Computer Programming Skills for Environmental Sciences

Denis Valle, Aaron Berdanier

University Program in Ecology, Duke University, Durham, NC 27708, USA

Citation: Valle, D; Berdanier, A. 2012. Bulletin of the Ecological Society of America, 93(4).

Denis Valle (corresponding author): drv4@duke.edu. University Program in Ecology, Duke University, Durham, NC 27708, USA

Aaron Berdanier: aaron.berdanier@duke.edu. University Program in Ecology, Duke University, Durham, NC 27708, USA

Introduction

Most undergraduate and graduate science degrees recognize the importance of statistics in the training of future scientists and include at least one basic statistics course in their curricula. On the other hand, programming skills are typically seen as a tool just for modelers or quantitative scientists. We dispute this view and argue that programming skills are extremely useful for almost any scientist, particularly with the advent of scripted analysis programs (e.g., Matlab and R). For example, these programming skills enable one to query, preprocess, visualize, and analyze datasets in a much less error-prone way than spreadsheets. Furthermore, these scripting languages allow for a natural documentation of the judgment calls that are often needed when preprocessing the data, being a critical step towards reproducible research (Peng et al. 2006, Borer et al. 2009, Ellison 2010, Reichman et al. 2011, Michener and Jones 2012). Unfortunately, the use of spreadsheets to store and manipulate data is still widespread among scientists, probably because programming skills are not yet part of the formal training of environmental scientists (Jones et al. 2006, Michener and Jones 2012). Here we advocate for the inclusion of programming skills as part of the core curriculum for environmental scientists and describe the challenges involved in doing so.

We start this article by illustrating the general usefulness of programming skills with several tasks that can be efficiently accomplished within these scripted analysis programs. We then report on the challenges of teaching programming skills and the approaches we have found to be useful, based on our own experience teaching programming languages to environmental science graduate students at Duke University. Finally, we conclude with a summary of our thoughts on programming skills for environmental scientists.

Usefulness of programming skills

In this section, we list and discuss a few tasks that can be efficiently accomplished with programming skills. Specific examples of these tasks are provided in Appendix 1.

Data pre-processing

Programming skills are critical for data pre-processing, allowing data to be combined, queried, and summarized. These skills are particularly relevant when using data collected by multiple researchers, which is becoming more frequent as we strive to understand environmental phenomena across larger regions and over longer time scales (Michener et al. 1997, Ellison 2010, Reichman et al. 2011, Michener and Jones 2012). For instance, consistency checks and data visualization allows one to quickly identify odd observations (e.g., spelling errors in species Latin names, birth date that does not match reported age, diameter measurements taken after the tree is reported to have died, or extreme values used to represent missing values).

Many of the tasks in data pre-processing are conceptually simple and can be done in regular spreadsheets. However, this process can be extremely error-prone and time consuming in the absence of programming skills. Furthermore, the benefit of doing these tasks with a scripting language is that these languages automatically document all the inherent judgment calls in data preprocessing (e.g., elimination of trees with odd diameter measurements or substitution of suspicious diameters by missing value) and allow the data preprocessing steps to be modified and reproduced effortlessly (Borer et al. 2009, Ellison 2010). Importantly, scripting languages help with data management by avoiding the proliferation of spread-sheets, a problem that typically occurs when multiple data versions are created (e.g., ‘raw’ vs. ‘edited’ datasets) and/or a data analysis has multiple steps. Unfortunately, few people seem to recognize the importance of formal training on information management skills (see related comments in Nature 2009). This is particularly evident if we consider that data pre-processing is a prior step to using formal statistical tests and yet only the latter is included in regular curricula.

Forward simulations

Critical intuition and insights can often be gained by creating models and running *in silico* (vs. *in vivo*) experiments; some experiments are even unfeasible otherwise (Peck 2004, CSTA Standards Task Force 2011). We believe that *in silico* experiments are particularly powerful didactic tools because it allows a level of interactivity unavailable otherwise (e.g., multiple *in silico* experiments can be run without requiring years of painstaking data collections and considerable funding to set up factorial experiments). For instance, these simulations allow students to immediately assess the importance of several model parameters. Furthermore, simulations show that apparently reasonable assumptions often lead to unrealistic long-term outcomes, challenging students to think of which assumptions are over-simplistic or all together wrong. It is our opinion that there would be much to gain if graduate students (as future professors) learned how to create these simulations given their importance as educational tools.

Numerical optimization

Numerical optimization is routinely used in a wide range of fields, such as statistics (e.g., to find maximum likelihood estimates), finance (e.g., minimize risk), and transportation (e.g., finding the fastest route from point A to B). Traditional applications of optimization in environmental sciences have focused on natural resource management, such as forestry and fisheries (e.g., determining maximum sustainable yield), but optimization has been increasingly used for research themes as diverse as protected area design (e.g., determining optimal size, location and number of protected areas to minimize extinction) (e.g., Wintle et al. 2011) and optimal plant trait combination (e.g., Marks and Lechowicz 2006). Unfortunately, in the absence of at least some computer programming skill, this powerful tool is inaccessible for environmental scientists.

Statistics

Computer intensive methods in statistics have arisen to liberate scientists from rather restrictive distributional assumptions (e.g., normal distribution) and statistics choices (e.g., medians instead of means), allowing the fit of models and assessment of uncertainties in ways that would be almost impossible otherwise. These methods include cross-validation, bootstrap, jackknife, permutation tests, Markov Chain Monte Carlo algorithms (MCMC), among others. Despite their relatively long history (e.g., Diaconis and Efron 1983), the most common approach in environmental sciences is still to transform the data and the research question so that one can use standard statistical tests (e.g., linear regression, ANOVA) instead of the other way around (i.e., modifying the method to suit the data and research question) (Ellison and Dennis 2010). An additional benefit of programming some of these computer intensive statistical methods is that it often makes one reflect on the inner workings of the statistical procedure. Because these methods tend to be more intuitive than their analytical counterparts, this process often enhances the understanding of hard-to-grasp concepts (e.g., p-values and confidence intervals) (Ellison and Dennis 2010).

Challenges and approaches in teaching programming skills

We have listed a few tasks that scripted analysis programs can do, which by no means represent the entire functionality of these tools. Nevertheless, these tasks illustrate how versatile and useful these tools can be for an environmental scientist, if he can master it. We provide examples of each one of these tasks in Appendix 1 together with pseudo-code. The common thread to all these examples is that they harness the power of computers to do repetitive tasks using predominantly customized code. We emphasize customized code because one often faces problems that are too specific to be solved by existing libraries/packages or canned programs.

How can environmental science students learn programming skills? We have taught a programming skills course for the past two years, which originated from a professor realizing that these skills could positively affect several existing quantitative courses here at Duke University that also employed scripted analysis programs (e.g., watershed hydrology, natural resource economics, water quality management, atmospheric chemistry, energy systems). The main idea was that a course devoted specifically to programming skills would free these other quantitative courses to spend more classroom time on discipline specific topics, rather than on teaching students to program. Thus, our course was implemented with the specific goal of teaching programming skills as a generic tool, with a heavy emphasis on programming logic rather than program specific syntax. Because of the interdisciplinary nature of environmental sciences, with different programming languages being used by different disciplines (e.g., remote sensing and GIS analysts favor Python, statisticians favor R, econometricians and engineers favor Matlab), the emphasis on logic, rather than syntax, was important.

A widely recognized problem in teaching how to program is that several students feel intimidated by the steep learning curve (Anderson et al. 2011). One can ease this learning process by building on students' existing knowledge. For instance, our approach has been to focus on a wide range of data management and environmental science problems that are familiar to most of our students and that can be tackled with simple algorithms (e.g., Appendix 1). Similarly, we can first show how a particular task would be accomplished in a regular spreadsheet or other GUI driven program, to then illustrate how it would be done using a programming language. We stimulate students to create algorithms that work, rather than focusing on improving algorithm efficiency or elegant code. The dichotomy between programming syntax and logic is exploited by asking students to write pseudo-code, describing how the problem can be divided into simpler pieces and how each piece will be dealt with, before effectively attempting to implement a particular algorithm. Visual depictions, using workflow / data-flow graphs (e.g., as in ArcMap model builder or in Ellison et al. 2006) or work sheets (Hasni and

Lodhi 2011), are often helpful during this process of writing pseudo-code. We believe that these course characteristics are critical to alleviate the steep learning curve that students experience. Unfortunately, environmental science programs rarely offer courses on programming skills (Box 1) and standard introductory programming classes offered by computer science departments seem to focus more on algorithmic efficiency and on examples that are too abstract to be applicable to environmental science problems. We provide more details regarding our course structure in Appendix 3.

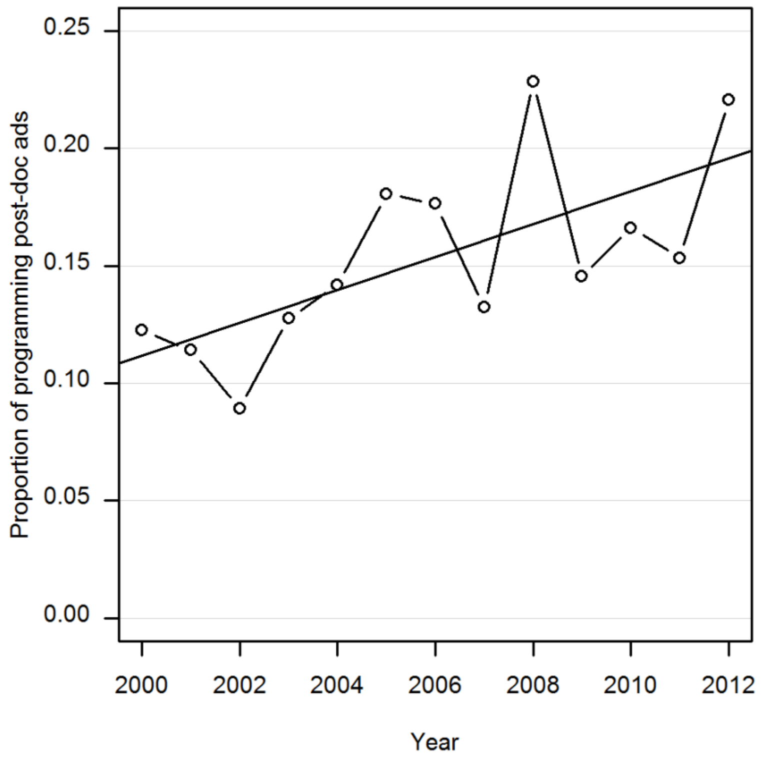
Conclusion

It has been our experience that programming skills dramatically open student's analytical horizons and quickly become indispensable in their toolkit. We acknowledge that defining core courses for environmental sciences is challenging given its multidisciplinary nature. However, we emphasize that computer programming skills are pervasive throughout environmental science, as evidenced by the range of quantitative courses that employ some level of programming (e.g., watershed hydrology, natural resource economics, water quality management, atmospheric chemistry, energy systems) and by the demand for these skills in the work force (Box 1). These tools allow data to be queried and graphed in novel ways and facilitate the customization of optimization routines, forward simulation and computer intensive statistical procedures. As a result, scientists are empowered to expand the approaches used in their field, often in ways that could not have been foreseen by the inventers of the original methodology. Perhaps more importantly, scripting languages document the innumerous data pre-processing and analysis steps. If these languages do not become part of the formal training of environmental scientists, current emphasizes on data sharing, reproducible research, and even statistical fluency (Cassey and Blackburn 2006, Jones et al. 2006, Allison 2009, Borer et al. 2009, Ellison and Dennis 2010, Vision 2010, Whitlock et al. 2010, Reichman et al. 2011, Whitlock 2011, Michener and Jones 2012) is unlikely to be effective. We believe that these programming skills will be critical to tackle

21st century environmental problems and should be part of the core curriculum of environmental sciences.

Box 1. Supply and demand for environmental scientists with computer programming skills.

We performed two surveys to obtain a rough idea of the supply and demand for environmental scientists with computer programming skills. The details regarding these surveys are given in Appendix 2. We found an overall lack of educational opportunities regarding these skills. Only 4 of the 20 top environmental science programs in the United States offered courses explicitly mentioning 'programming' in their course description. The lack of supply of environmental scientists with computer programming skills contrasts sharply with the increasing demand for these skills. Based on post-doctoral advertisements, we found that the demand for environmental scientists with programming skills rose from 12% in 1999 to 22% in 2011, with an overall average of 16% (Box Fig. 1).



Box Fig. 1. The proportion of post-doctoral advertisements that refer to programming skills increased substantially over the past 13 years. Continuous line depicts the fitted linear regression.

Acknowledgements

We thank Andy Read and Varun Swamy for kindly sharing their data on whale movement and tropical trees, respectively. We thank Gaby Katul for his contribution in conceptualizing and implementing our programming course, and Jim Clark for creating the original simulations of competing species that inspired one of our examples. Finally, we are grateful for comments provided by William Michener, Susan Rodger, Ben Bolker, Jason Roberts, John Fay, Varun Swamy, Gretchen Addington, Maria Soledad Benitez, Jim Clark, Andria Dawson, Rob Schick, and Ben Vierra.

References

Allison, D. B. 2009. The antidote to bias in research. Science **326**:522.

Anderson, M., A. McKenzie, B. Wellman, M. Brown, and S. Vrbsky. 2011. Affecting attitudes in first-year computer science using syntax-free robotics programming. ACM Inroads **2**:51-57.

Borer, E. T., E. W. Seabloom, M. B. Jones, and M. Schildhauer. 2009. Some simple guidelines for effective data management. Bulletin of the Ecological Society of America **90**:205-214.

Cassey, P. and T. M. Blackburn. 2006. Reproducibility and repeatability in ecology. Bioscience **56**:958-959.

CSTA Standards Task Force. 2011. CSTA K-12 Computer Science Standards. Computer Science Teachers Association (CSTA).

Diaconis, P. and B. Efron. 1983. Computer-Intensive Methods in Statistics. Scientific American:116-130.

Ellison, A. M. 2010. Repeatability and transparency in ecological research. Ecology **91**:2536-2539.

Ellison, A. M. and B. Dennis. 2010. Paths to statistical fluency for ecologists. Frontier in Ecology and Environment **8**:362-370.

Ellison, A. M., L. J. Osterweil, L. Clarke, J. L. Hadley, A. Wise, E. Boose, D. R. Foster, A. Hanson, D. Jensen, P. Kuzeja, E. Riseman, and H. Schultz. 2006. Analytic webs support the synthesis of ecological data sets. Ecology **87**:1345-1358.

Hasni, T. F. and F. Lodhi. 2011. Teaching problem solving effectively. ACM Inroads **2**:58-62.

Jones, M. B., M. Schildhauer, O. J. Reichman, and S. Bowers. 2006. The new bioinformatics: integrating ecological data from the gene to the biosphere. Annual Review of Ecology, Evolution, and Systematics **37**:519-544.

Marks, C. O. and M. J. Lechowicz. 2006. Alternative designs and the evolution of functional diversity. The American Naturalist **167**:55-66.

Michener, W. K., J. W. Brunt, J. J. Helly, T. B. Kirchner, and S. G. Stafford. 1997. Nongeospatial metadata for the ecological sciences. Ecological Applications **7**:330-342.

Michener, W. K. and M. B. Jones. 2012. Ecoinformatics: supporting ecology as a data-intensive science. Trends in Ecology and Evolution **27**:85-93.

Nature. 2009. Data's shameful neglect. Nature **461**:145-145.

Peck, S. L. 2004. Simulation as experiment: a philosophycal reassessment for biological modeling. Trends in Ecology and Evolution **19**:530-534.

Peng, R. D., F. Dominici, and S. L. Zeger. 2006. Reproducible epidemiologic research. American Journal of Epidemiology **163**.

Reichman, O. J., M. B. Jones, and M. Schildhauer. 2011. Challenges and opportunities of open data in ecology. Science **331**:703-705.

Vision, T. J. 2010. Open data and the social contract of scientific publishing. Bioscience **60**:330-330.

Whitlock, M. C. 2011. Data archiving in ecology and evolution: best practices. Trends in Ecology and Evolution **26**:61-65.

Whitlock, M. C., M. A. McPeek, M. D. Rausher, L. Rieseberg, and A. J. Moore. 2010. Data archiving. The American Naturalist **175**:145-146.

Wintle, B. A., S. A. Bekessy, D. A. Keith, B. W. van Wilgen, M. Cabeza, B. Schroder, S. B. Carvalho, A. Falcucci, L. Maiorano, T. J. Regan, C. Rondinini, L. Biotani, and H. P. Possingham. 2011. Ecological-economic optimization of biodiversity conservation under climate change. Nature Climate Change **1**:335-359.

<u>**Appendix 1. Examples of the tasks cited in the main text**</u>

Data pre-processing

Our first example illustrates how programming skills can be helpful to combine data from multiple files. A researcher has tagged pilot whales and these tags record information on animal movement (i.e., azimuth, pitch, and roll are measured) and feeding behavior (e.g., determined based on the clicks made by the animal to echolocate its prey) in separate files. The researcher wants to determine if, and how, the animal changes its movement pattern when it is close to its prey. To do this, we need to combine both of these datasets to compare body movement when the animal is "buzzing" (i.e., as the animal rapidly approaches a prey, the echolocation clicks occur at higher and higher frequencies, sounding like a buzz to human ears) versus, say, body movement 10 seconds before and after that. Because the times at which the different datasets were collected differ (i.e., angle measurements are taken every 0.8 seconds while the feeding behavior data only includes when the "buzzing" started and ended), it is not straightforward to merge them without creating a customized code. Our results after combining these datasets indicate that the animal tends to move considerably more when close to its prey (Fig. 1).

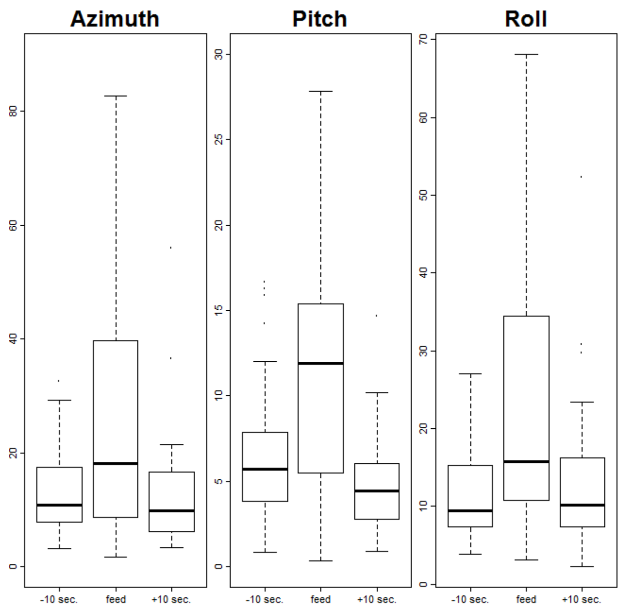


Fig. 1. Pilot whales move more while feeding. Animal movement 10 seconds prior (-10 sec.), during (feed), and 10 seconds after (+10 sec.) feeding is displayed. We defined movement as $\min(360 - |x_{t+1} - x_t|, |x_{t+1} - x_t|)$, where x_t is the angle measurement at time t, this way avoiding problems with circular statistics.

```
Pseudo-code

    1)  Determine each buzz duration
    2)  For each buzz record, summarize the angle measurements:
        A - taken during the buzz event
        B – taken 10 seconds before the buzz event
        C – taken 10 seconds after the buzz event
    3)  Store these angle measurement summaries
```

The next example uses long-term tree monitoring data from a tropical forest (Clark and Clark 2006) to illustrate how programming skills can be useful to identify potentially problematic observations. One way to identify problematic diameter measurements is to flag trees that are either unrealistically small or large. However, a subtler problem refers to diameter measurements that are unusual because they imply a diameter growth rate that is substantially different from the past or future growth rates of the same tree. We start by sub-setting the trees that have never changed the height of diameter measurement of the species *Simarouba amara*, yielding ~2,800 diameter measurements from 336 trees measured multiple times. Suppose we want to examine in greater detail trees that have diameter increments deemed unusual, here arbitrarily defined to be greater than 25 mm yr^{-1} or smaller than -5 mm yr^{-1}. The pseudo-code would be:

```
Pseudo-code

    1)  Subset the trees that have never changed the height of diameter measurement
    2)  For year i to year i+1:
        A - Calculate diameter increment for all trees
        B - Flag trees that have diameter increment greater than the 25 mm $yr^{-1}$ or smaller than -5 mm
        $yr^{-1}$
    3)  Subset the trees that were flagged at least once for a more detailed inspection
```

The plot of the history of diameter increments for all the trees deemed to have unusual diameter increments (Panel B in Fig. 2) shows that some trees were incorrectly flagged (e.g., trees 2 and 5 seem to be consistently growing less and less) while other trees have suspicious patterns (e.g., trees 3 and 4 show two diameter increments that stand out from the rest).

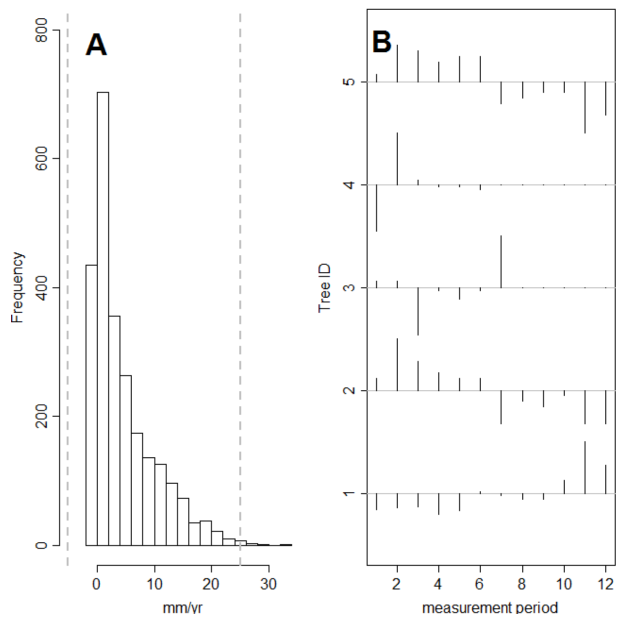


Fig. 2. Identification of trees with odd patterns in diameter growth. Panel A depicts the histogram of all diameter increments, with the thresholds used to identify unusual diameter increments highlighted by the grey vertical lines. Panel B depicts the sequence of diameter increments for trees with at least one unusual diameter increment. In this panel, diameter increment for tree *i* at time *t* ($DINC_{i,t}$) was standardized as $\frac{DINC_{i,t}-\overline{DINC_i}}{\max\left(abs\left(DINC_{i,t}-\overline{DINC_i}\right)\right)} \times 0.5$, ensuring that it lied between -0.5 and 0.5.

Forward simulations

One of the oldest yet unresolved ecological questions refers to how multiple species coexist (Siepielski and McPeek 2010). To illustrate the problem, we create a very simple model of two competing species. Say our hypothetical site is occupied by N individuals. In the beginning of our simulation, half of these individuals belong to one species ($N_{t,1}$) while the other half belongs to the other ($N_{t,2}$). We assume that these species have the same mortality rate and that, once an individual dies, its patch is occupied by a new individual of species 1 with probability $\frac{N_{t,1}}{N}$ and by species 2 otherwise. Since the simulation starts with the same number of individuals for each species and these species have identical demographic rates, one could expect that coexistence is guaranteed. Simulation results using the pseudo-code below can, however, quickly debunk this naïve expectation.

```
Pseudo-code

  1) Set the initial number of individuals ($N$, $N_{1,1}$, and $N_{1,2}$) and mortality rate.
  2) For each year:
     A – determine the number of individuals that die from each species
     B – determine which species will occupy each of these vacant areas
```

While deterministic simulations would indeed indicate coexistence, our simulation results show that adding stochasticity fundamentally changes the predicted outcomes, indicating that coexistence is relatively rare (Panel B in Fig. 3). These simulations allow students to quickly assess the importance of several model parameters to ensure coexistence. For example, students can evaluate how simulation outcomes change as the overall number of individuals changes (which illustrates that extinction happens faster as habitat size decreases). These simulations also challenge students to think of which types of processes could be in place to increase the probability of coexistence (e.g., rare species advantage, where mortality rate decreases as the density of conspecific individuals decreases; or spatial aggregation of species, where patch occupancy probability depends on the species of neighboring individuals).

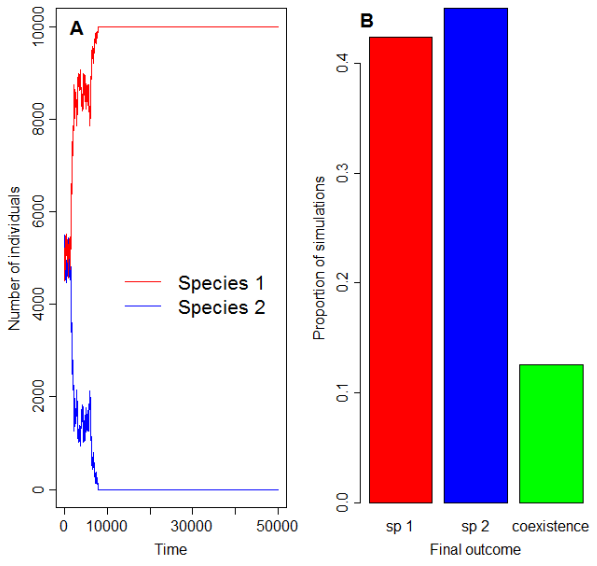


Fig. 3. Coexistence is rare even if species have the same demographic rates and initial population size. Panel A shows the result of one simulation. Panel B displays the summary results from 500 simulations, revealing the proportion of simulations in which species one dominated and species two went extinct (red), species two dominated and species one went extinct (blue), or none went extinct (green). Our simulations started with N=10,000, mortality rate of 0.2, and simulation length of 50,000 years.

Optimization

A conservation organization is interested in increasing the population of a particular endangered species to 100 individuals 20 years from now. Because resources are scarce, these resources will be considered wasted if the management action fails but also if this action results in a much larger overall population than originally targeted. Therefore, a researcher might ask by how much should one (or more) demographic parameter be increased to achieve the desired population size in year 20. This question can be recast into an optimization problem, where we want to find the parameter value that minimizes the distance between our target and the simulated population size. Here we assume that this is a stage-structured population, currently comprised of 11 pups, 9 juveniles, and 30 adults, and that the transition matrix A was estimated to be:

$$A = \begin{bmatrix} 0 & 0 & F \\ G_p & S_j & 0 \\ 0 & G_j & S_a \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0.530 \\ 0.856 & 0.2 & 0 \\ 0 & 0.389 & 0.645 \end{bmatrix}$$

where G_p, S_j, G_j, S_a, and F are the probabilities of pups becoming juveniles, juveniles remaining juveniles, juveniles becoming adults, adults remaining adults, and the per capita fecundity rate, respectively. Parameter values were taken from Wielgus et al. (2008), corresponding to the sea lion population at the Granito island. We choose to optimize just the fecundity parameter. We use the following pseudo-code:

```
Pseudo-code

    1) Forward simulation part:
Create a function to predict future population size, based on the set of demographic parameters
provided by the user. This function will:
        A - Create the transition matrix A using the provided demographic parameters.
        B - Set the vector with the initial number of individuals at each stage to $N = \begin{bmatrix} 11 \\ 9 \\ 30 \end{bmatrix}$
        C - For year 1 to year 20, update N by calculating $N = AN$
        D - Sum the number of individuals over all stages and output results to user

    2) Optimization part:
Use an optimizer to determine the fecundity rate that would make the projected population size in year
20 as close as possible to 100
```

Our results indicate that the fecundity rate would have to increase from 0.53 to 0.96 to ensure that this population will have 100 individuals 20 years from now.

Statistics

We illustrate computer intensive methods in statistics by analyzing spatial patterns of tropical trees. These spatial patterns can provide indirect information on dispersal and mortality of these tree species, yielding insights regarding the mechanisms that shape community structure and that allow so many species to coexist. Several metrics have been used to determine if these trees are more clumped or dispersed than expected (Condit et al. 2000, Seidler and Plotkin 2006, Terborgh et al. 2008), partly because this analysis does not fit nicely into standard statistical tests with their corresponding normal assumptions. Here we develop yet another metric to determine how clumped each species is. Our approach is to calculate the mean nearest conspecific neighbor distance (NCND) for the saplings of each species. Then, we estimate the p-value of this statistic by generating the NCND distribution under the null hypothesis that these saplings are randomly distributed, akin to a randomization test. This can be done using the following pseudo-code:

```
Pseudo-code

    1)  For each species $i$, calculate:
        A – the number of saplings from that species and the mean nearest conspecific neighbor distance. Denote these $N_i$ and $NCND_i^{obs}$, respectively.
        B – randomly distribute $N_i$ saplings across the plot and calculate $NCND_i^{sim}$.
        C – Repeat B 1000 times and estimate the p-value as $p\left(NCND_i^{sim} < NCND_i^{obs}\right) \cong \frac{\sum I\left(NCND_i^{sim} < NCND_i^{obs}\right)}{1000}$
```

Using data from four plots located in the Peruvian Amazon, two faunally intact and two hunted, our results suggest that saplings from most tree species tend to have a clumped spatial distribution (Panel A in Fig. 4). However, when we disaggregate by the faunal status of each plot, our results indicate that there tends to be more species classified as dispersed or very dispersed in faunally intact sites, whereas hunted sites tend to have more species classified as clustered (Panel B in Fig. 4). These results might be attributed to the presence of large bodied seed dispersers in these faunally intact sites.

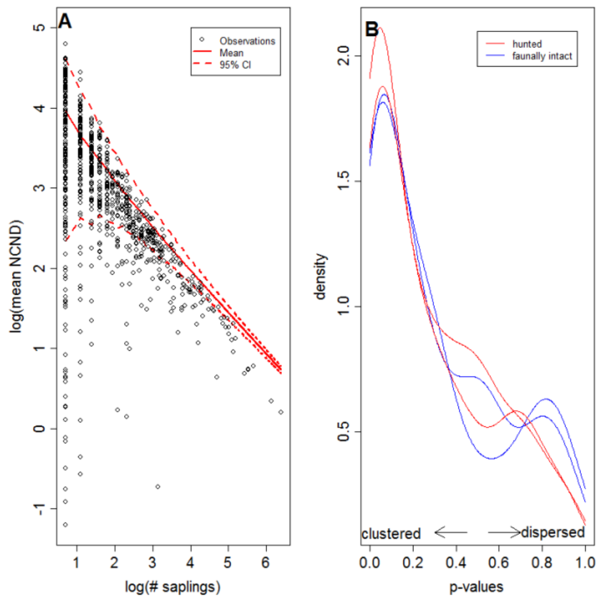


Fig. 4. Tree species are spatially clustered but hunted sites have more clustering than faunally intact sites. Panel A shows the relationship between mean nearest conspecific neighbor distance (NCND) and the number of saplings within the plot on a log-scale. Red lines (solid and dashed lines are mean and 95% confidence intervals, respectively) indicate the expected pattern if saplings were distributed at random, superimposed on the data (each point represents a species within a particular site). Panel B shows a kernel density estimate of the p-value (estimated from randomization tests) distribution for two hunted (red) and two faunally intact (blue) plots.

We note that the implementation of this randomization test illustrates the meaning of p-values in a more transparent and intuitive manner than using a GUI driven interface from a statistical package, enabling a more profound comprehension of statistics by the student.

<u>References for Appendix 1</u>

Allison, D. B. 2009. The antidote to bias in research. Science **326**:522.

Anderson, M., A. McKenzie, B. Wellman, M. Brown, and S. Vrbsky. 2011. Affecting attitudes in first-year computer science using syntax-free robotics programming. ACM Inroads **2**:51-57.

Borer, E. T., E. W. Seabloom, M. B. Jones, and M. Schildhauer. 2009. Some simple guidelines for effective data management. Bulletin of the Ecological Society of America **90**:205-214.

Cassey, P. and T. M. Blackburn. 2006. Reproducibility and repeatability in ecology. Bioscience **56**:958-959.

Clark, D. B. and D. A. Clark. 2006. Tree growth, mortality, physical condition, and microsite in an old-growth lowland tropical rain forest. Ecology **87**:2132.

Condit, R., P. S. Ashton, P. Baker, S. Bunyavejchewin, S. Gunatilleke, N. Gunatilleke, S. P. Hubbell, R. B. Foster, A. Itoh, J. V. Lafrankie, H. S. Lee, E. Losos, N. Manokaran, R. Sukumar, and T. Yamakura. 2000. Spatial patterns in the distribution of tropical tree species. Science **288**:1414-1418.

CSTA Standards Task Force. 2011. CSTA K-12 Computer Science Standards. Computer Science Teachers Association (CSTA).

Diaconis, P. and B. Efron. 1983. Computer-Intensive Methods in Statistics. Scientific American:116-130.

Ellison, A. M. 2010. Repeatability and transparency in ecological research. Ecology **91**:2536-2539.

Ellison, A. M. and B. Dennis. 2010. Paths to statistical fluency for ecologists. Frontier in Ecology and Environment **8**:362-370.

Ellison, A. M., L. J. Osterweil, L. Clarke, J. L. Hadley, A. Wise, E. Boose, D. R. Foster, A. Hanson, D. Jensen, P. Kuzeja, E. Riseman, and H. Schultz. 2006. Analytic webs support the synthesis of ecological data sets. Ecology **87**:1345-1358.

Hasni, T. F. and F. Lodhi. 2011. Teaching problem solving effectively. ACM Inroads **2**:58-62.

Jones, M. B., M. Schildhauer, O. J. Reichman, and S. Bowers. 2006. The new bioinformatics: integrating ecological data from the gene to the biosphere. Annual Review of Ecology, Evolution, and Systematics **37**:519-544.

Marks, C. O. and M. J. Lechowicz. 2006. Alternative designs and the evolution of functional diversity. The American Naturalist **167**:55-66.

Michener, W. K., J. W. Brunt, J. J. Helly, T. B. Kirchner, and S. G. Stafford. 1997. Nongeospatial metadata for the ecological sciences. Ecological Applications **7**:330-342.

Michener, W. K. and M. B. Jones. 2012. Ecoinformatics: supporting ecology as a data-intensive science. Trends in Ecology and Evolution **27**:85-93.

Nature. 2009. Data's shameful neglect. Nature **461**:145-145.

Peck, S. L. 2004. Simulation as experiment: a philosophycal reassessment for biological modeling. Trends in Ecology and Evolution **19**:530-534.

Peng, R. D., F. Dominici, and S. L. Zeger. 2006. Reproducible epidemiologic research. American Journal of Epidemiology **163**.

Reichman, O. J., M. B. Jones, and M. Schildhauer. 2011. Challenges and opportunities of open data in ecology. Science **331**:703-705.

Seidler, T. G. and J. B. Plotkin. 2006. Seed dispersal and spatial pattern in tropical trees. PLOS Biology **4**.

Siepielski, A. M. and M. A. McPeek. 2010. On the evidence for species coexistence: a critique of the coexistence program. Ecology **91**:3153-3164.

Terborgh, J., G. Nunez-Iturri, N. C. A. Pitman, F. H. C. Valverde, P. Alvarez, V. Swamy, E. G. Pringle, and C. E. T. Paine. 2008. Tree recruitment in an empty forest. Ecology **89**:1757-1768.

Vision, T. J. 2010. Open data and the social contract of scientific publishing. Bioscience **60**:330-330.

Whitlock, M. C. 2011. Data archiving in ecology and evolution: best practices. Trends in Ecology and Evolution **26**:61-65.

Whitlock, M. C., M. A. McPeek, M. D. Rausher, L. Rieseberg, and A. J. Moore. 2010. Data archiving. The American Naturalist **175**:145-146.

Wielgus, J., M. Gonzalez-Suarez, D. Aurioles-Gamboa, and L. R. Gerber. 2008. A noninvasive demographic assessment of sea lions based on stage-specific abundances. Ecological Applications **18**:1287-1296.

Wintle, B. A., S. A. Bekessy, D. A. Keith, B. W. van Wilgen, M. Cabeza, B. Schroder, S. B. Carvalho, A. Falcucci, L. Maiorano, T. J. Regan, C. Rondinini, L. Biotani, and H. P. Possingham. 2011. Ecological-economic optimization of biodiversity conservation under climate change. Nature Climate Change **1**:335-359.

Appendix 2. Description of the surveys

We performed two surveys, examining 1) course offerings from environmental science programs in the United States, and 2) trends in post-doctoral advertisements that mention computer programming.

We studied the 20 top environmental science programs in the United States, as ranked by the US News and World Report (2011, http://www.usnews.com/education/worlds-best-universities-rankings/best-universities-environmental-sciences). We searched the most recent course catalogs and bulletins of these programs for course descriptions that mentioned 'programming' or 'computation' and excluded courses that listed those as prerequisite skills, limiting our results to courses that teach computer programming to environmental scientists. This survey could potentially miss occasional course offerings, but demonstrates programmatic investment in computer programming education.

We also surveyed trends in post-doctoral advertisements posted to the ECOLOG email list between January 1, 2000 and March 20, 2012 (http://listserv.umd.edu/archives/ecolog-l.html). We identified emails that contained "postdoc" or "post-doc" or "post doc" in the subject and then subsetted the ones that contained "programming" or "computation" or "matlab" or "C++" or "Python" or "Visual Basic" in the message text. We removed duplicates and replies (i.e., containing "Re: " in the subject), resulting in 2606 total post-doc advertisements. Then, we summarized the results by year for analysis.

Appendix 3. Course structure

As mentioned in the main text, we target a wide range of data management and environmental science problems that are familiar to most of our students and that can be tackled with simple algorithms. As a result, we avoid simply listing the commands and explaining what they do. Rather, we focus on how to accomplish different real-world tasks. Our course uses R mainly because it is free; as a result, students do not need to worry about paying exorbitant license fees if they do not stay in academia. Furthermore, R is a highly versatile tool, with extensive documentation available online, and supported by a large community of users.

We divide our course into three main sections of approximately the same length. The first section covers exploratory data analysis / data visualization, reproducible research and effective data management, introducing the basic commands and syntax along the way. The second section introduces loops as the key to harness the power of computer. Finally, the final section brings all these concepts together by applying them to several real-world applications.

Data forms the basis of our course. Thus, our first section starts by describing how to import and export data and how to view data. We then lay-out the bread-and-butter tools for exploratory data analysis. We show how 2-D graphics (e.g., scatter plots, box-plots, histograms, bar-plots, pie-charts) can be easily created and used to identify outliers or extreme values that are used as missing data code. We also introduce Boolean logic to allow students to subset and query data in multiple ways. We emphasize the importance of consistency checks. For instance, one may receive data on the different land cover areas within a buffer of radius *r* around multiple cities. In this case, one can easily sum the area of each land use type to see if this sum equals the expected buffer size πr^2. Similarly, one can check for trees that have shrunk or people that are yet to be born. Fancier data visualizations are introduced at the end of the first section, including locally weighted polynomial regression (LOWESS), smoothed histograms, conditional plots, the usage of color and symbols to add a third dimension in 2-D plots (e.g., heat map, colored scatter-plots or colored box-plots), and finally 3-D graphs (e.g., surface plots and 3-D scatter plots).

Throughout this first session, we reiterate the advantages of using scripted analysis programs regarding data management and reproducible science. For instance, by keeping only the initial raw and the final edited dataset, one avoids the proliferation of spreadsheets that occurs even for well organized researchers. Furthermore, all judgment calls regarding deleted or imputed data points, as well as plain mistakes, are automatically recorded within the script.

The second session focuses predominantly on loops. In our experience, recursive structures, such as loops, are the major stumbling block for students, particularly for those with no or little programming experience. Thus, we spend considerable time building familiarity with loops and flow commands (e.g., if conditions). We often start by writing multiple lines of code to illustrate how a particular repetitive task would be solved in the absence of loops. We then draw the student's attention to the patterns that emerge from these multiple lines of code, which form the basis for creating a succinct loop. We employ abstract examples (e.g., create a series of even numbers, Fibonacci numbers, prime numbers, calculate 10 factorial) and applied examples (e.g., importing multiple data files that have similar names - 'rain1.txt', 'rain2.txt', 'rain3.txt'; creating multiple histograms of variables in the dataset; projecting future population size) to fix ideas. There is often considerable heterogeneity among students regarding how fast they pick up these concepts, which may be fruitfully exploited by pairing individuals with different skills. At the end of this session, we show how one can create customized functions, which allows for a more compartimentalized code and is the gateway for optimization procedures.

The third session aims at bringing all these concepts together. We target vastly different problems that nevertheless rely on the same building blocks introduced in earlier sessions, such as loops and Boolean logic. These examples involve data management, building complex figures, optimization, computer intensive statistical methods, and forward simulations (e.g., Appendix 1). During this session, we emphasize that these tasks are not mutually exclusive. For instance, an optimization problem may require forward simulations (e.g., optimal timber harvesting) while a computer intensive statistical methods may rely on optimization (e.g., bootstrap). Our goal here is to show how the standard toolbox of environmental scientists can be dramatically expanded by creatively combining these different tools to solve real-world problems.

Overall, our teaching strategy is to stimulate students to create algorithms that work, rather than focusing on improving algorithm efficiency or elegant code. For example, we focus predominantly on loops, even if the problem can be easily vectorized. Our students learn programming skills by practicing, mainly through weekly programming tasks. While other quantitative courses typically provide code that need to be slightly tweaked to answer the assignments, we prefer not to provide this baseline code. In fact, we believe that long lasting programming skills can only be acquired if one learns how to piece together the different commands that have been taught and how to troubleshoot their algorithm. In these weekly tasks, we try to show how the concepts in class can be used in several different contexts, as well as combined with each other. For example, while in class we worked on optimization as a tool for fitting a nonlinear curve to data, the weekly homework could include an example of optimizing returns on timber yield, which combines a forward simulator and an optimizer to determine the optimal harvest rate. We then provide our own solution code for these weekly tasks. By doing so, we hope to illustrate how different approaches exist to solve the same problem, and to foster good coding practices (e.g., using indentation and comments, avoiding hardwired numbers, creating compartmentalized code, etc.).

Throughout the course, we also avoid explaining each command in detail, rather exhorting students to seek information on the web. We find this to be important to ensure that, by the end of the course, the students are fully independent of the instructor, being capable of searching for information and understanding how a command not covered in class works.