Estimating local chlamydia incidence and prevalence using surveillance data: eAppendix 2

Joanna Lewis and Peter White

April 19, 2017

Contents

1 Example: Chlamydia in England, 2012

This example illustrates a method for using chlamydia surveillance data to estimate prevalence. Surveillance data on chlamydia testing and diagnosis rates in England in 2012 were downloaded from: http://www.chlamydiascreening.nhs.uk/ps/data.asp (downloaded 9 February 2016).

	Men			Women		
	15-19 years	20-24 years	Total	15-19 years	20-24 years	Total
Population	1685620	1833395	3519015	1600686	1788156	3388842
Tests	232668	334240	566908	520358	685538	1205896
Diagnoses	15213	33174	48387	42874	45227	88101

Data on sexual behaviour from the third National Study of Sexual Attitudes and Lifestyles (Natsal-3) are available from the UK data service: https://www.ukdataservice.ac.uk/ (downloaded 23 September 2015). These were used to infer 95% confidence intervals for the proportions of men and women, aged 16-19 and 20-24, who were sexually active (see the accompanying R script; note that no 15-year-olds were recruited to Natsal-3). These 95% confidence intervals were in turn used to derive beta-distribution priors for the proportion sexually active within each sex and age group.

1.1 Sampling for testing and diagnosis rates

```
In [1]: import numpy as np
        from numpy import *
        from scipy.stats import beta
        from scipy.optimize import fsolve


        ######################
        # parameters of beta distributions representing the proportion of the population sexually
```

```
# active, by sex and age group
######################

# men, 16-19
[alpha_m_16_19, beta_m_16_19] = fsolve(
    lambda x: array(beta.interval(0.95, x[0], x[1], loc=0, scale=1))
    - (0.6747424, 0.741327698),
    [1,1]
    )
# men, 20-24
[alpha_m_20_24, beta_m_20_24] = fsolve(
    lambda x: array(beta.interval(0.95, x[0], x[1], loc=0, scale=1))
    - (0.8844970, 0.933759842),
    [1,1]
    )
# men, 16-24
[alpha_m_16_24, beta_m_16_24] = fsolve(
    lambda x: array(beta.interval(0.95, x[0], x[1], loc=0, scale=1))
    - (0.8023836019, 0.843403825),
    [1,1]
    )
# women, 16-19
[alpha_f_16_19, beta_f_16_19] = fsolve(
    lambda x: array(beta.interval(0.95, x[0], x[1], loc=0, scale=1))
    - (0.6583593, 0.723554878),
    [1,1]
    )
# women, 20-24
[alpha_f_20_24, beta_f_20_24] = fsolve(
    lambda x: array(beta.interval(0.95, x[0], x[1], loc=0, scale=1))
    - (0.8904135, 0.934417684),
    [1,1]
    )
# women, 16-24
[alpha_f_16_24, beta_f_16_24] = fsolve(
    lambda x: array(beta.interval(0.95, x[0], x[1], loc=0, scale=1))
    - (0.7998634469, 0.837979601),
    [1,1]
    )
```

Next, sample from distributions for the probability of being sexually active, the size of the sexually active population and the testing and diagnosis rates per person per year.

```
In [2]: from scipy.stats import gamma
        from numpy.random import normal
        rs = random.RandomState(12345)

        n_sample = 10000

        # sexually-active populations:
        p_active_m_16_19 = rs.beta(alpha_m_16_19, beta_m_16_19, size=n_sample) # 16-19 yo only
        pop_active_m_15_19 = rs.binomial(1685620, p_active_m_16_19, size=n_sample)

        p_active_m_20_24 = rs.beta(alpha_m_20_24, beta_m_20_24, size=n_sample) # 20-24 yo only
        pop_active_m_20_24 = rs.binomial(1833395, p_active_m_20_24, size=n_sample)

        p_active_m_16_24 = rs.beta(alpha_m_16_24, beta_m_16_24, size=n_sample) # 16-24 yo only
        pop_active_m_15_24 = rs.binomial(3519015, p_active_m_16_24, size=n_sample)

        p_active_f_16_19 = rs.beta(alpha_f_16_19, beta_f_16_19, size=n_sample) # 16-19 yo only
        pop_active_f_15_19 = rs.binomial(1600686, p_active_f_16_19, size=n_sample)

        p_active_f_20_24 = rs.beta(alpha_f_20_24, beta_f_20_24, size=n_sample) # 20-24 yo only
        pop_active_f_20_24 = rs.binomial(1788156, p_active_f_20_24, size=n_sample)

        p_active_f_16_24 = rs.beta(alpha_f_16_24, beta_f_16_24, size=n_sample) # 16-24 yo only
        pop_active_f_15_24 = rs.binomial(3388842, p_active_f_16_24, size=n_sample)

        # testing and diagnosis rates, per person per year
```

```
test_rate_m_15_19 = rs.gamma(232668, 1, size=n_sample)/pop_active_m_15_19
test_rate_m_20_24 = rs.gamma(334240, 1, size=n_sample)/pop_active_m_20_24
test_rate_m_15_24 = rs.gamma(566908, 1, size=n_sample)/pop_active_m_15_24

diag_rate_m_15_19 = rs.gamma(15213, 1, size=n_sample)/pop_active_m_15_19
diag_rate_m_20_24 = rs.gamma(33174, 1, size=n_sample)/pop_active_m_20_24
diag_rate_m_15_24 = rs.gamma(48387, 1, size=n_sample)/pop_active_m_15_24

diag_rate_f_15_19 = rs.gamma(42874, 1, size=n_sample)/pop_active_f_15_19
diag_rate_f_20_24 = rs.gamma(45227, 1, size=n_sample)/pop_active_f_20_24
diag_rate_f_15_24 = rs.gamma(88101, 1, size=n_sample)/pop_active_f_15_24

test_rate_f_15_19 = rs.gamma(520358, 1, size=n_sample)/pop_active_f_15_19
test_rate_f_20_24 = rs.gamma(685538, 1, size=n_sample)/pop_active_f_20_24
test_rate_f_15_24 = rs.gamma(1205896, 1, size=n_sample)/pop_active_f_15_24
```

In [3]:
```
print percentile(test_rate_m_15_24,50)
print percentile(diag_rate_m_15_24,50)
```

```
0.195629009259
0.016699659345
```

1.2 Sampling natural history, behavioural and other parameters

1.2.1 Test performance

Priors for the test performance parameters are beta distributions parameterised directly from literature studies.

In [4]:
```
# test performance

# Horner J. Clin. Microbiol (2005): 32 of 32 infected samples tested +ve
p_true_pos_m = rs.beta(32+1, 0+1, size=n_sample)
# Horner J. Clin. Microbiol (2005): 2 of 952 uninfected samples tested +ve
p_false_pos_m = rs.beta(2+1, 950+1, size=n_sample)
# Low Health Technol Assess (2007): 129 of 141 infected samples tested +ve
p_true_pos_f = rs.beta(129+1, 12+1, size=n_sample)
# Low Health Technol Assess (2007): 4 of 2327 uninfected samples tested +ve
p_false_pos_f = rs.beta(4+1, 2323+1, size=n_sample)
```

1.2.2 Rate of treatment seeking by symptomatic cases

We use a Metropolis-Hastings algorithm to sample for the rate of treatment following onset of symptoms, assuming a constant hazard of treatment beginning with the onset of symptoms. Data consist of the estimated proportion of GUM clinic patients with symptoms whose symptoms had started < 1, 1-2, 2-4, 4-6 and > 6 weeks previously (Mercer *et al.*, *Sex. Transm. Infect.* **83**:400-405; 2007).

	Proportion	
	Estimate	95% Confidence Interval
< 1 week	26.7%	(14.4, 44.2)%
7-13 days	14.4%	(6.1, 30.2)%
14-27 days	20.8%	(13.3, 31.0)%
4-6 weeks	16.6%	(8.5, 29.9)%
> 6 weeks	21.5%	(5.5, 56.4)%

In [5]:
```
# function for calculating likelihood of multinomial data
%run multinomial_pmf.py
```

In [6]:
```
# Find beta distributions corresponding to 95% CIs reported in
# Mercer Sex. Transm. Infect. (2007) (see table above).

a = empty(5)
b = empty(5)
```

```
# < 1 week
[a[0], b[0]] = fsolve(
    lambda x: array(beta.interval(0.95, x[0], x[1], loc=0, scale=1))
    - (0.144, 0.442),
    [1,1]
    )

# 7-13 days
[a[1], b[1]] = fsolve(
    lambda x: array(beta.interval(0.95, x[0], x[1], loc=0, scale=1))
    - (0.061, 0.302),
    [1,1]
    )

# 14-27 days
[a[2], b[2]] = fsolve(
    lambda x: array(beta.interval(0.95, x[0], x[1], loc=0, scale=1))
    - (0.133, 0.310),
    [1,1]
    )

# 28-41 days
[a[3], b[3]] = fsolve(
    lambda x: array(beta.interval(0.95, x[0], x[1], loc=0, scale=1))
    - (0.085, 0.299),
    [1,1]
    )

# 42 days and over
[a[4], b[4]] = fsolve(
    lambda x: array(beta.interval(0.95, x[0], x[1], loc=0, scale=1))
    - (0.055, 0.564),
    [1,1]
    )
```

In [7]:
```
# Metropolis-Hastings to get a sample for rate of treatment

i = 0
att_symp = empty(n_sample+1000) # testing rate per person per year. Allow 1000 extra samples for burn-in
ll = empty(n_sample+1000) # log-likelihood
props = empty([n_sample+1000, 5]) # simulated data, for posterior predictive check
old = 0.04 # starting sample value
new = 0.04 # starting sample value

# simulate probabilities corresponding to data

# proportion expected in each time window
tps = array([0., 7., 14., 28., 42., Inf])
simp_old = exp(-old*tps[:5]) - exp(-old*tps[1:])
simp_new = exp(-new*tps[:5]) - exp(-new*tps[1:])

acc=0.
while i < n_sample+1000: # to do samples for p_test_symp

    new = rs.normal(old, 0.05) # generate a sample from normal distribution

    if new < 0:
        att_symp[i] = old # reject
        ll[i] = -1e10
    else:
        simp_old = exp(-old*tps[:5]) - exp(-old*tps[1:])
        simp_new = exp(-new*tps[:5]) - exp(-new*tps[1:])

        if sum(simp_new > 0) != len(tps) - 1:
            att_symp[i] = old # reject
            ll[i] = -1e10
        else:
            # simulate probabilities corresponding to the data
```

```
                log_ratio = \
                    sum(beta.logpdf(simp_new, a, b, loc=0, scale=1)) \
                    - sum(beta.logpdf(simp_old, a, b, loc=0, scale=1))

                if log(rs.uniform(0,1)) <  log_ratio:
                    att_symp[i] = new # accept
                    ll[i] = sum(beta.logpdf(simp_new, a, b, loc=0, scale=1))
                    old = new
                    acc = acc+1
                else:
                    att_symp[i] = old # reject
                    ll[i] = sum(beta.logpdf(simp_old, a, b, loc=0, scale=1))

            props[i] = simp_old
            i = i+1

        att_symp = att_symp[1000:] # remove burn-in samples
        ll = ll[1000:] # log-likelihood

        print acc/(n_sample+1000) # print the proportion of samples accepted
        print mean(att_symp)*365.25
        print array(percentile(att_symp, [2.5, 25, 50, 75, 97.5]))*365.25

        att_symp = att_symp*365.25 # convert rate from day^-1 to year^-1
```

```
0.226545454545
14.4054933827
[  8.59839927  11.77758657  13.98780214  16.69605121  22.22498957]
```

In [8]:
```
        # Figure 1
        # diagnostics and posterior predictive checks

        %matplotlib inline
        import matplotlib.pyplot as plt

        from numpy.random import multinomial

        fig = plt.figure(figsize = (10,10))

        ax1 = fig.add_subplot(221)
        ax1.plot(att_symp, alpha=0.5)

        ax2 = fig.add_subplot(222)

        ax2.plot(range(43),median(att_symp/365.25)*exp(-median(att_symp/365.25)*array(range(43))),'b')
        #plt.plot(range(50),percentile(att_symp, 2.5)*exp(-percentile(att_symp, 2.5)*array(range(50))),'b--')
        #plt.plot(range(50),percentile(att_symp, 97.5)*exp(-percentile(att_symp, 97.5)*array(range(50))),'b--')

        #ax2.set_ylim([0,0.1])
        ax2.set_xlim([0,50])
        ax2.errorbar([3.5,10.5,21,35, 46],
                     [0.267/7, 0.144/7, 0.208/14, 0.166/14, 0.215/10],
                     abs(array([[0.144/7, 0.061/7, 0.133/14, 0.085/14, 0.055/10],
                                [0.442/7, 0.302/7, 0.310/14, 0.299/14, 0.564/10]]
                               ) - array([0.267/7, 0.144/7, 0.208/14, 0.166/14, 0.215/10])
            ), color = 'r', fmt='.')

        ax2.plot([0,7,7,14,14,28,28,42], repeat(percentile(props[:,:4],50,0)/array([7,7,14,14]),2), 'b--')
        ax2.plot([42,50], repeat(percentile(props[:,4],50,0)/array([10]),2), 'b--')
        ax2.fill_between(
            [0,7,7,14,14,28,28,42,42,50],
            repeat(percentile(props,2.5,0)/array([7,7,14,14,10]),2),
            repeat(percentile(props,97.5,0)/array([7,7,14,14,10]),2),
            alpha=0.5
        )

        ax1.set_xlabel('Sample')
        ax1.set_ylabel('Rate of seeking treatment (year^-1)')
```

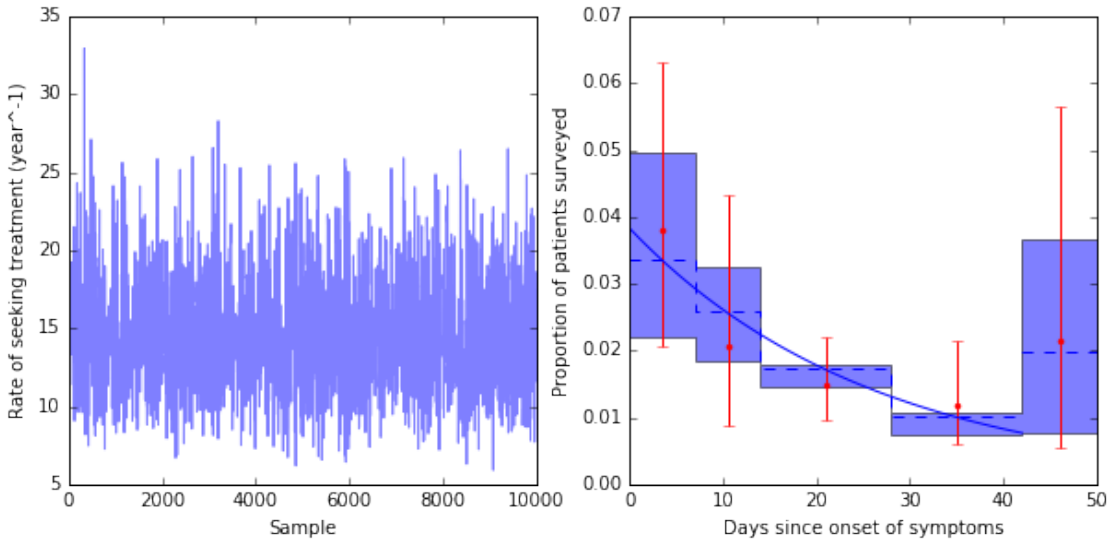

Figure 1: Diagnostic plots for MCMC sampling of treatment seeking rate in symptomatic patients. Left: MCMC chain. Right: posterior predictive check (for description, see text.)

```
ax2.set_xlabel('Days since onset of symptoms')
ax2.set_ylabel('Proportion of patients surveyed')
```

```
Out[8]: <matplotlib.text.Text at 0x10f370390>
```

The MCMC chain is illustrated in the left-hand panel, and seems to have converged well.

The right-hand panel shows the probability density of the time between onset of symptoms and attending the GUM clinic where patients were surveyed, to 42 days (solid blue line). The blue shaded area and dashed line show the central 95% and median of simulated histograms for waiting times to clinic, with bins corresponding to time windows reported in the data. The last bin contains all times longer than six weeks and has been divided by 10 (as opposed to the width of the window) to make it readable. For comparison, red error bars show the reported proportions of patients with treatment-seeking times within each time window (estimate and 95% CI), normalised to be on the same scale as the predictions (blue). The good predictive properties of the model are indicated by the agreement between the data, in red, and the posterior predictions in blue.

1.2.3 Rate of spontaneous clearance of infection

Rates of spontaneous clearance of infection in men and women were sampled using MCMC and the STAN software (see accompanying `R` scripts, STAN model files and references), following the model presented by Price *et al.* in *Stat. Med.* **32**:1547-1560.

```
In [9]: # Figure 2
```

```
import csv
sc_m = empty(n_sample) # clearance rate per person per year
with open('stan/chlamydia_two_exponentials_men.csv', 'rU') as m:
    reader = csv.reader(m)
    i=0
    next(reader) # skip the header row
    for row in reader:
        sc_m[i] = row[0]
        i = i+1

sc_f = empty(n_sample) # clearance rate per person per year
with open('stan/chlamydia_two_exponentials_women.csv', 'rU') as f:
    reader = csv.reader(f)
```

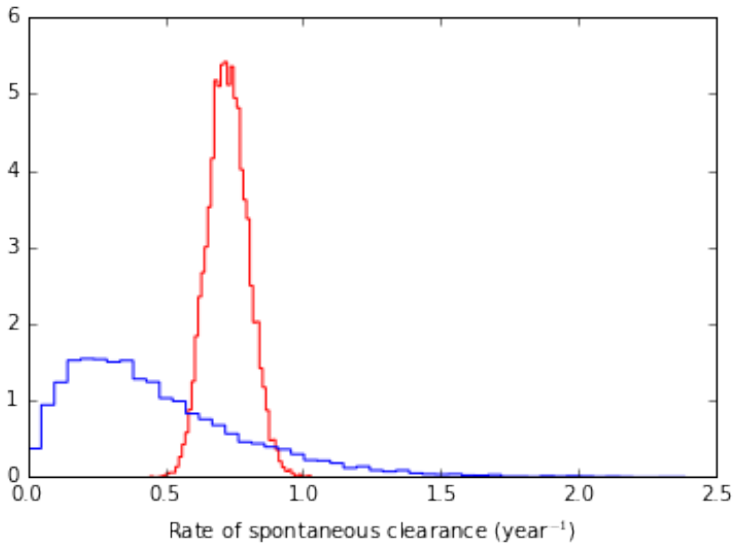

Figure 2: Sampled rates of spontaneous chlamydia clearance in men (blue) and women (red).

```
    i=0
    next(reader) # skip the header row
    for row in reader:
        sc_f[i] = row[0]
        i = i+1

h=plt.hist(sc_f, bins=50, histtype='step', normed=True, color='r')
h=plt.hist(sc_m, bins=50, histtype='step', normed=True, color='b')
plt.xlabel('Rate of spontaneous clearance (year$^{-1}$)')

print 'Mean spontaneous clearance rate in men:', mean(sc_m)
print 'Median (central 95% credible interval) for spontaneous clearance rate in men: \n \t', \
    percentile(sc_m, 50), percentile(sc_m, (2.5,97.5))
print 'Mean spontaneous clearance rate in women:', mean(sc_f)
print 'Median (central 95% credible interval) for spontaneous clearance rate in women:\n \t', \
    percentile(sc_f, 50), percentile(sc_f, (2.5,97.5))
```

```
Mean spontaneous clearance rate in men: 0.469884686724
Median (central 95% credible interval) for spontaneous clearance rate in men:
         0.395824502635 [ 0.05879628  1.27387539]
Mean spontaneous clearance rate in women: 0.727835970463
Median (central 95% credible interval) for spontaneous clearance rate in women:
         0.725636286054 [ 0.59114145  0.87428941]
```

1.2.4 Proportion of incident infections asymptomatic

Finally, we infer the proportion of infections which are asymptomatic by calibrating to the Natsal-3 prevalence estimates in 16-25-year-old men and women.

```
In [10]: from scipy.stats import beta

        [alpha_prev_m, beta_prev_m] = fsolve(
            lambda x: array(beta.interval(0.95, x[0], x[1], loc=0, scale=1))
            - (0.015, 0.034), # Natsal-3 prevalence in men
            [1,1]
            )
```

```
prev_m = rs.beta(alpha_prev_m, beta_prev_m, size=n_sample)

# generate samples for prevalence
[alpha_prev_f, beta_prev_f] = fsolve(
    lambda x: array(beta.interval(0.95, x[0], x[1], loc=0, scale=1))
    - (0.022, 0.043), # Natsal-3 prevalence in women
    [1,1]
    )

prev_f = rs.beta(alpha_prev_f, beta_prev_f, size=n_sample)
```

In [11]:
```
# This script also contains the functions linking observed tests, symptomatic/asymptomatic/toal diagnoses,
# incidence, prevalence, screening and other model parameters
# Running it takes a little while because of all the symbolic algebra
%run test_diag_fun.py
```

In [12]:
```
# incidence, screening and proportion of incident infections asymptomatic in men

inc_m = np.zeros(n_sample)
scr_m = np.zeros(n_sample)
p_asymp_m = np.zeros(n_sample)

for i in xrange(n_sample):
    def tmpfun(inc, scr, p_asymp):
        [tr, dr] = test_diag_fun(
            array([
                inc,
                scr,
                1-p_asymp, # proportion of incident infections which are symptomatic
                sc_m[i], # rate of self-clear
                att_symp[i],
                p_true_pos_m[i],
                p_false_pos_m[i]
                ]))
        prev = dyn_fun(
            inc*p_asymp,
            sc_m[i] + scr*p_true_pos_m[i],
            inc*(1-p_asymp),
            scr*p_true_pos_m[i] + att_symp[i]*p_true_pos_m[i]
        )
        return (tr - test_rate_m_15_24[i],
                dr - diag_rate_m_15_24[i],
                prev - prev_m[i])

    [inc_m[i], scr_m[i], p_asymp_m[i]] = fsolve(lambda x: tmpfun(x[0], x[1], x[2]), [0.09, 0.25, 0.9] )
```

In [13]:
```
# Figure 3
# incidence, screening and proportion of incident infections asymptomatic in women

inc_f = np.zeros(n_sample)
scr_f = np.zeros(n_sample)
p_asymp_f = np.zeros(n_sample)

for i in xrange(n_sample):
    def tmpfun(inc, scr, p_asymp):
        [tr, dr] = test_diag_fun(
            array([
                inc,
                scr,
                1-p_asymp, # proportion of incident infections which are symptomatic
                sc_f[i], # rate of self-clear
                att_symp[i],
                p_true_pos_f[i],
                p_false_pos_f[i]
                ]))
        prev = dyn_fun(
            inc*p_asymp,
            sc_f[i] + scr*p_true_pos_f[i],
            inc*(1-p_asymp),
            scr*p_true_pos_f[i] + att_symp[i]*p_true_pos_f[i]
```

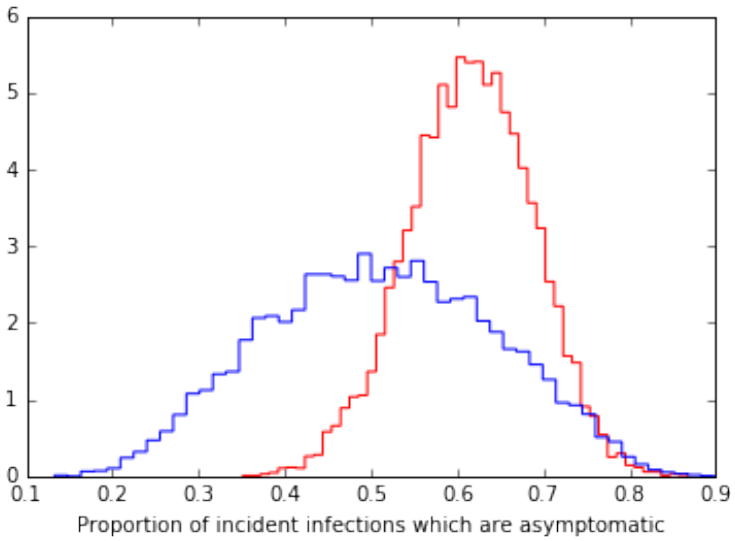

Figure 3: Samples for the proportion of incident infections which are asymptomatic in men (blue) and women (red), calibrated to Natsal-3 prevalence estimates in 16-24-year-olds.

```
            )
            return (tr - test_rate_f_15_24[i],
                    dr - diag_rate_f_15_24[i],
                    prev - prev_f[i])

        [inc_f[i], scr_f[i], p_asymp_f[i]] = fsolve(lambda x: tmpfun(x[0], x[1], x[2]), [0.09, 0.25, 0.9] )
```

```
In [14]: h=plt.hist(p_asymp_f, bins=50, histtype='step', normed=True, color='r')
         h=plt.hist(p_asymp_m, bins=50, histtype='step', normed=True, color='b')
         plt.xlabel('Proportion of incident infections which are asymptomatic')

         print 'Mean proportion asymptomatic in men:', mean(p_asymp_m)
         print 'Median (central 95% credible interval) for proportion asymptomatic in men: \n \t', \
             percentile(p_asymp_m, 50), percentile(p_asymp_m, (2.5,97.5))
         print 'Mean proportion asymptomatic in women:', mean(p_asymp_f)
         print 'Median (central 95% credible interval) for proportion asymptomatic in women:\n \t', \
             percentile(p_asymp_f, 50), percentile(p_asymp_f, (2.5,97.5))
```

```
Mean proportion asymptomatic in men: 0.510862240435
Median (central 95% credible interval) for proportion asymptomatic in men:
         0.509889142897 [ 0.26393408  0.75872662]
Mean proportion asymptomatic in women: 0.615291469484
Median (central 95% credible interval) for proportion asymptomatic in women:
         0.616465413009 [ 0.46763845  0.75205518]
```

1.3 Estimating national prevalence

The sampled parameter values are now used to infer prevalence in men and women in different age groups.

```
In [15]: from scipy.optimize import fsolve
```

```
In [16]: # men first...
         prev_m_15_19 = np.zeros(n_sample)
         inc_m_15_19 = np.zeros(n_sample)
         scr_m_15_19 = np.zeros(n_sample)

         for i in xrange(n_sample):
```

```
        [inc_m_15_19[i], scr_m_15_19[i]] = fsolve(lambda x: test_diag_fun(concatenate([
                        x, array([
                                1-p_asymp_m[i], # proportion of incident infections which are symptomatic
                                sc_m[i], # rate of self-clear
                                att_symp[i],
                                p_true_pos_m[i],
                                p_false_pos_m[i]
                            ])])) - array([test_rate_m_15_19[i],diag_rate_m_15_19[i]]), [0.09, 0.25])
        prev_m_15_19[i] = dyn_fun(
            inc_m_15_19[i]*p_asymp_m[i],
            sc_m[i] + scr_m_15_19[i]*p_true_pos_m[i],
            inc_m_15_19[i]*(1-p_asymp_m[i]),
            scr_m_15_19[i]*p_true_pos_m[i] + att_symp[i]*p_true_pos_m[i]
        )
```

In [17]:
```
prev_m_20_24 = np.zeros(n_sample)
inc_m_20_24 = np.zeros(n_sample)
scr_m_20_24 = np.zeros(n_sample)

for i in xrange(n_sample):
    [inc_m_20_24[i], scr_m_20_24[i]] = fsolve(lambda x: test_diag_fun(concatenate([
                    x, array([
                            1-p_asymp_m[i], # proportion of incident infections which are symptomatic
                            sc_m[i], # rate of self-clear
                            att_symp[i],
                            p_true_pos_m[i],
                            p_false_pos_m[i]
                        ])])) - array([test_rate_m_20_24[i],diag_rate_m_20_24[i]]), [0.09, 0.25])
    prev_m_20_24[i] = dyn_fun(
        inc_m_20_24[i]*p_asymp_m[i],
        sc_m[i] + scr_m_20_24[i]*p_true_pos_m[i],
        inc_m_20_24[i]*(1-p_asymp_m[i]),
        att_symp[i]*p_true_pos_m[i]
    )
```

In [18]:
```
# ... then women
prev_f_15_19 = np.zeros(n_sample)
inc_f_15_19 = np.zeros(n_sample)
scr_f_15_19 = np.zeros(n_sample)

for i in xrange(n_sample):
    [inc_f_15_19[i], scr_f_15_19[i]] = fsolve(lambda x: test_diag_fun(concatenate([
                    x, array([
                            1-p_asymp_f[i], # proportion of incident infections which are symptomatic
                            sc_f[i], # rate of self-clear
                            att_symp[i],
                            p_true_pos_f[i],
                            p_false_pos_f[i]
                        ])])) - array([test_rate_f_15_19[i],diag_rate_f_15_19[i]]), [0.03, 0.44])
    prev_f_15_19[i] = dyn_fun(
        inc_f_15_19[i]*p_asymp_f[i],
        sc_f[i] + scr_f_15_19[i]*p_true_pos_f[i],
        inc_f_15_19[i]*(1-p_asymp_f[i]),
        scr_f_15_19[i]*p_true_pos_f[i] + att_symp[i]*p_true_pos_f[i]
    )
```

In [19]:
```
prev_f_20_24 = np.zeros(n_sample)
inc_f_20_24 = np.zeros(n_sample)
scr_f_20_24 = np.zeros(n_sample)

for i in xrange(n_sample):
    [inc_f_20_24[i], scr_f_20_24[i]] = fsolve(lambda x: test_diag_fun(concatenate([
                    x, array([
                            1-p_asymp_f[i], # proportion of incident infections which are symptomatic
                            sc_f[i], # rate of self-clear
                            att_symp[i],
                            p_true_pos_f[i],
                            p_false_pos_f[i]
                        ])])) - array([test_rate_f_20_24[i],diag_rate_f_20_24[i]]), [0.03, 0.44])
    prev_f_20_24[i] = dyn_fun(
```

```
            inc_f_20_24[i]*p_asymp_f[i],
            sc_f[i] + scr_f_20_24[i]*p_true_pos_f[i],
            inc_f_20_24[i]*(1-p_asymp_f[i]),
            scr_f_20_24[i]*p_true_pos_f[i] + att_symp[i]*p_true_pos_f[i]
        )
```

In [20]:
```
#plt.rc("savefig", dpi=300) # for high-resolution version
# Figure 4
# ...and now plot sampled prevalence by age group

fig = plt.figure(figsize = (10,10))

ax1 = fig.add_subplot(221)
h_2012_m_15_19 = ax1.hist(
    prev_m_15_19, bins=20, normed=true, histtype='step', color='cyan', label='15-19 years')
h_2012_m_20_24 = ax1.hist(
    prev_m_20_24, bins=20, normed=true, histtype='step', color='blue', label='20-24 years')
ax1.errorbar(0.001, 25, xerr=[[0],[0.022-0.001]], ecolor='cyan', capsize=10)
ax1.errorbar(0.022, 30, xerr=[[0],[0.052-0.022]], ecolor='blue', capsize=10)
ax1.annotate('18-19 years', [0.001, 25], color='0.5')
ax1.annotate('20-24 years', [0.022, 30], color='0.5')
ax1.set_xlabel('Prevalence')
ax1.set_xlim(0,0.1)
ax1.set_ylim(0,115)
ax1.set_title('Sexually active men')
ax1.legend()

ax2 = fig.add_subplot(222)
h_2012_f_15_19 = ax2.hist(
    prev_f_15_19, bins=20, normed=true, histtype='step', color='fuchsia', label='15-19 years')
h_2012_f_20_24 = ax2.hist(
    prev_f_20_24, bins=20, normed=true, histtype='step', color='r', label='20-24 years')
ax2.errorbar(0.009, 20, xerr=[[0],[0.058-0.009]], ecolor='fuchsia', capsize=10)
ax2.errorbar(0.025, 25, xerr=[[0],[0.086-0.025]], ecolor='fuchsia', capsize=10)
ax2.errorbar(0.017, 30, xerr=[[0],[0.042-0.017]], ecolor='r', capsize=10)
ax2.annotate('16-17 years', [0.009, 20], color='0.5')
ax2.annotate('18-19 years', [0.025, 25], color='0.5')
ax2.annotate('20-24 years', [0.017, 30], color='0.5')
ax2.set_xlabel('Prevalence')
ax2.set_xlim(0,0.1)
ax2.set_ylim(0,115)
ax2.set_title('Sexually active women')
ax2.legend()

print 'Central 95% credible interval for sexually active men, 15-19 years: \n \t', \
    percentile(prev_m_15_19, (2.5, 97.5))
print 'Central 95% credible interval for sexually active men, 20-24 years: \n \t', \
    percentile(prev_m_20_24, (2.5, 97.5))
print 'Central 95% credible interval for sexually active women, 15-19 years: \n \t', \
    percentile(prev_f_15_19, (2.5, 97.5))
print 'Central 95% credible interval for sexually active women, 20-24 years: \n \t', \
    percentile(prev_f_20_24, (2.5, 97.5))
```

```
Central 95% credible interval for sexually active men, 15-19 years:
         [ 0.01122032  0.02554061]
Central 95% credible interval for sexually active men, 20-24 years:
         [ 0.01785592  0.04042643]
Central 95% credible interval for sexually active women, 15-19 years:
         [ 0.02590103  0.05018259]
Central 95% credible interval for sexually active women, 20-24 years:
         [ 0.01923659  0.03820337]
```

In these plots, step histograms show the sampled values for prevalence in men and women, by age group. The horizontal bars give 95% confidence intervals for prevalence in comparable age groups, estimated from Natsal-3. They show the agreement between our surveillance-based method and the population-based survey.

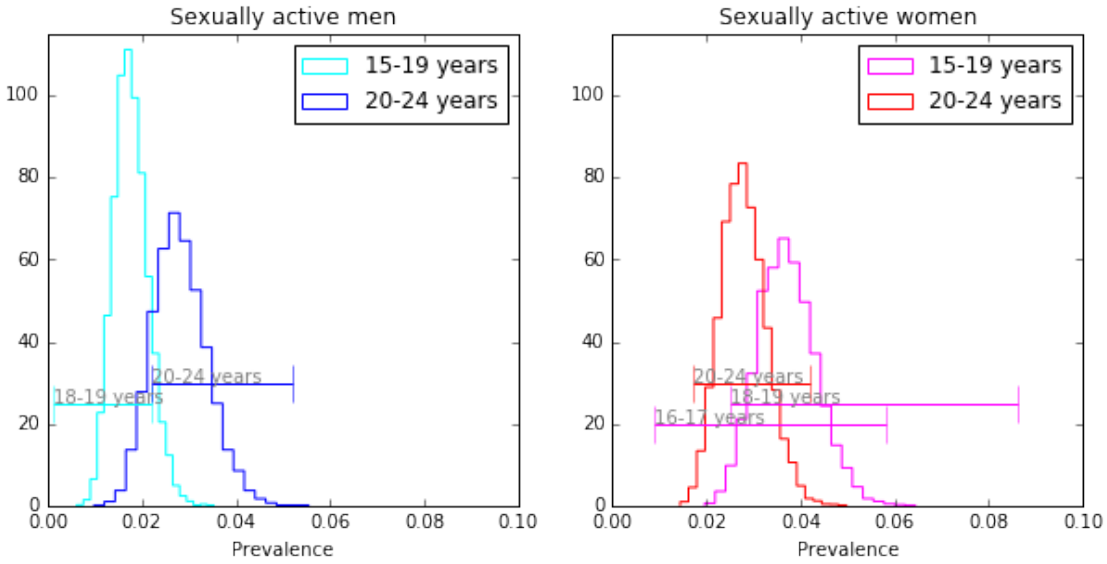

Figure 4: Sampled chlamydia prevalence in men (left) and women (right), by age group. Stepped histograms show samples. Horizontal bars give 95% confidence intervals for prevalence in comparable age groups, estimated from Natsal-3.

1.4 Symptomatic and asymptomatic diagnoses

Although the data does not report the number of diagnoses that were in symptomatic and asymptomatic cases, we can propose different possible numbers of symptomatic and asymptomatic diagnoses and examine the inferences which would have followed in each case.

```
In [21]: # men first...
         prev_m = np.zeros(n_sample)
         inc_m = np.zeros(n_sample)
         scr_m = np.zeros(n_sample)
         p_symp_m = np.zeros(n_sample)

         # there were 48387 diagnoses in men aged 15-24
         # don't allow all symptomatic or all asymptomatic - messes with gamma distributions
         sample_symp_m = ceil(48386*rs.uniform(size = n_sample))
         diag_rate_symp_m_15_24 = rs.gamma(sample_symp_m, 1, size=n_sample)/pop_active_m_15_24

         sample_asymp_m = 48387 - sample_symp_m
         diag_rate_asymp_m_15_24 = rs.gamma(sample_asymp_m, 1, size=n_sample)/pop_active_m_15_24

         for i in xrange(n_sample):
             [inc_m[i], scr_m[i], p_symp_m[i]] = fsolve(lambda x: test_diag_sym_asym_fun(concatenate([
                             x, array([
                                     sc_m[i], # rate of self-clear
                                     att_symp[i],
                                     p_true_pos_m[i],
                                     p_false_pos_m[i]
                                 ])])) - \
                                 array([
                                     test_rate_m_15_24[i],
                                     diag_rate_symp_m_15_24[i],
                                     diag_rate_asymp_m_15_24[i]
                                     ]),
                                 [0.01, 0.3, 0.21])
             prev_m[i] = dyn_fun(
                 inc_m[i]*(1-p_symp_m[i]),
                 sc_m[i] + scr_m[i]*p_true_pos_m[i],
```

```
        inc_m[i]*p_symp_m[i],
        sc_m[i] + scr_m[i]*p_true_pos_m[i] + att_symp[i]*p_true_pos_m[i])
```

In [22]:
```
# ...then women
prev_f = np.zeros(n_sample)
inc_f = np.zeros(n_sample)
scr_f = np.zeros(n_sample)
p_symp_f = np.zeros(n_sample)

# there were 88101 diagnoses in women aged 15-24
# don't allow all symptomatic or all asymptomatic - messes with gamma distributions
sample_symp_f = ceil(88100*rs.uniform(size = n_sample))
diag_rate_symp_f_15_24 = rs.gamma(sample_symp_f, 1, size=n_sample)/pop_active_f_15_24

sample_asymp_f = 88101 - sample_symp_f
diag_rate_asymp_f_15_24 = rs.gamma(sample_asymp_f, 1, size=n_sample)/pop_active_f_15_24

for i in xrange(n_sample):
    [inc_f[i], scr_f[i], p_symp_f[i]] = fsolve(lambda x: test_diag_sym_asym_fun(concatenate([
                    x, array([
                            sc_f[i], # rate of self-clear
                            att_symp[i],
                            p_true_pos_f[i],
                            p_false_pos_f[i]
                        ])])) - \
                        array([
                            test_rate_f_15_24[i],
                            diag_rate_symp_f_15_24[i],
                            diag_rate_asymp_f_15_24[i]
                            ]),
                        [0.01, 0.3, 0.21])
    prev_f[i] = dyn_fun(
        inc_f[i]*(1-p_symp_f[i]),
        sc_f[i] + scr_f[i]*p_true_pos_f[i],
        inc_f[i]*p_symp_f[i],
        sc_f[i] + scr_f[i]*p_true_pos_f[i] + att_symp[i]*p_true_pos_f[i])
```

In [23]:
```
# Figure 5

fig = plt.figure(figsize = (10,12))
xtk_m = [0, 10000, 20000, 30000, 40000] # x-axis ticks for men
xtk_f = [0, 20000, 40000, 60000, 80000] # x-axis ticks for women

ax1 = fig.add_subplot(421)
ax1.plot(100*(1-sample_symp_m/48387), prev_m, ".", alpha = 0.1)
ax1.fill_between([0,50000], 0.015, 0.034, facecolor="b", alpha=0.3)
ax1.plot([40,40],[0,1],"--b")
ax1.plot([20,20],[0,1],"--b")
ax1.set_xlim([0,100])
ax1.set_ylim([0,0.1])
ax1.set_ylabel("Prevalence")
ax1.set_title("Sexually active men, 15-24 years")

ax2 = fig.add_subplot(422)
ax2.plot(100*(1-sample_symp_f/88101), prev_f, ".r", alpha = 0.1)
ax2.fill_between([0,100000], 0.022, 0.043, facecolor="r", alpha=0.3)
ax2.plot([55,55],[0,1],"--r")
ax2.plot([30,30],[0,1],"--r")
ax2.set_xlim([0,100])
ax2.set_ylim([0,0.1])
ax2.set_title("Sexually active women, 15-24 years")

ax3 = fig.add_subplot(423)
ax3.plot(100*(1-sample_symp_m/48387), inc_m, ".", alpha = 0.1)
ax3.plot([40,40],[0,1.2],"--b")
ax3.plot([20,20],[0,1.2],"--b")
ax3.set_xlim([0,100])
ax3.set_ylim([0,0.2])
ax3.set_ylabel("Incidence")
```

```
ax4 = fig.add_subplot(424)
ax4.plot(100*(1-sample_symp_f/88101), inc_f, ".r", alpha = 0.1)
ax4.plot([55,55],[0,1.2],"--r")
ax4.plot([30,30],[0,1.2],"--r")
ax4.set_xlim([0,100])
ax4.set_ylim([0,0.2])

ax5 = fig.add_subplot(425)
ax5.plot(100*(1-sample_symp_m/48387), scr_m, ".", alpha = 0.1)
ax5.plot([40,40],[0,1],"--b")
ax5.plot([20,20],[0,1],"--b")
ax5.set_xlim([0,100])
ax5.set_ylim([0,0.5])
ax5.set_ylabel("Screening")

ax6 = fig.add_subplot(426)
ax6.plot(100*(1-sample_symp_f/88101), scr_f, ".r", alpha = 0.1)
ax6.plot([55,55],[0,1],"--r")
ax6.plot([30,30],[0,1],"--r")
ax6.set_xlim([0,100])
ax6.set_ylim([0,0.5])

ax7 = fig.add_subplot(427)
ax7.plot(100*(1-sample_symp_m/48387), 1 - p_symp_m, ".", alpha = 0.1)
ax7.plot([40,40],[0,1],"--b")
ax7.plot([20,20],[0,1],"--b")
ax7.plot([0,100],[0.76,0.76],"--b")
ax7.plot([0,100],[0.26,0.26],"--b")
ax7.set_xlim([0,100])
ax7.set_ylim([0,1])
ax7.set_xlabel("Proportion of diagnoses asymptomatic (%)")
ax7.set_ylabel("Proportion of incident infections asymptomatic")

ax8 = fig.add_subplot(428)
ax8.plot(100*(1-sample_symp_f/88101), 1 - p_symp_f, ".r", alpha = 0.1)
ax8.plot([55,55],[0,1],"--r")
ax8.plot([30,30],[0,1],"--r")
ax8.plot([0,100],[0.75,0.75],"--r")
ax8.plot([0,100],[0.47,0.47],"--r")
ax8.set_xlim([0,100])
ax8.set_ylim([0,1])
ax8.set_xlabel("Proportion of diagnoses asymptomatic (%)")
```

Out[23]: <matplotlib.text.Text at 0x1113aed10>

The dashed lines are intended as a guide to the eye, to indicate scenarios roughly compatible with the Natsal-3 prevalence estimates. The observed chlamydia prevalence in Natsal-3 would be consistent with around 60-80% of diagnoses in men and 45-70% in women being symptomatic.

In [24]:
```
# Figure 6
# plot top pair only, for figure in paper

fig = plt.figure(figsize = (10,3))

xtk_m = [0, 10000, 20000, 30000, 40000] # x-axis ticks for men
xtk_f = [0, 20000, 40000, 60000, 80000] # x-axis ticks for women

ax1 = fig.add_subplot(121)
ax1.plot(100*(1-sample_symp_m/48387), prev_m, '.', alpha = 0.1)
ax1.fill_between([0,50000], 0.015, 0.034, facecolor='b', alpha=0.3)
ax1.plot([40,40],[0,1],'--b')
ax1.plot([20,20],[0,1],'--b')
ax1.set_xlim([0,100])
ax1.set_ylim([0,0.1])
ax1.set_xlabel('Proportion of diagnoses asymptomatic (%)')
ax1.set_ylabel('Prevalence')
ax1.set_title('Sexually active men, 15-24 years')
```

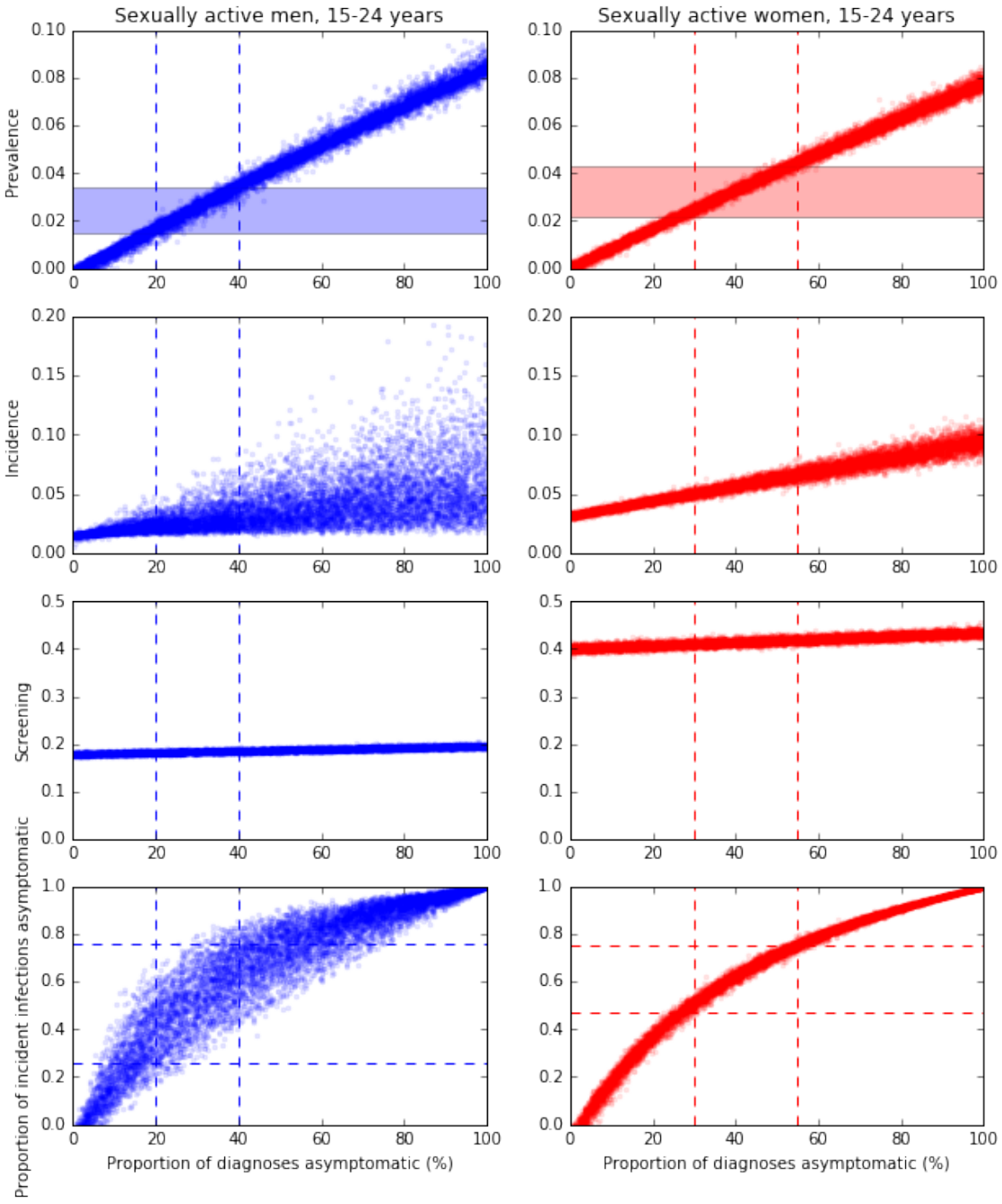

Figure 5: Samples for prevalence, incidence, screening rate and proportion of infections which are symptomatic, assuming different proportions of diagnoses made as a result of symptomms. The dashed lines are intended as a guide to the eye, to indicate scenarios roughly compatible with the Natsal-3 prevalence estimates.

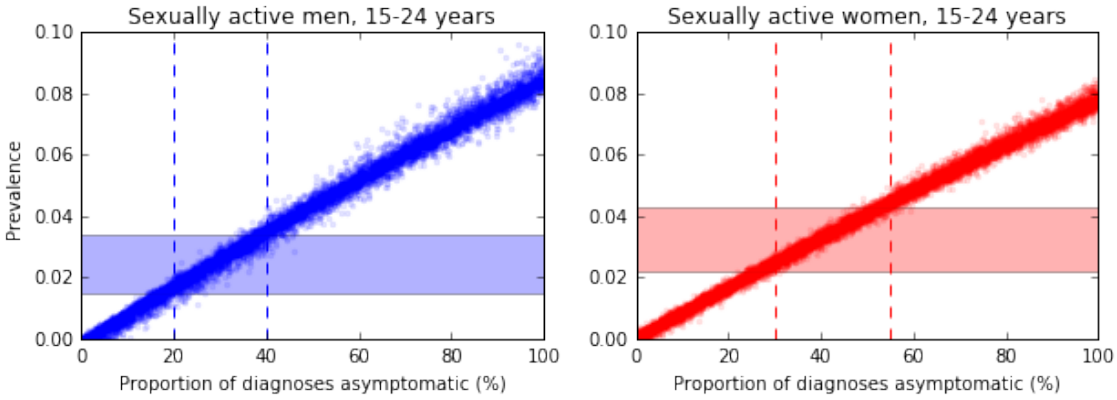

Figure 6: The upper two panels from the previous figure.

```
ax2 = fig.add_subplot(122)
ax2.plot(100*(1-sample_symp_f/88101), prev_f, '.r', alpha = 0.1)
ax2.fill_between([0,100000], 0.022, 0.043, facecolor='r', alpha=0.3)
ax2.plot([55,55],[0,1],'--r')
ax2.plot([30,30],[0,1],'--r')
ax2.set_xlim([0,100])
ax2.set_ylim([0,0.1])
ax2.set_xlabel('Proportion of diagnoses asymptomatic (%)')
ax2.set_title('Sexually active women, 15-24 years')
```

```
Out[24]: <matplotlib.text.Text at 0x10fd74910>
```

```
In [ ]:
```