Transparent Quantitative Musculoskeletal Imaging

Serena Bonaretti
https://sbonaretti.github.io/

Outline and Disclosures

1. Why transparent research?
2. How to conduct transparent research?
3. Example of transparent QMSKI: pyKNEEr

- I will show *some* of the tools for transparent research, chosen among the most commonly used
- I do not have conflicts of interest

What are openness and reproducibility?

- **Open science** refers to the free availability of data, software, and methods developed by researchers with the aim to share knowledge and tools to professionals and citizens (Woelfle 2011)

- **Reproducibility** is the ability of researchers to duplicate the results of a previous study using the same data, software, and methods used by the original authors (Bollen 2015)
  - **Replicability** is the recreation of same results using new data but the same experimental design (Gorgolewski)

Note: In some fields the two definitions are inverted

Replication crisis

- Discovery of personalized cancer treatment at Duke University


The NEW ENGLAND JOURNAL of MEDICINE

This article has been retracted.

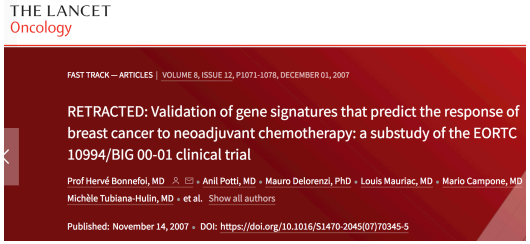
A correction has been published ↑

ORIGINAL ARTICLE

A Genomic Strategy to Refine Prognosis in Early-Stage Non–Small-Cell Lung Cancer

Anil Potti, M.D., Sayan Mukherjee, Ph.D., Rebecca Petersen, M.D., Holly K. Dressman, Ph.D., Andrea Bild, Ph.D., Jason Koontz, M.D., Robert Kratzke, M.D., Mark A. Watson, M.D., Ph.D., Michael Kelley, M.D., Geoffrey S. Ginsburg, M.D., Ph.D., Mike West, Ph.D., David H. Harpole, Jr., M.D., et al.

([Potti 2006](#))


THE LANCET Oncology

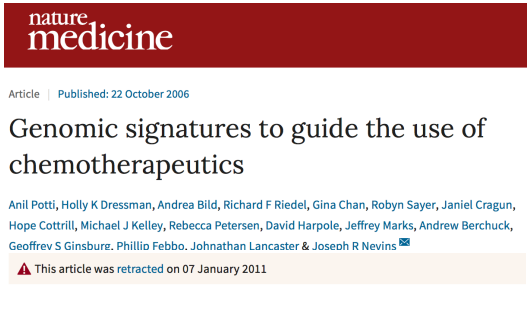
FAST TRACK — ARTICLES | VOLUME 8, ISSUE 12, P1071-1078, DECEMBER 01, 2007

RETRACTED: Validation of gene signatures that predict the response of breast cancer to neoadjuvant chemotherapy: a substudy of the EORTC 10994/BIG 00-01 clinical trial

Prof Hervé Bonnefoi, MD • Anil Potti, MD • Mauro Delorenzi, PhD • Louis Mauriac, MD • Mario Campone, MD • Michèle Tubiana-Hulin, MD • et al. Show all authors

Published: November 14, 2007 • DOI: https://doi.org/10.1016/S1470-2045(07)70345-5

([Bonnefoi 2007](#))


nature medicine

Article | Published: 22 October 2006

Genomic signatures to guide the use of chemotherapeutics

Anil Potti, Holly K Dressman, Andrea Bild, Richard F Riedel, Gina Chan, Robyn Sayer, Janiel Cragun, Hope Cottrill, Michael J Kelley, Rebecca Petersen, David Harpole, Jeffrey Marks, Andrew Berchuck, Geoffrey S Ginsburg, Phillip Febbo, Johnathan Lancaster & Joseph R Nevins

⚠ This article was retracted on 07 January 2011

([Potti 2006](#))


Journal of Clinical Oncology®

An American Society of Clinical Oncology Journal

Pharmacogenomic Strategies Provide a Rational Approach to the Treatment of Cisplatin-Resistant Patients With Advanced Cancer

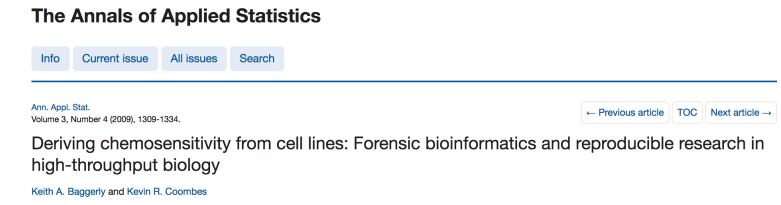
David S. Hsu, Bala S. Balakumaran, Chaitanya R. Acharya, Vanja Vlahovic, Kelli S. Walters, Katherine Garman, Carey Anders, Richard F. Riedel, Johnathan Lancaster, David Harpole, Holly K. Dressman, Joseph R. Nevins, Phillip G. Febbo, Anil Potti

From the Division of Medical Oncology, Department of Medicine; Institute for Genome Sciences and Policy; Department of Surgery, Duke University, Durham, NC; and the Division of Gynecologic Oncology, H. Lee Moffitt Cancer Center, Tampa, FL

Show Less

A retraction has been published.

([Hsu 2007](#))


The Annals of Applied Statistics

Info | Current issue | All issues | Search

Ann. Appl. Stat.
Volume 3, Number 4 (2009), 1309-1334.

← Previous article | TOC | Next article →

Deriving chemosensitivity from cell lines: Forensic bioinformatics and reproducible research in high-throughput biology

Keith A. Baggerly and Kevin R. Coombes

([Baggerly 2009](#))

V. Stodden[1]: "None of that was picked up in peer-review. Nobody looks under the covers that deeply in peer-review. They were able to read the paper. This kind of thing is in the code and in the data itself"

[1] https://www.youtube.com/watch?v=dF1-nkqwmjI from min 13:30, and https://www.youtube.com/watch?v=eV9dcAGaVU8

Computational replicability crisis

(Collberg 2015a) (Collberg 2015b)

- 508 papers from 8 conferences and 5 journals with a practical orientation
- Team of undergraduate students, graduate students, and postdocs
- They could replicate algorithms of 226 papers (44%)

Why is it so difficult to replicate studies?

- Papers must be concise (Vandewalle 2012)
  - Limited amount of words, figures, and tables
  - Authors have to select methods and parameters to present
- Data and software are in the supplementary material
  - Not in the paper body (Vandewalle 2012)
- Until a few years ago there were no permanent, large, and free repositories for data and software with DOI
  - Personal repositories often get deleted (Gil 2016)
- "Publish or perish" might favor quantity over quality and scientific bias (results have to be good) (Fanelli 2010)

Current vs. transparent way of publishing

- Currently
  - Publications are the tip of the iceberg
  - "It's impossible to verify most of the results that computational scientists present at conferences and in papers" Donoho 2009

- A transparency solution
  - "An author attaches to every figure caption a pushbutton or a name tag usable to recalculate the figure from all its data, parameters, and programs. This provides **a concrete definition of reproducibility in computationally oriented research**" Claerbout 1992

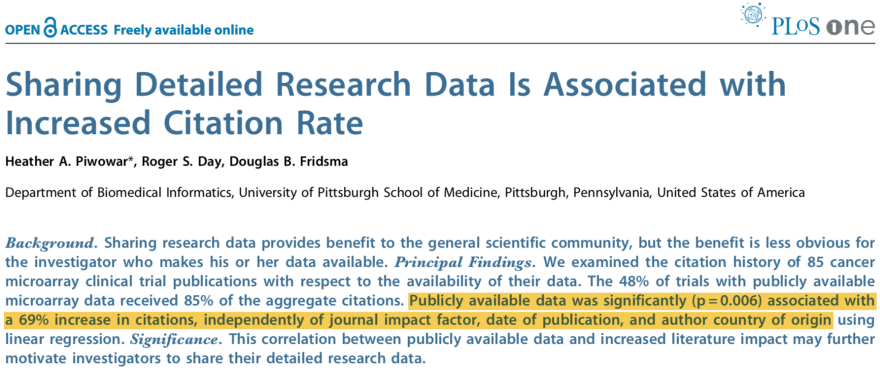
Slide modified from: http://mescal.imag.fr/membres/arnaud.legrand/teaching/2016/mosig_smpe_2_reproducible_research.pdf

What are the benefits of transparent research?

- Openness and reproducibility are essential to researchers to:
  - Assess the value of scientific claims (Sandve 2013)
  - Compare new methods to existing ones (Freire 2018)
  - Build on the work of other scientists with confidence and efficiency, i.e. without "reinventing the wheel" (Rule 2018)
  - Collaborate to improve and expand robust scientific workflows to accelerate scientific discoveries (Donoho 2009, Munafo 2017)

What are the benefits of transparent research?

- Increased citation rate



OPEN ACCESS Freely available online

PLoS one

Sharing Detailed Research Data Is Associated with Increased Citation Rate

Heather A. Piwowar*, Roger S. Day, Douglas B. Fridsma

Department of Biomedical Informatics, University of Pittsburgh School of Medicine, Pittsburgh, Pennsylvania, United States of America

Background. Sharing research data provides benefit to the general scientific community, but the benefit is less obvious for the investigator who makes his or her data available. ***Principal Findings.*** We examined the citation history of 85 cancer microarray clinical trial publications with respect to the availability of their data. The 48% of trials with publicly available microarray data received 85% of the aggregate citations. Publicly available data was significantly (p = 0.006) associated with a 69% increase in citations, independently of journal impact factor, date of publication, and author country of origin using linear regression. ***Significance.*** This correlation between publicly available data and increased literature impact may further motivate investigators to share their detailed research data.

(Piwowar 2007)

REPRODUCIBLE RESEARCH FOR SCIENTIFIC COMPUTING

PATRICK VANDEWALLE

Code Sharing Is Associated with Research Impact in Image Processing

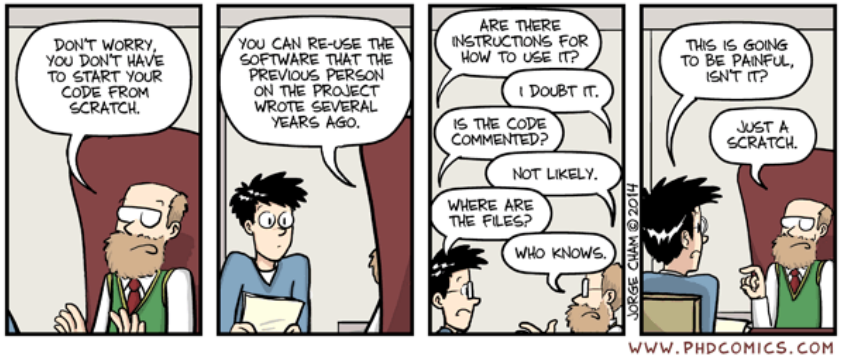
In computational sciences such as image processing, publishing usually isn't enough to allow other researchers to verify results. Often, supplementary materials such as source code and measurement data are required. Yet most researchers choose not to make their code available because of the extra time required to prepare it. Are such efforts actually worthwhile, though?

(Vandewalle 2012)

"The median number of citations […] increases with a factor of 3 when code is available online"

- Personal and group self-discipline (Donoho 2009)

- Reproducibility helps defeat self-deception (Nuzzo 2015)



Funding agencies support transparent research

- Europe: EOSC, Horizon 2020, OpenAire
- US: NIH, Gates Foundation, Chan-Zuckerberg Initiative
- Canada: Open data
- Australia, New Zealand, Asia, …

How can we conduct transparent research?

- Historically, research data, tools, and processes were rarely openly available because of limited storage and computational power (Munafo 2017)

- Nowadays there are several tools to conduct transparent research
  - Open access data
  - Reproducible workflows
  - Interactive publications

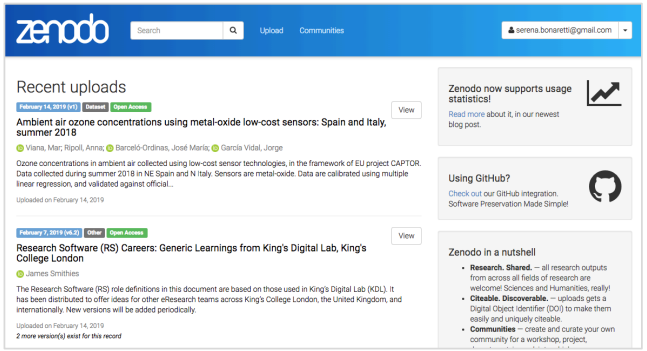
A few questions about our research practice

- How many of us have uploaded data and/or code to a public repository?
- How many of us use Jupyter notebook or R markdown for reproducible workflows?
- How many of us have written an interactive publication?

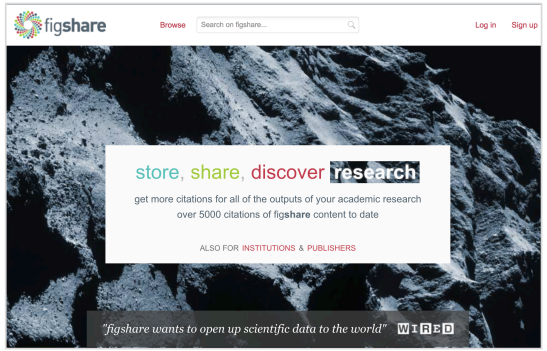
Transparent research:
Open-access data,
reproducible workflows,
and interactive publications

Data repositories

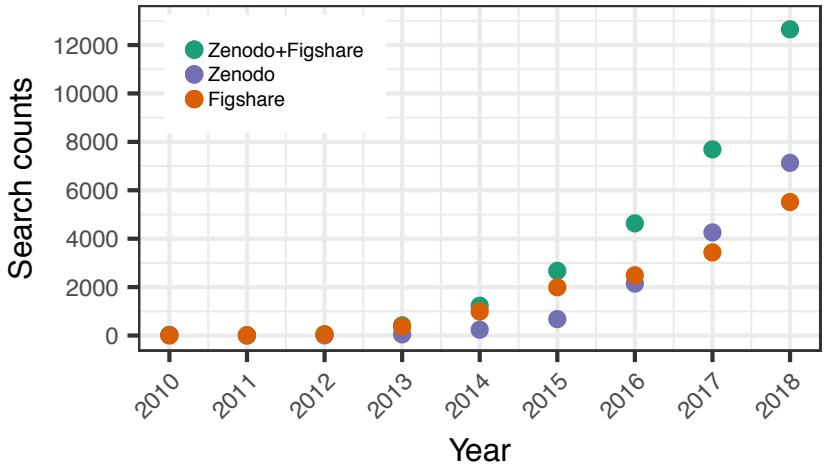
- Data repositories provide a DOI
  - Data and software are persistent and citable



https://zenodo.org/

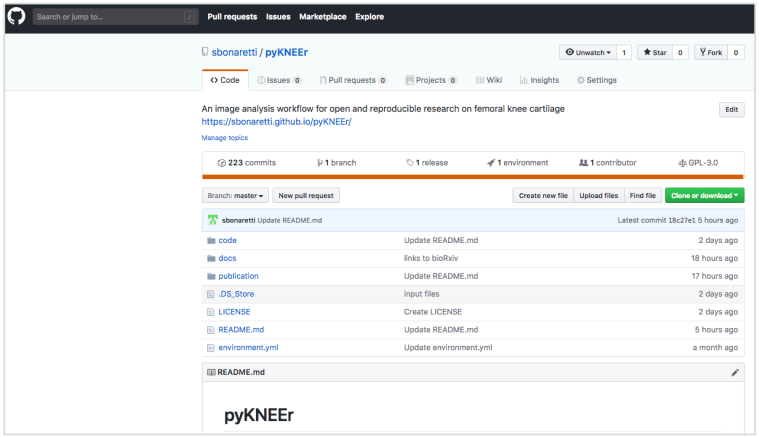


https://figshare.com/

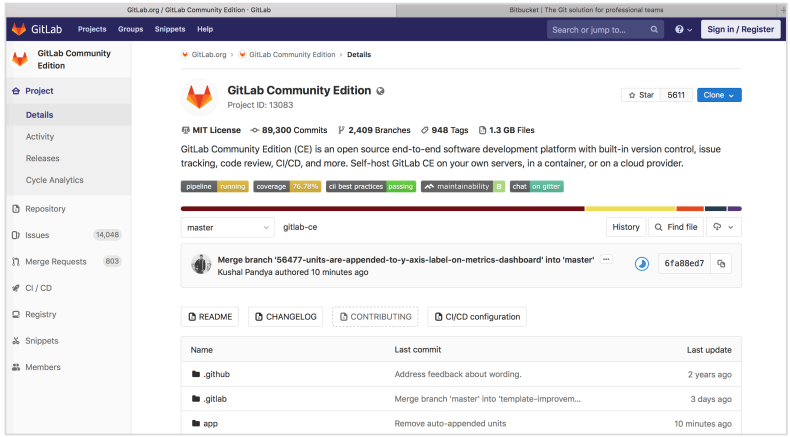


GitHub

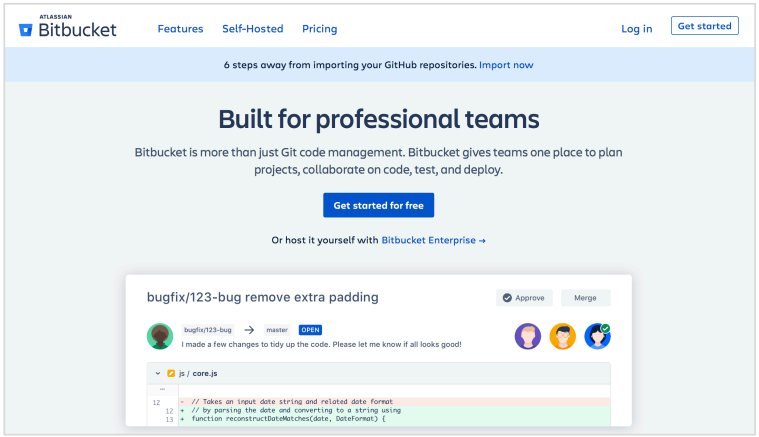
Software repositories



https://github.com/



https://about.gitlab.com



https://bitbucket.org

- Version control for reproducibility

Metadata and documentation

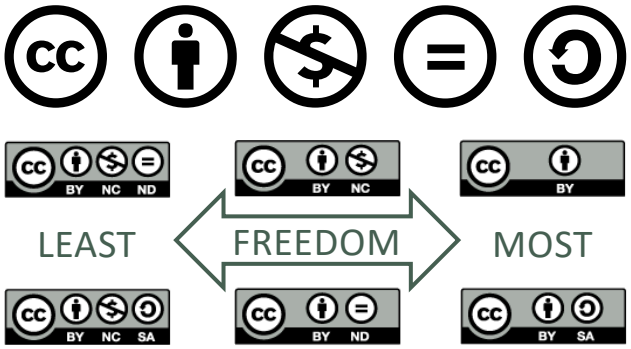
- Characteristics of raw and derived data
- Code documentation

- Formats
  - Fields in repositories (e.g. Zenodo)
  - Text files
  - README.md in GitHub
  - Website
  - …

License

- Online material is automatically protected by copyright → Add a license

DATA AND PUBLICATIONS
(https://creativecommons.org/choose)



CODE
(https://choosealicense.com/licenses/)

	GNU AGPLv3	GNU GPLv3	GNU LGPLv3	Mozilla Public 2.0	Apache License 2.0	MIT License	The Unlicense
Commercial use	●	●	●	●	●	●	●
Distribution	●	●	●	●	●	●	●
Modification	●	●	●	●	●	●	●
Patent use	●	●	●	●	●		
Private use	●	●	●	●	●	●	●
Disclose source	●	●	●	●			
License and copyright notice	●	●	●	●	●	●	
Network use is distribution	●						
Same license	●	●	●	●	●		
State changes	●	●	●				
Liability	●	●	●	●	●	●	●
Warranty	●	●	●	●	●	●	●
Trademark use				●	●		

Transparent research:
Open-access data,
reproducible workflows,
and interactive publications

How do we create reproducible workflows?

OPEN ACCESS Freely available online

PLOS | COMPUTATIONAL BIOLOGY

Editorial

Ten Simple Rules for Reproducible Computational Research

Geir Kjetil Sandve[1,2]*, Anton Nekrutenko[3], James Taylor[4], Eivind Hovig[1,5,6]

1 Department of Informatics, University of Oslo, Blindern, Oslo, Norway, **2** Centre for Cancer Biomedicine, University of Oslo, Blindern, Oslo, Norway, **3** Department of Biochemistry and Molecular Biology and The Huck Institutes for the Life Sciences, Penn State University, University Park, Pennsylvania, United States of America, **4** Department of Biology and Department of Mathematics and Computer Science, Emory University, Atlanta, Georgia, United States of America, **5** Department of Tumor Biology, Institute for Cancer Research, The Norwegian Radium Hospital, Oslo University Hospital, Montebello, Oslo, Norway, **6** Institute for Medical Informatics, The Norwegian Radium Hospital, Oslo University Hospital, Montebello, Oslo, Norway

(Sandve 2013)

PLOS | COMPUTATIONAL BIOLOGY

EDITORIAL

Ten Simple Rules for Taking Advantage of Git and GitHub

Yasset Perez-Riverol[1]*, Laurent Gatto[2], Rui Wang[1], Timo Sachsenberg[3], Julian Uszkoreit[4], Felipe da Veiga Leprevost[5], Christian Fufezan[6], Tobias Ternent[1], Stephen J. Eglen[7], Daniel S. Katz[8], Tom J. Pollard[9], Alexander Konovalov[10], Robert M. Flight[11], Kai Blin[12], Juan Antonio Vizcaíno[1]*

1 European Molecular Biology Laboratory, European Bioinformatics Institute (EMBL-EBI), Wellcome Trust Genome Campus, Hinxton, Cambridge, United Kingdom, **2** Computational Proteomics Unit, Cambridge Systems Biology Centre, University of Cambridge, Cambridge, United Kingdom, **3** Applied Bioinformatics and Department of Computer Science, University of Tübingen, Tübingen, Germany, **4** Medizinisches Proteom-Center, Ruhr-Universität Bochum, Bochum, Germany, **5** Department of Pathology, University of Michigan, Ann Arbor, Michigan, United States of America, **6** Institute of Plant Biology and Biotechnology, University of Münster, Münster, Germany, **7** Centre for Mathematical Sciences, University of Cambridge, Cambridge, United Kingdom, **8** National Center for Supercomputing Applications and Graduate School of Library and Information Science, University of Illinois, Urbana, Illinois, United States of America, **9** MIT Laboratory for Computational Physiology, Institute for Medical Engineering and Science, Massachusetts Institute of Technology, Cambridge, Massachusetts, United States of America, **10** Centre for Interdisciplinary Research in Computational Algebra, University of St Andrews, St Andrews, United Kingdom, **11** Department of Molecular Biology and Biochemistry, Markey Cancer Center, Resource Center for Stable Isotope-Resolved Metabolomics, University of Kentucky, Lexington, Kentucky, United States of America, **12** The Novo Nordisk Foundation Center for Biosustainability, Technical University of Denmark, Hørsholm, Denmark

(Perez-Riverol 2016)

PLOS | COMPUTATIONAL BIOLOGY

PERSPECTIVE

Good enough practices in scientific computing

Greg Wilson[1☯]*, Jennifer Bryan[2☯], Karen Cranston[3☯], Justin Kitzes[4☯], Lex Nederbragt[5☯], Tracy K. Teal[6☯]

1 Software Carpentry Foundation, Austin, Texas, United States of America, **2** RStudio and Department of Statistics, University of British Columbia, Vancouver, British Columbia, Canada, **3** Department of Biology, Duke University, Durham, North Carolina, United States of America, **4** Energy and Resources Group, University of California, Berkeley, Berkeley, California, United States of America, **5** Centre for Ecological and Evolutionary Synthesis, University of Oslo, Oslo, Norway, **6** Data Carpentry, Davis, California, United States of America

(Wilson 2016)

Computer Science > Other Computer Science

Ten Simple Rules for Reproducible Research in Jupyter Notebooks

Adam Rule, Amanda Birmingham, Cristal Zuniga, Ilkay Altintas, Shih-Cheng Huang, Rob Knight, Niema Moshiri, Mai H. Nguyen, Sara Brin Rosenthal, Fernando Pérez, Peter W. Rose

(Submitted on 13 Oct 2018)

Reproducibility of computational studies is a hallmark of scientific methodology. It enables researchers to build with confidence on the methods and findings of others, reuse and extend computational pipelines, and thereby drive scientific progress. Since many experimental studies rely on computational analyses, biologists need guidance on how to set up and document reproducible data analyses or simulations.
In this paper, we address several questions about reproducibility. For example, what are the technical and non-technical barriers to reproducible computational studies? What opportunities and challenges do computational notebooks offer to overcome some of these barriers? What tools are available and how can they be used effectively?
We have developed a set of rules to serve as a guide to scientists with a specific focus on computational notebook systems, such as Jupyter Notebooks, which have become a tool of choice for many applications. Notebooks combine detailed workflows with narrative text and visualization of results. Combined with software repositories and open source licensing, notebooks are powerful tools for transparent, collaborative, reproducible, and reusable data analyses.

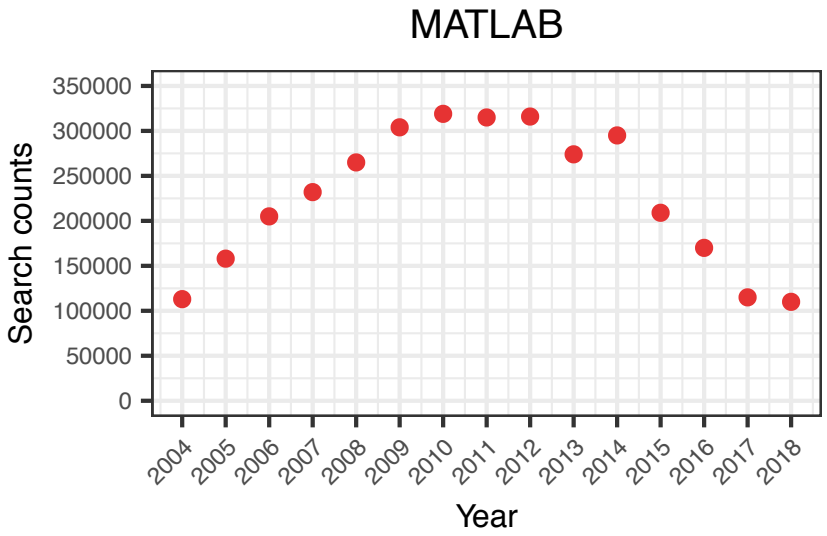
(Rule 2018)

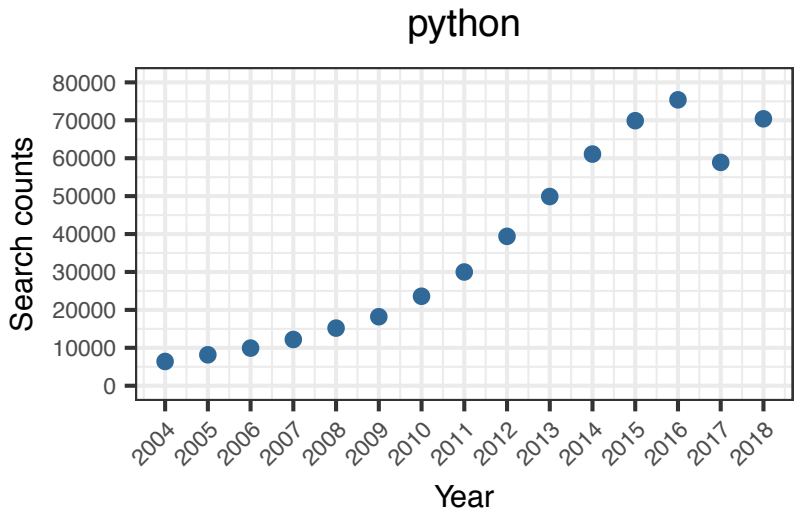
Summary of rules

- Use open source programming language and file formats
  - e.g. python, R, .txt, .csv, …
- Keep track of how you create results
  - Document computational provenance of derived data
    - Store raw data, including data behind plots
    - Use computational notebooks
  - Avoid manual data manipulation
    - Tidy data tables with scripts
  - Record dependencies
- Be your own user
  - Code well, be transparent, be simple

Open access programming language

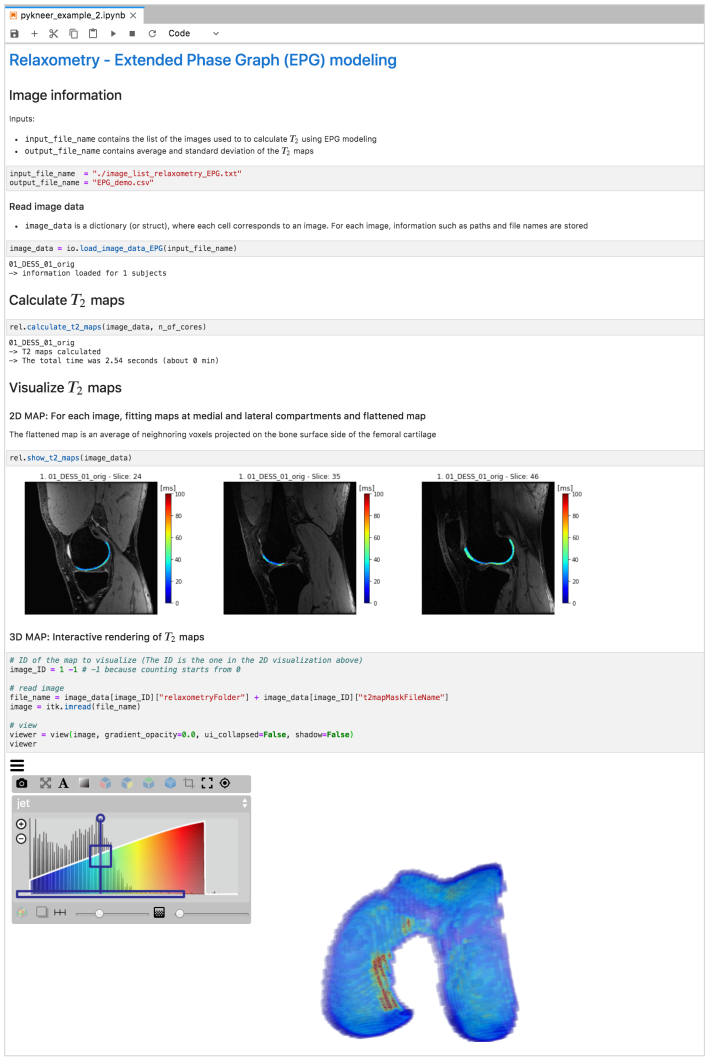
- Plenty of libraries, examples, and Q&A (somebody already had your issue!)
- No purchase needed





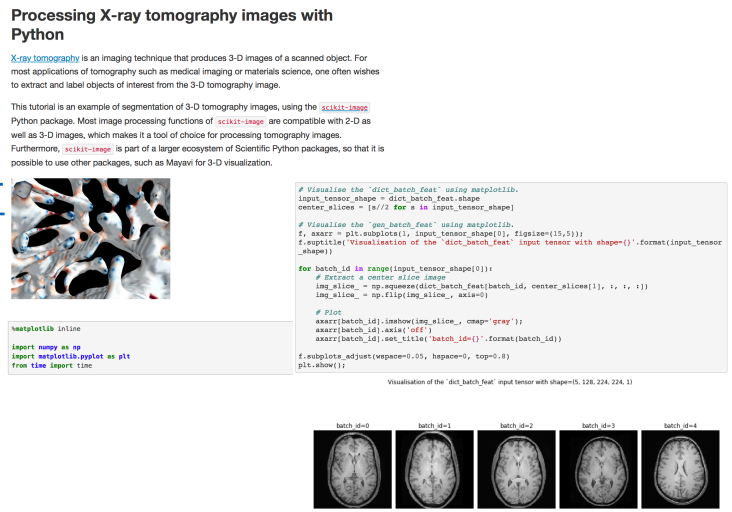
GitHub - Modified from Donoho 2015

Jupyter notebooks



- Open-source web application integrating
  - Live code
  - Narrative text with equations
  - Visualizations
- Easy to use and share among researchers

- Examples in medical imaging
  - Bone microCT
  - SimpleITK notebooks
  - SPIE 2019 workshop
  - Deep Learning Toolkit
  - VTK
  - pyKNEEr



Dependences

- Reproducibility of computational *environment*
  - Future versions might be incompatible

Dependencies

```
%reload_ext watermark
%watermark -v -m -p pandas,numpy,itk
```

```
CPython 3.7.3
IPython 7.4.0

pandas 0.24.2
numpy 1.16.2
itk unknown

compiler   : Clang 4.0.1 (tags/RELEASE_401/final)
system     : Darwin
release    : 17.7.0
machine    : x86_64
processor  : i386
CPU cores  : 4
interpreter: 64bit
```

```
%%R
sessionInfo()
```

```
R version 3.3.3 (2017-03-06)
Platform: x86_64-apple-darwin13.4.0 (64-bit)
Running under: macOS  10.13.6

locale:
[1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8

attached base packages:
[1] tools     stats     graphics  grDevices utils     datasets  methods
[8] base

other attached packages:
[1] ggplot2_3.1.0

loaded via a namespace (and not attached):
 [1] Rcpp_1.0.1       digest_0.6.18    withr_2.1.2      crayon_1.3.4
 [5] dplyr_0.8.1      assertthat_0.2.1 grid_3.3.3       plyr_1.8.4
 [9] R6_2.4.0         gtable_0.2.0     magrittr_1.5     scales_1.0.0
[13] pillar_1.4.0     rlang_0.3.4      lazyeval_0.2.1   labeling_0.3
[17] glue_1.3.1       purrr_0.3.2      munsell_0.5.0    pkgconfig_2.0.2
[21] colorspace_1.3-2 tidyselect_0.2.5 tibble_2.1.1
```

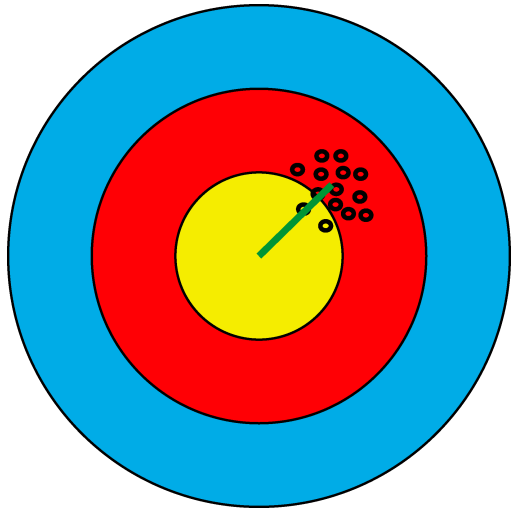
```
import datetime
now = datetime.datetime.now()
print ("Date: " + str(now.day) + " " + str(now.month) + " " + str(now.year))
```

```
Date: 17 6 2019
```

Last note on reproducibility

- Reproducibility does not imply correctness
- Lack of reproducibility does not imply incorrectness

- Incorrectness can be found with reproducibility



Holmes 2018

Transparent research: Open-access data, reproducible workflows, and interactive publications

How do we make interactive publications?

AGU PUBLICATIONS

Earth and Space Science

REVIEW

10.1002/2015EA000136

Special Section:
Geoscience Papers of the Future

Key Points:
- Describes best practices for documenting research to support open science
- Publishing computational provenance with software and data improves science transparency
- Promotes approaches to achieve equitable credit for all digital research products

Correspondence to:
Y. Gil,
gil@isi.edu

Toward the Geoscience Paper of the Future: Best practices for documenting and sharing research from data to software to provenance

Yolanda Gil[1], Cédric H. David[2], Ibrahim Demir[3], Bakinam T. Essawy[4], Robinson W. Fulweiler[5], Jonathan L. Goodall[4], Leif Karlstrom[6], Huikyo Lee[2], Heath J. Mills[7], Ji-Hyun Oh[2,8], Suzanne A. Pierce[9], Allen Pope[10,11], Mimi W. Tzeng[12], Sandra R. Villamizar[13], and Xuan Yu[14]

[1]Information Sciences Institute and Department of Computer Science, University of Southern California, Los Angeles, California, USA, [2]Jet Propulsion Laboratory, California Institute of Technology, Pasadena, California, USA, [3]IIHR Hydroscience and Engineering Institute, University of Iowa, Iowa City, Iowa, USA, [4]Department of Civil and Environmental Engineering, University of Virginia, Charlottesville, Virginia, USA, [5]Department of Earth and Environment, Department of Biology, Boston University, Boston, Massachusetts, USA, [6]Department of Earth Sciences, University of Oregon, Eugene, Oregon, USA, [7]Division of Natural Sciences, University of Houston–Clear Lake, Houston, Texas, USA, [8]Computer Science Department, University of Southern California, Los Angeles, California, USA, [9]Texas Advanced Computing Center and Jackson School of Geosciences, University of Texas at Austin, Austin, Texas, USA, [10]National Snow and Ice Data Center, University of Colorado Boulder, Boulder, Colorado, USA, [11]Polar Science Center, Applied Physics Laboratory, University of Washington, Seattle, Washington, USA, [12]Data Management Center, Dauphin Island Sea Lab, Dauphin Island, Alabama, USA, [13]Universidad Pontificia Bolivariana, Colombia, [14]Department of Geological Sciences, University of Delaware, Newark, Delaware, USA

(Gil 2016)

On Reproducible AI

Towards reproducible research, open science, and digital scholarship in AI publications

Odd Erik Gundersen, Yolanda Gil and David W. Aha

Abstract

Background: Artificial intelligence, like any science, must rely on reproducible experiments to validate results. **Objective:** To give practical and pragmatic recommendations for how to document AI research so that results are reproducible. **Method:** Our analysis of the literature shows that AI publications currently fall short of providing enough documentation to facilitate reproducibility. Our suggested best practices are based on a framework for reproducibility and recommendations for best practices given by scientific organizations, scholars, and publishers. **Results:** We have made a reproducibility checklist based on our investigation and described how every item in the checklist can be documented by authors and examined by reviewers. **Conclusion:** We encourage authors and reviewers to use the suggested best practices and author checklist when considering submissions for AAAI publications and conferences.

(Gundersen 2018)

- Buttons and links to data and code

Examples of interactive papers

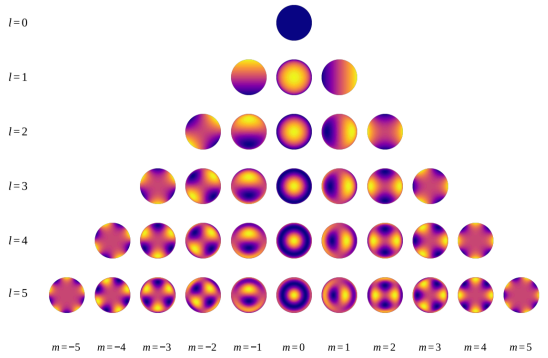


Figure 1. The real spherical harmonics up to degree $l = 5$ computed from Equation (1). In these plots, the x-axis points to the right, the y-axis points up, and the z-axis points out of the page.

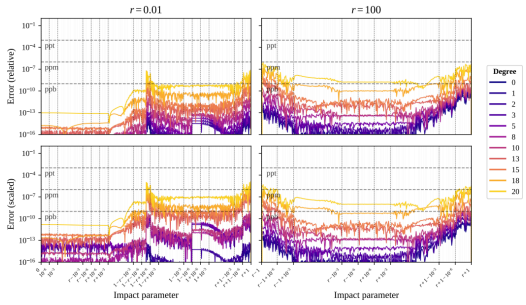


Figure 12. Similar to Figure 11, but showing instead the error on the *derivative* of the flux with respect to the impact parameter computed analytically with autodifferentiation. The error is computed relative to a numerical derivative computed at 128 bit precision.

(Luger 2018)

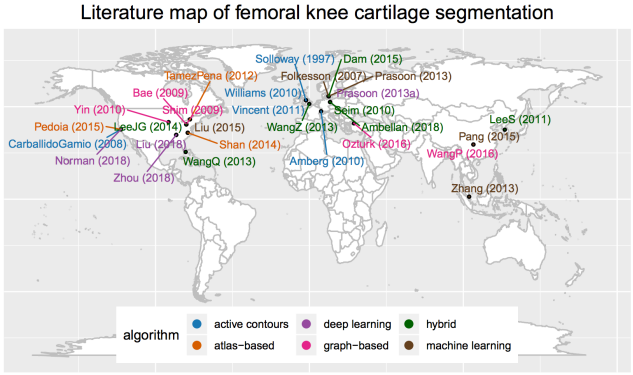


Figure 1: The visualization shows name of first author, year of publication, affiliation of last author, and segmentation method for 29 relevant publications on femoral knee cartilage segmentation from 1997 to 2018. Publications by segmentation method and in alphabetical order are: Active contours: Amberg(2010)[2], Carballido-Gamio(2008)[6], Solloway(1997)[62], Vincent(2011)[67], Williams(2010)[74]; Atlas-based: Pedoia(2015)[47], Shan(2014)[59], Tamez-Pena(2012)[64]; Deep-learning: Liu(2018)[33], Norman(2018)[43], Prasoon(2013a)[52], Zhou(2018)[81]; Graph-based: Bae(2009)[4], Ozturk(2016)[44], Shim(2009)[60], WangP(2016)[68], Yin(2010)[78]; Hybrid: Ambellan(2018)[1], Dam(2015)[11], LeeJG(2014)[30], LeeS(2011)[31], Seim(2010)[57], WangQ(2013)[69], WangZ(2013)[70]; Machine learning: Folkesson(2007)[18], Liu(2015)[34], Pang(2015)[46], Prasoon(2013)[51], Zhang(2013)[80]. This graph and graphs in Fig. 4 and Fig. 5 were made in Jupyter notebook using `ggplot2` [71], an R package based on the grammar of graphics [72].

	Repository	Metadata / Documentation	Software / Language	License	DOI	Citation
Software Used						
Preprocessing	Bitbucket	Wiki	C++, ITK	Apache	https://doi.org/10.1016/j.media.2014.05.008*	[59]*
elastix 4.8	GitHub	Github Wiki	C++, ITK	Apache	https://doi.org/10.1109/TMI.2009.2035616*	[28]*
Developed pyKNEEr	GitHub	Website	python, Jupyter notebook	GNU GPLv3	https://doi.org/10.5281/zenodo.2574172	Bonaretti S. et al. "pyKNEER" (v0.0.1). Zenodo. 2019. 10.5281/zenodo.2574172
Data						
Original	OAI	Website	-	Data user agreement	https://doi.org/10.1016/j.joca.2008.06.016*	[49]*
Derived (results)	Zenodo	Jupyter notebook	-	CC-BY-NC-SA	https://doi.org/10.5281/zenodo.2530609	Bonaretti S. et al. Dataset used in (Bonaretti et al. 2019). Zenodo. 2019. 10.5281/zenodo.2530609

Dataset	OAI1-DESS	OAI1-T_2	OAI2-BL	OAI2-FU	inHouse-DESS	inHouse-CQ
Number of subjects	19	19	88	88	4	4
I. Acquisition parameters						
Acquisition protocol	DESS	T_2-w	DESS		DESS	CubeQuant
Acquisition plane	sagittal	sagittal	sagittal		sagittal	sagittal
Number of images in series	2 (1 available)°	7	2 (1 available)°		2	4
In-plane spacing [mm]	0.3646 x 0.3646 (0.4270 x 0.4270)*	0.3125 x 0.3125 (0.4296 x 0.4296)*	0.3646 x 0.3646		0.3125 x 0.3125	0.3125 x 0.3125
Slice thickness [mm]	0.7 (0.75)*	3 (3.5)*	0.7		1.5	3
Echo time (TE) [ms]	4.7	10, 20, 30, 40, 50, 60, 70	4.7		42.52	-
Spin-lock time (TSL) [ms]	-	-	-		-	1, 10, 30, 60
Repetition time (TR) [ms]	16.32	2700 (2900)*	16.32		25	1302
Flip angle [°]	25	180	25		30	90
II. Ground truth segmentation						
Method	atlas-based		active models		- -	- -
Anatomy	femur, femoral cartilage		femoral cartilage		- -	- -
Type	mask		contour		- -	- -
III. Experimental results						
Image number in series	1	1 2-7	1	1	1 2	1 2-4
Preprocessing						
Spatial standardization	•	• •	•	•	• •	• •
Intensity standardization	•	• -	•	-	• -	• -
Segmentation						
Find reference	4, 8, 10, 13, 16	- -	-	-	- -	- -
Intersubject	•	- -	•	-	• -	- -
Longitudinal	-	- -	-	•	- -	- -
Multimodal	-	• -	-	-	- -	• -
Segmentation quality						
Dice coefficient	•	• -	•	•	- -	- -
Analysis						
Morphology	•○	•○ -	•○	•○	• -	• -
Relaxation	-	•○	-	-	•	•

(Bonaretti 2019)

Initiative at a community level

- QMSKI 2019
  - Wiki with How To: https://github.com/QMSKI/TransparentQMSKI/wiki

QMSKI 2021
in
Netherlands / Belgium

pyKNEEr: An image analysis workflow for open and reproducible research on femoral knee cartilage

Serena Bonaretti[1,2], Garry E. Gold[1], Gary S. Beaupre[2,3]

[1]Department of Radiology, Stanford University, Stanford, CA, USA
[2]Musculoskeletal Research Laboratory, VA Palo Alto Health Care System, CA, USA
[3]Department of Bioengineering, Stanford University, Stanford, CA, USA

https://sbonaretti.github.io/

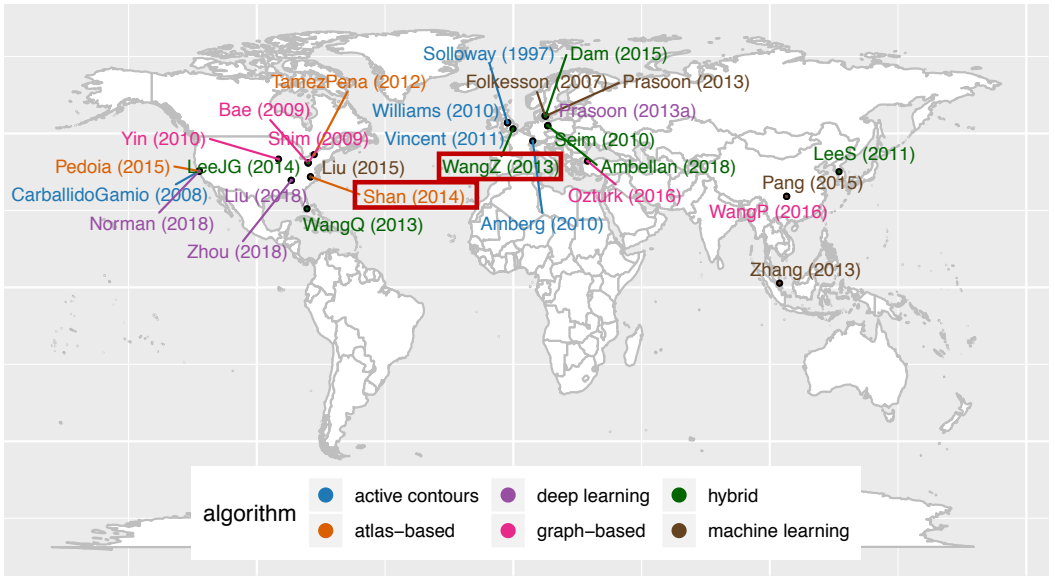
Image-based analysis of femoral knee cartilage

- In knee OA, analysis of femoral cartilage from MR images has assumed an important role (Hafezi-Nejad 2018)

- Analysis workflows include:
  - Preprocessing
  - Segmentation
  - Morphology and Relaxometry

- Segmentation is a major challenge
  - Of 29 segmentation algorithms, only 2 have open source code (Wang 2013, Shan 2014)

Literature map of femoral knee cartilage segmentation

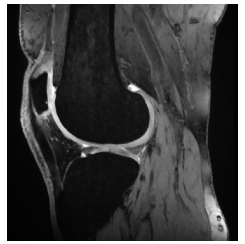


pyKNEEr: An image analysis workflow for open and reproducible research on femoral knee cartilage

Openness of *pyKNEEr*

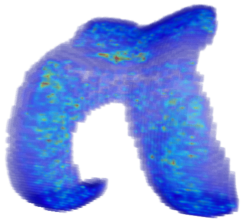
- Code written in *python*
- Use of open file formats
  - *.mha, .txt, .csv*
- Modular structure to favor code expansion
- Reuse of developed code
  - Intensity preprocessing from Shan et al.
  - Atlas-based segmentation using *elastix*
- Available on GitHub
  - GNU GPLv3 license and DOI

I. Preprocessing

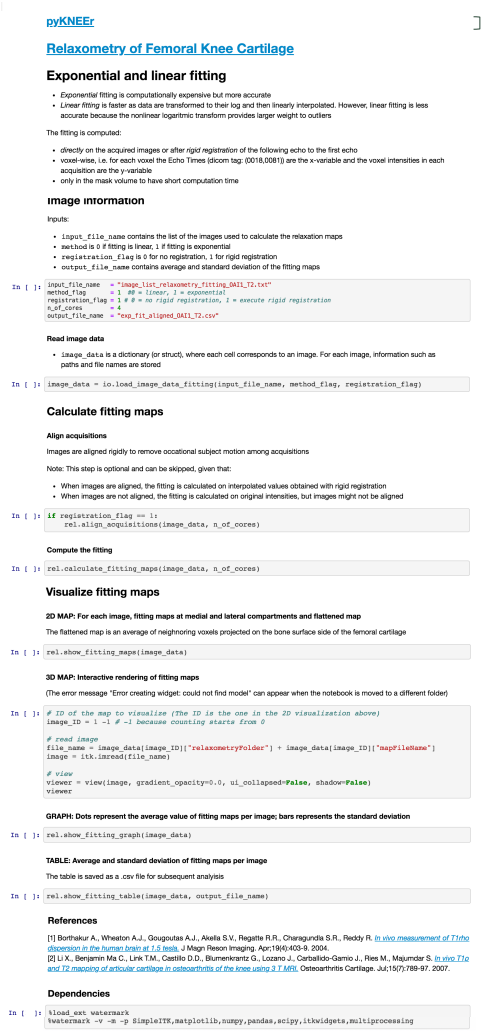


II. Segmentation

III. Analysis



Reproducibility of *pyKNEEr*: Jupyter notebook



pyKNEEr

Relaxometry of Femoral Knee Cartilage

Exponential and linear fitting

- *Exponential* fitting is computationally expensive but more accurate
- *Linear fitting* is faster as data are transformed to their log and then linearly interpolated. However, linear fitting is less accurate because the nonlinear logaritmic transform provides larger weight to outliers

The fitting is computed:

- *directly* on the acquired images or after *rigid registration* of the following echo to the first echo
- voxel-wise, i.e. for each voxel the Echo Times (dicom tag: (0018,0081)) are the x-variable and the voxel intensities in each acquisition are the y-variable
- only in the mask volume to have short computation time

Image information

Inputs:

- `input_file_name` contains the list of the images used to calculate the relaxation maps
- `method` is 0 if fitting is linear, 1 if fitting is exponential
- `registration_flag` is 0 for no registration, 1 for rigid registration
- `output_file_name` contains average and standard deviation of the fitting maps

```
In [ ]: input_file_name   = "image_list_relaxometry_fitting_DAI1_T2.txt"
method_flag     = 1 ## 0 = linear, 1 = exponential
registration_flag = 1 # 0 = no rigid registration, 1 = execute rigid registration
n_of_cores      = 4
output_file_name  = "exp_fit_aligned_DAI1_T2.csv"
```

Read image data

- `image_data` is a dictionary (or struct), where each cell corresponds to an image. For each image, information such as paths and file names are stored

```
In [ ]: image_data = io.load_image_data_fitting(input_file_name, method_flag, registration_flag)
```

Calculate fitting maps

Align acquisitions

Images are aligned rigidly to remove occational subject motion among acquisitions

Note: This step is optional and can be skipped, given that:

- When images are aligned, the fitting is calculated on interpolated values obtained with rigid registration
- When images are not aligned, the fitting is calculated on original intensities, but images might not be aligned

```
In [ ]: if registration_flag == 1:
            rel.align_acquisitions(image_data, n_of_cores)
```

Compute the fitting

```
In [ ]: rel.calculate_fitting_maps(image_data, n_of_cores)
```

Visualize fitting maps

2D MAP: For each image, fitting maps at medial and lateral compartments and flattened map

The flattened map is an average of neighnoring voxels projected on the bone surface side of the femoral cartilage

```
In [ ]: rel.show_fitting_maps(image_data)
```

3D MAP: Interactive rendering of fitting maps

(The error message "Error creating widget: could not find model" can appear when the notebook is moved to a different folder)

```
In [ ]: # ID of the map to visualize (The ID is the one in the 2D visualization above)
        image_ID = 1 -1 # -1 because counting starts from 0

        # read image
        file_name = image_data[image_ID]["relaxometryFolder"] + image_data[image_ID]["mapFileName"]
        image = itk.imread(file_name)

        # view
        viewer = view(image, gradient_opacity=0.0, ui_collapsed=False, shadow=False)
        viewer
```

GRAPH: Dots represent the average value of fitting maps per image; bars represents the standard deviation

```
In [ ]: rel.show_fitting_graph(image_data)
```

TABLE: Average and standard deviation of fitting maps per image

The table is saved as a .csv file for subsequent analysis

```
In [ ]: rel.show_fitting_table(image_data, output_file_name)
```

References

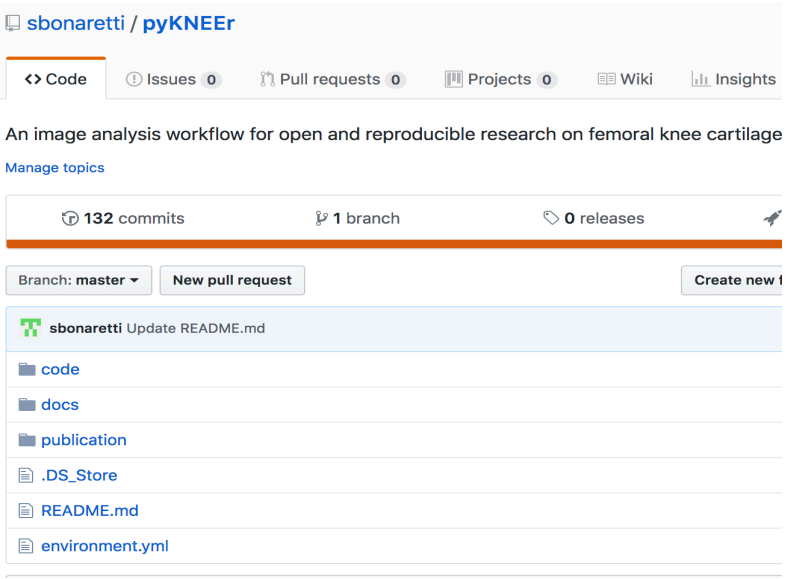
[1] Borthakur A., Wheaton A.J., Gougoutas A.J., Akella S.V., Regatte R.R., Charagundla S.R., Reddy R. *In vivo measurement of T1rho dispersion in the human brain at 1.5 tesla.* J Magn Reson Imaging. Apr;19(4):403-9. 2004.
[2] Li X., Benjamin Ma C., Link T.M., Castillo D.D., Blumenkrantz G., Lozano J., Carballido-Gamio J., Ries M., Majumdar S. *In vivo T1ρ and T2 mapping of articular cartilage in osteoarthritis of the knee using 3 T MRI.* Osteoarthritis Cartilage. Jul;15(7):789-97. 2007.

Dependencies

```
In [ ]: %load_ext watermark
        %watermark -v -m -p SimpleITK,matplotlib,numpy,pandas,scipy,itkwidgets,multiprocessing
```

1. Link to GitHub repository

https://github.com/sbonaretti/pyKNEEr



Reproducibility of *pyKNEEr*: Jupyter notebook

pyKNEEr

Relaxometry of Femoral Knee Cartilage

Exponential and linear fitting

- *Exponential* fitting is computationally expensive but more accurate
- *Linear fitting* is faster as data are transformed to their log and then linearly interpolated. However, linear fitting is less accurate because the nonlinear logaritmic transform provides larger weight to outliers

The fitting is computed:

- *directly* on the acquired images or after *rigid registration* of the following echo to the first echo
- voxel-wise, i.e. for each voxel the Echo Times (dicom tag: (0018,0081)) are the x-variable and the voxel intensities in each acquisition are the y-variable
- only in the mask volume to have short computation time

Image information

Inputs:

- `input_file_name` contains the list of the images used to calculate the relaxation maps
- `method` is 0 if fitting is linear, 1 if fitting is exponential
- `registration_flag` is 0 for no registration, 1 for rigid registration
- `output_file_name` contains average and standard deviation of the fitting maps

In [ ]:
```
input_file_name    = "image_list_relaxometry_fitting_DAI1_T2.txt"
method_flag        = 1 ## 0 = linear, 1 = exponential
registration_flag  = 1 # 0 = no rigid registration, 1 = execute rigid registration
n_of_cores         = 4
output_file_name   = "exp_fit_aligned_DAI1_T2.csv"
```

Read image data

- `image_data` is a dictionary (or struct), where each cell corresponds to an image. For each image, information such as paths and file names are stored

In [ ]:
```
image_data = io.load_image_data_fitting(input_file_name, method_flag, registration_flag)
```

Calculate fitting maps

Align acquisitions

Images are aligned rigidly to remove occational subject motion among acquisitions

Note: This step is optional and can be skipped, given that:

- When images are aligned, the fitting is calculated on interpolated values obtained with rigid registration
- When images are not aligned, the fitting is calculated on original intensities, but images might not be aligned

In [ ]:
```
if registration_flag == 1:
    rel.align_acquisitions(image_data, n_of_cores)
```

Compute the fitting

In [ ]:
```
rel.calculate_fitting_maps(image_data, n_of_cores)
```

Visualize fitting maps

2D MAP: For each image, fitting maps at medial and lateral compartments and flattened map

The flattened map is an average of neighnoring voxels projected on the bone surface side of the femoral cartilage

In [ ]:
```
rel.show_fitting_maps(image_data)
```

3D MAP: Interactive rendering of fitting maps

(The error message "Error creating widget: could not find model" can appear when the notebook is moved to a different folder)

In [ ]:
```
# ID of the map to visualize (The ID is the one in the 2D visualization above)
image_ID = 1 -1 # -1 because counting starts from 0

# read image
file_name = image_data[image_ID]["relaxometryFolder"] + image_data[image_ID]["mapFileName"]
image = itk.imread(file_name)

# view
viewer = view(image, gradient_opacity=0.0, ui_collapsed=False, shadow=False)
viewer
```

GRAPH: Dots represent the average value of fitting maps per image; bars represents the standard deviation

In [ ]:
```
rel.show_fitting_graph(image_data)
```

TABLE: Average and standard deviation of fitting maps per image

The table is saved as a .csv file for subsequent analysis

In [ ]:
```
rel.show_fitting_table(image_data, output_file_name)
```

References

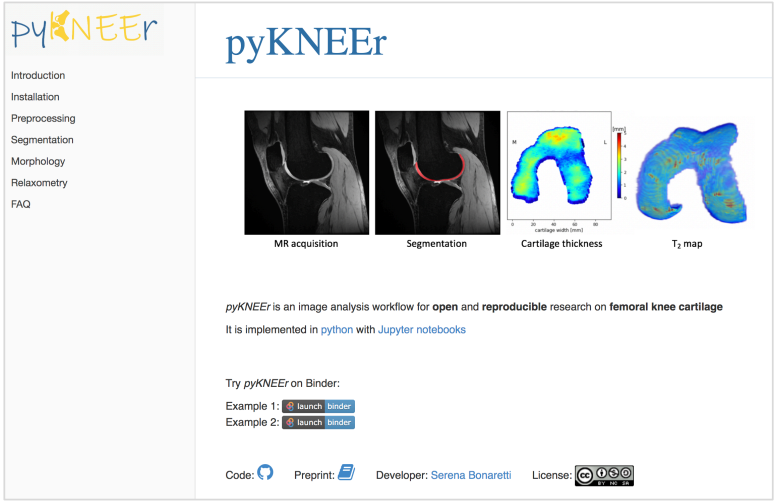
[1] Borthakur A., Wheaton A.J., Gougoutas A.J., Akella S.V., Regatte R.R., Charagundla S.R., Reddy R. *In vivo measurement of T1rho dispersion in the human brain at 1.5 tesla.* J Magn Reson Imaging. Apr;19(4):403-9. 2004.
[2] Li X., Benjamin Ma C., Link T.M., Castillo D.D., Blumenkrantz G., Lozano J., Carballido-Gamio J., Ries M., Majumdar S. *In vivo T1ρ and T2 mapping of articular cartilage in osteoarthritis of the knee using 3 T MRI.* Osteoarthritis Cartilage. Jul;15(7):789-97. 2007.

Dependencies

In [ ]:
```
%load_ext watermark
%watermark -v -m -p SimpleITK,matplotlib,numpy,pandas,scipy,itkwidgets,multiprocessing
```

1. Link to GitHub repository
2. Link to documentation

https://sbonaretti.github.io/pyKNEEr/



Reproducibility of *pyKNEEr*: Jupyter notebook

pyKNEEr

Relaxometry of Femoral Knee Cartilage

Exponential and linear fitting

- *Exponential* fitting is computationally expensive but more accurate
- *Linear fitting* is faster as data are transformed to their log and then linearly interpolated. However, linear fitting is less accurate because the nonlinear logaritmic transform provides larger weight to outliers

The fitting is computed:

- *directly* on the acquired images or after *rigid registration* of the following echo to the first echo
- voxel-wise, i.e. for each voxel the Echo Times (dicom tag: (0018,0081)) are the x-variable and the voxel intensities in each acquisition are the y-variable
- only in the mask volume to have short computation time

Image information

Inputs:

- `input_file_name` contains the list of the images used to calculate the relaxation maps
- `method` is 0 if fitting is linear, 1 if fitting is exponential
- `registration_flag` is 0 for no registration, 1 for rigid registration
- `output_file_name` contains average and standard deviation of the fitting maps

In [ ]:
```
input_file_name   = "image_list_relaxometry_fitting_DAI1_T2.txt"
method_flag       = 1 # 0 = linear, 1 = exponential
registration_flag = 1 # 0 = no rigid registration, 1 = execute rigid registration
n_of_cores        = 4
output_file_name  = "exp_fit_aligned_DAI1_T2.csv"
```

Read image data

- `image_data` is a dictionary (or struct), where each cell corresponds to an image. For each image, information such as paths and file names are stored

In [ ]:
```
image_data = io.load_image_data_fitting(input_file_name, method_flag, registration_flag)
```

Calculate fitting maps

Align acquisitions

Images are aligned rigidly to remove occational subject motion among acquisitions

Note: This step is optional and can be skipped, given that:

- When images are aligned, the fitting is calculated on interpolated values obtained with rigid registration
- When images are not aligned, the fitting is calculated on original intensities, but images might not be aligned

In [ ]:
```
if registration_flag == 1:
    rel.align_acquisitions(image_data, n_of_cores)
```

Compute the fitting

In [ ]:
```
rel.calculate_fitting_maps(image_data, n_of_cores)
```

Visualize fitting maps

2D MAP: For each image, fitting maps at medial and lateral compartments and flattened map

The flattened map is an average of neighboring voxels projected on the bone surface side of the femoral cartilage

In [ ]:
```
rel.show_fitting_maps(image_data)
```

3D MAP: Interactive rendering of fitting maps

(The error message "Error creating widget: could not find model" can appear when the notebook is moved to a different folder)

In [ ]:
```
# ID of the map to visualize (The ID is the one in the 2D visualization above)
image_ID = 1 -1 # -1 because counting starts from 0

# read image
file_name = image_data[image_ID]["relaxometryFolder"] + image_data[image_ID]["mapFileName"]
image = itk.imread(file_name)

# view
viewer = view(image, gradient_opacity=0.0, ui_collapsed=False, shadow=False)
viewer
```

GRAPH: Dots represent the average value of fitting maps per image; bars represents the standard deviation

In [ ]:
```
rel.show_fitting_graph(image_data)
```

TABLE: Average and standard deviation of fitting maps per image

The table is saved as a .csv file for subsequent analysis

In [ ]:
```
rel.show_fitting_table(image_data, output_file_name)
```

References

[1] Borthakur A., Wheaton A.J., Gougoutas A.J., Akella S.V., Regatte R.R., Charagundla S.R., Reddy R. *In vivo measurement of T1rho dispersion in the human brain at 1.5 tesla.* J Magn Reson Imaging. Apr;19(4):403-9. 2004.
[2] Li X., Benjamin Ma C., Link T.M., Castillo D.D., Blumenkrantz G., Lozano J., Carballido-Gamio J., Ries M., Majumdar S. *In vivo T1ρ and T2 mapping of articular cartilage in osteoarthritis of the knee using 3 T MRI.* Osteoarthritis Cartilage. Jul;15(7):789-97. 2007.

Dependencies

In [ ]:
```
%load_ext watermark
%watermark -v -m -p SimpleITK,matplotlib,numpy,pandas,scipy,itkwidgets,multiprocessing
```

1. Link to GitHub repository
2. Link to documentation
3. Introduction

Reproducibility of *pyKNEEr*: Jupyter notebook

pyKNEEr

Relaxometry of Femoral Knee Cartilage

Exponential and linear fitting

- *Exponential* fitting is computationally expensive but more accurate
- *Linear fitting* is faster as data are transformed to their log and then linearly interpolated. However, linear fitting is less accurate because the nonlinear logaritmic transform provides larger weight to outliers

The fitting is computed:

- *directly* on the acquired images or after *rigid registration* of the following echo to the first echo
- voxel-wise, i.e. for each voxel the Echo Times (dicom tag: (0018,0081)) are the x-variable and the voxel intensities in each acquisition are the y-variable
- only in the mask volume to have short computation time

Image information

Inputs:

- `input_file_name` contains the list of the images used to calculate the relaxation maps
- `method` is 0 if fitting is linear, 1 if fitting is exponential
- `registration_flag` is 0 for no registration, 1 for rigid registration
- `output_file_name` contains average and standard deviation of the fitting maps

```
input_file_name   = "image_list_relaxometry_fitting_OAI1_T2.txt"
method_flag       = 1  ## 0 = linear, 1 = exponential
registration_flag = 1 # 0 = no rigid registration, 1 = execute rigid registration
n_of_cores        = 4
output_file_name  = "exp_fit_aligned_OAI1_T2.csv"
```

Read image data

- `image_data` is a dictionary (or struct), where each cell corresponds to an image. For each image, information such as paths and file names are stored

```
image_data = io.load_image_data_fitting(input_file_name, method_flag, registration_flag)
```

Calculate fitting maps

Align acquisitions

Images are aligned rigidly to remove occational subject motion among acquisitions

Note: This step is optional and can be skipped, given that:

- When images are aligned, the fitting is calculated on interpolated values obtained with rigid registration
- When images are not aligned, the fitting is calculated on original intensities, but images might not be aligned

```
if registration_flag == 1:
    rel.align_acquisitions(image_data, n_of_cores)
```

Compute the fitting

```
rel.calculate_fitting_maps(image_data, n_of_cores)
```

Visualize fitting maps

2D MAP: For each image, fitting maps at medial and lateral compartments and flattened map

The flattened map is an average of neighboring voxels projected on the bone surface side of the femoral cartilage

```
rel.show_fitting_maps(image_data)
```

3D MAP: Interactive rendering of fitting maps

(The error message "Error creating widget: could not find model" can appear when the notebook is moved to a different folder)

```
# ID of the map to visualize (The ID is the one in the 2D visualization above)
image_ID = 1 -1 # -1 because counting starts from 0

# read image
file_name = image_data[image_ID]["relaxometryFolder"] + image_data[image_ID]["mapFileName"]
image = itk.imread(file_name)

# view
viewer = view(image, gradient_opacity=0.0, ui_collapsed=False, shadow=False)
viewer
```

GRAPH: Dots represent the average value of fitting maps per image; bars represents the standard deviation

```
rel.show_fitting_graph(image_data)
```

TABLE: Average and standard deviation of fitting maps per image

The table is saved as a .csv file for subsequent analysis

```
rel.show_fitting_table(image_data, output_file_name)
```

References

[1] Borthakur A., Wheaton A.J., Gougoutas A.J., Akella S.V., Regatte R.R., Charagundla S.R., Reddy R. *In vivo measurement of T1rho dispersion in the human brain at 1.5 tesla.* J Magn Reson Imaging. Apr;19(4):403-9. 2004.
[2] Li X., Benjamin Ma C., Link T.M., Castillo D.D., Blumenkrantz G., Lozano J., Carballido-Gamio J., Ries M., Majumdar S. *In vivo T1ρ and T2 mapping of articular cartilage in osteoarthritis of the knee using 3 T MRI.* Osteoarthritis Cartilage. Jul;15(7):789-97. 2007.

Dependencies

```
%load_ext watermark
%watermark -v -m -p SimpleITK,matplotlib,numpy,pandas,scipy,itkwidgets,multiprocessing
```

1. Link to GitHub repository
2. Link to documentation
3. Introduction
4. User inputs

```
input_file_name

./original/
001/BL
left
002/BL
left
003/BL
right
004/BL
left
005/BL
right
006/BL
left
007/BL
left
008/BL
left
009/BL
right
010/BL
left
011/BL
left
012/BL
right



n_of_cores

output_file_name

...
```

Reproducibility of *pyKNEEr*: Jupyter notebook

pyKNEEr

Relaxometry of Femoral Knee Cartilage

1. Link to GitHub repository
2. Link to documentation

Exponential and linear fitting

- *Exponential* fitting is computationally expensive but more accurate
- *Linear fitting* is faster as data are transformed to their log and then linearly interpolated. However, linear fitting is less accurate because the nonlinear logaritmic transform provides larger weight to outliers

The fitting is computed:

- *directly* on the acquired images or after *rigid registration* of the following echo to the first echo
- voxel-wise, i.e. for each voxel the Echo Times (dicom tag: (0018,0081)) are the x-variable and the voxel intensities in each acquisition are the y-variable
- only in the mask volume to have short computation time

3. Introduction

Image information

Inputs:

- `input_file_name` contains the list of the images used to calculate the relaxation maps
- `method` is 0 if fitting is linear, 1 if fitting is exponential
- `registration_flag` is 0 for no registration, 1 for rigid registration
- `output_file_name` contains average and standard deviation of the fitting maps

4. User inputs

```
In [ ]: input_file_name   = "image_list_relaxometry_fitting_DAI1_T2.txt"
method_flag      = 1 ## 0 = linear, 1 = exponential
registration_flag = 1 # 0 = no rigid registration, 1 = execute rigid registration
n_of_cores       = 4
output_file_name  = "exp_fit_aligned_DAI1_T2.csv"
```

Read image data

- `image_data` is a dictionary (or struct), where each cell corresponds to an image. For each image, information such as paths and file names are stored

```
In [ ]: image_data = io.load_image_data_fitting(input_file_name, method_flag, registration_flag)
```

Calculate fitting maps

Align acquisitions

Images are aligned rigidly to remove occational subject motion among acquisitions

Note: This step is optional and can be skipped, given that:

- When images are aligned, the fitting is calculated on interpolated values obtained with rigid registration
- When images are not aligned, the fitting is calculated on original intensities, but images might not be aligned

```
In [ ]: if registration_flag == 1:
            rel.align_acquisitions(image_data, n_of_cores)
```

Compute the fitting

```
In [ ]: rel.calculate_fitting_maps(image_data, n_of_cores)
```

5. Commands with narrative

Visualize fitting maps

2D MAP: For each image, fitting maps at medial and lateral compartments and flattened map

The flattened map is an average of neighnoring voxels projected on the bone surface side of the femoral cartilage

```
In [ ]: rel.show_fitting_maps(image_data)
```

3D MAP: Interactive rendering of fitting maps

(The error message "Error creating widget: could not find model" can appear when the notebook is moved to a different folder)

```
In [ ]: # ID of the map to visualize (The ID is the one in the 2D visualization above)
        image_ID = 1 -1 # -1 because counting starts from 0

        # read image
        file_name = image_data[image_ID]["relaxometryFolder"] + image_data[image_ID]["mapFileName"]
        image = itk.imread(file_name)

        # view
        viewer = view(image, gradient_opacity=0.0, ui_collapsed=False, shadow=False)
        viewer
```

GRAPH: Dots represent the average value of fitting maps per image; bars represents the standard deviation

```
In [ ]: rel.show_fitting_graph(image_data)
```

TABLE: Average and standard deviation of fitting maps per image

The table is saved as a .csv file for subsequent analysis

```
In [ ]: rel.show_fitting_table(image_data, output_file_name)
```

References

[1] Borthakur A., Wheaton A.J., Gougoutas A.J., Akella S.V., Regatte R.R., Charagundla S.R., Reddy R. *In vivo measurement of T1rho dispersion in the human brain at 1.5 tesla.* J Magn Reson Imaging. Apr;19(4):403-9. 2004.
[2] Li X., Benjamin Ma C., Link T.M., Castillo D.D., Blumenkrantz G., Lozano J., Carballido-Gamio J., Ries M., Majumdar S. *In vivo T1ρ and T2 mapping of articular cartilage in osteoarthritis of the knee using 3 T MRI.* Osteoarthritis Cartilage. Jul;15(7):789-97. 2007.

Dependencies

```
In [ ]: %load_ext watermark
        %watermark -v -m -p SimpleITK,matplotlib,numpy,pandas,scipy,itkwidgets,multiprocessing
```

Cartilage Thickness

Separating subcondral surface and articular surface of cartilage

To calculate cartilage thickness, first the cartilage surface is extracted from the binary mask. Then subcondral surface and articular surface are divided in two separate point clouds

```
morph.separate_cartilage_surfaces(image_data, n_of_cores)
```

Visual check

Subcondral bone surface (yellow) and articular surface (blue) are visualized as flattened point clouds. The flattening is with respect to a cylinder interpolated into the cartilage surface [2]

```
morph.show_cartilage_surfaces(image_data)
```

Calculating cartilage thickness

Assign the chosen algorithm

```
morph.algorithm(image_data, thickness_algo)
```

Calculate thickness

```
morph.calculate_thickness(image_data, n_of_cores)
```

Reproducibility of *pyKNEEr*: Jupyter notebook

pyKNEEr

Relaxometry of Femoral Knee Cartilage

1. Link to GitHub repository
2. Link to documentation

Exponential and linear fitting

- *Exponential* fitting is computationally expensive but more accurate
- *Linear fitting* is faster as data are transformed to their log and then linearly interpolated. However, linear fitting is less accurate because the nonlinear logaritmic transform provides larger weight to outliers

The fitting is computed:

- *directly* on the acquired images or after *rigid registration* of the following echo to the first echo
- voxel-wise, i.e. for each voxel the Echo Times (dicom tag: (0018,0081)) are the x-variable and the voxel intensities in each acquisition are the y-variable
- only in the mask volume to have short computation time

3. Introduction

Image information

Inputs:

- `input_file_name` contains the list of the images used to calculate the relaxation maps
- `method` is 0 if fitting is linear, 1 if fitting is exponential
- `registration_flag` is 0 for no registration, 1 for rigid registration
- `output_file_name` contains average and standard deviation of the fitting maps

4. User inputs

```
In [ ]: input_file_name   = "image_list_relaxometry_fitting_OAI1_T2.txt"
method_flag      = 1 ## 0 = linear, 1 = exponential
registration_flag = 1 # 0 = no rigid registration, 1 = execute rigid registration
n_of_cores       = 4
output_file_name  = "exp_fit_aligned_OAI1_T2.csv"
```

Read image data

- `image_data` is a dictionary (or struct), where each cell corresponds to an image. For each image, information such as paths and file names are stored

```
In [ ]: image_data = io.load_image_data_fitting(input_file_name, method_flag, registration_flag)
```

Calculate fitting maps

Align acquisitions

Images are aligned rigidly to remove occational subject motion among acquisitions

Note: This step is optional and can be skipped, given that:

- When images are aligned, the fitting is calculated on interpolated values obtained with rigid registration
- When images are not aligned, the fitting is calculated on original intensities, but images might not be aligned

5. Commands with narrative

```
In [ ]: if registration_flag == 1:
            rel.align_acquisitions(image_data, n_of_cores)
```

Compute the fitting

```
In [ ]: rel.calculate_fitting_maps(image_data, n_of_cores)
```

Visualize fitting maps

2D MAP: For each image, fitting maps at medial and lateral compartments and flattened map

The flattened map is an average of neighnoring voxels projected on the bone surface side of the femoral cartilage

```
In [ ]: rel.show_fitting_maps(image_data)
```

3D MAP: Interactive rendering of fitting maps

(The error message "Error creating widget: could not find model" can appear when the notebook is moved to a different folder)

```
In [ ]: # ID of the map to visualize (The ID is the one in the 2D visualization above)
image_ID = 1 -1 # -1 because counting starts from 0

# read image
file_name = image_data[image_ID]["relaxometryFolder"] + image_data[image_ID]["mapFileName"]
image = itk.imread(file_name)

# view
viewer = view(image, gradient_opacity=0.0, ui_collapsed=False, shadow=False)
viewer
```

6. Visualization of outputs

GRAPH: Dots represent the average value of fitting maps per image; bars represents the standard deviation

```
In [ ]: rel.show_fitting_graph(image_data)
```

TABLE: Average and standard deviation of fitting maps per image

The table is saved as a .csv file for subsequent analysis

```
In [ ]: rel.show_fitting_table(image_data, output_file_name)
```

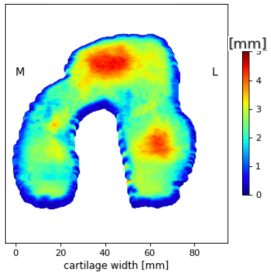
References

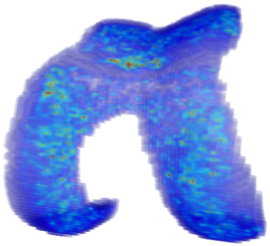
[1] Borthakur A., Wheaton A.J., Gougoutas A.J., Akella S.V., Regatte R.R., Charagundla S.R., Reddy R. *In vivo measurement of T1rho dispersion in the human brain at 1.5 tesla.* J Magn Reson Imaging. Apr;19(4):403-9. 2004.
[2] Li X., Benjamin Ma C., Link T.M., Castillo D.D., Blumenkrantz G., Lozano J., Carballido-Gamio J., Ries M., Majumdar S. *In vivo T1ρ and T2 mapping of articular cartilage in osteoarthritis of the knee using 3 T MRI.* Osteoarthritis Cartilage. Jul;15(7):789-97. 2007.

Dependencies

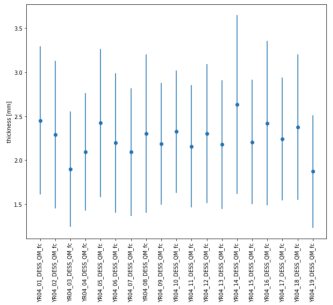
```
In [ ]: %load_ext watermark
%watermark -v -m -p SimpleITK,matplotlib,numpy,pandas,scipy,itkwidgets,multiprocessing
```

Qualitative visualizations



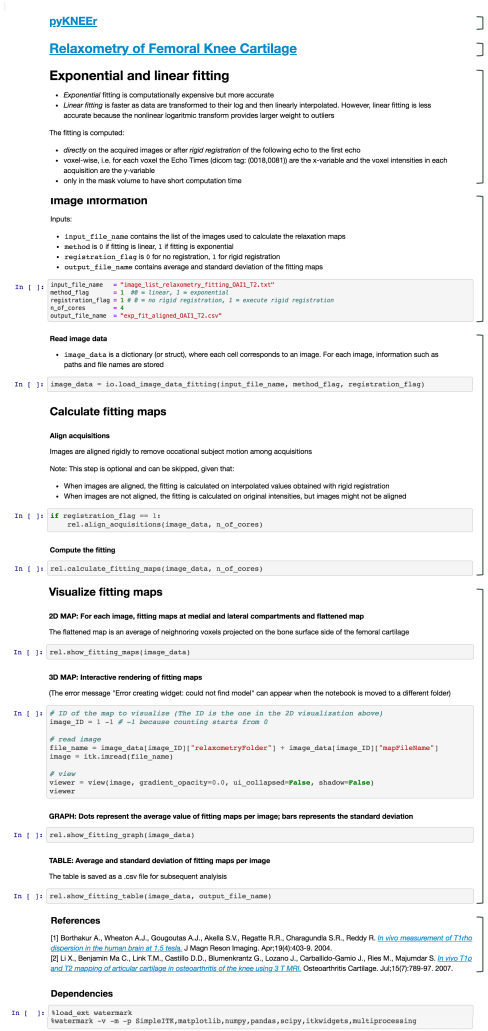


Quantitative visualizations



	Subjects	averageThickness	std.dev
1	YR04_01_DESS_QM_fc_thickness_1	2.45	0.84
2	YR04_02_DESS_QM_fc_thickness_1	2.29	0.84
3	YR04_03_DESS_QM_fc_thickness_1	1.90	0.65
4	YR04_04_DESS_QM_fc_thickness_1	2.09	0.67
5	YR04_05_DESS_QM_fc_thickness_1	2.42	0.84
6	YR04_06_DESS_QM_fc_thickness_1	2.20	0.79
7	YR04_07_DESS_QM_fc_thickness_1	2.09	0.72
8	YR04_08_DESS_QM_fc_thickness_1	2.30	0.90
9	YR04_09_DESS_QM_fc_thickness_1	2.19	0.69
10	YR04_10_DESS_QM_fc_thickness_1	2.32	0.70
11	YR04_11_DESS_QM_fc_thickness_1	2.16	0.69
12	YR04_12_DESS_QM_fc_thickness_1	2.30	0.79
13	YR04_13_DESS_QM_fc_thickness_1	2.18	0.73
14	YR04_14_DESS_QM_fc_thickness_1	2.63	1.02
15	YR04_15_DESS_QM_fc_thickness_1	2.21	0.71
16	YR04_16_DESS_QM_fc_thickness_1	2.42	0.93
17	YR04_17_DESS_QM_fc_thickness_1	2.24	0.70
18	YR04_18_DESS_QM_fc_thickness_1	2.38	0.82
19	YR04_19_DESS_QM_fc_thickness_1	1.87	0.64

Reproducibility of *pyKNEEr*: Jupyter notebook



pyKNEEr

Relaxometry of Femoral Knee Cartilage

Exponential and linear fitting

- *Exponential* fitting is computationally expensive but more accurate
- *Linear fitting* is faster as data are transformed to their log and then linearly interpolated. However, linear fitting is less accurate because the nonlinear logaritmic transform provides larger weight to outliers

The fitting is computed:

- *directly* on the acquired images or after *rigid registration* of the following echo to the first echo
- voxel-wise, i.e. for each voxel the Echo Times (dicom tag: (0018,0081)) are the x-variable and the voxel intensities in each acquisition are the y-variable
- only in the mask volume to have short computation time

Image information

Inputs:

- `input_file_name` contains the list of the images used to calculate the relaxation maps
- `method` is 0 if fitting is linear, 1 if fitting is exponential
- `registration_flag` is 0 for no registration, 1 for rigid registration
- `output_file_name` contains average and standard deviation of the fitting maps

```
In [ ]: input_file_name   = "image_list_relaxometry_fitting_DAI1_T2.txt"
method_flag      = 1 ## 0 = linear, 1 = exponential
registration_flag = 1 # 0 = no rigid registration, 1 = execute rigid registration
n_of_cores       = 4
output_file_name  = "exp_fit_aligned_DAI1_T2.csv"
```

Read image data

- `image_data` is a dictionary (or struct), where each cell corresponds to an image. For each image, information such as paths and file names are stored

```
In [ ]: image_data = io.load_image_data_fitting(input_file_name, method_flag, registration_flag)
```

Calculate fitting maps

Align acquisitions

Images are aligned rigidly to remove occational subject motion among acquisitions

Note: This step is optional and can be skipped, given that:

- When images are aligned, the fitting is calculated on interpolated values obtained with rigid registration
- When images are not aligned, the fitting is calculated on original intensities, but images might not be aligned

```
In [ ]: if registration_flag == 1:
            rel.align_acquisitions(image_data, n_of_cores)
```

Compute the fitting

```
In [ ]: rel.calculate_fitting_maps(image_data, n_of_cores)
```

Visualize fitting maps

2D MAP: For each image, fitting maps at medial and lateral compartments and flattened map

The flattened map is an average of neighboring voxels projected on the bone surface side of the femoral cartilage

```
In [ ]: rel.show_fitting_maps(image_data)
```

3D MAP: Interactive rendering of fitting maps

(The error message "Error creating widget: could not find model" can appear when the notebook is moved to a different folder)

```
In [ ]: # ID of the map to visualize (The ID is the one in the 2D visualization above)
        image_ID = 1 -1 # -1 because counting starts from 0

        # read image
        file_name = image_data[image_ID]["relaxometryFolder"] + image_data[image_ID]["mapFileName"]
        image = itk.imread(file_name)

        # view
        viewer = view(image, gradient_opacity=0.0, ui_collapsed=False, shadow=False)
        viewer
```

GRAPH: Dots represent the average value of fitting maps per image; bars represents the standard deviation

```
In [ ]: rel.show_fitting_graph(image_data)
```

TABLE: Average and standard deviation of fitting maps per image

The table is saved as a .csv file for subsequent analysis

```
In [ ]: rel.show_fitting_table(image_data, output_file_name)
```

References

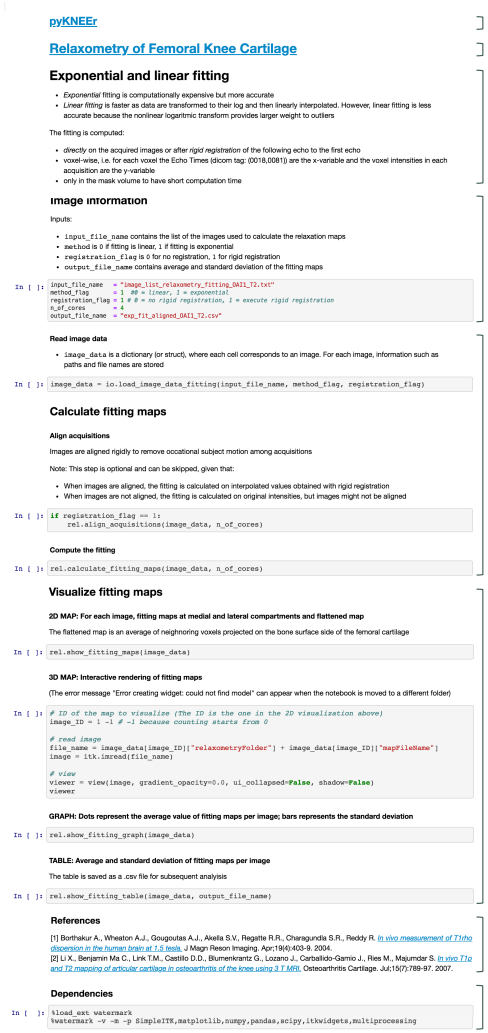
[1] Borthakur A., Wheaton A.J., Gougoutas A.J., Akella S.V., Regatte R.R., Charagundla S.R., Reddy R. *In vivo measurement of T1rho dispersion in the human brain at 1.5 tesla.* J Magn Reson Imaging. Apr;19(4):403-9. 2004.
[2] Li X., Benjamin Ma C., Link T.M., Castillo D.D., Blumenkrantz G., Lozano J., Carballido-Gamio J., Ries M., Majumdar S. *In vivo T1ρ and T2 mapping of articular cartilage in osteoarthritis of the knee using 3 T MRI.* Osteoarthritis Cartilage. Jul;15(7):789-97. 2007.

Dependencies

```
In [ ]: %load_ext watermark
        %watermark -v -m -p SimpleITK,matplotlib,numpy,pandas,scipy,itkwidgets,multiprocessing
```

1. Link to GitHub repository
2. Link to documentation
3. Introduction
4. User inputs
5. Commands with narrative
6. Visualization of outputs
7. References

Reproducibility of *pyKNEEr*: Jupyter notebook



pyKNEEr

Relaxometry of Femoral Knee Cartilage

Exponential and linear fitting

- *Exponential* fitting is computationally expensive but more accurate
- *Linear fitting* is faster as data are transformed to their log and then linearly interpolated. However, linear fitting is less accurate because the nonlinear logaritmic transform provides larger weight to outliers

The fitting is computed:

- *directly* on the acquired images or after *rigid registration* of the following echo to the first echo
- voxel-wise, i.e. for each voxel the Echo Times (dicom tag: (0018,0081)) are the x-variable and the voxel intensities in each acquisition are the y-variable
- only in the mask volume to have short computation time

Image information

Inputs:

- `input_file_name` contains the list of the images used to calculate the relaxation maps
- `method` is 0 if fitting is linear, 1 if fitting is exponential
- `registration_flag` is 0 for no registration, 1 for rigid registration
- `output_file_name` contains average and standard deviation of the fitting maps

```
input_file_name   = "image_list_relaxometry_fitting_DAI1_T2.txt"
method_flag       = 1 ## 0 = linear, 1 = exponential
registration_flag = 1 # 0 = no rigid registration, 1 = execute rigid registration
n_of_cores        = 4
output_file_name  = "exp_fit_aligned_DAI1_T2.csv"
```

Read image data

- `image_data` is a dictionary (or struct), where each cell corresponds to an image. For each image, information such as paths and file names are stored

```
image_data = io.load_image_data_fitting(input_file_name, method_flag, registration_flag)
```

Calculate fitting maps

Align acquisitions

Images are aligned rigidly to remove occational subject motion among acquisitions

Note: This step is optional and can be skipped, given that:

- When images are aligned, the fitting is calculated on interpolated values obtained with rigid registration
- When images are not aligned, the fitting is calculated on original intensities, but images might not be aligned

```
if registration_flag == 1:
    rel.align_acquisitions(image_data, n_of_cores)
```

Compute the fitting

```
rel.calculate_fitting_maps(image_data, n_of_cores)
```

Visualize fitting maps

2D MAP: For each image, fitting maps at medial and lateral compartments and flattened map

The flattened map is an average of neighboring voxels projected on the bone surface side of the femoral cartilage

```
rel.show_fitting_maps(image_data)
```

3D MAP: Interactive rendering of fitting maps

(The error message "Error creating widget: could not find model" can appear when the notebook is moved to a different folder)

```
# ID of the map to visualize (The ID is the one in the 2D visualization above)
image_ID = 1 -1 # -1 because counting starts from 0

# read image
file_name = image_data[image_ID]["relaxometryFolder"] + image_data[image_ID]["mapFileName"]
image = itk.imread(file_name)

# view
viewer = view(image, gradient_opacity=0.0, ui_collapsed=False, shadow=False)
viewer
```

GRAPH: Dots represent the average value of fitting maps per image; bars represents the standard deviation

```
rel.show_fitting_graph(image_data)
```

TABLE: Average and standard deviation of fitting maps per image

The table is saved as a .csv file for subsequent analysis

```
rel.show_fitting_table(image_data, output_file_name)
```

References

[1] Borthakur A., Wheaton A.J., Gougoutas A.J., Akella S.V., Regatte R.R., Charagundla S.R., Reddy R. *In vivo measurement of T1rho dispersion in the human brain at 1.5 tesla.* J Magn Reson Imaging. Apr;19(4):403-9. 2004.
[2] Li X., Benjamin Ma C., Link T.M., Castillo D.D., Blumenkrantz G., Lozano J., Carballido-Gamio J., Ries M., Majumdar S. *In vivo T1ρ and T2 mapping of articular cartilage in osteoarthritis of the knee using 3 T MRI.* Osteoarthritis Cartilage. Jul;15(7):789-97. 2007.

Dependencies

```
%load_ext watermark
%watermark -v -m -p SimpleITK,matplotlib,numpy,pandas,scipy,itkwidgets,multiprocessing
```

1. Link to GitHub repository
2. Link to documentation
3. Introduction
4. User inputs
5. Commands with narrative
6. Visualization of outputs
7. References
8. Dependencies

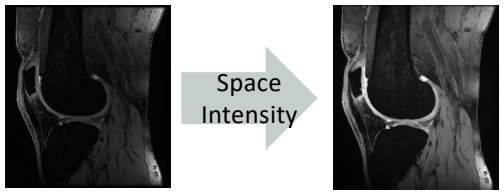
Dependencies

```
%load_ext watermark
%watermark -v -m -p matplotlib,numpy,pandas,scipy

CPython 3.7.1
IPython 7.2.0

matplotlib 2.2.3
numpy 1.16.1
pandas 0.24.1
scipy 1.2.1

compiler   : Clang 4.0.1 (tags/RELEASE_401/final)
system     : Darwin
release    : 17.7.0
machine    : x86_64
processor  : i386
CPU cores  : 4
interpreter: 64bit
```

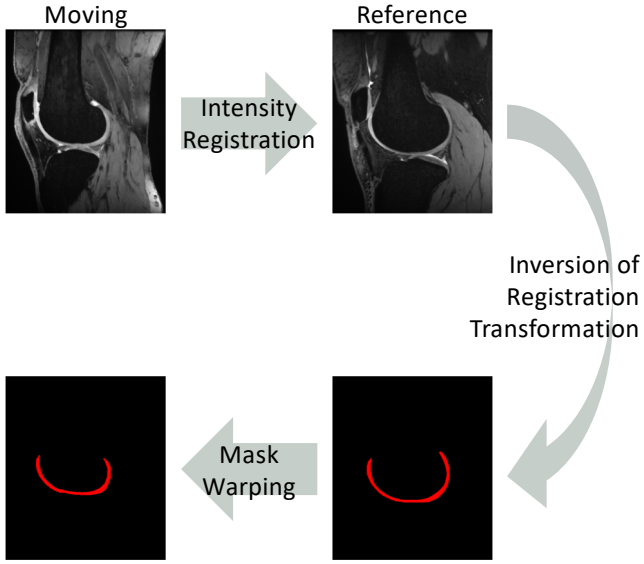
→ Reproducibility of computational environment

Algorithms in *pyKNEEr*

I. Preprocessing



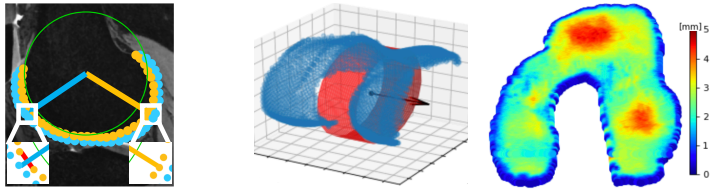
II. Segmentation



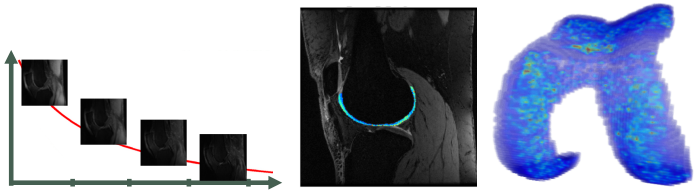
Atlas-based using *elastix*

III. Analysis

Morphology



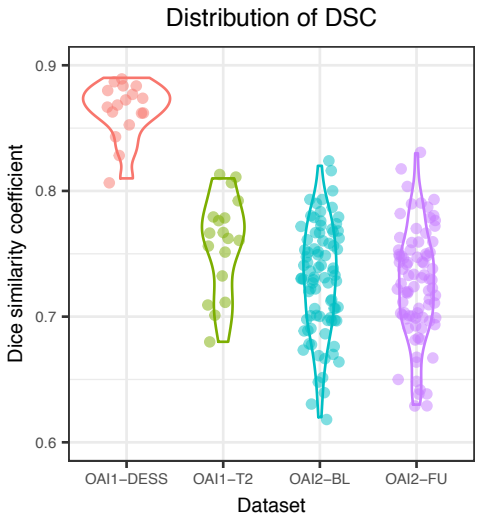
Relaxometry

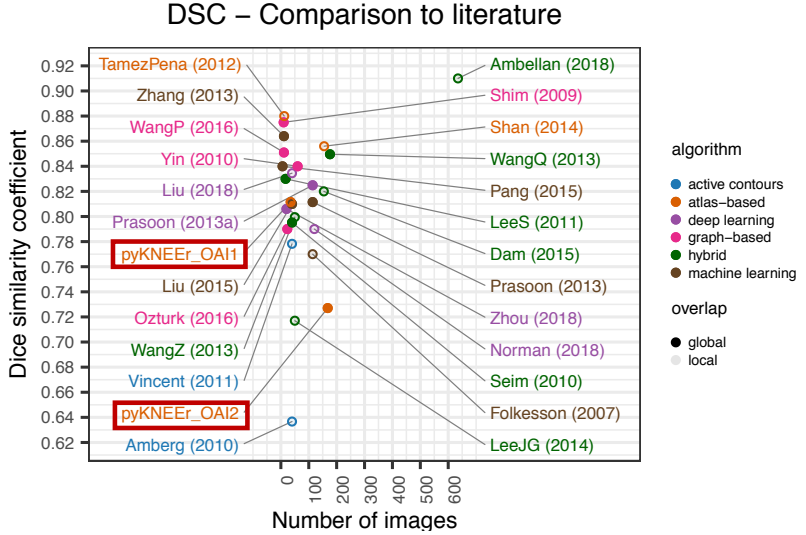


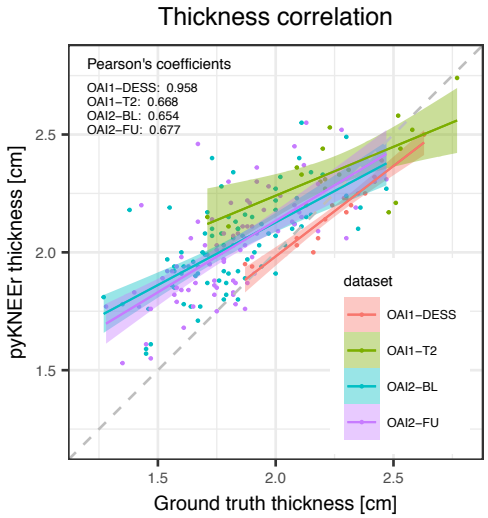
Validation experiments

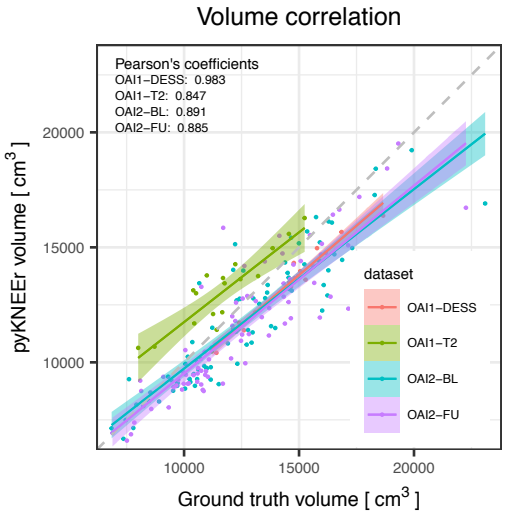
- OAI1: 19 subjects
  - OAI1-DESS
  - OAI1-T2
  - G. Truth: atlas-based (TamezPena 2012)
- OAI2: 88 subjects
  - OAI2-DESS BL
  - OAI2-DESS FU
  - G. Truth: AAM (Vincent 2011)

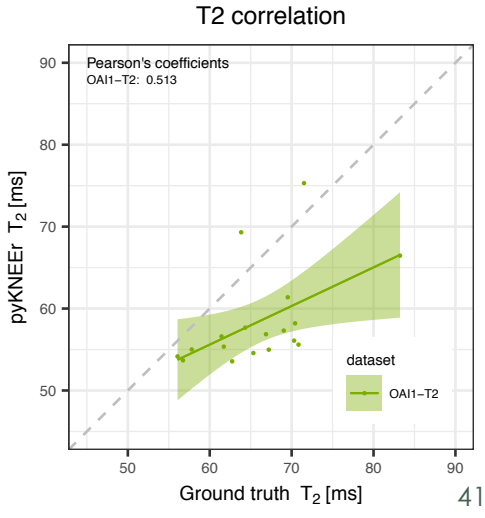
Data on Zenodo
Code on GitHub
Analysis on binder
Paper on bioRxiv









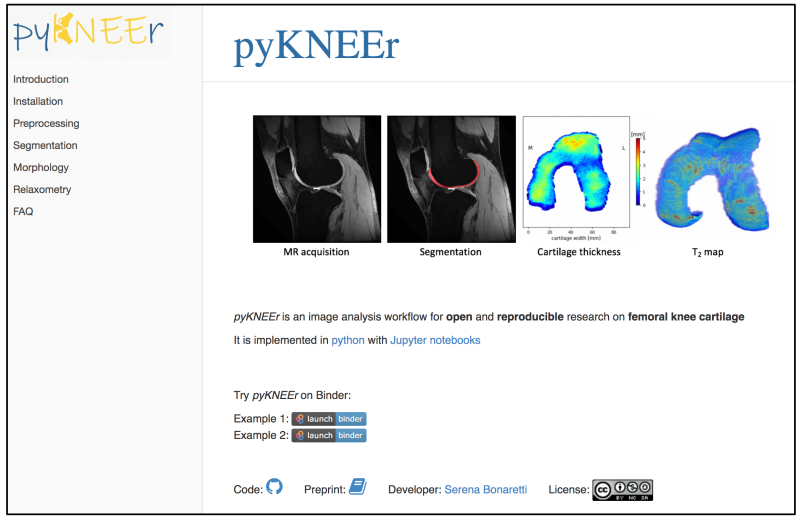


Discussion

- *pyKNEEr* can facilitate transparent research
  - Open source and reproducible
  - Notebooks can be shared and linked to publications

- Do you want to use *pyKNEEr*?
  - Go to terminal and type: `pip install pyKNEEr`
  - Download Jupyter notebooks from GitHub
  - Follow the instruction on the documentation (demo)

- Do you want to contribute to *pyKNEEr*?
  - Go to GitHub and clone the repository
  - Implement your contribution
    - Segmentation: machine and deep learning, hybrid, …
    - Thickness algorithms: surface normal vectors, potential field lines, …
    - Other knee tissues: tibial cartilage, patellar cartilage, menisci, …
  - Send a pull request and your code will be merged to the main project



https://sbonaretti.github.io/pyKNEEr/

Transparent Quantitative Musculoskeletal Imaging

Serena Bonaretti
https://sbonaretti.github.io/