Department of Civil and Environmental Engineering

Proceedings of the 3rd Open Source Geospatial Research & Education Symposium

Edited by Ari Jolma, Pekka Sarkola, and Lassi Lehto

Aalto University

SCIENCE + TECHNOLOGY

CONFERENCE PROCEEDINGS

THIRD OPEN SOURCE GEOSPATIAL RESEARCH & EDUCATION SYMPOSIUM (OGRS)
Espoo, Finland, 10-13 June 2014

Teaching spatio-temporal analysis and efficient data processing in open source environment

Giuseppe Amatulli[1], Stefano Casalegno[2], Remi D'Annunzio[3], Reija Haapanen[4], Pieter Kempeneers[5], Erik Lindquist[3], Anssi Pekkarinen[3], Adam M. Wilson[1], and Raul Zurita-Milla[6]

[1]Department of Ecology and Evolutionary Biology, Yale University, New Haven, USA
[2]Environment and Sustainability Institute, University of Exeter, Penryn, Cornwall, UK
[3]FAO Forestry Department, Rome, Italy
[4]Haapanen Forest Consulting, Kärjenkoskentie 38, 64810 Vanhakylä, Finland
[5]Flemish Institute for Technological Research (VITO), Mol, Belgium
[6]Faculty of Geo-Information Science and Earth Observation, University of Twente, The Netherlands

Abstract: Geo-data are getting larger and complex; consequently, their analysis requires new capabilities both in computer hardware and programming skills. We describe the effectiveness of teaching several open source programming languages to analyze spatio-temporal using Linux OS installed in a Virtual Machine (VM) and/or as LiveUSB. Courses organized in developed and developing countries haven shown positive results including improving students' motivation and engagement, encouraging experimentation and collaboration, and utilizing the allocated time for classes more effectively. Beside the tremendous effort of the developers to maintain and improve the functionality open source geo-spatial software more energy should be invested in organize courses to enlarge the use of this tools.

Keywords: Open source programming, teaching programming, big data

1 Introduction

In recent years there has been an explosion of geo-datasets derived from an increasing number of remote sensors, field instruments, sensor networks, and other GPS-equipped "smart" devices. Geo-datasets are available at increasingly fine spatio-temporal grains, which facilitate addressing old and new research questions across larger domains with increased resolution and unprecedented rigor. However, many of these datasets, especially those labeled as "big data", are complicated and their analysis requires data processing and analysis skills beyond those taught in basic courses (Donovan 2008). Furthermore, some data sets can only be efficiently analyzed using cluster and/or cloud computing facilities, such as Google Earth Engine, or NASA NEX. Access to these increasingly powerful systems generally requires knowledge of at least one programming language.

Indeed, as many of the potential users of these new data sets have limited access to often expensive proprietary software as well as to high speed internet and efficient hardware, they use desktops and laptops with limited processing power for their daily work. Free and open-source software under the Linux Operating System (OS) can address this gap, offering a powerful alternative to proprietary software which can be used in desktop PCs, laptops, low-cost computing hardware such as the RaspberryPi, and is also the standard OS on most scientific computer clusters. Thus proficiency in free and open-source software enables increased hardware efficiency, access to freely accessible code, and freedom from licensing constraints. These have the combined effect of increasing research transparency and reproducibility and advancing the ability of the geo-data community to analyze complex data sets.

Open source software also enables broad and relatively inexpensive access to scientific workflows, which is especially important in developing countries (Camara & Fonseca 2007). The scientific community in many areas is limited by technical skills, modern computer hardware, and software availability. Proprietary software are typically expensive and may not work in old hardware. Internet connections can be slow and inconsistent and frequent power outages may occur. In this situation, carrying out the computations in a cloud environment is a powerful option. Also, sending a script (text file), which performs the desired image processing tasks is lighter and less prone to cuts in electricity and internet than doing the processing in-house (Roy et al. 2010). For example, cloud-processing would solve the problem of downloading and processing Landsat from countries with low-bandwidth connections (Roy et al. 2010).

This paper presents our experiences in teaching spatio-temporal analysis and big data processing of geo-spatial data using multiple open source programming languages integrated in a Linux OS installed in a Virtual Machine (VM) and/or as LiveUSB. We describe our experience in teaching in academic institutions, research centres, and national or international organizations in developed and developing countries. We describe the effectiveness of teaching several programming languages, including course expectations and outcomes, in courses ranging from three hour workshops to one/two week intensive training to full semester-long traditional academic courses.

Section 2 illustrates the importance and the difficulty of learning programming skills to develop geo-spatial applications. Section 3 describes the design, organization, teaching approach and the technical aspects required to organize courses in open source geo-spatial programming. Section 4 reports on feedback from surveys collected during the courses and pros/cons of the implemented teaching methods with special attention paid to each participant's learning experience and how they relate to a priori expectations.

2 The importance and difficulty of programming in environmental sciences

Mastering skills in analyzing spatio-temporal data is fundamental for most environmental and engineering studies. This is why academic institutions have incorporated or increased basic GIS and remote sensing courses in their traditional and online curricula (Govorov & Gienko 2013). Nevertheless, these courses are mostly based on proprietary and/or graphical user interface-based (GUI) software. Graduate students in environmental sciences are rarely trained in the programming skills required to design and implement (complex) geo-spatial workflows. There are few example of academic institutions with courses in programming applied to environmental analysis which address only one programming language (e.g. Python and/or R) at introductory level. In the absence of targeted courses, learning these skills often requires several years of independent learning and strong motivation where personal experiences are accumulated and assimilated.

Learning programming languages enhances students’ practical problem-solving skills, potentially inspires new research questions and ideas stimulating critical, analytical and high-level thinking and promotes creative synthesis to get a first draft of task (Britto & Sá-Soares 2014). Programming allows the automation of repetitive procedures in a research workflow, but can also be used to test new hypotheses “on the fly”. For example, experienced developers frequently write new functions and algorithms years before they are assimilated into graphical user interfaces by major software companies or become operational additions to open source software. In this way, learning programming skills enables researchers to use and share custom and state-of-the-art analytical algorithms. Thus, programming enables an interactive approach to learning, where algorithms and new ideas can be immediately explored and tested by changing the code. As students advance, they often discover more sophisticated methods of analysis and workflow organization, iteratively upgrading their script routines with more complex geo-functions and improving processing speed and use of available computing resources. Thus, learning a programming language is much more than learning how to write code. In some ways, it teaches students how to think scientifically and develop transparent scientific workflows.

The inherent difficulty in learning programming is not new in the literaure. Studies have discussed several problems in teaching programming (Cook 2008), the inherent difficulty of programming (Teague & Roe 2008) and controlling what kind of mistakes students are more likely to make (Spohrer & Soloway 1986). Generally speaking, a programmer needs to instantiate abstract programming concepts and techniques to solve particular computational problems (Britto & Sá-Soares 2014). This is considered one of the most difficult aspects of programming. Applying abstract conceptualization for solving environmental and engineering issues/studies demands active experimentation with writing the code and reflective observations of the obtained results, keeping in mind the theory beyond the environmental problem (Giraffa et al. 2014). In other words you have to be able to write the code and also assess the results and the general data trend based on your expert knowledge of the subjects.

Many graduate students, especially those in environmental studies, do not have any programming experience or knowledge in how to deal with "big data" prior to starting a PhD research program or an environmental career requiring knowledge in programming (Tan et al. 2009, Donovan 2008). Many have the impression that programming is difficult to learn which may unconsciously prevent students from getting started (Tan et al. 2009). Besides the psychological dimensions, there are technical aspects which further discourage student's ability to start learning. Indeed, even if the online documentation is exhaustive in terms of quality and quantity, the users may encounter problems in learning an open-source language, including: 1) shifting to a new OS (e.g. Linux) requires time and effort, 2) learning new patterns and commands required to perform previously normative work is difficult, 3) installing necessary software is not always straightforward, 4) software updating, likewise, is not always simple.

3 Methods: developing open source geo-spatial programming course

To bridge the gaps in training and skills that are critical to perform cutting edge analyses of complex geo-data, we began organizing intensive training courses for graduate and postgraduate researchers and technicians. The courses target those who already possess basic knowledge in geospatial sciences therefore they focus on applied processing with no introductory GIS/RS lessons.

3.1 Main aspects of the course

Our primary tool in open source geo-spatial programming courses is a standardized Linux OS environment with pre-installed software packages and readily available tutorials and exercises. In order to facilitate organizing training courses and to expedite the learning process all the documentation and exercises are freely accessible via the internet (through wikis at www.spatial-ecology.net and www.openforis.org/wiki). The tutorials and exercises are contributed by researchers and experts who use open source tools in their daily work. The approach to teach data analysis is unique, integrating multiple programming languages such as

AWK, BASH, Python, R, GRASS, and other software to build workflows. Our initiative complements existing communities, such as OSGeo by providing specific tutorials and tools that help users to analyze their data and summarize results. The courses focus on automated processing and teach basic programming concepts for using command-line utilities to process large data sets. With simple scripts, we show how to automate necessary tasks and adapt existing programs for specific problems to gain higher efficiency and improved performance. From an outside perspective, the course program may be perceived as covering only technical aspects (e.g. computer programming or code writing). Nevertheless, scientific aspects are integrated as soon as the participants start to use the tools to solve specific tasks related to environmental issues.

Most scripts we use are based on powerful geo-tools that act as modular building blocks for the task at hand. Rather than focus in one programming language or software package, we give an introduction to several tools and languages and integrate them according to their features and strengths. Each language/tool is explained using a common structure: 1) language syntax including specification of various flags and options, 2) accessing available documentation, 3) identifying and explaining the typical problems in data processing and the procedures to solve them (debugging), 4) structuring a script to connect various tools and/or languages, 5) working with the output 6) assessing the results and evaluating them based on a full knowledge of the problem. Furthermore, cloud computing and remote server access are considered major milestones in processing large amounts of data, while multi-core computing allows several processes to run simultaneously. Near the end of the course, and only in case of outstanding course participants, we also give an introduction for using the tools in a cluster computing environment by transforming a simple “for loop” into a “multicore for loop” to allow simultaneous processing of massive data processes. We address this topic using specific R libraries (*foreach, parallel*) and in BASH (*xargs, qsub*).

Providing to each course participant a version of “ready-to-usue” Linux OS including all necessary software and course materials ensures: 1) minimum software installation effort 2) standardized working environment 3) easy portability to other computers, and 4) a gentle transition from the host OS to guest OS which does not intimidate beginning users. Furthermore, participants are able to create their own workspace for data and scripts within the Linux OS VM/LiveUSB, with the ability to precisely reproduce the computing environment for completing class experiments at home or on any other computer. From the trainers’ point of view, using a common cloned Linux OS on each participant's computer facilitates group problem-solving, debugging, and testing commands during the training. Furthermore, solutions tested on one machine can be easily transferred to other students. Specifically, for training we use an ad-hoc customization of the Ubuntu distribution with free and open source remote sensing, GIS, and statistics software pre-installed, including sample geo-data, scripts, and exercises directly linked to the web content.

Selection of the teaching environment depends mainly on the available hardware resources. If efficient PCs with more than 4GB of memory are available, VMs solutions can be efficiently applied. VMs are commonly used by developers and advanced programmers (Nieh & Vaill 2006) but not commonly used for teaching purposes. With older hardware and less memory, a Live system, in which the entire OS and required software is contained either in a USB or CD format, is a more robust solution as it can utilize all the hardware resources of the host computer yet does not use valuable and likely scarce hard-disk space for additional installations. However, the VM approach may be recommendable for users with sufficient hardware as it allows them to use two systems simultaneously. This may facilitate the transition from proprietary software to open source. The Linux OS VM/LiveUSB, in combination with the resources on the wiki web application, can also aid self-taught programmers. Indeed, users can download the Linux OS from the web application and use the scripts, data and web tutorials for self-learning even if they do not have access to an organized course. The materials, data, and software are all published under a Common Public License agreement, so that they can be redistributed or installed on other computers.

In the longer courses, we also mentor participants in personal projects typically related to their PhD/Master thesis. We select the subset of programming tools, idioms, and procedures that would best serve the participants' needs and suggest proper resources (e.g., sample code, blog posts, online tutorials, etc). We request the participants to build up a procedure for their own case study using the tools learned in class. This encourages the participants to focus on the coding and workflow aspects of their projects, having already assimilated the scientific aspect. Moreover, working on their own project increases the motivation in learning something that can be extremely relevant for their career and personal development.

3.2 Software tools and support material

In this section we describe several of the software tools incorporated into the course. In addition to commonly used geo-spatial software such as GRASS (reference) and QGIS (reference), we also incorporate several command-line utilities (such as the Open Foris Geospatial Toolkit and pktools) and the R statistical language which has a rapidly growing library of geo-spatial functionality. We recognize that mastering a programming language often represents a long-term learning process. The use of a variety of new tools help students to understand that the same task can be carried out in many ways, often using a different combination of tools. Exposing the participant to a number of new tools helps them also to understand that becoming an expert user does not require committing everything to memory. In fact, learning how to look for information as required is much more important than mastering a single software package. However, using the tools developed by some of the authors of this paper has allowed us to have the best possible response time as new functionality can be added on-the-fly during the course.

The tools we have chosen to include in our Linux OS distributions are mean operative processing of geo-spatial data sets, primarily for remote sensing, GIS application and spatial statistical analysis. Furthermore, many geo-tools are implemented primarily in a compiled programming language (such as C or C++) with high performance (e.g. processing speed and memory management) while providing a relatively simple user interface via the command line. In fact, these and similar tools have been used, for example, to process first high resolution forest maps for Europe (Kempeneers et al. 2011, Pekkarinen et al. 2009). In addition, some of them are currently in use in more than 20 countries by various institutions and individuals and are being used by a number of developing countries to map their natural resources. During the courses, the tools are used to address participants' problems in such a way that they can see how learning new skills benefits their performance and productivity. We also emphasize that they are free to use the tools for their own purposes including private consulting. In particular, we teach the use of Open Foris Geospatial Toolkit, pktools and R/BUGS for computational modeling. Here below we describe their main features.

Open Foris Geospatial Toolkit is part of the FAO's Open Foris initiative aimed at helping developing countries assess and monitor their forest resources, especially forest cover change. The tools are designed for efficient and automated interpretation of remote sensing data, and can be used for common image processing tasks such as radiometric normalization, image algebra, and image segmentation and classification. It contains modules also for change detection, extraction of image statistics, image filtering, feature extraction and gap-filling. The toolkit also allows interaction with R, which provides access to more sophisticated classification algorithms such as Random Forest. The tools are divided into stand-alone C-programs and scripts. Many of the stand-alone programs and scripts use GDAL/OGR libraries and command line utilities. Open Foris Geospatial Toolkit wiki has been visited more than 8000 times (Mar 2014) and the toolkit has users in more than 20 countries.

Pktools is another open source software toolbox that is implemented in a similar way as the GDAL/OGR utilities. This means all programs are executed from the command line using various options. Though some overlap exists between their functionalities, pktools is more oriented towards remote sensing and image processing applications than GDAL/OGR. Typical tasks include: image (morphological and other) filtering, compositing multiple images according to different rule sets. Typical machine learning algorithms are also implemented for image classification or data regression, including feature selection algorithms, artificial neural networks and support vector machines. Like GDAL/OGR utilities, maintaining a simple command line interface allows for a modular approach that can easily be integrated in other scripting languages.

We also introduce spatial statistics using with R and BUGS. We have developed several modules on spatial statistical modeling to enable inference of patterns and processes from the data. The growing size and complexity of geo-spatial datasets has facilitated increasingly sophisticated models of environmental processes. In order to evaluate and critically apply these models, it is vital that complete workflows are documented and repeatable. Our approach uses geo-spatial software mentioned above for data pre-processing and organization followed by R and various BUGS languages to fit statistical models and produce summary output. Examples of this type of approach are available in a public GitHub repository.

Beside these main geo-tools we also mention the potential use of other open source software such as CDO, Orfeo Toolbox, ImageMagick, and gnuplot pointing out their features and their aspects for solving specific tasks. These widen the view of the open source capability without adding new languages and commands. By scripting each stage of the analysis, from the 'raw' data to graphical and tabular summaries, we teach how to develop a transparent analytical workflow that is also easy to share and repeat. In this way, we provide students with the tools they need to understand, run, and share their complete analysis.

4 Results and discussion: course evaluation

Since 2008, more than 400 students and technicians have participated in trainings organized in by the authors of this paper. These training/courses took place in Italy, Spain, Denmark, Peru, Brazil, Paraguay, Mexico, Ecuador, Uganda, The Republic of South Sudan, UK, the Netherlands, Tanzania and the USA, in academic institutions, research centres, and in national and international organizations. The background of the course participants included some exposure to GIS, remote sensing and/or database software but the Linux OS was new to most of them.

Though the initial learning curve to the approach followed in our courses is somewhat steeper than that of proprietary software packages with graphical user interfaces, we are convinced this effort is well spent. The purpose of the courses is to guide geo-data users through basic programming tasks so that they are quickly able to perform similar tasks independently. At the end of each course we asked the participants to fill in a questionnaire so that we could better understand the impact of the course. The questionnaire consisted of four main parts: 1) general information about the student's motivation, 2) the capacity of the students to independently continue to develop skills in this area, and 3) the utility of using new tools (OFGT, pktools) for their work, and 4) detailed feedback about course organization and effectiveness. We also collected valuable information from personal comments. Participants had the option to remain anonymous and to skip questions so the number of respondents varies for each question.

The primary motivation for completing the course was to acquire new expertise in geo-spatial data analysis (88%), followed by the desire to improve pre-existent programming knowledge (31%). The results show that 87% of participants are interested in continuing to use open source software after the course. In particular, 83% found the tools related to their professional work; 14% have little professional interest and 3% no professional interest (see Table 1). About 45% of the course participants described Linux OS as indispensable and the remaining participants see its value for daily work. Concerning the self learning attitude, an average of 85% of the participants felt able to independently improve and learn more about open source tools (see Table 2). In particular, the command line tools most assimilated was BASH (82%) probably due to its extensive usage during the exercises including those based on GDAL and GRASS. About 72% of the participants felt they were able to continue to independently improve their knowledge following the course. In the same line, 86% reported they would likely keep using the open-source command line. It is also notable that those willing to continue using QGis is a bit less (75%) compared to the command-line option (see Table 2). This demonstrates that the course was able to transmit the importance and power of the command-line versus graphical user interface to software. With questions in Table 3 we evaluate the use of OFGT and pktools to perform remote sensing analysis. For both of the new tools we report a positive attitude in their use for different image analysis tasks. Finally, while considering the quality of the pedagogic efficiency, 78% participants felt it was well adapted to their needs, skills, and knowledge; 18% believed it was too difficult and 4% found the training boring and/or too basic (see Table 4).

Also, during trainings in developing countries, the government officials and technicians received the training well. There, we focused on understanding the users' operative needs and expectations. In the recent trainings organized in Peru and Uganda, 100% of the participants replied that they are "likely to use" the presented tools in their future work and that they would recommend the course for their colleagues. The feeling after the training is that more effort have to be done to explain GRASS, R, and BASH as compared to AWK, Gdal or to insist on the theoretical aspect of the scientific questions addressed. The supervision effort and the trainers preparation were positively rated by more than 97% of participants. At the same time, the most commonly listed drawback was the lack of time and therefore the participants wished to receive additional training on the subject.

To address this problem, a relatively innovative course design was tested in the Netherlands between December 2013 and January 2014. There we offered a week of intensive face-to-face training (lectures and practicals), followed by a month of self-study where course participants could assimilate the course contents and apply them to datasets/problems related to their own research. This put the participants in control of their own learning and allowed them to focus on tools and techniques directly related to their work. Finally, a second intensive face-to-face training was organized to present specialized tools (e.g. pktools). During this week, we also scheduled time for each participant to present the work done during the self-study period and after that both participants and course teachers could provide formative feedback to consolidate knowledge and enhance the learning process.

Our long term follow-up with some of the students reveal that the impact of the training outside of the academia depends on the institutional commitment. Student working in environments that are unfriendly to open-source software are less likely to adapt the tools and use them to solve their every-day problems because of the lack of software and hardware support. In other environments, it seems to be relatively easy to switch from proprietary software to open-source tools. In cases where that has happened, the use of open source tools has drastically increased among the colleagues of these students as soon as they demonstrated their efficiency.

Table 1: Student's expectations and interest in learning open source geo-spatial programming

Why did you attend a course? (42 participants replied to this questionnaire)			
Acquire new expertise 88%	Improve my skills 31%	Discuss related issues 9%	No similar course found 9%
Do you see any interest in using open source tools in your current and future job? (92 participants replay to this question))			
Yes - 83%	Little - 14%	No - 3%	
How do you rate the use of Linux OS in your daily work? (11 participants replied to this questionnaire)			
Indispensable 45%	Valuable 55%	Of general interest 0%	Not relevant to my activity 0%

Table 2: Ability of participants to independently learn and progress to use open source tools

Do you feel to be able to independently improve and learn more about open source tools?						
Tools	BASH	AWK	GDAL	GRASS	Qgis	R
Answers	73	43	69	71	69	69
Yes (%)	88	79	77	96	97	72
No (%)	12	21	23	4	3	28
Will you use and progress using open source tools in future?						
Answers	74	41	74	77	87	76
Yes (%)	94	92	89	97	75	86
No (%)	6	8	11	3	25	14

Table 3: Overall tools (OFGT, pktools) evaluation considering their use and their implementation in remote sensing analysis

Were your expectation met with regards to the following image processing lessons? (11 participants replied to this questionnaire)					
Image analysis using OFGT	Met entirely (Likely to Use)	Met to some degree (Likely to Use)	Met to some degree (Not likely to use)	Not met at all (Likely to use)	Not met at all (Not likely to use)
General imagery processing (clipping / stacking)	45%	55%	0%	0%	0%
Segmentation	45%	50%	0%	0%	0%
Supervised classification	45%	55%	0%	0%	0%
Verification of the automatic procedure	20%	70%	10%	0%	0%
Statistical analyses	29%	57%	5%	5%	5%

Are you going to use pktools in the future? (14 participants replied to this questionnaire)			
Yes 90%		No 10%	
Have you improved in working with pktools? (16 participants replay to this question)			
Yes 100%		No 0%	
Which part of the pktools would you have liked to see in more in detail? (16 participants replied to this questionnaire)			
Image filtering 50%	Image compositing, merging, subsetting 67%	Image extraction (vector to raster overlay) 73%	Image classification 53%
Do you see potential in the use of pktools? (18 participants replied to this questionnaire)			
Yes 89%	No 11%		
Is the help info understandable? (18 participants replied to this questionnaire)			
Yes but need to be improved 94%	Not understandable 6%		

Table 4: Overall courses evaluation considering the structural format and the trainers performances

Was the course adapt to your skills and knowledge ? (120 participants replied to this questionnaire)			
Well adapted 78%	Advanced and too difficult 18%	Basic and boring 4%	
Which is my prefered learning format ? (54 participants replay to this question)			
Exercise 40%	Theory 23%	Tutorials 23%	Projects 15%
The trainer were clear and prepared? (84 participants replay to this question)			
Yes, very well 60%	Yes, enough prepared 39%	Not enough 1%	
The trainer's supervision during some coding exercises was satisfactory? (82 participants replay to this question)			
Very much 54%	Yes satisfactory 44%	Not enough 2%	

5 Conclusion

This paper presented some experiences providing geo-spatial data processing educational courses to students, professionals, and technicians from a number of different countries using open source image processing tools, a standardized Linux OS environment with software pre-installed, and additional learning materials and tutorials available via web applications. Results confirmed several benefits of this approach including improving students' motivation and engagement, encouraging experimentation and collaboration, and utilizing the allocated time for classes more effectively. Students indicated that they are interested in continuing to use the open source tools. In trainings with a duration of one week the most commonly reported problem in the organization of the course was the lack of sufficient time to fully grasp the concepts of the new computing environment. Many students expressed the need for several days dedicated solely to becoming comfortable with the command-line nature of a completely new OS and interface.

Regarding day-to-day usage of open source tools and software, there is a tremendous effort from the developers to maintain and improve the functionality of the various systems, but our experience shows that usage on a much broader scale could be enhanced through increased formal course offerings as part of all geospatial science curricula. Today, most students in developed countries have enough computing power in their home computers to easily reproduce class experiments and exercises. However, in the developing countries the situation is often challenging due to general lack of computing and internet infrastructure which necessitates that class lessons are processed on old, typically low-performance hardware. Therefore, the selection of an appropriate approach (VM/Live OS) depending on the available computing resources, is a key for satisfactory user experience and a successful course.

Finally, we are confident that with the continuing advances in technology, increased access to internet and the overall improvement in available computing resources, interest in open-source tools and computer programming will subsequently increase and, as such, will be a very important tool in the learning about and processing geospatial data.

References

Brito, M. A. & de Sá-Soares, F. (2014), 'Assessment frequency in introductory computer programming disciplines', *Computers in Human Behavior* 30 (2014), 623–628.

Camara, G. & Fonseca, F. (2007), Information Policies and Open Source Software in Developing Countries. *Journal of the American Society for Information Science and Technology*, 58(1), 121–132.

Cook, W.R. (2008), 'High-level problems in teaching undergraduate programming languages', SIGPLAN Notices, 43, 55–58.

Donovan, S. (2008), Big data: teaching must evolve to keep up with advances. Nature, 455(7212), 461-461.

Giraffa, L.M., Moraes, M.C. & Uden, L. (2014), Teaching Object-Oriented Programming in First-Year Undergraduate Courses Supported By Virtual Classrooms. In The 2nd International Workshop on Learning Technology for Education in Cloud (pp. 15-26). Springer Netherlands.

Govorov, M. & Gienko, G. (2013), 'GIS learning objects: an approach to content aggregation', *International Research in Geographical and Environmental Education*. Vol. 22(2):155-171.

Nieh, J. & Vaill, C. (2005), Experiences teaching operating systems using virtual platforms and linux. In ACM SIGCSE Bulletin (Vol. 37, No. 1, pp. 520-524). ACM.

Kempeneers, P., Sedano, F., Seebach, L., Strobl, P. & San-Miguel-Ayanz, J. (2011), 'Data fusion of different spatial resolution remote sensing images applied to forest type mapping', *IEEE Transactions on Geoscience and Remote Sensing*, 49(12): 4977-4986.

Pekkarinen, A., Reithmaier, L. & Strobl, P. (2009), 'Pan-European forest/non-forest mapping with Landsat ETM+ and CORINE Land Cover 2000 data', *ISPRS Journal of Photogrammetry and Remote Sensing*, 64(2):171-183.

Roy, D.P., Ju, J., Mbow, C., Frost, P. & Loveland, T. (2010), Accessing free Landsat data via the Internet: Africa's challenge. *Remote Sensing Letters*, *1*(2), 111-117.

Spohrer, J. C., & Soloway, E. (1986). Novice mistakes: Are the folk wisdoms correct? Communications of the ACM, 29, 624–632.

Tan, P.H., Ting, C.Y. & Ling, S.W. (2009), Learning difficulties in programming courses: undergraduates' perspective and perception. In Computer Technology and Development, 2009. ICCTD'09. International Conference on (Vol. 1, pp. 42-46). IEEE.

Teague, D. & Roe, P. (2008), Collaborative learning: Towards a solution for novice programmers. Proceedings of the tenth conference on Australasian computing education (Vol. 78). Wollongong, NSW, Australia: Australian Computer Society, Inc.