Beginning at the Beginning: Teaching **Novice** Physicists Data Science **Programming**

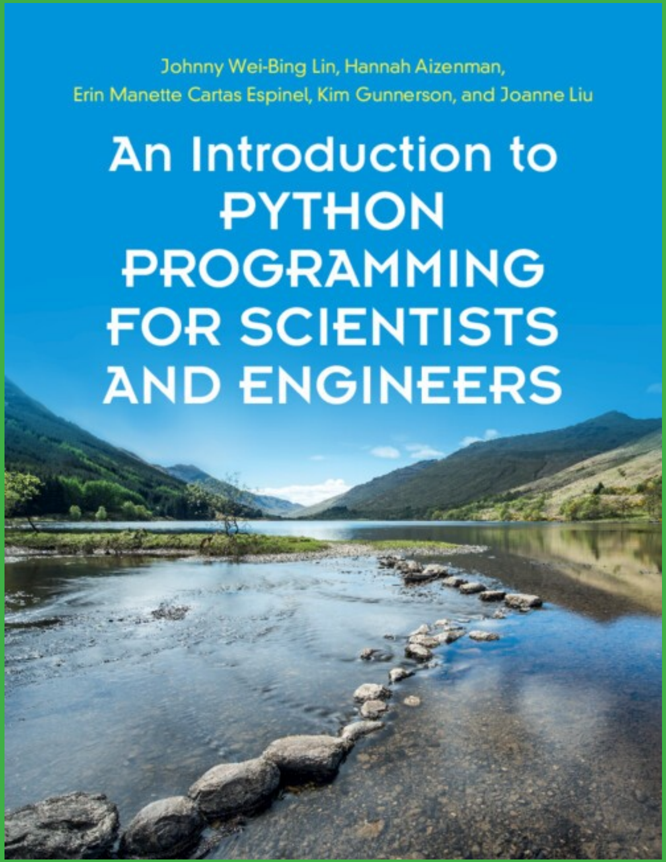
Johnny Wei-Bing Lin

Apologies to Lewis Carroll. Photo: Pexels.com, Pixabay.

Photo: Pexels.com, Pixabay.

Preview

- **Context** for my talk.
- **Challenges** of teaching novice scientists how to program.
- **General advice** when teaching beginning programming.
- **Specific advice** for teaching physicists and data scientists.



Endorsed by
Kari Jordan
(The Carpentries)

Book: `tinyurl.com/pyscibook`
Rules: `tinyurl.com/roaopk`
Email: `johnny@johnny-lin.com`

@Lin_etal_IPySEs

Features

- Science workflow-driven, not syntax-driven.
- Examples and practice problems are from the sciences.
- **Very gentle pacing,** with many practice problems.
- Additional discipline-specific Jupyter notebooks of practice and homework problems in biology, chemistry, and physics.
- **Teaches novices programming, not data science tools** per se.

Book: tinyurl.com/pyscibook
Rules: tinyurl.com/roaopk
Email: johnny@johnny-lin.com

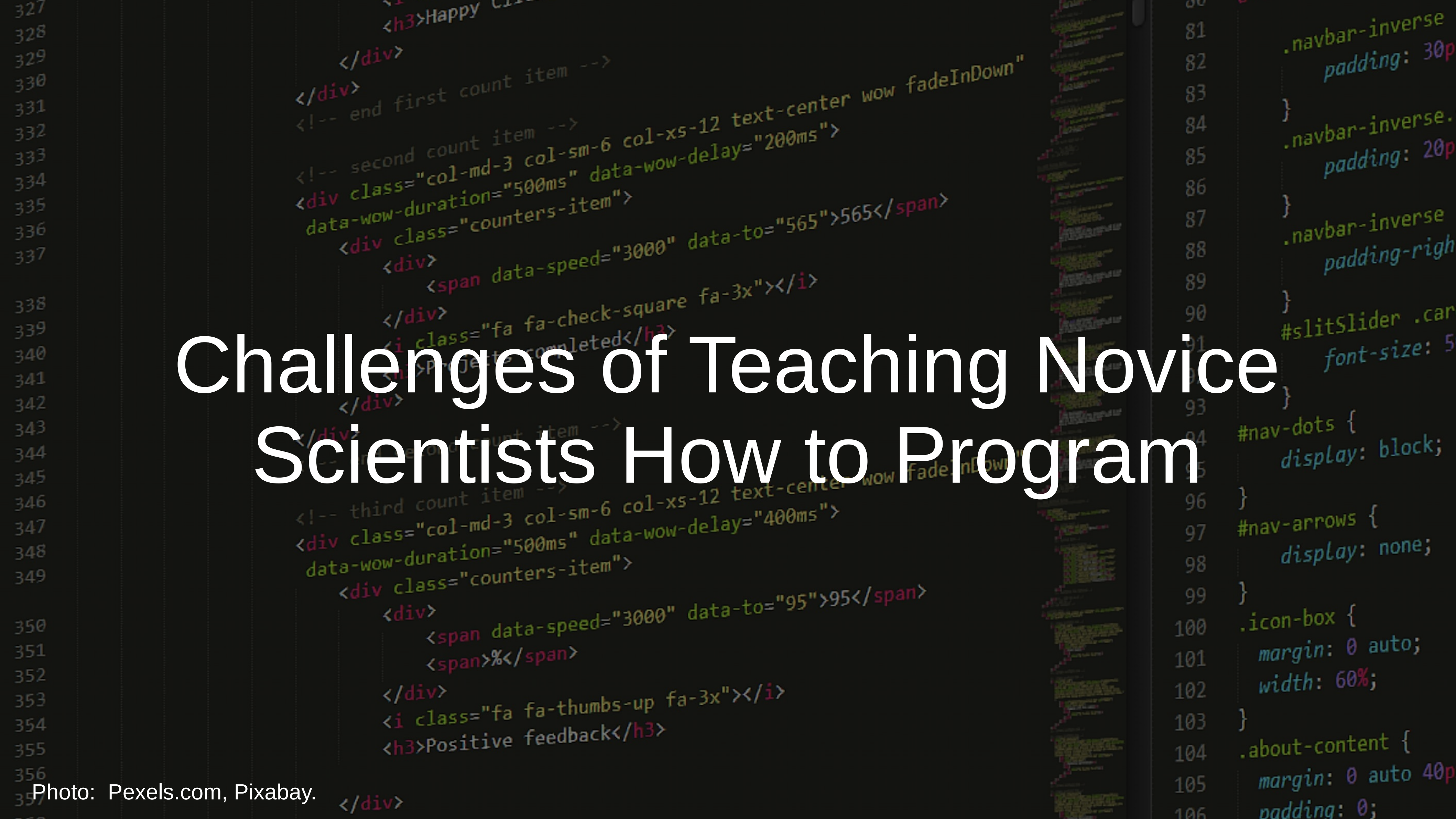
@Lin_etal_IPySEs

Contents

- Pt. 1: Getting Basic Tasks Done (e.g., Python as a calculator, basic plots, text files).
- Pt. 2: Doing More Complex Tasks (e.g., n-D data analysis, missing data).
- Pt. 3: Advanced Programming Concepts (e.g., inheritance, searching and sorting, other file formats, recursion).
- Pt. 4: Going From a Program Working to Working Well (e.g., documentation, profiling, unit testing).

Book: `tinyurl.com/pyscibook`
Rules: `tinyurl.com/roaopk`
Email: `johnny@johnny-lin.com`

@Lin_etal_IPySEs



Challenges of Teaching Novice Scientists How to Program

Photo: Pexels.com, Pixabay.

Programming Is Hard, Especially When You First Learn It

The “Rainfall Problem”: “Write a program that repeatedly reads in positive integers, until it reads the integer 99999. After seeing 99999, it should print out the average.”

Soloway et al. (1983), described in Guzdial (2010)

Student Group	% Correct Using Raw Pascal
CS 1	14%
CS 2	36%
Systems Course (Juniors and Seniors)	69%

Photo: Pexels.com, Pixabay.

Substantial Numbers of Students Do Not Pass CS 1

- Meta-analysis of studies of CS 1 courses from 1979-2013 (Watson & Li 2014).
  - **Overall global passing rate: 67%.**
  - No statistically significant variation over time.
  - No statistically significant variation amongst the 4 countries making up 80% of the sample.
- Bi-modal distribution commonly observed in classes (Guzdial 2010).

Photo: Pexels.com, Pixabay.

It's Tough to be a Novice Physicist Programmer

- Most scientific computing resources assume you already can program.
- Introductory CS resources often focus on CS puzzles.
- **We do not know of one simple, effective, scalable process to learn programming.**

Photo: Pexels.com, Pixabay.

General Advice When Teaching Beginning Programming

Photo: Pexels.com, Pixabay.

Tip #1: Teach How to Break Apart a Task

- **Do not start with code.**
- Outline the steps of the problem in normal English.
- Write the outline by hand.
- Make the outline have sub-levels.
- Avoid assuming there's a single function to solve the problem.

Photo: Pexels.com, Pixabay.

Tip #2: Don't Assume Programming Makes Sense

- Variables behave differently than in math.
- Defining and calling a function are different things.
- Students don't get what a function return value is.
- **Use real-world analogies to explain code concepts.**


Photo: Pexels.com, Pixabay.


Photo: Pexels.com, Pixabay.

Tip #3: Teach Line-By-Line Code Reading

- Ask what **each expression** in a line of code returns.
- For **each line** of code, ask:
  - What are the pre-states.
  - What are the post-states.
  - What did the line change.
- Teach how to make a **handwalk table** of variables.

```
1  def add_up(in_list):
2      if len(in_list) == 1:
3          return in_list[0]
4      else:
5          return in_list[0] + \
6                 add_up(in_list[1:])
7
8  print(add_up([2, 4, 6]))
```

Call Level	Line #	`in_list`	Returns
0	1	`[2, 4, 6]`	N/A
	4	Select `else` option	
	5-6	`[2, 4, 6]`	`2 + add_up([4, 6])`
1	1	`[4, 6]`	N/A
	4	Select `else` option	
	5-6	`[4, 6]`	`4 + add_up([6])`
... continued ...			

Specific Advice for Teaching Physicists and Data Scientists

Photo: Pexels.com, SpaceX.

Photo: Pexels.com, SpaceX.

Tip #4: Use Cool Examples Judiciously

- Cool science examples make programming relevant!
- Cool examples can be difficult for **novice programmers** to understand.
- Seeing a pattern does not, by itself, teach you how to program.
- **Scaffold the example into small steps.**
- **Use repetition** to ensure understanding. It's okay to be basic.

Fitting and predicting: estimator basics

`Scikit-learn` provides dozens of built-in machine learning algorithms and models, called estimators. Each estimator can be fitted to some data using its fit method.

Here is a simple example where we fit a `RandomForestClassifier` to some very basic data:

```
>>> from sklearn.ensemble import RandomForestClassifier
>>> clf = RandomForestClassifier(random_state=0)
>>> X = [[ 1,  2,  3],  # 2 samples, 3 features
...      [11, 12, 13]]
>>> y = [0, 1]  # classes of each sample
>>> clf.fit(X, y)
RandomForestClassifier(random_state=0)
```

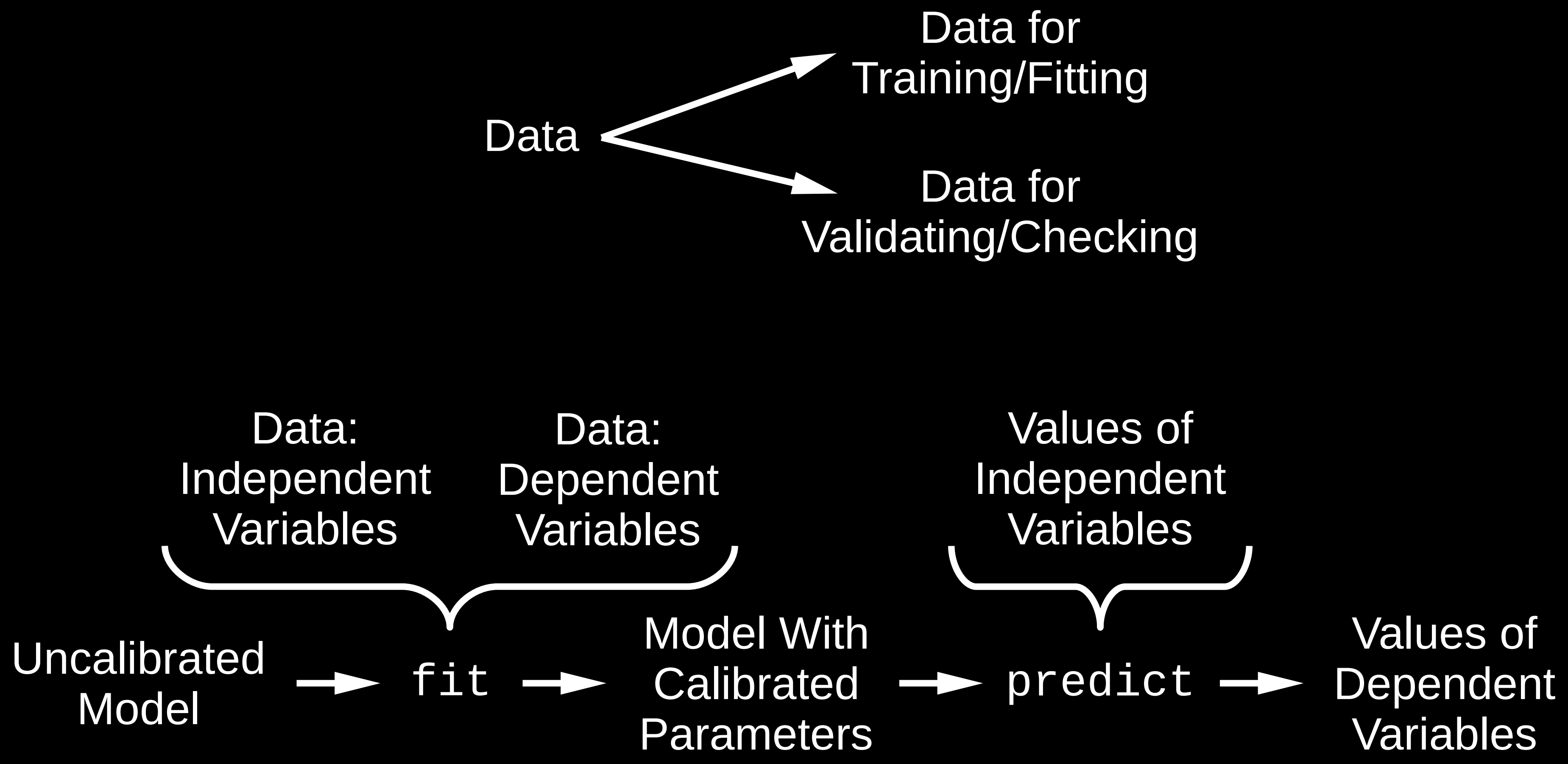
The fit method generally accepts 2 inputs:

- The samples matrix (or design matrix) X. The size of `X` is typically `(n_samples, n_features)`, which means that samples are represented as rows and features are represented as columns.
- The target values y which are real numbers for regression tasks, or integers for classification (or any other discrete set of values). For unsupervized learning tasks, `y` does not need to be specified. `y` is usually 1d array where the `i` th entry corresponds to the target of the `i` th sample (row) of `X`.

https://scikit-learn.org/stable/getting_started.html

Some Novice Questions That Are Unanswered

- What is a sample?
- What is a feature?
- What is a class of a sample? Is this OOP?
- What does the estimator estimate?
- What does it mean to "fit" an estimator?
- What is a "target value?" How does that relate to a "classes of each sample?"
- Why do I want a fitted estimator?


Data
Data for Training/Fitting
Data for Validating/Checking
Data: Independent Variables
Data: Dependent Variables
Values of Independent Variables
Uncalibrated Model
fit
Model With Calibrated Parameters
predict
Values of Dependent Variables

Building Novice Understanding Through Repetition

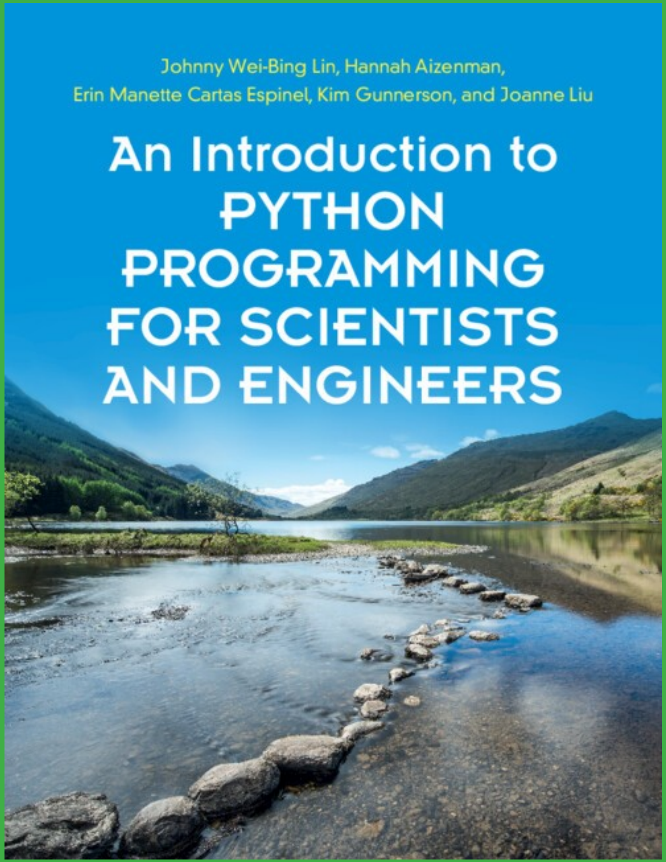
- We want to do a regression of the Object A Position vs. Time:
  - What is a sample?
  - What is a feature?
  - What is a class of a sample?
- How can we create a fitted estimator using this data? How can we check how well the fitted estimator is doing?
- Can we use `predict` with non-integral values of time? Why or why not?
- The scikit-learn manual says that the feature and class have to be 2-D arrays. But these are 1-D arrays. What do we do?

Time [s]	Obj. A Pos. [m]	Obj. B Pos. [m]	Obj. C Pos. [m]
0	0	1.3	4.3
1	1	1.3	6.7
2	2	1.3	10.1
3	3	1.3	22.3
... continued ...			


Photo: Pexels.com, Pixabay.

Conclusions

- Learning how to program is hard.
- Teach how to break down a solution into tasks a computer can do.
- Use examples students can connect with.
- Take small steps in your scaffold.



Endorsed by
Kari Jordan
(The Carpentries)

Book: `tinyurl.com/pyscibook`
Rules: `tinyurl.com/roaopk`
Email: `johnny@johnny-lin.com`

@Lin_etal_IPySEs

References

- Guzdial, M. (2010): Ch. 7 of Oram, A. and G. Wilson, *Making Software: What Really Works, and Why We Believe It,* O'Reilly Media.
- Soloway, E., J. Bonar, et al., 1983. Cognitive strategies and looping constructs: An empirical study. *Communications of the ACM* **26,** 11, 853-860.
- Watson, C. and F. W.B. Li, Failure rates in introductory programming revisited, *ItICSE'14,* June 21-25, 2014, 39-44.