Numerical methods in (non-hyperbolic) chaos

Part 2: Monte Carlo sampling

Caroline Wormell, Sorbonne Université/CNRS

How rigorous to be?

- As with paper calculations, there are different levels of rigour.
- They are all useful!
  - We regularly make mathematical hypotheses based on inductive (scientist-style) reasoning.

Suppose we use algorithm A to compute proposition X. We could have:

1. X is definitely, mathematically true (i.e. A constitutes a proof).

Example: The Lorenz flow is a Geometric Lorenz flow (Tucker 1999)

1. That A converges is a theorem, (1) would be true if we computed the (small) approximation errors explicitly.

Example: Running some proven-to-work approximation algorithm but not keeping track of the errors.

1. That A converges is a theorem, (1) would be true if we computed the (small) approximation errors explicitly.

Example: Running some proven-to-work approximation algorithm but not keeping track of the errors.

1. We have a good idea of how to prove A converges, (2) would be true if we did that.

Example: Minor extensions of existing algorithms. "What if we used a Lipschitz observable instead of C^1 like in the theorem"

1. A would converge if clearly true condition C holds, (2) or (3) would be true if we could prove C.

Example: Assuming a dynamical system that appears to be chaotic, exponentially mixing, etc, is actually those things

1. A would converge if clearly true condition C holds, (2) or (3) would be true if we could prove C.

Example: Assuming a dynamical system that appears to be chaotic, exponentially mixing, etc, is actually those things

1. (2) or (3) is true in an analogous setting, and would be true if we could extend it to our setting.

Example: Applying an algorithm proven for Anosov maps to a non-uniformly hyperbolic map

1. A would converge if clearly true condition C holds, (2) or (3) would be true if we could prove C.

Example: Assuming a dynamical system that appears to be chaotic, exponentially mixing, etc, is actually those things

1. (2) or (3) is true in an analogous setting, and would be true if we could extend it to our setting.

Example: Applying an algorithm proven for Anosov maps to a non-uniformly hyperbolic map

1. We have some formal calculation/intuition that A should compute X (usually plus some evidence in practice).

Example: Dynamic mode decomposition, etc

All of these are useful for both mathematicians and scientists!

Non-uniformly hyperbolic systems will almost always fall into cases 4–6.

All of these are useful for both mathematicians and scientists!

Non-uniformly hyperbolic systems will almost always fall into cases 4–6.

General exercise: find or recall examples of numerics that you have seen corresponding to cases 1–6.

Last lecture:

- Physical measures are important
- For most (ie non-structurally stable) systems it is hard or impossible to make a priori bounds

Estimating a physical measure is easiest done in a weak sense, i.e. by estimating integrals against bounded observables

$$\int_{\mathcal{M}} A \, \mathrm{d}\mu.$$

We could try doing this by computing a Birkhoff sum. But how can we know our estimates are correct?

Estimating a physical measure is easiest done in a weak sense, i.e. by estimating integrals against bounded observables

$$\int_{\mathcal{M}} A \, \mathrm{d}\mu.$$

We could try doing this by computing a Birkhoff sum. But how can we know our estimates are correct?

Meta-theorem (truth level 4): For regular enough functions $A : M \to \mathbb{R}$ and n large enough, $A(x)$ and $A(f^n(x))$ are close to being uncorrelated.

The consequence is that a lot of properties that are true of *i.i.d.* random variables also hold for chaotic signals. We can use this to our advantage…

Monte Carlo estimation: *i.i.d.* case

Suppose we have probability measure $\mu \in \mathcal{P}(\mathcal{M})$, "observable" function $A \in L^1(\mathcal{M}, \mathbb{R})$.

We are given ind. samples $A(x_1), A(x_2), \ldots, A(x_M) \sim \mu$.

We want to estimate $\int_{\mathcal{M}} A \, \mathrm{d}\mu$.

Monte Carlo estimation: *i.i.d.* case

Suppose we have probability measure $\mu \in \mathcal{P}(\mathcal{M})$, "observable" function $A \in L^1(\mathcal{M}, \mathbb{R})$.

We are given ind. samples $A(x_1), A(x_2), \ldots, A(x_M) \sim \mu$.

We want to estimate $\int_{\mathcal{M}} A \, \mathrm{d}\mu$.

Theorem (Strong Law of Large Numbers): With probability 1,

$$\bar{A}_M := \frac{1}{M} \sum_{m=1}^{M} A(x_m) \to \int_{\mathcal{M}} A \, \mathrm{d}\mu$$

as $M \to \infty$.

So, we can estimate the average by taking a really large sample:

So, we can estimate the average by taking a really large sample:

In [384]:

```
A(x) = x^(-0.9)

using QuadGK
expectationA = quadgk(A,0,1)[1] # true expectation of A
```

Out[384]:

```
9.999999279144092
```

So, we can estimate the average by taking a really large sample:

In [384]:

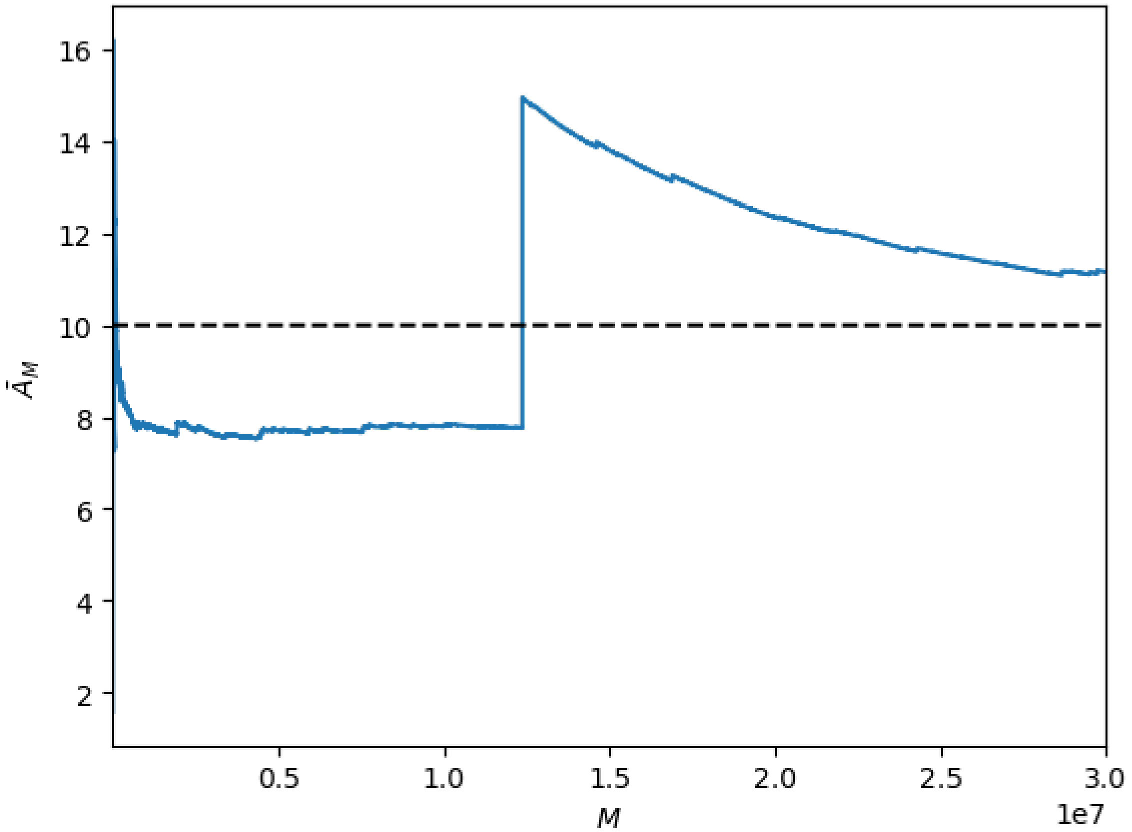
```
A(x) = x^(-0.9)

using QuadGK
expectationA = quadgk(A,0,1)[1] # true expectation of A
```

Out[384]:

```
9.999999279144092
```

In [387]:

```
Mmax = 30000000

sample = rand(Mmax) # μ is uniform on [0,1]

plot(1:Mmax, cummean(A.(sample[1:Mmax])))
xlabel("\$M\$"); xlim(1,Mmax)
ylabel("\$\\bar{A}_M\$")

plot(1:Mmax,fill(expectationA,Mmax),"k--");
```



16
14
12
10
8
6
4
2
$\bar{A}_M$
0.5
1.0
1.5
2.0
2.5
3.0
1e7
M

So: we want some quantitative convergence estimates!

Convergence rates

The SLLN says:

"If an expectation of A exists, then sample means of A will converge."

This is as general as possible, and completely qualitative: there are L^1 functions A for which the sample means converge arbitrarily slowly.

To get quantitative convergence rates, we will need *quantitative assumptions* on A (in particularly, on its tails).

Let's make a strong quantitative assumption on the tails of A: A is bounded (or $A \in L^\infty(\mu)$).

Let's make a strong quantitative assumption on the tails of A: A is bounded (or $A \in L^\infty(\mu)$).

Theorem (concentration bound): Suppose A is a bounded random variable. In particular suppose $|A - \mathbb{E}[A]| \leq \alpha$ and $\mathbb{V}[A] \leq \sigma^2$. Then

$$\mathbb{P}[\left|\bar{A}_M - \mathbb{E}[A]\right| > w] \leq 2\exp\left(-\frac{w^2}{2\sigma^2/M}(2 - e^{w\alpha/\sigma^2})\right)$$

as $M \to \infty$.

This says that except at the tails ($\bar{A}_M \geqq \sigma^2/\alpha$) then $\bar{A}_M$ has exponential decay like a normal random variable of standard deviation $\sigma/\sqrt{M}$.

Note also that $\mathbb{V}[A] \leq \alpha$.

Example: μ is uniform on $[0,1]$, $A = \sin(100/x)$. (So $\sigma \leq \alpha \leq 2$.)

We can therefore estimate

In [389]:

```
using Statistics
M = 10000; A(x) = sin(100/x)

sample_means = Array{Float64}(undef,100)
for i = 1:100 # generate 100 sample means
    x_vector = rand(M) # generate M sample points
    sample_means[i] = mean(A.(x_vector))
end
```

In [392]:

```
using Roots
a = 2; σ = true_σ
w = fzero(w->log(2exp(-w^2/(2σ^2/M)*(2-exp(w*a/σ^2)))/0.1),1/sqrt(M))
```

Out[392]:

```
0.01795346111089137
```

In [392]:

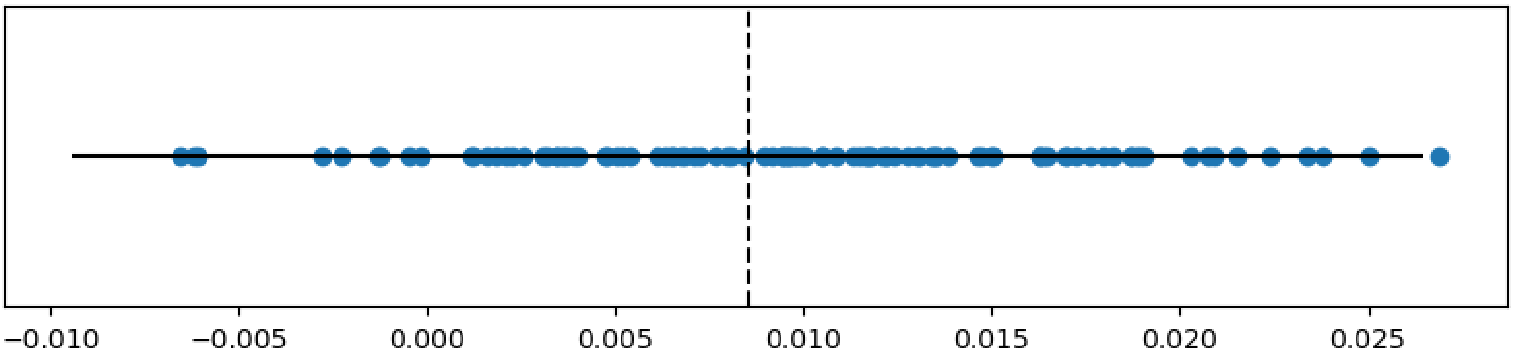
```
using Roots
a = 2; σ = true_σ
w = fzero(w->log(2exp(-w^2/(2σ^2/M)*(2-exp(w*a/σ^2)))/0.1),1/sqrt(M))
```

Out[392]:

```
0.01795346111089137
```

In [393]:

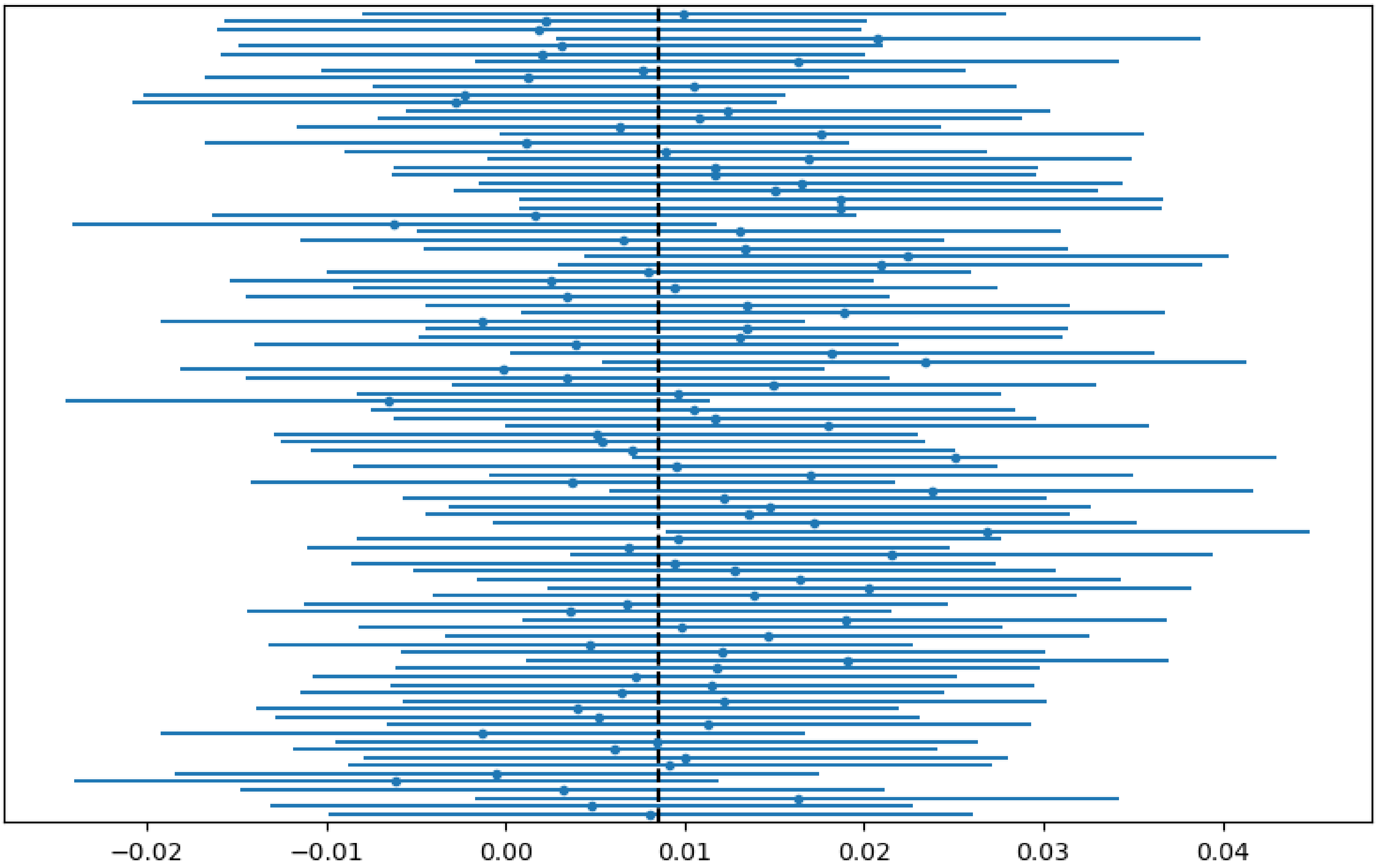
```
figure(figsize=(10,2))
scatter(sample_means,zeros(100))
errorbar([true_Amean],[0],xerr=[w],c="k")
plot(fill(true_Amean,2),[-1,1],"k--"); ylim(-1,1)
yticks([]);
```



Our theorem tells us that $\bar{A}_M$ are w close to $\mathbb{E}[A]$ almost always, but we can swap which of the two we put the error bound on:

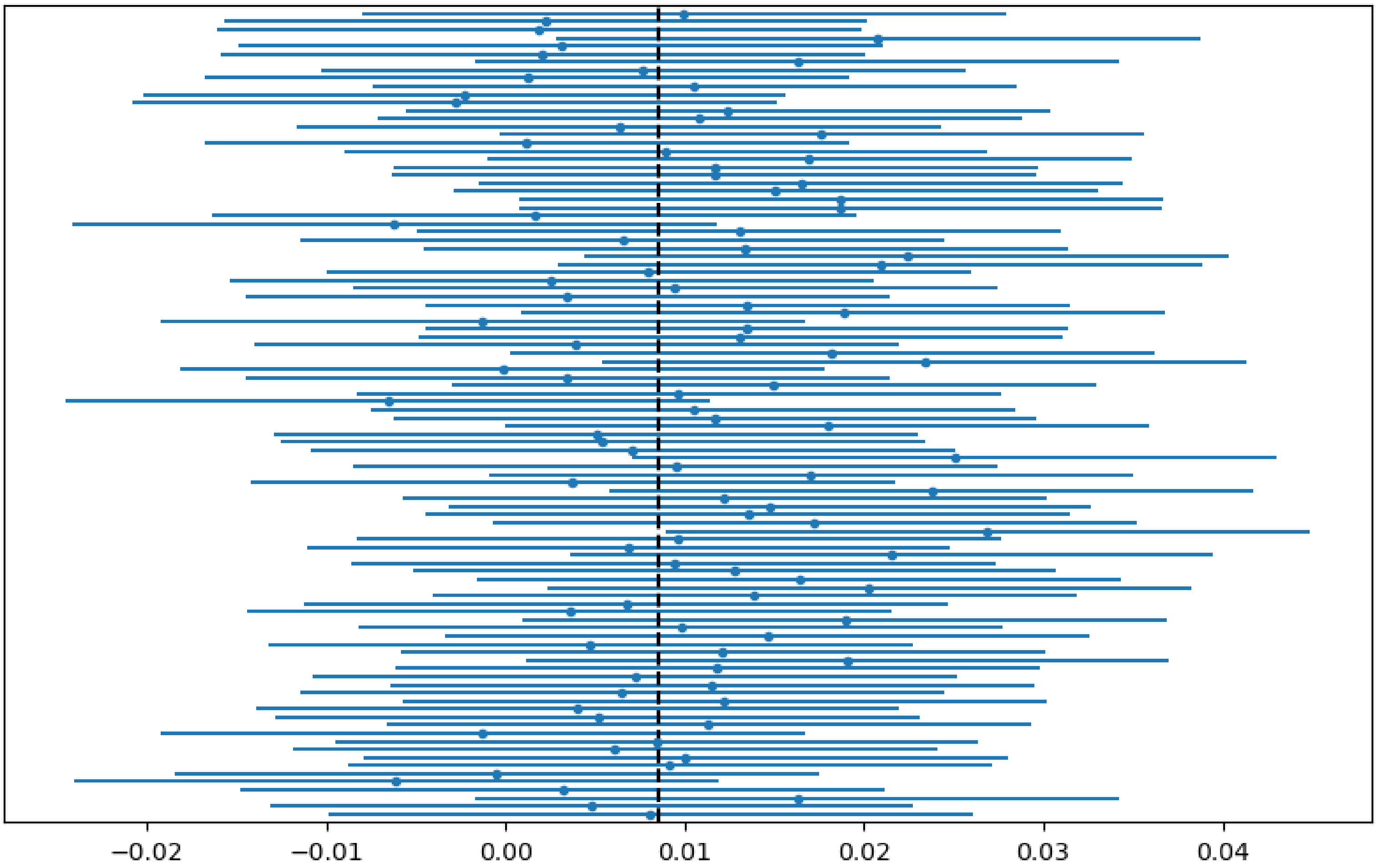
In [394]:

```
figure(figsize=(10,6))
errorbar(sample_means,1:100,xerr=w,linestyle="",marker=".")
plot(fill(true_Amean,2),[0,101],"k--"); ylim(0,101)
yticks([]);
```



In [394]:

```
figure(figsize=(10,6))
errorbar(sample_means,1:100,xerr=w,linestyle="",marker=".")
plot(fill(true_Amean,2),[0,101],"k--"); ylim(0,101)
yticks([]);
```



We are only asking for the true mean to lie inside the sample mean 90% of the time. So our error bars are a bit wide.

Central Limit Theorem

The central limit theorem gives us asymptotically the correct bounds:

Theorem (CLT): Suppose A has bounded variance σ^2. Then for all $\theta \in \mathbb{R}$,

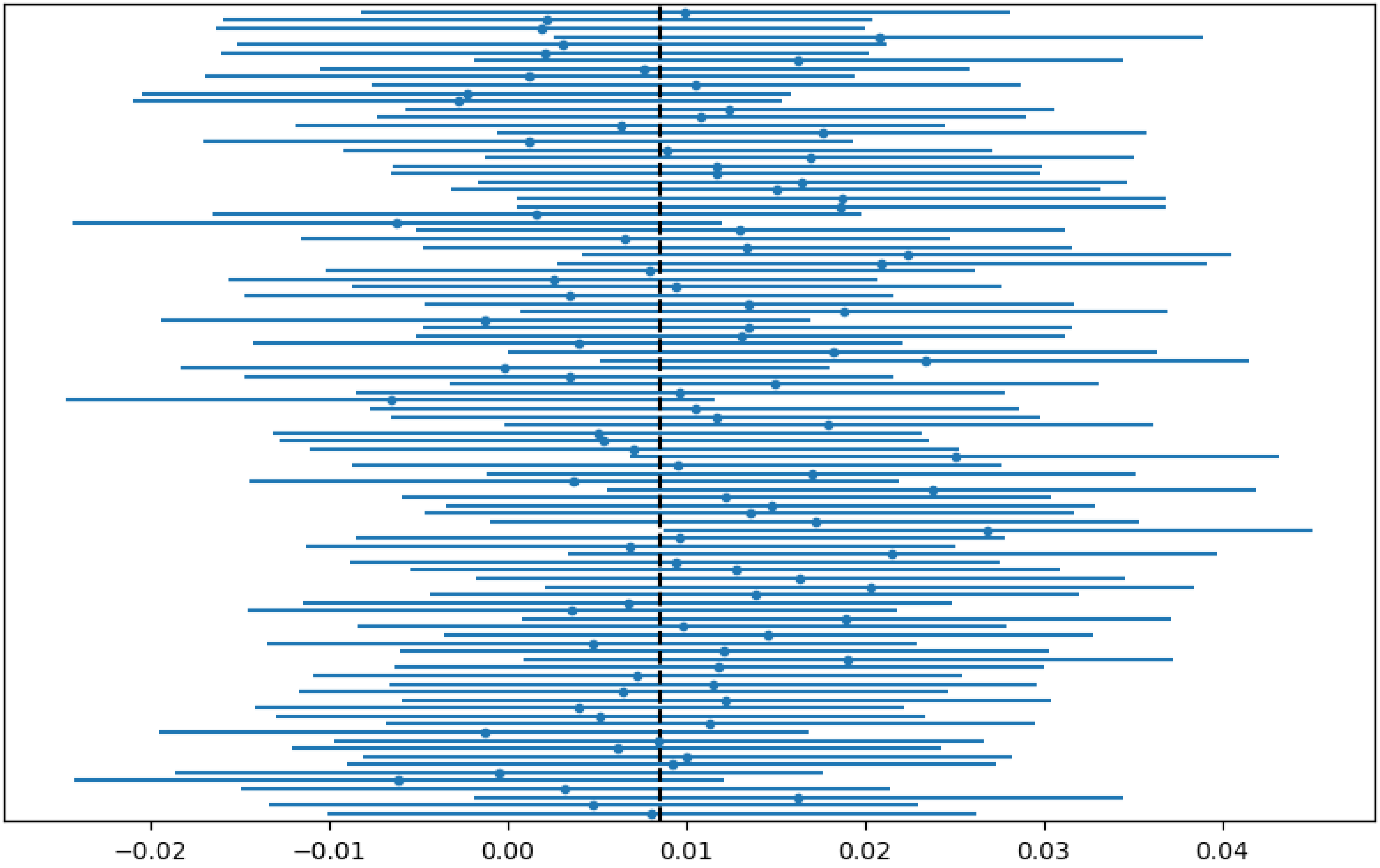
$$\lim_{M \to \infty} \mathbb{P}\left[|\bar{A}_M - \mathbb{E}[A]| > \frac{\theta}{\sqrt{M}}\right] = 2\mathbb{P}(\mathcal{N}(0, \sigma^2) > \theta)$$

Again, this is qualitative. There are ways to make it tighter, but statisticians have some guidelines for when it's reasonable to use in statistical tests:

- M is sufficiently large (≥ 20 for unimodal data with short tails), or
- the distribution of $A(x)$ already approximates a Gaussian

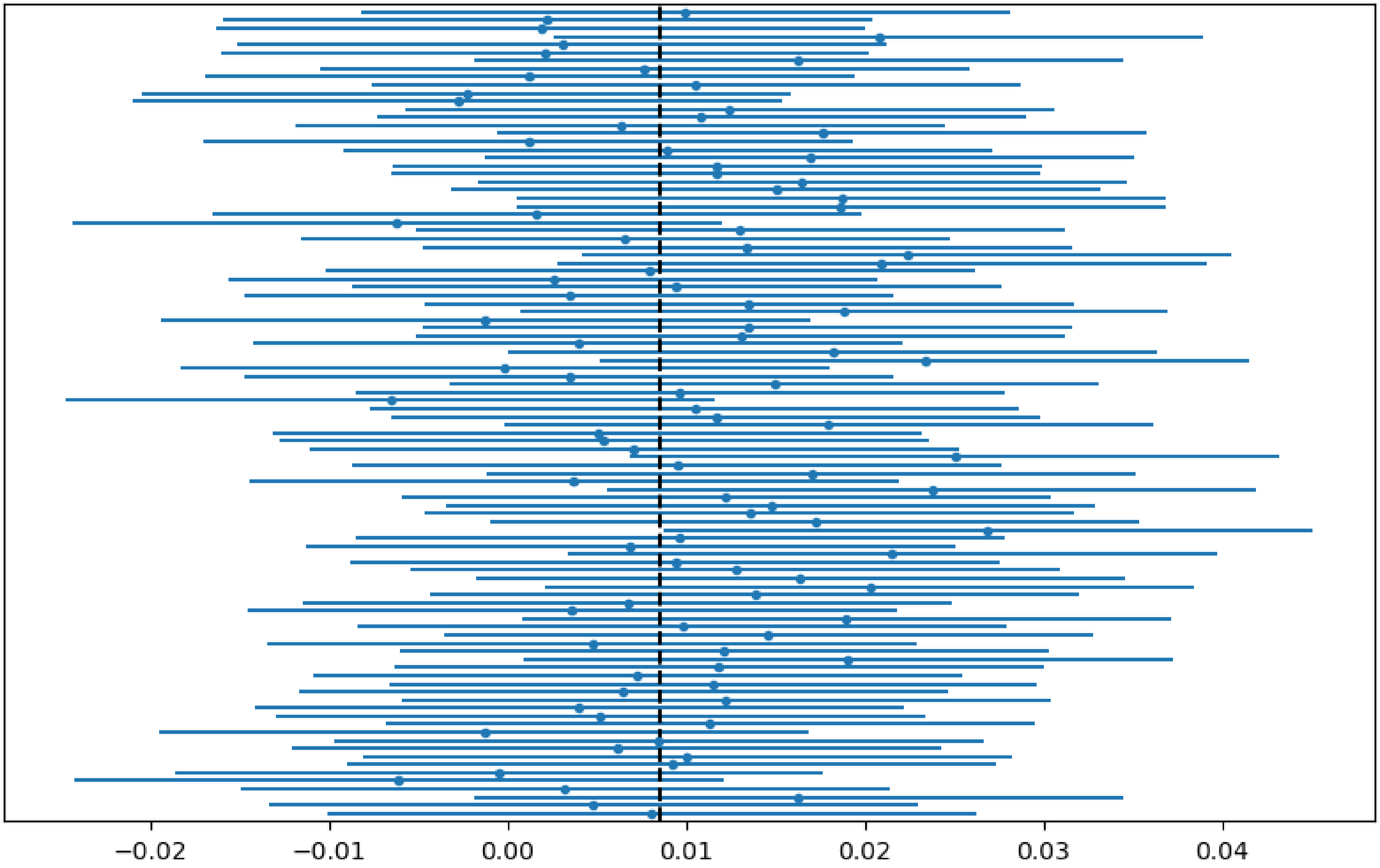
In [397]:

```
using Distributions
w = cquantile(Normal(0,true_σ),0.01/2)/sqrt(M)

figure(figsize=(10,6))
errorbar(sample_means,1:100,xerr=w,linestyle="",marker=".")
plot(fill(true_Amean,2),[0,101],"k--"); ylim(0,101)
yticks([]);
```



In [397]:

```
using Distributions
w = cquantile(Normal(0,true_σ),0.01/2)/sqrt(M)

figure(figsize=(10,6))
errorbar(sample_means,1:100,xerr=w,linestyle="",marker=".")
plot(fill(true_Amean,2),[0,101],"k--"); ylim(0,101)
yticks([]);
```



OK, but we don't expect to know the true variance either. How to estimate?

Basic estimator for $\mathbb{V}(A)$ (implemented in `std` etc):

$$s_M^2 = \frac{1}{M-1}\sum_{m=1}^{M}(A(x_m) - \bar{A}_M)^2$$

Basic estimator for $\mathbb{V}(A)$ (implemented in `std` etc):

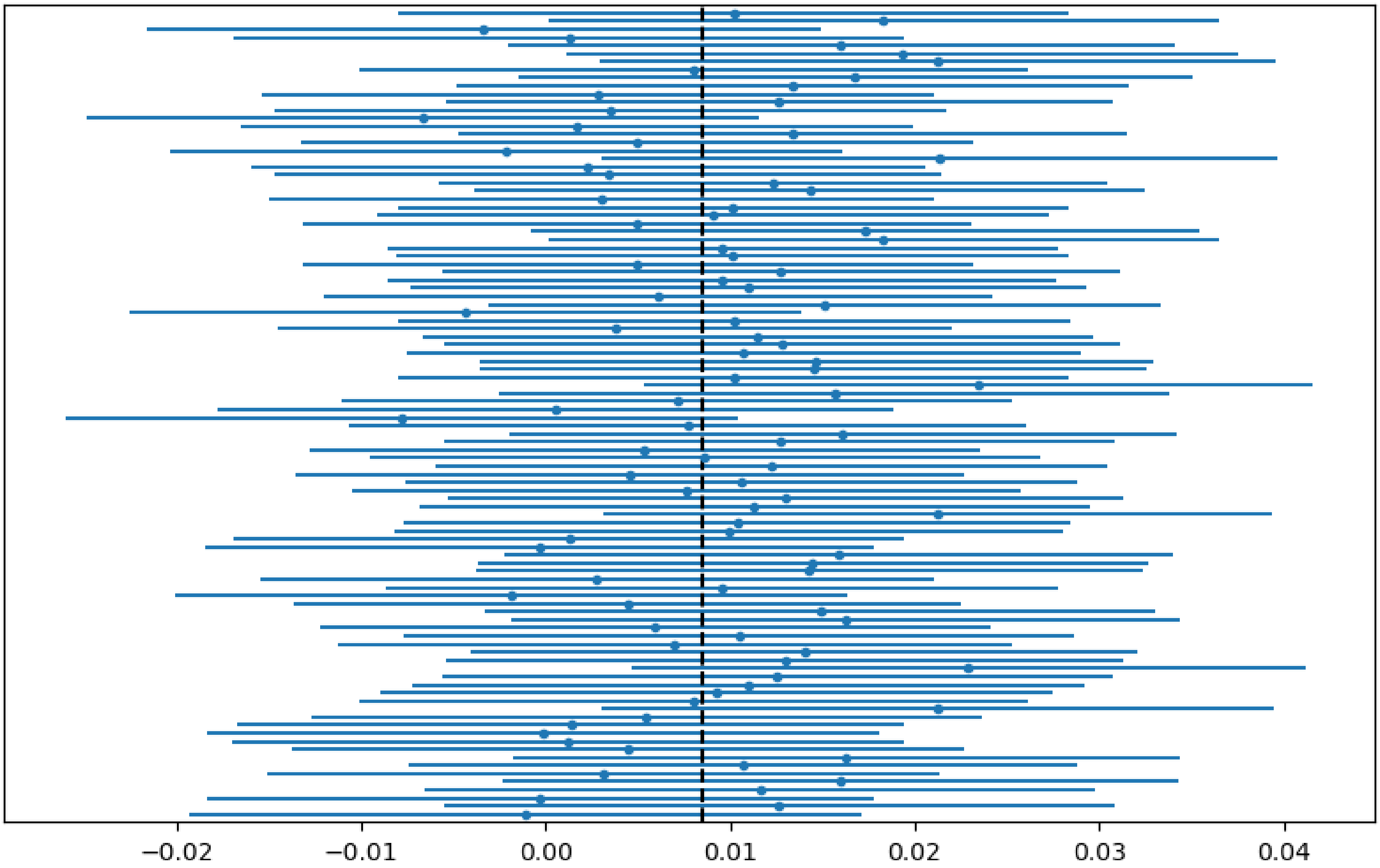
$$s_M^2 = \frac{1}{M-1} \sum_{m=1}^{M} (A(x_m) - \bar{A}_M)^2$$

If $A(x_m)$ are Gaussian, then

$$\sqrt{M} \frac{\bar{A}_M - \mathbb{E}[A]}{s_M} \sim t_{M-1}$$

In [399]:

```
sample_means = Array{Float64}(undef,100)
sample_stds = Array{Float64}(undef,100)
for i = 1:100 # generate 100 sample means
    x_vector = rand(M) # generate M sample points
    sample_means[i] = mean(A.(x_vector))
    sample_stds[i] = std(A.(x_vector))
end

tconst = cquantile(TDist(M-1),0.01/2)/sqrt(M)

figure(figsize=(10,6))
errorbar(sample_means,1:100,xerr=tconst*sample_stds,linestyle="",marker=".")
plot(fill(true_Amean,2),[0,101],"k--"); ylim(0,101)
yticks([]);
```


−0.02
−0.01
0.00
0.01
0.02
0.03
0.04

Estimating physical measures

We want to find a variable whose expectation is $\int_M A \,\mathrm{d}\rho$, but we can't always directly access ρ.

Since

$$B_N(x) = \frac{1}{N} \sum_{n=0}^{N-1} A(f^n(x)),\ x \sim \mu$$

is bounded and converges to $\int_M A \,\mathrm{d}\rho$ almost surely as $N \to \infty$ if $\mathrm{d}\mu = h \,\mathrm{d}x$, we could try to use that.

i.e.

$$\sqrt{M}(\overline{(B_N)}_M - \mathbb{E}[B_N]) \to \mathbb{N}(0, \mathbb{V}[B_N])$$

But we don't actually know much about $\mathbb{E}[B_N]$:

- Almost sure convergence is not the same as convergence in expectation
- Convergence rates?

Need an "obviously true" condition...

Exponential decay of correlations (from Lebesgue)

We need some **"obviously" true condition C**:

Suppose A is sufficiently regular and μ is sufficiently nice (e.g. absolutely continuous with respect to Lebesgue). Then there exist $p \in \mathbb{N}_+$ and $\lambda_2 < 1$ independent of A, μ,

$$\int_M A \circ f^n \,\mathrm{d}\mu = \int_M A \,\mathrm{d}\rho \int_M \mathrm{d}\mu + \sum_{q=1}^{p-1} c_q \xi^{qn} + Q_n$$

where $\xi = e^{2\pi i/p}$ and $|Q_n| < C\lambda_2^n$ (C could be very large).

We can then estimate:

$$\mathbb{E}[B_N] = \int_M \frac{1}{N}\sum_{n=0}^{N-1} A(f^n(x))\,\mathrm{d}\mu = \int_M A\,\mathrm{d}\rho + \frac{1}{N}\sum_{q=1}^{p-1} c_q \frac{1-\xi^{qN}}{1-\xi^q} + \frac{1}{N}\sum_{n=0}^{N-1} Q_n$$

$$= \int_M A\,\mathrm{d}\rho + \mathcal{O}(\tfrac{1}{N})$$

In [400]:

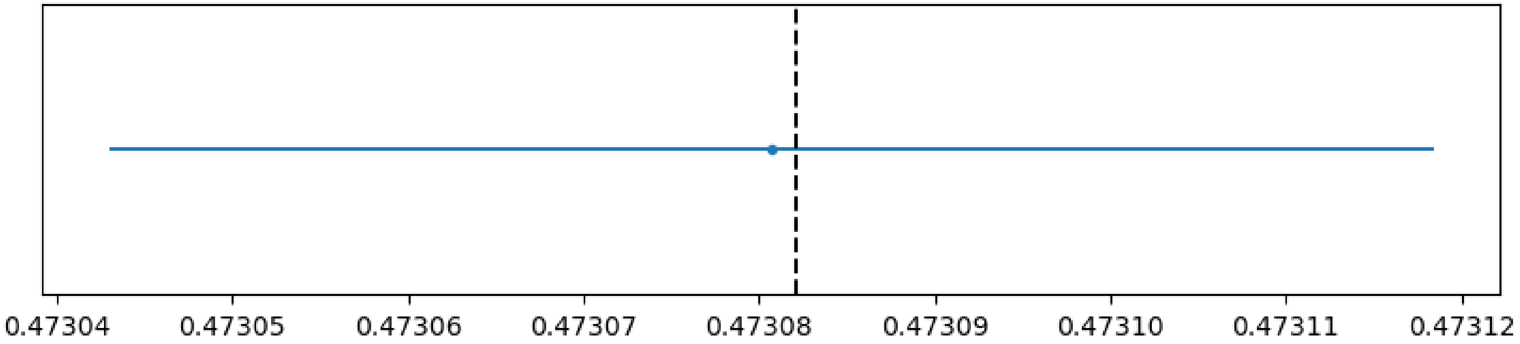
```
function birkhoff_mean( f, # map
                        A, # observation function
                        N; # time series length
                        x0=rand()) # initial value

    x = x0

    birkhoff_sum = 0.
    for n = 1:N
        birkhoff_sum += A(x)
        x = f(x)
    end
    birkhoff_sum / N # mean
end
```

Out[400]:

```
birkhoff_mean (generic function with 1 method)
```

In [405]:

```
M = 200; N = 30000
means_of_birkhoff_means = Array{Float64}(undef,100)
stds_of_birkhoff_means = Array{Float64}(undef,100)
f(x) = 3.8x*(1-x); A(x) = x^2

birkhoffmeans = Array{Float64}(undef,M)
for m = 1:M # generate M birkhoff means
    birkhoffmeans[m] = birkhoff_mean(f,A,N;x0=rand())
end
mean_of_birkhoff_means = mean(birkhoffmeans)
std_of_birkhoff_means = std(birkhoffmeans)

tconst = cquantile(TDist(M-1),0.05/2)/sqrt(M)

figure(figsize=(10,2))
errorbar(mean_of_birkhoff_means,[1],xerr=tconst*std_of_birkhoff_means,linestyle="",marker=".")
plot(fill(true_logistic38mean,2),[0,101],"k--"); ylim(0,2)
yticks([]);
```



Let's instead try to have some spin-up:

$$B_{N_0,N} = \frac{1}{N}\sum_{n=0}^{N-1} A(f^{n+N_0}(x)),\ x \sim \mu$$

Let's instead try to have some spin-up:

$$B_{N_0,N} = \frac{1}{N}\sum_{n=0}^{N-1} A(f^{n+N_0}(x)),\ x \sim \mu$$

$$\mathbb{E}[B_{N,N_0}] = \int_M \frac{1}{N}\sum_{n=0}^{N-1} A(f^{n+N_0}(x))\,\mathrm{d}\mu = \int_M A\,\mathrm{d}\rho + \frac{1}{N}\sum_{q=1}^{p-1} c_q \xi^{qN_0}\frac{1-\xi^q}{1-\xi^q} + \frac{1}{N}\sum_{n=0}^{N-1} Q_{n+N_0}$$

$$= \int_M A\,\mathrm{d}\rho + \mathcal{O}(\tfrac{1}{N}\lambda^{-N_0})$$

if N is a multiple of p.

In [407]:

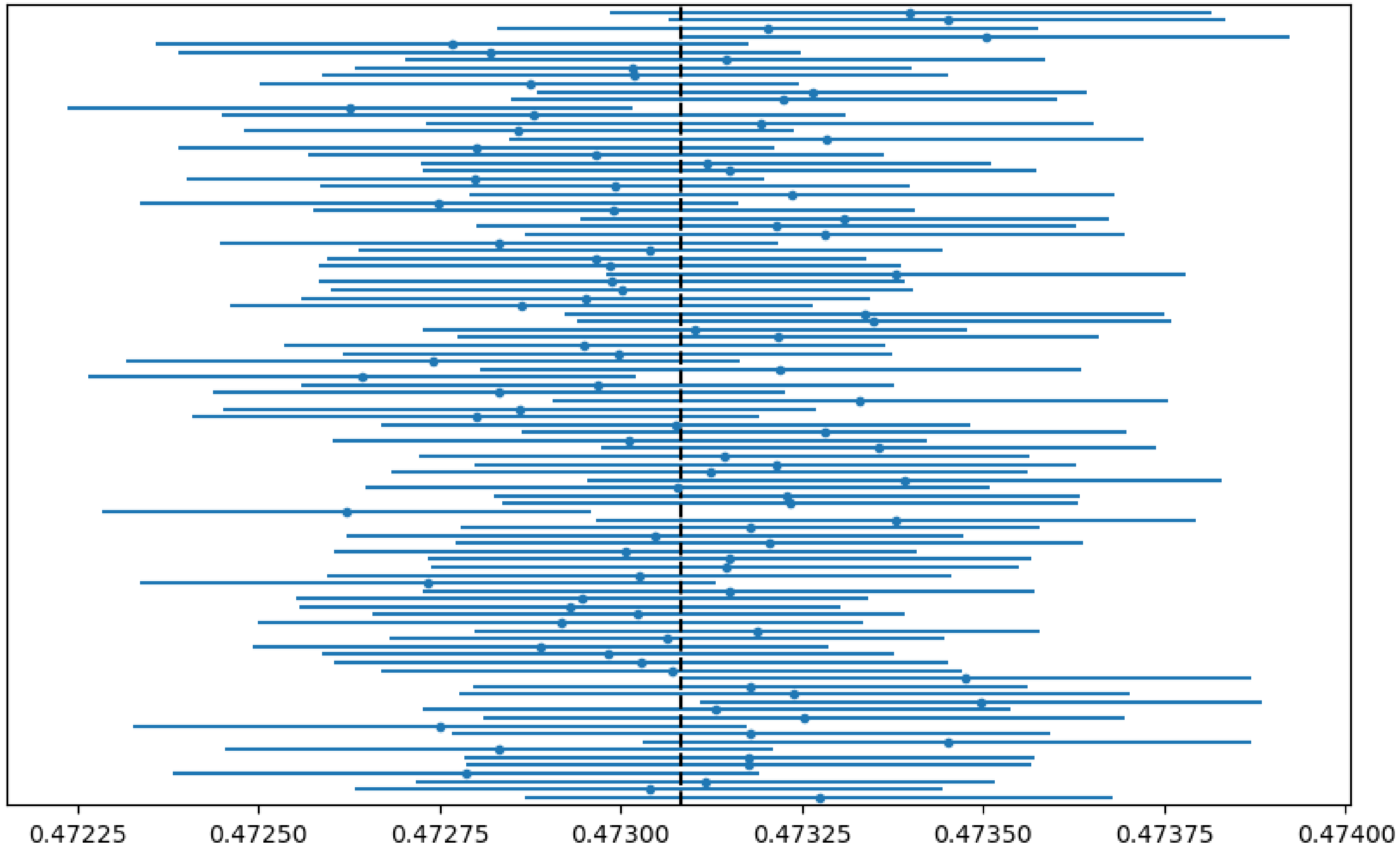
```julia
function birkhoff_mean( f, # map
                        A, # observation function
                        N; # time series length
                        N0=0, # spinup time
                        x0=rand()) # initial value

    x = x0
    for i = 1:N0 #spin-up time
        x = f(x)
    end

    birkhoff_sum = 0.
    for n = 1:N
        birkhoff_sum += A(x)
        x = f(x)
    end
    birkhoff_sum / N # mean
end
```

Out[407]:

```
birkhoff_mean (generic function with 1 method)
```

In [408]:

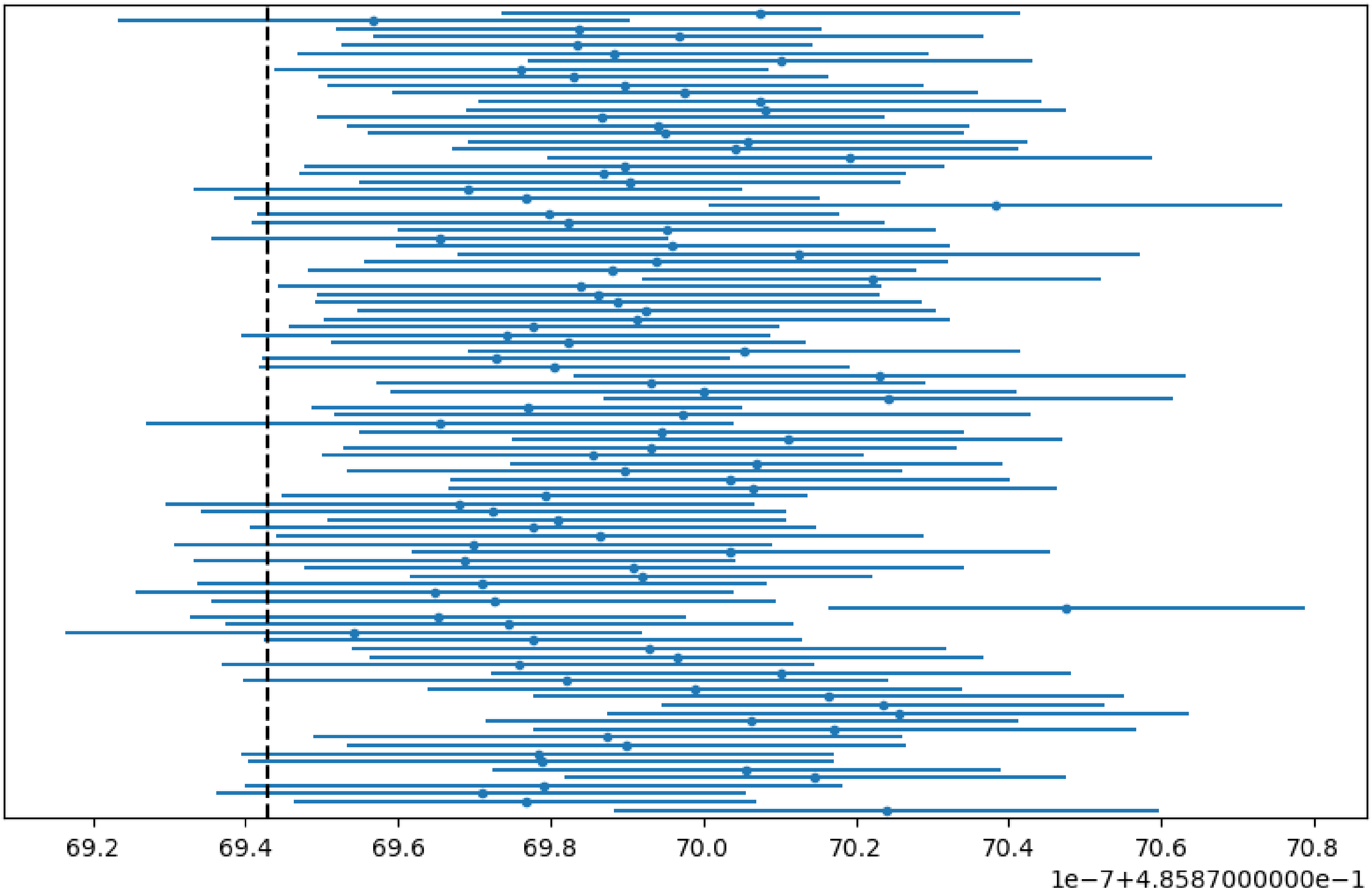
```
M = 200; N = 300; N0 = 10000
means_of_birkhoff_means = Array{Float64}(undef,100)
stds_of_birkhoff_means = Array{Float64}(undef,100)

for i = 1:100 # generate 100 mean-of-means
    birkhoffmeans = Array{Float64}(undef,M)
    for m = 1:M # generate M birkhoff means
        birkhoffmeans[m] = birkhoff_mean(f,A,N;N0=N0,x0=rand())
    end
    means_of_birkhoff_means[i] = mean(birkhoffmeans)
    stds_of_birkhoff_means[i] = std(birkhoffmeans)
end

tconst = cquantile(TDist(M-1),0.05/2)/sqrt(M)

figure(figsize=(10,6))
errorbar(means_of_birkhoff_means,1:100,xerr=tconst*stds_of_birkhoff_means,linestyle="",marker=".")
plot(fill(true_logistic38mean,2),[0,101],"k--"); ylim(0,101)
yticks([]);
```


0.47225
0.47250
0.47275
0.47300
0.47325
0.47350
0.47375
0.47400

Spin-up is also very important if it takes a while to "find" the physical measure:

In [415]:

```
f(x) = 3.7030314384x*(1-x) # remember this guy

M = 20; N = 3*10^6; N0 = 10^6
means_of_birkhoff_means = Array{Float64}(undef,100)
stds_of_birkhoff_means = Array{Float64}(undef,100)

for i = 1:100 # generate 100 mean-of-means
    birkhoffmeans = Array{Float64}(undef,M)
    for m = 1:M # generate M birkhoff means
        birkhoffmeans[m] = birkhoff_mean(f,A,N;N0=N0,x0=rand())
    end
    means_of_birkhoff_means[i] = mean(birkhoffmeans)
    stds_of_birkhoff_means[i] = std(birkhoffmeans)
end

tconst = cquantile(TDist(M-1),0.05/2)/sqrt(M)

figure(figsize=(10,6))
errorbar(means_of_birkhoff_means,1:100,xerr=tconst*stds_of_birkhoff_means,linestyle="",marker=".")
plot(fill(true_logistic37030314384mean,2),[0,101],"k--"); ylim(0,101)
yticks([]);
```


69.2
69.4
69.6
69.8
70.0
70.2
70.4
70.6
70.8
1e−7+4.8587000000e−1

We've been doing a lot of simulations, and they seem to work, but they don't always work:

We've been doing a lot of simulations, and they seem to work, but they don't always work:

In [421]:

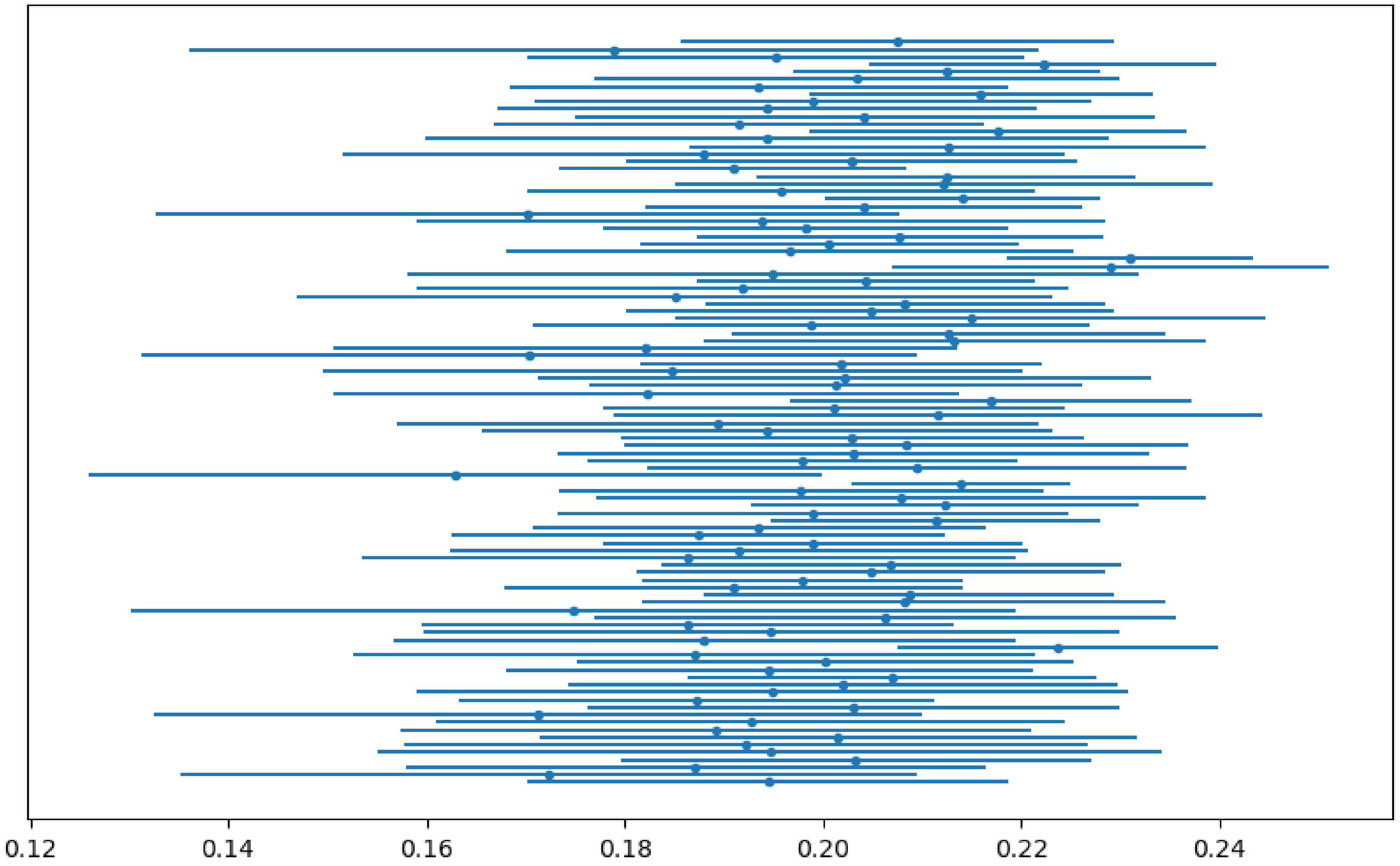
```
LSV(x) = x < 0.5 ? x*(1+(2x)^(0.9)) : 2x-1
A(x) = x
```

Out[421]:

```
A (generic function with 1 method)
```

In [422]:

```
M = 20; N = 30000; N0 = 30000

means_of_birkhoff_means = Array{Float64}(undef,100)
stds_of_birkhoff_means = Array{Float64}(undef,100)

for i = 1:100 # generate 100 mean-of-means
    birkhoffmeans = Array{Float64}(undef,M)
    for m = 1:M # generate M birkhoff means
        birkhoffmeans[m] = birkhoff_mean(LSV,A,N;N0=N0,x0=rand())
    end
    means_of_birkhoff_means[i] = mean(birkhoffmeans)
    stds_of_birkhoff_means[i] = std(birkhoffmeans)
end

tconst = cquantile(TDist(M-1),0.05/2)/sqrt(M)

figure(figsize=(10,6))
errorbar(means_of_birkhoff_means,1:100,xerr=tconst*stds_of_birkhoff_means,linestyle="",marker=".")
# plot(fill(true_LSVmean,2),[0,101],"k--"); ylim(0,101)
yticks([]);
```



0.12
0.14
0.16
0.18
0.20
0.22
0.24

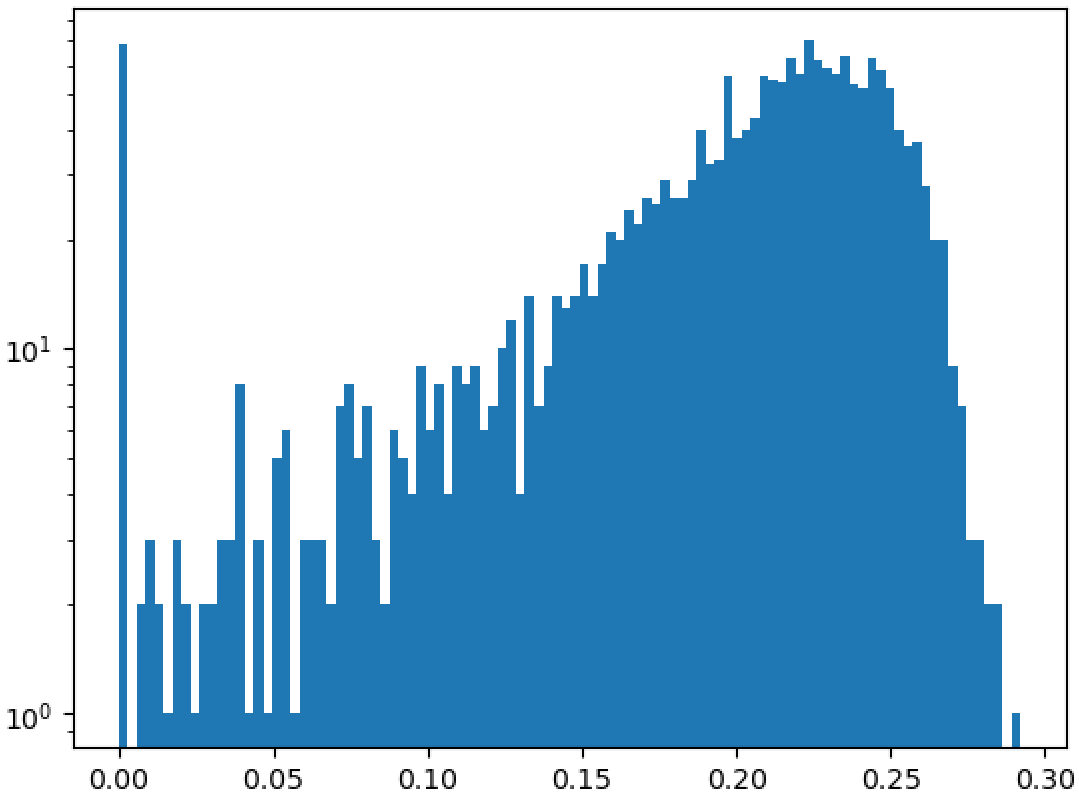
For this map/observable combination, our Birkhoff means aren't normally distributed:

In [423]:

```
M = 2000
birkhoffmeans = Array{Float64}(undef,M)
for m = 1:M # generate M birkhoff means
    birkhoffmeans[m] = birkhoff_mean(LSV,A,N;N0=N0,x0=rand())
end
```

In [424]:

```
hist(birkhoffmeans,bins=100);
semilogy()
```



Out[424]:

```
Any[]
```

Statistical test for chaos

(Really a statistical test for decay of correlations)

In [374]:

```
bumpfunc_raw(x) = exp(-1/(x*(1-x)))
const bumpfunc_raw_cons = quadgk(bumpfunc_raw,0,1)[1]
bumpfunc(x) = bumpfunc_raw(x) / bumpfunc_raw_cons
```

```
WARNING: redefinition of constant bumpfunc_raw_cons. Th
is may fail, cause incorrect answers, or produce other
errors.
```

Out[374]:

```
bumpfunc (generic function with 1 method)
```

In [375]:

```
function weighted_birkhoff_mean( f, # map
                        A, # observation function
                        N; # time series length
                        twist_angle = 0, # twist
                        N0=0, # spinup time
                        x0=rand()) # initial value

    x = x0
    for i = 1:N0 #spin-up time
        x = f(x)
    end

    twist = cis(twist_angle)
    twistpow = 1.
    birkhoff_sum = 0.
    for n = 1:N
        birkhoff_sum += twistpow*A(x)*bumpfunc(n/N)
        x = f(x)
        twistpow *= twist
    end
    birkhoff_sum / N # mean
end
```

Out[375]:

```
weighted_birkhoff_mean (generic function with 1 method)
```

In [379]:

```julia
weighted_birkhoff_mean(x->3.6x*(1-x), A, 1000;twist_angle=2)
```

Out[379]:

```
0.0003829297923169078 - 0.0007249808053832529im
```

In [ ]:

In [ ]: