Scientific Software: Art, Engineering, or Science?

Robert Lupton, Princeton University
LSST Pipeline Scientist

2nd ASTERICS-OBELICS School
2018-06-08

La deuxième école d'Astérics et Obélics

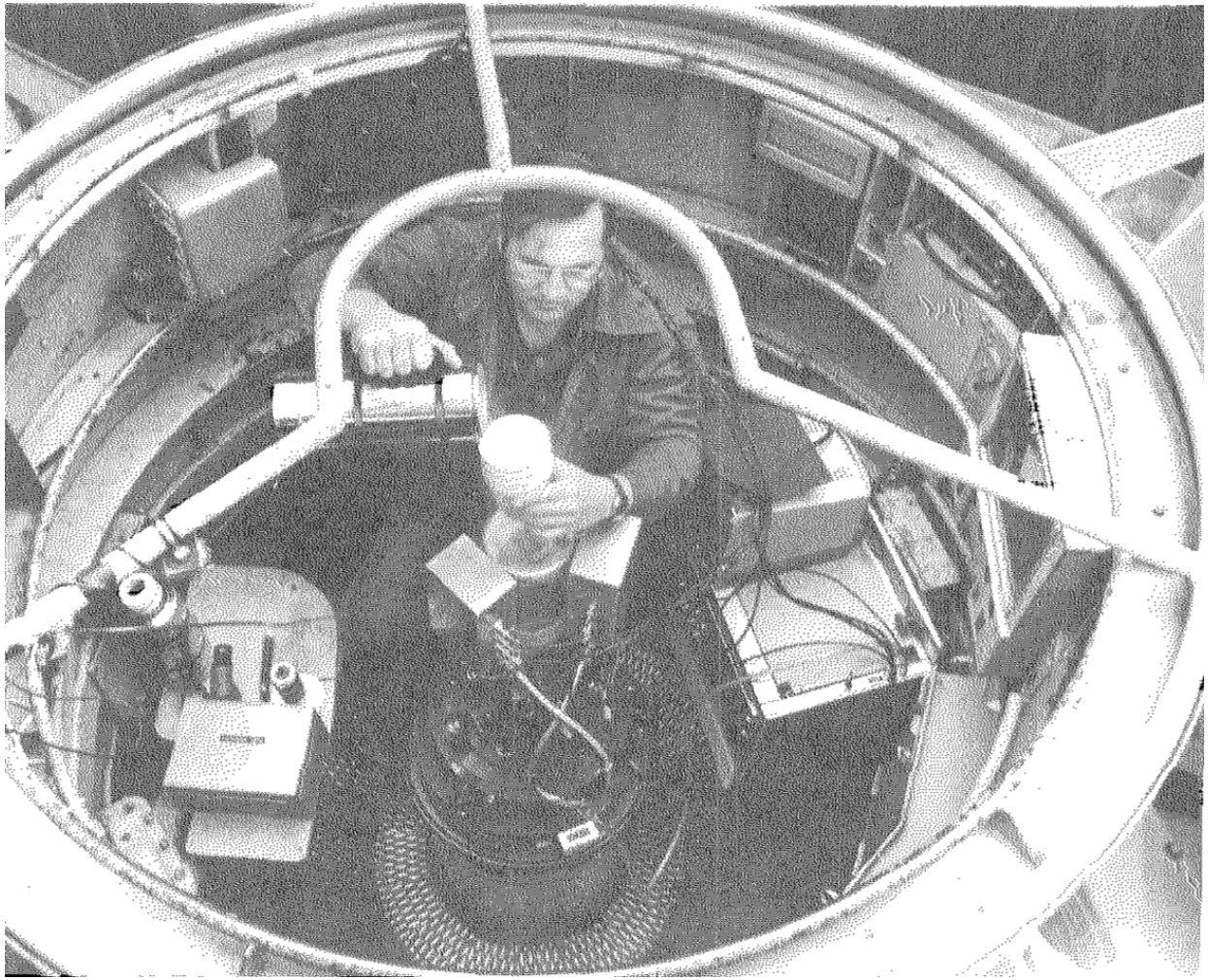
The Good Old Days

The Good Old Days

- Small data volumes

The Good Old Days

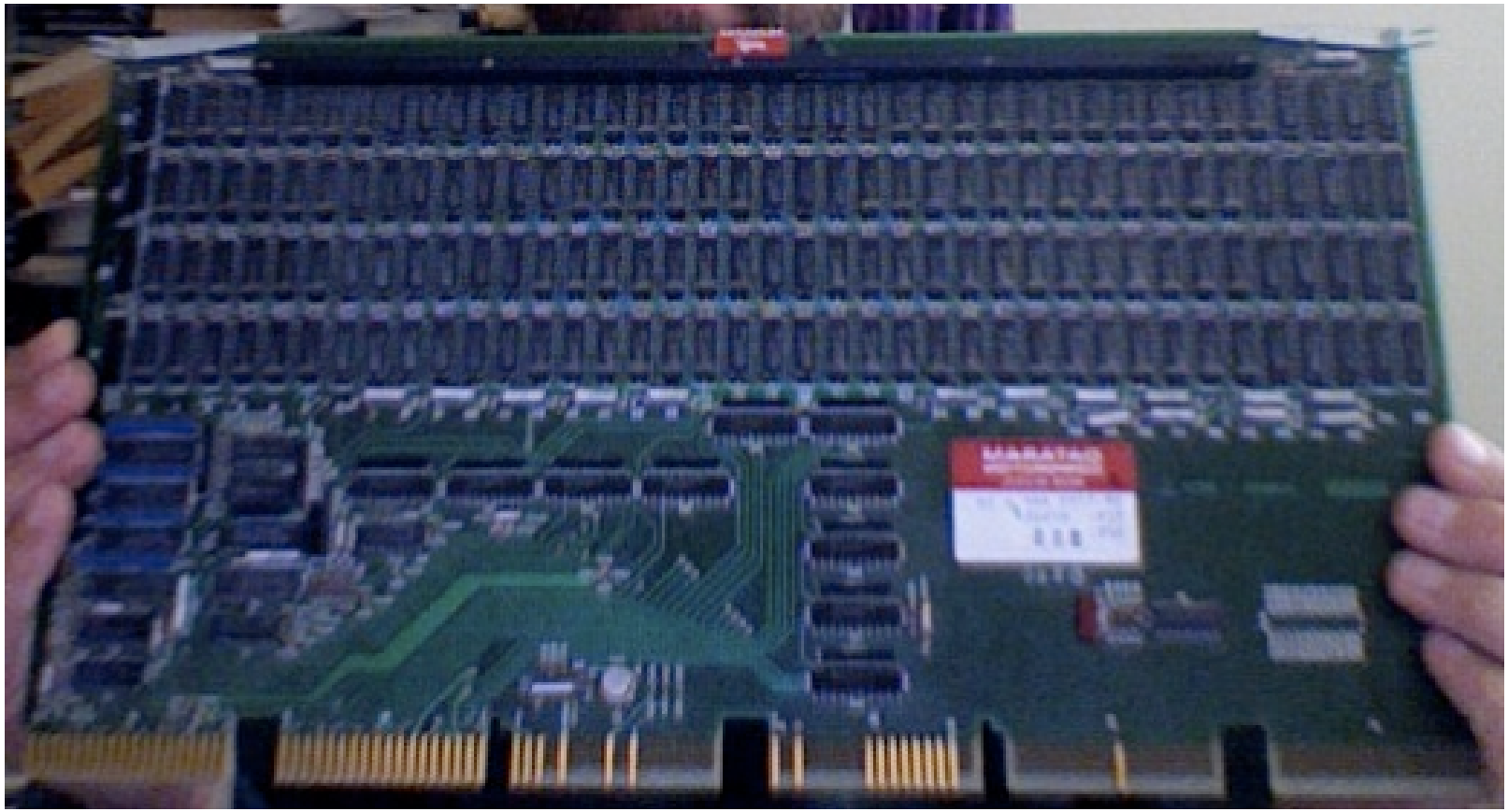
- Small data volumes
  - Small cameras and small teams



Ed Danielson with Gunn and Westphal's PFUEI (Gustafson, 1983)

The Good Old Days

- Small data volumes
  - Small cameras and small teams
  - But small computers



4*(32 + 7) 16 kbi chips; 256 kB

The Good Old Days

- Small data volumes
  - Small cameras and small teams
  - But small computers
- Simple Computers with Simple Operating Systems

The Good Old Days

- Small data volumes
  - Small cameras and small teams
  - But small computers
- Simple Computers with Simple Operating Systems … but more than one of them (JCL *v.* VMS *v.* Unix *v.* DOS *v.* …)

The Good Old Days

- Small data volumes
  - Small cameras and small teams
  - But small computers
- Simple Computers with Simple Operating Systems
  - Simple *Slow* Computers

The Good Old Days

- Small data volumes
  - Small cameras and small teams
  - But small computers
- Simple Computers with Simple Operating Systems
  - Simple *Slow* Computers . . . but getting a factor of two faster every 18 months.

The Good Old Days

- Small data volumes
  - Small cameras and small teams
  - But small computers
- Simple Computers with Simple Operating Systems
  - Simple *Slow* Computers . . . but getting a factor of two faster every 18 months.
- Simple Algorithms

The Good Old Days

- Small data volumes
  - Small cameras and small teams
  - But small computers
- Simple Computers with Simple Operating Systems
  - Simple *Slow* Computers . . . but getting a factor of two faster every 18 months.
- Simple Algorithms
  - Experts sometimes used Maximum Likelihood Estimators

The Good Old Days

- Small data volumes
  - Small cameras and small teams
  - But small computers
- Simple Computers with Simple Operating Systems
  - Simple *Slow* Computers . . . but getting a factor of two faster every 18 months.
- Simple Algorithms
  - Experts sometimes used Maximum Likelihood Estimators
- Limited Choice of Language
  - Fortran
  - C

The Good Old Days

- Small data volumes
  - Small cameras and small teams
  - But small computers
- Simple Computers with Simple Operating Systems
  - Simple *Slow* Computers . . . but getting a factor of two faster every 18 months.
- Simple Algorithms
  - Experts sometimes used Maximum Likelihood Estimators
- Limited Choice of Language
  - Fortran IV
  - C K&R

The Good Old Days

- Small data volumes
  - Small cameras and small teams
  - But small computers
- Simple Computers with Simple Operating Systems
  - Simple *Slow* Computers . . . but getting a factor of two faster every 18 months.
- Simple Algorithms
  - Experts sometimes used Maximum Likelihood Estimators
- Limited Choice of Language
  - Fortran 77
  - C 89

The Good Old Days

- Small data volumes
  - Small cameras and small teams
  - But small computers
- Simple Computers with Simple Operating Systems
  - Simple *Slow* Computers . . . but getting a factor of two faster every 18 months.
- Simple Algorithms
  - Experts sometimes used Maximum Likelihood Estimators
- Limited Choice of Language
  - Fortran 77
  - C 89

Swæ clæne [lar] wæs oðfeallenu on Angelcynne ðæt swiðe feawa wæron behionan Humbre ðe hiora ðeninga cuðen understondan on Englisc, oððe furðum an ærendgewrit of Lædene on Englisc areccean; ond ic wene ðætte noht monige begiondan Humbre næren.
Ælfred se Cyning, c. 895 CE

The Good Old Days

- Small data volumes
  - Small cameras and small teams
  - But small computers
- Simple Computers with Simple Operating Systems
  - Simple *Slow* Computers . . . but getting a factor of two faster every 18 months.
- Simple Algorithms
  - Experts sometimes used Maximum Likelihood Estimators
- Limited Choice of Language
  - Fortran 77
  - C 89

So general was [learning's] decay in England that there were very few on this side of the Humber who could understand their rituals in English, or translate a letter from Latin into English; and I believe that there were not many beyond the Humber.

Alfred the Great, c. 895 CE

The Problem

I mixed up a lot of issues on that slide!

The Problem

I mixed up a lot of issues on that slide! Astronomy is becoming Big Science

The Problem

I mixed up a lot of issues on that slide! Astronomy is becoming Big Science

- Ever-larger instruments

The Problem

I mixed up a lot of issues on that slide! Astronomy is becoming Big Science

- Ever-larger instruments
- Ever-more sophisticated algorithms

The Problem

I mixed up a lot of issues on that slide! Astronomy is becoming Big Science

- Ever-larger instruments
- Ever-more sophisticated algorithms
- Ever-larger computing facilities

The Problem

I mixed up a lot of issues on that slide! Astronomy is becoming Big Science

- Ever-larger instruments
- Ever-more sophisticated algorithms
- Ever-larger computing facilities
- Ever-more complex codes to be written, debugged, and maintained

The Problem

I mixed up a lot of issues on that slide! Astronomy is becoming Big Science

- Ever-larger instruments
- Ever-more sophisticated algorithms
- Ever-larger computing facilities
- Ever-more complex codes to be written, debugged, and maintained

All of these lead to specialisation, and Astronomy needs to adapt.

The Problem

I mixed up a lot of issues on that slide! Astronomy is becoming Big Science

- Ever-larger instruments
- Ever-more sophisticated algorithms
- Ever-larger computing facilities
- Ever-more complex codes to be written, debugged, and maintained

All of these lead to specialisation, and Astronomy needs to adapt.

Let’s take a look at some of these issues.

Instruments are becoming larger

Instruments are becoming larger

- PFUEI had one 512×512 CCD (1 amplifier)

Instruments are becoming larger

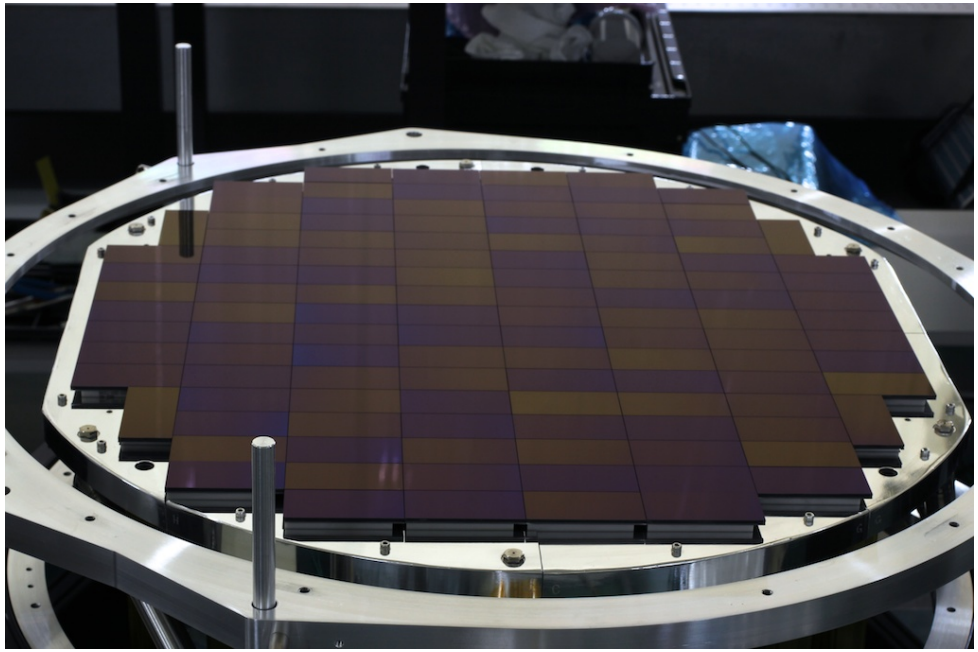
- PFUEI had one 512×512 CCD (1 amplifier)
- 4-Shooter had four 800×800 CCDS (4 amplifiers), and a fifth identical CCD for a slit spectrograph

Instruments are becoming larger

- PFUEI had one 512×512 CCD (1 amplifier)
- 4-Shooter had four 800×800 CCDS (4 amplifiers), and a fifth identical CCD for a slit spectrograph
- SDSS had 30 2k×2k CCDs (c. 45 amplifiers), 24 512×2k CCDs (c. 35 amplifiers), and two fibre spectrographs, each with a dichroic and two 2k×2k CCDs (8 amplifiers)

Instruments are becoming larger

- PFUEI had one 512×512 CCD (1 amplifier)
- 4-Shooter had four 800×800 CCDS (4 amplifiers), and a fifth identical CCD for a slit spectrograph
- SDSS had 30 2k×2k CCDs (c. 45 amplifiers), 24 512×2k CCDs (c. 35 amplifiers), and two fibre spectrographs, each with a dichroic and two 2k×2k CCDs (8 amplifiers)
- HSC has 104 2k×4k CCDs and 4 2k×4k guider/wavefront chips (432 amplifiers)



Instruments are becoming larger

- PFUEI had one 512×512 CCD (1 amplifier)
- 4-Shooter had four 800×800 CCDS (4 amplifiers), and a fifth identical CCD for a slit spectrograph
- SDSS had 30 2k×2k CCDs (c. 45 amplifiers), 24 512×2k CCDs (c. 35 amplifiers), and two fibre spectrographs, each with a dichroic and two 2k×2k CCDs (8 amplifiers)
- HSC has 104 2k×4k CCDs and 4 2k×4k guider/wavefront chips (432 amplifiers)
- LSST has 189 4k×4k CCDs and 12 4k×4k guider/wavefront chips (3216 amplifiers)

Instruments are becoming larger

- PFUEI had one 512×512 CCD (1 amplifier)
- 4-Shooter had four 800×800 CCDS (4 amplifiers), and a fifth identical CCD for a slit spectrograph
- SDSS had 30 2k×2k CCDs (c. 45 amplifiers), 24 512×2k CCDs (c. 35 amplifiers), and two fibre spectrographs, each with a dichroic and two 2k×2k CCDs (8 amplifiers)
- HSC has 104 2k×4k CCDs and 4 2k×4k guider/wavefront chips (432 amplifiers)
- LSST has 189 4k×4k CCDs and 12 4k×4k guider/wavefront chips (3216 amplifiers)
- PFS has four fibre spectrographs, each with four 2k× 4k CCDs and one 4k×4k HgCdTe detector and two dichroics (192 or *c.* 67 million amplifiers)

Instruments are becoming larger

- PFUEI had one 512×512 CCD (1 amplifier)
- 4-Shooter had four 800×800 CCDS (4 amplifiers), and a fifth identical CCD for a slit spectrograph
- SDSS had 30 2k×2k CCDs (c. 45 amplifiers), 24 512×2k CCDs (c. 35 amplifiers), and two fibre spectrographs, each with a dichroic and two 2k×2k CCDs (8 amplifiers)
- HSC has 104 2k×4k CCDs and 4 2k×4k guider/wavefront chips (432 amplifiers)
- LSST has 189 4k×4k CCDs and 12 4k×4k guider/wavefront chips (3216 amplifiers)
- PFS has four fibre spectrographs, each with four 2k× 4k CCDs and one 4k×4k HgCdTe detector and two dichroics (192 or *c.* 67 million amplifiers) and a 12k×12k CMOS metrology camera and six 2k×2k CMOS guide cameras

Instruments are becoming larger

- PFUEI had one 512×512 CCD (1 amplifier)
- 4-Shooter had four 800×800 CCDS (4 amplifiers), and a fifth identical CCD for a slit spectrograph
- SDSS had 30 2k×2k CCDs (c. 45 amplifiers), 24 512×2k CCDs (c. 35 amplifiers), and two fibre spectrographs, each with a dichroic and two 2k×2k CCDs (8 amplifiers)
- HSC has 104 2k×4k CCDs and 4 2k×4k guider/wavefront chips (432 amplifiers)
- LSST has 189 4k×4k CCDs and 12 4k×4k guider/wavefront chips (3216 amplifiers)
- PFS has four fibre spectrographs, each with four 2k× 4k CCDs and one 4k×4k HgCdTe detector and two dichroics (192 or *c.* 67 million amplifiers) and a 12k×12k CMOS metrology camera and six 2k×2k CMOS guide cameras and 2×2396 micromotors and 125 km of optical fibre

Instruments

Why do you care that instruments are becoming larger?

Instruments

Why do you care that instruments are becoming larger?

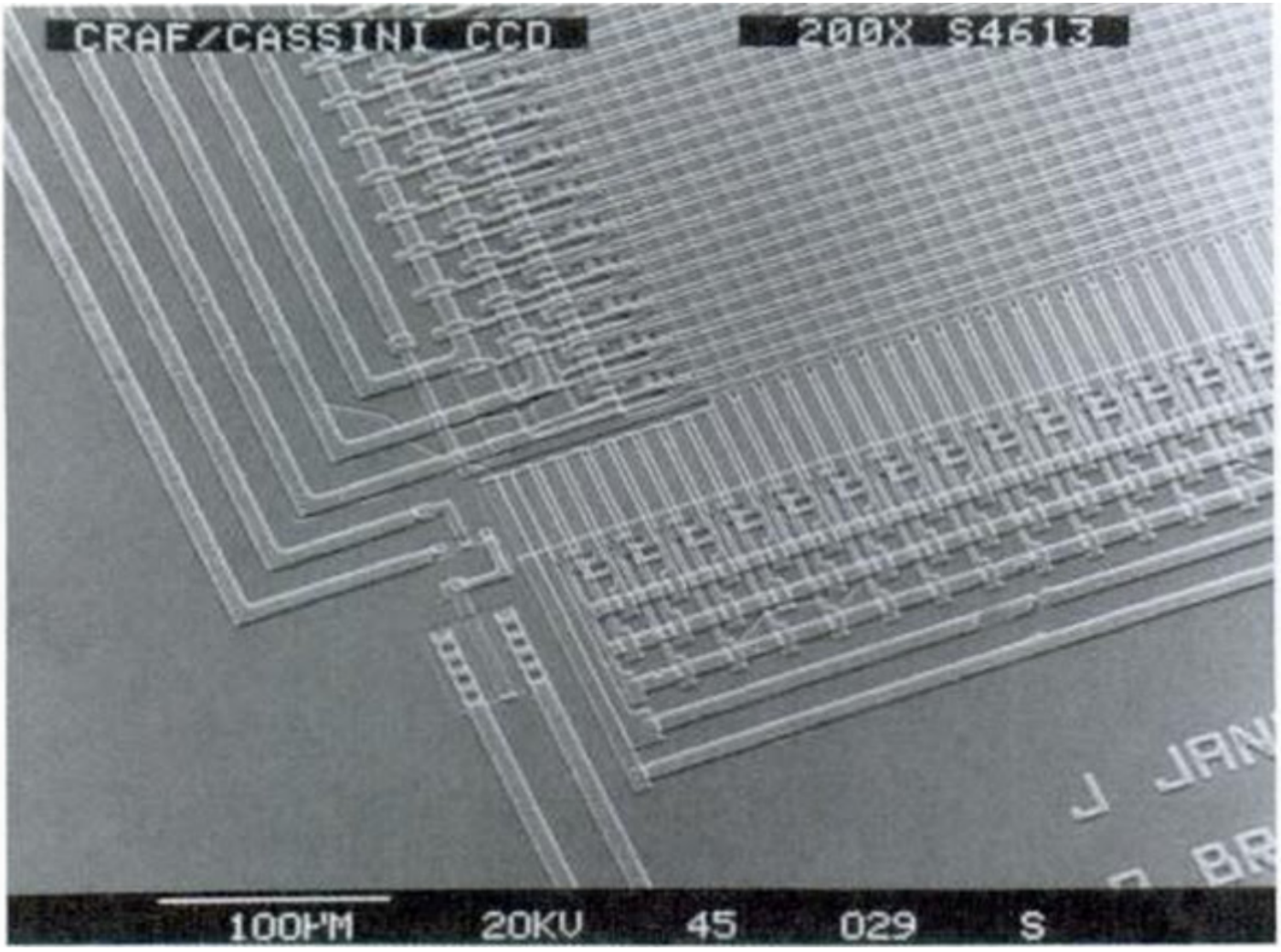
Because someone has to understand them in detail

Instruments

Why do you care that instruments are becoming larger?

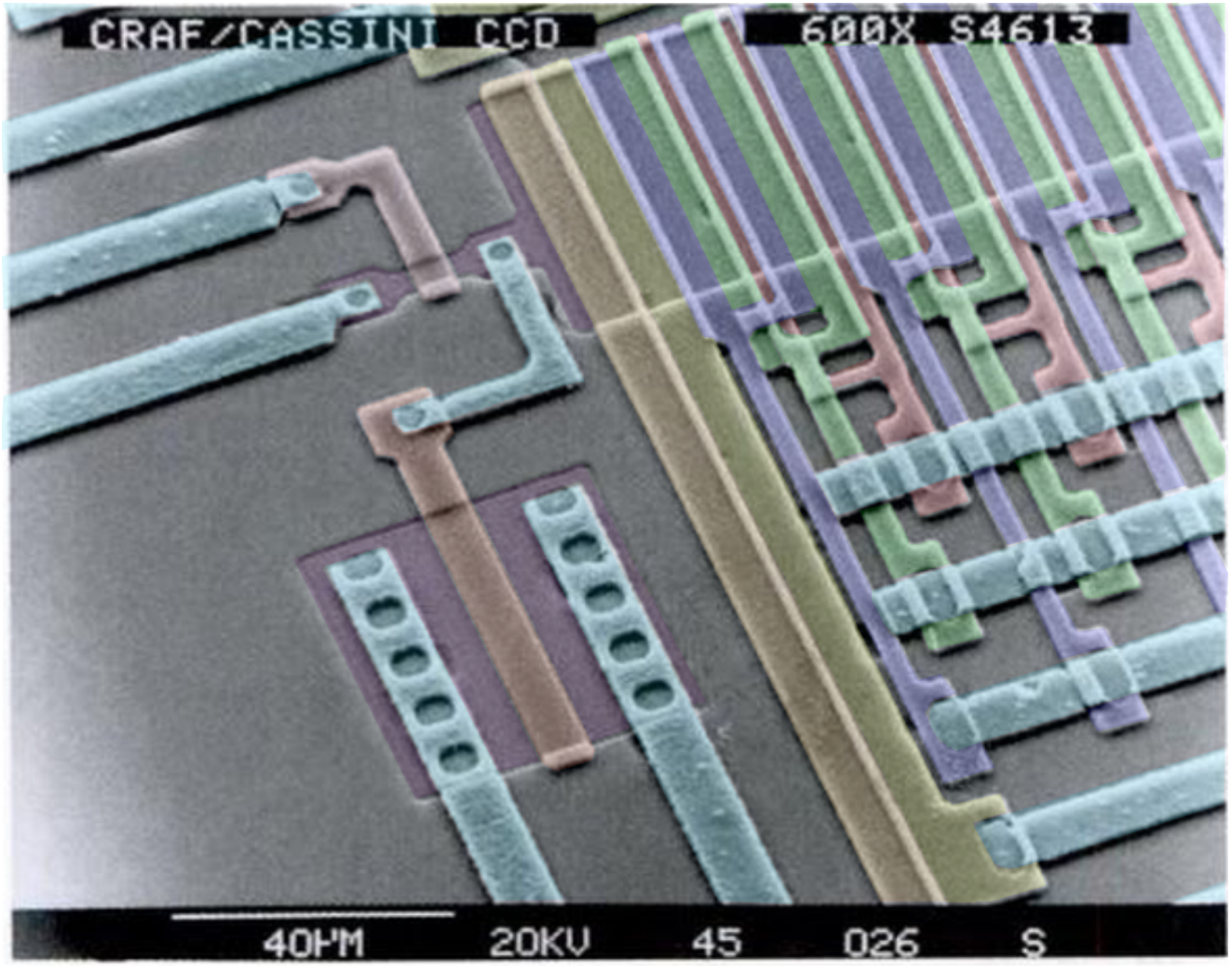
Because someone has to understand them in detail, and it may be you.

Real CCDs (a Tektronix CCD; Janesick)



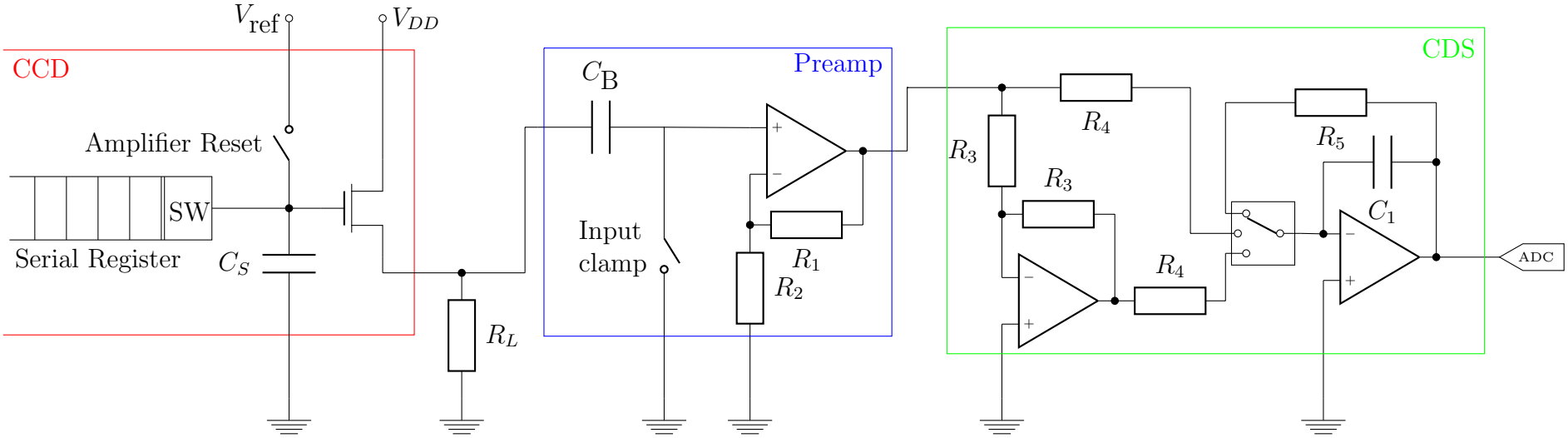
Corner of active area, parallel gates, serial register, and output capacitor

Real CCDs (a Tektronix CCD; Janesick)



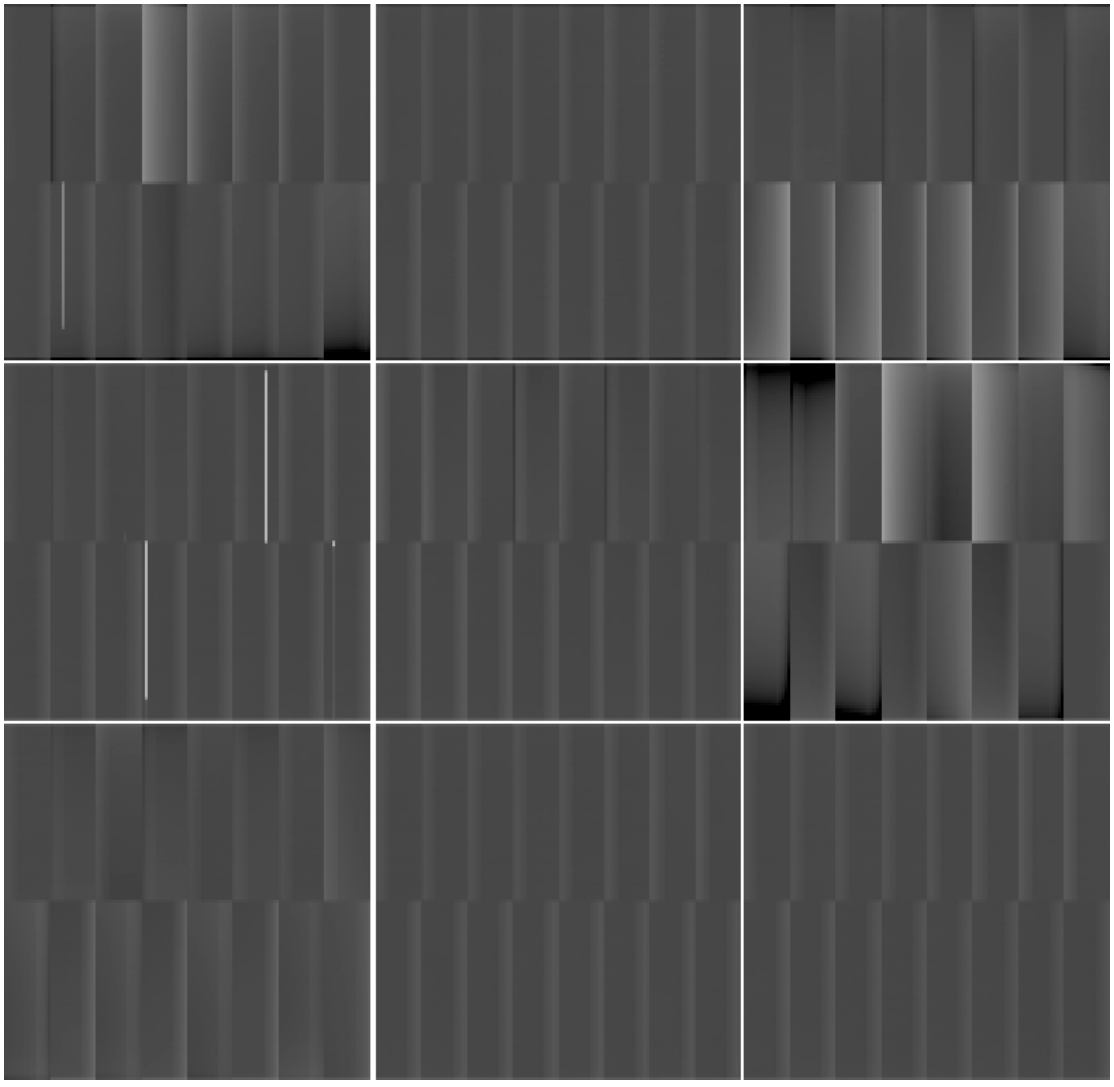
Serial register, reset gate, and output capacitor

Real CCDs (a Tektronix CCD; Janesick)



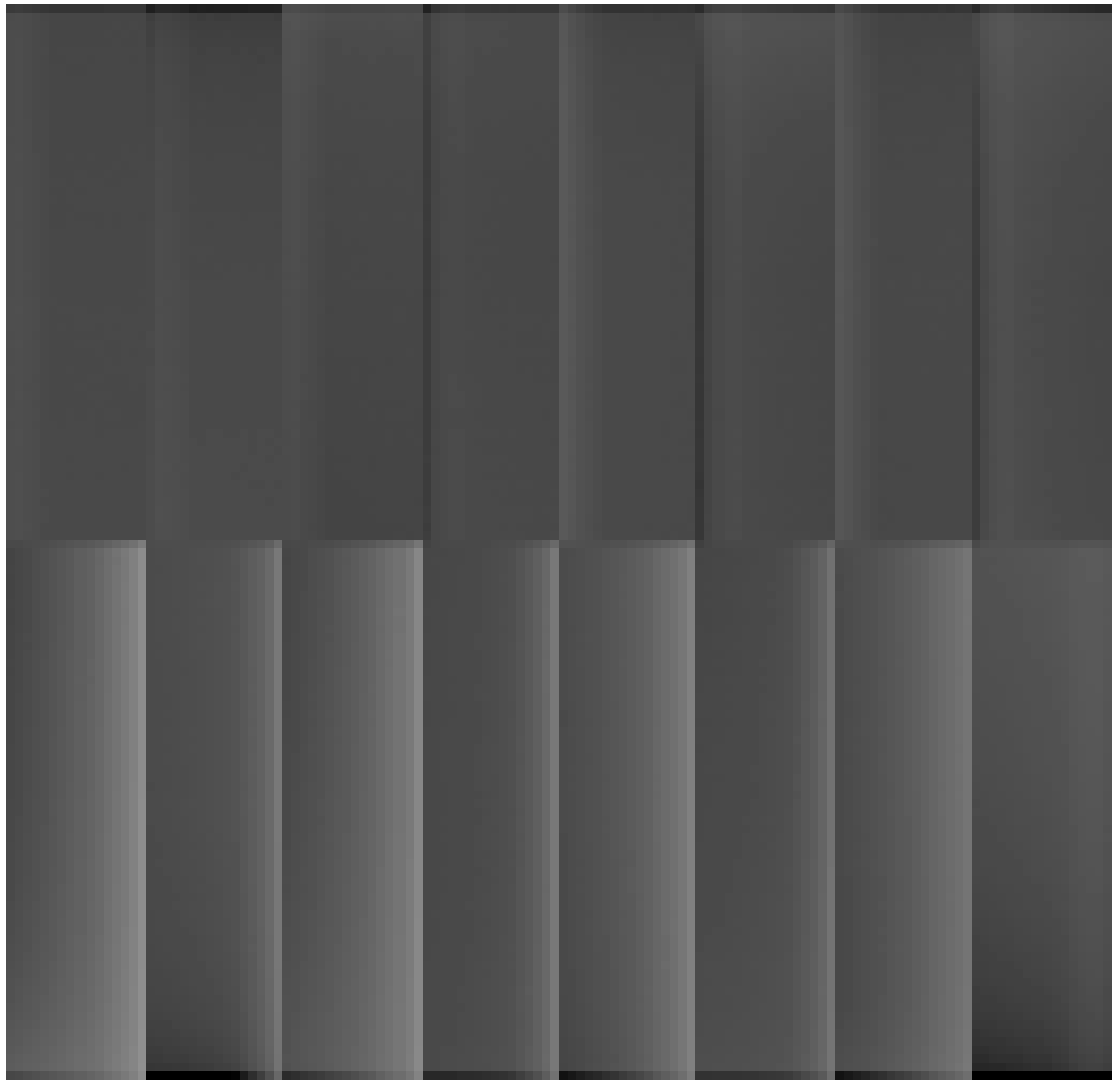
Video chain

Why does my data look like this?



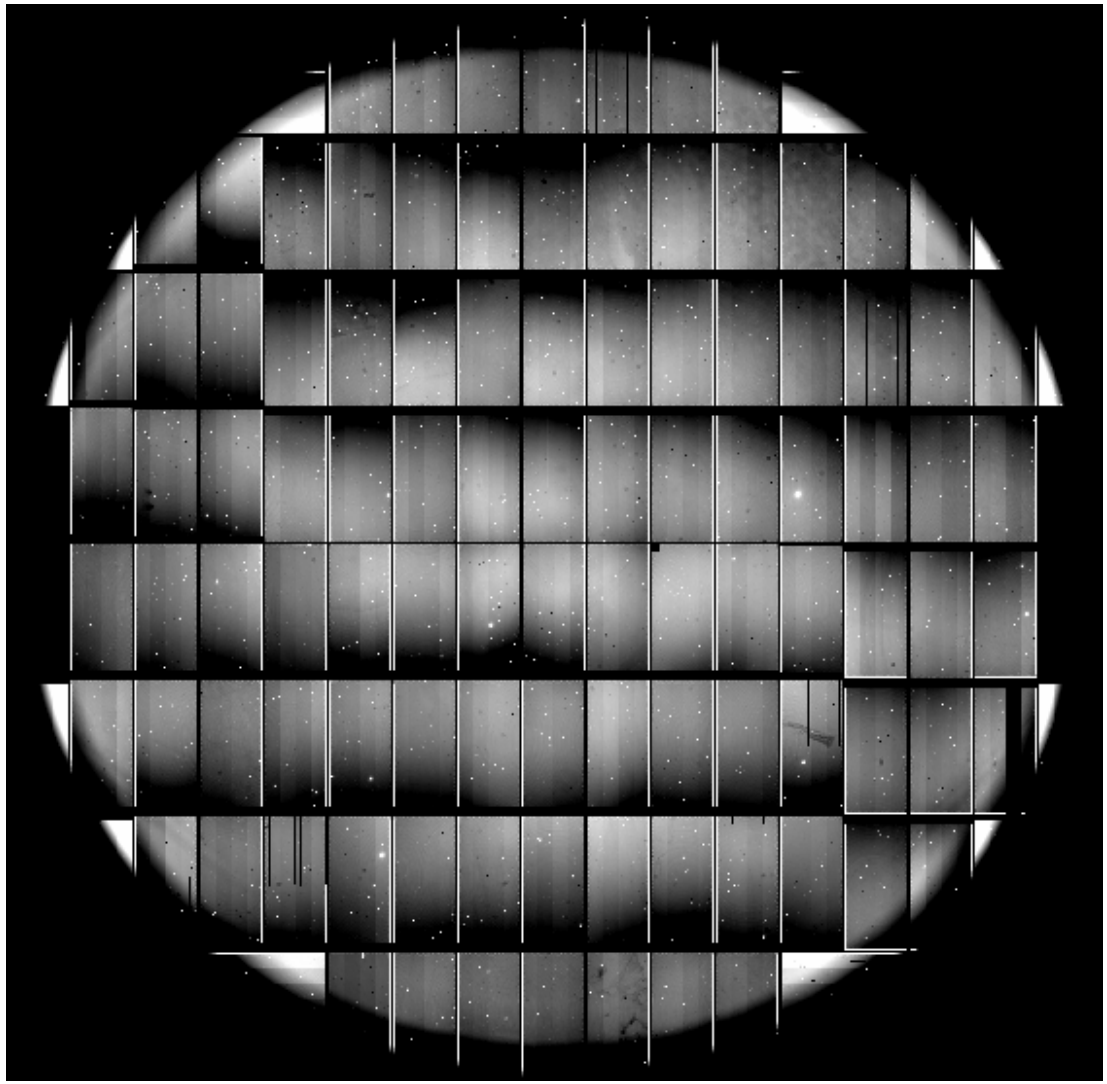
LSST Raft RTM5

Why does my data look like this?



LSST Raft RTM5 S22

Why does my data look like this?



HSC g-band

Why does my data look like this?

What’s going on?

Why does my data look like this?

What’s going on?

The LSST data shows capacitive feed-through from the serial register, and it takes a while for the voltage levels to recover after a parallel transfer (basically an *RC* time).

Why does my data look like this?

What's going on?

The LSST data shows capacitive feed-through from the serial register, and it takes a while for the voltage levels to recover after a parallel transfer (basically an *RC* time).

The HSC data shows spatial variation of the CCD's quantum efficiency (probably the thickness of the anti-reflection coatings) with distance from the serial register.

Why does my data look like this?

What's going on?

The LSST data shows capacitive feed-through from the serial register, and it takes a while for the voltage levels to recover after a parallel transfer (basically an *RC* time).

The HSC data shows spatial variation of the CCD's quantum efficiency (probably the thickness of the anti-reflection coatings) with distance from the serial register. You are also seeing spatial variation in the properties of the filter.

You might hope that all these problems live in the realm of the electrical and optical engineers

Why does my data look like this?

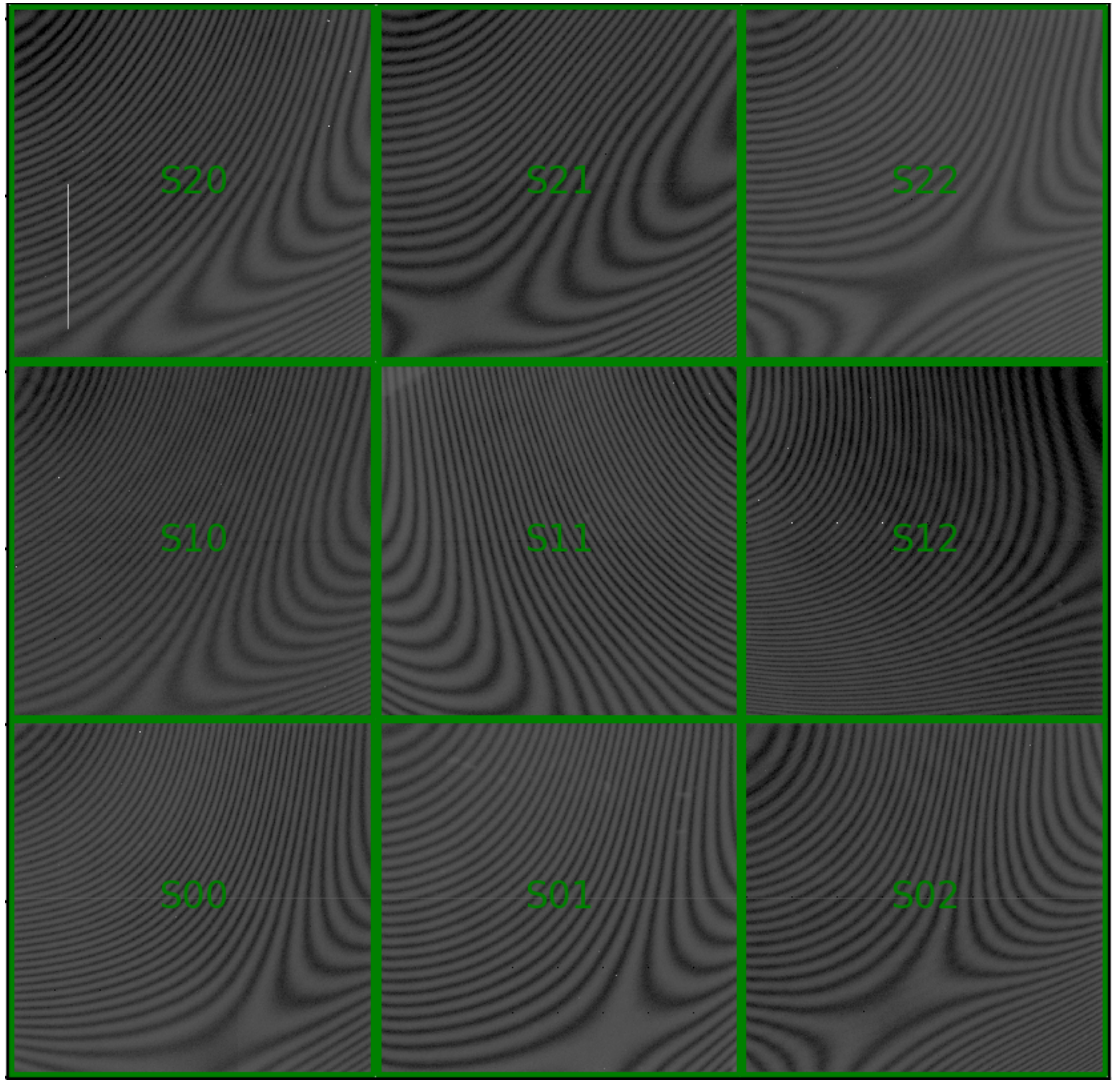
What's going on?

The LSST data shows capacitive feed-through from the serial register, and it takes a while for the voltage levels to recover after a parallel transfer (basically an *RC* time).

The HSC data shows spatial variation of the CCD's quantum efficiency (probably the thickness of the anti-reflection coatings) with distance from the serial register. You are also seeing spatial variation in the properties of the filter.

You might hope that all these problems live in the realm of the electrical and optical engineers; unfortunately, while we have brilliant colleagues, many CCD issues end up in our data waiting for us to find.

LSST Raft RTM5 post-ISR



Large Instruments mean Large Data

- The PFUEI data rate was *c.* 600 B/s (900s)
- The 4-Shooter data rate was *c.* 6 kB/s (900s)
- The SDSS data rate was *c.* 8 MB/s (54s)
- The HSC data rate is *c.* 7 MB/s (240s)
- The LSST data rate will be *c.* 400 MB/s (15s)

Large Instruments mean Large Data

- The PFUEI data rate was *c.* 600 B/s (900s)
- The 4-Shooter data rate was *c.* 6 kB/s (900s)
- The SDSS data rate was *c.* 8 MB/s (54s)
- The HSC data rate is *c.* 7 MB/s (240s)
- The LSST data rate will be *c.* 400 MB/s (15s)

4-shooter’s data rate was *c.* 10 cm/s on a 1600bpi tape (2400’; 730m)

Large Instruments mean Large Data

- The PFUEI data rate was *c.* 600 B/s (900s)
- The 4-Shooter data rate was *c.* 6 kB/s (900s)
- The SDSS data rate was *c.* 8 MB/s (54s)
- The HSC data rate is *c.* 7 MB/s (240s)
- The LSST data rate will be *c.* 400 MB/s (15s)

4-shooter’s data rate was *c.* 10 cm/s on a 1600bpi tape (2400’; 730m)
SDSS’s data rate was *c.* 6 3.5" HD floppy disks/s

Large Instruments mean Large Data

- The PFUEI data rate was *c.* 600 B/s (900s)
- The 4-Shooter data rate was *c.* 6 kB/s (900s)
- The SDSS data rate was *c.* 8 MB/s (54s)
- The HSC data rate is *c.* 7 MB/s (240s)
- The LSST data rate will be *c.* 400 MB/s (15s)

4-shooter’s data rate was *c.* 10 cm/s on a 1600bpi tape (2400’; 730m)
SDSS’s data rate was *c.* 6 3.5" HD floppy disks/s. But we actually used DLTs

Large Instruments mean Large Data

- The PFUEI data rate was *c.* 600 B/s (900s)
- The 4-Shooter data rate was *c.* 6 kB/s (900s)
- The SDSS data rate was *c.* 8 MB/s (54s)
- The HSC data rate is *c.* 7 MB/s (240s)
- The LSST data rate will be *c.* 400 MB/s (15s)

4-shooter's data rate was *c.* 10 cm/s on a 1600bpi tape (2400'; 730m)
SDSS's data rate was *c.* 6 3.5" HD floppy disks/s. But we actually used DLTs

We need large disks and large computer systems

Large Computing

Because Moore’s Law applies to detectors as well as processors, we’ve actually been doing Big Data for 40 years.

Large Computing

Because Moore’s Law applies to detectors as well as processors, we’ve actually been doing Big Data for 40 years. Reducing 4-Shooter data on a VAX-750 was a challenge.

Large Computing

Because Moore's Law applies to detectors as well as processors, we've actually been doing Big Data for 40 years. Reducing 4-Shooter data on a VAX-750 was a challenge.

E.g. rather than dividing by a flatfield image, it was worth computing the inverse flatfield and multiplying.

Large Computing

Because Moore's Law applies to detectors as well as processors, we've actually been doing Big Data for 40 years. Reducing 4-Shooter data on a VAX-750 was a challenge.

E.g. rather than dividing by a flatfield image, it was worth computing the inverse flatfield and multiplying. Or computing `int(8192.0/flatfield)`, multiplying, and >>= 13.

Large Computing

Because Moore's Law applies to detectors as well as processors, we've actually been doing Big Data for 40 years. Reducing 4-Shooter data on a VAX-750 was a challenge.

E.g. rather than dividing by a flatfield image, it was worth computing the inverse flatfield and multiplying. Or computing `int(8192.0/flatfield)`, multiplying, and >>= 13.

If we'd bought the computers when construction started on the LSST project, it would been one of the largest machines in the (public) world.

Large Computing

Because Moore's Law applies to detectors as well as processors, we've actually been doing Big Data for 40 years. Reducing 4-Shooter data on a VAX-750 was a challenge.

E.g. rather than dividing by a flatfield image, it was worth computing the inverse flatfield and multiplying. Or computing `int(8192.0/flatfield)`, multiplying, and >>= 13.

If we'd bought the computers when construction started on the LSST project, it would been one of the largest machines in the (public) world.

The final LSST processing machines will have 70 kCores, 1200 TFLOPS

- around 30 on the June 2014 Top-500 list
- about 115 on the November 2017 Top-500 list
- unremarkable by the time we buy it

Large Computing

Because Moore's Law applies to detectors as well as processors, we've actually been doing Big Data for 40 years. Reducing 4-Shooter data on a VAX-750 was a challenge.

E.g. rather than dividing by a flatfield image, it was worth computing the inverse flatfield and multiplying. Or computing `int(8192.0/flatfield)`, multiplying, and >>= 13.

If we'd bought the computers when construction started on the LSST project, it would been one of the largest machines in the (public) world.

The final LSST processing machines will have 70 kCores, 1200 TFLOPS

- around 30 on the June 2014 Top-500 list
- about 115 on the November 2017 Top-500 list
- unremarkable by the time we buy it

We need to write efficient codes

Large Computing

Because Moore's Law applies to detectors as well as processors, we've actually been doing Big Data for 40 years. Reducing 4-Shooter data on a VAX-750 was a challenge.

E.g. rather than dividing by a flatfield image, it was worth computing the inverse flatfield and multiplying. Or computing `int(8192.0/flatfield)`, multiplying, and >>= 13.

If we'd bought the computers when construction started on the LSST project, it would been one of the largest machines in the (public) world.

The final LSST processing machines will have 70 kCores, 1200 TFLOPS

- around 30 on the June 2014 Top-500 list
- about 115 on the November 2017 Top-500 list
- unremarkable by the time we buy it

We need to write efficient codes, or buy *very* large computer systems

Large Computing

Because Moore's Law applies to detectors as well as processors, we've actually been doing Big Data for 40 years. Reducing 4-Shooter data on a VAX-750 was a challenge.

E.g. rather than dividing by a flatfield image, it was worth computing the inverse flatfield and multiplying. Or computing `int(8192.0/flatfield)`, multiplying, and >>= 13.

If we'd bought the computers when construction started on the LSST project, it would been one of the largest machines in the (public) world.

The final LSST processing machines will have 70 kCores, 1200 TFLOPS

- around 30 on the June 2014 Top-500 list
- about 115 on the November 2017 Top-500 list
- unremarkable by the time we buy it

We need to write efficient codes, or buy *very* large computer systems. Or wait a very long time.

Large Computing

The "large" problem doesn’t worry me too much; the rest of the world uses more compute than we do and our hardware is fixed.

Large Computing

The "large" problem doesn’t worry me too much; the rest of the world uses more compute than we do and our hardware is fixed.

Writing efficient code might seem to be mostly a financial question

Large Computing

The "large" problem doesn’t worry me too much; the rest of the world uses more compute than we do and our hardware is fixed.
Writing efficient code might seem to be mostly a financial question; you need to hire good programmers, but they needn’t know anything about astronomy.

Large Computing

The "large" problem doesn’t worry me too much; the rest of the world uses more compute than we do and our hardware is fixed.

Writing efficient code might seem to be mostly a financial question; you need to hire good programmers, but they needn’t know anything about astronomy.

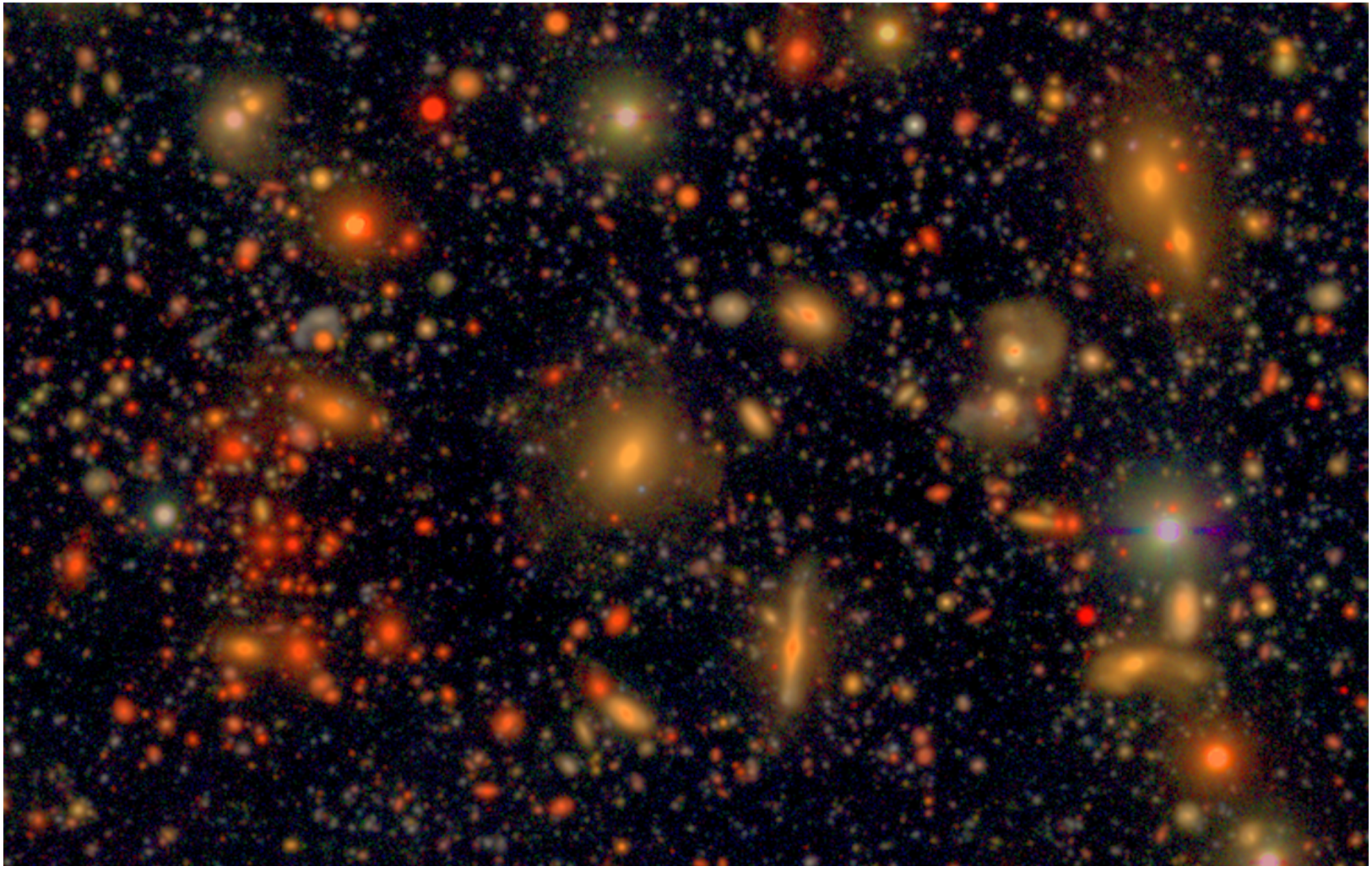
I am not convinced that this is true, even if we could compete financially with Google and Microsoft.

Memory

Inefficient codes may also require lots of memory

Memory

Inefficient codes may also require lots of memory; actually, efficient codes may too.



Memory

Inefficient codes may also require lots of memory; actually, efficient codes may too.

Most of our processing is trivially parallel so you might hope to avoid learning `MPI` or `openMP` or `CUDA` or `openCL`.

Memory

Inefficient codes may also require lots of memory; actually, efficient codes may too.

Most of our processing is trivially parallel so you might hope to avoid learning `MPI` or `openMP` or `CUDA` or `openCL`.

What actually matters more is the required memory per core;

Memory

Inefficient codes may also require lots of memory; actually, efficient codes may too.

Most of our processing is trivially parallel so you might hope to avoid learning `MPI` or `openMP` or `CUDA` or `openCL`.

What actually matters more is the required memory per core; the next generation of processors are expected to have ≥ 50 cores and run at ≥ 3 GHz.

Memory

Inefficient codes may also require lots of memory; actually, efficient codes may too.

Most of our processing is trivially parallel so you might hope to avoid learning `MPI` or `openMP` or `CUDA` or `openCL`.

What actually matters more is the required memory per core; the next generation of processors are expected to have ≥ 50 cores and run at ≥ 3 GHz.

> *If you were plowing a field, which would you rather use: Two strong oxen or 1024 chickens?*
> *Seymour Cray*

Memory

Inefficient codes may also require lots of memory; actually, efficient codes may too.

Most of our processing is trivially parallel so you might hope to avoid learning `MPI` or `openMP` or `CUDA` or `openCL`.

What actually matters more is the required memory per core; the next generation of processors are expected to have ≥ 50 cores and run at ≥ 3 GHz.

LSST is hoping that cores will be smart enough to not need GPU-style programming, and memory large enough that our work will fit on a (smart) node.

Memory

Inefficient codes may also require lots of memory; actually, efficient codes may too.

Most of our processing is trivially parallel so you might hope to avoid learning `MPI` or `openMP` or `CUDA` or `openCL`.

What actually matters more is the required memory per core; the next generation of processors are expected to have ≥ 50 cores and run at ≥ 3 GHz.

LSST is hoping that cores will be smart enough to not need GPU-style programming, and memory large enough that our work will fit on a (smart) node. So maybe only `openMP`?

Memory

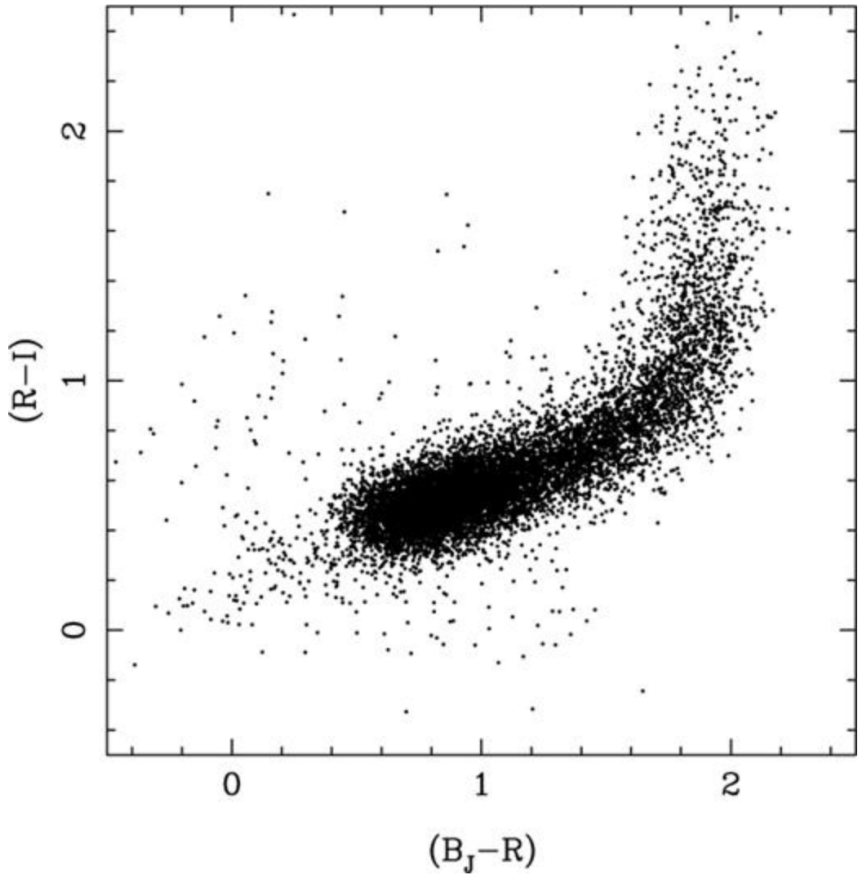
Inefficient codes may also require lots of memory; actually, efficient codes may too.

Most of our processing is trivially parallel so you might hope to avoid learning `MPI` or `openMP` or `CUDA` or `openCL`.

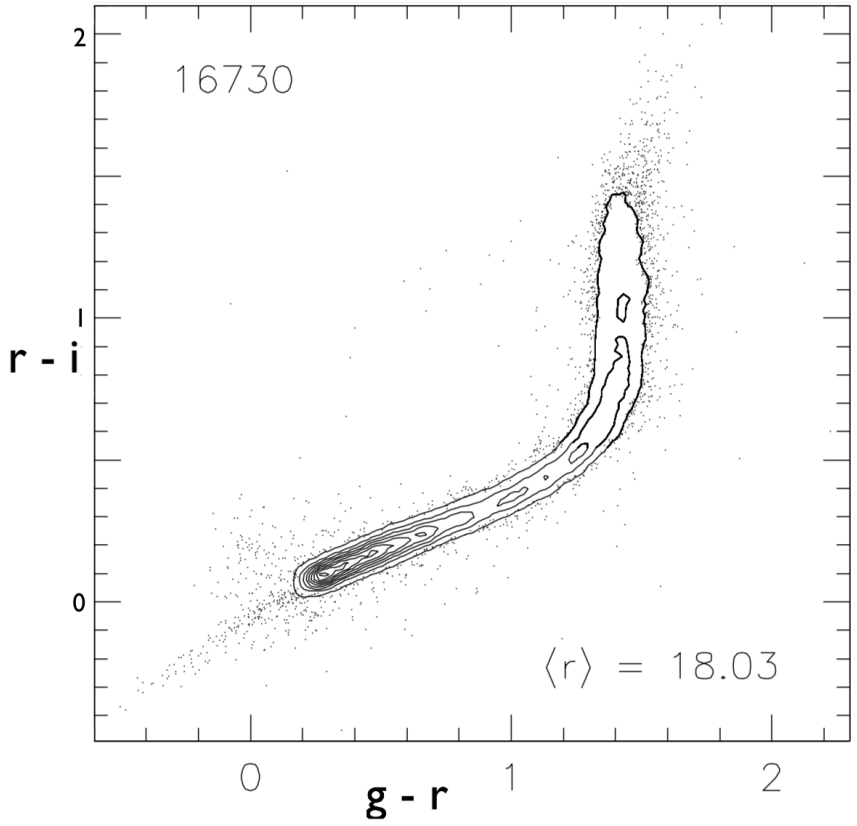
What actually matters more is the required memory per core; the next generation of processors are expected to have ≥ 50 cores and run at ≥ 3 GHz.

LSST is hoping that cores will be smart enough to not need GPU-style programming, and memory large enough that our work will fit on a (smart) node. So maybe only `openMP`? But beware Python's `GIL`.

Better Data → Better Science

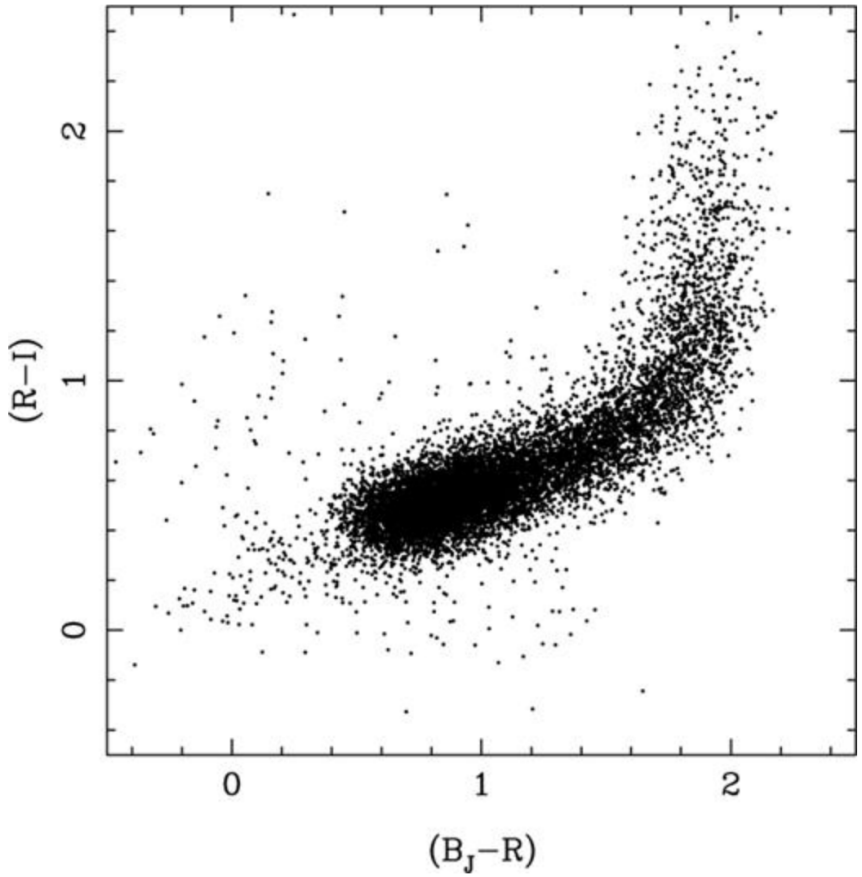


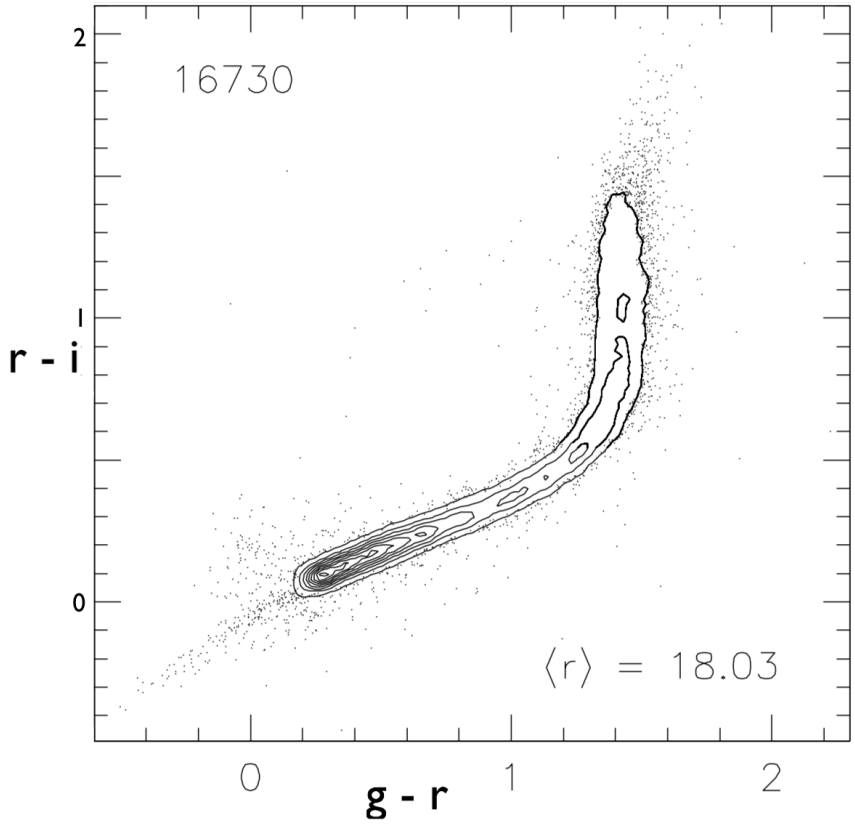
SuperCosmos (Hambly *et al.* 2001)



SDSS, *c.* 2002

Better Data → Better Science





SuperCosmos (Hambly *et al.* 2001)

SDSS, *c.* 2002

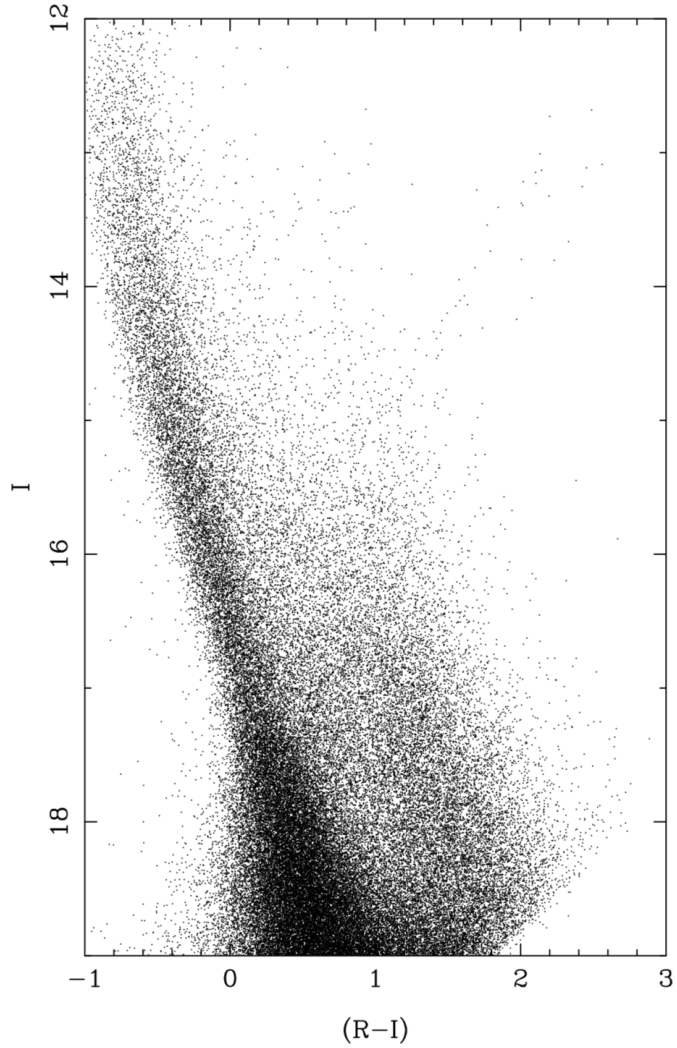
On the left, scanned Palomar plates; on the right SDSS CCD data.

Better Data $\rightarrow$ Better Analysis

Getting something as good as that SuperCOSMOS colour-colour diagram took heroic efforts fighting photographic emulsions:

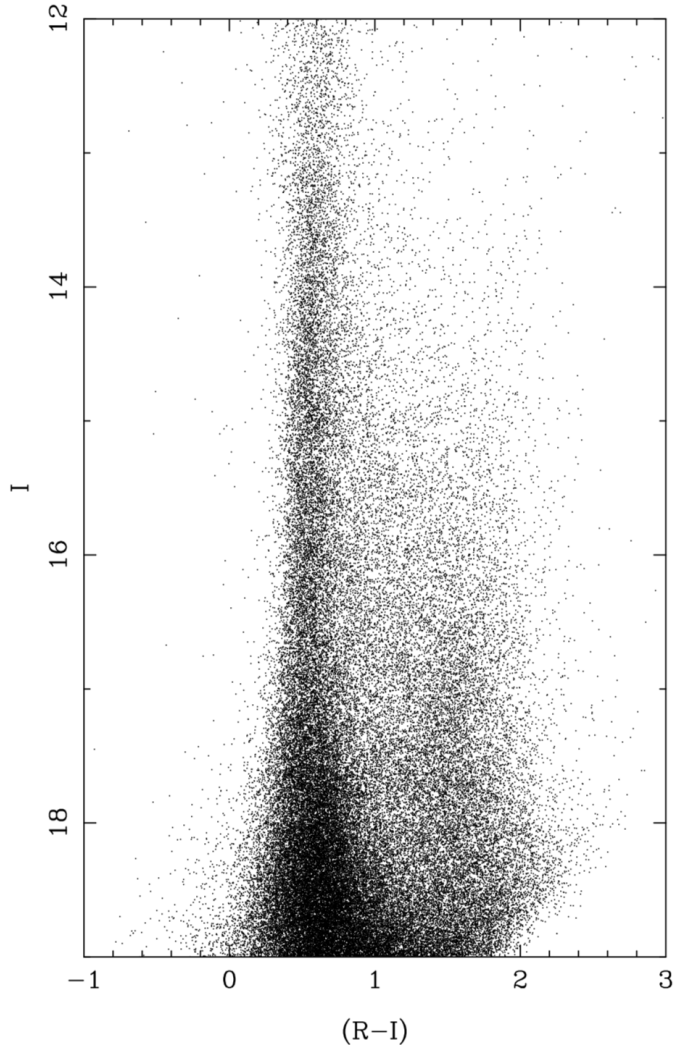
Better Data → Better Analysis

Getting something as good as that SuperCOSMOS colour-colour diagram took heroic efforts fighting photographic emulsions:



Better Data → Better Analysis

Getting something as good as that SuperCOSMOS colour-colour diagram took heroic efforts fighting photographic emulsions:



Better Data → Better Analysis

Getting something as good as that SuperCOSMOS colour-colour diagram took heroic efforts fighting photographic emulsions:
CCDs have their little ways, but most of our challenge in SDSS was algorithmic/statistical.

Better Data → Better Analysis

Getting something as good as that SuperCOSMOS colour-colour diagram took heroic efforts fighting photographic emulsions:
CCDs have their little ways, but most of our challenge in SDSS was algorithmic/statistical. *E.g.* for stellar photometry should we use:

- Fixed aperture photometry
- PSF fit photometry
- Galaxy-model fit photometry
- Adaptive aperture (*e.g.* Petrosian) photometry

Better Data → Better Analysis

Getting something as good as that SuperCOSMOS colour-colour diagram took heroic efforts fighting photographic emulsions:
CCDs have their little ways, but most of our challenge in SDSS was algorithmic/statistical. *E.g.* for stellar photometry should we use:

- Fixed aperture photometry
- PSF fit photometry
- Galaxy-model fit photometry
- Adaptive aperture (*e.g.* Petrosian) photometry

Once you think about even humble aperture photometry it's quite subtle

Better Data → Better Analysis

Getting something as good as that SuperCOSMOS colour-colour diagram took heroic efforts fighting photographic emulsions:
CCDs have their little ways, but most of our challenge in SDSS was algorithmic/statistical. *E.g.* for stellar photometry should we use:

- Fixed aperture photometry
- PSF fit photometry
- Galaxy-model fit photometry
- Adaptive aperture (*e.g.* Petrosian) photometry

Once you think about even humble aperture photometry it's quite subtle

- How does the surface brightness vary across the pixel?

Better Data → Better Analysis

Getting something as good as that SuperCOSMOS colour-colour diagram took heroic efforts fighting photographic emulsions:
CCDs have their little ways, but most of our challenge in SDSS was algorithmic/statistical. *E.g.* for stellar photometry should we use:

- Fixed aperture photometry
- PSF fit photometry
- Galaxy-model fit photometry
- Adaptive aperture (*e.g.* Petrosian) photometry

Once you think about even humble aperture photometry it's quite subtle

- How does the surface brightness vary across the pixel?
- What do the pixel values mean anyway?

Better Data → Better Analysis

Getting something as good as that SuperCOSMOS colour-colour diagram took heroic efforts fighting photographic emulsions:
CCDs have their little ways, but most of our challenge in SDSS was algorithmic/statistical. *E.g.* for stellar photometry should we use:

- Fixed aperture photometry
- PSF fit photometry
- Galaxy-model fit photometry
- Adaptive aperture (*e.g.* Petrosian) photometry

Once you think about even humble aperture photometry it's quite subtle

- How does the surface brightness vary across the pixel?
- What do the pixel values mean anyway?
- What did the flat-field do to those values?

Better Data → Better Analysis

Getting something as good as that SuperCOSMOS colour-colour diagram took heroic efforts fighting photographic emulsions:
CCDs have their little ways, but most of our challenge in SDSS was algorithmic/statistical. *E.g.* for stellar photometry should we use:

- Fixed aperture photometry
- PSF fit photometry
- Galaxy-model fit photometry
- Adaptive aperture (*e.g.* Petrosian) photometry

Once you think about even humble aperture photometry it's quite subtle

- How does the surface brightness vary across the pixel?
- What do the pixel values mean anyway?
- What did the flat-field do to those values?
- How should I relate the measured DN to the flux of standard stars?

Better Data → Better Analysis

Getting something as good as that SuperCOSMOS colour-colour diagram took heroic efforts fighting photographic emulsions:
CCDs have their little ways, but most of our challenge in SDSS was algorithmic/statistical. *E.g.* for stellar photometry should we use:

- Fixed aperture photometry
- PSF fit photometry
- Galaxy-model fit photometry
- Adaptive aperture (*e.g.* Petrosian) photometry

Once you think about even humble aperture photometry it's quite subtle

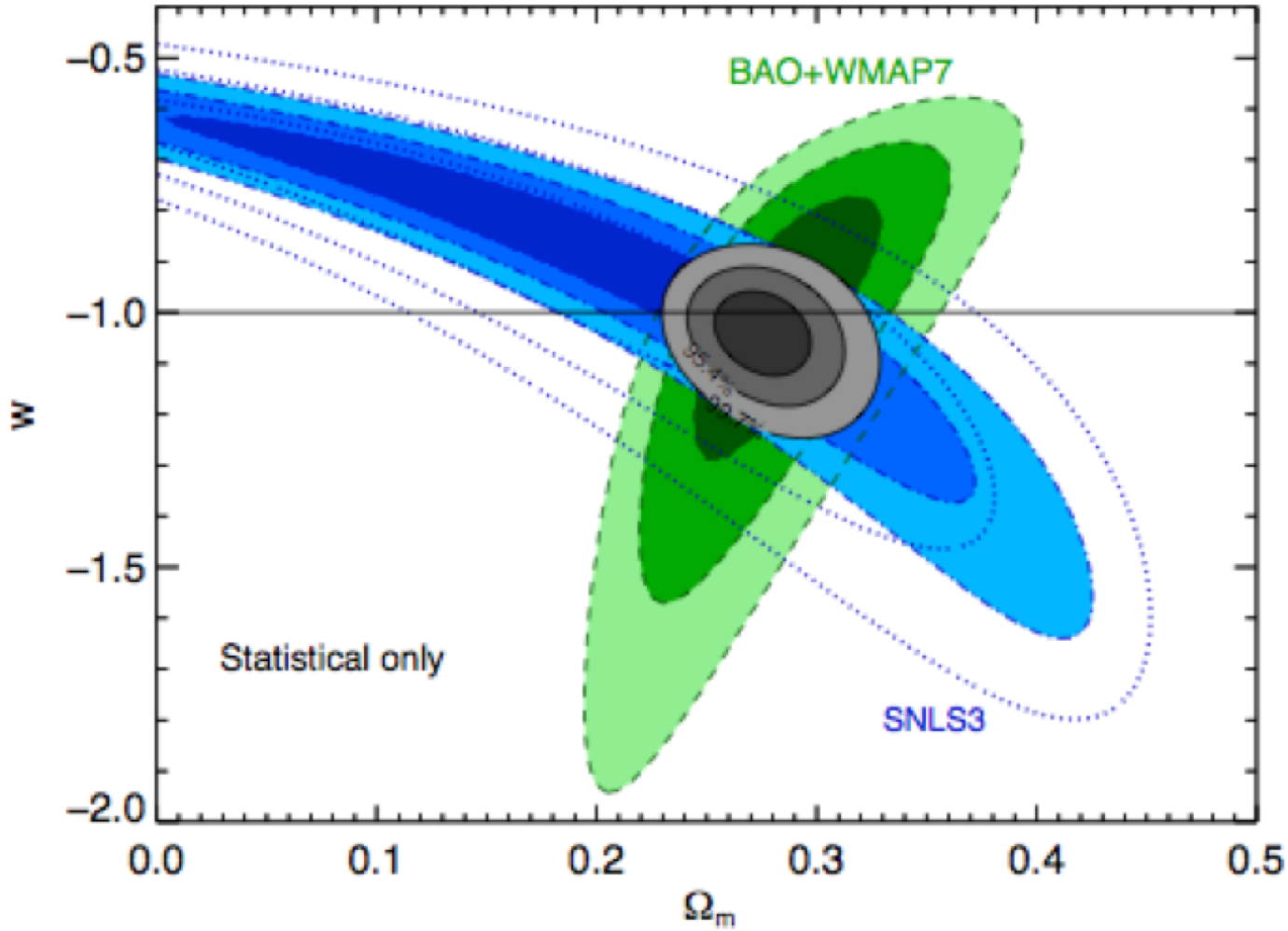
- How does the surface brightness vary across the pixel?
- What do the pixel values mean anyway?
- What did the flat-field do to those values?
- How should I relate the measured DN to the flux of standard stars?
- How about to Janskys?

Large Data → Systematics

- As *statistical* errors become smaller, *systematic* errors come to dominate.

Large Data → Systematics

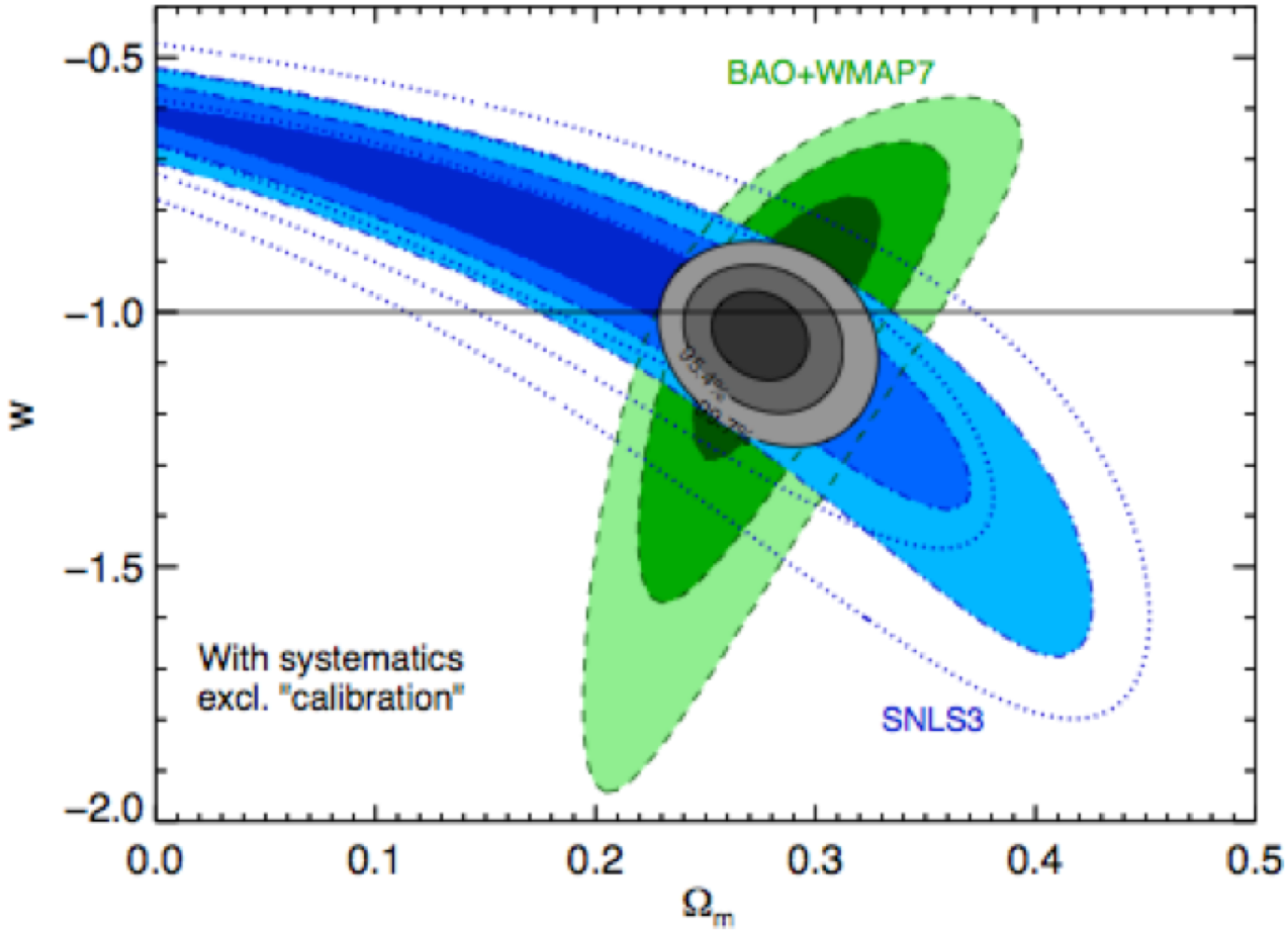
- As *statistical* errors become smaller, *systematic* errors come to dominate.



Betoule *et al.* SNLS

Large Data → Systematics

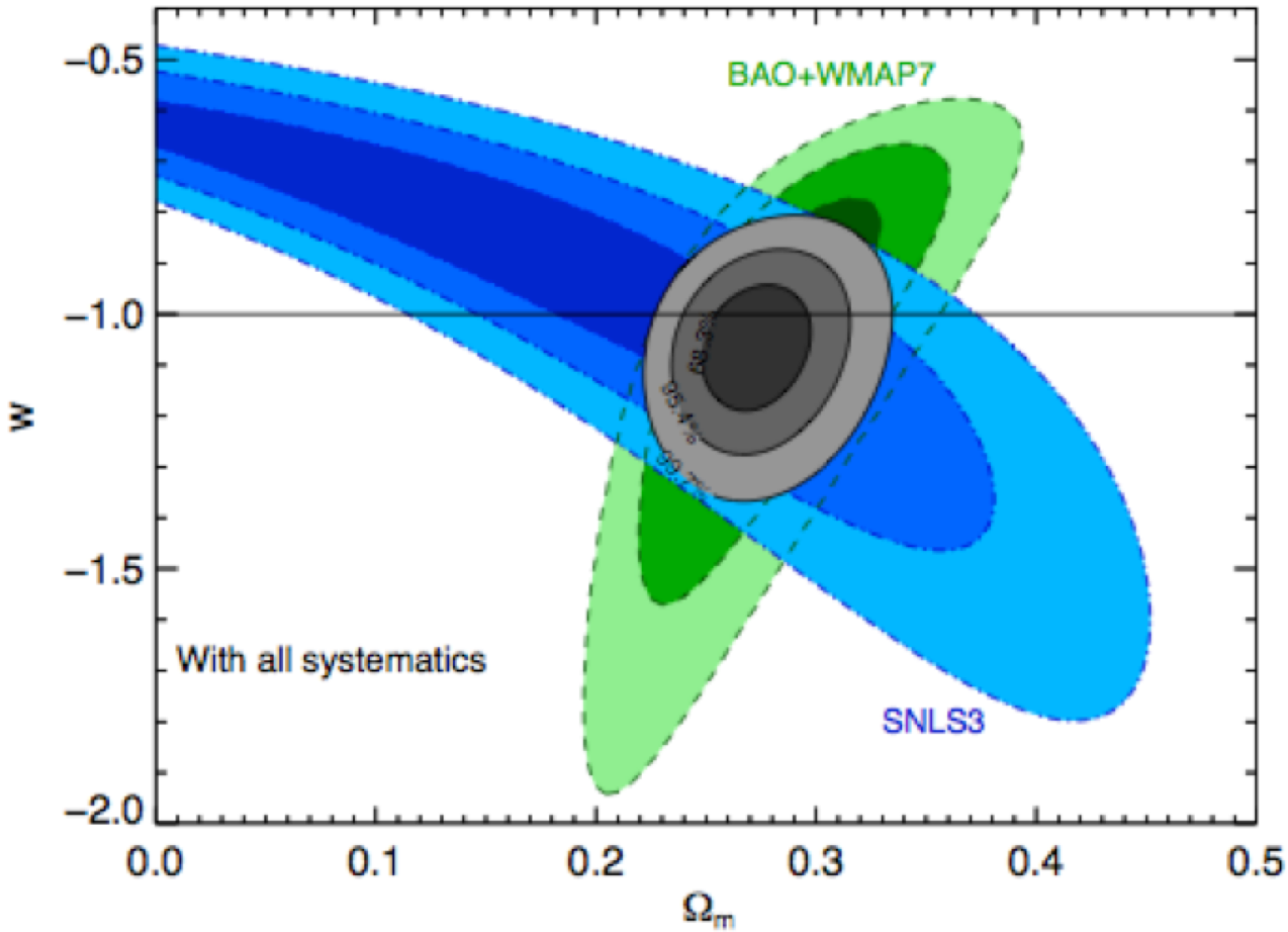
– As *statistical* errors become smaller, *systematic* errors come to dominate.



Betoule *et al.* SNLS

Large Data → Systematics

– As *statistical* errors become smaller, *systematic* errors come to dominate.



Betoule *et al.* SNLS

Large Data $\rightarrow$ Systematics

- As *statistical* errors become smaller, *systematic* errors come to dominate.
  Only about 50% of SNLS's variance in Ω_m based on *c.* 200 high-z SNe Ia is statistical.

Large Data $\rightarrow$ Systematics

- As *statistical* errors become smaller, *systematic* errors come to dominate.
  Only about 50% of SNLS's variance in Ω_m based on *c.* 200 high-z SNe Ia is statistical. LSST will detect *c.* 5×10^5 SNe Ia, of which maybe 20000 will be useful for cosmological analysis.

Large Data → Systematics

Software can contribute to systematic errors.

Large Data → Systematics

Software can contribute to systematic errors. Shhhhhh!

Large Data $\rightarrow$ Systematics

Software can contribute to systematic errors. Shhhhhh!

For example, a statistical model for an image containing a star (with known PSF ϕ) at $\boldsymbol{x}$ is

$$I(\boldsymbol{x}) = A\phi(\boldsymbol{x} - \boldsymbol{x}_0) + \boldsymbol{\epsilon}$$

where I have subtracted the background level.

Large Data $\rightarrow$ Systematics

Software can contribute to systematic errors. Shhhhhh!
For example, a statistical model for an image containing a star (with known PSF ϕ) at $\boldsymbol{x}$ is

$$I(\boldsymbol{x}) = A\phi(\boldsymbol{x} - \boldsymbol{x}_0) + \boldsymbol{\epsilon}$$

where I have subtracted the background level. If the noise $\boldsymbol{\epsilon}$ has known variance σ^2, and if I know the position, the (unbiased) optimal linear estimator for the flux A is

$$\hat{A} = \frac{\sum_i \phi(\boldsymbol{x}_i - \boldsymbol{x}_0) I(\boldsymbol{x}_i)/\sigma_i^2}{\sum_i \phi(\boldsymbol{x}_i - \boldsymbol{x}_0)^2/\sigma_i^2}$$

and this is the MLE (and attains the Cramér-Rao bound) if the noise is Gaussian.

Large Data $\rightarrow$ Systematics

If the star is faint, the noise is the same in every pixel ($\sigma_i^2 = \sigma_0^2$) and this reduces to

$$\hat{A} = \frac{\sum_i \phi(\boldsymbol{x}_i - \boldsymbol{x}_0) I(\boldsymbol{x}_i)}{\sum_i \phi(\boldsymbol{x}_i - \boldsymbol{x}_0)^2}$$

Large Data $\rightarrow$ Systematics

If the star is faint, the noise is the same in every pixel ($\sigma_i^2 = \sigma_0^2$) and this reduces to

$$\hat{A} = \frac{\sum_i \phi(\boldsymbol{x}_i - \boldsymbol{x}_0) I(\boldsymbol{x}_i)}{\sum_i \phi(\boldsymbol{x}_i - \boldsymbol{x}_0)^2}$$

I can in fact spurn my statistical friends and replace $\phi(\boldsymbol{x}_i - \boldsymbol{x}_0)/\sigma_i^2$ with any weight function w_i of my choice:

$$\begin{aligned}\hat{A} =& \frac{\sum_i w_i I(\boldsymbol{x}_i)}{\sum_i w_i \phi(\boldsymbol{x}_i - \boldsymbol{x}_0)} \\ \equiv& C \sum_i w_i I(\boldsymbol{x}_i)\end{aligned}$$

e.g.

$$w_i = \begin{cases} 1 & |\boldsymbol{x}_i - \boldsymbol{x}_0| < R \\ 0 & \text{otherwise} \end{cases}$$

Large Data → Systematics

If the star is faint, the noise is the same in every pixel ($\sigma_i^2 = \sigma_0^2$) and this reduces to

$$\hat{A} = \frac{\sum_i \phi(\boldsymbol{x}_i - \boldsymbol{x}_0) I(\boldsymbol{x}_i)}{\sum_i \phi(\boldsymbol{x}_i - \boldsymbol{x}_0)^2}$$

I can in fact spurn my statistical friends and replace $\phi(\boldsymbol{x}_i - \boldsymbol{x}_0)/\sigma_i^2$ with any weight function w_i of my choice:

$$\begin{aligned}\hat{A} =& \frac{\sum_i w_i I(\boldsymbol{x}_i)}{\sum_i w_i \phi(\boldsymbol{x}_i - \boldsymbol{x}_0)} \\ \equiv& C \sum_i w_i I(\boldsymbol{x}_i)\end{aligned}$$

e.g.

$$w_i = \begin{cases} 1 & |\boldsymbol{x}_i - \boldsymbol{x}_0| < R \\ 0 & \text{otherwise} \end{cases}$$

How should we choose w?

Large Data $\rightarrow$ Systematics

- If you want as small a *statistical* error as possible, $w_i = \phi(\boldsymbol{x}_i)/\sigma_i^2$
- If you want as small a *total* error as possible for *faint* sources, use $w_i = \phi(\boldsymbol{x}_i)$
- If you want as small a *total* error as possible for *bright* sources, use $w_i = (r < R\,?\,1 : 0)$ where R is large enough to include most of the photons and not too much background.

Large Data $\rightarrow$ Systematics

- If you want as small a *statistical* error as possible, $w_i = \phi(\boldsymbol{x}_i)/\sigma_i^2$
- If you want as small a *total* error as possible for *faint* sources, use $w_i = \phi(\boldsymbol{x}_i)$
- If you want as small a *total* error as possible for *bright* sources, use $w_i = (r < R \,?\, 1 : 0)$ where R is large enough to include most of the photons and not too much background. You'll probably be dominated by the uncertainty in C, and in all the difficulties in calibration.

Large Data $\rightarrow$ Systematics

- If you want as small a *statistical* error as possible, $w_i = \phi(\boldsymbol{x}_i)/\sigma_i^2$
- If you want as small a *total* error as possible for *faint* sources, use $w_i = \phi(\boldsymbol{x}_i)$
- If you want as small a *total* error as possible for *bright* sources, use $w_i = (r < R\,?\,1 : 0)$ where R is large enough to include most of the photons and not too much background. You'll probably be dominated by the uncertainty in C, and in all the difficulties in calibration. *E.g.* whether the wind blows from the East or West changes the wavelength-dependence of the absorption by aerosols in the atmosphere, and thus the effective filter passband.

Large Data $\rightarrow$ Systematics

- If you want as small a *statistical* error as possible, $w_i = \phi(\boldsymbol{x}_i)/\sigma_i^2$
- If you want as small a *total* error as possible for *faint* sources, use $w_i = \phi(\boldsymbol{x}_i)$
- If you want as small a *total* error as possible for *bright* sources, use $w_i = (r < R\,?\,1 : 0)$ where R is large enough to include most of the photons and not too much background. You'll probably be dominated by the uncertainty in C, and in all the difficulties in calibration. *E.g.* whether the wind blows from the East or West changes the wavelength-dependence of the absorption by aerosols in the atmosphere, and thus the effective filter passband.
- And if you're using photographic plates you're on your own

Large Data $\rightarrow$ Systematics

- If you want as small a *statistical* error as possible, $w_i = \phi(\boldsymbol{x}_i)/\sigma_i^2$
- If you want as small a *total* error as possible for *faint* sources, use $w_i = \phi(\boldsymbol{x}_i)$
- If you want as small a *total* error as possible for *bright* sources, use $w_i = (r < R\,?\,1 : 0)$ where R is large enough to include most of the photons and not too much background. You'll probably be dominated by the uncertainty in C, and in all the difficulties in calibration. *E.g.* whether the wind blows from the East or West changes the wavelength-dependence of the absorption by aerosols in the atmosphere, and thus the effective filter passband.
- And if you're using photographic plates you're on your own

There are similar discussions to be had about galaxy photometry, but they are harder as the uses of the fluxes are more complex (and non-linear)

Large Data → Systematics

- If you want as small a *statistical* error as possible, $w_i = \phi(\boldsymbol{x}_i)/\sigma_i^2$
- If you want as small a *total* error as possible for *faint* sources, use $w_i = \phi(\boldsymbol{x}_i)$
- If you want as small a *total* error as possible for *bright* sources, use $w_i = (r < R\,?\,1 : 0)$ where R is large enough to include most of the photons and not too much background. You'll probably be dominated by the uncertainty in C, and in all the difficulties in calibration. *E.g.* whether the wind blows from the East or West changes the wavelength-dependence of the absorption by aerosols in the atmosphere, and thus the effective filter passband.
- And if you're using photographic plates you're on your own

There are similar discussions to be had about galaxy photometry, but they are harder as the uses of the fluxes are more complex (and non-linear); *e.g.* as inputs to photo-z codes used to estimate weak lensing kernels.

Better Analysis → Complex Codes

Better Analysis → Complex Codes

Finally something related to this school!

Better Analysis → Complex Codes

Finally something related to this school!

- You can no longer rely on one person to hold the complete codebase and bug list in their head, exploit the catalogue, and write the papers.

Better Analysis → Complex Codes

Finally something related to this school!

- You can no longer rely on one person to hold the complete codebase and bug list in their head, exploit the catalogue, and write the papers.
- Astronomers have to learn the tools of software engineering

SDSS and Hipparcos

This part of my presentation started out with a conversation with Michael Perryman (then P.I. of GAIA) about Hipparcos and SDSS.

Lessons I Learnt from the SDSS

Lessons I Learnt from the SDSS

- Lesson 1: You need a Project Manager
- Lesson 2: Don’t join under-funded projects
- Lesson 3: Don’t generate an inverted management structure
- Lesson 4: Learn when, what, and how to review

Lessons I Learnt from the SDSS

- Lesson 1: You need a Project Manager
- Lesson 2: Don’t join under-funded projects
- Lesson 3: Don’t generate an inverted management structure
- Lesson 4: Learn when, what, and how to review
- Lesson 5: Practice good software engineering
- Lesson 5b: Don’t Write Your Main Program in C

Lessons I Learnt from the SDSS

- Lesson 1: You need a Project Manager
- Lesson 2: Don’t join under-funded projects
- Lesson 3: Don’t generate an inverted management structure
- Lesson 4: Learn when, what, and how to review
- Lesson 5: Practice good software engineering
- Lesson 5b: Don’t Write Your Main Program in C
- Lesson 6: Distribute Data and Information Freely
- Lesson 7: Strive to ensure that the software takes full advantage of the hardware, even at the beginning of a project

Lessons I Learnt from the SDSS

- Lesson 1: You need a Project Manager
- Lesson 2: Don’t join under-funded projects
- Lesson 3: Don’t generate an inverted management structure
- Lesson 4: Learn when, what, and how to review
- Lesson 5: Practice good software engineering
- Lesson 5b: Don’t Write Your Main Program in C
- Lesson 6: Distribute Data and Information Freely
- Lesson 7: Strive to ensure that the software takes full advantage of the hardware, even at the beginning of a project
- Lesson 8: Treat neither science nor software as a democracy
- Lesson 9: Avoid single points of failure
- Lesson 10: Find some way to reward people working on the project

Lessons I Learnt from the SDSS

- Lesson 1: You need a Project Manager
- Lesson 2: Don’t join under-funded projects
- Lesson 3: Don’t generate an inverted management structure
- Lesson 4: Learn when, what, and how to review
- Lesson 5: Practice good software engineering
- Lesson 5b: Don’t Write Your Main Program in C
- Lesson 6: Distribute Data and Information Freely
- Lesson 7: Strive to ensure that the software takes full advantage of the hardware, even at the beginning of a project
- Lesson 8: Treat neither science nor software as a democracy
- Lesson 9: Avoid single points of failure
- Lesson 10: Find some way to reward people working on the project

I wrote these down in 2002; what have we learned in the last 16 years?

Lessons I Learnt from the SDSS

- Lesson 1: You need a Project Manager
- Lesson 2: Don’t join under-funded projects
- Lesson 3: Don’t generate an inverted management structure
- Lesson 4: Learn when, what, and how to review
- Lesson 5: Practice good software engineering
- Lesson 5b: Don’t Write Your Main Program in C
- Lesson 6: Distribute Data and Information Freely
- Lesson 7: Strive to ensure that the software takes full advantage of the hardware, even at the beginning of a project
- Lesson 8: Treat neither science nor software as a democracy
- Lesson 9: Avoid single points of failure
- Lesson 10: Find some way to reward people working on the project

I wrote these down in 2002; what have we learned in the last 16 years? Some of these are obvious, but were nevertheless ignored by SDSS.

Lessons I Learnt from the SDSS

- Lesson 1: You need a Project Manager
- Lesson 2: Don’t join under-funded projects
- Lesson 3: Don’t generate an inverted management structure
- Lesson 4: Learn when, what, and how to review
- Lesson 5: Practice good software engineering
- Lesson 5b: Don’t Write Your Main Program in C
- Lesson 6: Distribute Data and Information Freely
- Lesson 7: Strive to ensure that the software takes full advantage of the hardware, even at the beginning of a project
- Lesson 8: Treat neither science nor software as a democracy
- Lesson 9: Avoid single points of failure
- Lesson 10: Find some way to reward people working on the project

I wrote these down in 2002; what have we learned in the last 16 years?
Some of these are obvious, but were nevertheless ignored by SDSS.
Many are being ignored by XXX and YYY too.

Disclaimer

The advice in this talk comes from a PCA analysis of my involvement in

- SDSS
- PanSTARRS
- ACT
- HSC
- LSST
- Euclid
- PFS
- WFIRST

Disclaimer

The advice in this talk comes from a PCA analysis of my involvement in

- SDSS
- PanSTARRS
- ACT
- HSC
- LSST
- Euclid
- PFS (Ask me about my job at `http://web.astro.princeton.edu/jobs`)
- WFIRST

Any resemblance to actual projects, living or dead, or actual events is largely coincidental.

No animals were harmed in the writing of this talk.

First the good news

We’ve learned a lot about software engineering:

First the good news

Some of us have learned a lot about software engineering:

- C++
- `python`

First the good news

Some of us have learned a lot about software engineering:

- C++
- `python`
- `git`

First the good news

Some of us have learned a lot about software engineering:

- C++
- `python`
- `git`
- `jUnit`

First the good news

Some of us have learned a lot about software engineering:

- C++
- `python`
- `git`
- `jUnit`
- Jenkins, travis, ...

First the good news

Some of us have learned a lot about software engineering:

- C++
- `python`
- `git`
- `jUnit`
- Jenkins, travis, ...
- JIRA / `github` issues

First the good news

Some of us have learned a lot about software engineering:

- C++
- `python`
- `git`
- `jUnit`
- Jenkins, travis, ...
- JIRA / `github` issues
- docker, conda, ...

First the good news

Some of us have learned a lot about software engineering:

- C
- TCL
- cvs
-
-
- GNATS
-

Actually SDSS didn’t do too badly.

Douglas Adams (The Hitchhiker’s Guide to the Galaxy. 1978)

Orbiting this at a distance of roughly ninety-two million miles is an utterly insignificant little blue green planet whose ape-descended life forms are so amazingly primitive that they still think fortran is a pretty neat idea.

First the goodish news

Some of us have learned a lot about software engineering.

First the goodish news

Some of us have learned a lot about tools for software engineering. What have we learned about software design and practice?

First the goodish news

Some of us have learned a lot about tools for software engineering. What have we learned about software design and practice?
Quite a lot:

- testing
- interfaces
- reusing packages
- community standards
- documentation tools
- code review

First the goodish news

Some of us have learned a lot about tools for software engineering. What have we learned about software design and practice?
Quite a lot:

- testing
- interfaces
- reusing packages
- community standards
- documentation tools
- code review

Unfortunately our new-found wisdom sometimes makes the job of managing software groups harder

First the goodish news

Some of us have learned a lot about tools for software engineering. What have we learned about software design and practice?
Quite a lot:

- testing
- interfaces
- reusing packages
- community standards
- documentation tools
- code review

Unfortunately our new-found wisdom sometimes makes the job of managing software groups harder; some people love the journey more than the destination

First the goodish news

Some of us have learned a lot about tools for software engineering. What have we learned about software design and practice?
Quite a lot:

- testing
- interfaces
- reusing packages
- community standards
- documentation tools
- code review

Unfortunately our new-found wisdom sometimes makes the job of managing software groups harder; some people love the journey more than the destination, others are Luddites who don’t care about technical debt.

Project Managers

Robert Lupton "Lesson 1: You need a Project Manager"

You need a strong and impartial project manager. SDSS is a collaboration of a large number of institutions and we have never managed to take technical decisions unimpeded by politics.

Project Managers

Robert Lupton "Lesson 1: You need a Project Manager"

You need a strong and impartial project manager. SDSS is a collaboration of a large number of institutions and we have never managed to take technical decisions unimpeded by politics.

H. H. Munro (Reginald at the Theatre. 1904)

"When I was younger, boys of your age used to be nice and innocent.
"Now we are only nice. One must specialise these days"

I no longer innocently believe that all we need to do is to hire a project manager, as they mostly come from quite another world.

Project Managers

Robert Lupton "Lesson 1: You need a Project Manager"

You need a strong and impartial project manager. SDSS is a collaboration of a large number of institutions and we have never managed to take technical decisions unimpeded by politics.

H. H. Munro (Reginald at the Theatre. 1904)

"When I was younger, boys of your age used to be nice and innocent.
"Now we are only nice. One must specialise these days"

I no longer innocently believe that all we need to do is to hire a project manager, as they mostly come from quite another world. We *do* need them, of course, but we need to choose them carefully.

The problem[s] with project managers

Many project managers come from hardware backgrounds.

The problem[s] with project managers

Many project managers come from hardware backgrounds. They are comfortable writing contracts with requirements for stiffness and mass and knowing that the sub-contractor will deliver (and that the milestones mean something).

The problem[s] with project managers

Many project managers come from hardware backgrounds. They are comfortable writing contracts with requirements for stiffness and mass and knowing that the sub-contractor will deliver (and that the milestones mean something).

Strategies that I have witnessed for managing Data Management (DM):

The problem[s] with project managers

Many project managers come from hardware backgrounds. They are comfortable writing contracts with requirements for stiffness and mass and knowing that the sub-contractor will deliver (and that the milestones mean something).

Strategies that I have witnessed for managing Data Management (DM):

- Ignore the problem

The problem[s] with project managers

Many project managers come from hardware backgrounds. They are comfortable writing contracts with requirements for stiffness and mass and knowing that the sub-contractor will deliver (and that the milestones mean something).

Strategies that I have witnessed for managing Data Management (DM):

- Ignore the problem
- Make the scientists responsible for DM

The problem[s] with project managers

Many project managers come from hardware backgrounds. They are comfortable writing contracts with requirements for stiffness and mass and knowing that the sub-contractor will deliver (and that the milestones mean something).

Strategies that I have witnessed for managing Data Management (DM):

- Ignore the problem
- Make the scientists responsible for DM
- Treat DM as a subcontract; believe what you are told

The problem[s] with project managers

Many project managers come from hardware backgrounds. They are comfortable writing contracts with requirements for stiffness and mass and knowing that the sub-contractor will deliver (and that the milestones mean something).

Strategies that I have witnessed for managing Data Management (DM):

- Ignore the problem
- Make the scientists responsible for DM
- Treat DM as a subcontract; believe what you are told
- Appoint a DM Project Manager; believe what you are told

The problem[s] with project managers

Many project managers come from hardware backgrounds. They are comfortable writing contracts with requirements for stiffness and mass and knowing that the sub-contractor will deliver (and that the milestones mean something).

Strategies that I have witnessed for managing Data Management (DM):

- Ignore the problem
- Make the scientists responsible for DM
- Treat DM as a subcontract; believe what you are told
- Appoint a DM Project Manager; believe what you are told

So we have a recursive problem. How do we manage the DM Project Manager?

The problem[s] with project managers

Many project managers come from hardware backgrounds. They are comfortable writing contracts with requirements for stiffness and mass and knowing that the sub-contractor will deliver (and that the milestones mean something).

Strategies that I have witnessed for managing Data Management (DM):

- Ignore the problem
- Make the scientists responsible for DM
- Treat DM as a subcontract; believe what you are told
- Appoint a DM Project Manager; believe what you are told

So we have a recursive problem. How do we manage the DM Project Manager? Or, better, how should we find managers who don't need managing?

People

Tools don’t write programs, people do.

People

Tools don’t write programs, people do. Some of our problems are engineering (‘How should we handle multiprocessing?’), some are algorithmic (‘Please write me a weak-lensing-quality deblender’), and some are computational (‘I’ll give you 10ms per object to fit a galaxy model’), but most are about working together, making the best use of our varied skills.

People

Tools don’t write programs, people do. Some of our problems are engineering (‘How should we handle multiprocessing?’), some are algorithmic (‘Please write me a weak-lensing-quality deblender’), and some are computational (‘I’ll give you 10ms per object to fit a galaxy model’), but most are about working together, making the best use of our varied skills.

Robert Lupton "Lesson 8: Treat neither science nor software as a democracy"

Neither Science nor Software can be run as a democracy. Not all participants are equal, and it’s folly to pretend that they are. This is not to say that the most senior (or smartest) individual should simply lay down the law.

People

Tools don’t write programs, people do. Some of our problems are engineering (‘How should we handle multiprocessing?’), some are algorithmic (‘Please write me a weak-lensing-quality deblender’), and some are computational (‘I’ll give you 10ms per object to fit a galaxy model’), but most are about working together, making the best use of our varied skills.

Robert Lupton "Lesson 8: Treat neither science nor software as a democracy"

Neither Science nor Software can be run as a democracy. Not all participants are equal, and it’s folly to pretend that they are. This is not to say that the most senior (or smartest) individual should simply lay down the law.

A piece of good news:

Janel Garvin, Dr. Dobbs Journal (2015-10-01)

So, all told, developers are not the lonely, antisocial nerds that they are portrayed to be, nor are they free-wheeling socialites.

Roberts’ Paradox

Roberts' Paradox

Unfortunately I'm naming it not for me, but for Eric Roberts at Stanford who in 2000 wrote a report for the US National Academy with the blessing of the ACM. The paradox is that:

Roberts' Paradox

Unfortunately I'm naming it not for me, but for Eric Roberts at Stanford who in 2000 wrote a report for the US National Academy with the blessing of the ACM. The paradox is that:

- There are unemployed software engineers

Roberts' Paradox

Unfortunately I'm naming it not for me, but for Eric Roberts at Stanford who in 2000 wrote a report for the US National Academy with the blessing of the ACM. The paradox is that:

- There are unemployed software engineers
- There is a shortage of software engineers

Roberts' Paradox

Unfortunately I'm naming it not for me, but for Eric Roberts at Stanford who in 2000 wrote a report for the US National Academy with the blessing of the ACM. The paradox is that:

- There are unemployed software engineers
- There is a shortage of software engineers

The resolution is that the shortage is of the best engineers, not the median:

> *If the best software developer can do the work of 10, 20, or even 100 run-of-the-mill employees, a single-person company that attracts such a superstar can compete effectively against a much larger enterprise*
> *[. . .]*
> *In some cases, software developers who fall at the low end of the productivity curve may be essentially nonproductive or even counterproductive*

Single point failures

Robert Lupton "Lesson 9: Avoid single points of failure"

OK, so this is totally obvious, but there are subtler aspects:

- If one person is allowed to become essential it implies that it’s proved impossible to find someone else who could fill their rôle
  In consequence, if they are on the critical path, and problems arise, it’s hard to add resources to solve the problem.
- If someone with an essential job isn’t very good, then an essential component of your system isn’t going to work very well.

Single point failures

Robert Lupton "Lesson 9: Avoid single points of failure"

OK, so this is totally obvious, but there are subtler aspects:

- If one person is allowed to become essential it implies that it’s proved impossible to find someone else who could fill their rôle
  In consequence, if they are on the critical path, and problems arise, it’s hard to add resources to solve the problem.
- If someone with an essential job isn’t very good, then an essential component of your system isn’t going to work very well.

My only update would be:
Hire as many people as you can who have with the ability to become single points of failure; then try to manage the project so that they don’t.

Single point failures

Robert Lupton "Lesson 9: Avoid single points of failure"

OK, so this is totally obvious, but there are subtler aspects:

- If one person is allowed to become essential it implies that it's proved impossible to find someone else who could fill their rôle
  In consequence, if they are on the critical path, and problems arise, it's hard to add resources to solve the problem.
- If someone with an essential job isn't very good, then an essential component of your system isn't going to work very well.

My only update would be:
Hire as many people as you can who have with the ability to become single points of failure; then try to manage the project so that they don't.

Robert's Corollary

Roberts' paradox implies that we'll almost always have single points of failure

The Emperor’s New Clothes

I’ve spent most of my "career", and written and re-written hundreds of thousands of lines of code, processing imaging data.

The Emperor’s New Clothes

I’ve spent most of my "career", and written and re-written hundreds of thousands of lines of code, processing imaging data.
One of the many things I worry about when giving talks like this is the thought that actually it’s an easy problem, and that my concerns are really just a way of making myself feel important.

The Emperor’s New Clothes

I’ve spent most of my "career", and written and re-written hundreds of thousands of lines of code, processing imaging data.
One of the many things I worry about when giving talks like this is the thought that actually it’s an easy problem, and that my concerns are really just a way of making myself feel important.

Maybe the real question is, "What should we expect from our university faculty?" Which of:

The Emperor’s New Clothes

I’ve spent most of my "career", and written and re-written hundreds of thousands of lines of code, processing imaging data.
One of the many things I worry about when giving talks like this is the thought that actually it’s an easy problem, and that my concerns are really just a way of making myself feel important.

Maybe the real question is, "What should we expect from our university faculty?" Which of:

- being able to do SPH calculations in your head and knowing all about resonances in magnetised disks

The Emperor’s New Clothes

I’ve spent most of my "career", and written and re-written hundreds of thousands of lines of code, processing imaging data.
One of the many things I worry about when giving talks like this is the thought that actually it’s an easy problem, and that my concerns are really just a way of making myself feel important.

Maybe the real question is, "What should we expect from our university faculty?" Which of:

- being able to do SPH calculations in your head and knowing all about resonances in magnetised disks

and

- dreaming about CRTP and inventing cunning new deblending algorithms that enable us to characterise high-z Luminous Red Galaxies

is of intrinsically higher worth?

The Emperor’s New Clothes

I’ve spent most of my "career", and written and re-written hundreds of thousands of lines of code, processing imaging data.
One of the many things I worry about when giving talks like this is the thought that actually it’s an easy problem, and that my concerns are really just a way of making myself feel important.

Maybe the real question is, "What should we expect from our university faculty?" Which of:

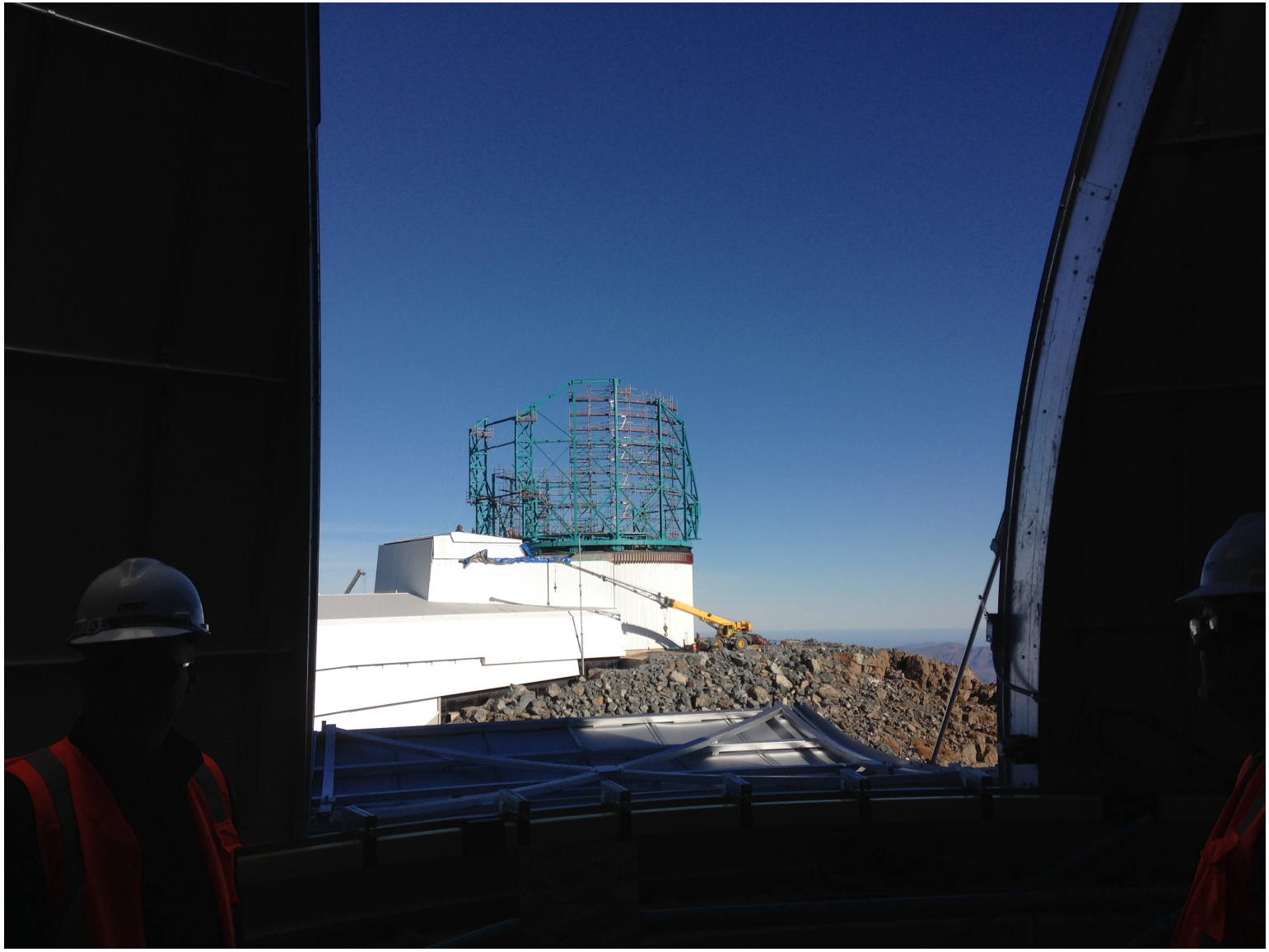
- being able to do SPH calculations in your head and knowing all about resonances in magnetised disks

and

- dreaming about CRTP and inventing cunning new deblending algorithms that enable us to characterise high-z Luminous Red Galaxies

is of intrinsically higher worth? And which is a more valuable skill to teach our students?

LSST Status this Week



LSST Status this Week

Conclusions

Conclusions

- We build big complicated instruments

Conclusions

- We build big complicated instruments
- Our instruments generate big complicated data

Conclusions

- We build big complicated instruments
- Our instruments generate big complicated data
- Big data requires good algorithms and good code

Conclusions

- We build big complicated instruments
- Our instruments generate big complicated data
- Big data requires good algorithms and good code
  - Good algorithms are difficult

Conclusions

- We build big complicated instruments
- Our instruments generate big complicated data
- Big data requires good algorithms and good code
  - Good algorithms are difficult
  - Good code is difficult

Conclusions

- We build big complicated instruments
- Our instruments generate big complicated data
- Big data requires good algorithms and good code
  - Good algorithms are difficult
  - Good code is difficult
- You have to think

Conclusions

- We build big complicated instruments
- Our instruments generate big complicated data
- Big data requires good algorithms and good code
  - Good algorithms are difficult
  - Good code is difficult
- You have to think
  - thinking is difficult

Conclusions

- We build big complicated instruments
- Our instruments generate big complicated data
- Big data requires good algorithms and good code
  - Good algorithms are difficult
  - Good code is difficult
- You have to think
  - thinking is difficult

I asked

Scientific Software: Art, Engineering, or Science?

Conclusions

- We build big complicated instruments
- Our instruments generate big complicated data
- Big data requires good algorithms and good code
  - Good algorithms are difficult
  - Good code is difficult
- You have to think
  - thinking is difficult

I asked

Scientific Software: Art, Engineering, or Science?

and now we know the answer:

Scientific Software: Art, Engineering, and Science.

Conclusions

- We build big complicated instruments
- Our instruments generate big complicated data
- Big data requires good algorithms and good code
  - Good algorithms are difficult
  - Good code is difficult
- You have to think
  - thinking is difficult

I asked

Scientific Software: Art, Engineering, or Science?

and now we know the answer:

Scientific Software: Art, Engineering, and Science. And Fun.

LSST Status this Week

A Condor over Cerro Pachón

LSST Status this Week

A Condor over Cerro Pachón