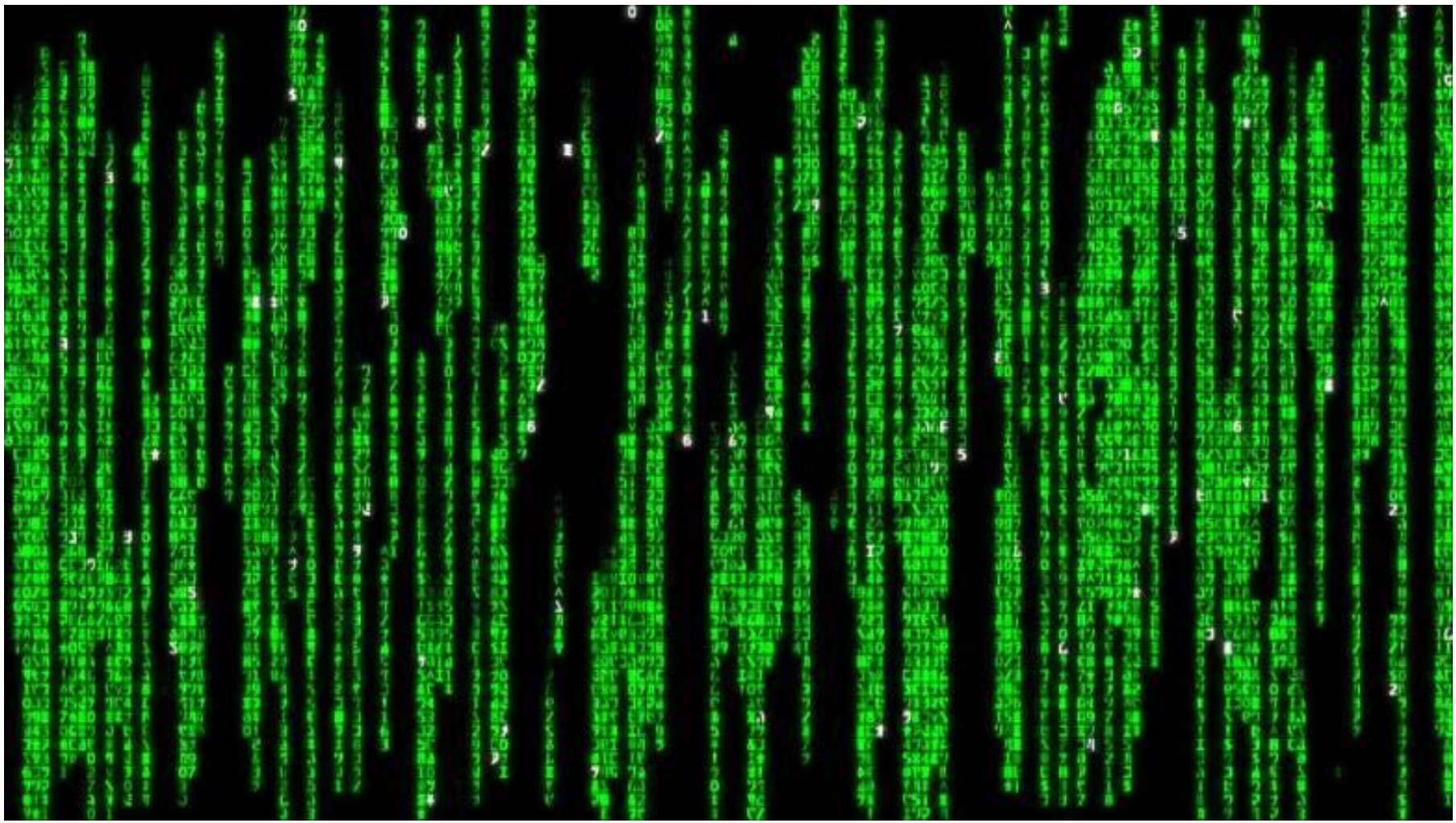
Aside:

Average engineer's tenure at major tech companies is too short



https://twitter.com/typesfast/status/1554098128768315392?t=GOaJqM8pniVIoOeMQyQIhQ&s=03

More importantly, both for jobs and research repeatability ...



Naming, commenting & laying out your code

```
/* Record the argument list in a NOTE annotation. */
for (i = j = 0, p = auxbuf+1; i < argc; i++) {
    int len;
    j += len = strlen(argv[i]);
    if (j < 255) {
        strncpy(p, argv[i], len);
        p += len;
        *p++ = ' ';
        j++;
    }
    else {
        j -= len;
        break;
    }
}
*(--p) = '\0';
auxbuf[0] = j;
annot.subtyp = annot.chan = annot.num = 0;
annot.aux = auxbuf;
annot.anntyp = NOTE;
annot.time = (WFDB_Time)0;
putann(0, &annot);
```

Commenting

- Commenting is vital for not only others to understand your code (i.e. supervisor), but for yourself later

> “Commenting your code is like cleaning your bathroom—you never want to do it, but it really does create a more pleasant experience for you and your guests.”

- Commenting later will never lead to this situation - you should comment your code *as* you write it, not *after*.

Compare:

```
n=1;
while n<=length(lbl)
    TF=strcmp(rawlbl,lbl(n));
    if isempty(TF)
        error('Data parameter not found in raw.');
    end
    data(:,n)=cell2mat(raw(sind:end,TF));
    mv_temp=isnan(data(:,n));
    if sum(mv_temp)>0
        mv(:,n)=mv_temp;
        if mvflag==1
            nv_temp=nv{2,strcmp(nv(1,:),lbl(n))};
            data(mv_temp,n)=nv_temp;
        elseif mvflag==2
            if meanflag==1
                data(mv_temp,n)=meandata(n);
            elseif meanflag==0
                meandata(n)=nanmean(data(:,n));
                data(mv_temp,n)=meandata(n);
            end
        else
        end
    end
    n=n+1;
end
```

Compare:

```
n=1;
while n<=length(lbl)
    % find location of input parameter in data header label
    TF=strcmp(rawlbl,lbl(n));
    if isempty(TF)
        error('Data parameter not found in raw.');
    end
    % input the data from raw into corresponding column in data
    data(:,n)=cell2mat(raw(sind:end,TF));
    mv_temp=isnan(data(:,n));
    if sum(mv_temp)>0 % missing values are present
        mv(:,n)=mv_temp; % store missing value indices for output
        % replace missing values with...
        if mvflag==1 % predefined 'normal' values
            nv_temp=nv{2,strcmp(nv(1,:),lbl(n))}; % get the normal value
            data(mv_temp,n)=nv_temp; % insert it into the data matrix
        elseif mvflag==2 % mean values
            % check if mean data has been input
            if meanflag==1 % use input mean data
                data(mv_temp,n)=meandata(n);
            elseif meanflag==0 % use mean of data provided
                meandata(n)=nanmean(data(:,n));
                data(mv_temp,n)=meandata(n);
            end
        else % nothing! will delete them later
        end
    end
    n=n+1;
end
```

Commenting new files

TODO:

1. ... add syntax to the header comment
2. ... add examples to the header comment
3. ... describe the inputs
4. ... describe the outputs
5. ... copyright and licensing
6. ... don't forget SI units where needed!

```
* SVN revision information:
* @version $Revision: 82 $:
* @author $Author: smoot $:
* @date $Date: 2010-07-10 08:36:02 +0200
      (Sat, 10 Jul 2010) $
```

Despite all the marketing buzz related to Git, such notable open source projects as FreeBSDand LLVM continue to use Subversion as the main version control system. About 47% of other open source projects use Subversion too (while only 38% are on Git). The numbers are much better for companies, because Subversion is de facto standard enterprise version control system. Moreover, every month a number of companies **migrate to Subversion** from such version control systems as ClearCase and TFS.

https://svnvsgit.com/

Linting!

Lint, or a **linter**, is a static code analysis tool used to flag programming errors, bugs, stylistic errors and suspicious constructs.

The term originates from a Unix utility that examined C language source code

towardsdatascience.com/using-pylint-to-write-clean-python-code-660eff40ed8

Sign

Pylint: Star your Python Code!

A code linter is basically a program that inspects your code and gives feedback. A linter can tell you the issues in your program and also, a way to resolve them. You can run it anytime to ensure that your code is matching standard quality.

Linters look at aspects of code and detect lints:

1. Logical Lint: tells about code errors, dangerous code patterns
2. Statistical Lint: looks at formatting issues

There are many Python linters like Flake8, Pylint, Pylama, and many others. In this article, I will be talking about Pylint because it handles both logical and statistical lint.

https://towardsdatascience.com/using-pylint-to-write-clean-python-code-660eff40ed8

Add authors and license asap.

- You can change the license later (it's your code)
- If it's someone else's, and the license doesn't automatically get inherited, ask them what license they want to use and keep some evidence of that.
- If it's not your code, note from where you took the code and add comments where you changed it.
- Mark version numbers in the code.

License choice

FOSS stands for "Free and Open Source Software". There is no one universally agreed-upon definition of FOSS and various groups maintain approved lists of licenses.

The Open Source Initiative (OSI) is one such organization keeping a list of open-source licenses.

Contains:

- **Linking** - linking of the licensed code with code licensed under a different license (e.g. when the code is provided as a library)
- **Distribution** - distribution of the code to third parties
- **Modification** - modification of the code by a licensee
- **Patent grant** - protection of licensees from patent claims made by code contributors regarding their contribution, and protection of contributors from patent claims made by licensees
- **Private use** - whether modification to the code must be shared with the community or may be used privately (e.g. internal use by a corporation)
- **Sublicensing** - whether modified code may be licensed under a different license (for example a copyright) or must retain the same license under which it was provided
- **Trademark grant** - use of trademarks associated with the licensed code or its contributors by a licensee

License choice

License	Author	Latest version	Publication date	Linking	Distribution	Modification	Patent grant	Private use	Sublicensing	TM grant
Academic Free License[10]	Lawrence E. Rosen	3.0	2002	Permissive	Permissive	Permissive	Yes	Yes	Permissive	No
Affero General Public License	Affero Inc	2.0	2007	Copylefted[11]	Copyleft except for the GNU AGPL[11]	Copyleft[11]	?	Yes[11]	?	?
Apache License	Apache Software Foundation	2.0	2004	Permissive[12]	Permissive[12]	Permissive[12]	Yes[12]	Yes[12]	Permissive[12]	No[12]
Apple Public Source License	Apple Computer	2.0	August 6, 2003	Permissive	?	Limited	?	?	?	?
Artistic License	Larry Wall	2.0	2000	With restrictions	With restrictions	With restrictions	No	Permissive	With restrictions	No
Beerware	Poul-Henning Kamp	42	1987	Permissive	Permissive	Permissive	No	Permissive	Permissive	No
BSD License	Regents of the University of California	3.0	?	Permissive[13]	Permissive[13]	Permissive[13]	Manually[13]	Yes[13]	Permissive[13]	Manually[13]
Boost Software License	?	1.0	August 17, 2003	Permissive	?	Permissive	?	?	?	?
Creative Commons Zero	Creative Commons	1.0	2009	Public Domain[14][15]	Public Domain	Public Domain	No	Public Domain	Public Domain	No
CC-BY	Creative Commons	4.0	2002	Permissive[16]	Permissive	Permissive	No	Yes	Permissive	?
CC-BY-SA	Creative Commons	4.0	2002	Copylefted[16]	Copylefted	Copylefted	No	Yes	No	?
CeCILL	CEA / CNRS / INRIA	2.1	June 21, 2013	Permissive	Permissive	Permissive	No	Permissive	With restrictions	No
Common Development and Distribution License	Sun Microsystems	1.0	December 1, 2004	Permissive	?	Limited	?	?	?	?
Common Public License	IBM	1.0	May 2001	Permissive	?	Copylefted	?	?	?	?
Cryptix General License	Cryptix Foundation	N/A	1995	Permissive	Permissive	Permissive	Manually	Yes	?	Manually
Eclipse Public License	Eclipse Foundation	1.0	February 2004	Limited[17]	Limited[17]	Limited[17]	Yes[17]	Yes[17]	Limited[17]	Manually[17]
Educational Community License	Indiana University[18]	1.0	2007	Permissive	?	Permissive	?	?	?	?
Eiffel Forum License	NICE	2	2002	?	?	?	?	?	?	?
European Union Public Licence	European Commission	1.2	May 2017	Copylefted, with an explicit compatibility list[19]	Copylefted, with an explicit compatibility list[19]	Copylefted, with an explicit compatibility list[19]	Yes[20]	Yes[20]	Copylefted, with an explicit compatibility list[19]	No[20]
freebsd	freebsd	1.0	2018	?	?	?	?	?	?	?
GNU Affero General Public License	Free Software Foundation	3.0	2007	GNU GPLv3 only[21]	Copylefted[22]	Copylefted[22]	Yes[23]	Copylefted[23]	Copylefted[22]	Yes[23]
GNU General Public License	Free Software Foundation	3.0	June 2007	GPLv3 compatible only[24][25]	Copylefted[22]	Copylefted[22]	Yes[26]	Yes[26]	Copylefted[22]	Yes[26]
GNU Lesser General Public License	Free Software Foundation	3.0	June 2007	With restrictions[27]	Copylefted[22]	Copylefted[22]	Yes[28]	Yes	Copylefted[22]	Yes[28]

https://en.wikipedia.org/wiki/Comparison_of_free_and_open-source_software_licenses

License choice

License	Author	Latest version	Publication date	Linking	Distribution	Modification	Patent grant	Private use	Sublicensing	TM grant
IBM Public License	IBM	1.0	August 1999	Copylefted	?	Copylefted	?	?	?	?
Intel Open Source License	Intel Corporation	N/A	?	?	?	?	?	?	?	?
ISC license	Internet Systems Consortium	N/A	June 2003	Permissive	Permissive	Permissive	?	?	?	?
LaTeX Project Public License	LaTeX project	1.3c	?	Permissive	?	Permissive	?	?	?	?
Microsoft Public License	Microsoft	N/A	?	Permissive	Permissive	Permissive	No	Permissive	?	No
MIT license / X11 license	MIT	N/A	1988	Permissive[29]	Permissive[29]	Permissive[29]	Manually[29]	Yes[29]	Permissive[29]	Manually[29]
Mozilla Public License	Mozilla Foundation	2.0	January 3, 2012	Permissive[30]	Copylefted[30]	Copylefted[30]	Yes[30]	Yes[30]	Copylefted[30]	No[30]
Netscape Public License	Netscape	1.1	?	Limited	?	Limited	?	?	?	?
Open Software License[10]	Lawrence Rosen	3.0	2005	Permissive	Copylefted	Copylefted	Yes	Yes	Copylefted	?
OpenSSL license	OpenSSL Project	N/A	?	Permissive	?	Permissive	?	?	?	?
PHP License	PHP Group	3.01	?	?	?	?	?	?	?	?
Python Software Foundation License	Python Software Foundation	2	?	Permissive	?	Permissive	?	?	?	?
Q Public License	Trolltech	?	?	Limited	?	Limited	?	?	?	?
RealNetworks Public Source License	RealNetworks	?	?	?	?	?	?	?	?	?
Reciprocal Public License	Scott Shattuck	1.5	2007	?	?	?	?	?	?	?
Sleepycat License	Sleepycat Software	N/A	1996	Permissive	With restrictions	Permissive	No	Yes	No	No
Sun Industry Standards Source License	Sun Microsystems	?	?	?	?	?	?	?	?	?
Sun Public License	Sun Microsystems	?	?	?	?	?	?	?	?	?
Sybase Open Watcom Public License	Open Watcom	N/A	2003-01-28	?	?	?	?	?	?	?
Unlicense	unlicense.org	1	December 2010	Permissive/Public domain	Permissive/Public domain	Permissive/Public domain	?	Permissive/Public domain	Permissive/Public domain	?
W3C Software Notice and License	W3C	20021231	December 31, 2002	Permissive	?	Permissive	?	?	?	?
Do What The Fuck You Want To Public License (WTFPL)	Banlu Kemiyatorn, Sam Hocevar	2	December 2004	Permissive/Public domain	Permissive/Public domain	Permissive/Public domain	No	Yes	Yes	No
XCore Open Source License also separate "Hardware License Agreement"	XMOS	?	February 2011	Permissive	Permissive	Permissive	Manually	Yes	Permissive	?
XFree86 1.1 License	The XFree86 Project, Inc	?	?	Permissive	?	Permissive	?	?	?	?
zlib/libpng license	Jean-Loup Gailly and Mark Adler	?	?	Permissive	?	Permissive	?	?	?	?
Zope Public License	Zope Foundation	2.1	?	?	?	?	?	?	?	?

https://en.wikipedia.org/wiki/Comparison_of_free_and_open-source_software_licenses

Copyright, IP and licensing

Related concepts:

Intellectual Property: A work or invention that is the result of creativity, such as a manuscript or a design, to which one has rights and for which one may apply for a *patent*, *copyright*, trademark, etc. A Trade Secret is also a form of IP.

Copyright: A type of intellectual property that protects original works of authorship as soon as an author fixes the work in a tangible form of expression. In copyright law, there are a lot of different types of works, including paintings, photographs, illustrations, musical compositions, sound recordings, computer programs, books, poems, blog posts, movies, architectural works, plays, etc.

Patent: An exclusive (time-limited) right granted for an invention, which is a product or a process that provides, in general, a new way of doing something, or offers a new technical solution to a problem. To get a patent, technical information about the invention must be disclosed to the public in a patent application. It is a reward for public disclosure so others may use your work freely at a later date (usually about two decades later).

License: The rules that govern the use of IP; an agreement that lets someone else commercially make, use, and sell your invention for a specified period. The owner of the invention (patent) is the 'licensor,' and the person who is receiving the license is the 'licensee.

When you publish you may sign over your copyright, or just grant a limited license to the publisher. If you don't think about this carefully, you may lose the right to use your own work.

See here for detailed information on how to ensure you don't accidentally shoot yourself in the foot by signing over your copyright to a publisher.

https://en.wikipedia.org/wiki/Comparison_of_free_and_open-source_software_licenses

Copyright, IP and licensing

	Copyright	Trademark	Patent	Trade Secret
Protects:	Creative works	Brand identifiers	Inventions	Confidential business information
Protection Begins:	Upon fixation	Upon use in commerce	Upon registration	Upon creation
Length of Protection:	Life + 70 years (WFH: 95/120 years)	For as long as in use	20 years (14 for designs)	For as long as kept secret
Registration:	Gives extra protection	Gives extra protection	Required	None
Example:	Books, paintings, recorded music	"Coca-Cola" 	Incandescent light bulb, Coca-Cola bottle shape	KFC's 11 herbs and spices

Fair Use

You may be able to use limited amounts of material for certain purposes under “fair use”, which can be complicated and is evaluated on a case-by-case basis. Emory provides a helpful explanation of fair use:

> A use is more likely to be a fair use when it is used to advance an argument or is essential to comment and criticism, and when no more than what is needed to make the argument is used. A use is less likely t be a fair use when it is illustrative or decorative, or when large portions of the work are used.

The Wikipedia article on fair use provides more guidance:
https://en.wikipedia.org/wiki/Fair_use

In general, you do not need to seek permission to use a figure/image from an article if you meet one or more of the following criteria:

1. You own the copyright to the material.
2. The material is in the public domain.
3. The material was published under a Creative Commons or another public copyright license.
4. The publishers do not require permission for your use of the material (see below).
5. Your use of the material is considered **“fair use”** under U.S. copyright law

Several publishers have decided that you do not need to seek explicit permission from them for using small portions of materials for academic purposes when they own the copyright to those materials. The following publishers allow a maximum of two figures or tables from a journal article or five figures or tables from a journal volume, and they allow a single text extract of less than 100 words or multiple text extracts totaling less than 300 words per quotation. See here.

If the copyright for the material is not held by one of the publishers linked above, or you are using more material than allowed, then you do not meet the criteria for waiving explicit permission.

General Rules on Commenting

1. Self-commenting
All files should have pretty explicitly file names. e.g. "run.m" is terrible, but "runRegressionModel.m" is better.

2. Summary
Always have a summary, especially with Matlab, since help is so useful.

3. Syntax
Always add the syntax to that header comment, so when you type help <function> you can easily see the syntax

4. Examples
This will make your code 100x easier for other people to use/understand

5. Descriptions
Very useful. See any MATLAB file with liberal use of letters for variables combined with a comment famine

6. Delineation
Using %%%%%%% or whatever to split up the code into chunks makes it much easier to logically follow.

7. SI units
Avoid logical/magnitude errors.

8. In-line comments
Makes everything easier to understand

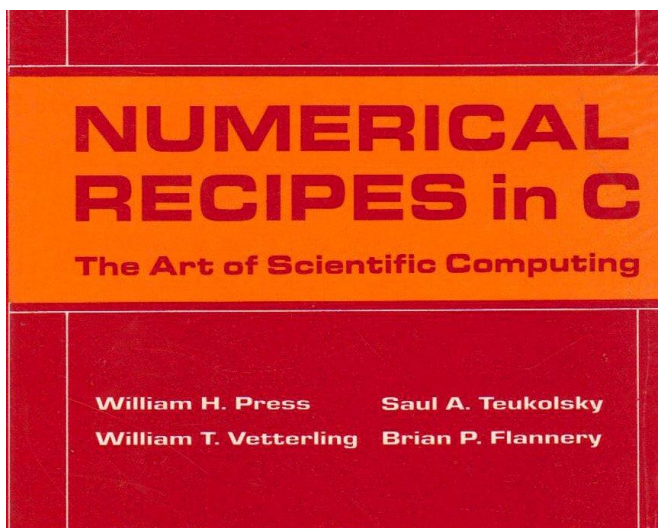
9. Update
Countless times I see comments in my code which don't make sense, since I know I have updated the code to do something else. Others won't have a clue. Update comments!

10. Never stop commenting

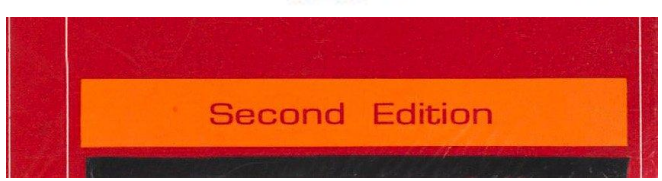
E.g.: Numerical Recipes in C

Ch 14.1: Moments of a distribution

$$\upsilon_n = E\{(x-\mu_x)^n\} = \int_{-\infty}^{+\infty} (x-\mu_x)^n p_x(x)\mathrm{d}x$$



$$\hat{\kappa}(x) = \frac{1}{M}\sum_{i=1}^{M}\left[\frac{x_i - \hat{\mu}_x}{\hat{\sigma}}\right]^n$$



Many textbooks use the binomial theorem to expand out the definitions into sums of various powers of the data, e.g., the familiar

$$\mathrm{Var}(x_1 \ldots x_N) = \frac{1}{N-1}\left[\left(\sum_{j=1}^{N} x_j^2\right) - N\overline{x}^2\right] \approx \overline{x^2} - \overline{x}^2 \qquad (14.1.7)$$

but this can magnify the roundoff error by a large factor and is generally unjustifiable in terms of computing speed. A clever way to minimize roundoff error, especially for large samples, is to use the *corrected two-pass algorithm* [1]: First calculate $\overline{x}$, then calculate $\mathrm{Var}(x_1 \ldots x_N)$ by

$$\mathrm{Var}(x_1 \ldots x_N) = \frac{1}{N-1}\left\{\sum_{j=1}^{N}(x_j - \overline{x})^2 - \frac{1}{N}\left[\sum_{j=1}^{N}(x_j - \overline{x})\right]^2\right\} \qquad (14.1.8)$$

The second sum would be zero if $\overline{x}$ were exact, but otherwise it does a good job of correcting the roundoff error in the first term.

```
#include <math.h>

void moment(float data[], int n, float *ave, float *adev, float *sdev,
    float *var, float *skew, float *curt)
Given an array of data[1..n], this routine returns its mean ave, average deviation adev,
standard deviation sdev, variance var, skewness skew, and kurtosis curt.
{
    void nrerror(char error_text[]);
    int j;
    float ep=0.0,s,p;

    if (n <= 1) nrerror("n must be at least 2 in moment");
    s=0.0;                                          First pass to get the mean.
    for (j=1;j<=n;j++) s += data[j];
    *ave=s/n;
    *adev=(*var)=(*skew)=(*curt)=0.0;               Second pass to get the first (absolute), sec-
    for (j=1;j<=n;j++) {                                ond, third, and fourth moments of the
        *adev += fabs(s=data[j]-(*ave));                deviation from the mean.
        ep += s;
        *var += (p=s*s);
        *skew += (p *= s);
        *curt += (p *= s);
    }
    *adev /= n;
    *var=(*var-ep*ep/n)/(n-1);                      Corrected two-pass formula.
    *sdev=sqrt(*var);                               Put the pieces together according to the con-
    if (*var) {                                         ventional definitions.
        *skew /= (n*(*var)*(*sdev));
        *curt=(*curt)/(n*(*var)*(*var))-3.0;
    } else nrerror("No skew/kurtosis when variance = 0 (in moment)");
}
```

http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.129.5354&rep=rep1&type=pdf

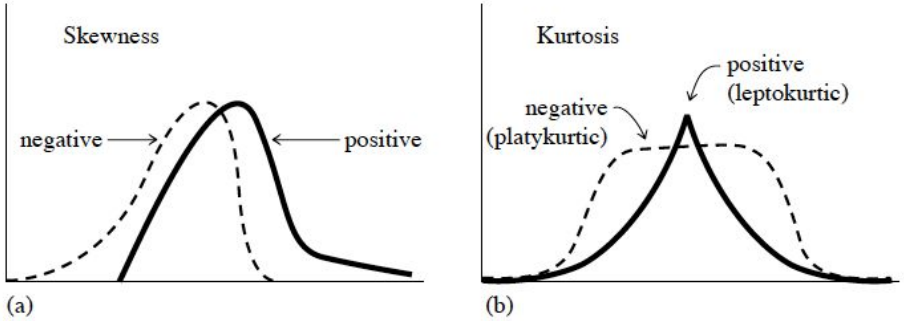


Figure 14.1.1. Distributions whose third and fourth moments are significantly different from a normal (Gaussian) distribution. (a) Skewness or third moment. (b) Kurtosis or fourth moment.

That being the case, the *skewness* or *third moment*, and the *kurtosis* or *fourth moment* should be used with caution or, better yet, not at all.

The skewness characterizes the degree of asymmetry of a distribution around its mean. While the mean, standard deviation, and average deviation are *dimensional* quantities, that is, have the same units as the measured quantities x_j, the skewness is conventionally defined in such a way as to make it *nondimensional*. It is a pure number that characterizes only the shape of the distribution. The usual definition is

$$\text{Skew}(x_1 \dots x_N) = \frac{1}{N}\sum_{j=1}^{N}\left[\frac{x_j - \overline{x}}{\sigma}\right]^3 \tag{14.1.5}$$

where $\sigma = \sigma(x_1 \dots x_N)$ is the distribution's standard deviation (14.1.3). A positive value of skewness signifies a distribution with an asymmetric tail extending out towards more positive x; a negative value signifies a distribution whose tail extends out towards more negative x (see Figure 14.1.1).

Of course, any set of N measured values is likely to give a nonzero value for (14.1.5), even if the underlying distribution is in fact symmetrical (has zero skewness). For (14.1.5) to be meaningful, we need to have some idea of *its* standard deviation as an estimator of the skewness of the underlying distribution. Unfortunately, that depends on the shape of the underlying distribution, and rather critically on its tails! For the idealized case of a normal (Gaussian) distribution, the standard deviation of (14.1.5) is approximately $\sqrt{15/N}$ when $\overline{x}$ is the true mean, and $\sqrt{6/N}$ when it is estimated by the sample mean, (14.1.1). In real life it is good practice to believe in skewnesses only when they are several or many times as large as this.

The kurtosis is also a nondimensional quantity. It measures the relative peakedness or flatness of a distribution. Relative to what? A normal distribution, what else! A distribution with positive kurtosis is termed *leptokurtic*; the outline of the Matterhorn is an example. A distribution with negative kurtosis is termed *platykurtic*; the outline of a loaf of bread is an example. (See Figure 14.1.1.) And, as you no doubt expect, an in-between distribution is termed *mesokurtic*.

The conventional definition of the kurtosis is

$$\text{Kurt}(x_1 \dots x_N) = \left\{\frac{1}{N}\sum_{j=1}^{N}\left[\frac{x_j - \overline{x}}{\sigma}\right]^4\right\} - 3 \tag{14.1.6}$$

where the -3 term makes the value zero for a normal distribution.

The standard deviation of (14.1.6) as an estimator of the kurtosis of an underlying normal distribution is $\sqrt{96/N}$ when σ is the true standard deviation, and $\sqrt{24/N}$ when it is the sample estimate (14.1.3). However, the kurtosis depends on such a high moment that there are many real-life distributions for which the standard deviation of (14.1.6) as an estimator is effectively infinite.

Calculation of the quantities defined in this section is perfectly straightforward. Many textbooks use the binomial theorem to expand out the definitions into sums of various powers of the data, e.g., the familiar

$$\text{Var}(x_1 \dots x_N) = \frac{1}{N-1}\left[\left(\sum_{j=1}^{N} x_j^2\right) - N\overline{x}^2\right] \approx \overline{x^2} - \overline{x}^2 \tag{14.1.7}$$

but this can magnify the roundoff error by a large factor and is generally unjustifiable in terms of computing speed. A clever way to minimize roundoff error, especially for large samples, is to use the *corrected two-pass algorithm* [1]: First calculate $\overline{x}$, then calculate $\text{Var}(x_1 \dots x_N)$ by

$$\text{Var}(x_1 \dots x_N) = \frac{1}{N-1}\left\{\sum_{j=1}^{N}(x_j - \overline{x})^2 - \frac{1}{N}\left[\sum_{j=1}^{N}(x_j - \overline{x})\right]^2\right\} \tag{14.1.8}$$

The second sum would be zero if $\overline{x}$ were exact, but otherwise it does a good job of correcting the roundoff error in the first term.

```
#include <math.h>

void moment(float data[], int n, float *ave, float *adev, float *sdev,
    float *var, float *skew, float *curt)
Given an array of data[1..n], this routine returns its mean ave, average deviation adev,
standard deviation sdev, variance var, skewness skew, and kurtosis curt.
{
    void nrerror(char error_text[]);
    int j;
    float ep=0.0,s,p;

    if (n <= 1) nrerror("n must be at least 2 in moment");
    s=0.0;                                          First pass to get the mean.
    for (j=1;j<=n;j++) s += data[j];
    *ave=s/n;
    *adev=(*var)=(*skew)=(*curt)=0.0;               Second pass to get the first (absolute), sec-
    for (j=1;j<=n;j++) {                                ond, third, and fourth moments of the
        *adev += fabs(s=data[j]-(*ave));                deviation from the mean.
        ep += s;
        *var += (p=s*s);
        *skew += (p *= s);
        *curt += (p *= s);
    }
    *adev /= n;
    *var=(*var-ep*ep/n)/(n-1);                      Corrected two-pass formula.
    *sdev=sqrt(*var);                               Put the pieces together according to the con-
    if (*var) {                                         ventional definitions.
        *skew /= (n*(*var)*(*sdev));
        *curt=(*curt)/(n*(*var)*(*var))-3.0;
    } else nrerror("No skew/kurtosis when variance = 0 (in moment)");
}
```

E.g.: Numerical Recipes in C

```
#include <math.h>

void moment(float data[], int n, float *ave, float *adev, float *sdev,
    float *var, float *skew, float *curt)
```
Given an array of `data[1..n]`, this routine returns its mean `ave`, average deviation `adev`, standard deviation `sdev`, variance `var`, skewness `skew`, and kurtosis `curt`.
```
{
    void nrerror(char error_text[]);
    int j;
    float ep=0.0,s,p;

    if (n <= 1) nrerror("n must be at least 2 in moment");
    s=0.0;                                  First pass to get the mean.
    for (j=1;j<=n;j++) s += data[j];
    *ave=s/n;
    *adev=(*var)=(*skew)=(*curt)=0.0;       Second pass to get the first (absolute), sec-
    for (j=1;j<=n;j++) {                        ond, third, and fourth moments of the
        *adev += fabs(s=data[j]-(*ave));        deviation from the mean.
        ep += s;
        *var += (p=s*s);
        *skew += (p *= s);
        *curt += (p *= s);
    }
    *adev /= n;
    *var=(*var-ep*ep/n)/(n-1);              Corrected two-pass formula.
    *sdev=sqrt(*var);                       Put the pieces together according to the con-
    if (*var) {                                 ventional definitions.
        *skew /= (n*(*var)*(*sdev));
        *curt=(*curt)/(n*(*var)*(*var))-3.0;
    } else nrerror("No skew/kurtosis when variance = 0 (in moment)");
}
```

Naming conventions help everyone

There's lots of good material on this on the web. E.g.:
https://google.github.io/styleguide/cppguide.html
https://www.ee.columbia.edu/~marios/matlab/MatlabStyle1p5.pdf
... but try to find out what your research group uses and stick to that, and if there isn't one, pick one and try to persuade others to adopt it.

General rules:

- Constant, variable or function must be self describing
- Constants are always capitalized:

  ```
  #define TIMELIMIT  100 /* ms before user response times out */
  ```

- Variable names should be in mixed case starting with lower case:

  ```
  linearity, credibleThreat, qualityOfLife
  ```

  This is common practice in the C++ development community. Some communities sometimes starts variable names with upper case, but that usage is commonly reserved for types or structures in other languages.

Naming convention: Variables

Variables with a large scope should have meaningful names. Variables with a small scope can have short names.

In practice most variables should have meaningful names. The use of short names should be reserved for conditions where they clarify the structure of the statements. Scratch variables used for temporary storage or indices can be kept short. A programmer reading such variables should be able to assume that its value is not used outside a few lines of code. Common scratch variables for integers are `i, j, k, m, n` and for doubles `x, y` and `z.`

The prefix *n* should be used for variables representing the number of objects.

This notation is taken from mathematics where it is an established convention for indicating the number of objects.

```
nFiles, nSegments
```

A MATLAB-specific addition is the use of m for number of rows (based on matrix notation), as in

```
mRows
```

A convention on pluralization should be followed consistently.

A suggested practice is to make all variable names either singular or plural. Having two variables with names differing only by a final letter s should be avoided. An acceptable alternative for the plural is to use the suffix Array.

```
point,   pointArray
```

Variables representing a single entity number can be suffixed by *No* or prefixed by *i*.

The No notation is taken from mathematics where it is an established convention for indicating an entity number.

```
tableNo, employeeNo
```

The i prefix effectively makes the variables named iterators.

```
iTable, iEmployee
```

Naming convention: Variables

Iterator variables should be named or prefixed with *i*, *j*, *k* etc.

The notation is taken from mathematics where it is an established convention for indicating iterators.

```
for iFile = 1:nFiles
    :
end
```

Note that applications using complex numbers should reserve i, j or both for use as the imaginary number.

For nested loops the iterator variables should be in alphabetical order.

For nested loops the iterator variables should be helpful names.

```
for iFile = 1:nFiles
   for jPosition = 1:nPositions
      :
   end
   :
end
```

Negated boolean variable names should be avoided.

A problem arises when such a name is used in conjunction with the logical negation operator as this results in a double negative. It is not immediately apparent what `~isNotFound` means.

Use `isFound`

Avoid `isNotFound`

Acronyms, even if normally uppercase, should be mixed or lower case.

Using all uppercase for the base name will give conflicts with the naming conventions given above. A variable of this type would have to be named `dVD`, `hTML` etc. which obviously is not very readable. When the name is connected to another, the readability is seriously reduced; the word following the abbreviation does not stand out as it should.

Use `html`, `isUsaSpecific`, `checkTiffFormat()`

Avoid `hTML`, `isUSASpecific`, `checkTIFFFormat()`

Avoid using a keyword or special value name for a variable name.

MATLAB can produce cryptic error messages or strange results if any of its reserved words or builtin special values is redefined. Reserved words are listed by the command iskeyword. Special values are listed in the documentation.

Constants

Named constants (including globals) should be all uppercase using underscore to separate words.

This is common practice in the C++ development community. Although TMW may appear to use lower case names for constants, for example pi, such builtin constants are actually functions.

`MAX_ITERATIONS`, `COLOR_RED`

Constants can be prefixed by a common type name.

This gives additional information on which constants belong together and what concept the constants represent.

`COLOR_RED`, `COLOR_GREEN`, `COLOR_BLUE`

Structures

Structure names should begin with a capital letter.

This usage is consistent with C++ practice, and it helps to distinguish between structures and ordinary variables.

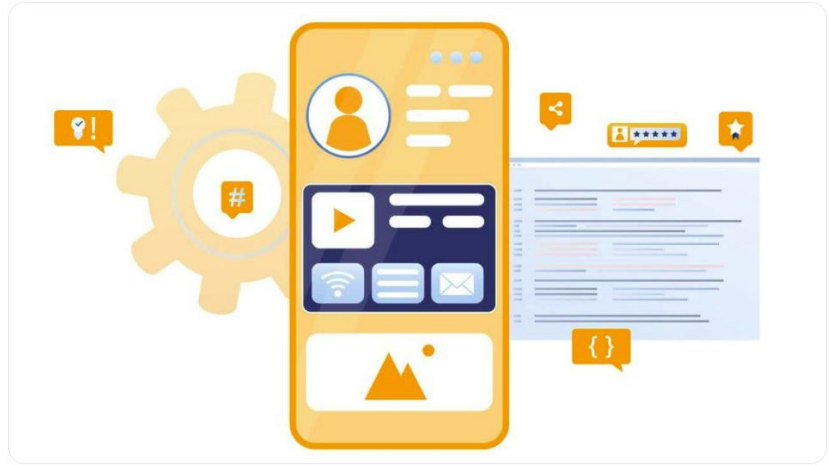
The name of the structure is implicit, and need not be included in a fieldname.

Repetition is superfluous in use, as shown in the example.

Use `Segment.length`

Avoid `Segment.segmentLength`

Code profiling



Why should I profile code?

- To make it more efficient
- To spot bugs in your logic
- To ‘sniff-test’ the code
- To avoid hogging resources
- To compute ethically (using the minimum carbon footprint)

A code profiler allows you to optimize your code by analyzing:

- Time spent in each function / subroutine
- Number of times a function is called
- The order in which functions are called
- The memory usage at each step

Code profiling

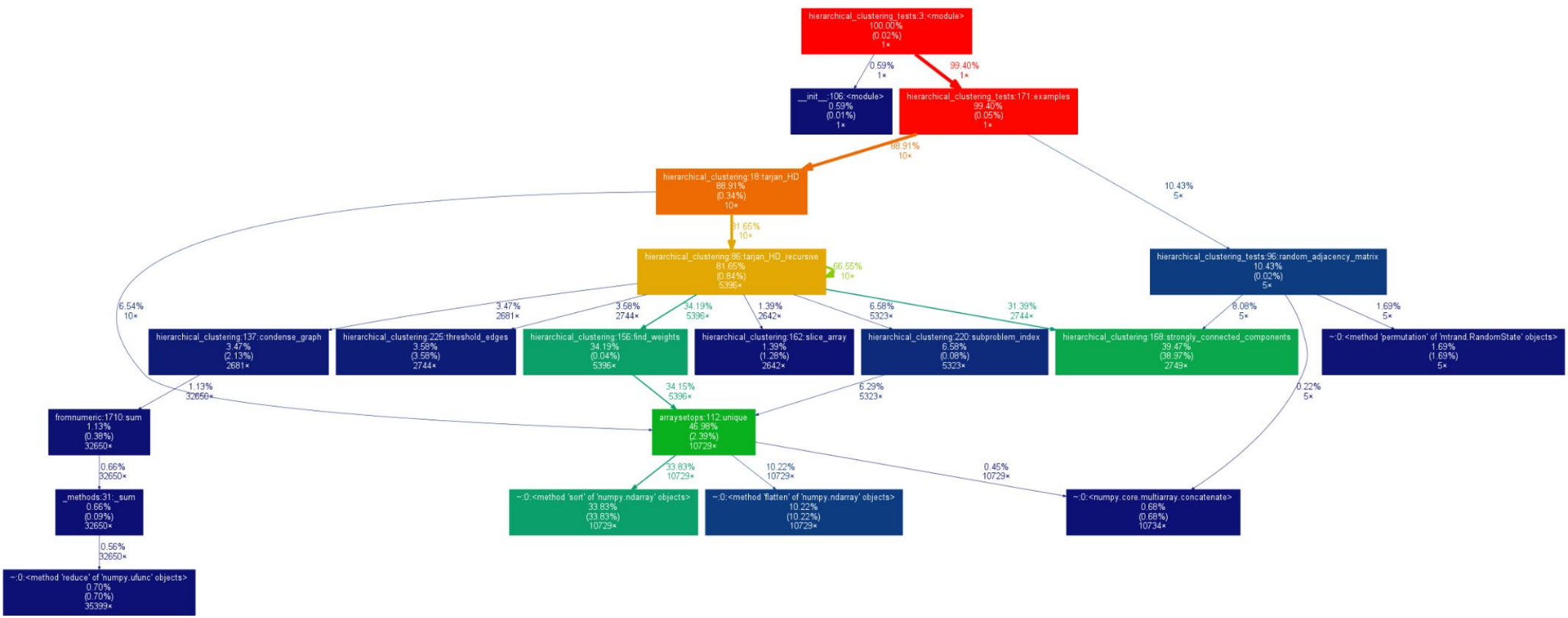
E.g., *cProfile* in Python:

```
import cProfile
import re

cProfile.run('re.compile("foo|bar")')

197 function calls (192 primitive calls) in 0.002 seconds
Ordered by: standard name
ncalls tottime percall cumtime percall filename:lineno(function)
1 0.000 0.000 0.001 0.001 :1()
1 0.000 0.000 0.001 0.001 re.py:212(compile)
1 0.000 0.000 0.001 0.001 re.py:268(_compile)
1 0.000 0.000 0.000 0.000 sre_compile.py:172(_compile_charset)
1 0.000 0.000 0.000 0.000 sre_compile.py:201(_optimize_charset)
4 0.000 0.000 0.000 0.000 sre_compile.py:25(_identityfunction)
3/1 0.000 0.000 0.000 0.000 sre_compile.py:33(_compile)
```

Python code profiling example

Example: If your script is ‘example.py’, then you can run:
python -m cProfile -o output.pstats example.py
gprof2dot -f pstats output.pstats | dot -Tpng -o output.png

Below is a real-life visualization of the output from these commands. It shows the function calls, the numbers of function calls, and the cumulative run times of the function calls for the ‘example.py’ script.

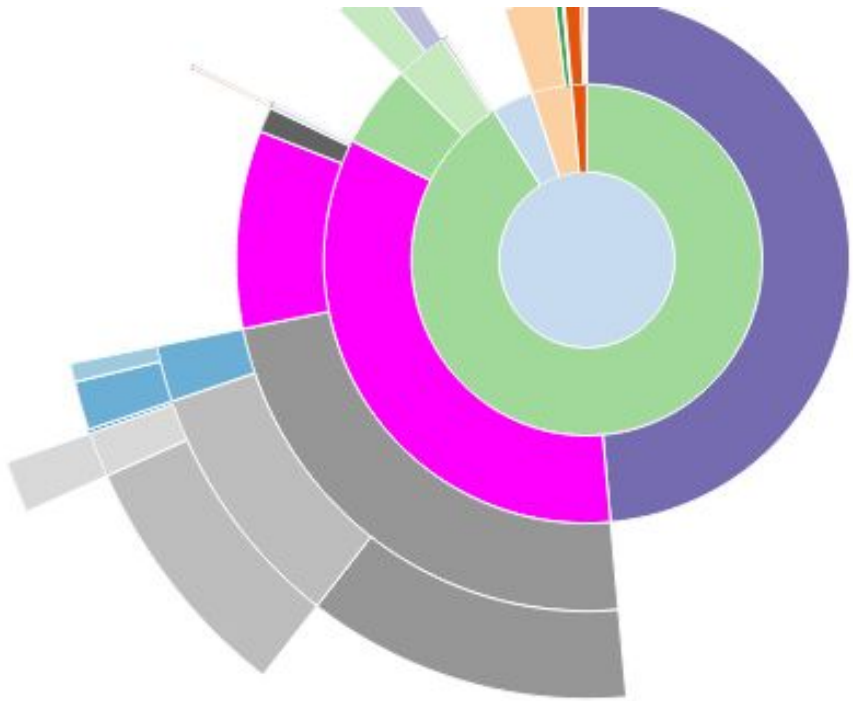


Code Profiling and Visualization

Profile viewers aren't really profilers, but they can help turn your profiling statistics into a more visually pleasing display.

One example is SnakeViz, which is a browser-based graphical viewer for the output of Python's cProfile module.

Name:
`filter`
Cumulative Time:
`0.000294 s (31.78 %)`
File:
`fnmatch.py`
Line:
`48`
Directory:
`/Users/jiffyclub/miniconda3/envs/snakevizdev/lib/python3.4/`



Code Profiling - Python

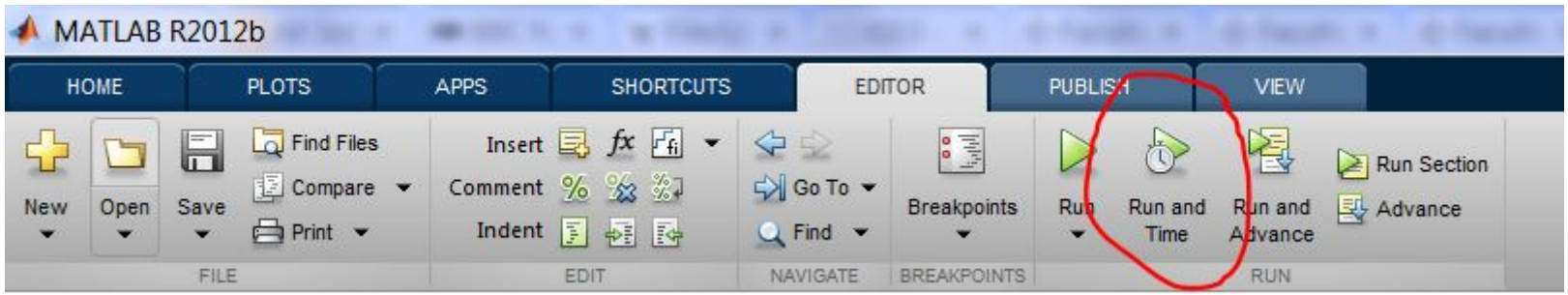
You can display statistics from *cProfile* using *Tuna*.

Tuna handles runtime and import profiles,

It uses D3 and Bootstrap as the underlying technologies for display.

See here for more info on profiling in Python.

Code Profiling - Matlab



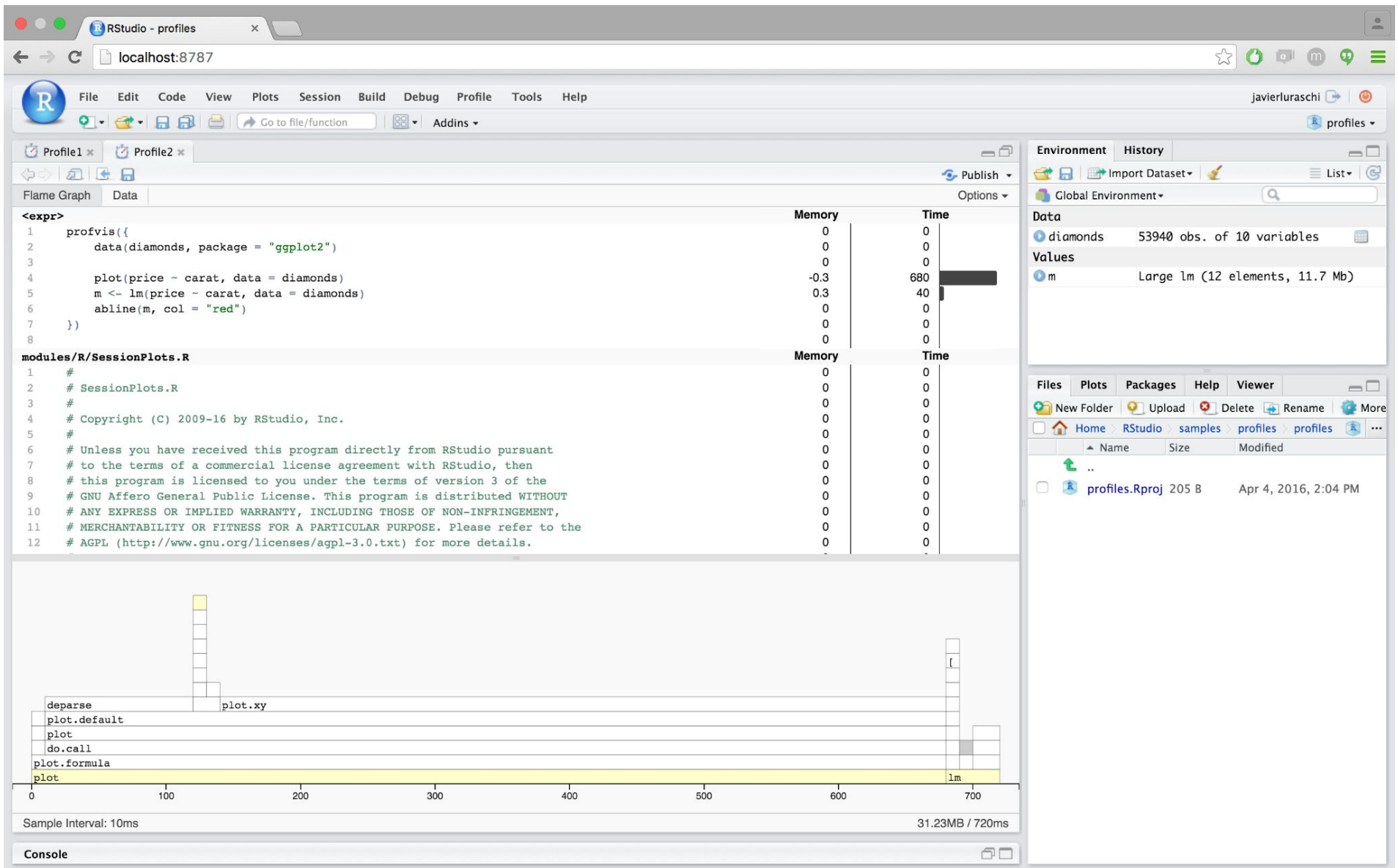
Function Name	Function Type	Calls
@codistributed/private/mtimesImpl	function	1

Lines where the most time was spent including the top 5 code lines from the comparison worker (maroon)

Line Number (for worker 50 and *62*)	Code	Calls	Total Time	Data Sent	Data Rec	Comm Waiting Time	Active Comm Time	% Time	Time Plot
121	`LPC = LPC + LPA*LPB(k, :);`	63 *63*	136.204 s *127.895 s*	0 b *0 b*	0 b *0 b*	0 s *0 s*	0 s *0 s*	78.4% *76.0%*	
119	`LPA = labSendReceive(to, from,...`	63 *63*	34.335 s *37.679 s*	18.34 Gb *18.34 Gb*	18.34 Gb *18.34 Gb*	0.095 s *22.925 s*	32.579 s *13.525 s*	19.8% *22.4%*	
114	`LPC(:, :) = LPA*LPB(k, :);`	1 *1*	3.032 s *2.567 s*	0 b *0 b*	0 b *0 b*	0 s *0 s*	0 s *0 s*	1.7% *1.5%*	
120	`k = codistrA.globalIndices(2, ...`	63 *63*	0.046 s *0.048 s*	0 b *0 b*	0 b *0 b*	0 s *0 s*	0 s *0 s*	0.0% *0.0%*	
130	`end  % End of hMtimesImpl`	1 *1*	0.041 s *0.019 s*	0 b *0 b*	0 b *0 b*	0 s *0 s*	0 s *0 s*	0.0% *0.0%*	
111	`codistrC = codistributed(2`	1	0.041 s	0 b	0 b	0 s	0 s	0.0%	

https://www.mathworks.com/help/matlab/ref/profile.html

Code Profiling - R



https://support.rstudio.com/hc/en-us/articles/218221837-Profiling-R-code-with-the-RStudio-IDE

Code Profiling - C

Gprof, Valgrind,
… and many others.

```
$ gcc -Wall -pg test_gprof.c test_gprof_new.c -o test_gprof
```

```
$ ./test_gprof

 Inside main()

 Inside func1

 Inside new_func1()

 Inside func2
```

```
Flat profile:

Each sample counts as 0.01 seconds.
%    cumulative self          self   total
time seconds    seconds calls s/call s/call name
33.86 15.52     15.52    1    15.52  15.52  func2
33.82 31.02     15.50    1    15.50  15.50  new_func1
33.29 46.27     15.26    1    15.26  30.75  func1
0.07  46.30     0.03                       main

% the percentage of the total running time of the
time program used by this function.

cumulative a running sum of the number of seconds accounted
seconds for by this function and those listed above it.
```

```
$  gprof test_gprof gmon.out > analysis.txt
```

Performing a code review

- The Standard of Code Review
- What to Look For In a Code Review
- Navigating a CL in Review
- Speed of Code Reviews
- How to Write Code Review Comments
- Handling Pushback in Code Reviews

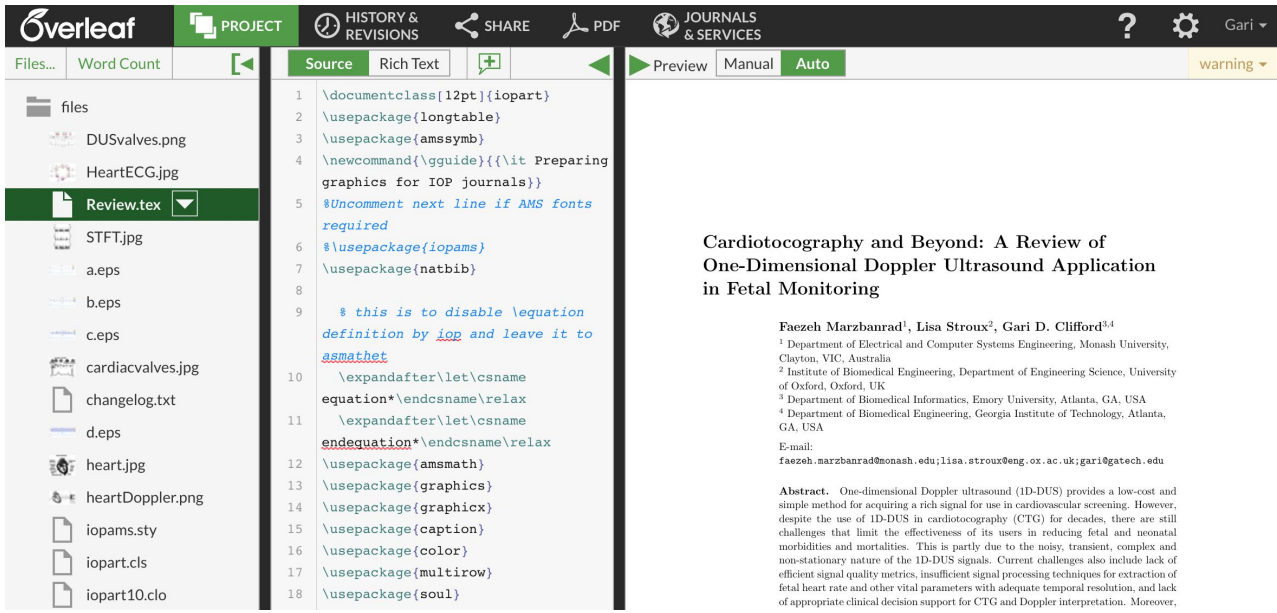
1. Know What to Look for in a Code Review
2. **Build and Test — Before Review**
3. Don't Review Code for Longer Than 60 Minutes
4. Check No More Than 400 Lines at a Time
5. Give Feedback That Helps (Not Hurts)
6. Communicate Goals and Expectations
7. Include Everyone in the Code Review Process
8. Foster a Positive Culture
9. **Automate to Save Time**

https://smartbear.com/learn/code-review/best-practices-for-peer-code-review/
https://google.github.io/eng-practices/review/reviewer/
https://www.perforce.com/blog/qac/9-best-practices-for-code-review

Documentation

Lab Diary:

- Google Docs
- Jupyter?



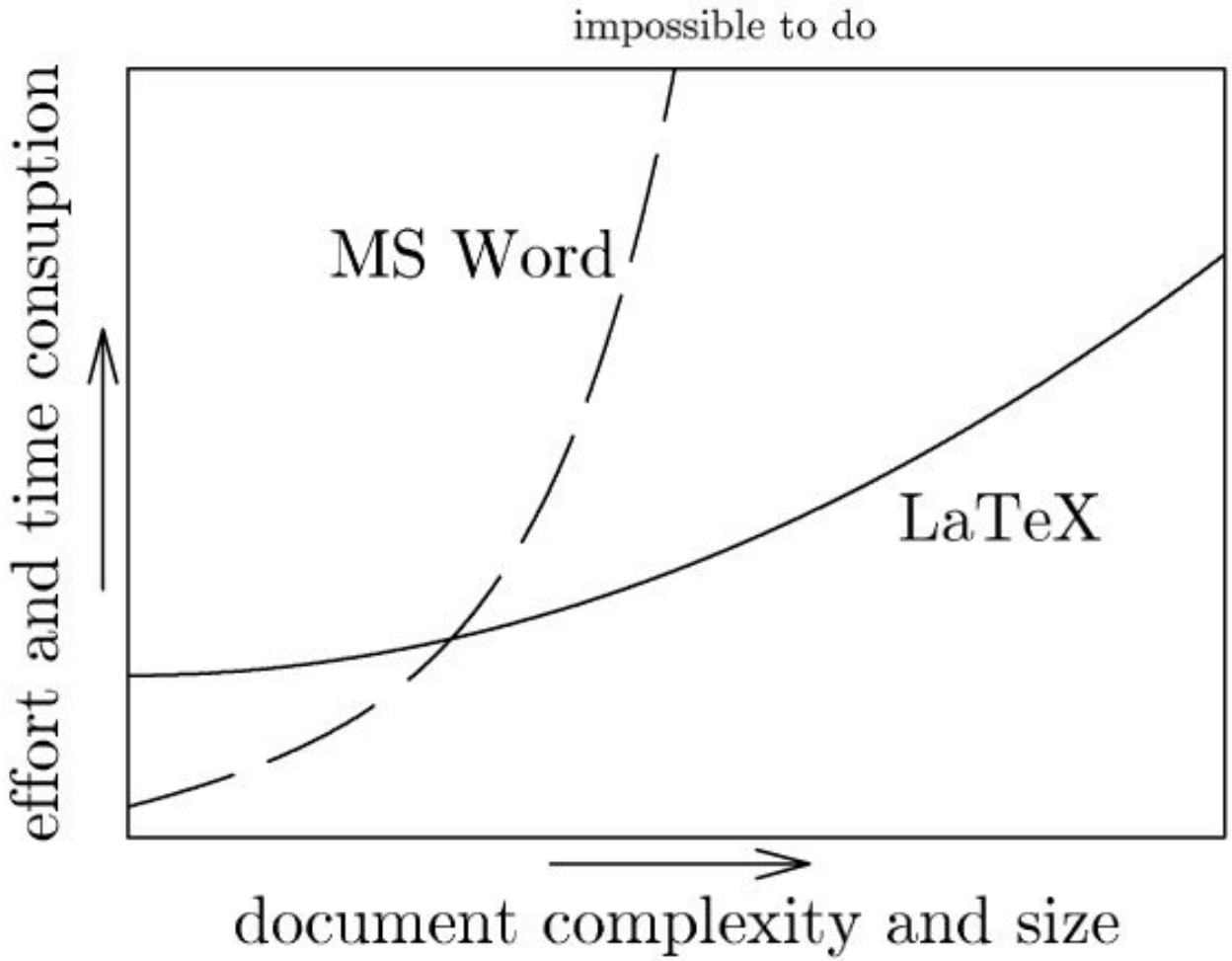
Articles for publication:

- Overleaf.com
- Google Docs with reference manager plugin (Paperpile)
- Avoid Word if you can - it crashes for large files, online collaboration doesn't work well & version incompatibility issues exist with Endnote and word itself.
- If you must use it, back up often.

Thesis:

- Latex - locally (with github) or better still **Overleaf.com**
- Thesis Guide: https://www.overleaf.com/read/vdtvfvkbbxdn

Why Latex?



https://ebiquity.umbc.edu/blogger/wp-content/uploads/2009/07/miktex.gif

Formatting and structure tips

Editorial: How to write a decent scientific article or thesis

Gari D Clifford[1,2] and Bob B Smoot[3]
[1] Department of Biomedical Informatics, Emory University, Atlanta, GA USA
[2] Department of Biomedical Engineering, Georgia Institute of Technology, Atlanta, GA, USA
[3] Institute for Medical Engineering & Science, Massachusetts Institute of Technology, USA

E-mail: gari@gatech.edu

Abstract.

This article is designed to serve as a guide to writing a coherent and acceptable scientific document, particularly for scientific journals or a research thesis. (The LaTeXsource for this editorial can be found here: https://www.overleaf.com/read/vdtvfvkbbxdn.) The rationale for doing this is that it seems that graduate students are no longer taught these skills, and I grew tired of saying the same things over and over. As an editor I have seen a rise in lower quality publications, and as a supervisor, I've seen a drop in quality of weekly reports.

Although I take all the blame for the mistakes in here, I have borrowed liberally from other authors and although I've attempted to credit them where I remembered, I have surely forgotten to in parts, since the information contained in this article has accumulated in my brain over multiple decades. If you think I've forgotten to cite you, let me know and I'll try to find the time to correct any errors or omissions.

This article is structured such at the information for each section appears in the section to which it refers. Therefore, the rest of this abstract details how to write an abstract, and as such is much longer than any abstract should be.

In fact, the abstract should be just a couple of paragraphs to set the scene and provide the motivation (including the clinical rationale if appropriate), then describe your key contributions to scientific knowledge and key results.

https://www.overleaf.com/read/vdtvfvkbbxdn

Homework for the semester ...

- Copy the overleaf template: https://www.overleaf.com/read/vdtvfvkbbxdn

- Make a copy of the article and remove the contents (no logon necessary). Make sure the project is publicly readable.

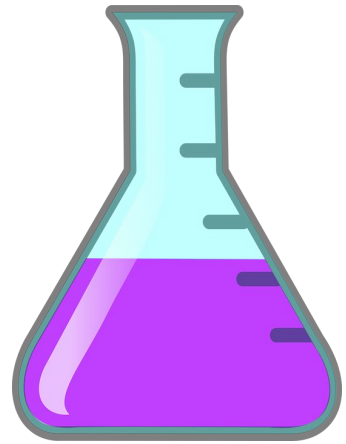
- Rename it [SURNAME] BMI 500 Fall 2021 - Final Course Assignment", where [SURNAME] = your surname.

- Write a 4-8 page review on a research topic of your choice in this overleaf template, following the formatting instructions and tips on how to make a high quality article.
    - Please include at least one table and one figure.
    - The topic is unimportant, but must be somewhat related to the course.
    - It does not have to be a comprehensive review
    - Your grade will depend on your ability to follow formatting requirements, rather than how comprehensive you survey the field.
    - Pay attention to captions, images, spelling, bibliography, etc.

- Send URL of the project to your friendly TA and gari [at] dbmi.emory.edu with the subject header "[SURNAME] BMI 500 Fall 2021 - Final Course Assignment" where [SURNAME] = your surname.

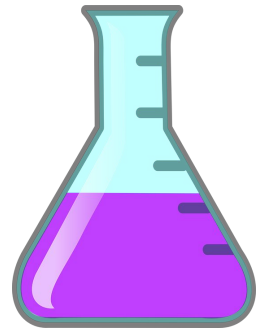
- Deadline = 4th December at 5pm Eastern Standard Time. (1607119200 UTC)

Now to the lab ...



1. Create backup folder in the cloud
2. Create code
3. Check in to Github
4. Check out your neighbor’s code
5. Perform a code review
6. Email me a link to your Github repository for a grade and point me to the exact file you modified

Now to the lab ...



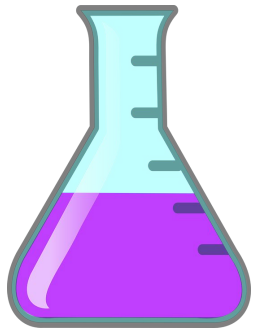
- 1a. Take the MATLAB Tutorial (Onramp): https://www.mathworks.com/learn/tutorials/matlab-onramp.html?s_eid=PDR_23412 and upload the certificate of completion of this course.to:https://drive.google.com/drive/folders/1ex_HL1hRxdEq4ufrS9TKQYDplfWtLw1s (Filename: SURNAME_Matlab_OnRamp_Cert.pdf)

- 1b. Read this guide to writing efficient code in Matlab: https://drive.google.com/file/d/13PZ0oanKyL39OfPUO0p1IcPdF0s5aHA4/view?usp=sharing
  And read the appendices to this presentation.

- 1c. Profile code https://www.mathworks.com/help/matlab/ref/profile.html and submit a screenshot of it to the same GDrive link above (Filename: SURNAME_Matlab_Profile.pdf)

  STANDARD DEADLINES APPLY - Monday 5pm

Now to the lab ...



- 2. Read this Python tutorial: https://www.learnpython.org
  and take this Python quiz
  https://drive.google.com/file/d/1_Zx7tW1wgPIzwszsBaVxt2EnbkD8QDqe/view?usp=sharing
  Submit via email to TA

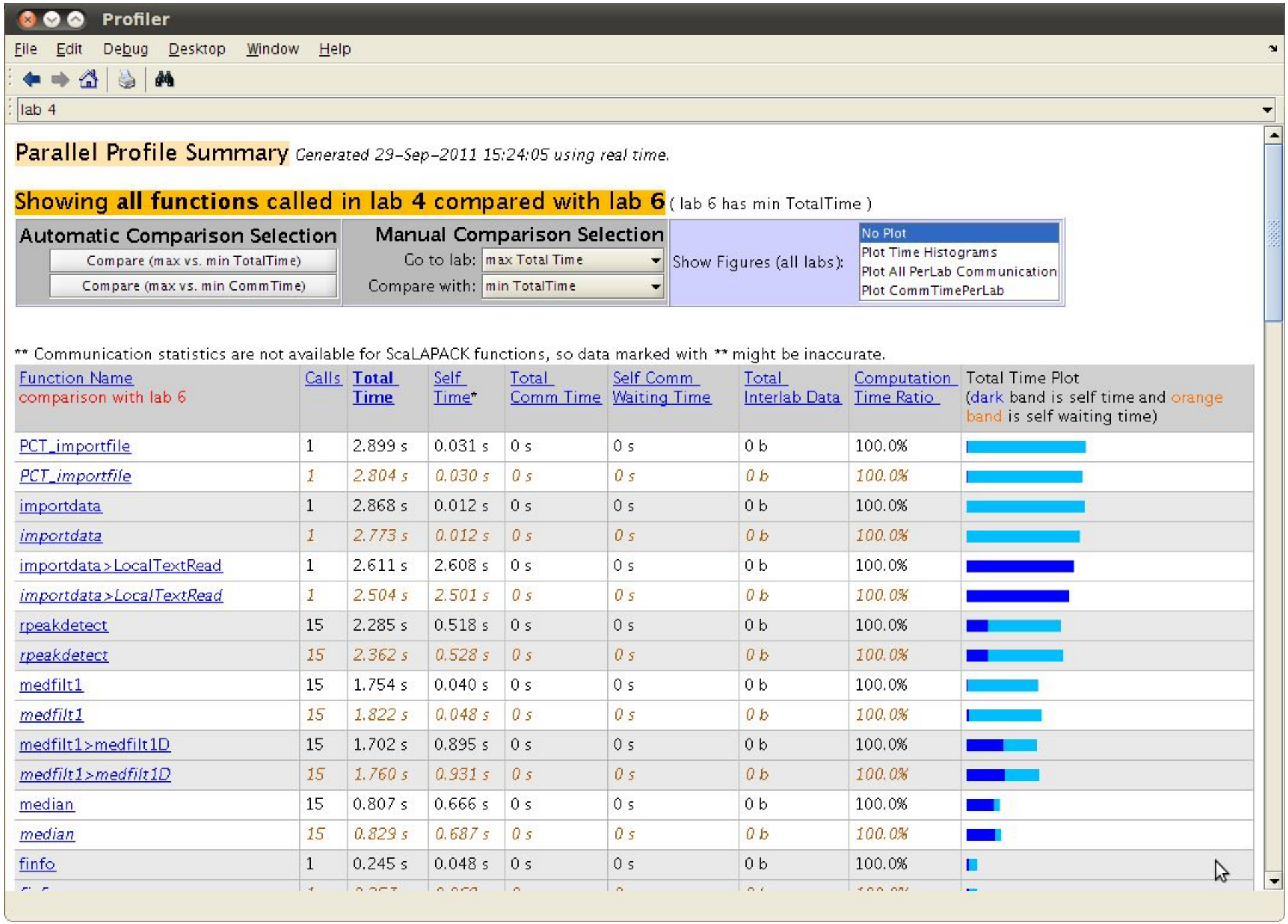
  STANDARD DEADLINES APPLY - Mon 5pm before the next lecture!

- 2b - Profile code and submit it: https://docs.python.org/3/library/profile.html

  submit a screenshot of it to the same GDrive link above
  (Filename: SURNAME_Python_Profile.pdf)

  3. Check all code into github and share a link with TA

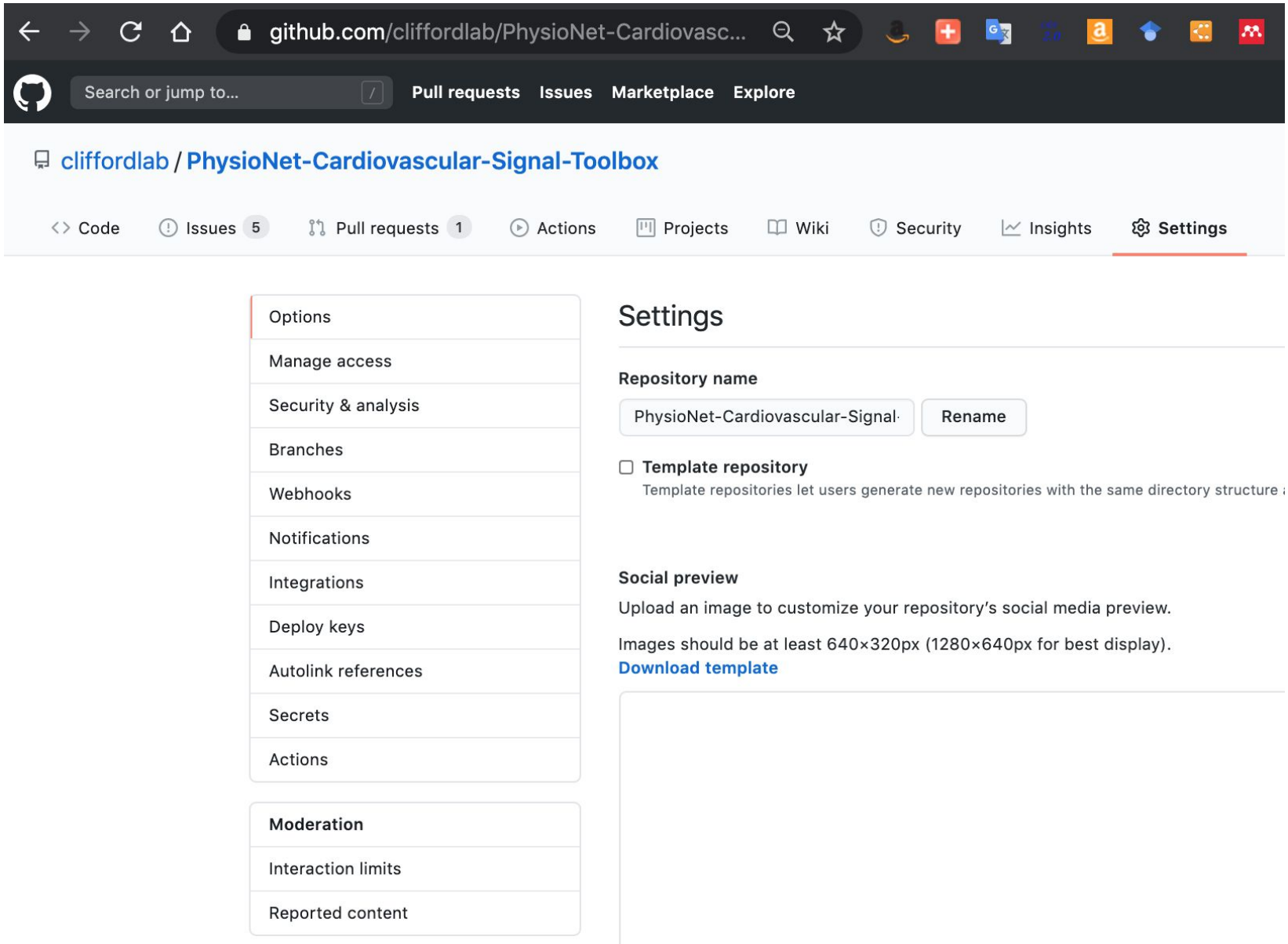
Profile Your Code!!!



Function Name comparison with lab 6	Calls	Total Time	Self Time*	Total Comm Time	Self Comm Waiting Time	Total Interlab Data	Computation Time Ratio	Total Time Plot (dark band is self time and orange band is self waiting time)
PCT_importfile	1	2.899 s	0.031 s	0 s	0 s	0 b	100.0%	
PCT_importfile	*1*	*2.804 s*	*0.030 s*	*0 s*	*0 s*	*0 b*	*100.0%*	
importdata	1	2.868 s	0.012 s	0 s	0 s	0 b	100.0%	
importdata	*1*	*2.773 s*	*0.012 s*	*0 s*	*0 s*	*0 b*	*100.0%*	
importdata>LocalTextRead	1	2.611 s	2.608 s	0 s	0 s	0 b	100.0%	
importdata>LocalTextRead	*1*	*2.504 s*	*2.501 s*	*0 s*	*0 s*	*0 b*	*100.0%*	
rpeakdetect	15	2.285 s	0.518 s	0 s	0 s	0 b	100.0%	
rpeakdetect	*15*	*2.362 s*	*0.528 s*	*0 s*	*0 s*	*0 b*	*100.0%*	
medfilt1	15	1.754 s	0.040 s	0 s	0 s	0 b	100.0%	
medfilt1	*15*	*1.822 s*	*0.048 s*	*0 s*	*0 s*	*0 b*	*100.0%*	
medfilt1>medfilt1D	15	1.702 s	0.895 s	0 s	0 s	0 b	100.0%	
medfilt1>medfilt1D	*15*	*1.760 s*	*0.931 s*	*0 s*	*0 s*	*0 b*	*100.0%*	
median	15	0.807 s	0.666 s	0 s	0 s	0 b	100.0%	
median	*15*	*0.829 s*	*0.687 s*	*0 s*	*0 s*	*0 b*	*100.0%*	
finfo	1	0.245 s	0.048 s	0 s	0 s	0 b	100.0%	

PCT_spmd_profiler.m

Use Github always!



Github and online notebooks

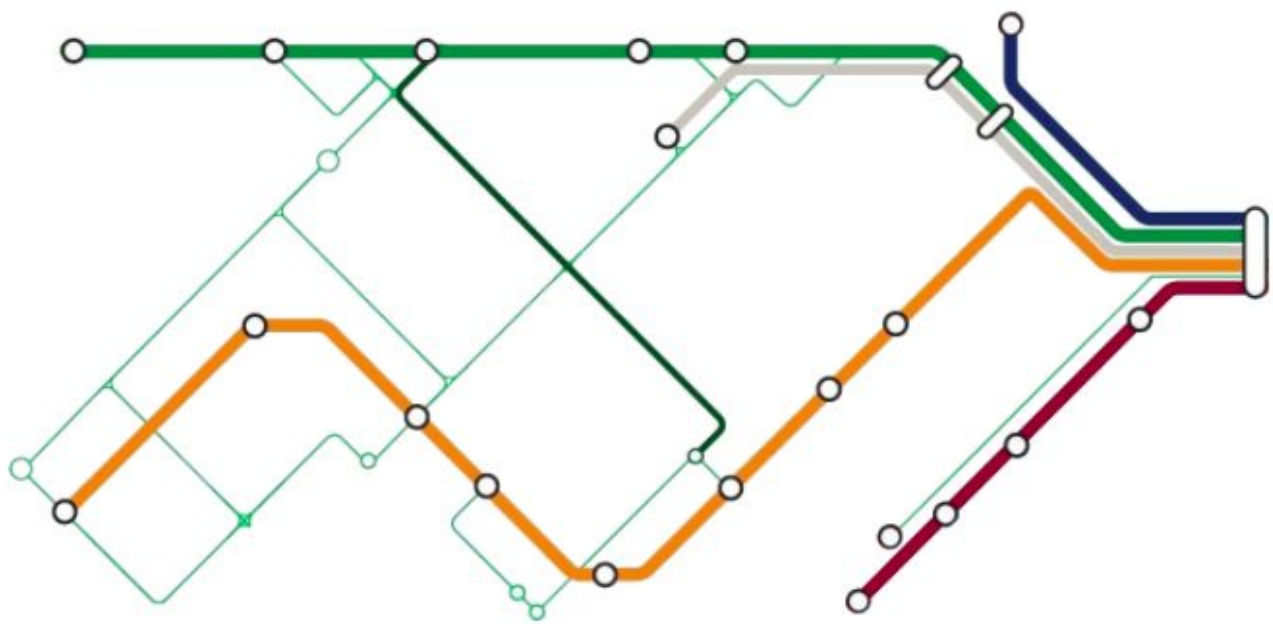
David Schmudde / Nov 08 2019 / PUBLISHED

with bharendt, Andrea Amantini, Philippa Markovics and Martin Kavalar

Remix of Python by › Nextjournal

How to Version Control Jupyter Notebooks

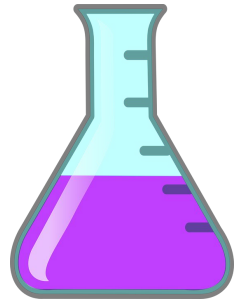
The Definitive Guide



https://nextjournal.com/schmudde/how-to-version-control-jupyter

WARNING

- Do not put protected health information or security access codes in github
- Don’t make it public until it has been reviewed by your supervisor (thoroughly)
- Notebooks are dangerous!

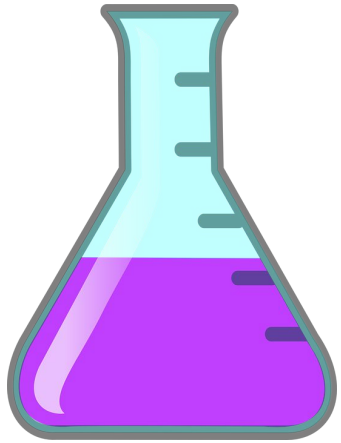


Comparison or Integration?

- Here's a nice cheat sheet that compares Matlab and Python commands: https://cheatsheets.quantecon.org

- Github Matlab integration: https://www.mathworks.com/help/matlab/matlab_prog/set-up-git-source-control.html

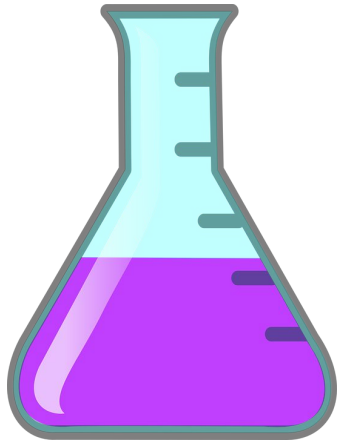
Now to the lab ...



Matlab code example:
https://github.com/cliffordlab/PhysioNet-Cardiovascular-Signal-Toolbox
(My research group's HRV code)

- Modify Basic_Demo.m:
  https://github.com/cliffordlab/PhysioNet-Cardiovascular-Signal-Toolbox/blob/master/Demos/Basic_Demo.m
- Line up the detected peaks and estimate periodicity in it
- Github Matlab integration:
  https://www.mathworks.com/help/matlab/matlab_prog/set-up-git-source-control.html

Now to the lab ...



Or if you prefer …

Python code example:
https://github.com/PIA-Group/BioSPPy/blob/master/README.md

- Line up the detected peaks and estimate periodicity in it

Cluster computing @BMI

Quick introduction to the BMI Computational Cluster

The BMI Computational Cluster consists of multiple separate nodes. For the BMI 500 class, you will run jobs primarily on the below nodes:

- 12 general-purpose nodes (cnode[1-12])

There's no direct access to the computational nodes outside of making a resource request through our cluster management software (Slurm); all jobs running on the computational cluster are submitted through a submission server: **oddjobs.bmi.emory.edu**.

Log into Oddjobs via SSH using your BMI account (you will be issued one today as long as you've completed the training from last week's lab). *You must be connected to the Emory VPN in order to access this system off-campus.*

Emory VPN & SSH connections

Emory VPN: Software available at https://it.emory.edu/vpntools

- Windows/MacOS users: Download, install, and use the appropriate VPN Standard Client application.
- Linux users:
  - Download and install the 2 VPN Helper Tools packages (linux_f5epi & linux_f5vpn) in the format appropriate for your Linux distribution (Debian/Ubuntu derivatives: .deb; Fedora/openSUSE derivatives: .rpm).
  - Once the Helper Tools are installed, log in to https://vpn.emory.edu via a web browser.

SSH:

- All latest versions of the major OSes (Windows 10/MacOS/Linux) have a SSH client. To connect to the BMI cluster's submission server, open a terminal/command prompt & type **ssh *your_net_id*@oddjobs.bmi.emory.edu** to log in.
  - If you are running an older build of Windows 10 you will need to update your system if you don't have a SSH client & it's not available to install in the Add/Remove Windows Components option in the control panel.
  - If you are running Windows 8 or earlier & can't update to a supported version of Windows, you'll need to install a 3rd party SSH client like PuTTY: https://www.putty.org/

SSH with X Forwarding

Linux systems traditionally render graphics using the X11 protocol, which is handled by a piece of software called a X11 Server.

This protocol can be tunnelled to a remote system via SSH using the **-X** parameter on most SSH clients, but requires your local system to also have the ability to render X11 display data and thus have some sort of X11 Server software installed.

- Windows users: Some free X11 Server options are VcXsrv (https://sourceforge.net/projects/vcxsrv/) or one of the components of the Cygwin project (http://x.cygwin.com/). Xming is also an option, but the free version is very old and has some compatibility issues, so it's not recommended for use.
- MacOS users: You will need to install XQuartz (https://xquartz.org).
- Linux users: Install the X11 server software available in your distribution's package repositories.

After logging into a system with X-forwarding for the first time, you might receive a message similar to this: `/usr/bin/xauth:  file /home/username/.Xauthority does not exist`

- Normally the .Xauthority file will be automatically generated immediately after this message is printed, so just run **ls -la .Xauthority** to see if it already exists in your home directory. If it isn't, then the server you're connecting to may have X11 forwarding disabled.

Running jobs on the BMI Computational Cluster

Jobs on the cluster are managed by the Slurm (Simple Linux Utility for Resource Management) Workload Manager.

All jobs are submitted into queues/"partitions" based on their needs, and an appropriate queue must be selected when submitting a job.

BMI 500 students will have access to the following queues:

- **batch** - allows for CPU-only processing; no GPU access.
- **overflow** - allows for CPU & GPU processing; jobs (particularly GPU jobs) may be preempted if resources are needed by BMI researchers.
    - Due to the possibility of preemption, your code must handle gracefully saving state and stopping if it receives a SIGCONT or SIGTERM signal which will be sent by Slurm to indicate it's about to be preempted; otherwise, you may lose any progress your job has made when it is sent a SIGKILL signal to force it to stop.

You can run the **sinfo** command to see all the queues you have access to run jobs on.

Running jobs on the BMI Computational Cluster

2 types of jobs can be run on the cluster:

- **batch jobs** - Submitted via the **sbatch** command, preferably via a submission script (an example is available at */opt/tools/slurm_example_script.sh* on Oddjobs).
  - This is the recommended way of submitting long-running jobs.
- **interactive jobs** - Submitted via the **srun** command directly from the command prompt (*srun* has a different purpose when used in a batch job started by *sbatch*). All parameters for the job must be specified on the command line; there's no script file to set these like there is for batch jobs.
  - The *srun* command's last parameters must be **--pty bash -i** in order to start a generic interactive session on the cluster node your job's been assigned to.
  - This is a good way to interactively test and debug code on the cluster if you're having problems, but once it's working it's best to format it to run as a batch job instead of an interactive one.
  - It's important to make sure to end your interactive session when your code's finished running so that the resources are freed up for others to use.

You can see all currently running & queued jobs on the cluster with the command **squeue -a.** This will show all the jobs running, including those in queues you may not be able to submit jobs to yourself.

Important consideration for interactive jobs:

As stated on the previous slide, it's important to make sure to end your interactive session when your code's finished running so that the resources are freed up for others to use and aren’t going to waste.

When jobs are submitted to the cluster, the resources they requested are allocated specifically for them and can’t be used by others even if they’re not currently being used by the requesting job. The resources are only available again once the job’s ended and its resource allocation is released.

- I've seen users request GPU resources but then leave their session sitting at a Bash prompt doing nothing for days until their job allocation times out; pulling a stunt like this on AWS or GCP would cost you a lot of money for nothing because of the allocated resources, so it's a good idea to get into the habit now of making sure your session ends when your code has finished running.

A note about Python on the BMI cluster:

The BMI cluster has special environments for Python 2.7, 3.6, & 3.8 installed using the Software Collections Library framework, allowing for packages to be installed for those versions without affecting the system's default Python install. In order to use Python on the cluster, you will need to run your code within these environments using the appropriate command:

- Python 3.6: "scl enable rh-python36 '<command>'"
- Python 3.8: "scl enable rh-python38 '<command>'"
- Python 2.7: "scl enable python27 '<command>'"

Example: **scl enable rh-python36 'python myPythonScript.py'**

If <command> is "bash" or is a standard script file, then this will give you an environment where all the Python commands (python, pip, virtualenv, etc.) will use the version of Python associated with the SCL.

The best way to run multiple scripts or commands within the same environment would be to wrap them within a bash script file, so that you can do something like this: **scl enable rh-python36 'runMyCode.sh'**

Final notes to keep in mind when using BMI's cluster:

- Do not store large datasets in your home directory located in /home/*username*; the total amount of space is shared among all members of BMI.
  - If you need to download a large public dataset for this class, or if you want to build a virtual environment using virtenv/Conda rather than using the SCL environments on the cluster nodes, use **/opt/bmi-585r/BMI-500-Fall2022/** to do so - this location has better performance for file I/O than /home does, but note that it is *scratch space* and nothing important should have its one and only copy located here for an extended amount of time.

- While the primary method of interacting with the cluster is through SSH, there is a web interface available at https://oddjobs.bmi.emory.edu which can be used to launch Jupyter Lab sessions or a generic desktop session on a computational node for use with GUI-based programs like Spyder, RStudio, or Matlab.
  - You can leave these sessions open to return to them later, but if you know you're finished with them please shut them down so that the allocated resources are freed for others to use.
  - The GUI also has functionality for managing batch submit jobs if you'd prefer to use that rather than the terminal, however you may find it limited in capabilities compared to using SSH (or the terminal that's also available through the GUI).

Back to the lab ...

github.com/cliffordlab/BPimageTranscribe

Google Calendar | Messages for web | G Photos | Google Bookmark | Google Docs | Gmail Offline | Save to Mendeley | Google Scholar Ci... | Other Bookmarks

main | 1 branch | 1 tag | Go to file | Add file | Code

nkatebi Add files via upload — bdb8f0e on May 24 — 58 commits

Dataset	Delete readme.txt	8 months ago
GoogleVisionAPI	Add files via upload	8 months ago
Tesseract_OCR	Add files via upload	8 months ago
Test_case/test_data	Add files via upload	8 months ago
code	Update requirements.txt	8 months ago
Blood_Pressure_Digitization_from_...	Add files via upload	3 months ago
Copyright	Create Copyright	8 months ago
LICENSE	Update LICENSE	8 months ago
README.md	Update README.md	3 months ago
transcription_results.jpg	Add files via upload	8 months ago

README.md

CNN-based LCD Transcription of Blood Pressure from a Mobile Phone Camera

DOI 10.5281/zenodo.4383306

This repository contains the source code for the work in this article: SS Kulkarni, N Katebi, CE Valderrama, P Rohloff, GD Clifford, CNN-based LCD Transcription of Blood Pressure from a Mobile Phone Camera, Front. Artif. Intell., 4, 36, 21 May 2021 as part of the Medicine and Public Health Special Edition on Reimagining Health Systems: leveraging AI/ML for Low- and Middle-Income Countries.

Please cite the above article when using the code in this repository.

About

Code base to transcribe blood pressure readings from photos of LCD display of Omron oscillometric devices

Readme

View license

Releases 1

v1.0.0 Latest
on Dec 21, 2020

Packages

No packages published
Publish your first package

Contributors 3

nkatebi
skulkarni307
gariclifford Gari

Languages

Python 100.0%

https://github.com/cliffordlab/BPimageTranscribe

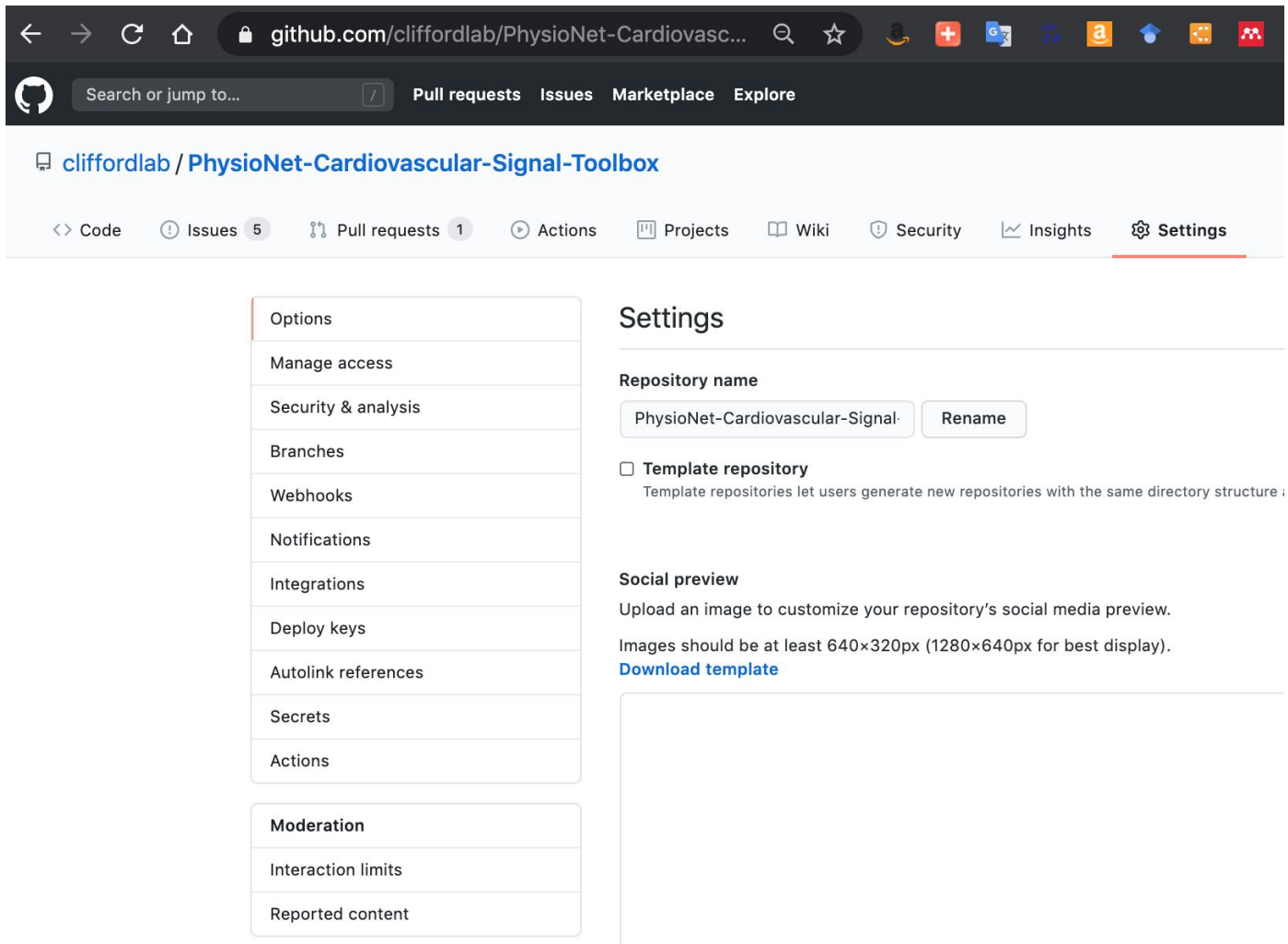
https://github.com/cliffordlab/BPimageTranscribe

Task:

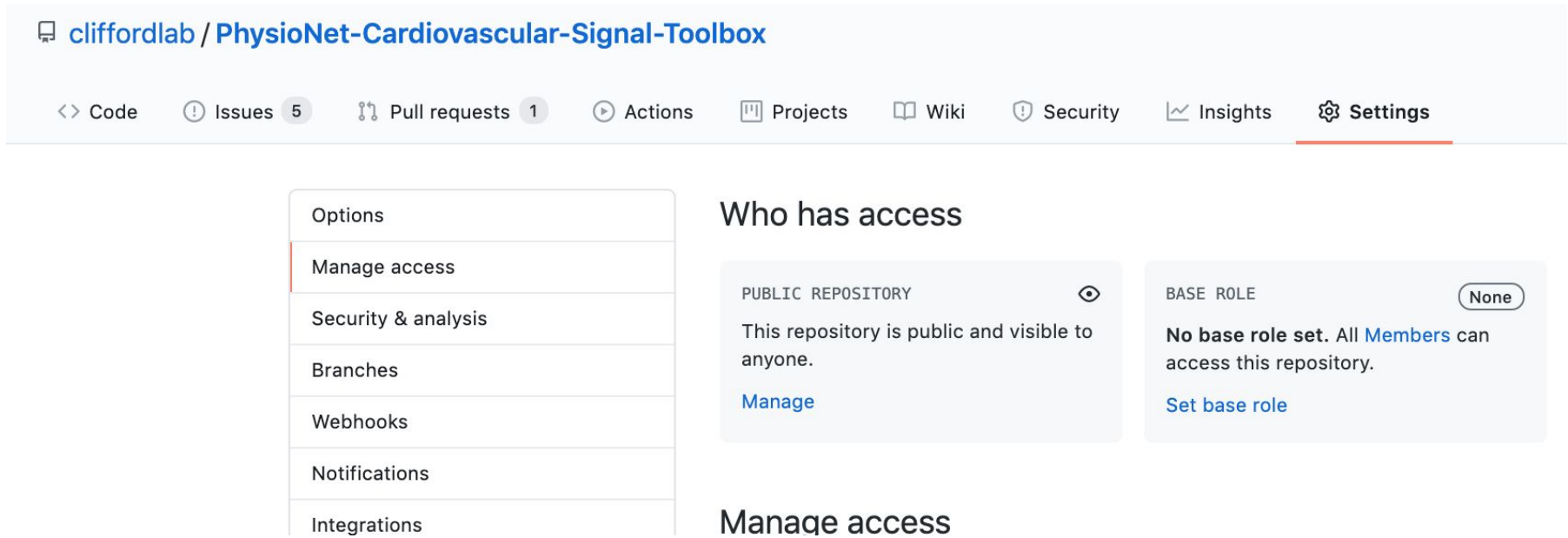
- Log onto cluster.
- Checkout code. https://github.com/cliffordlab/BPimageTranscribe
- Add in profiling code.
- Run code with profiler.
- Export results.
- Visualize results.
- Analyze them (interpret).
- Write this up with the visualization, and submit as a single PDF to the homework folder: https://drive.google.com/drive/folders/1XtkeT6fETJcbEDAKY7Lg4aWvxlxs19QC
  (Name your work appropriately - like last week!!!)
  [Please make sure your PDF includes a link to your github (shared with the instructors]

Appendices

Github


github.com/cliffordlab/PhysioNet-Cardiovasc...
Search or jump to...
Pull requests
Issues
Marketplace
Explore
cliffordlab / PhysioNet-Cardiovascular-Signal-Toolbox
Code
Issues 5
Pull requests 1
Actions
Projects
Wiki
Security
Insights
Settings
Options
Manage access
Security & analysis
Branches
Webhooks
Notifications
Integrations
Deploy keys
Autolink references
Secrets
Actions
Moderation
Interaction limits
Reported content
Settings
Repository name
PhysioNet-Cardiovascular-Signal
Rename
Template repository
Template repositories let users generate new repositories with the same directory structure
Social preview
Upload an image to customize your repository's social media preview.
Images should be at least 640×320px (1280×640px for best display).
Download template

Github



Danger Zone

Change repository visibility
This repository is currently public.
Change visibility

Transfer ownership
Transfer this repository to another user or to an organization where you have the ability to create repositories.
Transfer

Archive this repository
Mark this repository as archived and read-only.
Archive this repository

Delete this repository
Once you delete a repository, there is no going back. Please be certain.
Delete this repository

Installing *Git* – the easy way

> *Git* is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency.

– The *Git* website

Choose one of the following options.

- Instructions for *Windows*
- Instructions for *Mac*
- Instructions for *Linux*

Installing Python:

Use Python 3.6 from: https://www.anaconda.com/download/

Installing Matlab:

http://it.emory.edu/software/software_distribution.html

Checking out code with git

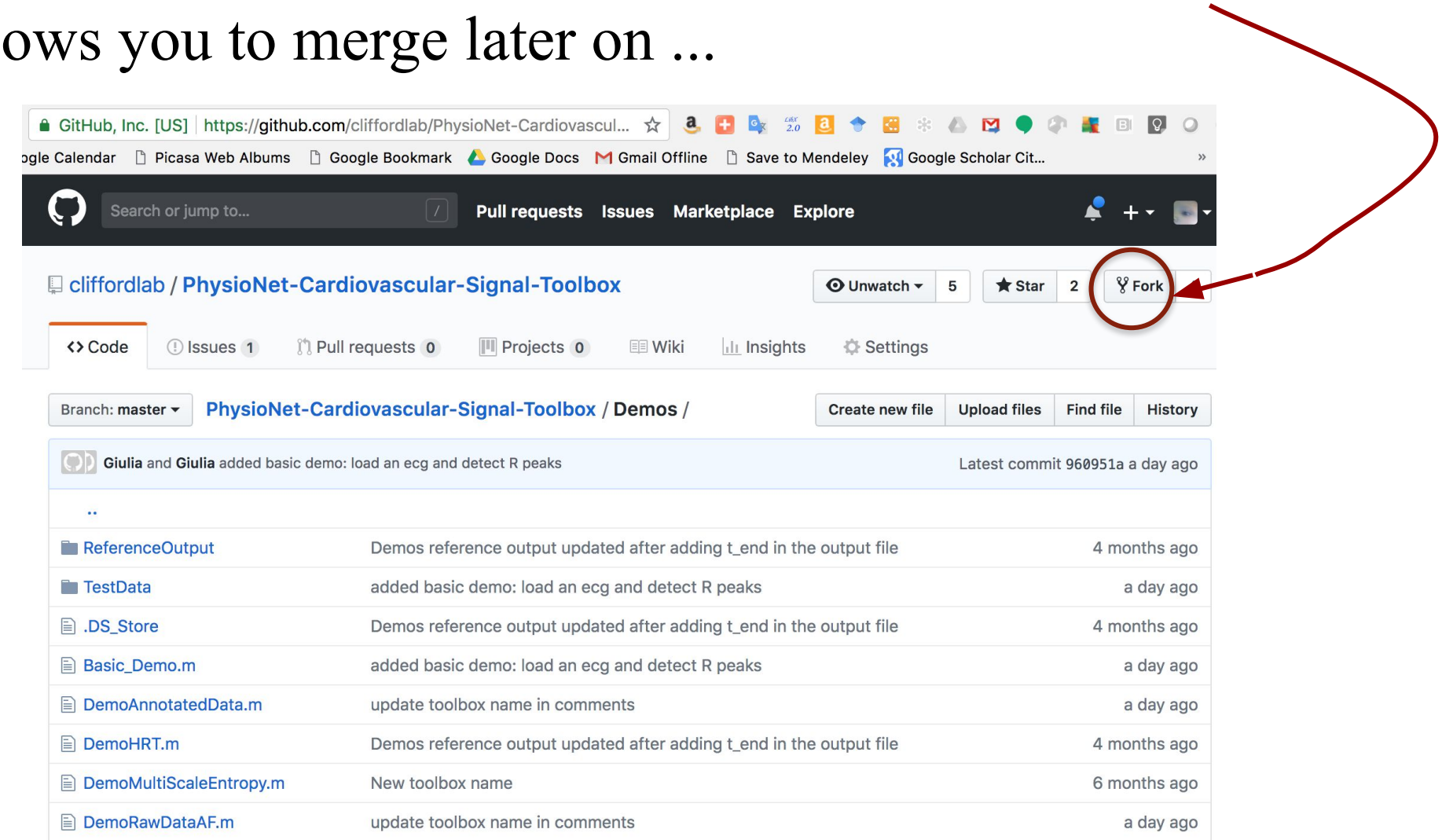
https://dont-be-afraid-to-commit.readthedocs.io/en/latest/git/commandlinegit.html

```
Gari’s MacBook: gari$ cd ~/CODE

Gari’s MacBook:CODE gari$ git clone
https://github.com/cliffordlab/PhysioNet-Cardiovascular-Signal-Toolbox
Cloning into 'PhysioNet-Cardiovascular-Signal-Toolbox'...
remote: Counting objects: 2560, done.
remote: Compressing objects: 100% (29/29), done.
remote: Total 2560 (delta 9), reused 21 (delta 6), pack-reused 2525
Receiving objects: 100% (2560/2560), 37.87 MiB | 7.17 MiB/s, done.
Resolving deltas: 100% (1560/1560), done.

Gari’s MacBook:CODE gari$ ls -alt
total 0
drwxr-xr-x   9 gari  staff   288 Sep  7 08:09
PhysioNet-Cardiovascular-Signal-Toolbox
drwxr-xr-x   3 gari  staff    96 Sep  7 08:09 .
drwxr-xr-x@ 60 gari  staff  1920 Sep  7 08:06 ..
Gari’s MacBook:CODE gari$
```

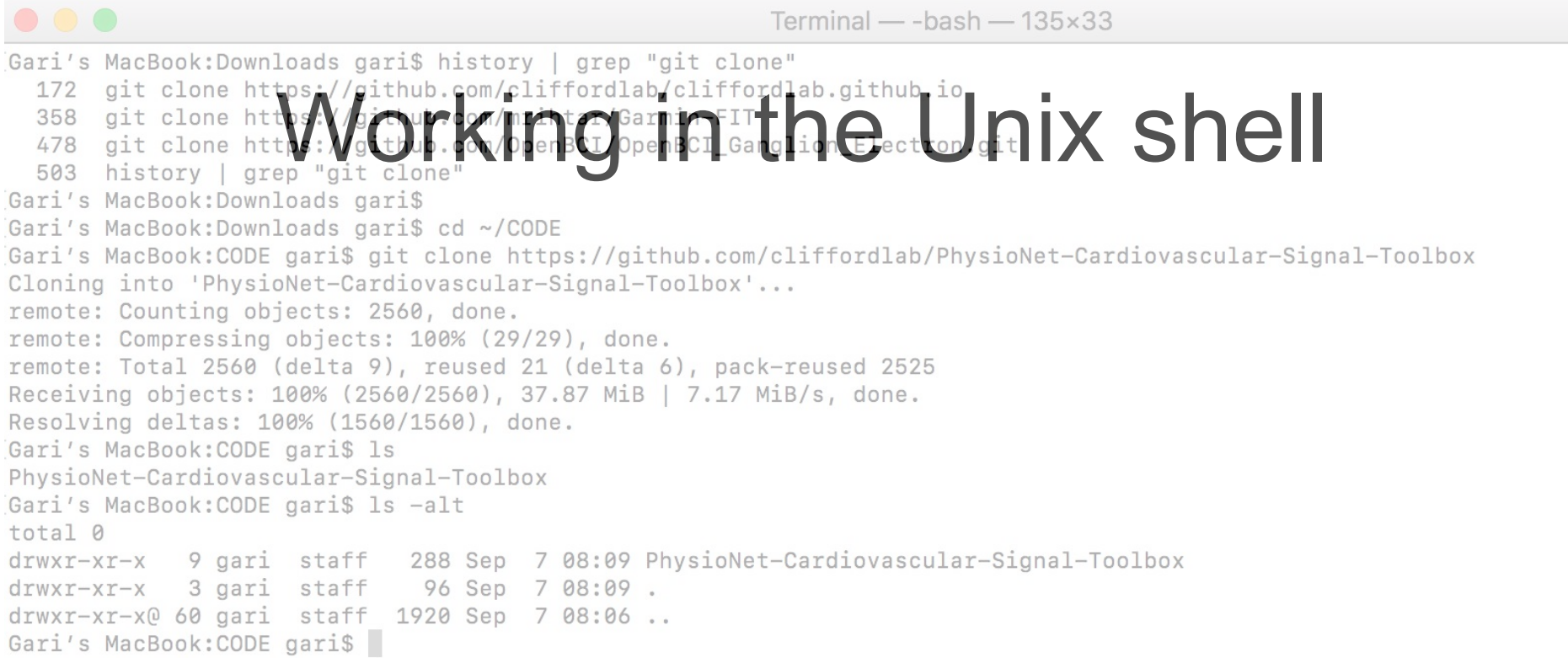
Forking Code

https://dont-be-afraid-to-commit.readthedocs.io/en/latest/git/commandlinegit.html

- Fork the code in the Github interface, then clone
- Allows you to merge later on ...



Working in the Unix shell



The Unix Shell: Summary of Basic Commands:

https://swcarpentry.github.io/shell-novice/reference/

Opening a terminal

- How to Use Terminal on a Mac
- Git for Windows
- How to Install Bash shell command-line tool on Windows 10
- Install and Use the Linux Bash Shell on Windows 10
- Using the Windows 10 Bash Shell
- Using a UNIX/Linux emulator (Cygwin) or Secure Shell (SSH) client (Putty)

How to perform a code review

- Essential to make code repeatable
- Check out the code onto another machine
- Check for hard-coded parameters / paths
- Use a global parameter file
- Review comments first as if it should be human readable
- Check for license and README explaining what the code should do
- Does it do what the README says and is there an example function with dummy data and a gold standard output?

- https://www.atlassian.com/company/events/summit-us/watch-sessions/2016/build-deploy/ten-tips-for-effective-code-review
- https://blog.ndepend.com/effective-team-code-reviews/

Matlab: Function Basics

Function Types and Scope
Function Input/Output
Help files

Function Types and Scope

- Primary Function
- Subfunction
- Nested Function
- Overloaded Function
- Private Function
- Anonymous Function

Primary Function

- Basic function type
- Requires first line to be function definition line

```
function [ out ] = example_function_name( in )
%EXAMPLE_FUNCTION_NAME You should always write a brief description
% of your function here, followed by the syntax.
%
%   [ out ] = example_function_name( in )
%       Description of this specific syntax.

% Function body goes here
out=in;
end
```

Primary Function

```
function [ avg, med ] = calc_stats( in )
%CALC_STATS Calculates the average and median of the input vector.
%
%   [ avg, med ] = calc_stats( in )
%       Calculates the average and median of in.

parse_input(in);
N = length(in);

avg = sum(in) / N; % Calculate average
in = sort(in);
if rem(in,2)==1 % Odd number of elements
    med = in( (N+1)/2 ); % median is middle value
else % Even number of elements
    med = (in(N/2) + in(N/2+1)) / 2; % median is average of 2 middle values
end

end
```

Subfunction

- Functions embedded within a primary function
- Appear after primary function's body

```
function [ avg, med ] = calc_stats( in )
%CALC_STATS Calculates the average and median of the input vector.
%
%   [ avg, med ] = calc_stats( in )
%       Calculates the average and median of in.

parse_input(in);
N = length(in);

...

function [] = parse_input(in)
    if isempty(in)
        error('Input is empty');
    end
    sz=size(in);
    if sz(1)~=1 && sz(2)~=1
        error('Input must be a row or column vector.');
    end

end
```

Nested Function

- Function within another function
- Inherits the workspace of the parent function
- Cannot be used inside program control statements (e.g., if, switch, try...)
- Can nest functions within functions

Nested Function

```
function [ dream_time ] = inception( time )
%INCEPTION Calculates the amount of dream time occured
%   during a given amount of real time
%
%   [ dream_time ] = inception ( time )
%       Converts real-time minutes to dream-time minutes

dream_time=time*12;
deeper;
    function deeper
        dream_time=dream_time*12;
        deeper;
        function deeper
            dream_time=dream_time*12;
            deeper;
            function deeper
                dream_time=dream_time*12;
            end
        end
    end
end
```

Nested Function

```
function [ avg, med ] = calc_stats( in )
%CALC_STATS Calculates the average and median of the input vector.
%
%   [ avg, med ] = calc_stats( in )
%       Calculates the average and median of in.

    parse_input(in);
    N = length(in);
    avg=calc_avg(N);
    med=calc_med(N);

    function [avg] = calc_avg(N)
        avg = sum(in) / N; % Calculate average
    end

    function [med] = calc_med(N)
        in = sort(in);
        if rem(in,2)==1 % Odd number of elements
            med = in( (N+1)/2 ); % median is middle value
        else % Even number of elements
            med = (in(N/2) + in(N/2+1)) / 2; % median is average of 2 middle values
        end
    end
end
```

Overloaded Functions

- When two functions must have different functionalities for different types of inputs
- Each function must go in a *class path*

- Example:
  - ~/home/me/@double/calc_average.m
  - ~/home/me/@int32/calc_average.m

Overloaded Functions

```
function [ avg ] = calc_average( in )
%CALC_STATS Calculates the average of the input vector.
%
%   [ avg, med ] = calc_stats( in )
%       Calculates the average and median of in.

    N = length(in);
    avg = sum(in) / N; % Calculate average
end
```

```
>> z=10*rand(1,5)

z =

    0.9754    2.7850    5.4688    9.5751    9.6489

>> calc_average(double(z))

ans =

    5.6906

>> calc_average(int32(z))

ans =

           6
```

```
function [ avg ] = calc_average( in )
%CALC_STATS Calculates the average of the input vector.
%
%   [ avg, med ] = calc_stats( in )
%       Calculates the average and median of in.

    N = length(in);
    in=double(in); % Cast as double
    avg = int32(sum(in) / N); % Calculate average, recast as int32
end
```

Private Function

- Located in a sub-folder named *private*
- Identical to primary function
- Only visible to **functions** in the parent folder
- Sub-folder *private* should NOT be in the MATLAB path

- Example:


```
function [ out ] = ninja_stats( in )
%NINJA_STATS Private function which calculates the
%   standard deviation of the input.
out = std(in);
end
```

Private Function


```
function [ out ] = has_access( in )
%HAS_ACCESS Parent function which has access to ninja stats
% Within ParentFolder
out=ninja_stats(in)
end
function [ out ] = no_access( in )
%HAS_ACCESS Parent function which does NOT have access to ninja stats
% Not in ParentFolder
out=ninja_stats(in)
end
```

Example Execution:

```
>> has_access(1:10);

out =

    3.0277

>> no_access(1:10);
??? Undefined function or method 'ninja_stats' for input arguments of
type 'double'.

Error in ==> no_access at 4
out=ninja_stats(in)

4    out=ninja_stats(in)
```

Anonymous Function

- Defined in-line
- fhandle = @(arg1,arg2) *expression*
- Useful for quick function handles to pass to other functions
  - @(x) x.^2
  - y=cellfun(@(x) x.^2, X);

```
>>
A = [2 3 4];     B = [5 6 7];
sumAxBy = @(x, y) (A*x + B*y);
sumAxBy(1,2)

ans =

    12    15    18
```

Function Input/Output

- nargin, nargout
  - Give the number of input/output arguments
  - Useful for argument checking

```
function out=testnarg(a,b)
% TESTNARG Array power
%
%   out = testnarg(a,b) calculates the element-wise bth power of a.
%       If unspecified, b is defaulted to 2.
if nargin<2
    b=2;
end
out=a.^b;
end
```

```
>> out=testnarg([3,4])

out =

     9    16
```

```
>> out=testnarg([3,4],3)

out =

    27    64
```

Variable Function Input

varargin

- All inputs combined into a cell array

```
function x=testvarargs(varargin)
% TESTVARARGS Combines variable inputs into a single array.
%
%   testvarargs(varargin)
x=[];
for k = 1:length(varargin)
    x = [x,varargin{k}];
end
end
```

```
>> x=testvarargs(1,[3,4],8,23)

x =

     1     3     4     8    23
```

```
K>> varargin

varargin =

    [1]    [1x2 double]    [8]    [23]
```

Variable Function Output

- varargout

```
function varargout=testvarargs2(x)
% TESTVARARGS2 Splits a matrix into column variables
%
%   [o1,o2,...] = testvarargs2(x)
for k = 1:nargout
    varargout{k}=x(:,k);
end
end
```

```
>> [o1,o2,o3]=testvarargs2([1,1,1;2,2,2;3,3,3])

o1 =          o2 =          o3 =

     1             1             1
     2             2             2
     3             3             3
```

Input parser

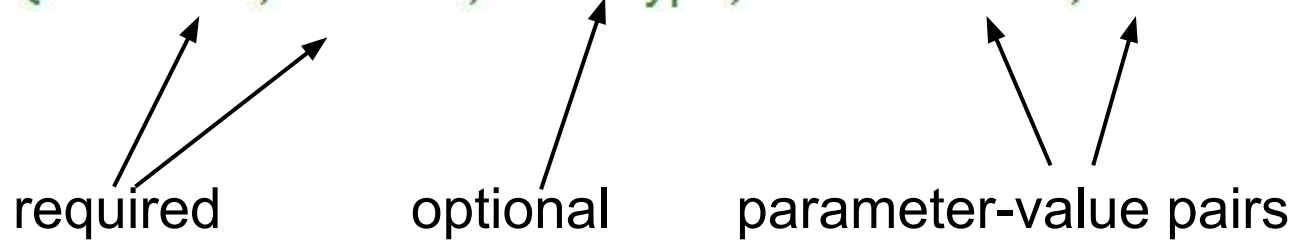
- MATLAB has an inputParser class which is useful for parsing input arguments... as you might have guessed.

- inpParserExample.m is an example MATLAB function which uses the main features of the input parser

Input parser

- inpParserExample.m inputs are:

```
function inpParserExample(filename, format, varargin)
% Function for demonstrating the use of the Input Parser
%   inpParserExample(filename, format, dataType, 'Parameter', Value...)
```

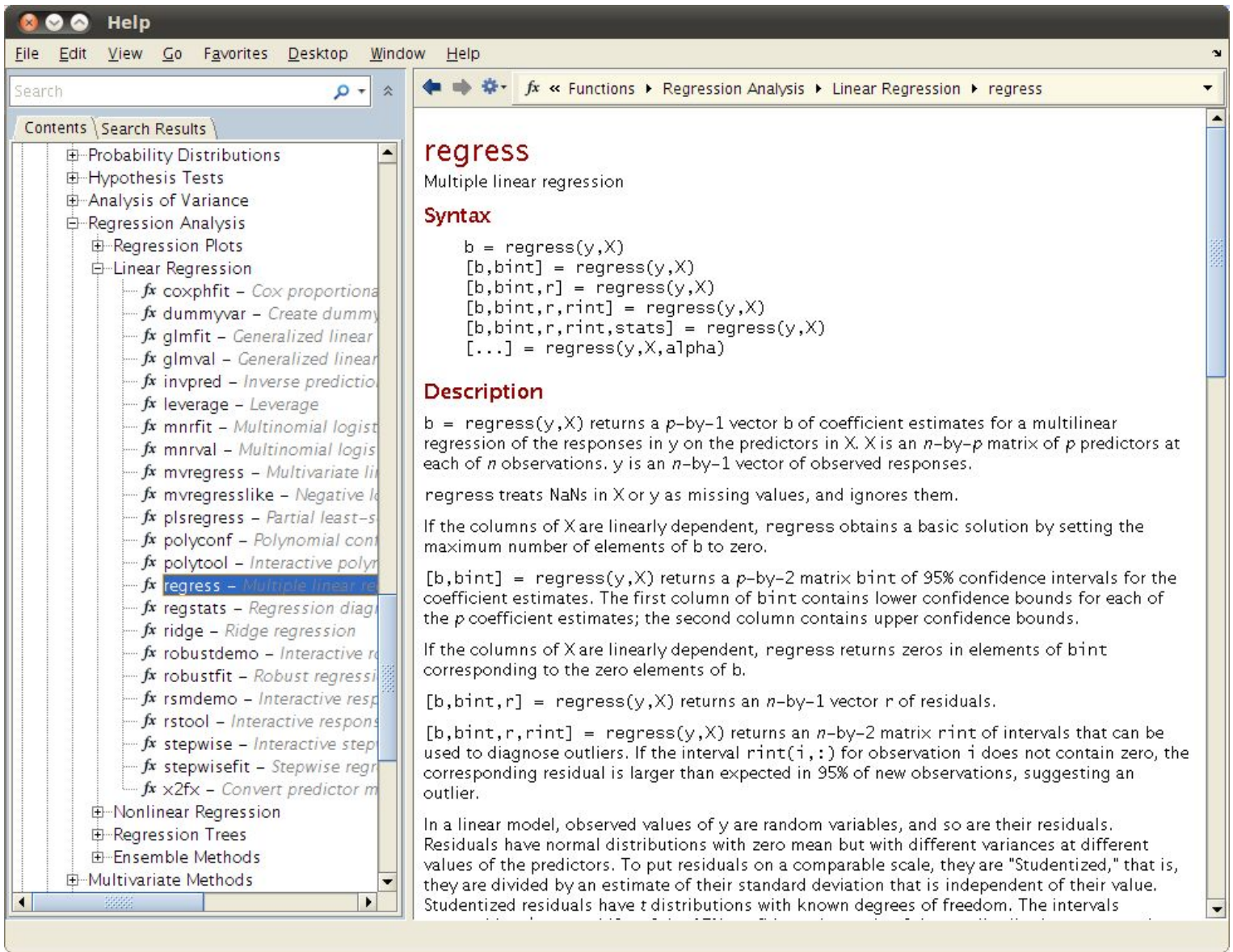


```
function inpParserExample(filename, format, varargin)
% Function for demonstrating the use of the Input Parser
%   inpParserExample(filename, format, dataType, 'Parameter', Value...)

% Create an instance of the inputParser class.
p = inputParser;

% Define inputs that one must pass on every call:
validStrings = {'txt', 'csv', 'xls', 'xlsx'};
p.addRequired('filename', @ischar);
p.addRequired('format', @(x)any(strcmp(x,validStrings)));

% Define inputs that default when not passed:
validDataTypes = {'cell', 'double', 'integer'};
p.addOptional('dataType', 'double', ...
   @(x)any(strcmpi(x,validDataTypes)));

% Define inputs passed in parameter/value format.
validHeaderLines = @(x)validateattributes(x, {'numeric'}, ...
   {'scalar', 'integer', 'positive', '>=', 0});
p.addParamValue('horizHeaderDim', 1, validHeaderLines);
p.addParamValue('vertHeaderDim', 0, validHeaderLines);

% Parse and validate all input arguments.
p.parse(filename, format, varargin{:});

% Show the value of all arguments.
fprintf('\n')
disp 'List of all arguments:'
disp(p.Results)

end
```

Help files

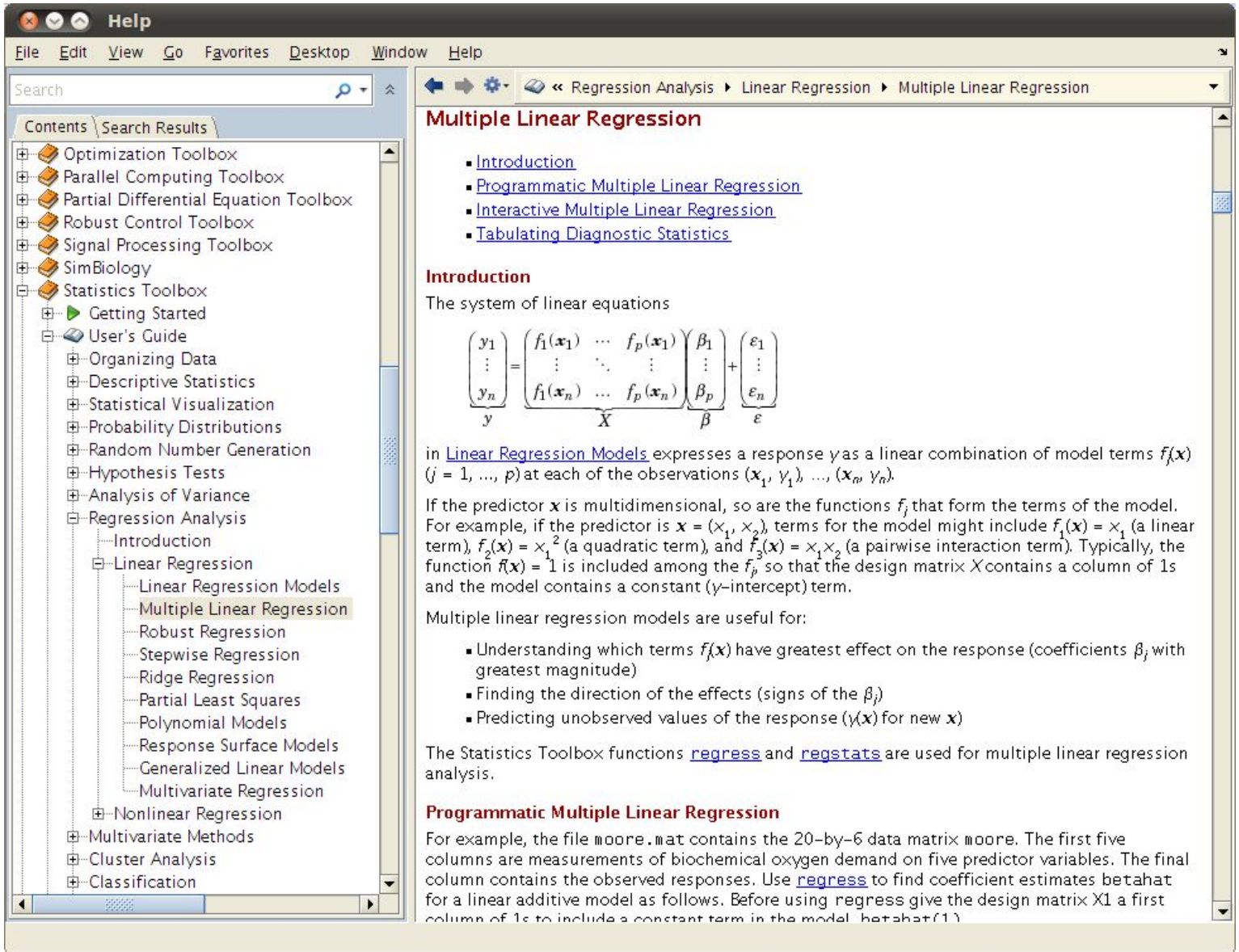
- MATLAB Help is excellent

```
>> doc regress
```



Help files

- Be sure to check the user's guide!



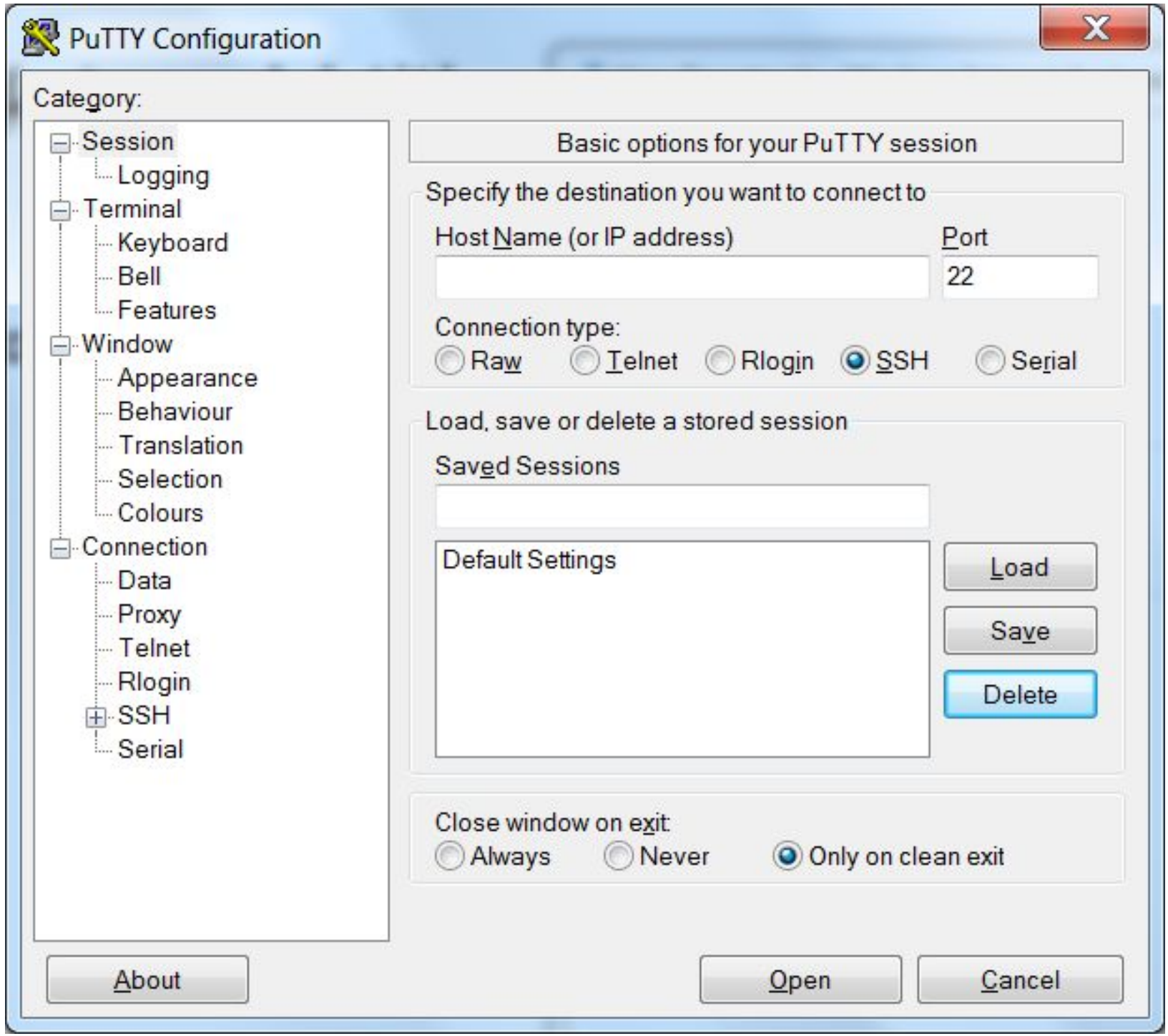
Remote and Parallel Computing

putty + SSH
Parallel Computing Toolbox (PCT)
Licensing
PCT Exercise
Graphical Processing Units

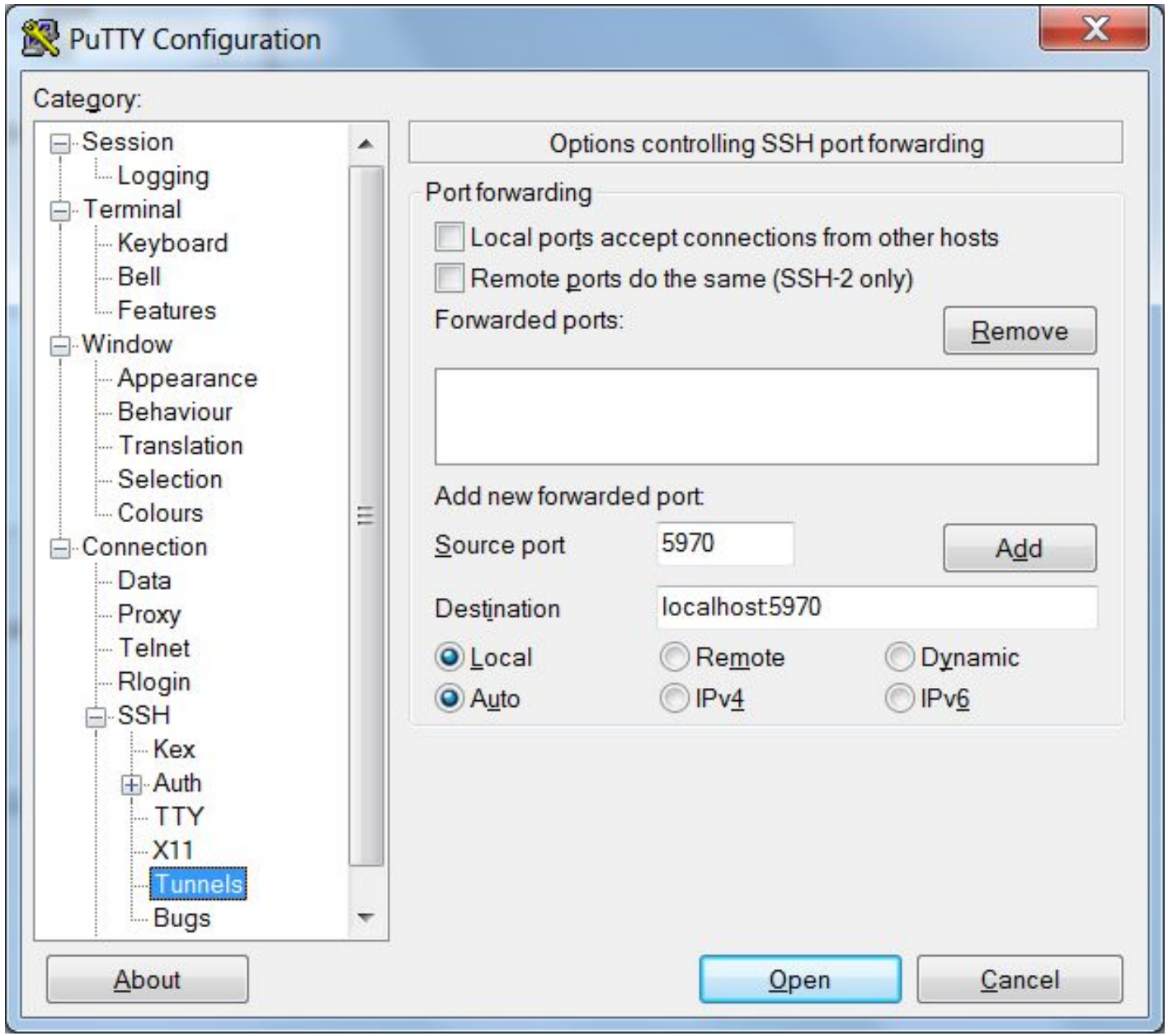
Download and run putty

- Google "putty" and download the .exe

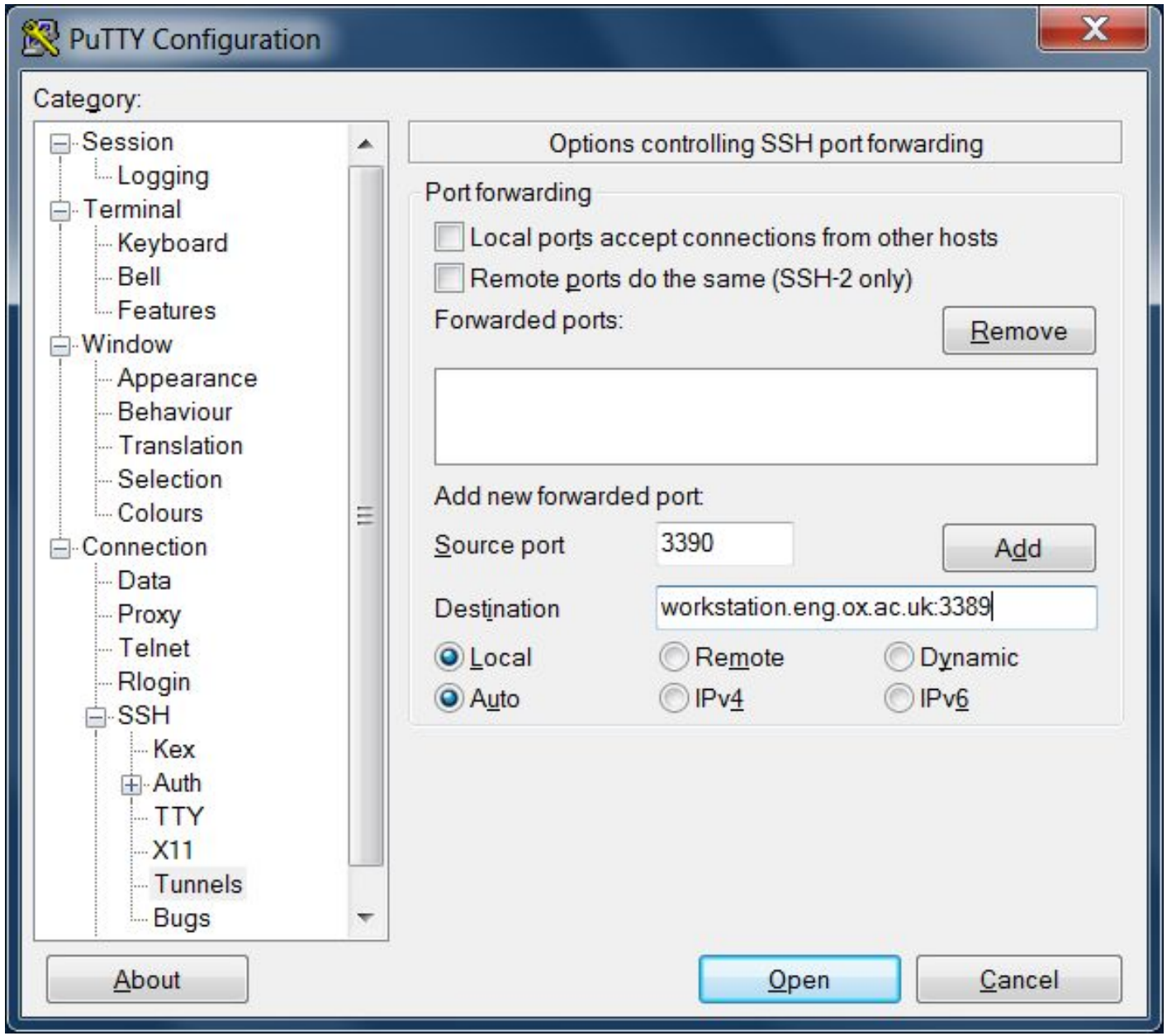
- Or type out this URL to get there directly:
- http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html

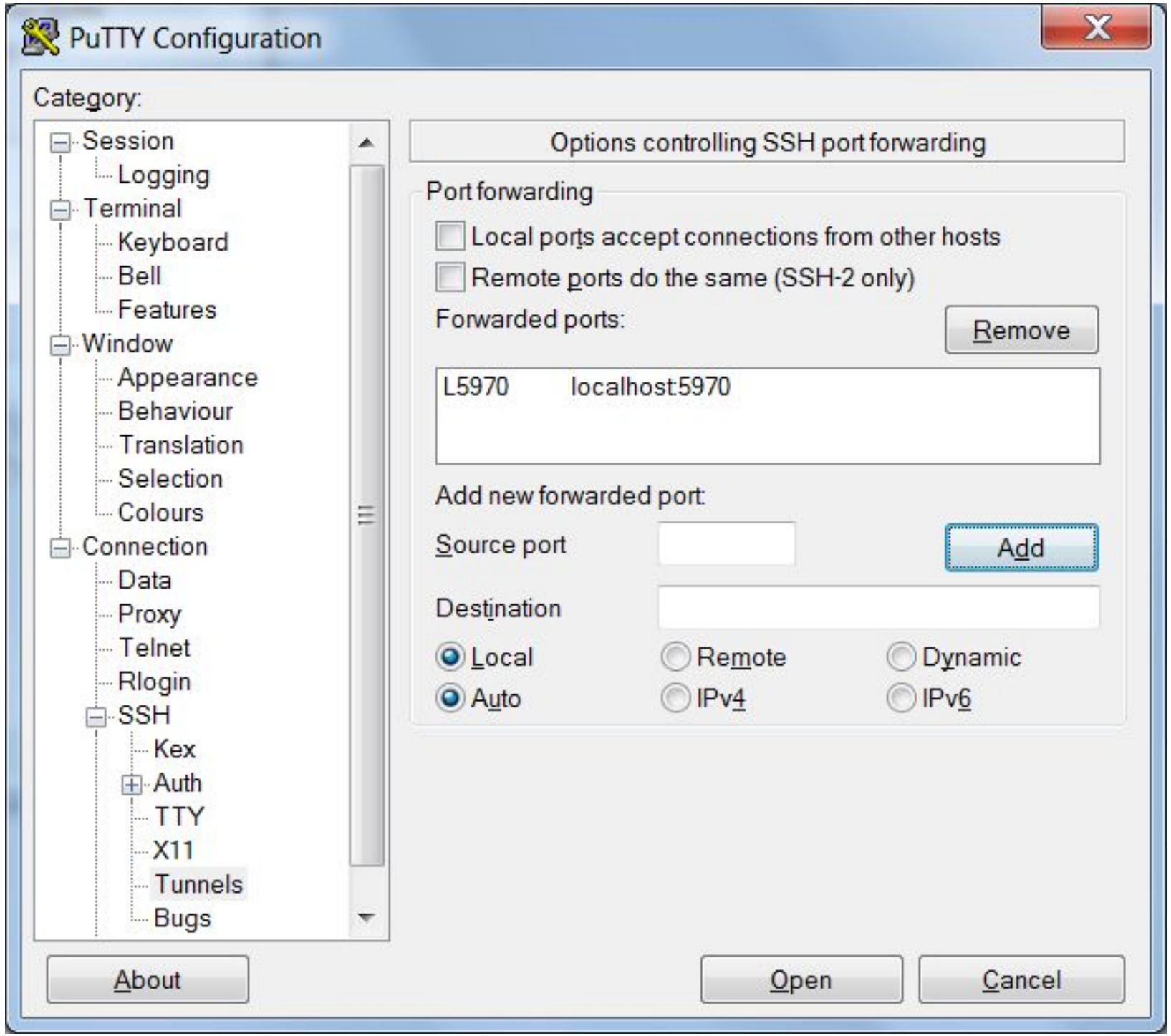

PuTTY Configuration
Category:
Session
Logging
Terminal
Keyboard
Bell
Features
Window
Appearance
Behaviour
Translation
Selection
Colours
Connection
Data
Proxy
Telnet
Rlogin
SSH
Serial
Basic options for your PuTTY session
Specify the destination you want to connect to
Host Name (or IP address)
Port
22
Connection type:
Raw
Telnet
Rlogin
SSH
Serial
Load, save or delete a stored session
Saved Sessions
Default Settings
Load
Save
Delete
Close window on exit:
Always
Never
Only on clean exit
About
Open
Cancel

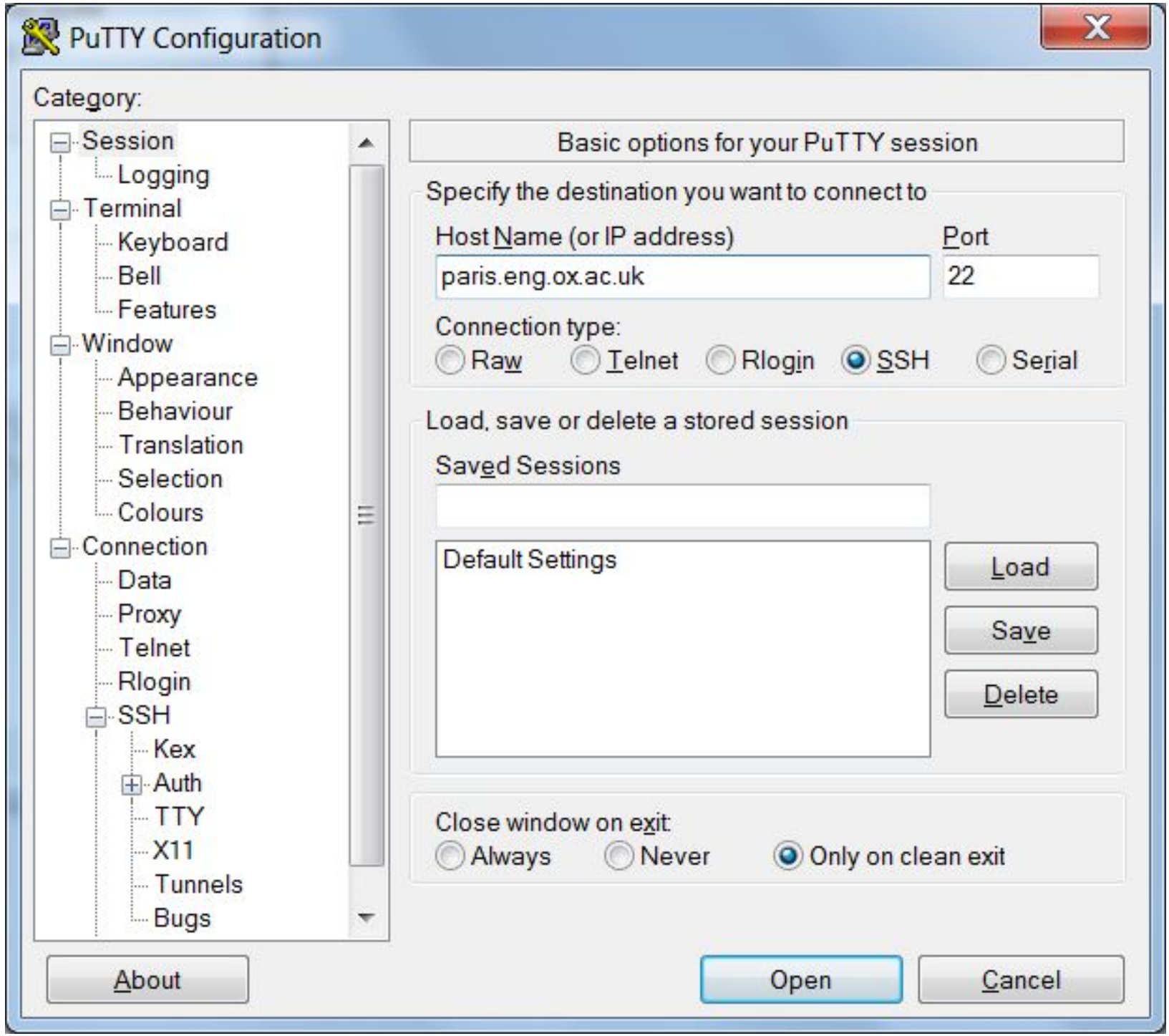
For Linux Users....

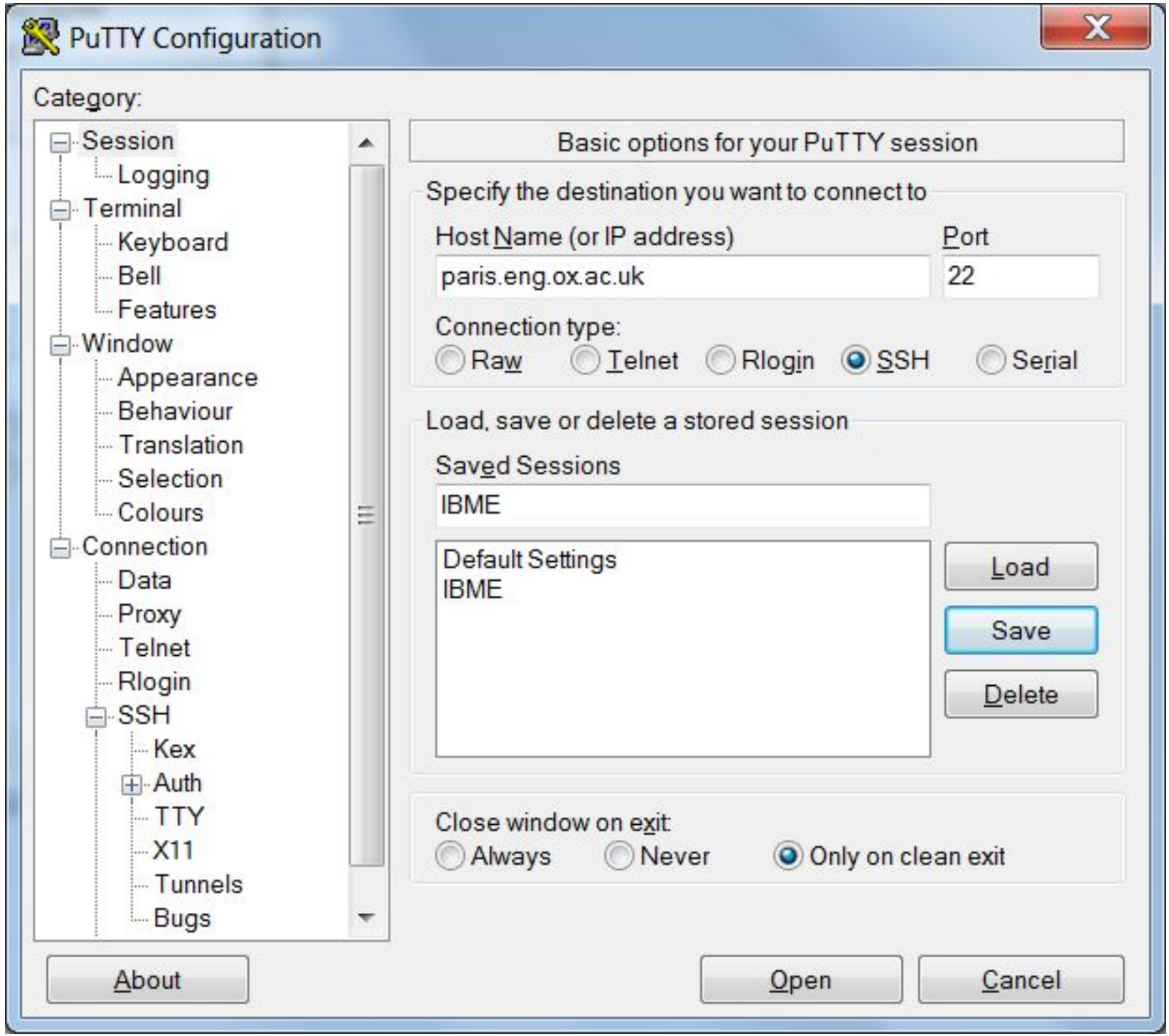


For Windows Users....
Replace "workstation"




PuTTY Configuration
Category:
Session
Logging
Terminal
Keyboard
Bell
Features
Window
Appearance
Behaviour
Translation
Selection
Colours
Connection
Data
Proxy
Telnet
Rlogin
SSH
Kex
Auth
TTY
X11
Tunnels
Bugs
Options controlling SSH port forwarding
Port forwarding
Local ports accept connections from other hosts
Remote ports do the same (SSH-2 only)
Forwarded ports:
Remove
L5970 localhost:5970
Add new forwarded port:
Source port
Add
Destination
Local
Remote
Dynamic
Auto
IPv4
IPv6
About
Open
Cancel


PuTTY Configuration
Category:
Session
Logging
Terminal
Keyboard
Bell
Features
Window
Appearance
Behaviour
Translation
Selection
Colours
Connection
Data
Proxy
Telnet
Rlogin
SSH
Kex
Auth
TTY
X11
Tunnels
Bugs
Basic options for your PuTTY session
Specify the destination you want to connect to
Host Name (or IP address)
Port
paris.eng.ox.ac.uk
22
Connection type:
Raw
Telnet
Rlogin
SSH
Serial
Load, save or delete a stored session
Saved Sessions
Default Settings
Load
Save
Delete
Close window on exit:
Always
Never
Only on clean exit
About
Open
Cancel


PuTTY Configuration
Category:
Session
Logging
Terminal
Keyboard
Bell
Features
Window
Appearance
Behaviour
Translation
Selection
Colours
Connection
Data
Proxy
Telnet
Rlogin
SSH
Kex
Auth
TTY
X11
Tunnels
Bugs
Basic options for your PuTTY session
Specify the destination you want to connect to
Host Name (or IP address)
Port
paris.eng.ox.ac.uk
22
Connection type:
Raw
Telnet
Rlogin
SSH
Serial
Load, save or delete a stored session
Saved Sessions
IBME
Default Settings
IBME
Load
Save
Delete
Close window on exit:
Always
Never
Only on clean exit
About
Open
Cancel

Now log in with your user credentials

For Windows users...

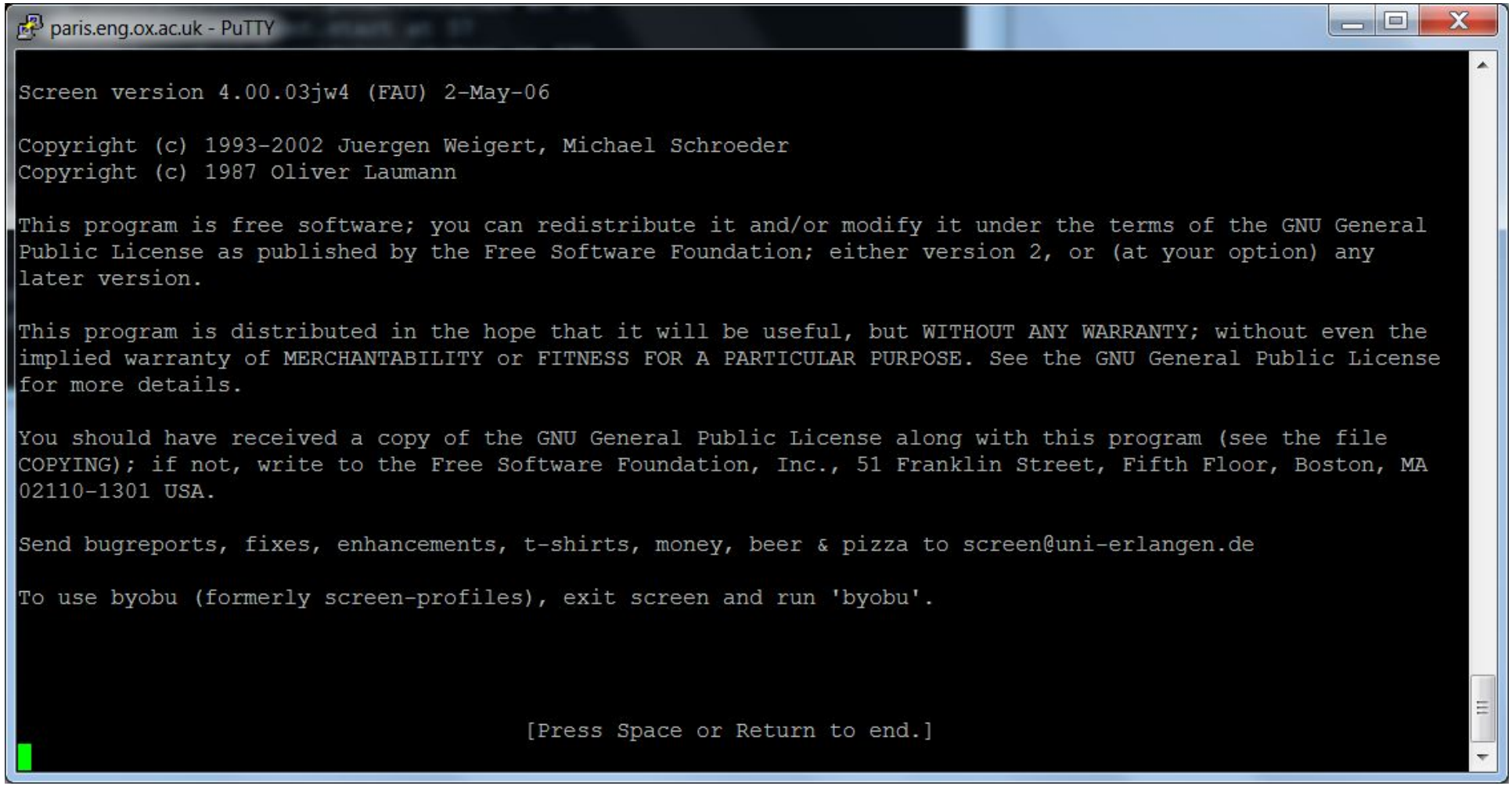
Linux users - connect to your computer

Done!

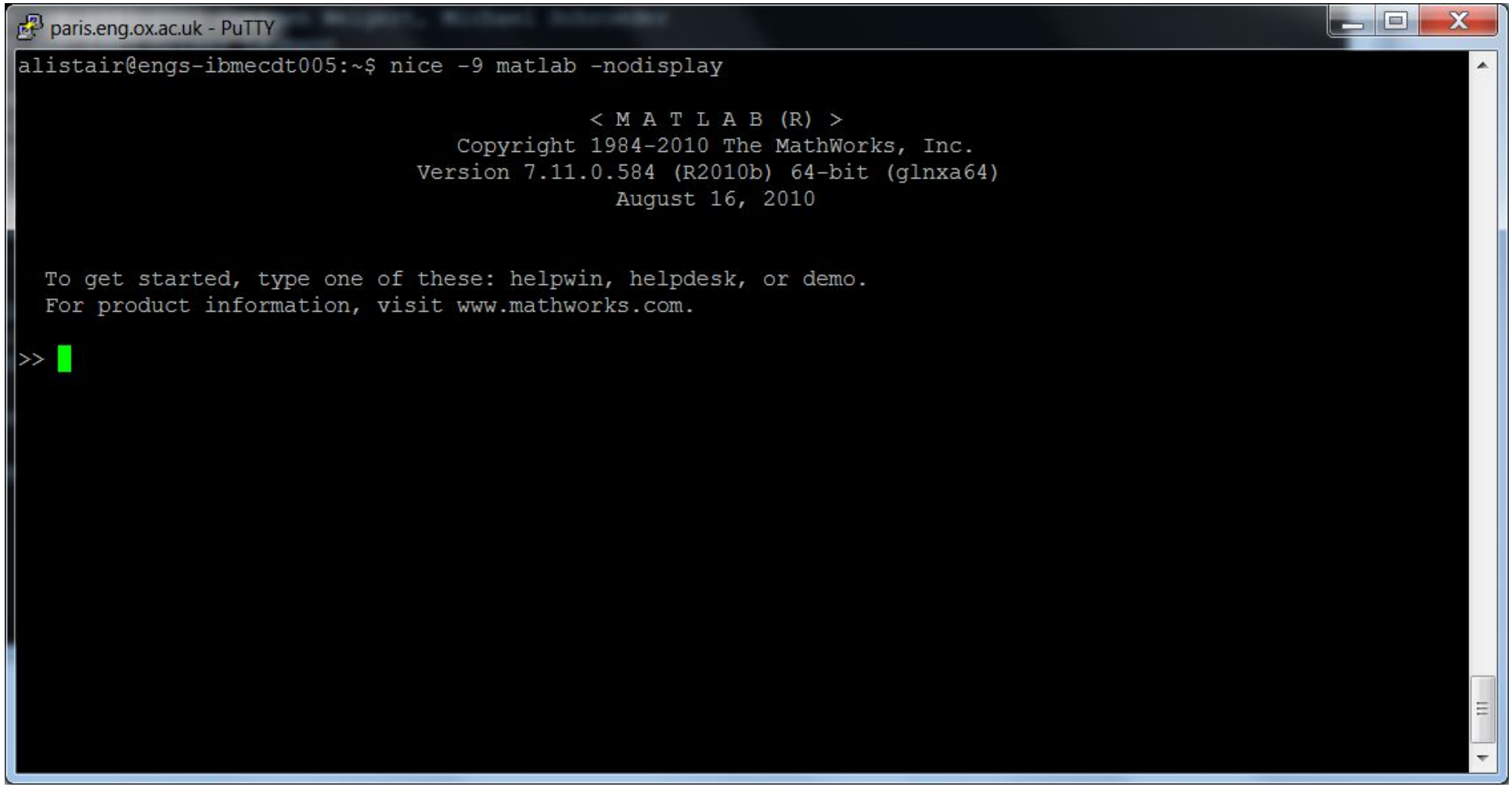

paris.eng.ox.ac.uk - PuTTY
login as: alistair
Using keyboard-interactive authentication.
Password:
Permissions on the password database may be too restrictive.
Last login: Wed Oct 5 08:58:35 2011 from engs-dopc17.eng.ox.ac.uk
alistair@paris:~> ssh engs-ibmecdt005
alistair@engs-ibmecdt005's password:
Linux engs-ibmecdt005 2.6.32-34-generic #77-Ubuntu SMP Tue Sep 13 19:39:17 UTC 2
011 x86_64 GNU/Linux
Ubuntu 10.04.3 LTS
Welcome to Ubuntu!
 * Documentation: https://help.ubuntu.com/
8 packages can be updated.
7 updates are security updates.
Last login: Wed Oct 5 09:00:38 2011 from paris.eng.ox.ac.uk
alistair@engs-ibmecdt005:~$

Running MATLAB...

```
alistair@engs-ibmecdt005:~$ screen
```



Running MATLAB...



Parallel Computing

- Some tasks lend themselves nicely to parallelization
- The trade off
  - Computation Overhead vs Computation Time
- Note that MATLAB already has parallelization!

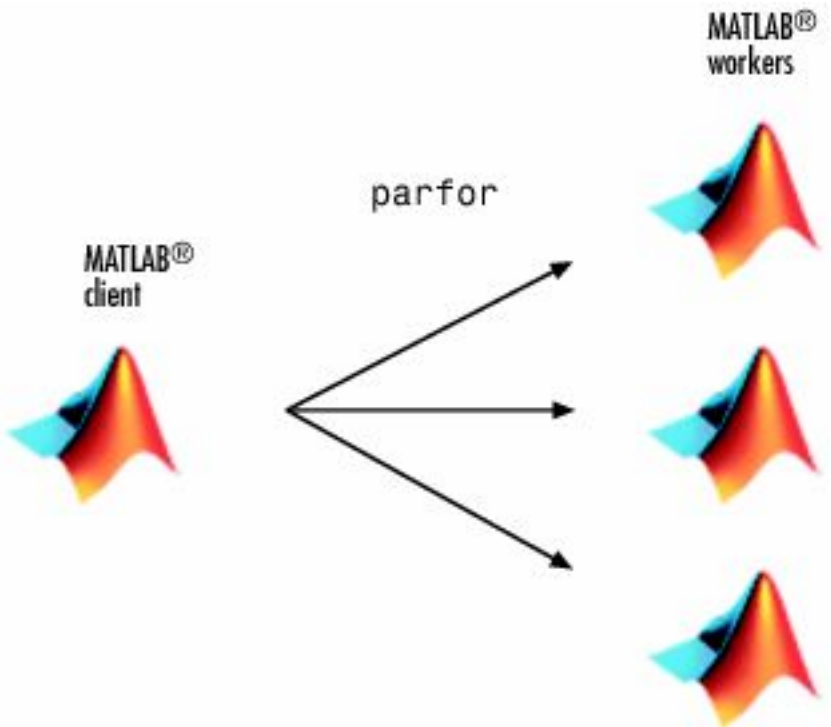
```
cpu_x = rand(1,100000000)*10*pi;
for k=1:20
tic;cpu_y = sin(cpu_x);
cpu(k) = toc;
end
mean(cpu)

ans =

    4.0359
```

```
cpu_x = rand(1,100000000)*10*pi;
for k=1:20
tic;cpu_y = sin(cpu_x);
cpu(k) = toc;
end
mean(cpu)

ans =

    0.8953
```

Parallel Computing Toolbox (PCT)

- You will likely work with *parfor* loops as opposed to *for* loops
- Calculations performed are identical, but **how** the calculations are performed changes
- The MATLAB client starts a pool of **workers**
- A general parallel script may look li

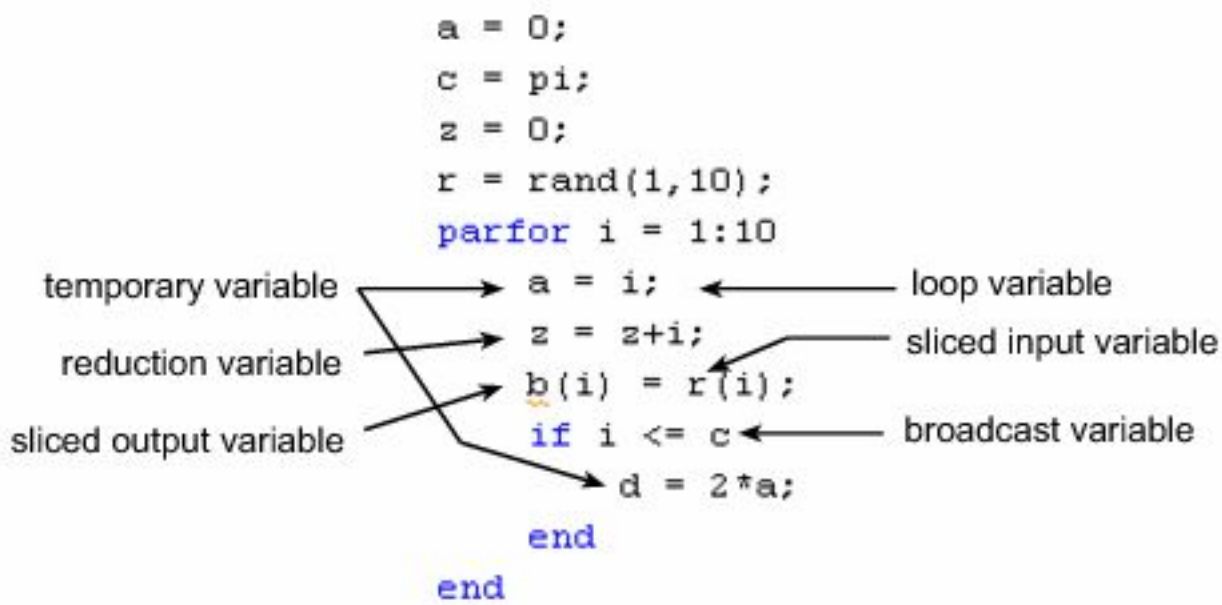
```
% PCT example
matlabpool 3; % Instantiate 3 workers
output=cell(20,1);
parfor k=1:20
    output{k}=someFunction(data(:,k));
end
matlabpool close; % Release workers
```



parfor

- *parfor* requires all iterations be independent of order and each other
- Variable assignment command line output and graphical displays (i.e. figures) will not display from workers
- *parfor* rules:
  - Don't use *clear* or *eval*
  - Don't load/save data in a *parfor* loop
  - Don't use *break*, *return*, *global*, or *persistent*
  - Use feval for function handles

parfor variables



Try to make sure any large arrays are sliced, as this will reduce overhead, i.e.:

- All of its first level indices are identical throughout the loop
- Exactly one of the first level indices is the loop variable

parfor indexing

Example workaround for ensuring a variable is indexed properly

```
% BAD code example
matlabpool; % Instantiate default worker #
output=cell(20,2);
parfor k=1:20
    output{k,1}=someFunction(data1(:,k));
    output{k,2}=someFunction(data2(:,k));
matlabpool close; % Release workers
```

In a PARFOR loop, variable 'output' is indexed in different ways, potentially causing dependencies between iterations.

```
% GOOD code example
matlabpool; % Instantiate default worker #
output1=cell(20,1);
output2=cell(20,1);
parfor k=1:20
    output1{k,1}=someFunction(data1(:,k));
    output2{k,1}=someFunction(data2(:,k));
end
output=[output1,output2];
matlabpool close; % Release workers
```

Example

- Open *PCT_importfile.m* and *PCT_importfilecsv.m*
- Ensure data.zip is extracted, and the data folder is in your path

```
% Load data
data=struct();
addpath('./data');
fn={'pat1_s0014lre.csv',...
    'pat2_s0015lre.csv',...
    'pat3_s0017lre.csv',...
    'pat5_s0025lre.csv',...
    'pat6_s0022lre.csv',...
    'pat6_s0027lre.csv',...
    'pat7_s0026lre.csv',...
    'pat8_s0028lre.csv'};

for k=1:length(fn)
    [rawData, name] = PCT_importfile(fn{k});
    data.(name)=rawData;
end
% Now data is a structure with all of our loaded data
```

PCT_peak_detector_serial.m

```
% PCT using peak detector

% Defaults
fs=1000; % 1000 kHz sampling frequency.
thresh=0.2; % Default peak detector threshold
ITER=20; % Number of iterations to repeat timing calculation

% Preallocate
normRunTime=zeros(ITER,1); % ~170s
parforRunTime=zeros(ITER,1); % ~48s

load PCT_peakdetectdata.mat % Load data

% Run peak detector serially across all leads (columns)
% Repeat a few times to have a decent median value
for rep=1:ITER
    tic;
    for k=2:size(dataCat,2)
        [hrv, R_t, R_amp, R_index, S_t, S_amp]  = rpeakdetect_final(dataCat(:,k),fs,thresh,0);
    end
    normRunTime(rep,1)=toc;
end
```

Parallelizing Peak Detector

- PCT_peak_detector.m
  - Runs the peak detector on 8 separate data measurements concatenated together
  - Doesn't keep the output (not interested in it for demo purposes)

```
Serial median run-time: 6.43s.
Parallel median run-time: 1.88s.
Parallel speed up ratio: 3.43:1.
```

PCT crashes

- A PCT crash will give you the line, but not the workspace

```
??? Error using ==> parallel_function at 598

Error in ==> train_alg at 187
parfor k=2:N+1

Error in ==> direct_model at 36
svmCELL=train_alg(raw,lbl,opt);
```

- This can make standard debugging annoying
- Usually best to remove "par" from *parfor* and debug normally

Miscellaneous PCT tips

Open:

```
if matlabpool('size')<=0
    matlabpool 8
end
```

Close:

```
if matlabpool('size')>0
    matlabpool close;
end
```

```
try
    matlabpool;
    % Insert code here...
catch me
    matlabpool close;
    throw me;
end
```

onCleanup:

```
function [ outputs ] = example_onCleanup( inputs )
    matlabpool;
    % c Executes upon function termination
    c = onCleanup(@()matlabpool('CLOSE'));

    outputs=num2str('hello');
end
```

Parallelizing Peak Detector exercise

- *Write a function which loads data from 8 separate .csv files (provided), runs the peak detector in parallel on each lead for each patient, and stores the result in a data type of your choosing.*
  - Do not concatenate data as was done in the previous demo!
  - Use a *parfor* loop, but do not initialize ***matlabpool*** as we do not have the proper licenses

Parallelizing Peak Detector

- PCT_peak_detector.m
  - Runs the peak detector on 8 separate data measurements concatenated together
  - Doesn't keep the output (not interested in it for demo purposes)

```
Serial median run-time: 6.43s.
Parallel median run-time: 1.88s.
Parallel speed up ratio: 3.43:1.
```

```
% Defaults/Pre-allocate
N_CORES=8; N_DATA=N_CORES; N_LEADS=15; N_OUTPUTS=6;
fs=1000; thresh=0.2; ITER=20;
spmdRunTime=zeros(1,ITER);

matlabpool 8;
for rep=1:ITER % Repeat for timing purposes
    tic;
    spmd (N_CORES) % across data
        % Variables defined in this loop are "composites", similar to cell
        % arrays, but with one index for each worker
        workerData = PCT_importfile(fn{labindex});
        workerResults = cell(N_OUTPUTS,N_LEADS);

        for k=1:N_LEADS % across leads
            [workerResults{1,k}, workerResults{2,k}, workerResults{3,k}, ...
                workerResults{4,k}, workerResults{5,k}, workerResults{6,k} ]...
                = rpeakdetect(workerData(:,k+1),fs,thresh,0);
        end
    end
    spmdRunTime(rep)=toc;
end

% Extract final results
finalResults=cell(N_OUTPUTS,N_LEADS,N_DATA);
for k=1:N_DATA
    finalResults(:,:,k)=workerResults{k};
end
```

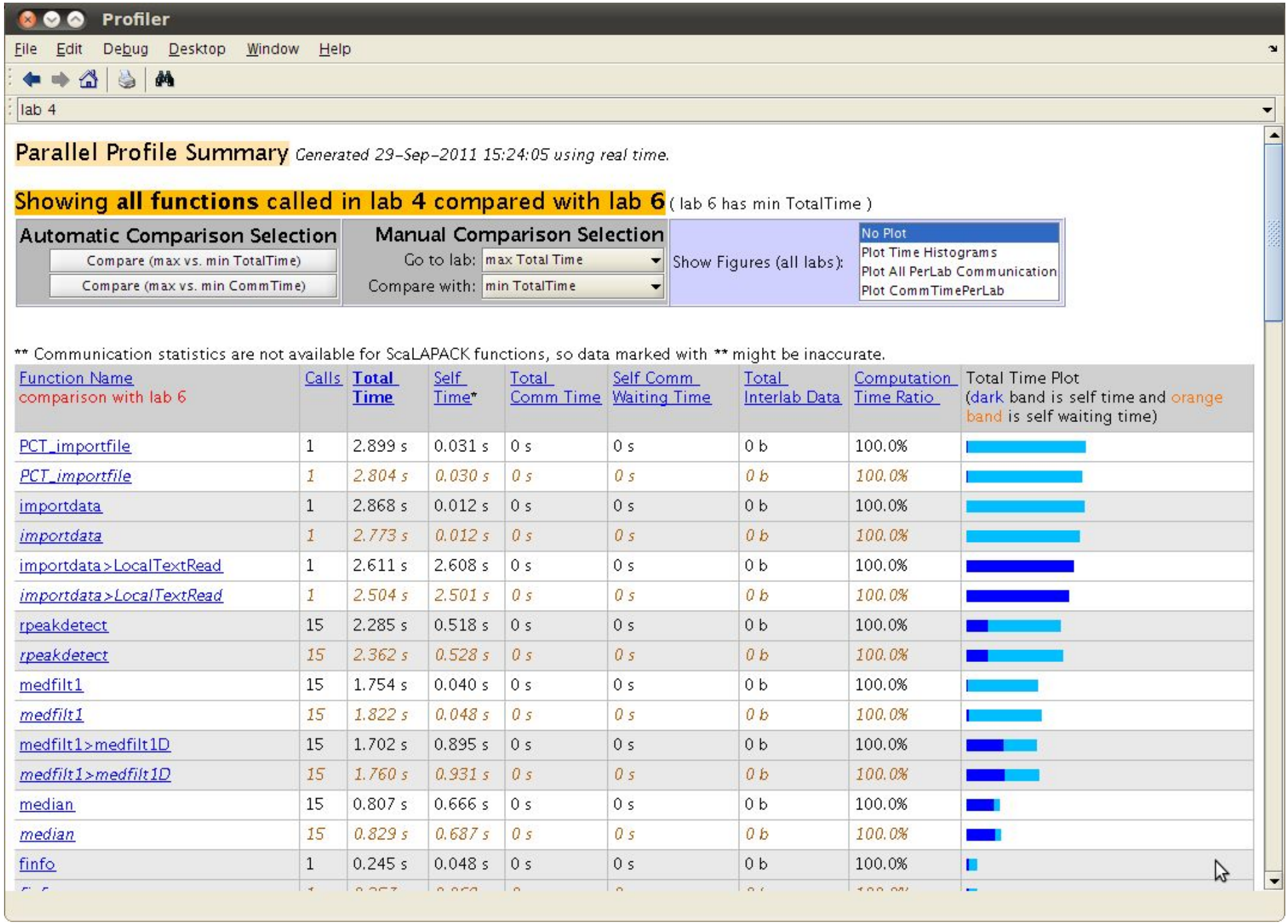
SPMD example

Loading data, performed with spmd...

```
SPMD median run-time: 5.01s.
```

All three variants (for, parfor, and spmd) are compared in PCT_comparison.m

Try it out (note: it assumes you have at least 4 workers available for use, change the defaults if necessary)



PCT_spmd_profiler.m

Parallelize EVERYTHING

parfor - simple, effective, and a good way to get started

spmd - complicated, but useful, allows for communication between cores

batch/task - multiple independent programs, very advanced

Further reading

- As always, the MATLAB help is invaluable

- Note that there are also demos available! At the MATLAB prompt:

```
demo 'toolbox' 'parallel computing'
```

Licensing

- Using MATLAB in the lab (or on campus/after VPN) uses the network license

```
>> license('inuse')
matlab
>> matlabpool 8;
Starting matlabpool using the 'local' configuration ...
>> license('inuse')
distrib_computing_toolbox
matlab
>> matlabpool close;
Sending a stop signal to all the labs ... stopped.
>> license('inuse')
distrib_computing_toolbox
matlab
```

Licensing

- MATLAB licenses are released:
  - After closing MATLAB
  - After a period of idle time elapses

```
>> license('inuse')
matlab
>> matlabpool 8;
Starting matlabpool using the 'local' configuration ...
>> license('inuse')
distrib_computing_toolbox
matlab
>> matlabpool close;
Sending a stop signal to all the labs ... stopped.
>> license('inuse')
distrib_computing_toolbox
matlab
```

Licensing

- Try to release your licenses!
- They are not an infinite resource ...

```
Users of Distrib_Computing_Toolbox:  (Total of 20 licenses issued;  Total of 5 licenses in use)

  "Distrib_Computing_Toolbox" v26, vendor: MLM
  floating license

    alistair engs-ibmecdt005 /dev/tty (v24) (flexlm.nsms.ox.ac.uk/1709 53321), start Wed 9/28 12:55
    hamid hbapc1 /dev/pts/1 (v24) (flexlm.nsms.ox.ac.uk/1709 62006), start Wed 9/28 11:02
    rosser huckleberry-hound.maths.ox.ac.uk /dev/pts/1 (v25) (flexlm.nsms.ox.ac.uk/1709 65508), start Wed 9/28 11:42
    scat3893 engs-018688 engs-018688 (v22) (flexlm.nsms.ox.ac.uk/1709 22160), start Wed 9/28 11:00
    yannis elbow /dev/pts/8 (v21) (flexlm.nsms.ox.ac.uk/1709 57911), start Wed 9/28 11:29
```

./lmutil lmstat -a -c /opt/MATLAB/R2010b/licenses/network.lic
More detail in speaker notes.

Licensing

- Try to release your licenses!
- Here is an example of how this can be accomplished:

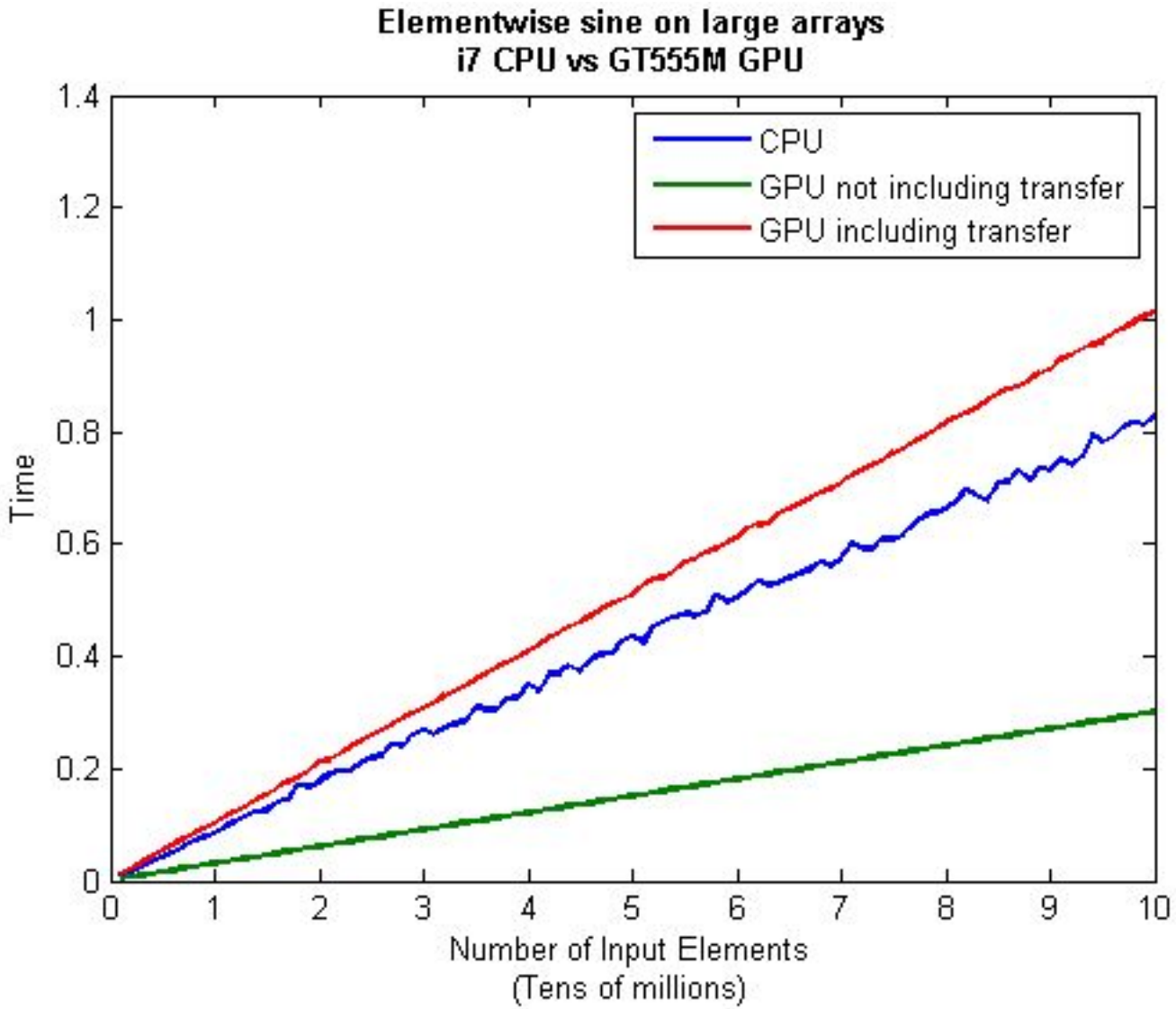
```
diary(['commandWindow-' datestr(now,1) '.txt']);
load data;
output=cell(100,1);
matlabpool 8;
parfor k=1:100
  output{k}=doWork(data,k);
end
matlabpool close;
save('output.mat','output');
diary off;
exit;
```

Graphics Processing Unit (GPU)

- GPUs can also be used for parallel processing
- GPUs have much more cores (e.g 512) but less processing power per core

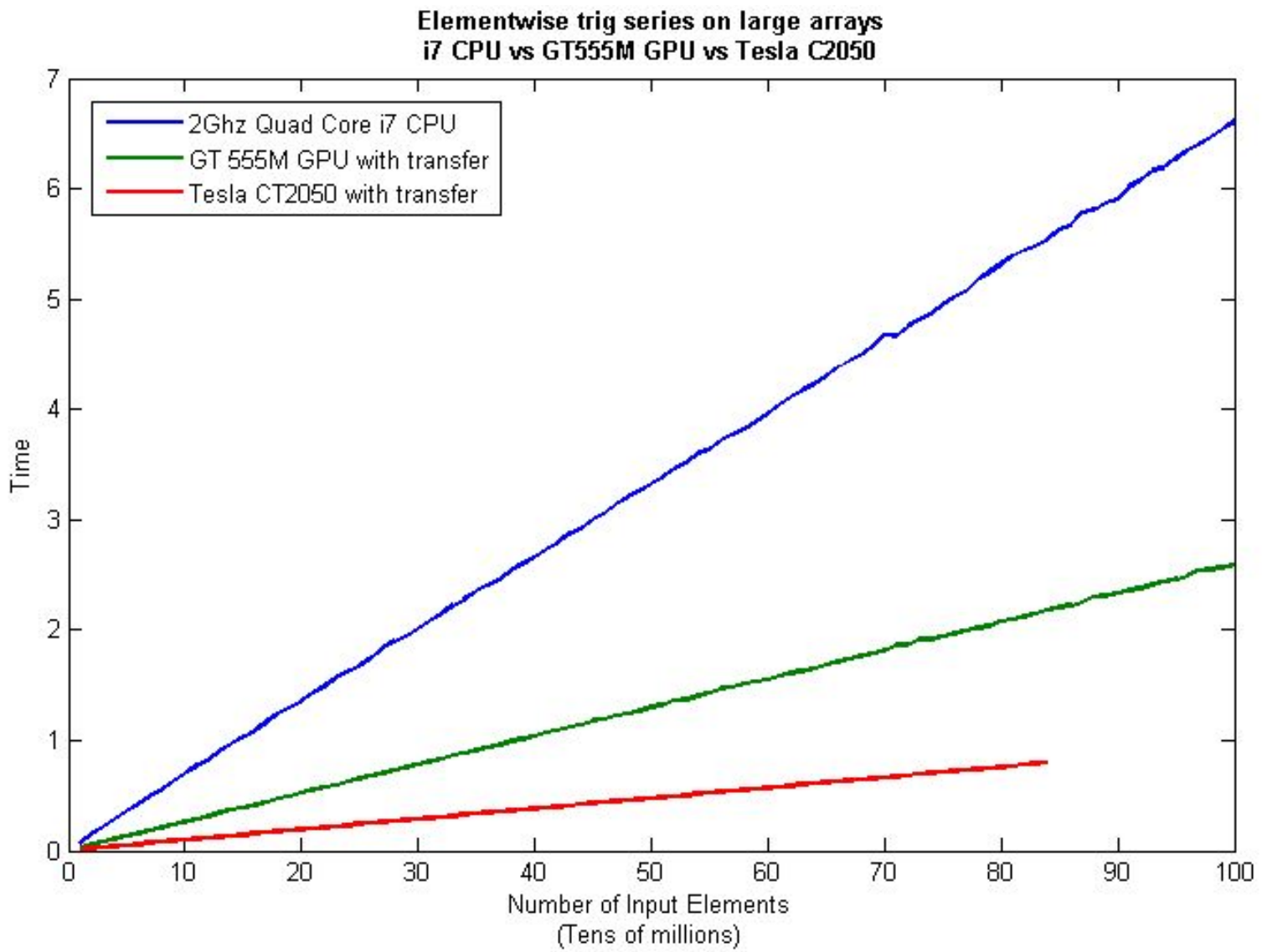
GPU speed-up

```
out = sin(t);
```



GPU speed-up

```
out = 2*sin(t)-sin(2*t)+2/3*sin(3*t)-1/2*sin(4*t)+...
   2/5*sin(5*t)-1/3*sin(6*t)+2/7*sin(7*t);
```



Matlab Executables
MEX

Basics
Simple example
LIBSVM example

MATLAB Executable (MEX)

- Allow MATLAB to run external libraries as if they were .m functions

- Use if...
  - There exists Fortran, C, or C++ code you want to use in MATLAB
  - You'd like to speed up computation

MEX basics

Let's assume you don't want to code in C, and simply have pre-coded source mex files you want to compile

3 files needed for using MEX

source MEX	C, C++, or Fortran source code
binary MEX	Subroutine executed by MATLAB
mex build script	Builds binary from source

MEX Requirements

- You will need a compiler (3rd party software)
- For Windows 64-bit:
  - Microsoft Visual C++ 2010 Express
  - Microsoft Windows SDK 7.1
- For Linux 64-bit:
  - GNU gcc/g++ 4.3 or higher
- For OS X 64-bit:
  - Apple XCode 4.0 (Snow Leopard) or 4.1 (Lion)

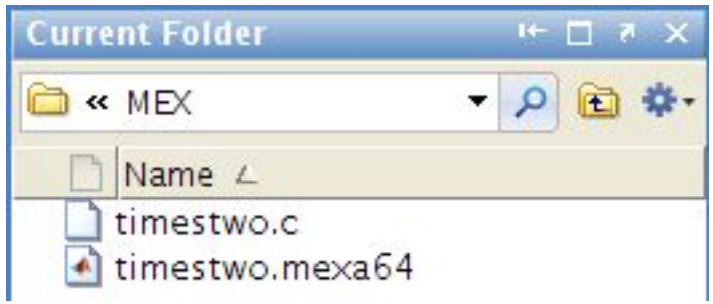
Select Compiler

- Make sure your compiler is using the right language (C, C++, or Fortran)
- We'll be using C
  - `mex -setup`
- Check your compiler's info:
  - `cc = mex.getCompilerConfigurations`

Easy example

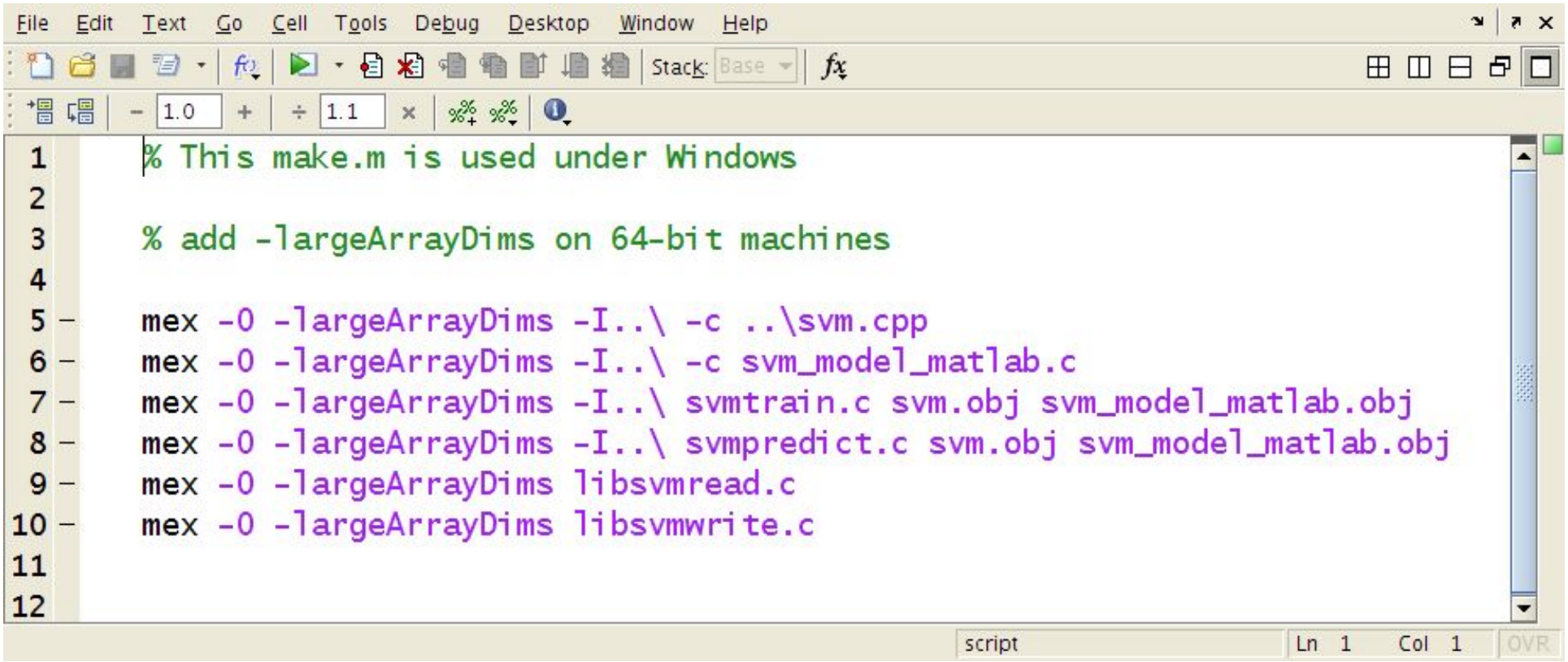
```
>> cd('/home/alistair/code/MATLAB_Course/MEX/')
copyfile([matlabroot '\extern\examples\refbook\timestwo.c']);
```

```
>> mex timestwo.c

Warning: You are using gcc version "4.4.3-4ubuntu5)".  The version
         currently supported with MEX is "4.3.4".
         For a list of currently supported compilers see:
         http://www.mathworks.com/support/compilers/current_release/
```



LIBSVM Practical Example

- LIBSVM is a C library for SVM classification/regression
- We are going to compile it in MATLAB
- They have already created a make.m file, so run it!



Source Code Control

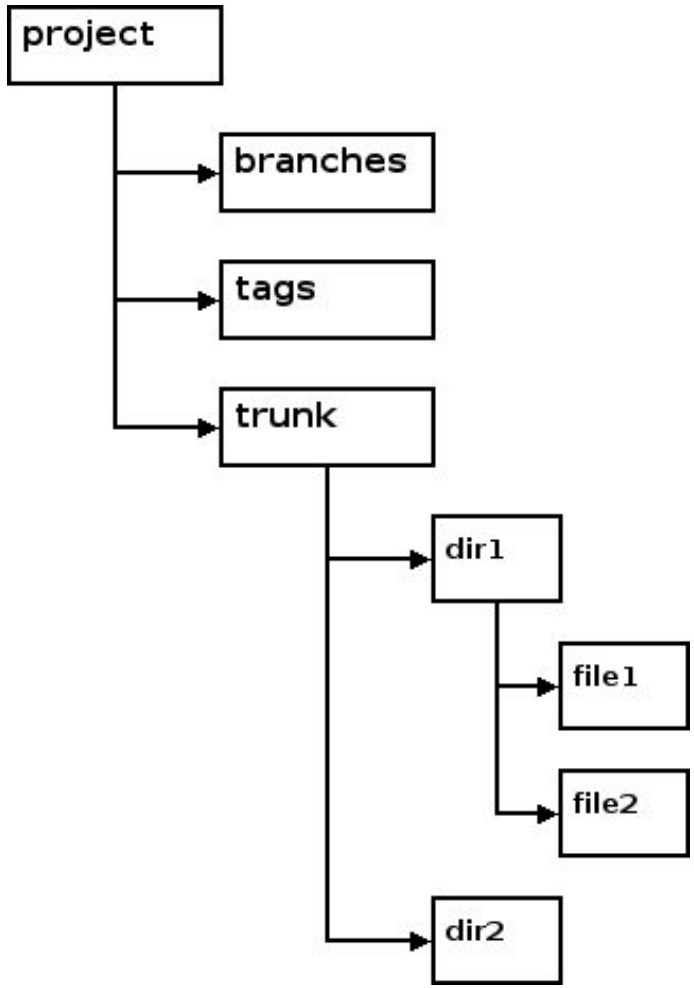
Subversion Control
Git

Subversion control

- Backup and track changes in your code ...
- Store all code related to a project in a "repository"
- Can share repositories, or keep it for your own work
- Clients
  - SmartSVN, TortoiseSVN, RabbitVCS ...
- Each have their own "How To" and getting started tutorials

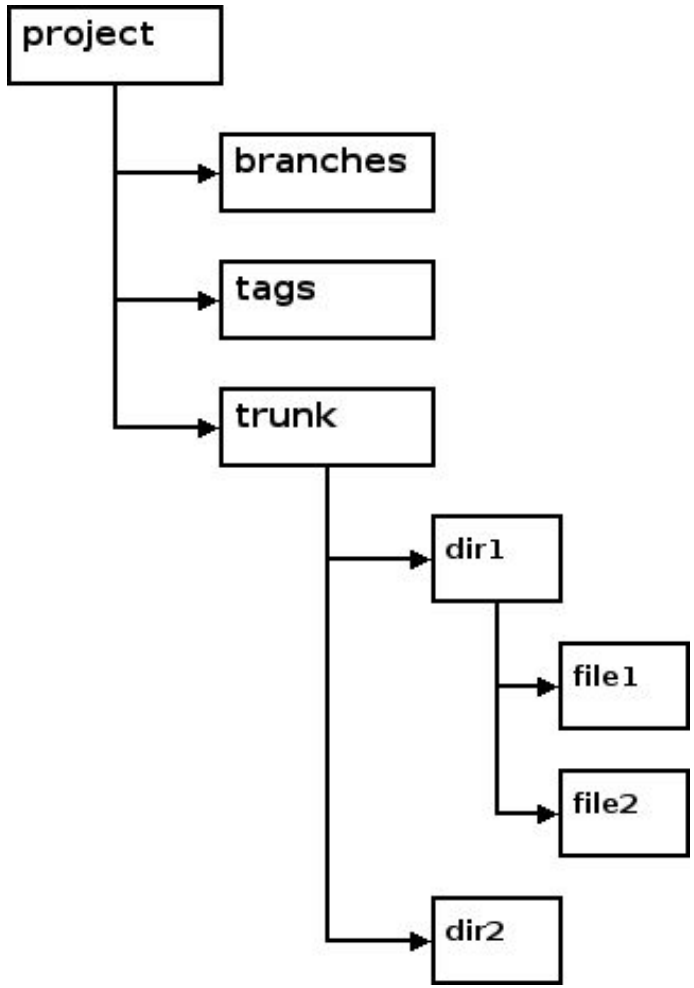
SVN - Example

`svn co` https://url.whatever.edu/svn/ProjectName/



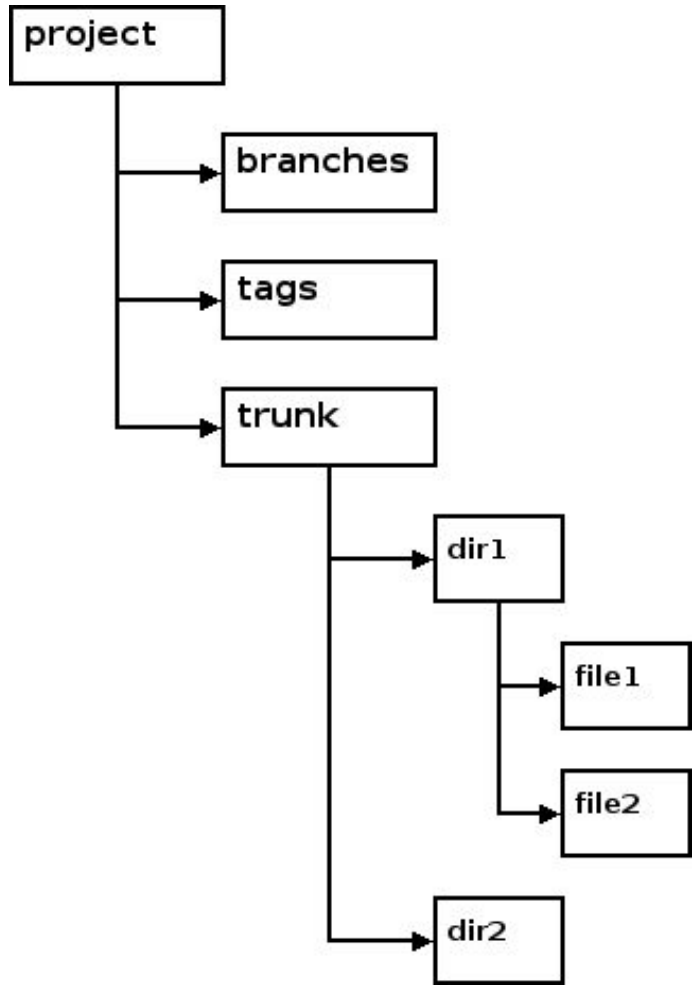
SVN - Example

`svn add file_or_dir_name`



SVN - Example

```
svn commit -m "Saving recent changes"
```



Google Code + Git

- Because there is not enough google in life, you can also do source code control using google code

1. http://code.google.com/hosting/
2. Click "Create a new project" 

3. Fill in the details (Project Name, etc..)
4. Choose "Git" as the subversion system
5. Repository created!

Working with gcode/git

- Add the following to your .netrc file (essentially a global configuration file)
  - machine code.google.com login [google e-mail] password [generated googlecode.com password]
- Clone the repository locally
  - git clone https://code.google.com/p/genetic-algorithm-feature-selection/
- Work with git as you normally would!
    - git add *.m
    - git commit -a
    - git push --all
  - A good cheat sheet:
  - https://rtime.felk.cvut.cz/hw/images/thumb/1/18/Git-cheat-sheet-medium.png/994px-Git-cheat-sheet-medium.png

Finally it's over..