Galaxy

The challenge of reproducibility

www.galaxyproject.org

@jxtx / #usegalaxy

Biology has *rapidly* become data intensive,
and dependent on complex computational and
statistical methods

How can we ensure that these methods are
accessible to researchers?

...while also ensuring that scientific results
remain **reproducible**?

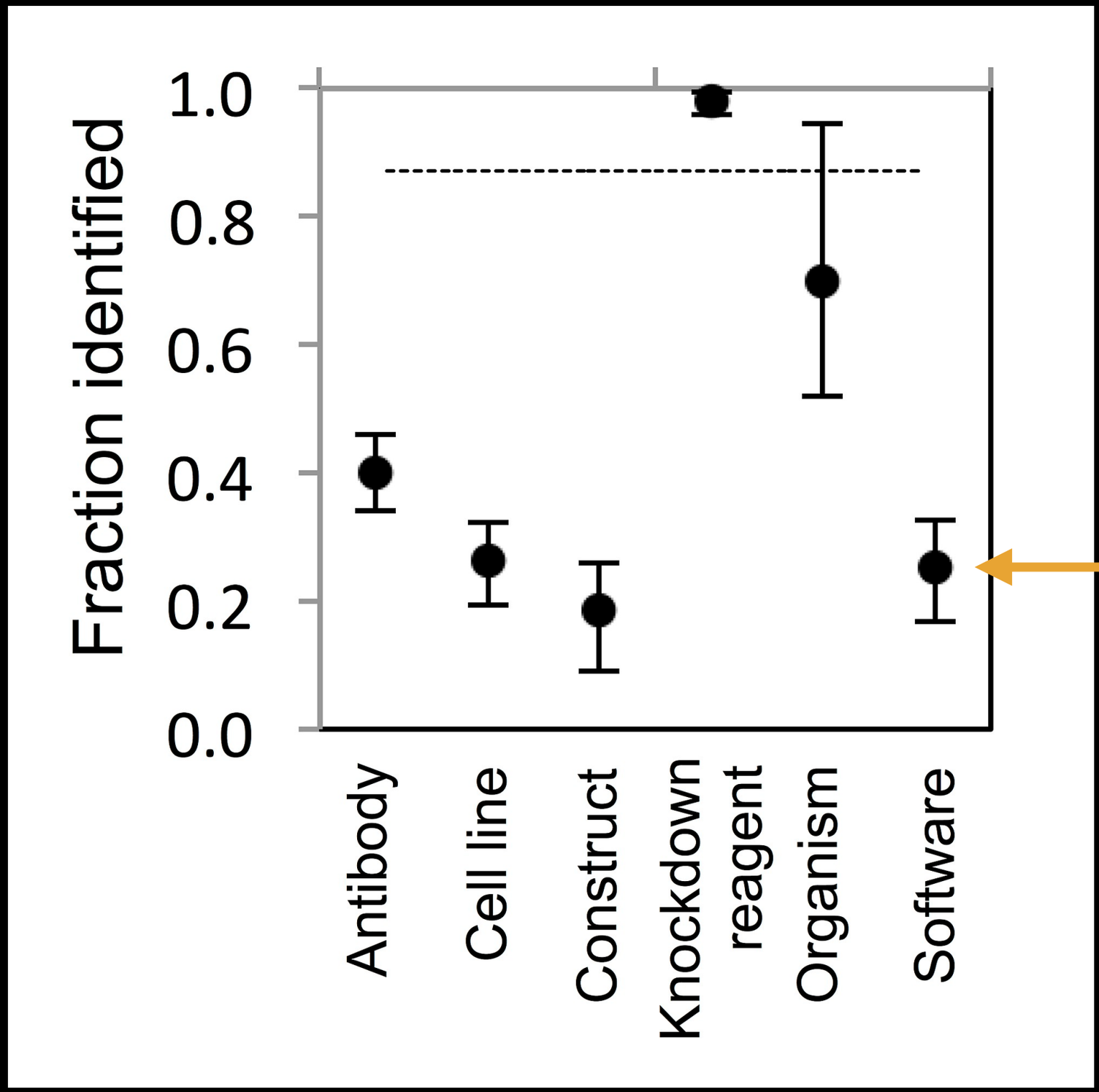
A crisis in genomics research:

reproducibility

Reproducibility Project: Cancer Biology

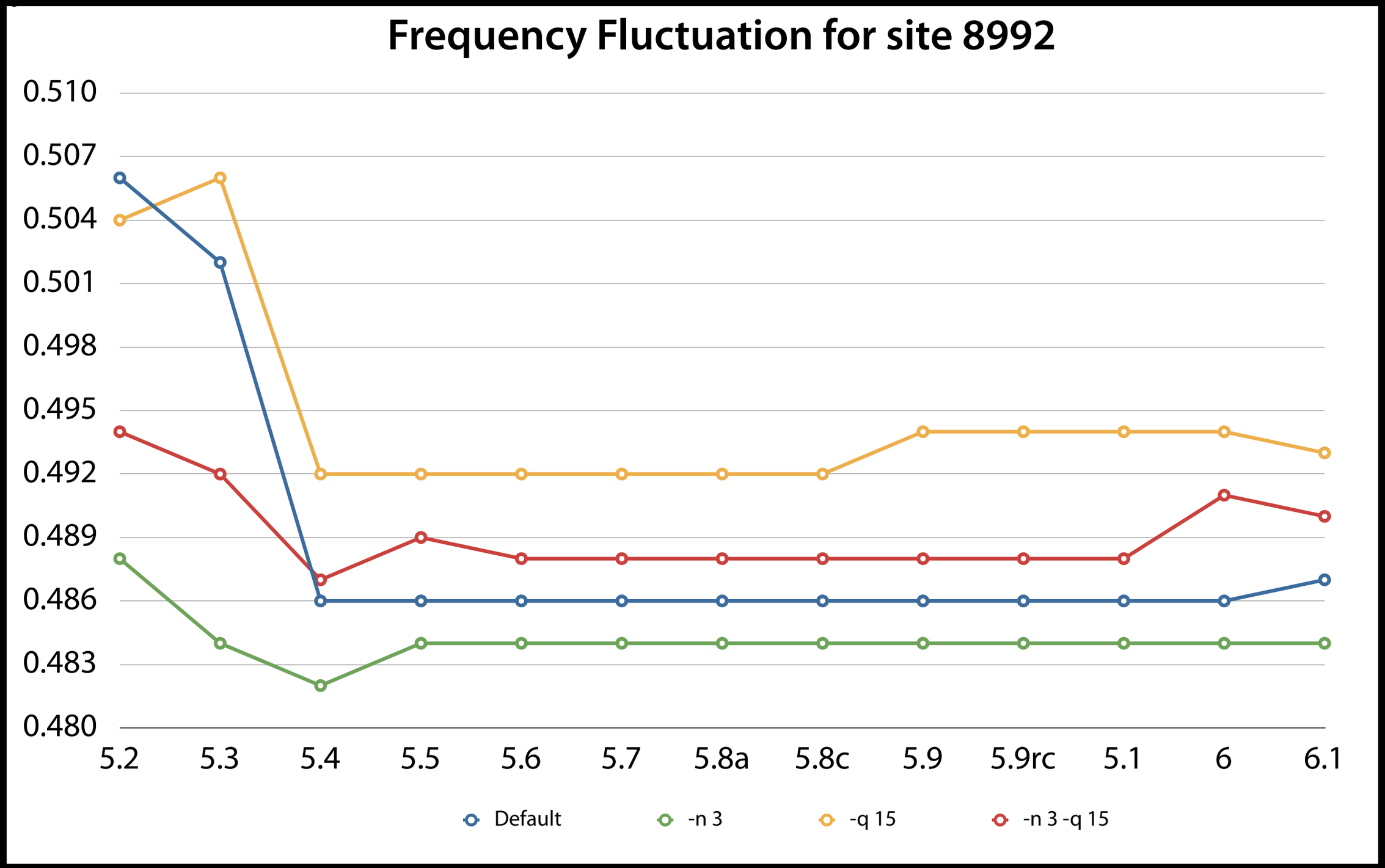
Independently replicating 50 "high-impact" cancer studies from 2010-2012

(https://osf.io/e81xl/wiki/home/)



Vasilevsky, Nicole; Kavanagh, David J; Deusen, Amy Van; Haendel, Melissa; Iorns, Elizabeth (2014): Unique Identification of research resources in studies in Reproducibility Project: Cancer Biology. figshare.
http://dx.doi.org/10.6084/m9.figshare.987130

Methods and details matter!



(Nekrutenko and Taylor, *Nature Reviews Genetics*, 2012)

What is reproducibility?

Provenance *is not* reproducibility

Reproducibility *is not* reusability

Reproducibility *is certainly not* correctness

Reproducibility means that an analysis is described in sufficient detail that it *can* be precisely reproduced

(by another person, in another environment)

Most published analysis are not reproducible.

Missing software, versions, parameters (even data)

Recommendations

1. Accept that computation is an integral component of biomedical research

2. Always provide access to raw primary data

3. Record versions of all auxiliary datasets, or archive

4. Store the exact versions of *all* software used. Ideally archive the software

5. Record *all* parameters, even if default values are used.

(Abridged from Nekrutenko and Taylor, *Nature Reviews Genetics,* 2012)

Can reproducibility be achieved?

A spectrum of solutions

Command line best practices (Version control, Makefiles, …)

Analysis environments (Galaxy, GenePattern, Mobyle, …)

Workflow systems (Taverna, Pegasus, VisTrails, …)

Notebook style (iPython notebook, …)

Literate programming style (Sweave/knitR, …)

System level provenance capture (ReproZip, …)

Complete environment capture (VMs, containers, …)

Can reproducibility be achieved?

A spectrum of solutions

Command line best practices (Version control, Makefiles, …)

Analysis environments (**Galaxy**, GenePattern, Mobyle, …)

Workflow systems (Taverna, Pegasus, VisTrails, …)

Notebook style (iPython notebook, …)

Literate programming style (Sweave/knitR, …)

System level provenance capture (ReproZip, …)

Complete environment capture (VMs, **containers**, …)