Chaos, Complexity, and Inference (36-462)

Lecture 1

Cosma Shalizi

15 January 2008

Course Goals

$\ast$ Learn about developments in dynamics and systems theory
$\ast$ Understand how they relate to fundamental questions in stochastic modeling (what is randomness? when can we use stochastic models?)
$\ast$ Think about how to do statistical infrence for dependent data
$\ast$ Get some practice with building and using simulation models
$\ast$ You have learned a lot about linear regression with independent samples and Gaussian noise
$\ast$ We are going to break all that

Approach

- ∗ Read, simulate, do a few calculations
- ∗ No or almost no theorems
- ∗ Much rigor necessarily skipped
- ∗ A lot of reading — this is deliberate
- ∗ Move from lectures to discussions as the course goes

stat.cmu.edu/ cshalizi/462/syllabus.html

Grading

Homework one problem set every 2–3 weeks
$1/2$ of grade

Class participation $1/6$ of grade

Final paper 10–20 pages, due on final exam date
$1/3$ of grade

About the paper

Experiment in practicing writing about technical material

Possibilities:

- Detailed review of some chunk of course material
- Exposition of one of the optional papers
- Critique of paper or material from the syllabus/literature
- Implementing your own model or applying a technique to data

Topics *must* be approved by me in advance

You will turn in drafts for feedback well before final

Exact dates TBD

Topics

Dynamical Systems Jan. 15–Feb. 7
Models, dynamics, chaos, information, randomness

Self-organization Feb. 12–Feb. 21
Self-organizing systems, cellular automata

Heavy-tailed Distributions Feb. 26–Mar. 6
Examples, properties, origins, estimation, testing

Inference from Simulations Mar. 18–Mar. 27
Severity; Monte Carlo; direct and indirect inference

Complex Networks and Agent-Based Models Apr. 1–Apr. 29
Network structures & growth; collective phenomena; inference; real-world example

Chaos, Complexity and Inference May 1

Models and Simulations

Model is a way of representing dependencies in some part of the world

Hope: tracing consequences in the model lets you predict reality

E.g., a map: tracing a route predicts what you will see and how you can get from A to B

Regressions are models of input/output

Simulating is tracing through consequences step by step in a particular case

Simulation is basic; analytical results are short-cuts to avoid exhaustive simulation (which may not be possible)

Dynamical Systems

We are particularly interested in *dynamical* models, which represent changes over time

Components of a dynamical system

- state space : fundamental variables which determine what will happen
- update rule : rule for how the state changes over time, may be stochastic.
  A.k.a. **map** or **evolution equations** or **equations of motion**:
- observables : variables we actually measure; functions of state (+ possible noise)

initial condition: starting state

trajectory or **orbit**: sequence of states over time

A work-horse example: the logistic map

state x, population of some animal, rescaled to some maximum value (so $x \in [0, 1]$)

map $x_{t+1} = 4rx_t(1 - x_t) \equiv f(x)$
the x factor means that animals make more animals
$1 - x$ factor means that too many animals keep there from being as many animals
r is control parameter in $[0, 1]$ (following notation in Flake)

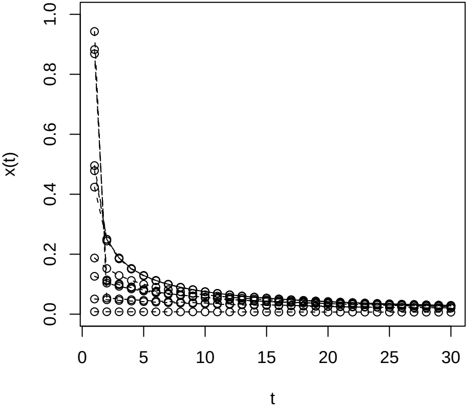
observable : we get to observe x directly, without noise

horrible caricature — we will see much better population models — but mathematically simple and it illustrates many important points

Set $r = 0.25$ and pick some random starting points
First some code — R doesn't like iteration but we need it here

```
logistic.map <- function(x,r) {
  return(4*r*x*(1-x))
}

logistic.map.ts <- function (timelength,r,initial.cond=NULL) {
  x <-vector(mode="numeric",length=timelength)
  if(is.null(initial.cond)) {
    x[1] <-runif(1)
  } else {
    x[1] <-initial.cond
  }
  for (t in 2:timelength) {
      x[t] = logistic.map(x[t-1],r)
    }
  return(x)
}
```

```
plot.logistic.map.trajectories <- function(timelength,
                                           num.traj,r) {
  plot(1:timelength,logistic.map.ts(timelength,r),lty=2,
       type="b",ylim=c(0,1),xlab="t",ylab="x(t)")
  i = 1
  while (i < num.traj) {
    i <- i+1
    x <- logistic.map.ts(timelength,r)
    lines(1:timelength,x,lty=2)
    points(1:timelength,x)
  }
}

plot.logistic.map.trajectories(30,10,0.25)
```



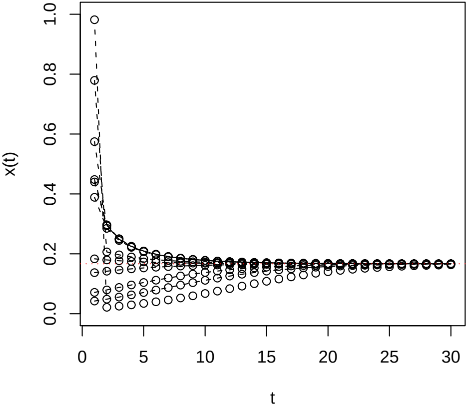
All trajectories seem to be converging to the same value

They are! They are going to a **fixed point**
Solve:

$$\begin{aligned} x &= 4(0.25)x(1-x) \\ x &= x - x^2 \\ 0 &= x^2 \end{aligned}$$

Not very interesting!

Let's change r let's say 0.3.

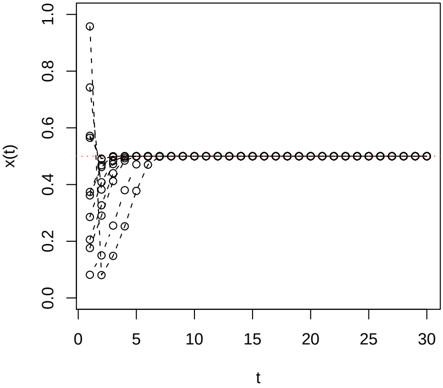


Still converging but to a different value

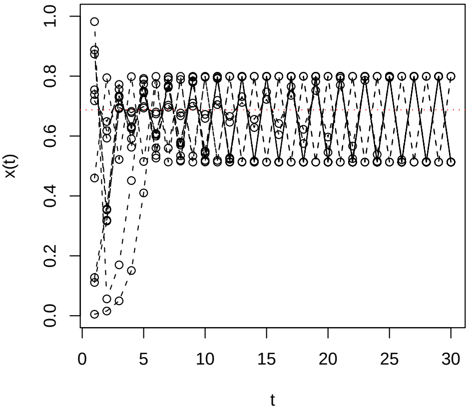
$$
\begin{aligned}
x &= 1.2x - 1.2x^2 \\
0 &= 0.2x - 1.2x^2 \\
0 &= x - 6x^2
\end{aligned}
$$

Solutions are obviously $x = 0$ and $x = 1/6$. Note all the trajectories converging to 1/6 (marked in red).
Why do they like 1/6 more than 0?
Can you show that 0 is always a fixed point?

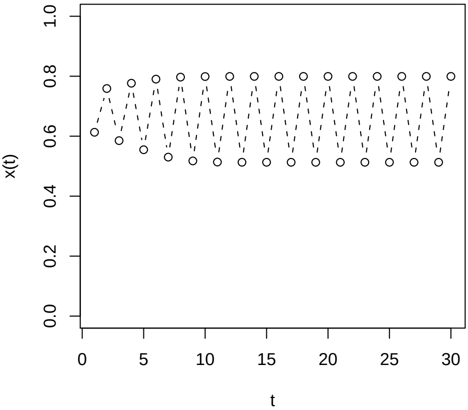
Crank up r again, to 0.5; fixed points at $x = 0$ and $x = 0.5$
Again they like one fixed point but not the other



Now $r = 0.8$; the fixed points are $x = 0$ and $x = 11/16$



What the bleep? Let's look at just one trajectory



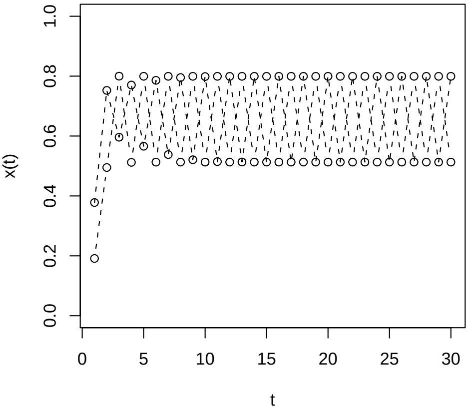
It's gone to a **cycle** or **periodic orbit**, of period two

This means that there are two solutions to $x = f(f(x))$ which are not solutions of $x = f(x)$

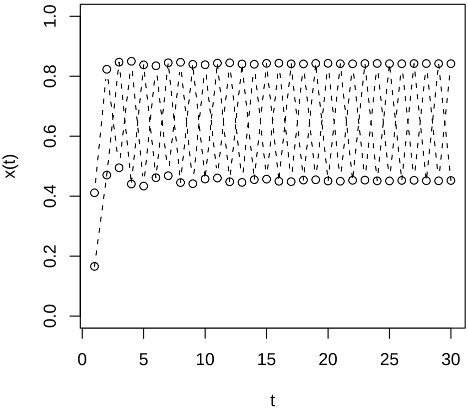
$$x = 3.2[3.2x(1-x)][1-3.2x(1-x)]$$

Quartic equation, so four solutions — we know two of them ($x = 0$, $x = 11/16$) because they are fixed points; the other two are the points of the periodic cycle

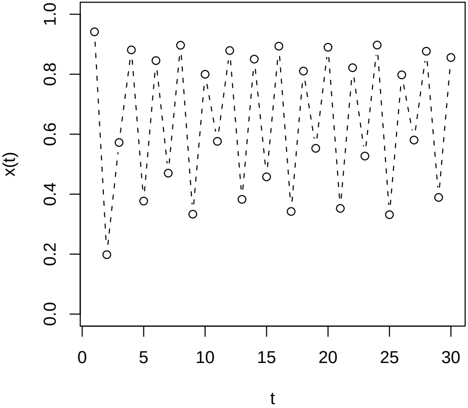
Phase of the cycle depends on the **initial condition**



Increasing *r* increases the **amplitude** of the **oscillation**



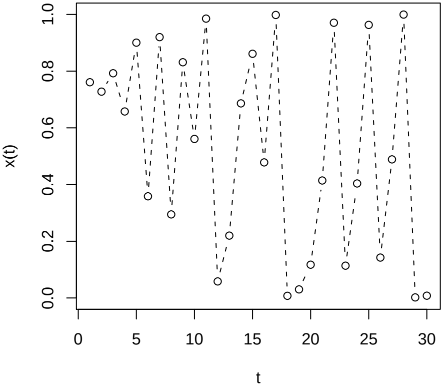
Increasing r even more (0.9) I get period 4



You will work out more about the periodic orbits in the homework!

Now all the way to $r = 1$

Not periodic *at all* and never converges — **chaos**



Properties of Chaos

We will define “chaos” more strictly next time
For now look at some characteristics

- Sensitive dependence on initial conditions
- Statistical stability of multiple trajectories
- Individual trajectories look representative samples (ergodicity)
- Short-term nonlinear predictability

Sensitive dependence on initial conditions

Deterministic: same initial point has the same future trajectory

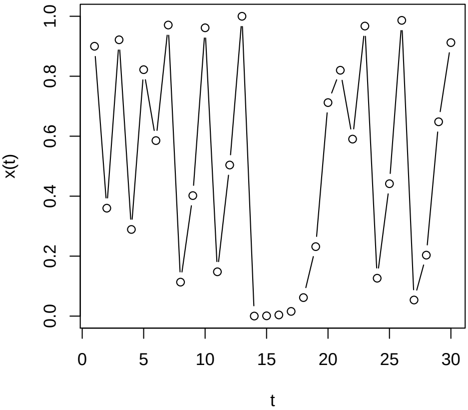
Continuity: can get arbitrarily small differences in trajectory by arbitrarily small differences in initial condition

BUT

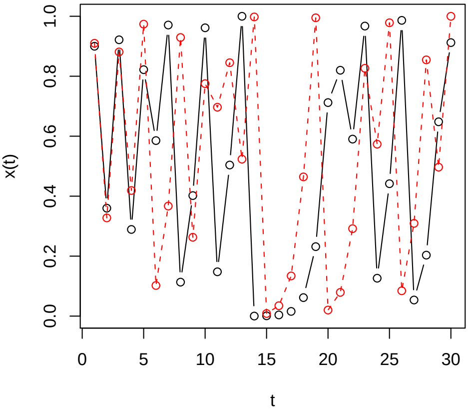
Amplification of differences in initial conditions: if $|x_1 - y_1| = \epsilon$, then $|x_t - y_t| \approx \epsilon e^{\lambda t}$ for some $\lambda > 0$

Simplest SDIC: $x_{n+1} = \alpha x_n$ for $\alpha > 1$

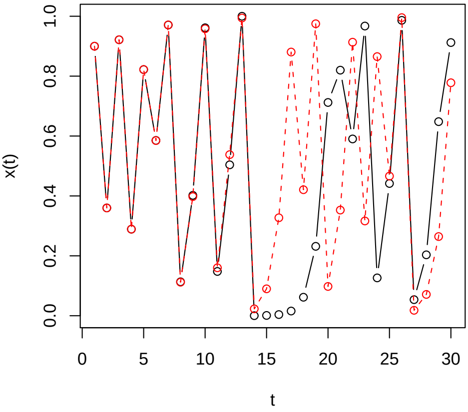
More complicated behavior when SDIC isn't combined with run-away growth fix $x_1 = 0.90$



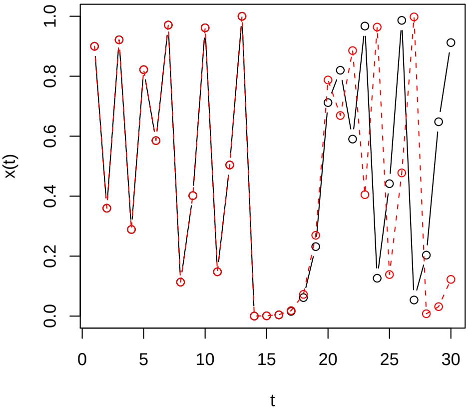
compare $x_1 = 0.90$ to $y_1 = 0.91$; tracking to about $t = 4$



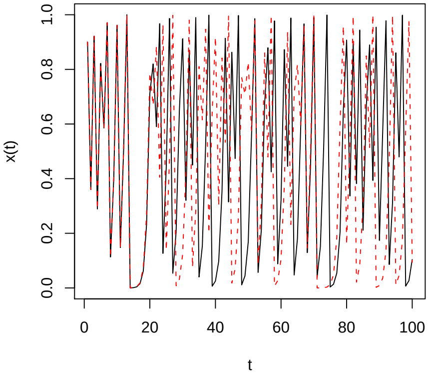
compare $x_1 = 0.90$ to $y_1 = 0.90001$; tracking to about $t = 12$



$x_1 = 0.90$ vs. $y_1 = 0.9000001$; tracking to about $t = 20$



extend both trajectories

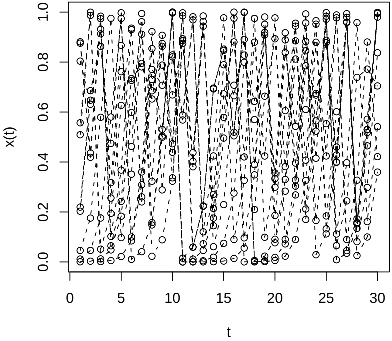


note that they get back together again around $t = 60$

Statistical stability

Look at what happens to an **ensemble** of trajectories

Seem to be more dots near the edges than in the middle

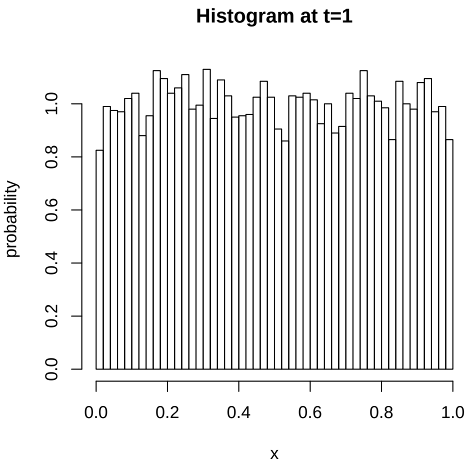


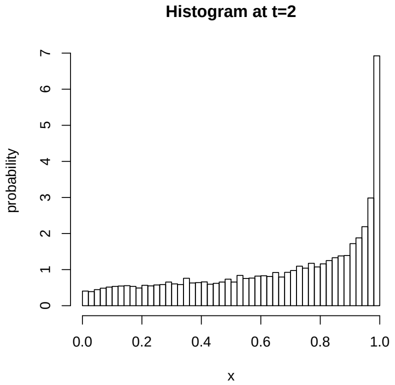
This is true!
To check it we need to evolve many trajectories in parallel

```
logistic.map.evolution <- function(timesteps,r,x) {
  t=0
  while (t < timesteps) {
    x <- logistic.map(x,r)
    t <- t+1
  }
  return(x)
}
```

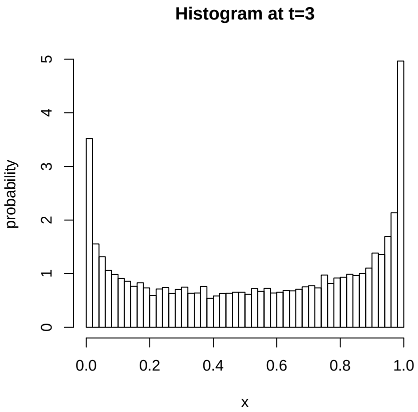
Now run 10^4 initial points, uniformly distributed

```
> x1=runif(10000)
> hist(logistic.map.evolution(999,1,x1),freq=FALSE,xlab="x",
      ylab="probability",main="Histogram at t=1000",n=41)
```

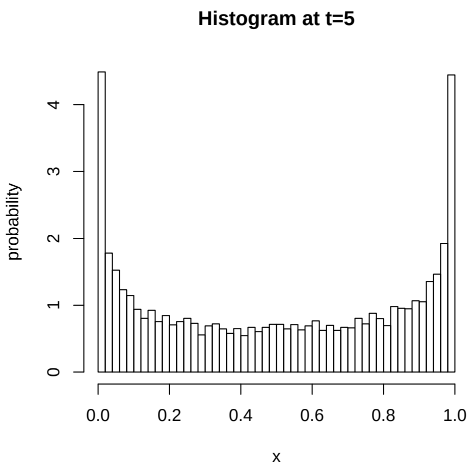

Histogram at t=1
probability
0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.2
0.4
0.6
0.8
1.0
x



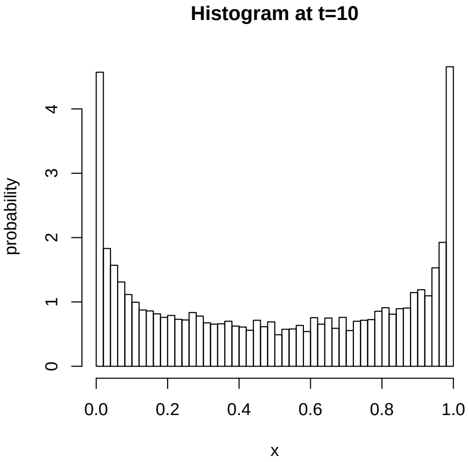
Points near 0.5 get mapped towards 1, and the map function changes slowly there, but only points near 0 or 1 get mapped to 0, and the function changes rapidly in those places

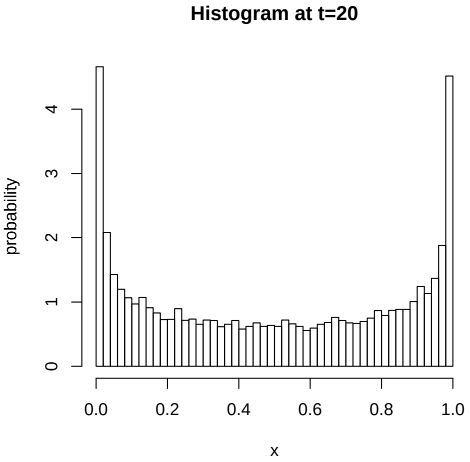


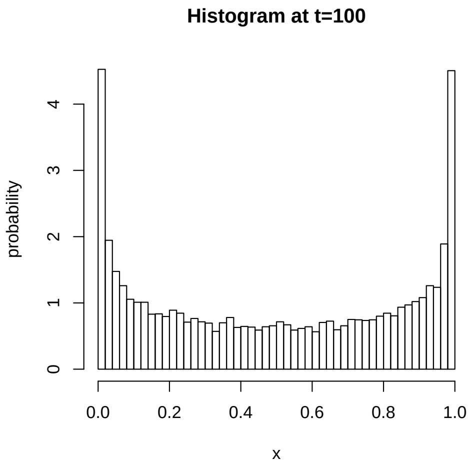
Many points which had gotten near 1 get mapped to near 0, but those near $1/2$ are still mapped towards 1

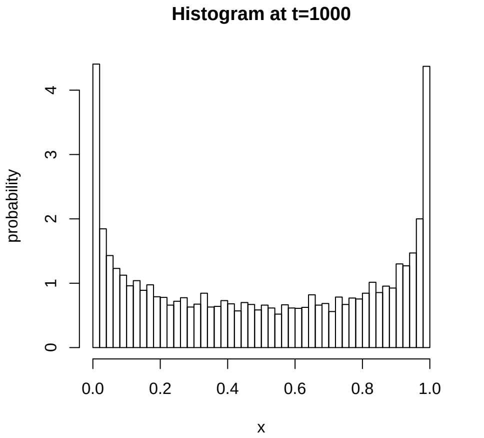


The two modes are getting balanced


Histogram at t=10
probability
0
1
2
3
4
0.0
0.2
0.4
0.6
0.8
1.0
x


Histogram at t=20
probability
0
1
2
3
4
0.0
0.2
0.4
0.6
0.8
1.0
x

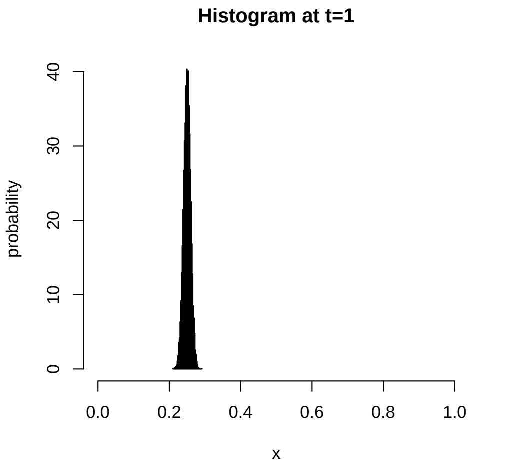

Histogram at t=100
probability
0
1
2
3
4
0.0
0.2
0.4
0.6
0.8
1.0
x

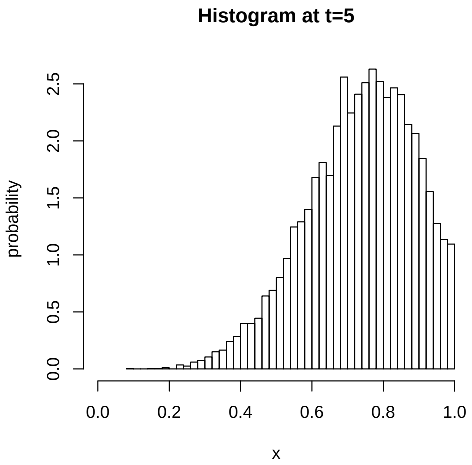


Distribution converges rapidly to an **invariant** distribution

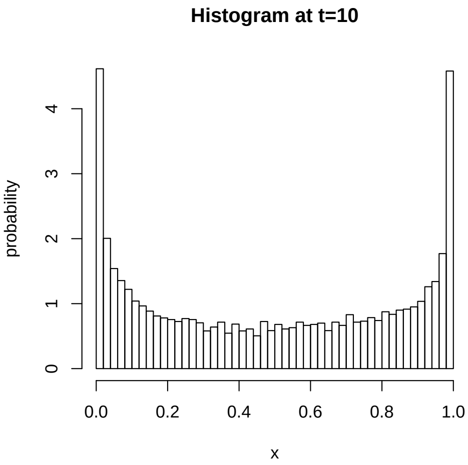
To see that let's try a different initial distribution, say a Gaussian with mean 0.25, s.d. 0.01, cutting out those outside $[0, 1]$.

```
> x2 = rnorm(1e4,0.25,0.01)
> x2 = x2[x2 >= 0]
> x2 = x2[x2 <= 1]
> length(x2)
[1] 10000
> hist(x2,freq=FALSE,xlab="x",ylab="probability",
      main="Histogram at t=1",n=41,xlim=c(0,1))
> hist(logistic.map.evolution(4,1,x2),freq=FALSE,xlab="x",
      ylab="probability",main="Histogram at t=5",n=41,
      xlim=c(0,1))
```


Histogram at t=1
probability
0
10
20
30
40
0.0
0.2
0.4
0.6
0.8
1.0
x


Histogram at t=5
probability
0.0
0.5
1.0
1.5
2.0
2.5
0.0
0.2
0.4
0.6
0.8
1.0
x

by $t \approx 10$ it looks like as though initial conditions were uniform



Even though individual trajectories fluctuate all over, the *distribution* converges

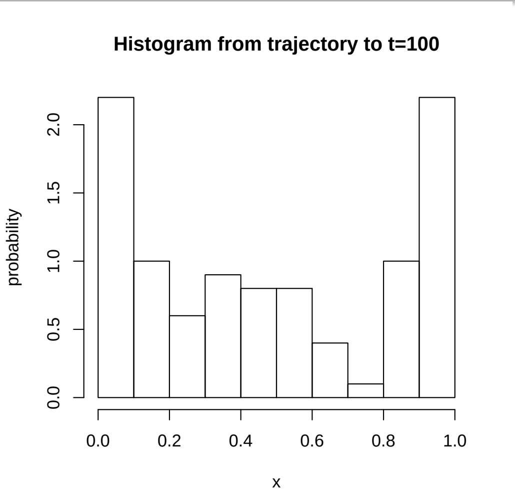
The **invariant distribution** is in fact

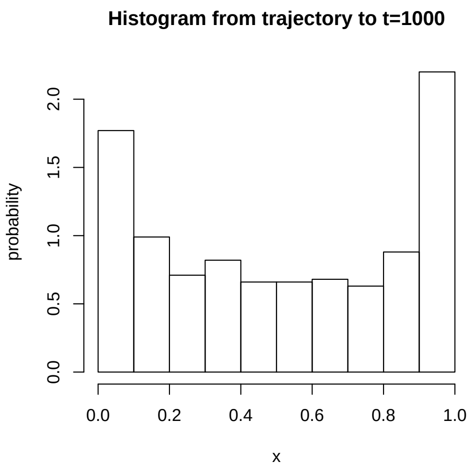
$$p(x) = \frac{1}{\pi\sqrt{x(1-x)}}$$

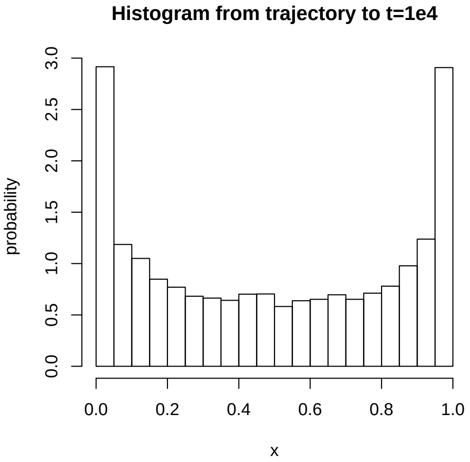
Ergodicity

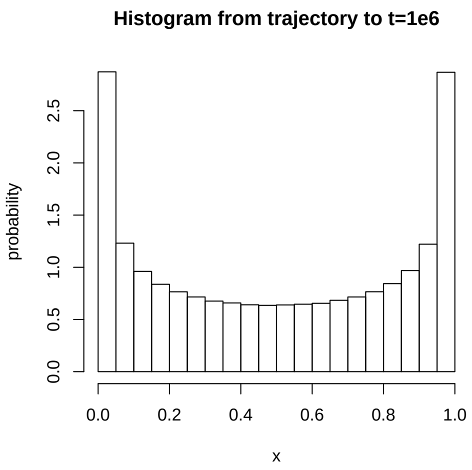
If we do look at an individual trajectory, it looks similar to the whole ensemble of trajectories; here is $x_1 = 0.9$

```
> hist(logistic.map.ts(1e3,1,0.9),freq=FALSE,xlab="x",
      ylab="probability",
      main="Histogram from trajectory to t=1000")
```


Histogram from trajectory to t=100
probability
0.0
0.5
1.0
1.5
2.0
0.0
0.2
0.4
0.6
0.8
1.0
x

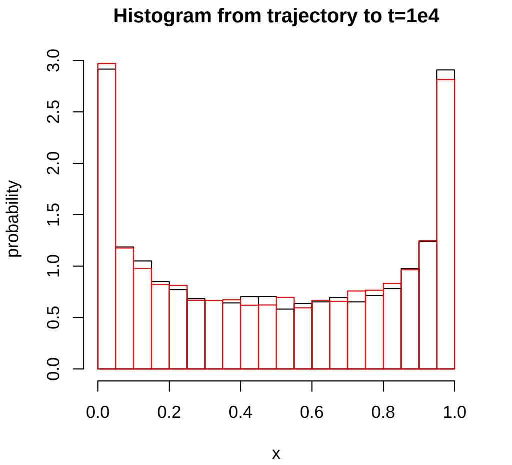

Histogram from trajectory to t=1000
probability
0.0
0.5
1.0
1.5
2.0
0.0
0.2
0.4
0.6
0.8
1.0
x

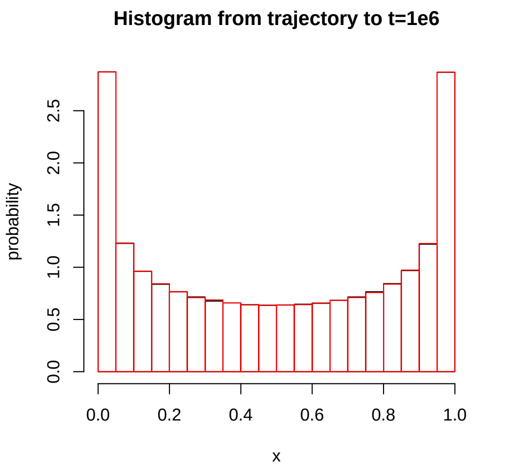

Histogram from trajectory to t=1e4
probability
0.0
0.5
1.0
1.5
2.0
2.5
3.0
0.0
0.2
0.4
0.6
0.8
1.0
x


Histogram from trajectory to t=1e6
probability
0.0
0.5
1.0
1.5
2.0
2.5
0.0
0.2
0.4
0.6
0.8
1.0
x

looks pretty much like what you see from any one other trajectory (here is $y_1 = 0.91$ in red)

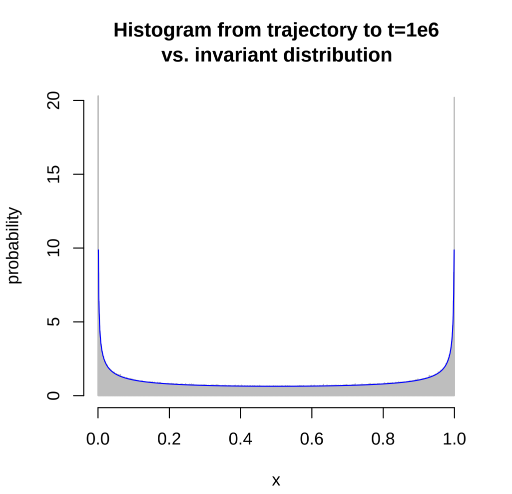
```
> hist(logistic.map.ts(1e6,1,0.9),freq=FALSE,xlab="x",
       ylab="probability",
       main="Histogram from trajectory to t=1e6",
       n=1001)
> hist(logistic.map.ts(1e6,1,0.91),freq=FALSE,xlab="x",
       ylab="probability",
       main="Histogram from trajectory to t=1e6",
       add=TRUE,border="red",n=1001)
```


Histogram from trajectory to t=1e4
probability
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0.0
0.2
0.4
0.6
0.8
1.0
x


Histogram from trajectory to t=1e6
probability
0.0
0.5
1.0
1.5
2.0
2.5
0.0
0.2
0.4
0.6
0.8
1.0
x

In every case they are converging on the exact invariant distribution

```
> hist(logistic.map.ts(1e6,1,0.9),freq=FALSE,xlab="x",
      ylab="probability",
      main="Histogram from trajectory to t=1e6\nvs.
            invariant distribution",
      n=1001,border="grey")
> curve(1/(pi*sqrt(x*(1-x))),col="blue",add=TRUE,n=1001)
```


Histogram from trajectory to t=1e6
vs. invariant distribution
probability
0
5
10
15
20
0.0
0.2
0.4
0.6
0.8
1.0
x

Ergodicity means that almost any long trajectory looks like a representative sample from the invariant distribution

We will define this more precisely later, and explore why it is so important for stochastic modeling

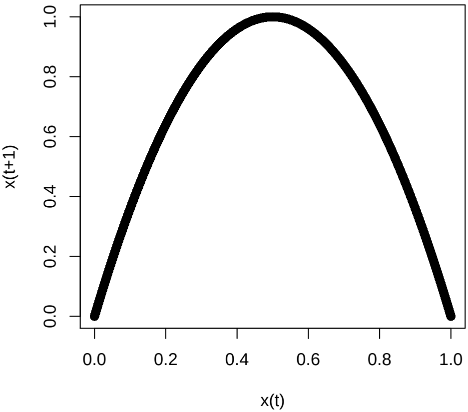
Short-Term Nonlinear Predictability

```
x.ts <- logistic.map.ts(1e6,1,0.9)
```

x_{t+1} on x_t

```
plot(x.ts[1:1e4],x.ts[2:(1e4+1)],xlab="x(t)",ylab="x(t+1)",
     type="p")
```

only 10^4 points so it plots in a reasonable amount of time

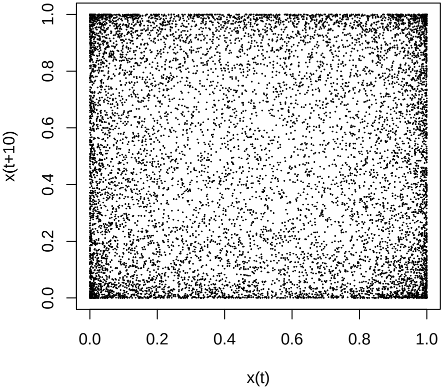

x(t+1)
0.0
0.2
0.4
0.6
0.8
1.0
x(t)

Linear regression is *not* your friend:

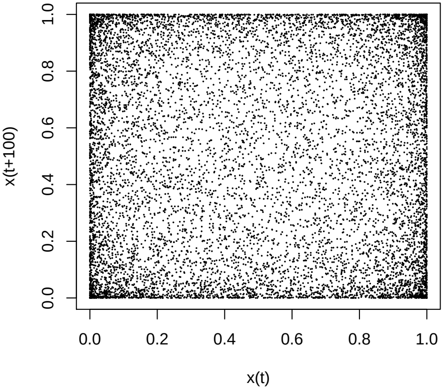
```
> lm1 <- lm(x.ts[2:1e6] ~ x.ts[1:(1e6-1)])
> summary(lm1)

Call:
lm(formula = x.ts[2:1e+06] ~ x.ts[1:(1e+06 - 1)])

Residuals:
       Min         1Q     Median         3Q        Max
-0.5005069 -0.3535795  0.0005158  0.3531829  0.4999921

Coefficients:
                     Estimate Std. Error t value Pr(>|t|)
(Intercept)         0.4995090  0.0006124 815.687   <2e-16 ***
x.ts[1:(1e+06 - 1)] 0.0009979  0.0010000   0.998    0.318
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.3536 on 999997 degrees of freedom
Multiple R-Squared: 9.958e-07,  Adjusted R-squared: -4.188e-09
F-statistic: 0.9958 on 1 and 999997 DF,  p-value: 0.3183
```

x_{t+10} on x_t

The joint distribution here is very close to being independent



x_{t+100} on x_t
Even closer to independence



... except that x_{t+k} is a *determistic function* of x_t, no matter what k is, so how can they be independent?