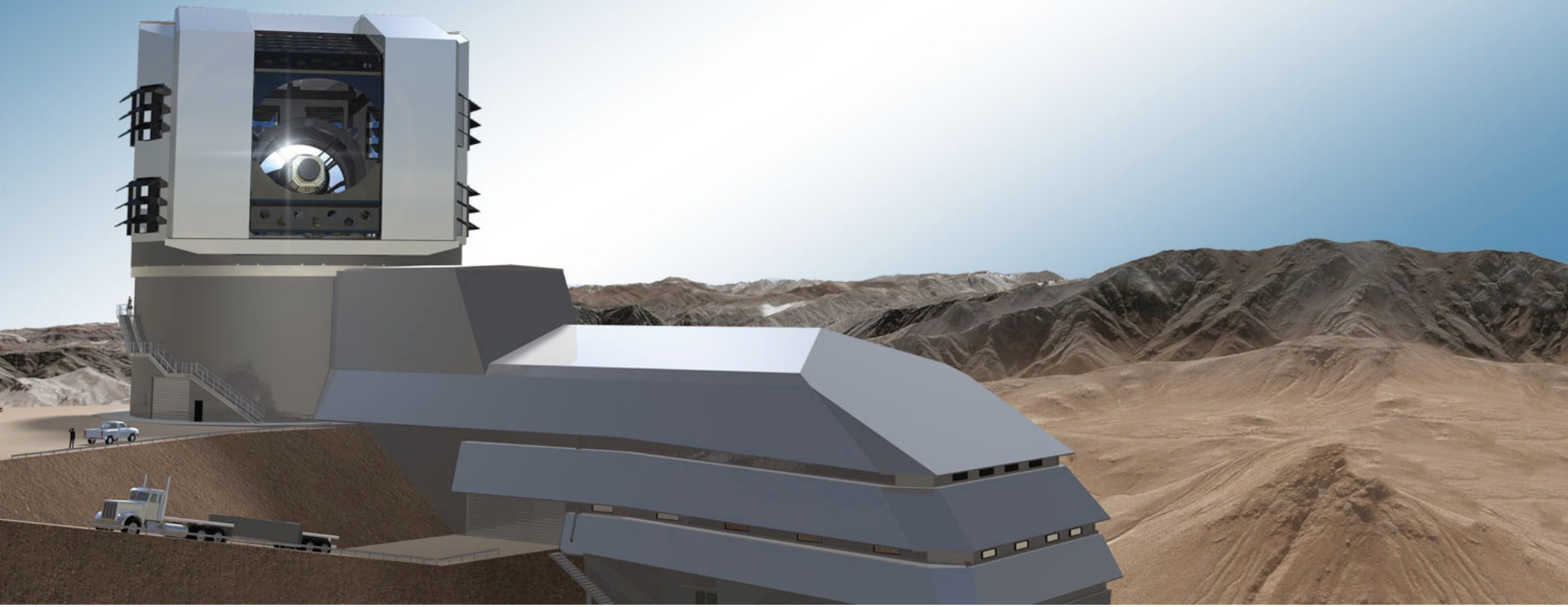
Photometric Redshifts for LSST

Jeffrey Newman, U. Pittsburgh / PITT-PACC

Deputy Spokesperson, LSST Dark Energy Science Collaboration



Before we start

- Please download *data_trim.fits.gz* and *Photo-z lessons.ipynb* from our Confluence site to a working directory, if you haven't done so already

Outline

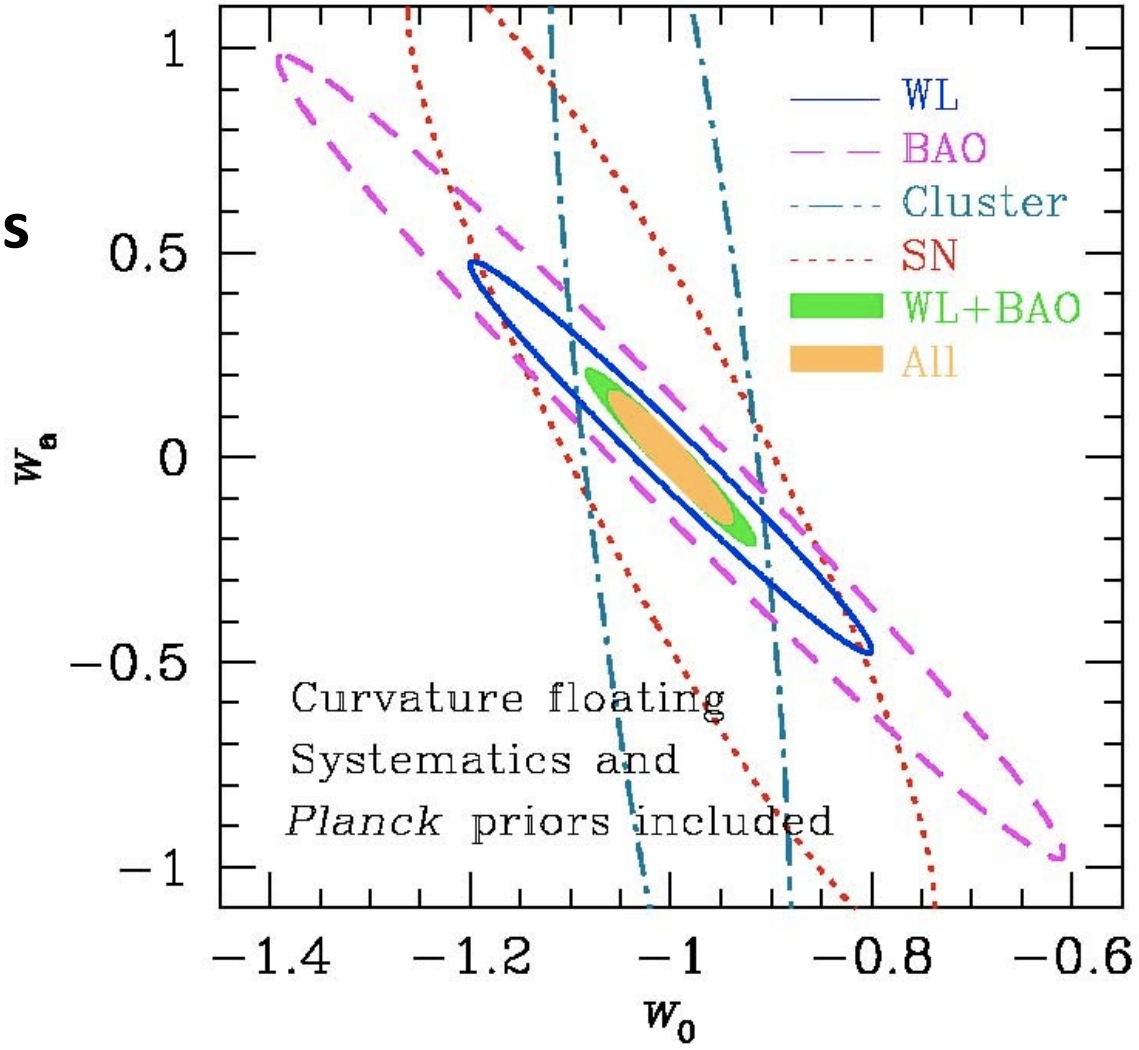
- Overview of photometric redshifts
  - Template methods
  - Training-based methods
- Some open issues
  - Spectroscopic incompleteness
  - Robust training
  - p(z) coverage
  - Combining results from multiple codes
  - p(z,α) storage
  - Optimizing spectroscopic samples
  - Defining ideal LSST algorithm
- Some examples of issues with current codes

LSST constrains dark energy in many ways… all will rely on redshift information

- LSST will constrain dark energy via 4 major probes:
  - Weak gravitational lensing
  - Baryon Acoustic Oscillations
  - Type Ia supernovae
  - Cluster counts

  (Plus strong lensing, etc.)

- For all of these, as well as much galaxy and AGN science, we want to measure some observable as a function of redshift



LSST Science Book

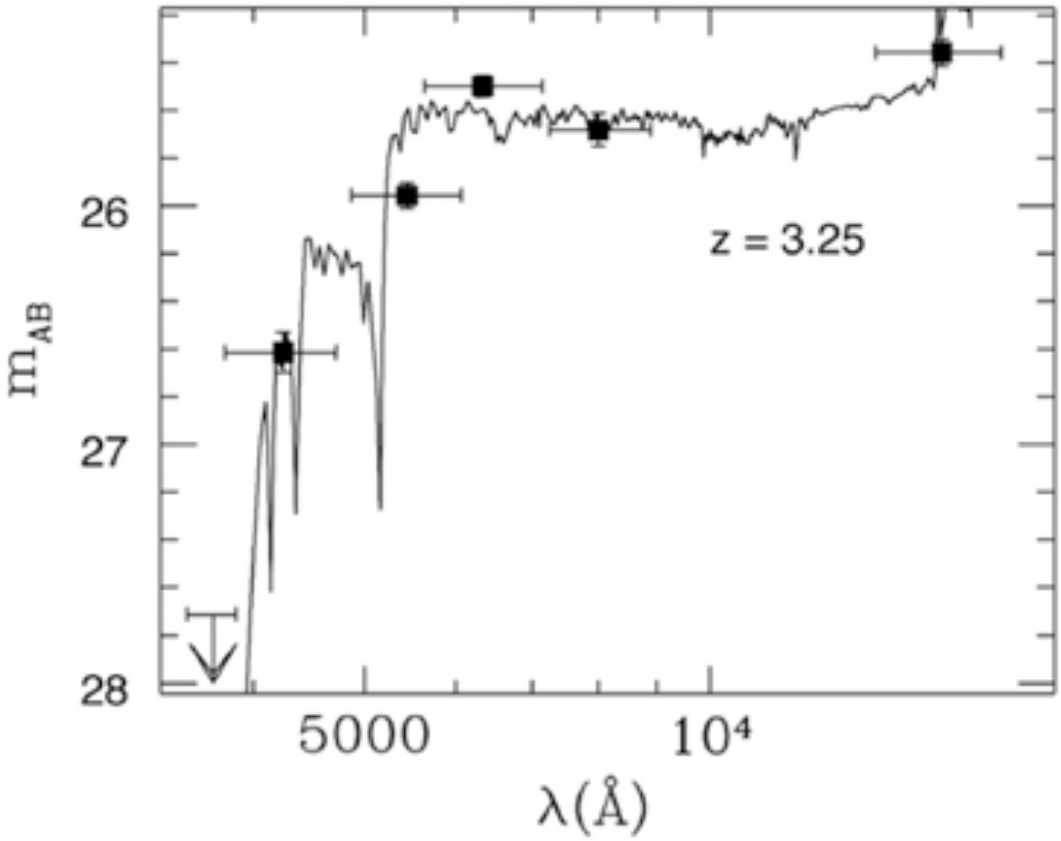
Spectroscopy provides ideal redshift measurements – but is infeasible for large samples

- Redshift ('z') measurements allow us to determine how far back in Universe's history we are looking for an object
- Study galaxy evolution, cosmology, etc. by measuring properties as a function of redshift
- To determine: measure spectrum of light from object with spectrograph; compare observed wavelengths of spectral features to rest frame values to get z
- At LSST "gold sample" (i<25.3) depths, ~100 hours on a 10m telescope to determine a redshift (75% of time) spectroscopically
- With a next-generation, 5000-fiber spectrograph on a 10m telescope, still >50,000 telescope-years to measure redshifts for LSST "gold" weak lensing sample (4 billion galaxies)!

Credit: ESO

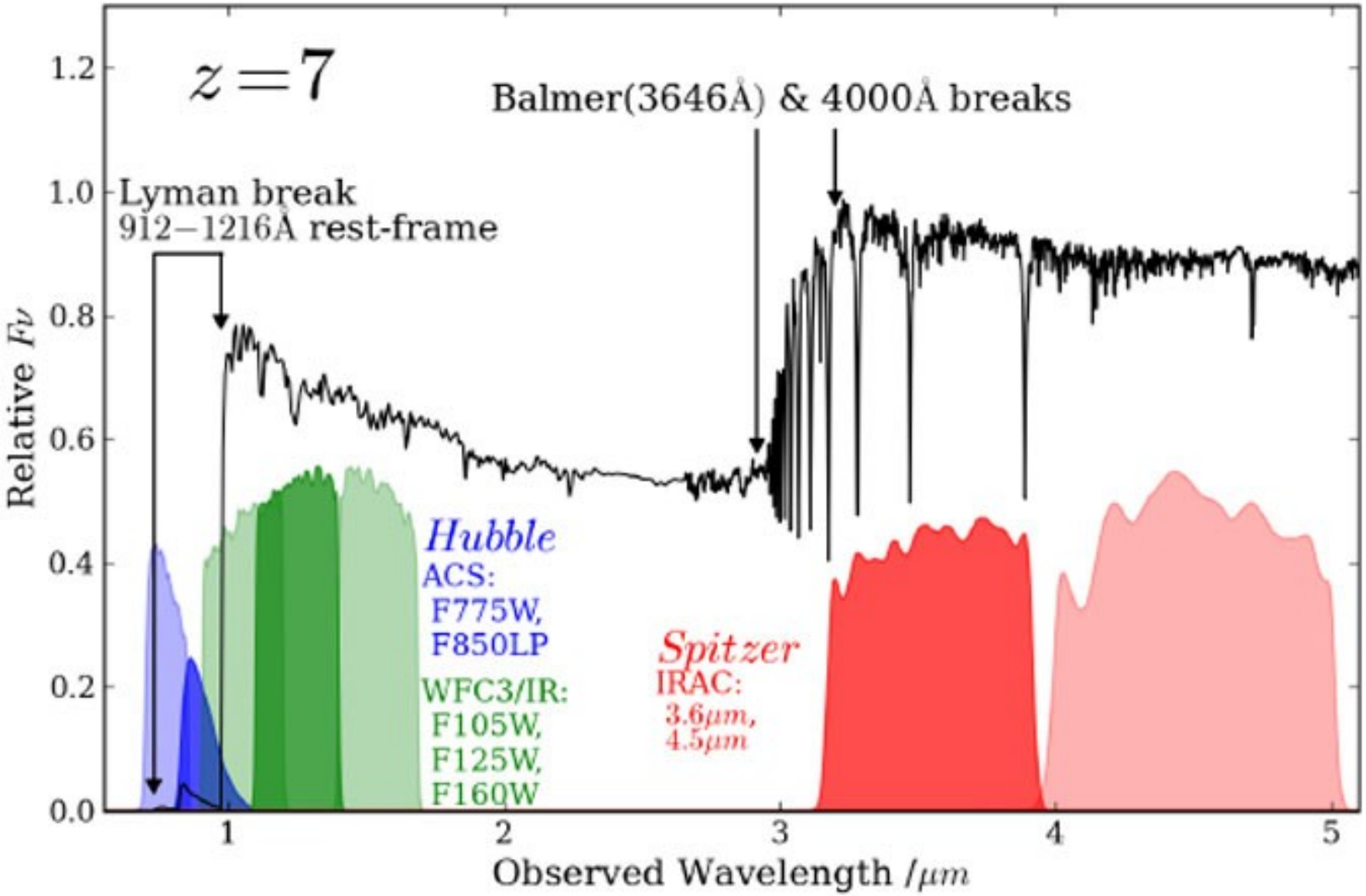
Spectroscopy provides ideal redshift measurements – but is infeasible for large samples

- Alternative: use broad spectral features to determine *z* : a photometric redshift or photo-z
- Advantage: high multiplexing
- Disadvantages: lower precision, calibration uncertainties



Credit: ESO

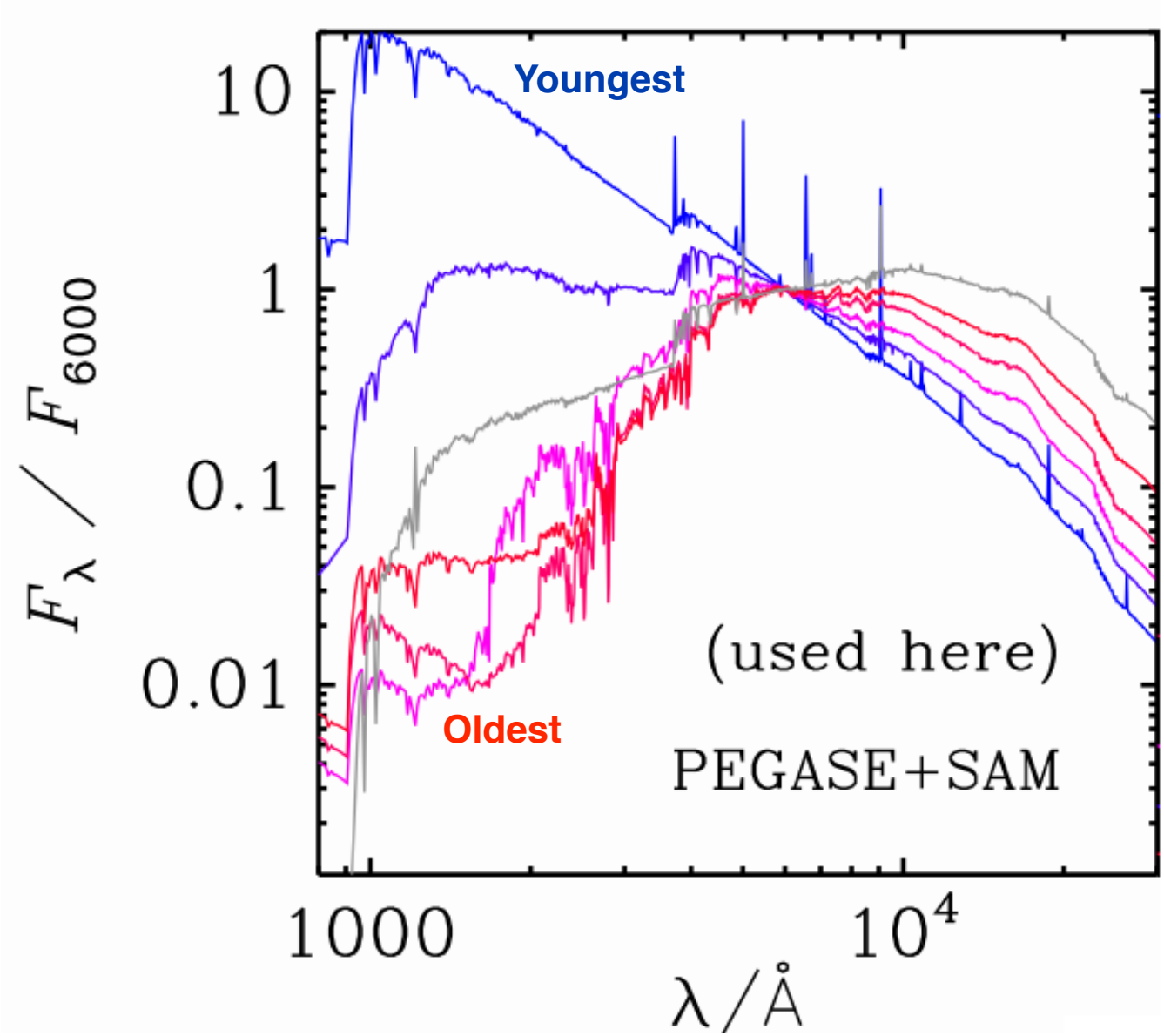
Photometric redshifts rely on the existence of broad spectral features in galaxy spectra...



Dunlop 2012

but those features are stronger in some galaxies than others

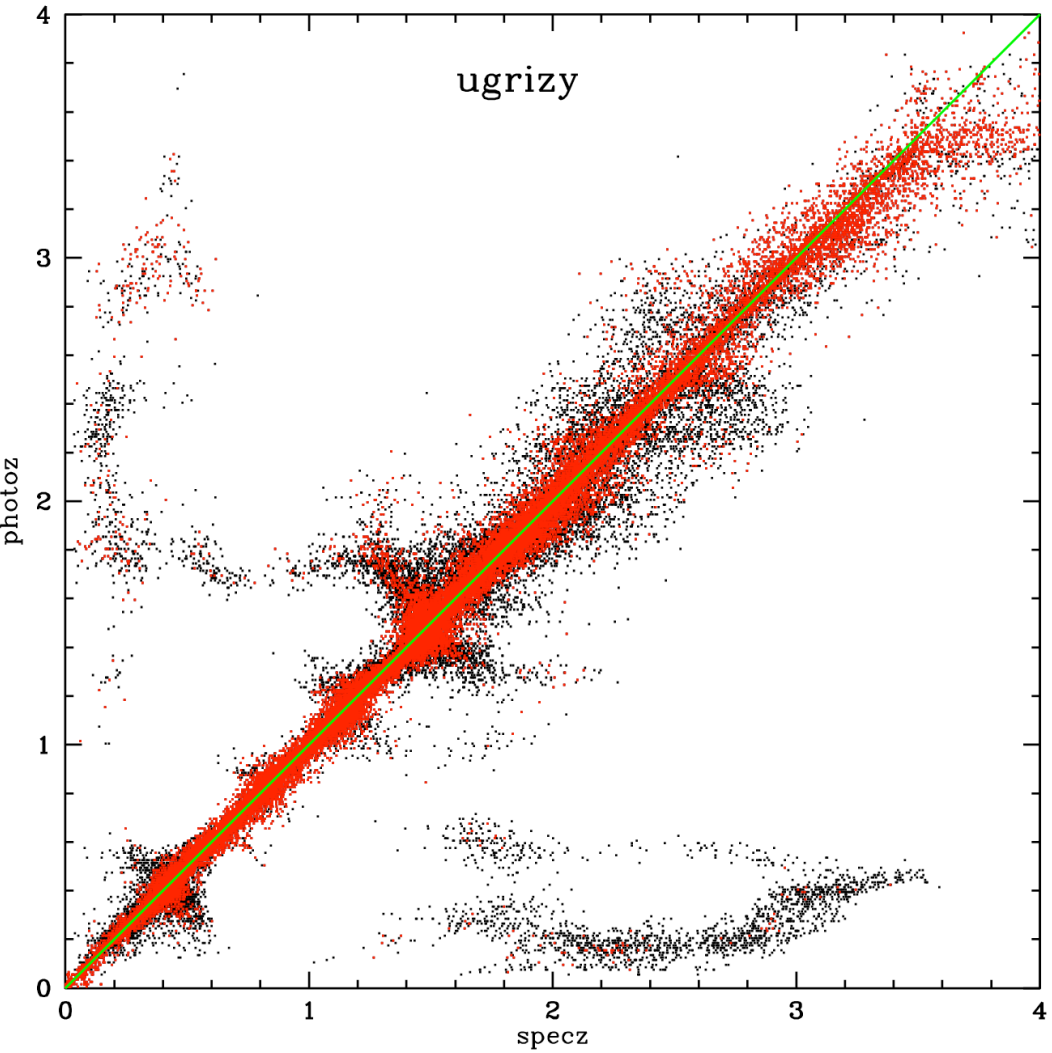
- Galaxies with older stellar populations exhibit stronger 'breaks'
- As a result, photo-z's can be more precise for redder galaxies
- At higher redshifts, blue galaxies with young stellar populations dominate - photo-z problem gets harder



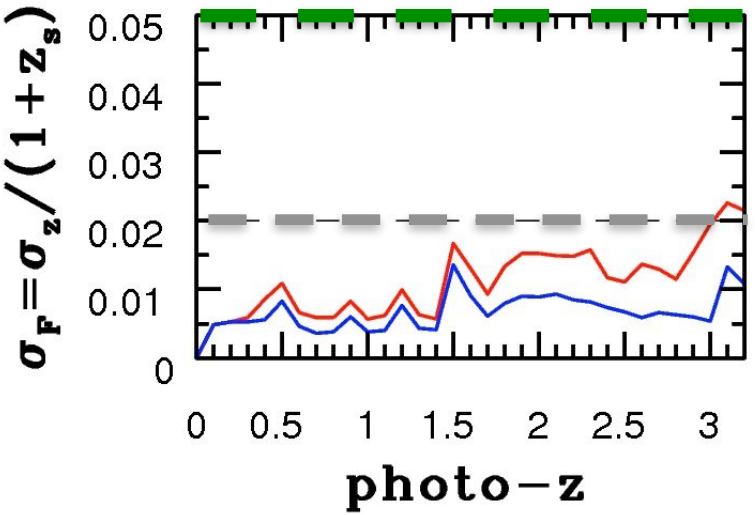
Brammer et al. 2008

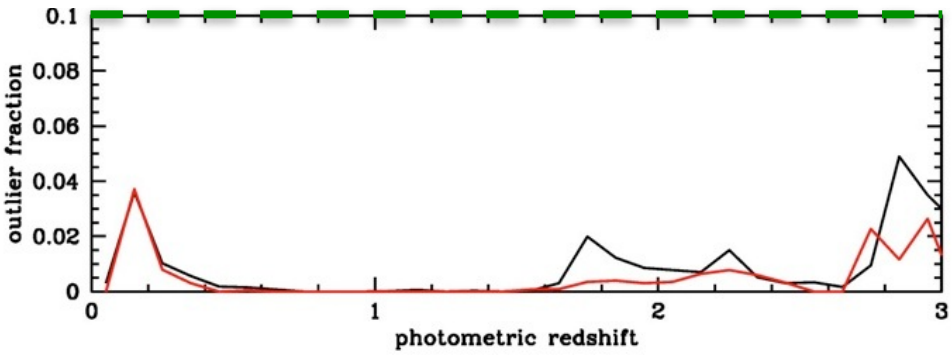
Example: expected photo-z performance for LSST

ugrizy



Green: Requirements on actual performance; **grey**: requirements on performance <u>with perfect template knowledge</u> (as in these sims)



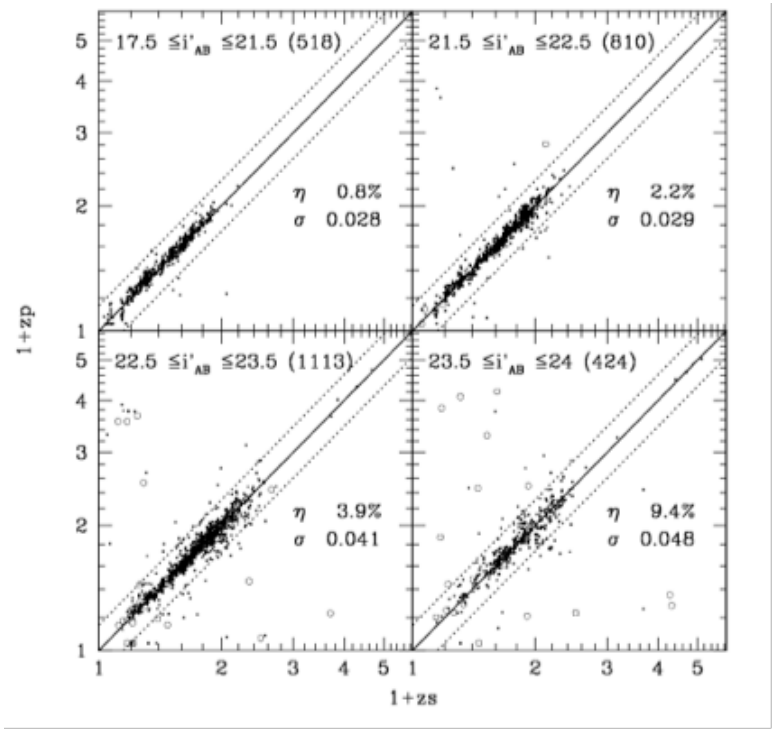


S. Schmidt

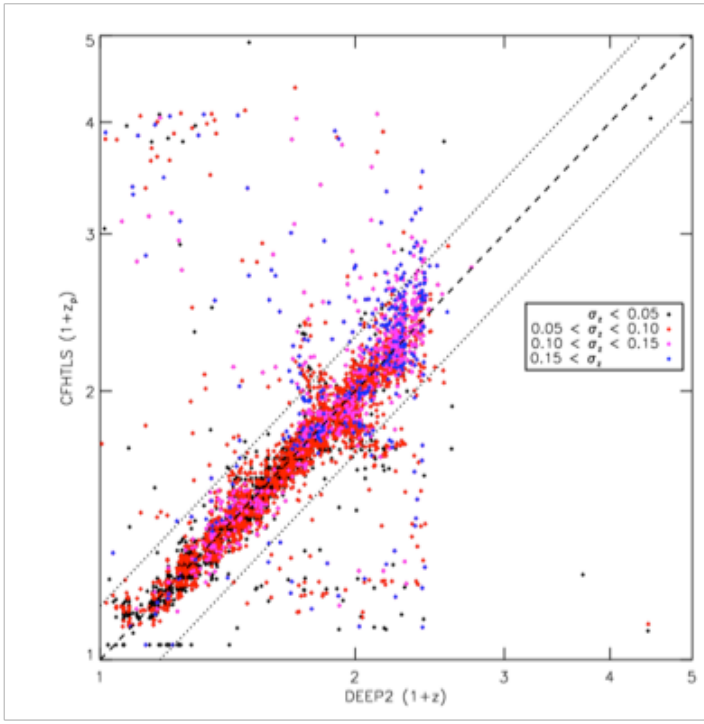
Basic methods: Template fitting photo-z's

- **Use galaxies with known *z* to calibrate set of underlying galaxy spectral energy distributions (SEDs) and photometric band-passes**
  - **Determine posterior probability distribution for *z* | *ugrizy***
  - **Also provides info on galaxy properties from template fit**

Needs spectra of galaxies spanning full range of possible properties to tune templates, establish priors, etc.



Ilbert et al. 2006

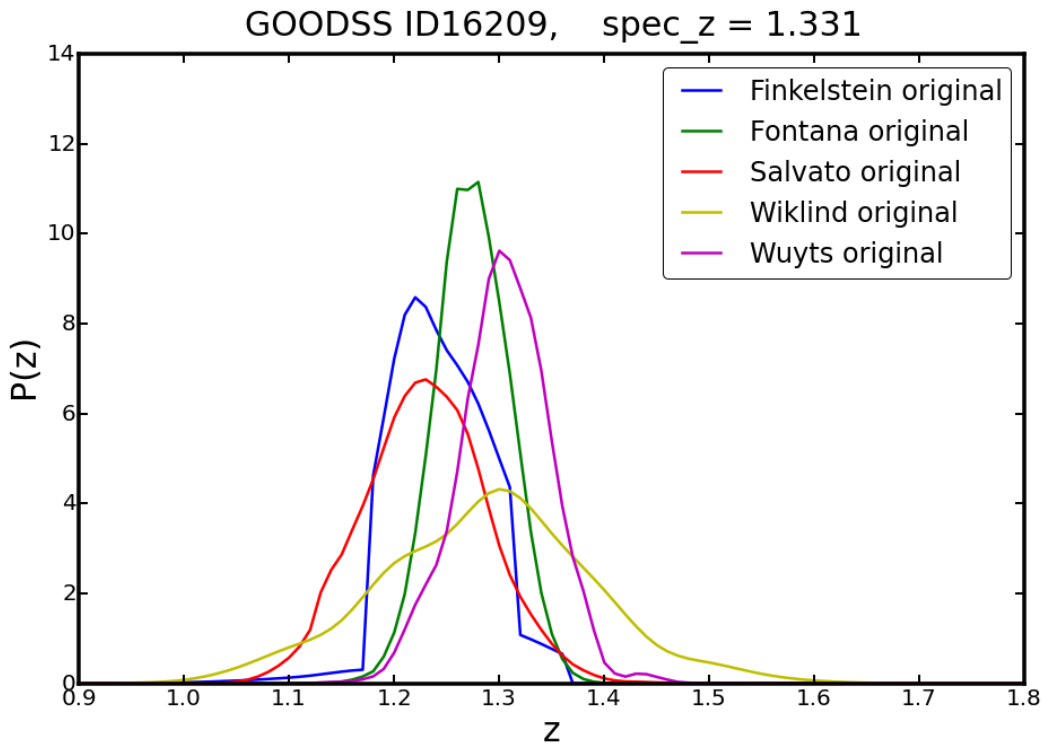


Ilbert photo-z's vs. DEEP2 *z*

Plot by Ben Weiner

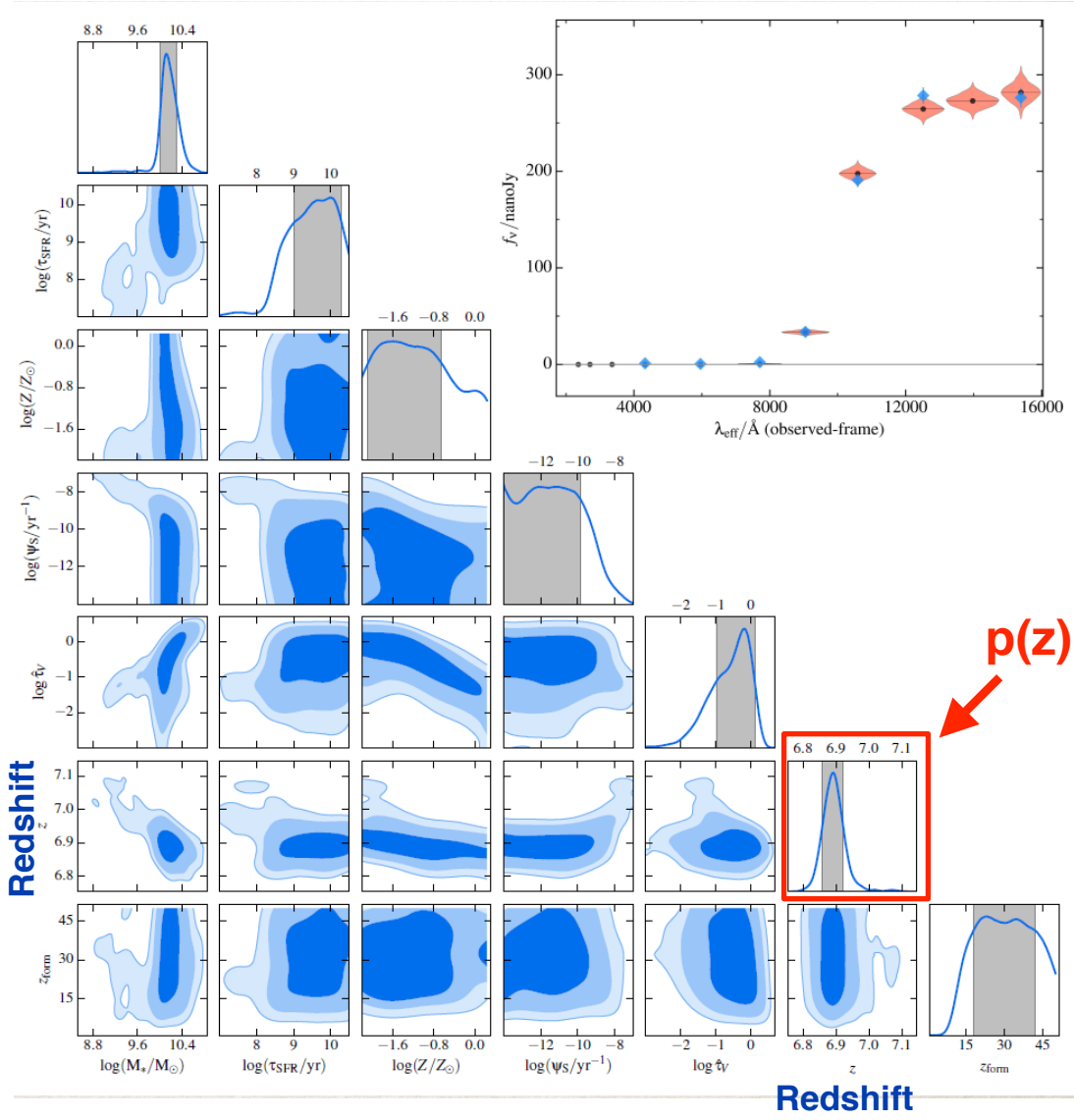
Describing photometric redshift measurements

- Some codes simply output the best-fit *z* with errors
- Generally better to use the posterior probability distribution for *z* | fluxes: *p(z)*
- probability that a < *z* < b = $\int_a^b p(z)\,dz$
  - $\int_0^\infty p(z)\,dz = 1$
- Various definitions for 'point'/ single estimate from *p(z)*:
  - Peak of distribution
  - Expectation value of *z* (possibly only calculating using highest peak)



Describing photometric redshift measurements

- Can also provide info on galaxy properties from template fit
- E.g., template index *T* or galaxy parameters α_i such as stellar mass, star formation rate, etc.): **p(z,α)**

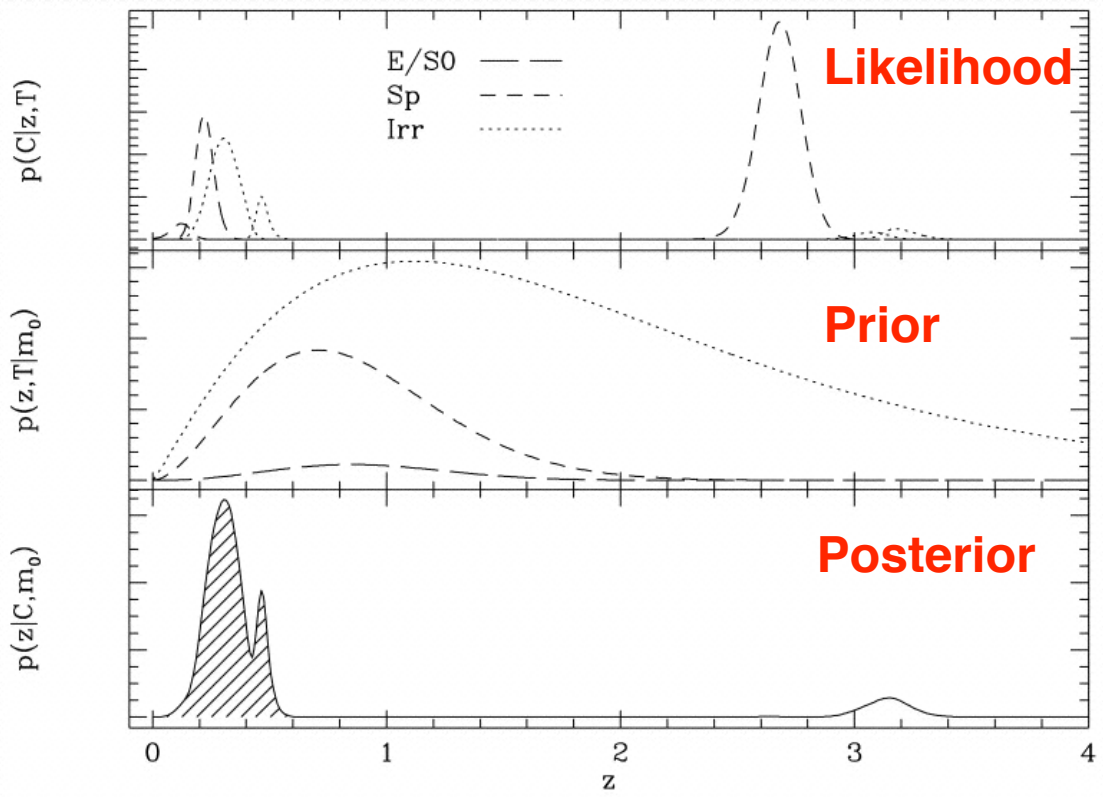


Chevallard & Charlot 2016

Question to discuss in groups: What quantities do *you* want for each LSST galaxy?

Basic methods: Template fitting photo-z's

- Typical algorithms:
  - Determine likelihood of colors (=ratios of fluxes between bands) as a function of *z* and template
    - Often via $\chi^2(z,T)$ or min($\{\chi^2(z|T)\}$); some algorithms use linear combinations of templates



Benitez 2000

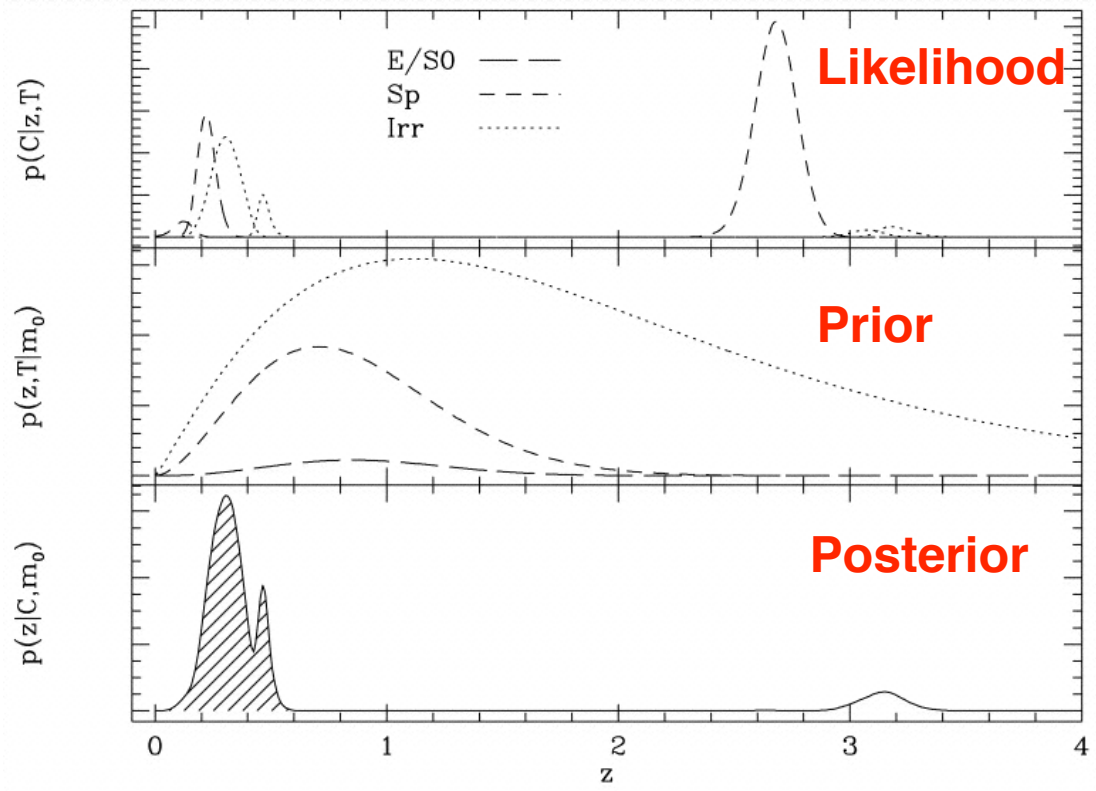
  - Typically utilize prior for redshift or redshift & type based on magnitude (sometimes size/morphology as well)
  - Then multiply to get posterior. . .

Use spectra of galaxies spanning full range of possible properties to tune templates/filter systems, establish priors, etc.

Basic methods: Template fitting photo-z's

- Note that this is standard Bayesian language:
- *p(fluxes | z)* = likelihood
- *p(z)* = prior
- *p(z | fluxes)* = posterior
- By Bayes' theorem,

$$p(z|\text{fluxes}) = \frac{p(\text{fluxes}|z)\, p(z)}{p(\text{fluxes})}$$

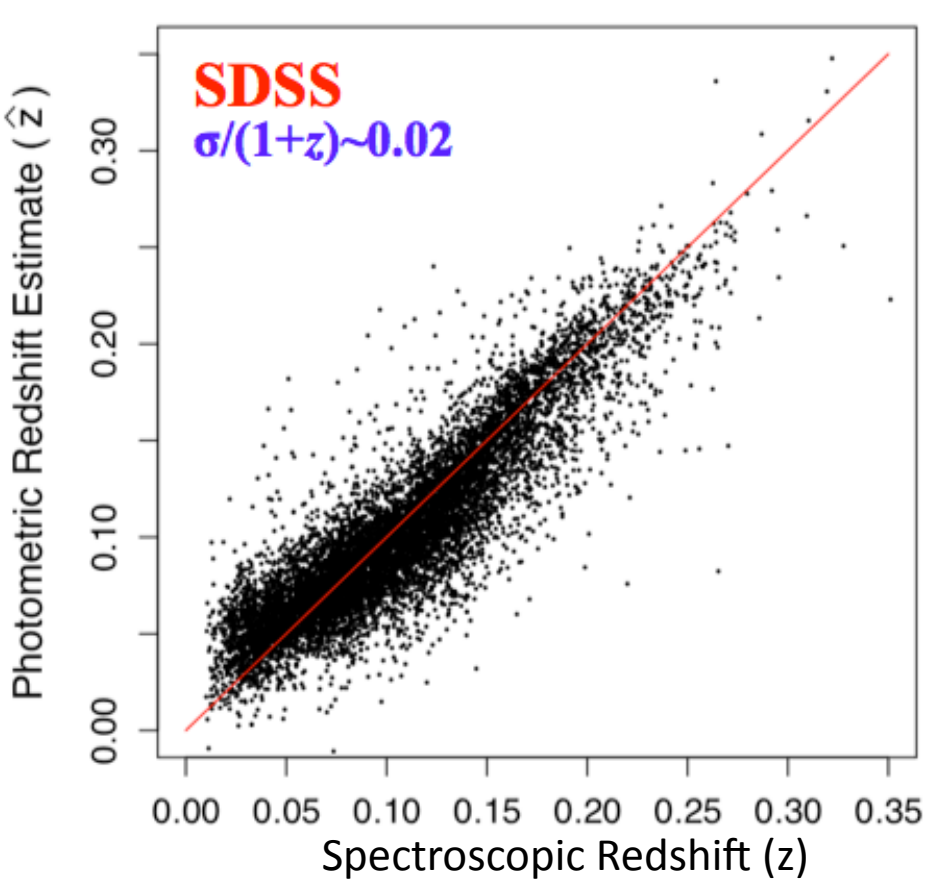


Benitez 2000

- We often just normalize the integral of the posterior to be 1 rather than calculating *p*(fluxes)

Basic methods: Training-based photo-z's

- Use galaxies with known redshift and uniform/well-understood sampling to determine relationship between *z* and colors/fluxes
- Can take advantage of progress in machine learning & stats, but generally extrapolate poorly; Training set MUST span full range of properties & z of galaxies

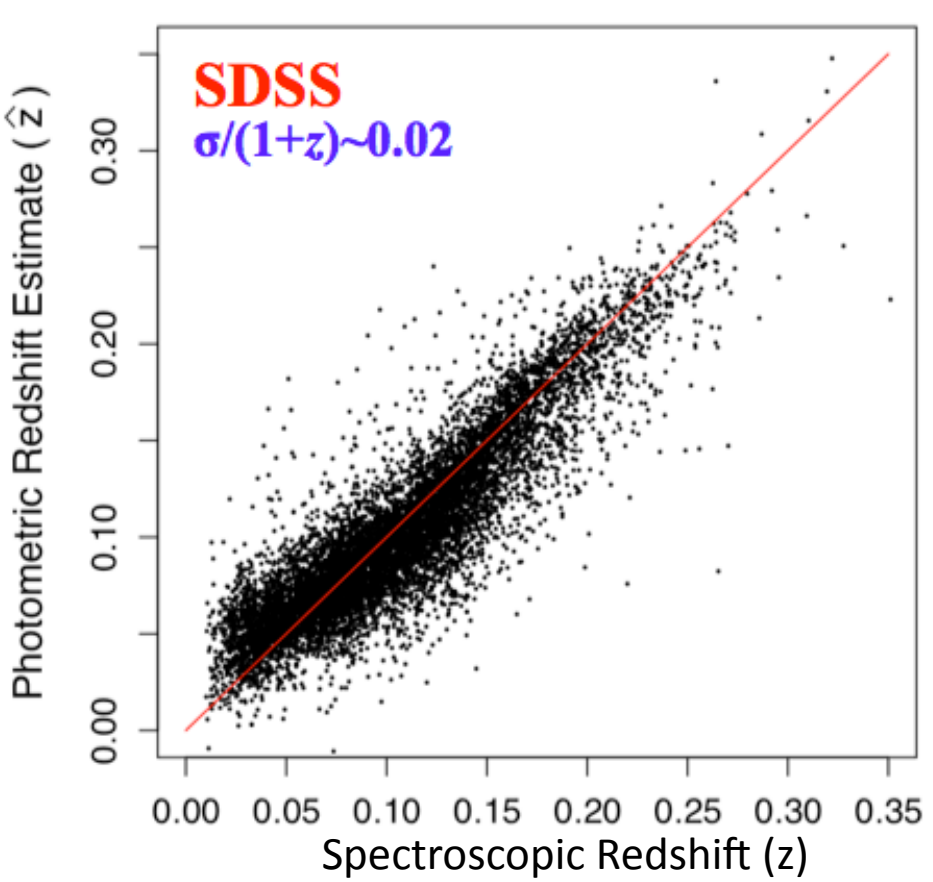


Freeman, JN et al. 2009

Basic methods: Training-based photo-z's

- Many algorithms: e.g.
  - Neural networks
  - Boosted Decision Trees
  - Random Forest regression
  - k-Nearest Neighbor
  - Diffusion map + regression

- For bright, nearby galaxies, training sets are ~complete and both template-based & training-set-based algorithms perform extremely similarly

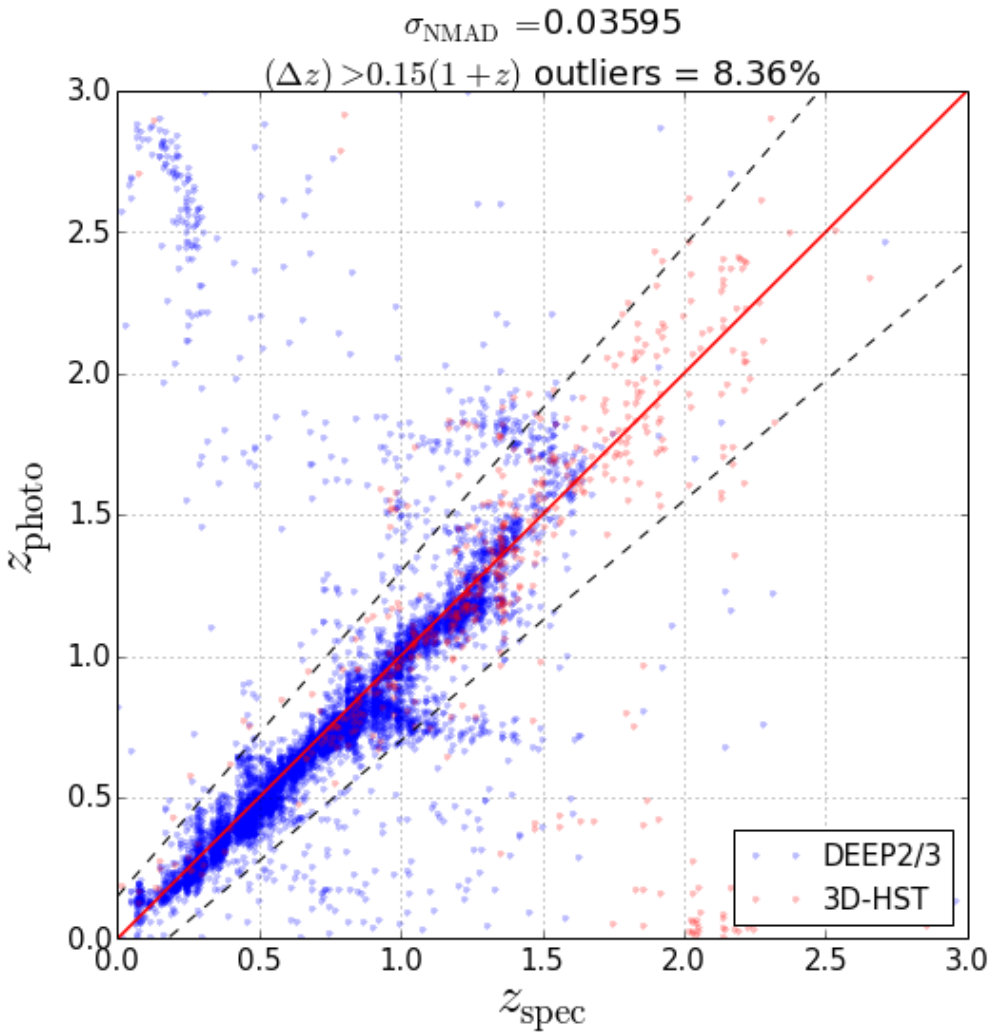


Freeman, JN et al. 2009

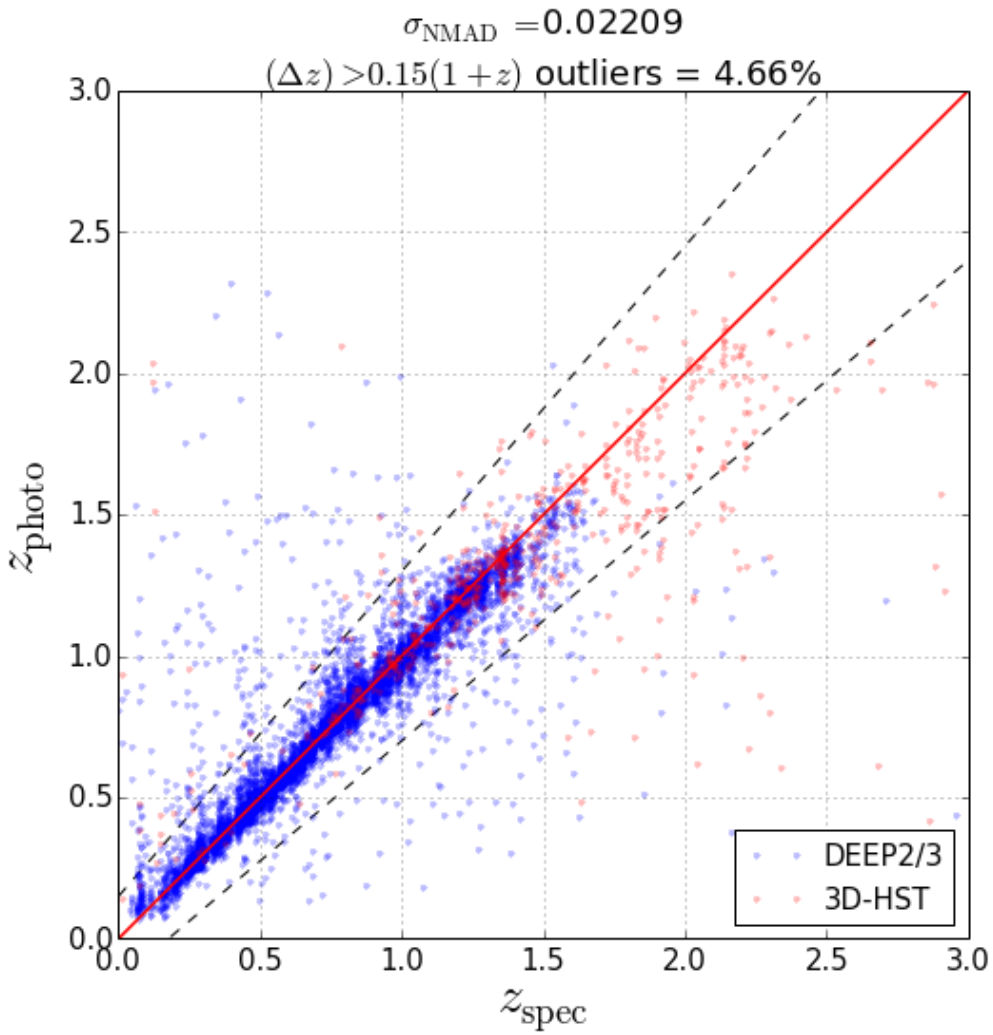
At higher redshifts, the photo-z problem is more difficult

- **Zhou et al. 2016 (in prep.): empirical, LSST-like dataset: CFHT LS *ugriz* + Subaru *y* + DEEP2/DEEP3/3D-HST redshifts**

EAZY (template code, untuned)



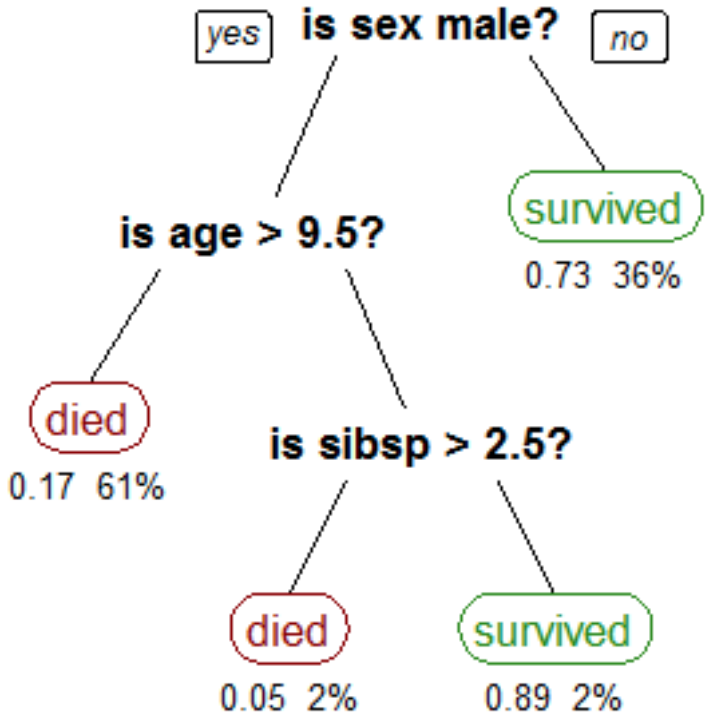
Random Forest Regression



Zhou, JN et al. 2016, in prep.

Let's try determining our own photo-z's!

- We'll apply the random forest regression routine in the scikit-learn Python package to the dataset produced by Rongpu Zhou.
- Random forest is a highly effective machine learning technique based on decision trees
- In random forest, a wide variety of decision trees get trained by a training set of data, randomly bootstrapping amongst that data for optimizing each tree
- A random subset of possible data features are used in each tree
- In the end, results from all trees get combined to give a prediction (in our case, a prediction for *z*)



Walking through the code: importing routines

- Load the Jupyter notebook (e.g. go to a Terminal; type 'jupyter notebook Photo-z\ Lessons.ipynb', or just type 'jupyter notebook' and choose the right file.
- This notebook contains text and python code. To execute a block of code, click in the box of code, hold down shift, and press enter.
- We start by importing python modules and routines

[1]:
```
%matplotlib inline
from __future__ import division #avoids integer division; 1/2 = 0.5, not 0
import numpy as np #numpy routines
from astropy.table import Table   #astropy routine for reading tables
import matplotlib.pyplot as plt   #plotting routines

# Random forest routine from scikit-learn:
from sklearn.ensemble import RandomForestRegressor

# Cross-Validation routines:
from sklearn.cross_validation import KFold
from sklearn.cross_validation import train_test_split
from sklearn.cross_validation import cross_val_predict
```

Walking through the code: reading in the data with Table.read()

```
In [2]: #read in catalog of magnitudes and redshifts
        cat = Table.read('data_trim.fits.gz')

        # How big is the catalog?
        print 'Full catalog: ',len(cat),' objects'

        # What information does it contain for each object?
        print cat.colnames
        print cat
```

```
Full catalog:  8508  objects
['U', 'G', 'R', 'I', 'Z', 'Y', 'UERR', 'GERR', 'RERR', 'IERR', 'ZERR',
 'YERR', 'RADIUS_ARCSEC', 'ZHELIO', 'MAGB', 'UB_0']
```

- **U, G, R, I, Z, and Y are magnitude measurements for each object (=2.5 times the $\log_{10}$ of the flux relative to some reference flux, the "zero point"). Is everyone familiar with the idea of magnitudes?**
- **UERR, GERR, RERR, IERR, ZERR, and YERR are uncertainties in the magnitudes**

Walking through the code: reading in the data with Table.read()

```
In [2]: #read in catalog of magnitudes and redshifts
        cat = Table.read('data_trim.fits.gz')

        # How big is the catalog?
        print 'Full catalog: ',len(cat),' objects'

        # What information does it contain for each object?
        print cat.colnames
        print cat
```

```
Full catalog:  8508  objects
['U', 'G', 'R', 'I', 'Z', 'Y', 'UERR', 'GERR', 'RERR', 'IERR', 'ZERR',
 'YERR', 'RADIUS_ARCSEC', 'ZHELIO', 'MAGB', 'UB_0']
```

- **RADIUS_ARCSEC is a measure of the radius of the object in arc seconds**
- **ZHELIO is the redshift ($z=\lambda_{observed}/\lambda_{restframe}$ -1) in the heliocentric reference frame (why does that make a difference?)**
- **UB_0 is an estimate of the restframe U-B color of the object (=-2.5 times the $\log_{10}$ of the flux ratio between the U and B bands; bigger = redder)**

Walking through the code: a function to make plots and calculate statistics

In [3]:
```
#A function that we will call a lot: makes the zphot/zspec plot and calc
ulates key statistics
def plot_and_stats(zspec,zphot):

    x = np.arange(0,5.4,0.05)
    outlier_upper = x + 0.15*(1+x)
    outlier_lower = x - 0.15*(1+x)

    mask = np.abs((z_phot - z_spec)/(1 + z_spec)) > 0.15
    notmask = ~mask

        #Standard Deviation of the predicted redshifts compared to the d
ata:
    std_result = np.std((z_phot - z_spec)/(1 + z_spec), ddof=1)
    print 'Standard Deviation: %6.4f' % std_result

    #Normalized MAD (Median Absolute Deviation):
    nmad = 1.48 * np.median(np.abs((z_phot - z_spec)/(1 + z_spec)))
    print 'Normalized MAD: %6.4f' % nmad

    #Percentage of delta-z > 0.15(1+z) outliers:
    eta = np.sum(np.abs((z_phot - z_spec)/(1 + z_spec)) > 0.15)/len(z_sp
ec)
    print 'Delta z >0.15(1+z) outliers: %6.3f percent' % (100.*eta)

    #Median offset (normalized by (1+z); i.e., bias:
    bias = np.median(((z_phot - z_spec)/(1 + z_spec)))
    sigbias=std_result/np.sqrt(0.64*len(z_phot))
    print 'Median offset: %6.3f +/- %6.3f' % (bias,sigbias)
```

Walking through the code: a function to make plots and calculate statistics

```
    # make photo-z/spec-z plot
    plt.figure(figsize=(8, 8))

    #add lines to indicate outliers
    plt.plot(x, outlier_upper, 'k--')
    plt.plot(x, outlier_lower, 'k--')
    plt.plot(z_spec[mask], z_phot[mask], 'r.', markersize=6,  alpha=0.5)
    plt.plot(z_spec[notmask], z_phot[notmask], 'b.',  markersize=6, alph
a=0.5)
    plt.plot(x, x, linewidth=1.5, color = 'red')
    plt.title('$\sigma_\mathrm{NMAD} \ = $%6.4f\n'%nmad+'$(\Delta z)>0.1
5(1+z) $ outliers = %6.3f'%(eta*100)+'%', fontsize=18)
    plt.xlim([0.0, 2])
    plt.ylim([0.0, 2])
    plt.xlabel('$z_{\mathrm{spec}}$', fontsize = 27)
    plt.ylabel('$z_{\mathrm{photo}}$', fontsize = 27)
    plt.grid(alpha = 0.8)
    plt.tick_params(labelsize=15)
    plt.show()
```

Walking through the code: defining convenience arrays

```
# create convenience arrays for all magnitudes
u_mag = cat['U']
g_mag = cat['G']
r_mag = cat['R']
i_mag = cat['I']
z_mag = cat['Z']
y_mag = cat['Y']

# and for magnitude errors
u_err = cat['UERR']
g_err = cat['GERR']
r_err = cat['RERR']
i_err = cat['IERR']
z_err = cat['ZERR']
y_err = cat['YERR']

#Redshift array
z = cat['ZHELIO']

#Size array: radius in arcseconds
rad = cat['RADIUS_ARCSEC']

#Restframe color
ub_color = cat['UB_0']

#Photometry perturbed: doubling sizes of all errors
u_magn=u_mag + np.sqrt(3)*u_err*np.random.randn(len(u_mag))
g_magn=g_mag + np.sqrt(3)*g_err*np.random.randn(len(g_mag))
r_magn=r_mag + np.sqrt(3)*r_err*np.random.randn(len(r_mag))
i_magn=i_mag + np.sqrt(3)*i_err*np.random.randn(len(i_mag))
z_magn=z_mag + np.sqrt(3)*z_err*np.random.randn(len(z_mag))
y_magn=y_mag + np.sqrt(3)*y_err*np.random.randn(len(y_mag))
```

Walking through the code: putting data in the right format for scikit_learn regression routines

```
# Now, set up input data arrays for random forest regression

# First: magnitudes only
data_mags = np.column_stack((u_mag,g_mag,r_mag,i_mag,z_mag,y_mag))

# Next: colors only
data_colors = np.column_stack((u_mag-g_mag, g_mag-r_mag, r_mag-i_mag, i_mag-z_mag, z_mag-y_mag))

# Next: colors and one magnitude
data_colmag = np.column_stack((u_mag-g_mag, g_mag-r_mag, r_mag-i_mag, i_mag-z_mag, z_mag-y_mag, i_mag))
perturbed_colmag=np.column_stack((u_magn-g_magn, g_magn-r_magn, r_magn-i_magn, i_magn-z_magn, z_magn-y_magn, i_magn))

# Finally: colors, magnitude, and size
data_colmagsize = np.column_stack((u_mag-g_mag, g_mag-r_mag, r_mag-i_mag, i_mag-z_mag, z_mag-y_mag, i_mag, rad))


data_z = z

# We need to set up an implementation of the scikit-learn RandomForestRegressor in an object called 'regrn'.
regrn = RandomForestRegressor(n_estimators = 50, max_depth = 30, max_features = 'auto')
```

- **Not particularly interesting... scikit-learn routines expect the features to be used for predictions to be in a 2d array, with dimensions (# of objects, # of features)**

Next: plotting colors, magnitude and size as a function of redshift

- Note: to an astronomer, color is the difference between magnitudes in 2 filters (with bluest specified first); e.g. $g - r$ = g-band magnitude - r-band magnitude = - 2.5 $\log_{10}$ (g flux / r flux)
- Color is larger for redder objects
- Some photo-z codes work in flux (better for handling uncertainties, which are closer to Gaussian in flux)

Next: plotting colors, magnitude and size as a function of redshift

```
if 0:
# Plots of colors vs. redshift

    plt.figure(figsize=(8, 5))

    plt.plot(z, u_mag-g_mag, '.', color = 'red', markersize=3, alpha=0.2)
    # Axis limits:
    plt.xlim([0., 1.5])
    plt.ylim([-0.5, 2.5])
    #Axis labels
    plt.ylabel('$u-g$ color', fontsize=20)
    plt.xlabel('Redshift',fontsize=20)
    #Add grid lines
    plt.grid()
```

- Change 'if 0' to 'if 1' . What did we just do?
- Take a look at all of the plots of color etc. vs. redshift. Discuss in your groups: what colors constrain redshift best for objects at z~0.2? 0.6? 1? Why should different colors be important at different redshifts?

Next: plotting colors, magnitude and size as a function of redshift

```
if 0:

    # To better assess the quality of the Random Forest fitting,
    # we split the data into Training (50%) and Test (50%) sets.
    # The code below performs this task on the data_mags and data_z arrays:
    data_train, data_test, z_train, z_test = train_test_split(data_mags, data_z,
                        test_size = 0.50, train_size = 0.50, random_state=7182016)

    #Train the regressor using the training data
    regrn.fit(data_train,z_train)

    #Apply the regressor to predict values for the test data
    z_phot = regrn.predict(data_test)
    z_spec = z_test

    #Make a photo-z/spec-z plot and output summary statistics.
    plot_and_stats(z_spec,z_phot)
```

- Change 'if 0' to 'if 1' .
- This does all the work: divides data into training and test sets, trains the random forest regression, uses it to predict redshifts for the test set, and has plots_and_stats do all the calculations.

Worse and better ways to do this

- **Worse: use the same data for training & test. Try it; discuss in your groups what went wrong.**
- **Better: use k-fold cross-validation. Compare the plot for 5-fold cross-validation to our first plot. What is different?**

In k-fold cross-validation, we split the data into k subsets. We loop over the subsets, training with all but one and testing with the other; in the end, we get the performance of training with a fraction $\frac{k-1}{k}$ of the data, but are able to get test statistics based on the *entire* dataset.

This is easy to do in scikit-learn, but does mean running the training k times on more data than before...

Exploiting the underlying structure of the data can improve machine learning results

- Next, we explore using different features to do the photo-z's:
  - Colors, instead of magnitudes
  - Add a magnitude (emulating effect of a prior)
  - Add size information
  - Before running the new ways, predict what will happen and discuss with your group.
  - How significant are the improvements?

Exploiting the underlying structure of the data can improve machine learning results

- Finally, let's explore doing photo-z's restricted to one particular population of galaxies
  - Red galaxies have strong breaks in their spectra and can yield tighter photo-z's (σ~0.01).
  - Let's select the red population in our catalog. First, we plot the distribution of rest frame colors of galaxies:

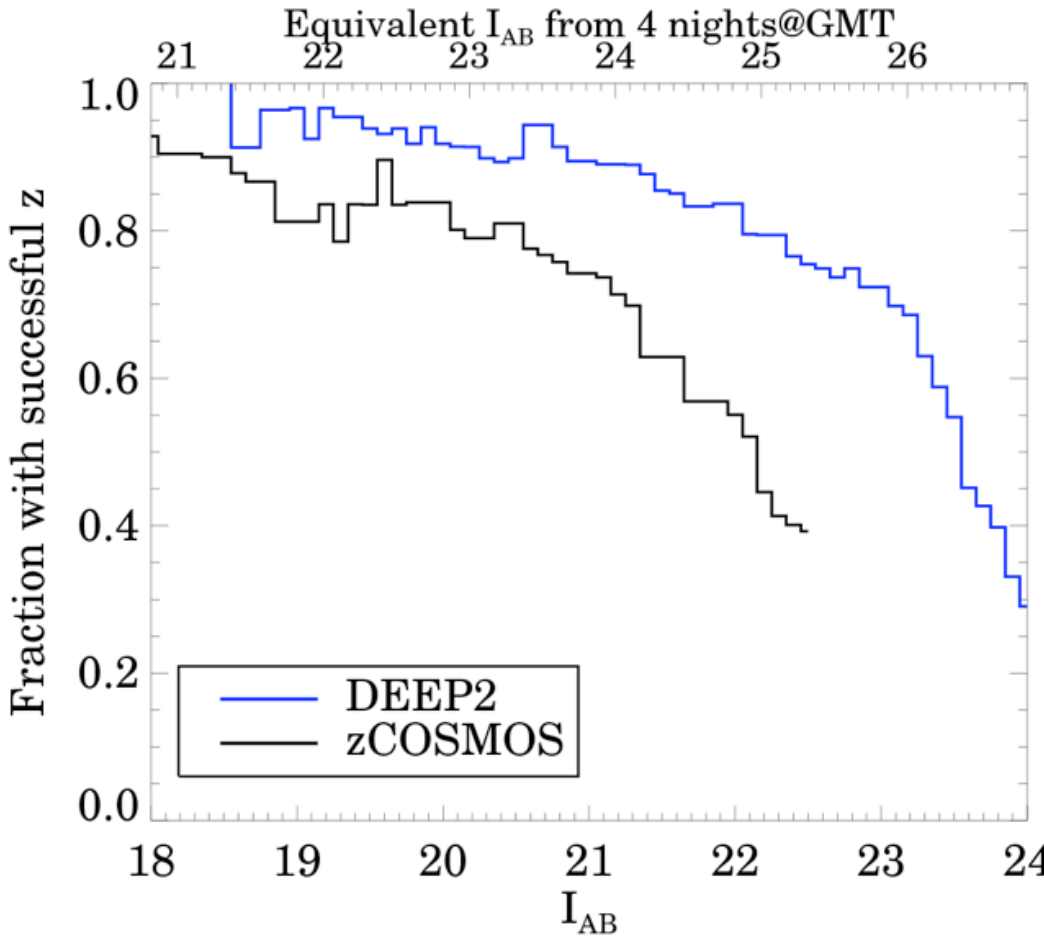
```
if 0:
    plt.figure(figsize=(8, 5))

    inrange=(ub_color > -0.5) & (ub_color < 2)
    plt.hist(ub_color[inrange],bins=100)
    # Axis limits:
    plt.xlim([-0.3, 1.5])
    plt.xlabel('Restframe $U-B$ color', fontsize=20)
    plt.grid()
```

Exploiting the underlying structure of the data can improve machine learning results

- In the next code box, change *redlimit* to a value that effectively divides red galaxies from blue ones.
- Discuss: did you get better photo-z's for this population by isolating it first?

Open issues: dealing with incompleteness in training/ calibration datasets

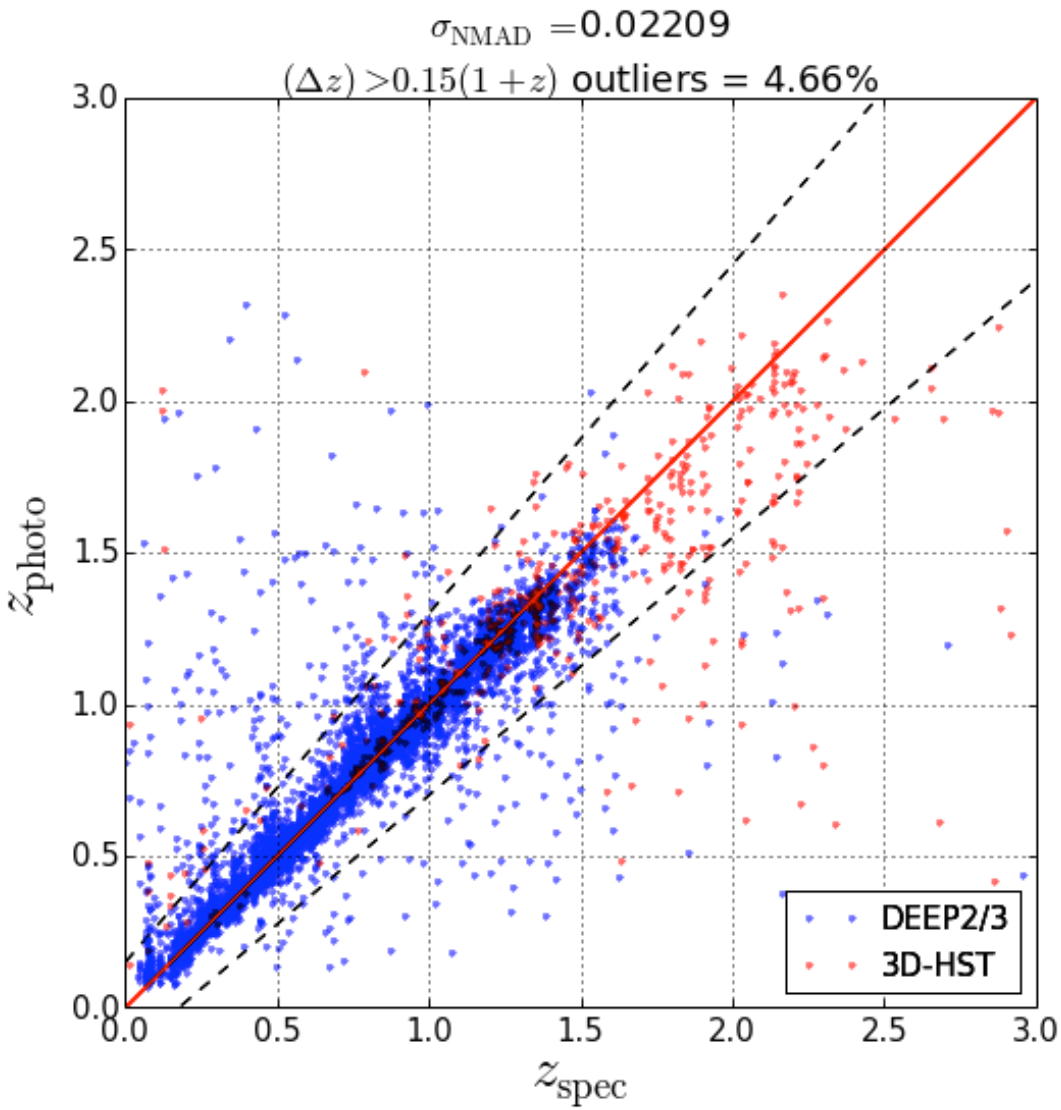
- In current deep spectroscopic surveys, 25-60% of targets fail to yield secure redshifts
- z success rate depends on galaxy properties
- Estimated need 99-99.9% completeness to prevent systematic errors in calibration, unless apply other methods (e.g., cross-correlations)
- Major issue for training-set techniques



Data from DEEP2 (Newman et al. 2013) and zCOSMOS (Lilly et al. 2009)

Open issues: Robust training methods

- 1% incorrect-redshift rate is sufficient to bias photo-z's beyond tolerances
- Depending on survey, up to 5% of 'secure' redshifts are incorrect
- If can train algorithms in a manner robust to outlier/ wrong redshifts, could use the broader set of less-secure spectroscopic redshifts
- ML methods that extrapolate well would also be interesting



Zhou, JN et al. 2016, in prep.

Let's explore what happens when we mis-train

- Let's try three different sorts of mis-training:
  - Training with bright objects & applying to faint (or vice versa)
  - Training with red (in observed color) objects and applying to blue
  - Training with low-z objects and applying to high-z
  - Discuss: Which of these was worst?
- Note: real spec-z samples have strong selection effects in redshift
- They tend to be less successful for redder and fainter galaxies as well
- Hence, our training sets are likely to suffer from all 3 problems.

Open issues: Making posteriors great again

- **CANDELS code comparison: Dahlen et al. 2013**
- **11 code/template combinations were tested using ~600 redshifts in GOODS-S (trained with a separate set of 600 redshifts)**
- **Generally χ^2 minimization, generally with some sort of prior.**
- **Codes with p(z)'s available are marked by ★**

Code		Code ID	Template set	bias_z^a	OLF^b	σ_F^c	σ_O^d	
Rainbow		A	PEGASE[b]	-0.010	0.092	0.167	0.041	
GOODZ		B	CWW[c], Kinney[d]	-0.007	0.036	0.099	0.035	
EAZY	★	C	EAZY[e]+BX418[f]	-0.009	0.051	0.114	0.044	
SPOC		D	BC03[g]	-0.030	0.147	0.197	0.073	
zphot	★	E	PEGASEv2.0[b]	-0.007	0.041	0.104	0.037	
EAZY		C	EAZY[e]	-0.009	0.053	0.121	0.037	
SATMC		F	BC03[g]	-0.008	0.093	0.272	0.064	
HyperZ		G	Maraston05[h]	0.013	0.078	0.189	0.050	
LePhare	★	H	BC03[g]+Polletta07[i]	-0.008	0.048	0.132	0.038	
WikZ	★	I	BC03[g]	-0.023	0.046	0.153	0.049	
EAZY	★	C	EAZY[e]	-0.005	0.039	0.127	0.034	
				-0.008	0.029	0.088	0.031	median(all)
				-0.009	0.031	0.079	0.029	median(5)

Dahlen et al. 2013

Open issues: Making posteriors great again

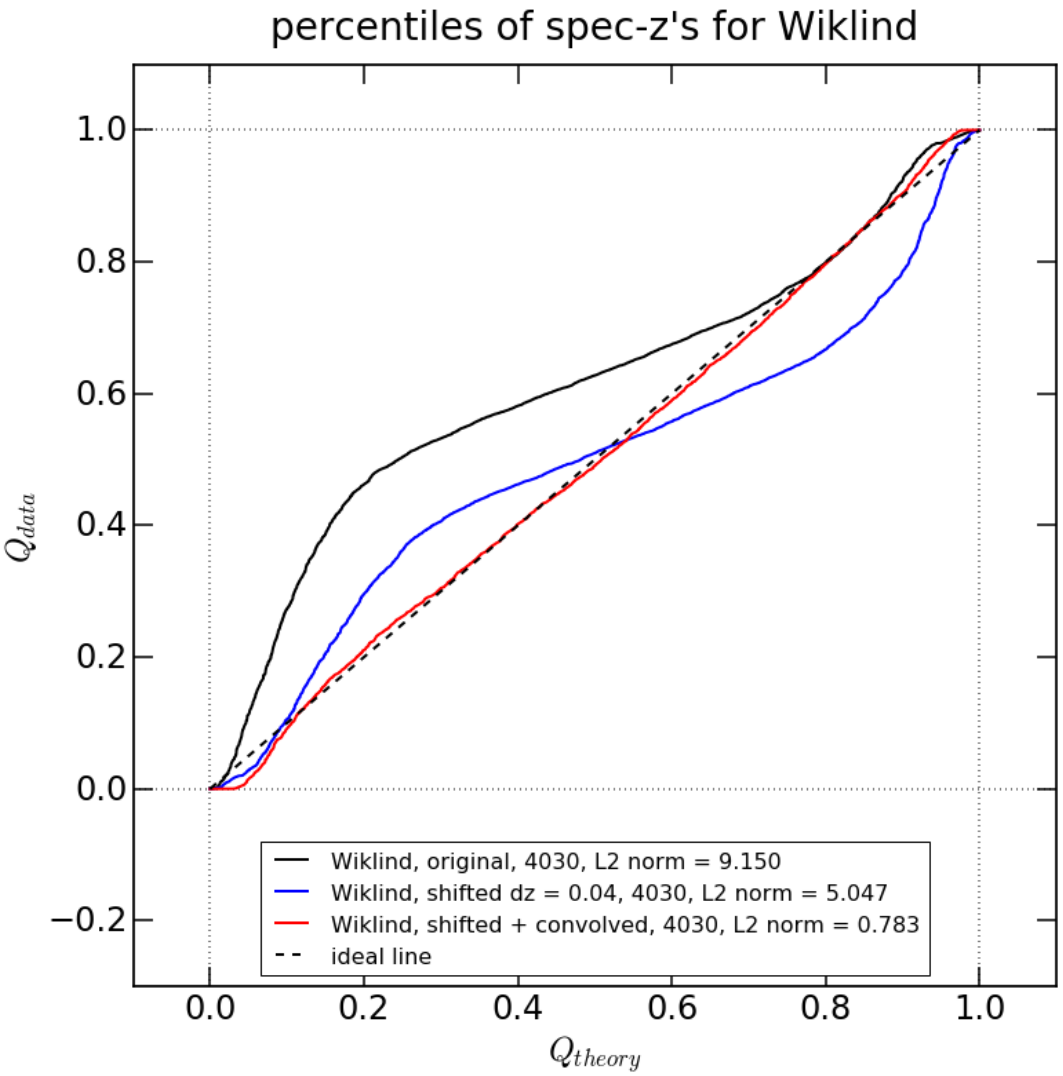
- Many analyses assume that photo-z codes are providing posterior PDFs with proper coverage (and assuming that they can add PDFs to get N(z); talk to Alex Malz if you want to learn about the right way to do that...)
- Dahlen et al. 2013 tested the fraction of spectroscopic redshifts that are in the inner 68% or inner 95% of their PDFs
- Coverage is all over the place; no codes were good at both 68% and 95% points

Code		WFC3 H-selected	
conf. int:		68.3%	95.4%
2A		46.1	
3B		81.6	92.8
4C	★	64.0	88.2
5D		2.5	4.2
6E	★	52.0	84.7
7C		65.0	87.3
8F		15.3	15.6
9G		16.3	44.1
11H	★	35.2	54.0[a]
12I	★	88.7	96.7
13C	★	52.0	72.7

Dahlen et al. 2013

Open issues: Making posteriors great again

- LSST Dark Energy Science Collaboration is setting up controlled tests of the problem
- Meanwhile, kludge in Kodra et al. 2016: modify *p(z)*'s for CANDELS HST survey
  - Shift by constant in *z* direction; convolve with Gaussian kernel; and take to a power (equivalent to rescaling errors in χ^2 calculation)
- Optimize parameters by minimizing total L2 norm of deviation in Q-Q plot from expected line

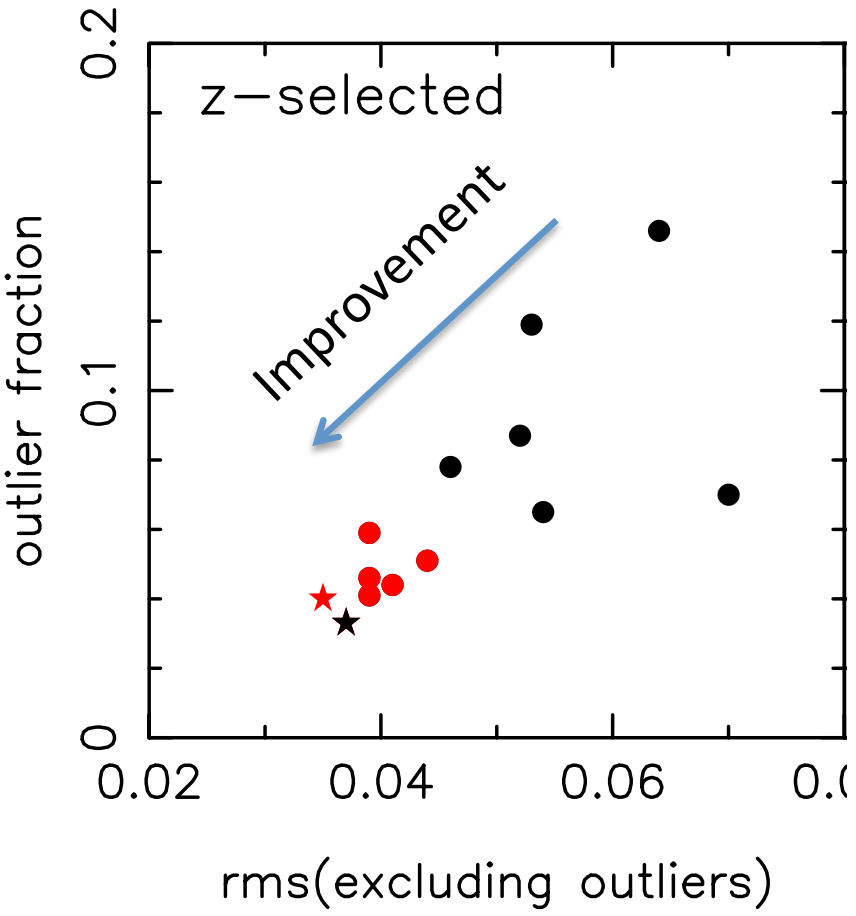


Kodra, JN et al. 2016, in prep.

Open issues: Combining PDF results from multiple codes

- **Dahlen et al. found that medians of point estimates from multiple codes (★'s) have smaller scatter (relative to spec-z) than any individual code**

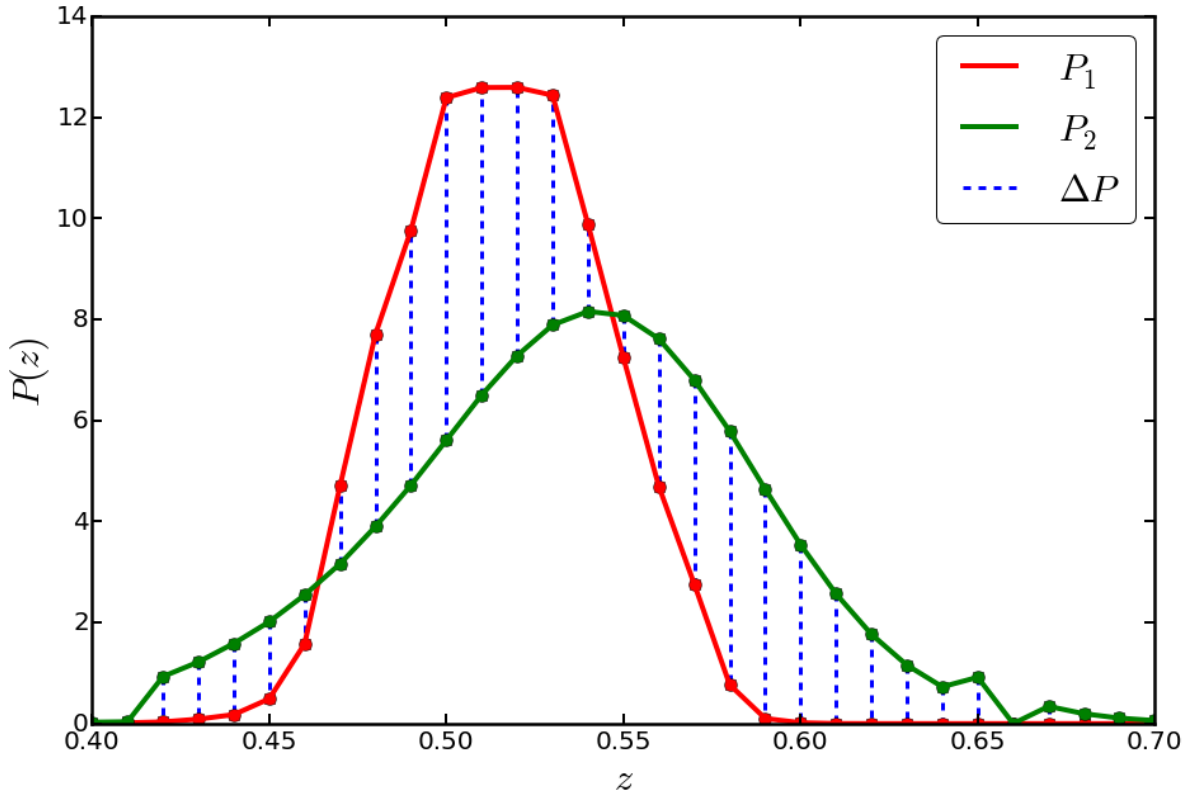
- **All codes are run on the same data! Current codes do not make optimal use of available information...**



Dahlen et al. 2013

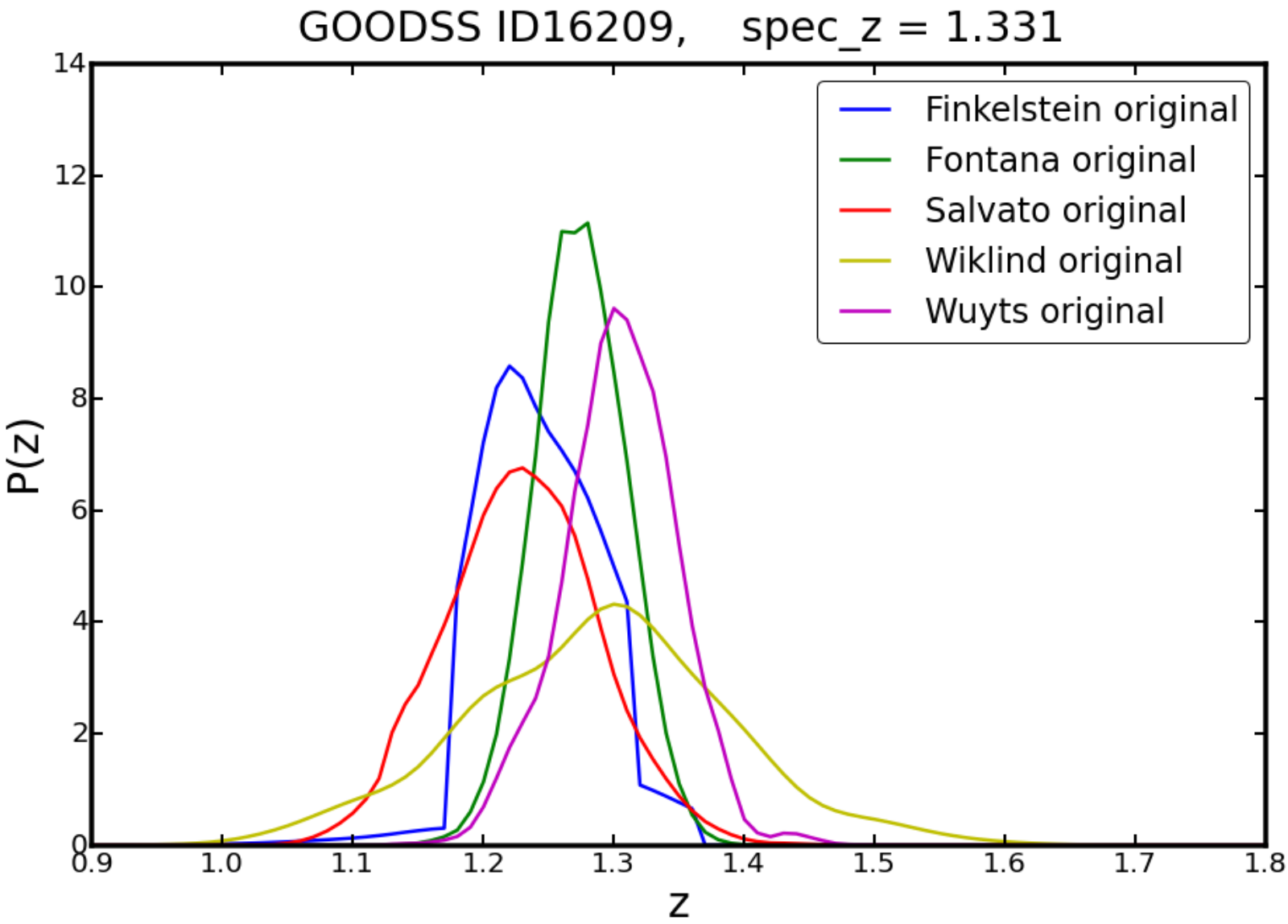
Open issues: Combining PDF results from multiple codes

- Dahlen et al. presented a hierarchical Bayesian combination method (cf. Press & Kochanek, Lang & Hogg, etc.)
- Izbicki & Lee 2016 use weighted combinations of codes
- Kodra et al. (in prep) investigates using PDF that minimizes total Fréchet distance to remaining PDFs: analogous to median



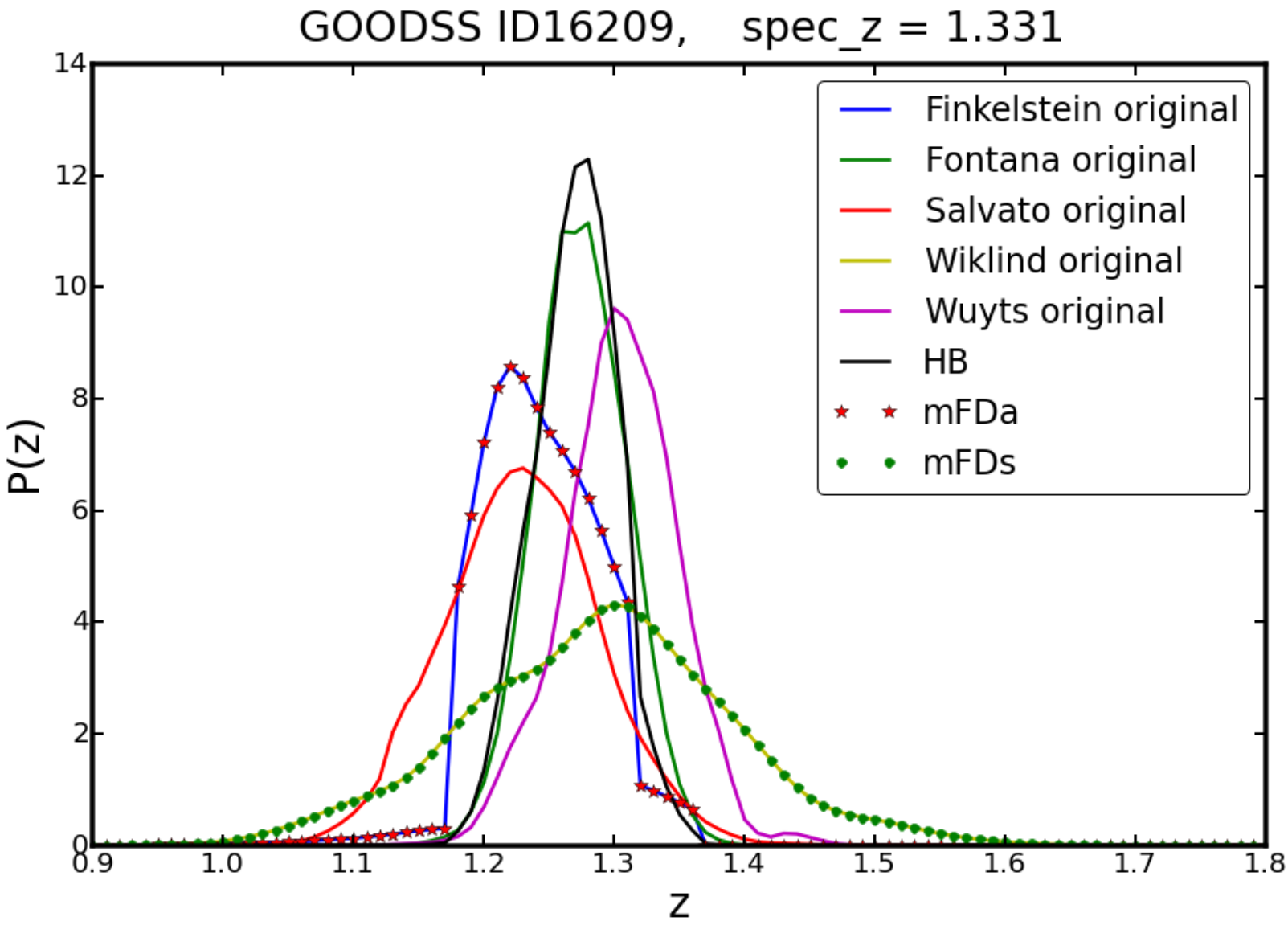
D. Kodra

Open issues: Combining PDF results from multiple codes



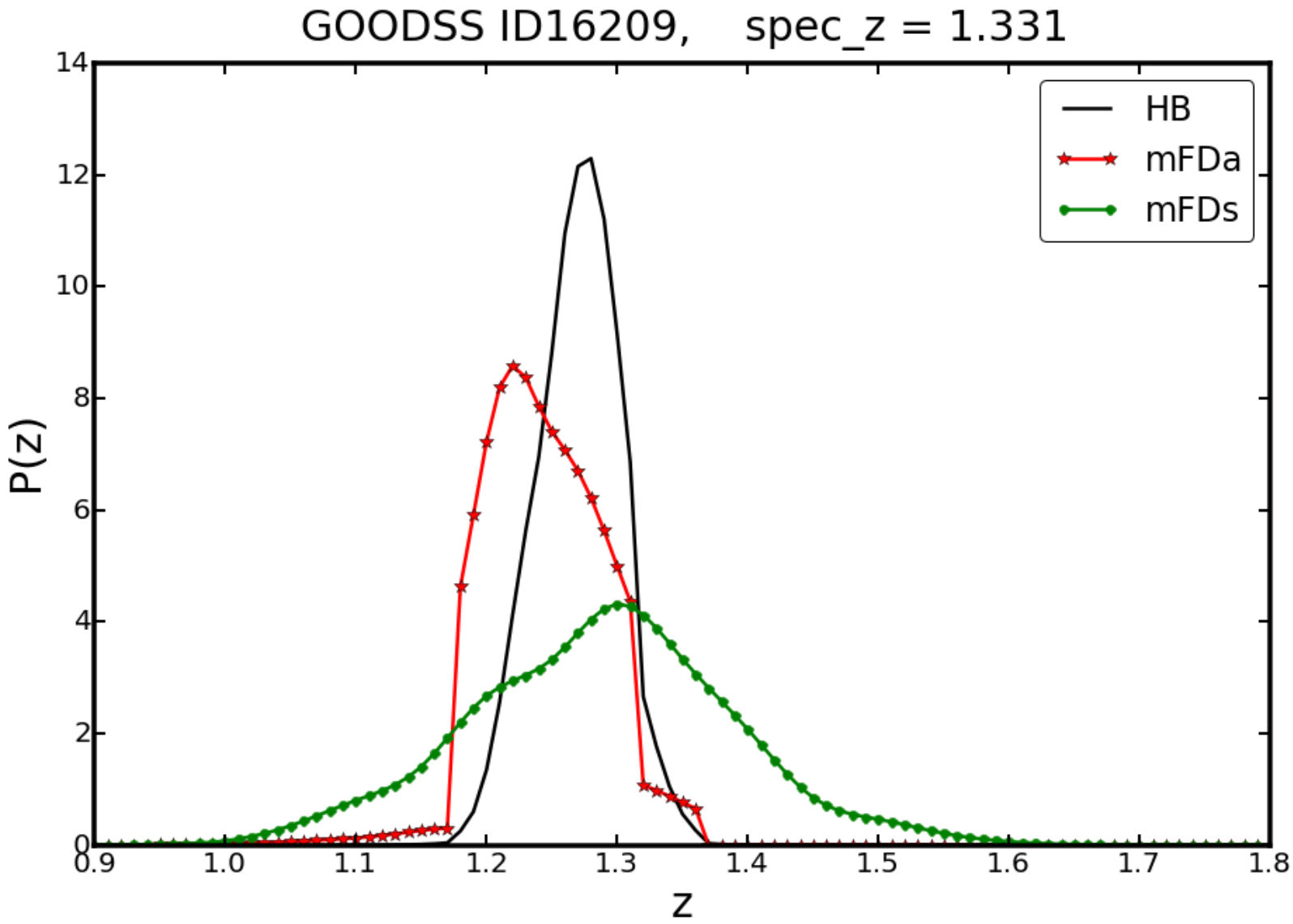
D. Kodra

Open issues: Combining PDF results from multiple codes



D. Kodra

Open issues: Combining PDF results from multiple codes

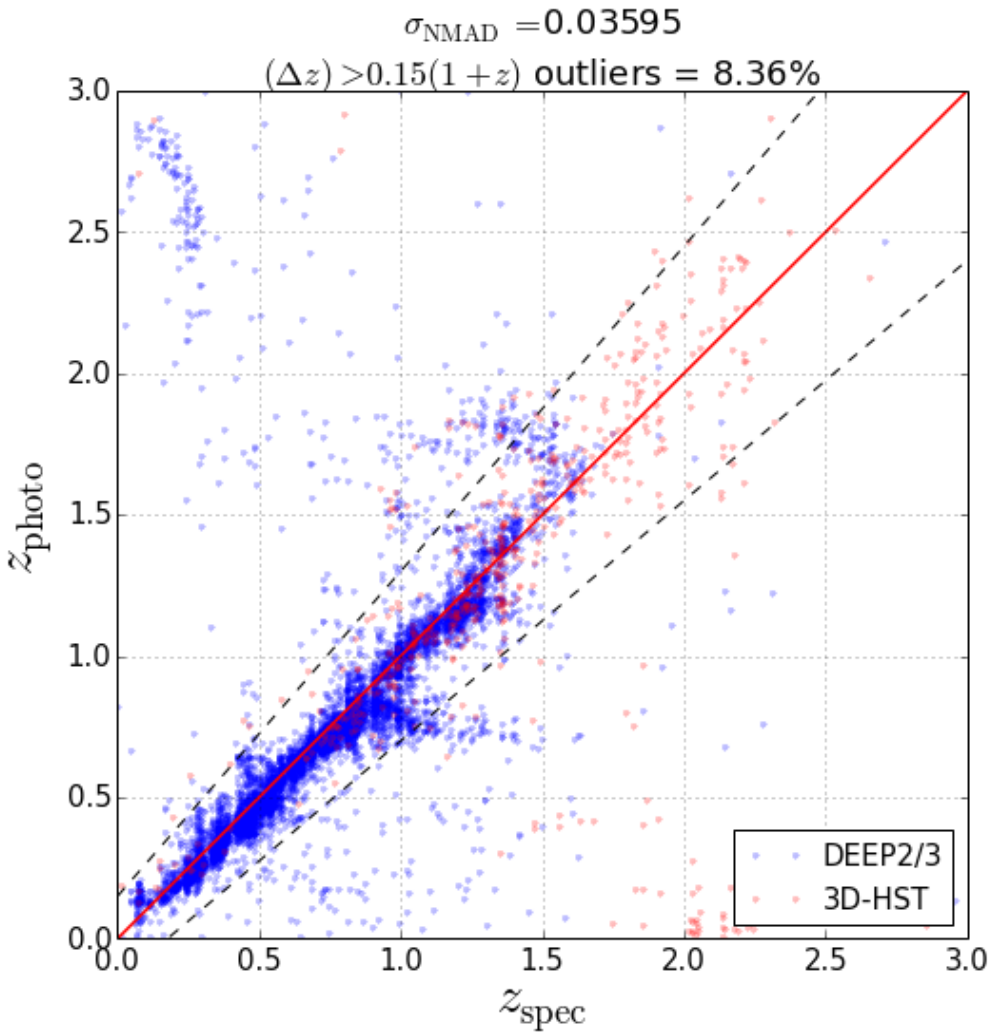


D. Kodra

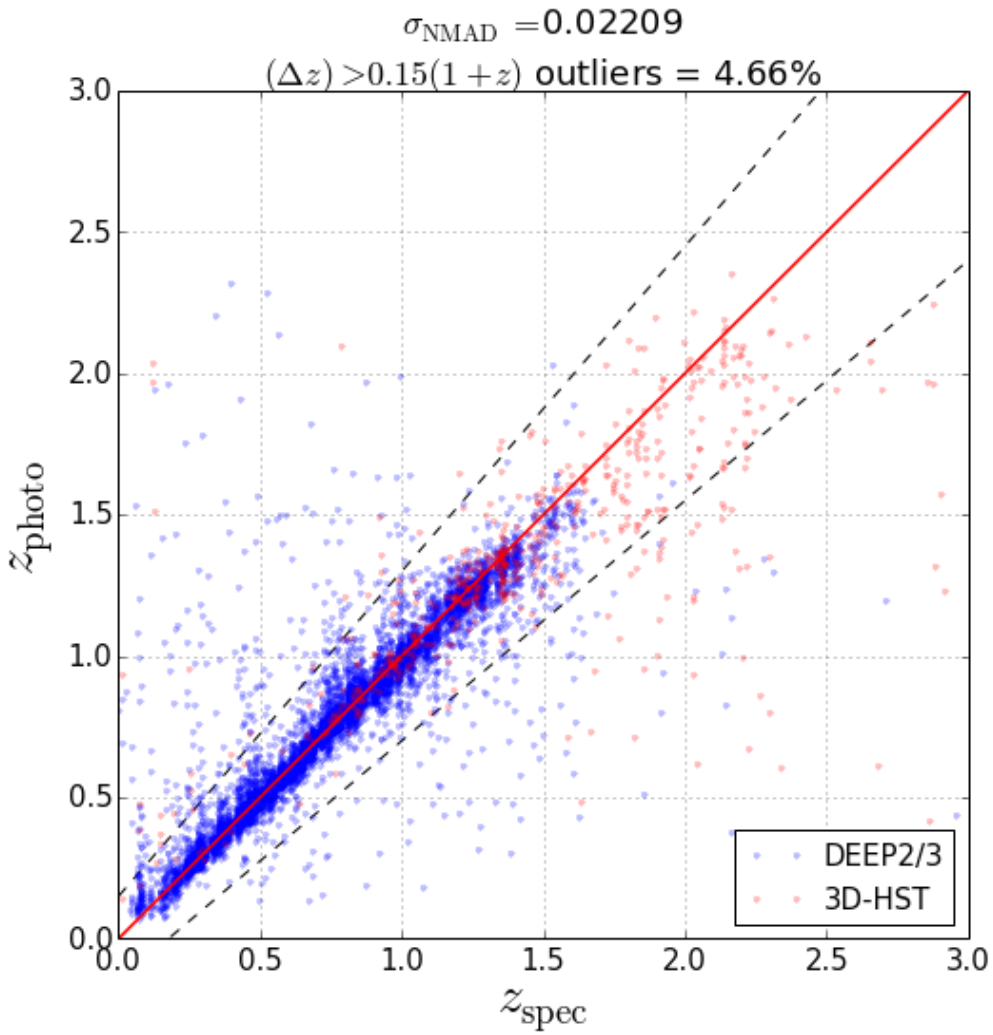
Another possible use case: template-based and training-based methods have different failure modes

- **Identify potential outliers from discrepant results?**

EAZY (template code, untuned)



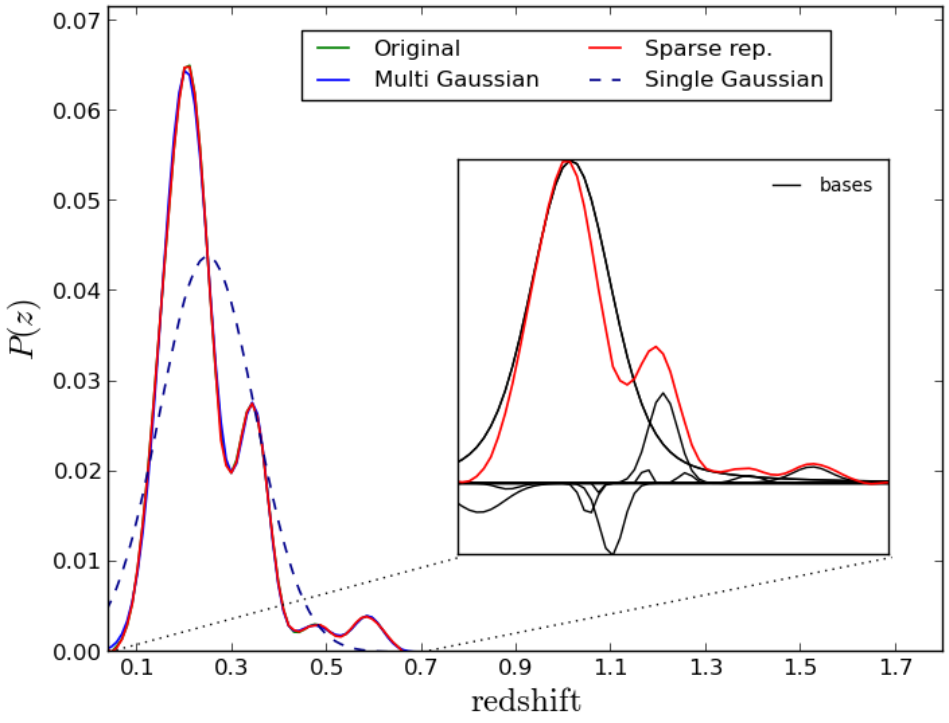
Random Forest Regression

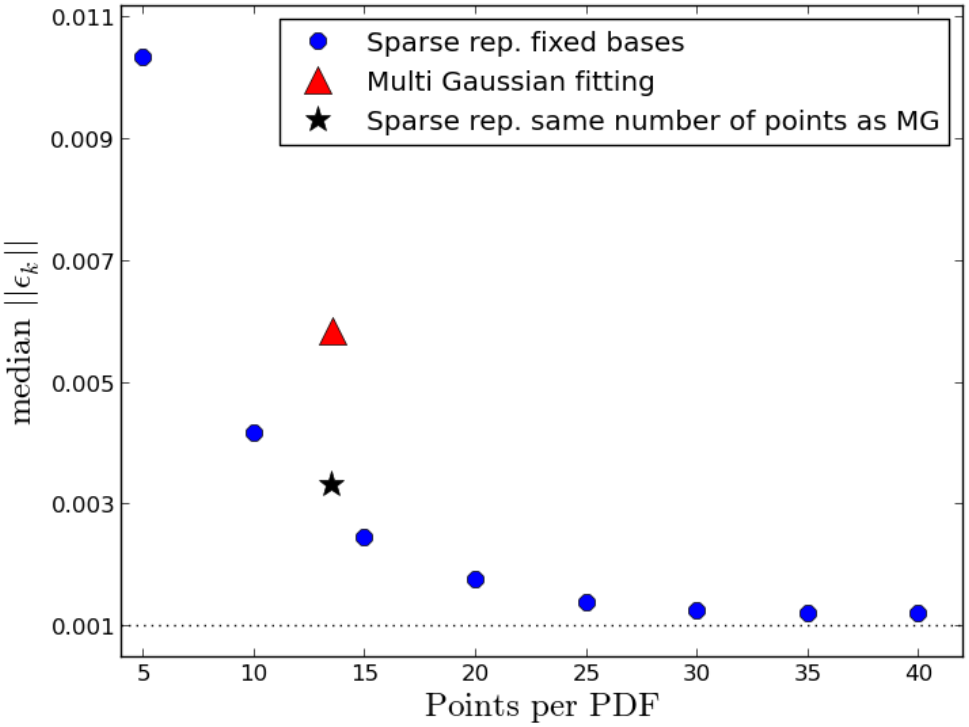


Zhou, JN et al. 2016, in prep.

Open issues: Storing p(*z*,*α*)

- Carrasco-Kind & Brunner 2014 achieved strong compression of photo-z PDFs using sparse representation and well-chosen basis set
- For many LSST applications, want 2+-dimensional PDFs
- Can suitably sparse (<few hundred #s) representations be achieved?
- Are samples from PDFs OK for all science cases?

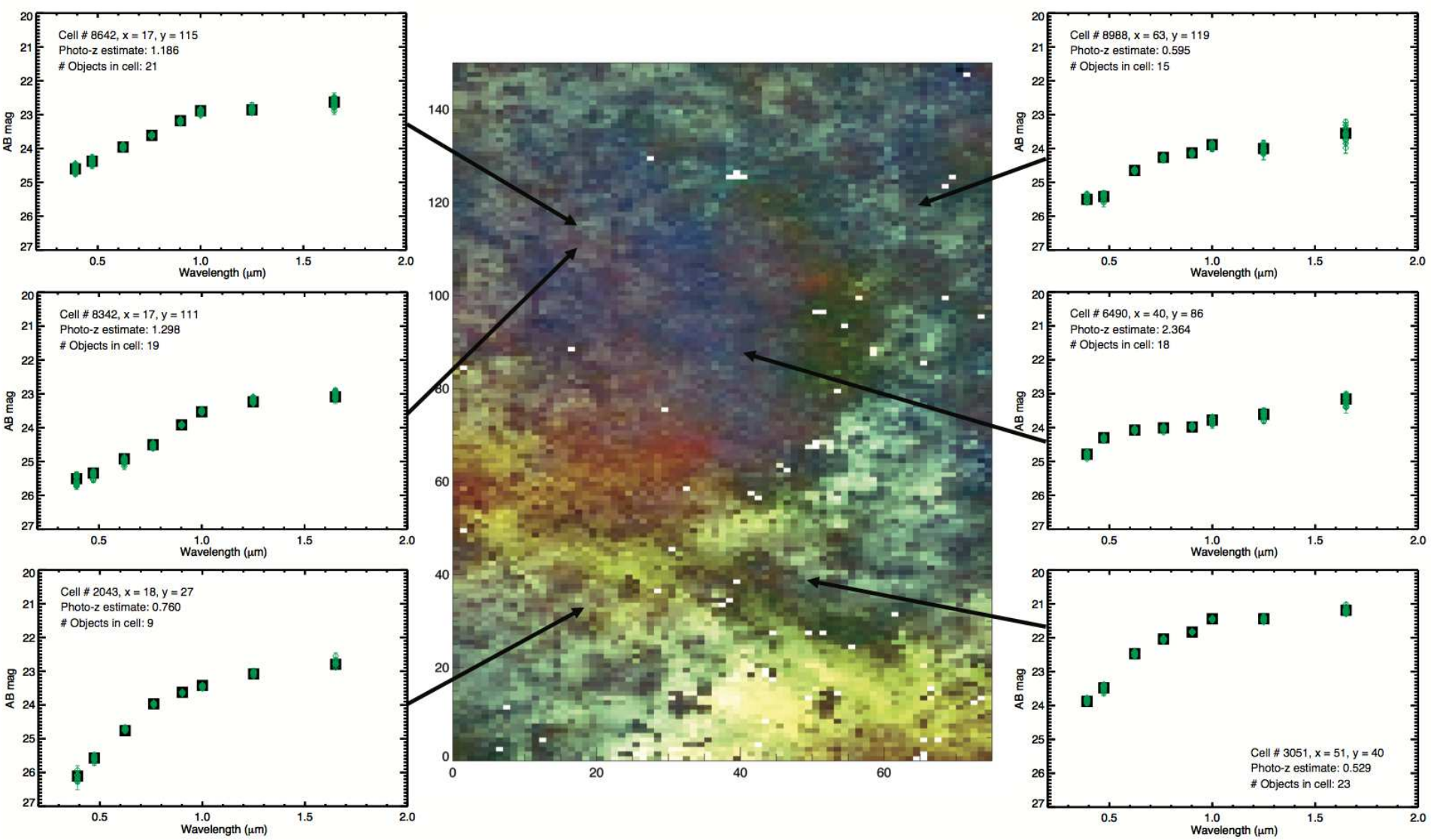




Carrasco-Kind & Brunner 2014

Open issues: Optimizing spectroscopic targeting

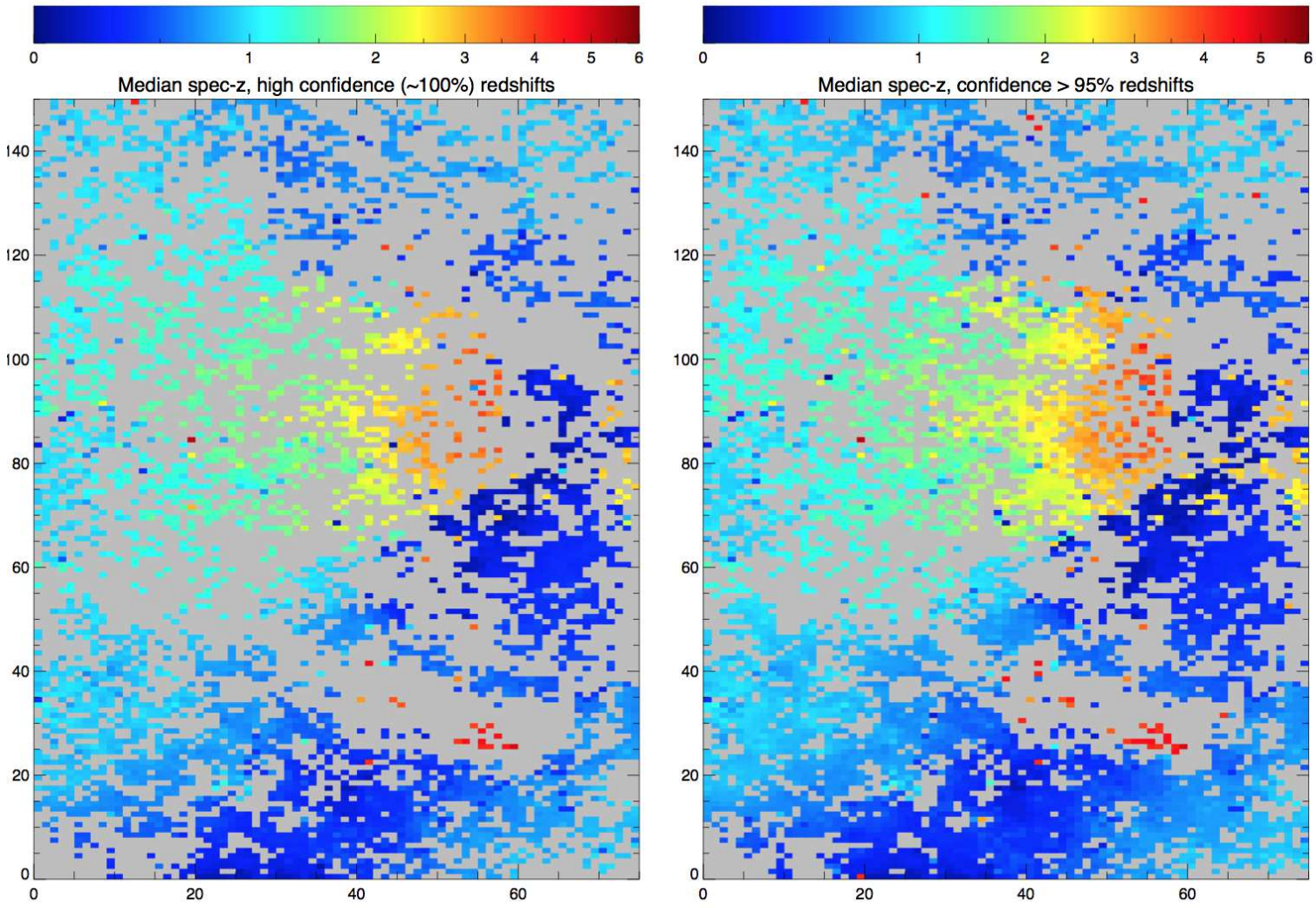
- **Current state of the art: Masters et al. 2015**
- **Self-organized map of galaxy colors**



Masters et al. 2015

Open issues: Optimizing spectroscopic targeting

- **Prioritize cells with few redshifts for spectroscopic follow-up**
- **Are there better ways to do this?**



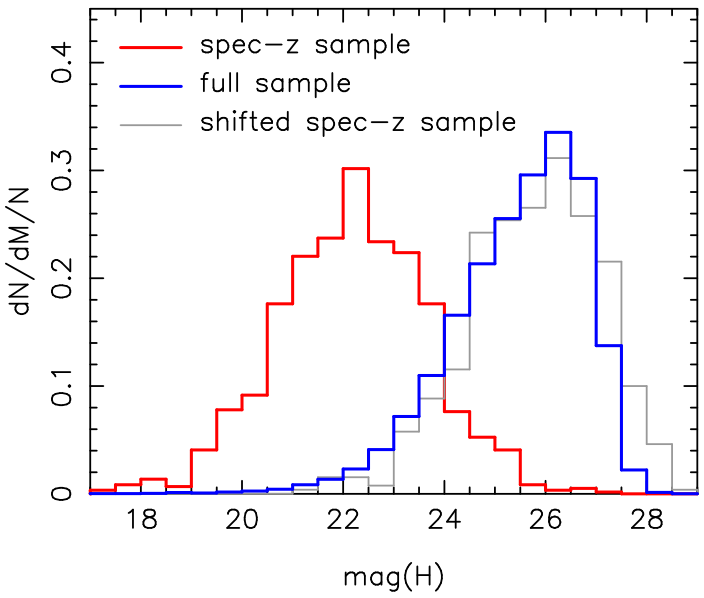
Masters et al. 2015

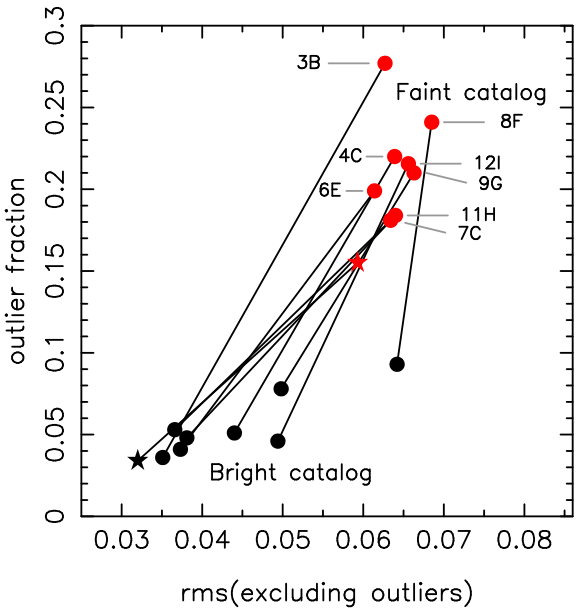
Open issues: Ideal photo-z code?

- What might an ideal LSST photo-z algorithm look like?
  - Trained with >30,000 spectra spanning range of spectra
  - Develops priors & tweaks templates via hierarchical Bayesian hyperparameters
  - Incorporates variations in effective filter wavelengths due to observational conditions: requires applying algorithm to O(1000) measurements instead of O(6)
  - Incorporates AGN classification and AGN photo-z determination: colors are not constant with time for many objects!
  - Want algorithms to be fast: create ML-based emulators for template photo-z's?
  - For bright objects, may also be useful to compare to ML techniques to identify potential outliers

CANDELS code comparison: Dahlen et al. 2013

- **Many tests of photo-z algorithms with deep, high-redshift dataset. Examples:**
  - **Test photo-z performance as degrade photometry (using same test spectroscopic data)**
  - **Dependence of errors on redshift, magnitude, & color**
  - **Investigation of (lack of) consistency between photometric zero point shifts from different codes**
  - **Empirical test of photo-z errors using Δz between close pairs**



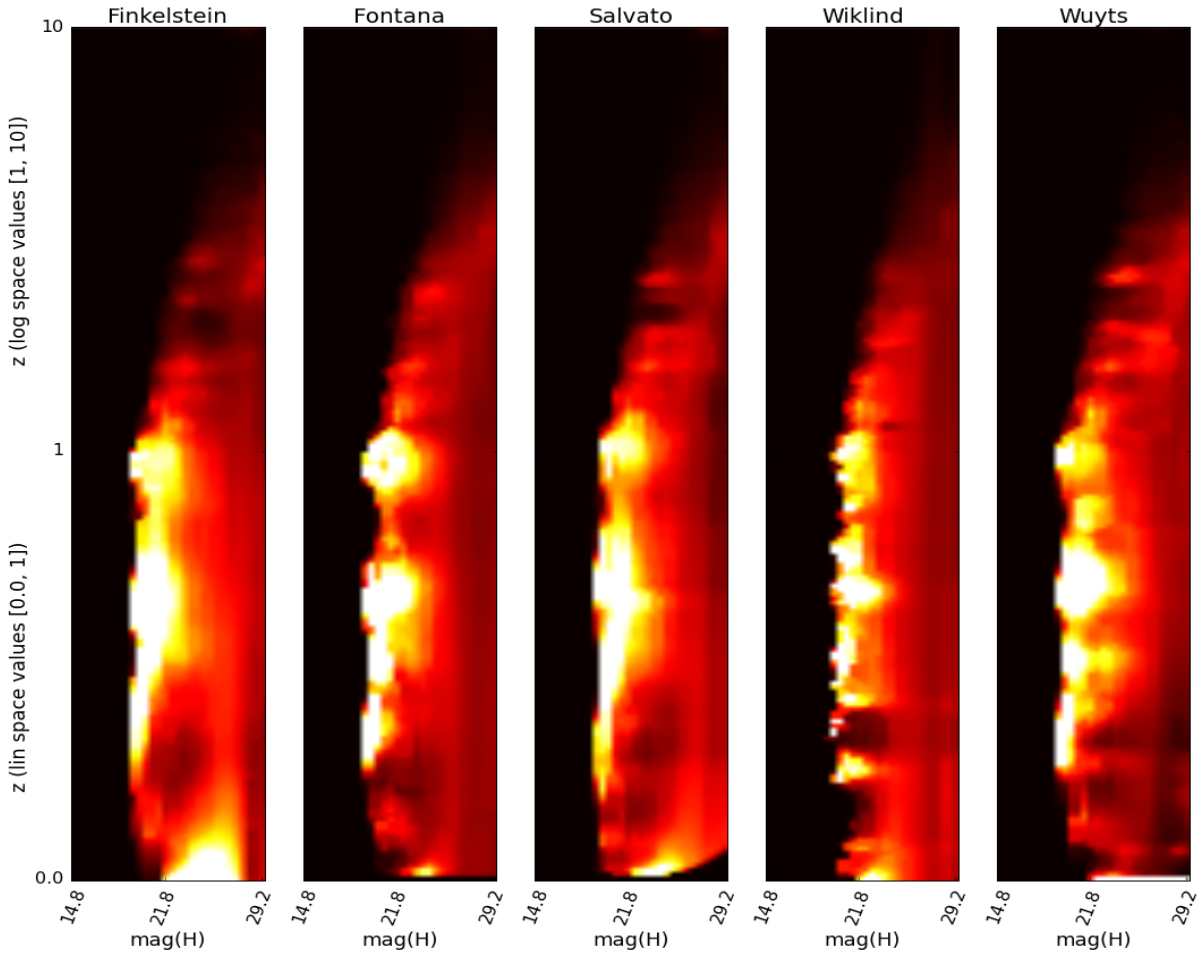


Dahlen et al. 2013

New work: Kodra et al. 2016

- Compare predictions of codes in space of p(z | H)

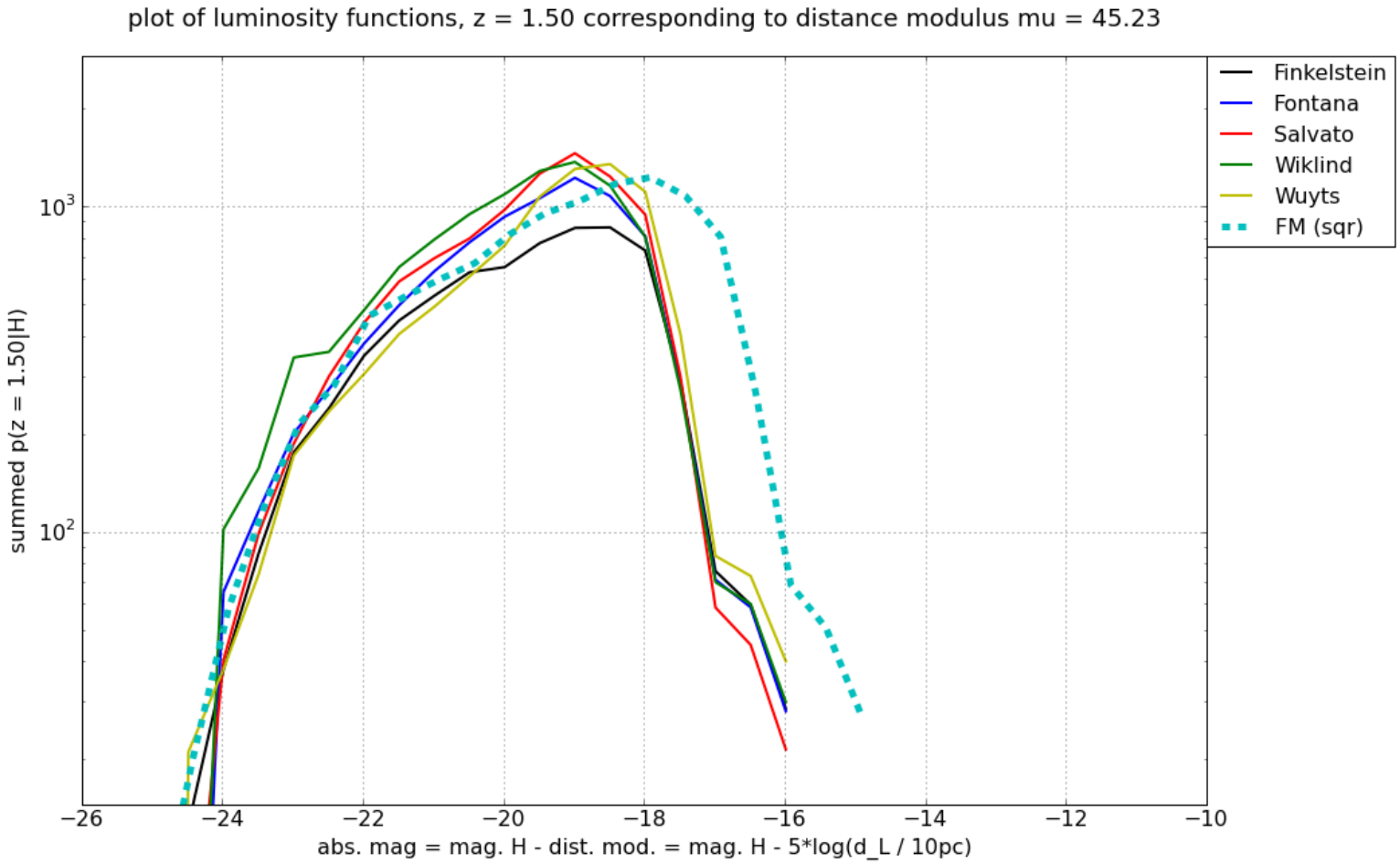
- Disagreement on where there are redshift spikes
- Priors have huge effect at low z (non-monotonic behavior)
- Different effective smoothings
- The performance of these codes for z_{peak} isn't all that different. . .



D. Kodra

New work: Kodra et al. 2016

- **This can have large (factor of few) effects on the inferred number of objects at a given redshift**



D. Kodra

Conclusions

- Training-based methods are easier to get good results from than template-based methods, but don't extrapolate well
- Key issue for LSST is inability to get complete training sets
- Other interesting issues for the next few years:
  - Training algorithms in the presence of false redshifts
  - Making sure p(z)'s have proper coverage
  - Combining results from multiple algorithms
  - Storing multi-dimensional PDFs compactly
  - Optimizing spectroscopic follow-up
  - Defining parameters of ideal LSST algorithm
- Current codes appear sufficient to meet LSST requirements, but are clearly suboptimal. Better photo-z's will greatly increase the value of LSST - e.g. 30% increase in Dark Energy Figure of Merit