Figure generation for

"A single pass approach to reducing sampling variation, removing errors, and scaling *de novo* assembly of shotgun sequences"

See http://arxiv.org/abs/1203.4802 for more information.

In [1]:
```
import numpy
datadir = '../data/'
num_points_to_plot = 50000  # affects size, rendering t
```

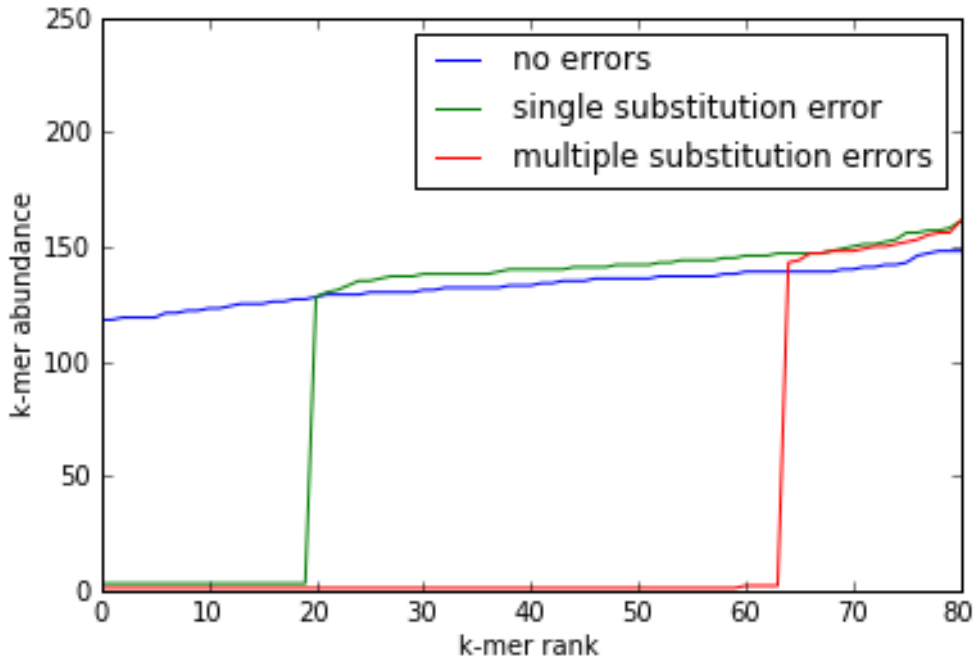
Figure 1

k-mer rank abundance plots

In [2]:
```
rr1 = [ x.split() for x in open(datadir + 'read2.ranks'
rr1 = numpy.array([ (int(a), int(b)) for (a,b) in rr1 ]
rr1_x = [ a for (a, b) in rr1 ]
rr1_y = [ b for (a, b) in rr1 ]

rr2 = [ x.split() for x in open(datadir + 'read3.ranks'
rr2 = numpy.array([ (int(a), int(b)) for (a,b) in rr2 ]
rr2_x = [ a for (a, b) in rr2 ]
rr2_y = [ b for (a, b) in rr2 ]

rr3 = [ x.split() for x in open(datadir + 'read15.ranks
rr3 = numpy.array([ (int(a), int(b)) for (a,b) in rr3 ]
rr3_x = [ a for (a, b) in rr3 ]
rr3_y = [ b for (a, b) in rr3 ]
```

In [3]:
```
clf()
plot(rr1[:,0], rr1[:,1])
plot(rr2[:,0], rr2[:,1])
plot(rr3[:,0], rr3[:,1])
xlabel('k-mer rank')
ylabel('k-mer abundance')
legend(['no errors', 'single substitution error', 'mult
```

```
axis([0, 80, 0, 250])
savefig('diginorm-ranks.pdf')
show()
```



Figures 2 and 3

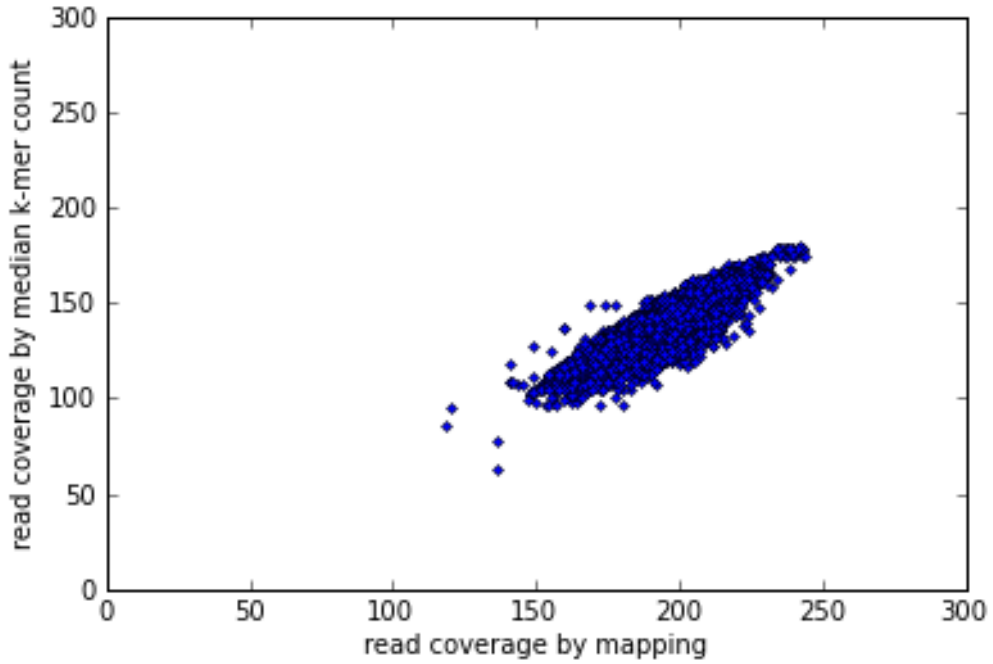
Correlation between median k-mer count and read coverage calculated from mapping, from both simulated data and genomic & transcriptomic data.

In [4]:
```
import itertools, gzip
def load_cmp_file(filename, limit=None):
    if filename.endswith('.gz'):
        fp = gzip.open(filename)
    else:
        fp = open(filename)
    if limit:
        lines = [ line.split() for i, line in itertools
    else:
        lines = [ line.split() for line in fp ]
    lines = [ (float(a[0]), float(a[1])) for a in lines

    print 'loaded %d lines' % len(lines)
    return numpy.array(lines)
```

In [5]:
```
genome_counts = load_cmp_file(datadir + 'genome-reads.fa
```

```
loaded 50000 lines
```

In [6]:
```
clf()
axis([0, 300, 0, 300])
#axis(ymax=1000)

xlabel('read coverage by mapping')
ylabel('read coverage by median k-mer count')
plot(genome_counts[:,0], genome_counts[:,1], 'b.', raste

savefig('diginorm-sim-genome.pdf')
show()

c = numpy.corrcoef(genome_counts[:,0], genome_counts[:,
print c**2
```



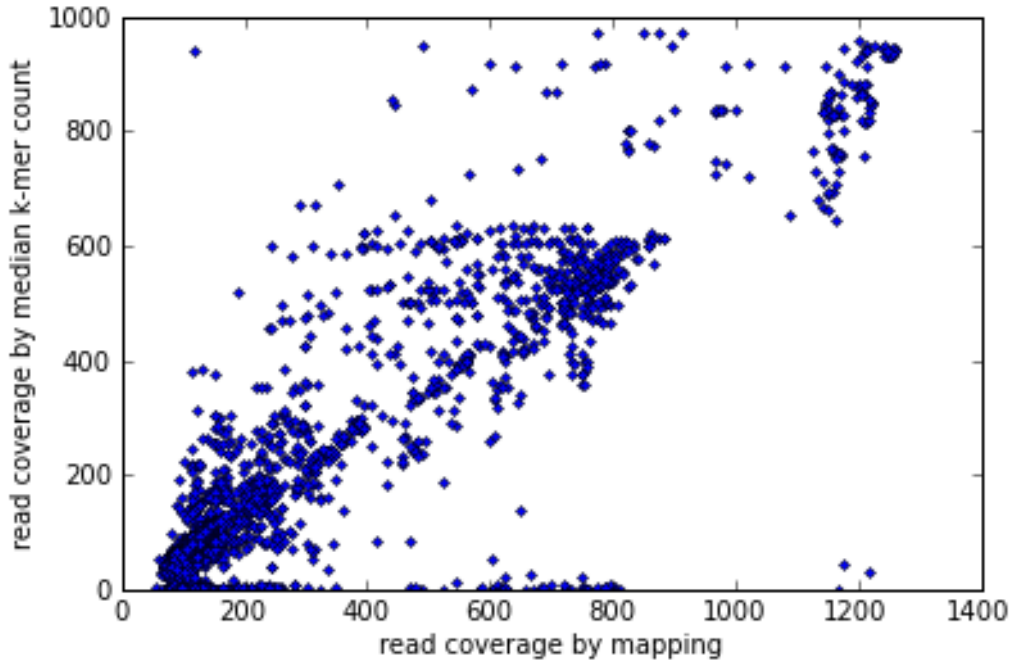

```
0.786287931714
```

In [7]:
```
ecoli_counts = load_cmp_file(datadir + 'ecoli_ref-5m.fas

# remove points from Illumina crap; see text for detail
ecoli_counts = numpy.array([ (a,b) for (a,b) in ecoli_c
```

```
loaded 50000 lines
```

In [8]:
```
clf()
#axis([0, 300, 0, 300])
```

```
#axis(xmax=200, ymax=200)
#axis(ymax=10500,ymin=10000)
#title('ecoli reads')
xlabel('read coverage by mapping')
ylabel('read coverage by median k-mer count')
plot(ecoli_counts[:,0], ecoli_counts[:,1], 'b.', raster
savefig('diginorm-ecoli-genome.pdf')
show()

c = numpy.corrcoef(ecoli_counts[:,0], ecoli_counts[:,1]
print c**2
```



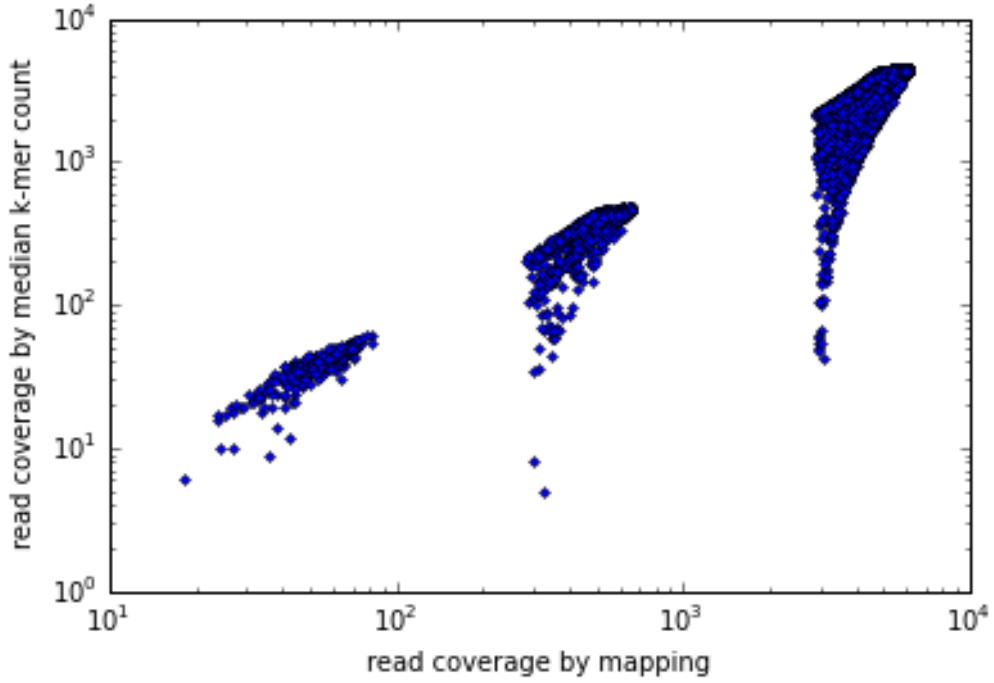

```
0.798609411012
```

In [9]:
```
transcript_counts = load_cmp_file(datadir + 'transcript

# remove points from Illumina crap; see text for detail
transcript_counts = numpy.array([ (a,b) for (a,b) in tra
```

```
loaded 50000 lines
```

In [10]:
```
clf()
#axis([0, 250, 0, 250])
#title('transcript reads')
xlabel('read coverage by mapping')
ylabel('read coverage by median k-mer count')
loglog(transcript_counts[:,0], transcript_counts[:,1],

savefig('diginorm-sim-transcr.pdf')
show()
```

```
c = numpy.corrcoef(transcript_counts[:,0], transcript_c
print 'linear r^2', c**2

c = numpy.corrcoef([ math.log(z) for z in transcript_co
print 'log r^2', c**2
```



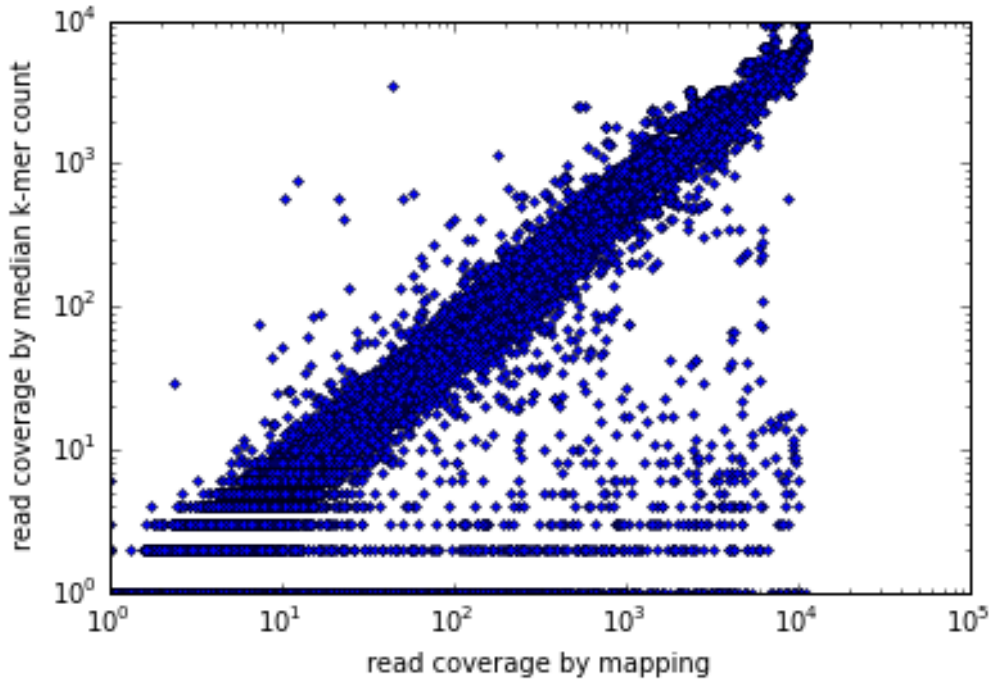

```
linear r^2 0.931638777367
log r^2 0.956816045607
```

In [11]:
```
mouse_counts = load_cmp_file(datadir + 'mouse-5m.fq.cou

# remove points from Illumina crap; see text for detail
mouse_counts = numpy.array([ (a,b) for (a,b) in mouse_c
```

```
loaded 50000 lines
```

In [12]:
```
clf()
#axis([0, 250, 0, 250])
#title('mouse reads')
xlabel('read coverage by mapping')
ylabel('read coverage by median k-mer count')
loglog(mouse_counts[:,0], mouse_counts[:,1], 'b.', raste

savefig('diginorm-mouse-transcr.pdf')
show()

c = numpy.corrcoef(mouse_counts[:,0], mouse_counts[:,1]
print 'linear r^2', c**2
```

```
za = []
zb = []
for a, b in mouse_counts:
    if a and b:
        za.append(math.log(a))
        zb.append(math.log(b))

c = numpy.corrcoef(za, zb)[0,1]
print 'log r^2', c**2
```




```
linear r^2 0.904776248799
log r^2 0.872367665664
```

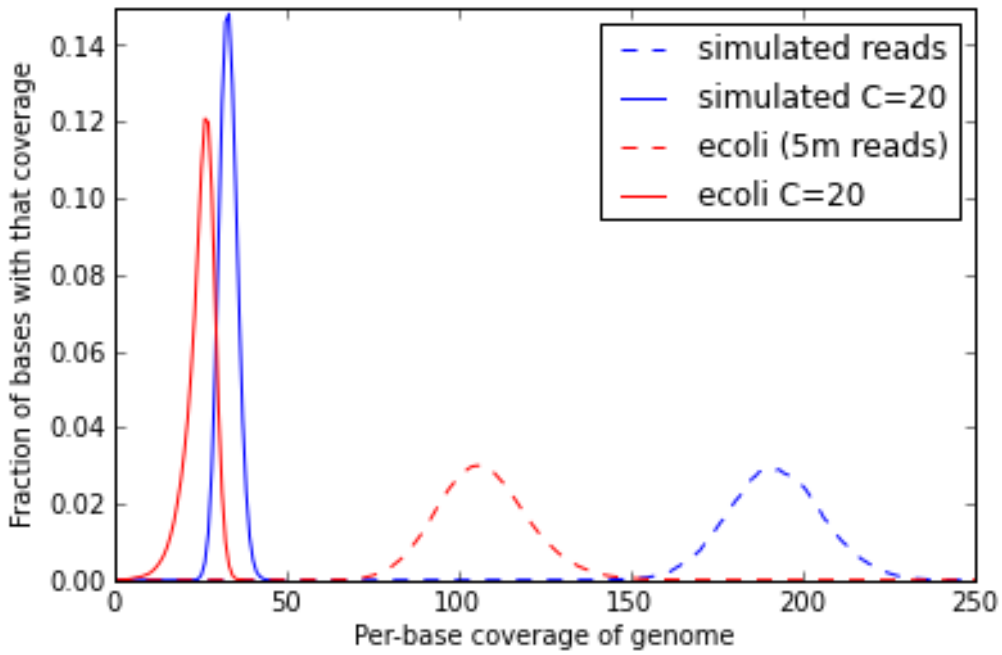
Figure 4

In [13]:
```
mapcov = numpy.loadtxt(datadir + 'genome-reads.fa.map.c
keepcov = numpy.loadtxt(datadir + 'genome-reads.fa.keep
ecolicov = numpy.loadtxt(datadir + 'ecoli_ref-5m.fastq.
ecoli20cov = numpy.loadtxt(datadir + 'ecoli_ref.fastq.b

transcript_cov = numpy.loadtxt(datadir + 'transcript-re
mousecov = numpy.loadtxt(datadir + 'mouse-5m.fq.map.cov

transcript_keepcov = numpy.loadtxt(datadir + 'transcrip
```

```
mousekeepcov = numpy.loadtxt(datadir + 'mouse-5m.fq.kee
```

In [14]:
```
mapcov[:,1] /= sum(mapcov[:,1])
keepcov[:,1] /= sum(keepcov[:,1])
ecolicov[:,1] /= sum(ecolicov[:,1])
ecoli20cov[:,1] /= sum(ecoli20cov[:,1])

transcript_cov[:,1] /= sum(transcript_cov[:,1])
mousecov[:,1] /= sum(mousecov[:,1])

transcript_keepcov[:,1] /= sum(transcript_keepcov[:,1])
mousekeepcov[:,1] /= sum(mousekeepcov[:,1])
```

In [15]:
```
clf()
xlabel("Per-base coverage of genome")
ylabel("Fraction of bases with that coverage")
axis([0, 250, 0, .15])


plot(mapcov[:,0], mapcov[:,1], 'b--')
plot(keepcov[:,0], keepcov[:,1], 'b-')
plot(ecolicov[:,0], ecolicov[:,1], 'r--')
plot(ecoli20cov[:,0], ecoli20cov[:,1], 'r-')
legend([ 'simulated reads', 'simulated C=20', 'ecoli (5
savefig('diginorm-coverage.pdf')
show()
```



In [16]:
```
# not actually in the paper :)
```

In [16]:
```
# not actually in the paper :)

clf()
xlabel("Per-base coverage of transcriptome")
ylabel("Fraction of bases with that coverage")
#axis([0, 125, 0, .15])
#axis(ymin=-.05, xmax=14000, xmin=-100)
#axis(xmin=0, xmax=50)
axis(ymin=1e-8, ymax=50)

loglog(transcript_cov[:,0], transcript_cov[:,1], 'b--')
loglog(transcript_keepcov[:,0], transcript_keepcov[:,1]

loglog(mousecov[:,0], mousecov[:,1], 'r--')
loglog(mousekeepcov[:,0], mousekeepcov[:,1], 'r-')

legend([ 'simulated reads', 'simulated reads C=20', 'mou
savefig('diginorm-transcript-coverage.pdf')
show()
```

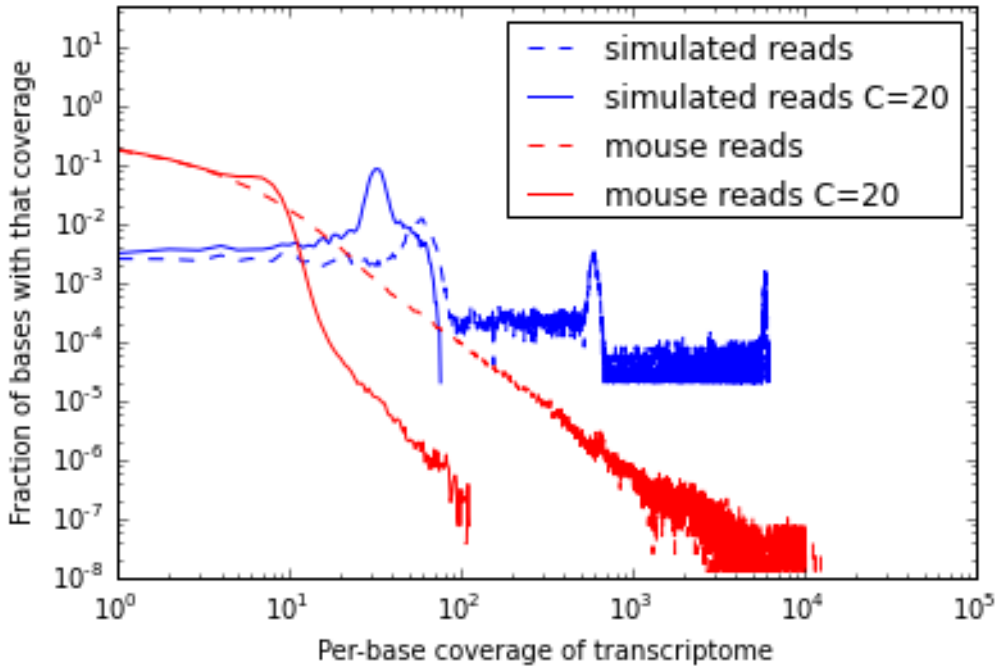


Figure 5

In [17]:
```
fp = open(datadir + 'ecoli_ref.report')
report = [ x.split()[:2] for x in fp ]
report = [ (float(a) / 1e6, int(b)) for (a,b) in report
```

```
report = [ (a, b, float(b)/float(a)/1e6) for (a,b) in r
print report[0]
```

```
(0.02, 19958, 0.9979)
```

In [18]:
```
x = [ a for (a,b,c) in report ]
y = [ c for (a,b,c) in report ]
```

In [19]:
```
clf()
xlabel("Number of reads (m)")
ylabel("Total fraction of reads kept")
axis([0, 30, 0, 1.0])
plot(x,y)
savefig('diginorm-accumulation.pdf')
```

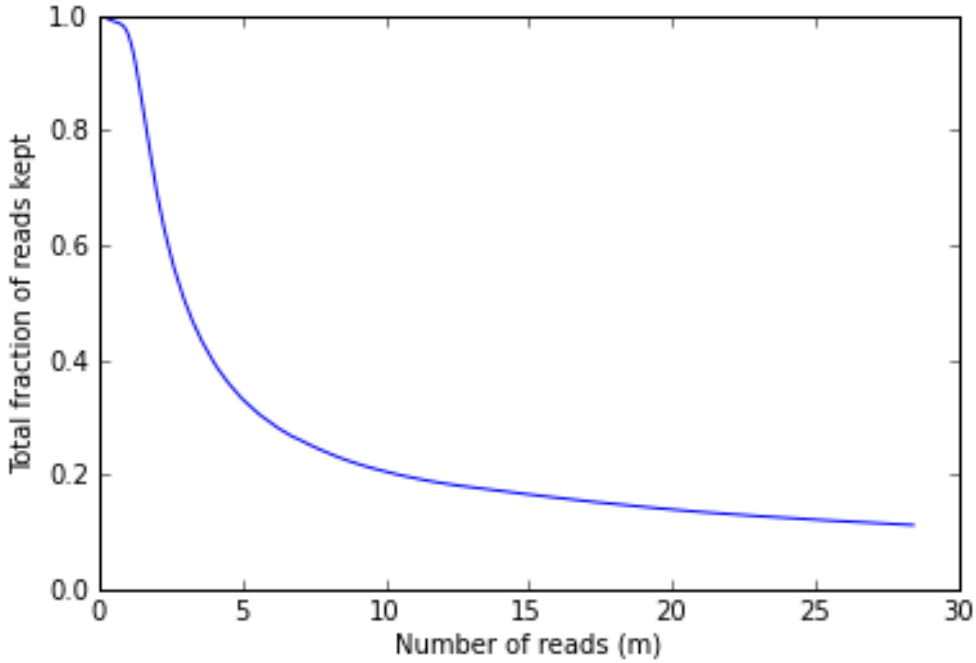


Figure 6 - end bias stuff

In [20]:
```
def load_endbias(filename):
    data = [ i.split() for i in open(filename) ]
    x = [ int(a) for (a,_,b,_) in data ]
    y = [ float(b) for (a,_,b,_) in data ]

    return (x, y)
```

```
end_tr_x, end_tr_y = load_endbias(datadir + 'endbias-tra
end_g_x, end_g_y = load_endbias(datadir + 'endbias-genor
```

In [21]:
```
clf()
xlabel("Distance from k-mer to end of source sequence")
ylabel("Fraction of positions with missing k-mers")
axis([-5,50,-.05, 1.1])
plot(end_tr_x, end_tr_y)
plot(end_g_x, end_g_y)
legend(['simulated transcripts', 'simulated genome'])
savefig('diginorm-endbias.pdf')
show()
```