Python for Statisticians

(permute—Permutation tests and confidence sets for Python)

K. Jarrod Millman
Division of Biostatistics
University of California, Berkeley

SciPy India 2015
IIT Bombay

http://www.jarrodmillman.com/talks/scipyindia2015/python_for_statisticians.pdf

GUEST EDITORS' INTRODUCTION

Python for Scientists and Engineers

During the past decade, Python (an interpreted, high-level programming language) has arguably become the de facto standard for exploratory, interactive, and computation-driven scientific research. This issue discusses Python's advantages for scientific research and presents several of the core Python libraries and tools used in this domain. Although the issue's articles are self-contained, they nicely complement those in *CiSE*'s May/June 2007 special issue, "Python: Batteries Included."[1]

In addition to the technical advantages described in this issue, one of Python's most compelling assets as a platform for scientific computing is the SciPy community. The SciPy community is a well-established and growing group of scientists, engineers, and researchers using, extending, and promoting Python's use for scientific research.

1521-9615/11/$26.00 © 2011 IEEE
Copublished by the IEEE CS and the AIP

K. Jarrod Millman
University of California, Berkeley
Michael Aivazis
California Institute of Technology

Python for Statisticians

- Statistical computing
- Permutation testing

Statistical computing landscape

History of statistical computing (at Berkeley)

- Census data
- Bombing research (WWII)
- DEC PDP-11/45 (1974)

Credit: `en.wikipedia.org/wiki/Marchant_calculator`

History of statistical programming

Once upon a time, statistical programming involved calling Fortran subroutines directly.

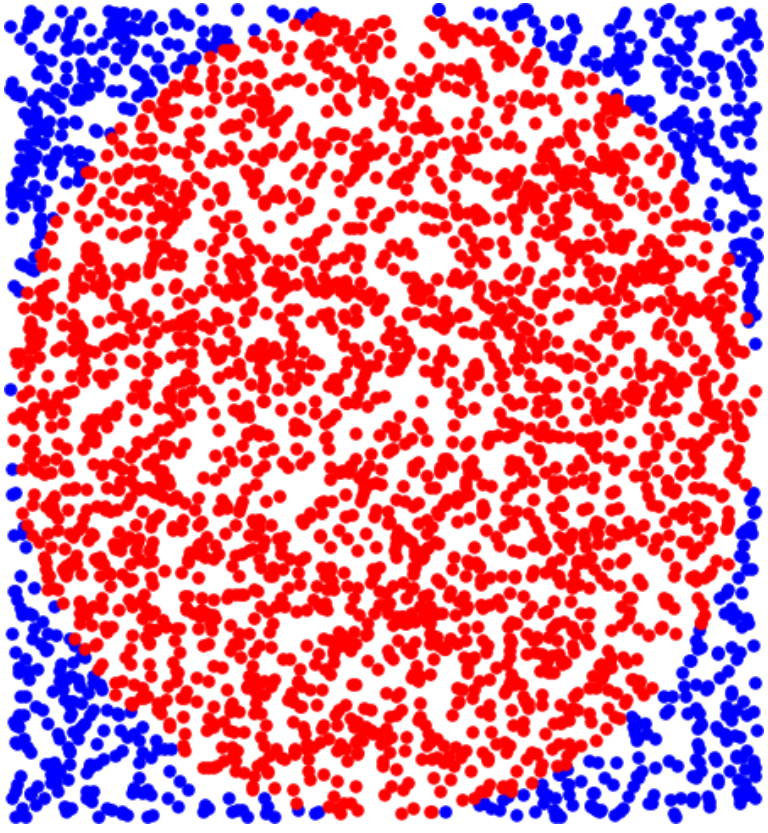
S provided a common environment to interactively explore data.

- Fortran (1950s)
- APL (1960s)
- S (1970s)
- R (1990s)
- Python (1990s)

Credit: en.wikipedia.org/wiki/APL_(programming_language)

Monte Carlo

```
>>> from numpy import sqrt
>>> from numpy.random import random
```



```
>>> x = 2*random(10**8) - 1
>>> y = 2*random(10**8) - 1
>>> length = sqrt(x**2 + y**2)
>>> in_circle = length <= 1
>>> 4 * in_circle.mean()
3.14152224
```

Resampling

- Bootstrap
- Permutation tests

Deep learning

Blocks and Fuel: Frameworks for deep learning

Bart van Merriënboer BART.VAN.MERRIENBOER@UMONTREAL.CA
Montreal Institute for Learning Algorithms, University of Montreal, Montreal, Canada

Dzmitry Bahdanau D.BAHDANAU@JACOBS-UNIVERSITY.DE
Jacobs University, Bremen, Germany

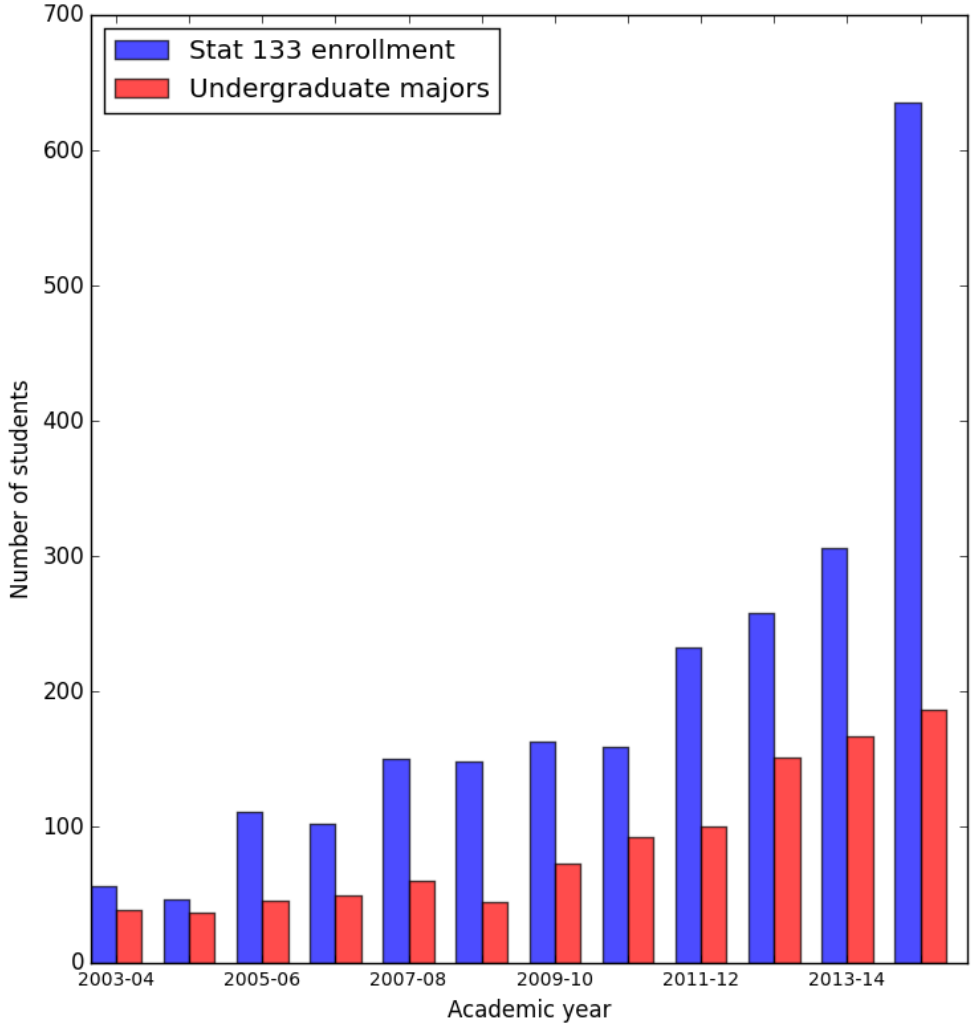
Vincent Dumoulin DUMOULIV@IRO.UMONTREAL.CA
Dmitriy Serdyuk SERDYUK@IRO.UMONTREAL.CA
David Warde-Farley WARDEFAR@IRO.UMONTREAL.CA
Montreal Institute for Learning Algorithms, University of Montreal, Montreal, Canada

Jan Chorowski JAN.CHOROWSKI@II.UNI.WROC.PL
University of Wrocław, Wrocław, Poland

Yoshua Bengio YOSHUA.BENGIO@UMONTREAL.CA
Montreal Institute for Learning Algorithms, University of Montreal, Montreal, Canada
CIFAR Senior Fellow

arxiv.org/abs/1506.00619

Stat 133: Concepts in Computing with Data



Why Python?

- General purpose language with batteries included
- Popular for wide-range of scientific applications
- Growing number of libraries statistical applications
  - `pandas`, `scikit-learn`, `statsmodels`

Stat 94: Foundations of Data Science

Credit: `www.dailycal.org/2015/09/02/uc-berkeley-piloting-new-data-science-class-fall`

data8.org

STAT 94

Foundations of Data Science
Fall 2015

Principal Instructor:
Ani Adhikari

Co-Instructors:
John DeNero
Michael I. Jordan
Tapan Parikh
David Wagner

Calendar

Resources

Course Info

Staff

`datascience` Reference

Calendar

Week	Date	Lecture	Lab/Assignment
1	Wed 8/26	Why Data Science? (Video, Slides)	Lab 1: Python and Jupyter
	Fri 8/28	Causality and Experiments (Video, Slides)	Homework 1 (due Wed 9/2) (Solutions)
2	Mon 8/31	Expressions (Video, Slides)	
	Wed 9/2	Sequences (Video, Slides)	Lab 2: Sequences
	Fri 9/4	Tables (Video, Slides)	Homework 2 (due Wed 9/9) (Solutions)
3	Mon 9/7	No Lecture: Labor Day	
	Wed 9/9	Visualizations (Video, Slides)	Lab 3: Tables
	Fri 9/11	Bar Charts and Histograms (Video, Slides)	Homework 3 (due Wed 9/16) (Solutions)
4	Mon 9/14	Functions (Video, Slides)	
	Wed 9/16	Sampling (Video, Slides)	Lab 4: Histograms

More Python in the statistics curriculum

- Stat 159/259: Reproducible and Collaborative Statistical Data Science
- Stat 222: Masters of Statistics Capstone Project
- Stat 243: Introduction to Statistical Computing

Permutation testing

Permutation tests (sometimes referred to as randomization, re-randomization, or exact tests) are a nonparametric approach to statistical significance testing.

- Permutation tests were developed to test hypotheses for which relabeling the observed data was justified by *exchangeability* of the observed random variables.
- In these situations, the conditional distribution of the test statistic under the null hypothesis is completely determined by the fact that all relabelings of the data are equally likely.

Exchangeability

A sequence $X_1, X_2, X_3, \ldots, X_n$ of random variables is exchangeable if their joint distribution is invariant to permutations of the indices; that is, for all permutations π of $1, 2, \ldots, n$

$$p(x_1, \ldots, x_n) = p(x_{\pi(1)}, \ldots, x_{\pi(n)})$$

Exchangeability II

Exchangeability is closely related to the notion of *independent and identically-distributed* (iid) random variables.

- iid random variables are exchangeable.
- But, simple random sampling without replacement produces an exchangeable, but **not** independent, sequence of random variables.

Effect of treatment in a randomized controlled experiment

www.stat.berkeley.edu/~stark/Teach/S240/Notes/lec1.pdf

11 pairs of rats, each pair from the same litter.

Randomly—by coin tosses—put one of each pair into "enriched" environment; other sib gets "normal" environment.

After 65 days, measure cortical mass (mg).

treatment	689	656	668	660	679	663	664	647	694	633	653
control	657	623	652	654	658	646	600	640	605	635	642
difference	32	33	16	6	21	17	64	7	89	-2	11

How should we analyze the data?

Informal Hypotheses

www.stat.berkeley.edu/~stark/Teach/S240/Notes/lec1.pdf

Null hypothesis: treatment has "no effect."

Alternative hypothesis: treatment increases cortical mass.

Suggests 1-sided test for an increase.

Test contenders

www.stat.berkeley.edu/~stark/Teach/S240/Notes/lec1.pdf

- 2-sample Student t-test:

$$\frac{\text{mean(treatment) - mean(control)}}{\text{pooled estimate of SD of difference of means}}$$

- 1-sample Student t-test on the differences:

$$\frac{\text{mean(differences)}}{\text{SD(differences)}/\sqrt{11}}$$

  Better, since littermates are presumably more homogeneous.
- Permutation test using t-statistic of differences: same statistic, different way to calculate P-value. Even better?

Strong null hypothesis

www.stat.berkeley.edu/~stark/Teach/S240/Notes/lec1.pdf

Treatment has no effect whatsoever—as if cortical mass were assigned to each rat before the randomization.

Then equally likely that the rat with the heavier cortex will be assigned to treatment or to control, independently across littermate pairs.

Gives $2^{11} = 2,048$ equally likely possibilities:

difference	± 32	± 33	± 16	± 6	± 21	± 17	± 64	± 7	± 89	± 2

For example, just as likely to observe original differences as

difference	-32	-33	-16	-6	-21	-17	-64	-7	-89	-2	-11

Weak null hypothesis

www.stat.berkeley.edu/~stark/Teach/S240/Notes/lec1.pdf

On average across pairs, treatment makes no difference.

Alternatives

www.stat.berkeley.edu/~stark/Teach/S240/Notes/lec1.pdf

Individual's response depends only on that individual's assignment

Special cases: shift, scale, etc.

Interactions/Interference: my response could depend on whether you are assigned to treatment or control.

Assumptions of the tests

www.stat.berkeley.edu/~stark/Teach/S240/Notes/lec1.pdf

- 2-sample t-test: masses are iid sample from normal distribution, same unknown variance, same unknown mean. Tests weak null hypothesis (plus normality, independence, non-interference, etc.).
- 1-sample t-test on the differences: mass differences are iid sample from normal distribution, unknown variance, zero mean. Tests weak null hypothesis (plus normality, independence, non-interference, etc.)
- Permutation test: Randomization fair, independent across pairs. Tests strong null hypothesis.

Assumptions of the permutation test are true by design: That's how treatment was assigned.

Student t-test calculations

www.stat.berkeley.edu/~stark/Teach/S240/Notes/lec1.pdf

Mean of differences: 26.73mg
Sample SD of differences: 27.33mg
t-statistic: $26.73/(27.33/\sqrt{11}) = 3.244$
P-value for 1-sided t-test: 0.0044

Why do cortical weights have normal distribution?

Why is variance of the difference between treatment and control the same for different litters?

Treatment and control are *dependent* because assigning a rat to treatment excludes it from the control group, and vice versa.

Does P-value depend on assuming differences are iid sample from a normal distribution? If we reject the null, is that because there is a treatment effect, or because the other assumptions are wrong?

Permutation t-test calculations

www.stat.berkeley.edu/~stark/Teach/S240/Notes/lec1.pdf

Could enumerate all $2^{11} = 2,048$ equally likely possibilities.
Calculate t-statistic for each.
P-value is

$$P = \frac{\text{number of possibilities with } t \geq 3.244}{2{,}048}$$

(For mean instead of t, would be $2/2,048 = 0.00098$.)

For more pairs, impractical to enumerate, but can simulate:

Assign a random sign to each difference. Compute t-statistic
Repeat 100,000 times

$$P \approx \frac{\text{number of simulations with } t \geq 3.244}{100{,}000}$$

Compute

```
from itertools import product
from numpy import array, sqrt

t = [689, 656, 668, 660, 679, 663, 664, 647, 694, 633, 653
c = [657, 623, 652, 654, 658, 646, 600, 640, 605, 635, 642
d = array(t) - array(c)
n = len(d)

x = array(list(product([1, -1], repeat=11)))
exact = x * d
dist = exact.mean(axis=1) / (exact.std(axis=1) / sqrt(n))
```

Simulate ($n \gg 11$)

```
from numpy import array, sqrt
from numpy.random import binomial as binom

t = [689, 656, 668, 660, 679, 663, 664, 647, ...]
c = [657, 623, 652, 654, 658, 646, 600, 640, ...]
d = array(t) - array(c)
n = len(d)

reps = 100000
x = 1 - 2 * binom(1, .5, n*reps)
x.shape = (reps, n)
sim = x * d
dist = sim.mean(axis=1) / (sim.std(axis=1) / sqrt(n))
```

Compare

```
>>> from numpy import mean
>>> observed_ts = d.mean() / (d.std() / sqrt(n))
>>> mean(dist >= observed_ts)
0.0009765625
```

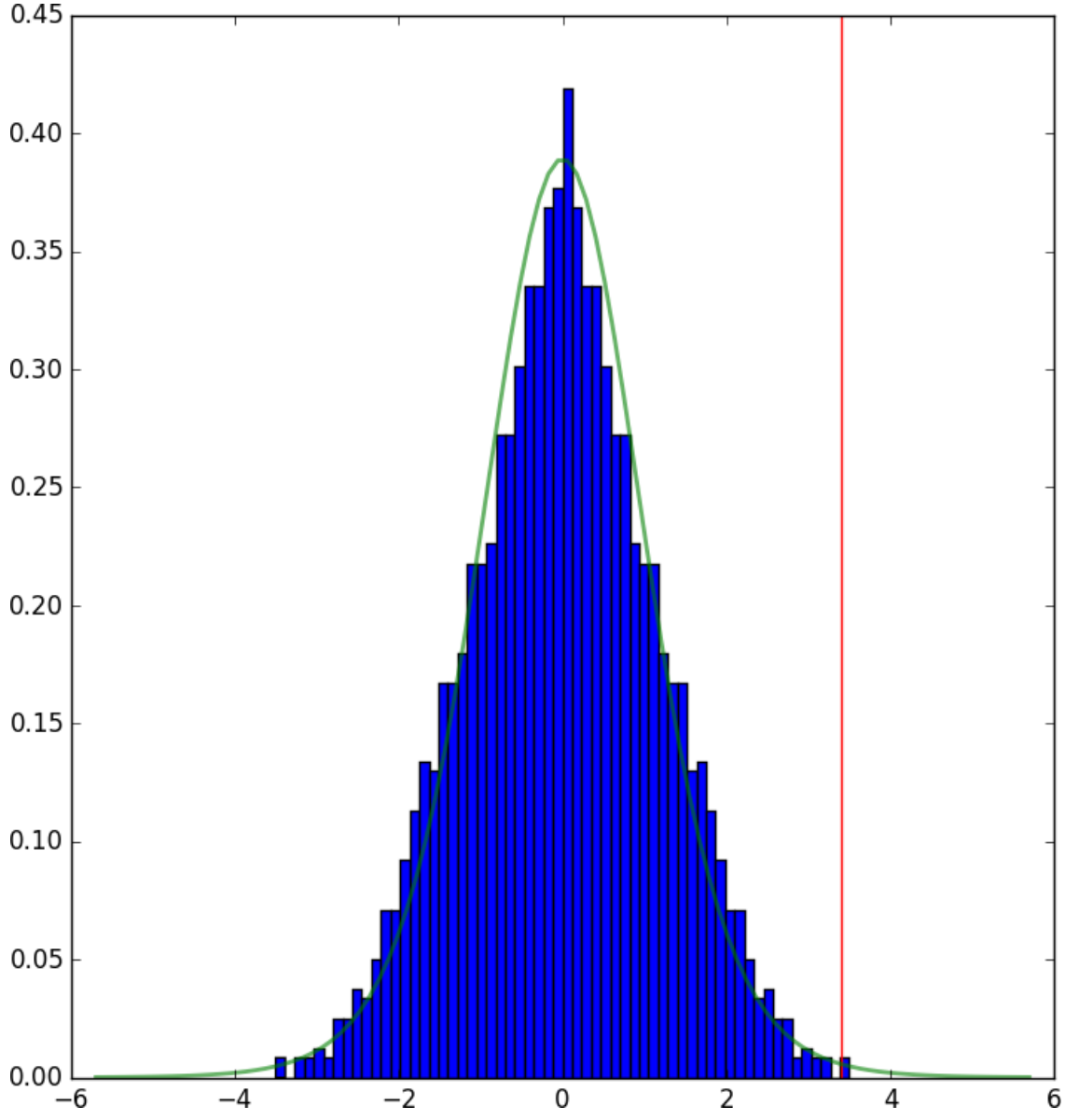
(versus 0.0044 for 1-sided t-test)

Visualize

```
import matplotlib.pyplot as plt
from numpy import linspace
from scipy.stats import t

plt.hist(dist, 100, histtype='bar', normed=True)
plt.axvline(observed_ts, color='red')
df = n - 1
x = linspace(t.ppf(0.0001, df), t.ppf(0.9999, df), 100)
plt.plot(x, t.pdf(x, df), lw=2, alpha=0.6)
plt.show()
```

Visualize



permute

Permutation tests and confidence sets

build passing coverage 99%

Permutation tests and confidence sets for a variety of nonparametric testing and estimation problems, for a variety of randomization designs.

- **Website (including documentation):** http://statlab.github.io/permute
- **Mailing list:** http://groups.google.com/group/permute
- **Source:** https://github.com/statlab/permute
- **Bug reports:** https://github.com/statlab/permute/issues

Installation from binaries

```
$ pip install permute
```

Collaborators

Philip B. Stark
pbstark

Kellie Ottoboni
kellieotto

Stefan van der Walt
stefanv