Title: Markov Chain Monte Carlo

Speakers: Dustin Lang
Date: September 17, 2019 - 1:00 PM

URL: http://pirsa.org/19090120

Markov Chain Monte Carlo

Dustin Lang

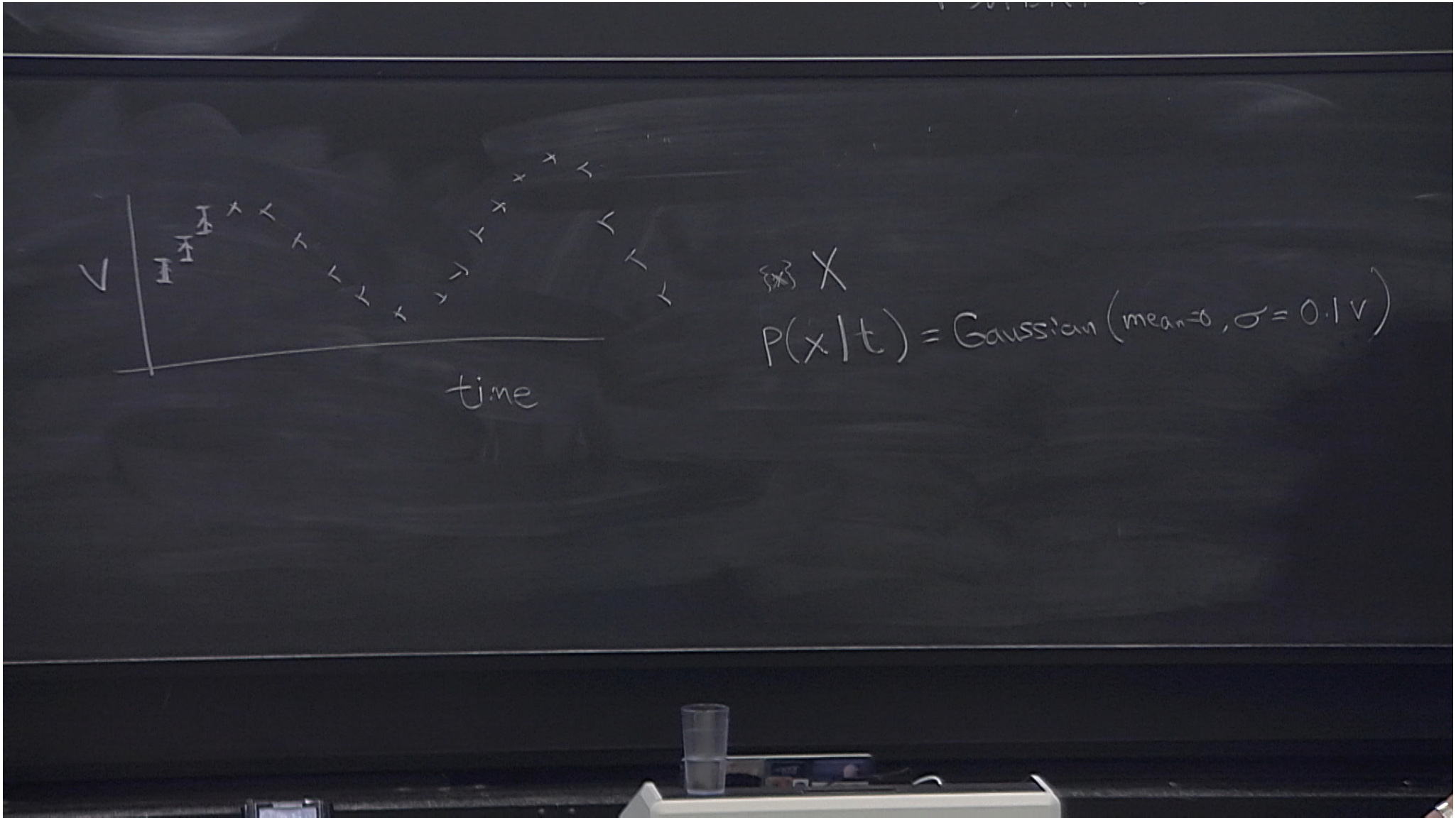
2019-09-17

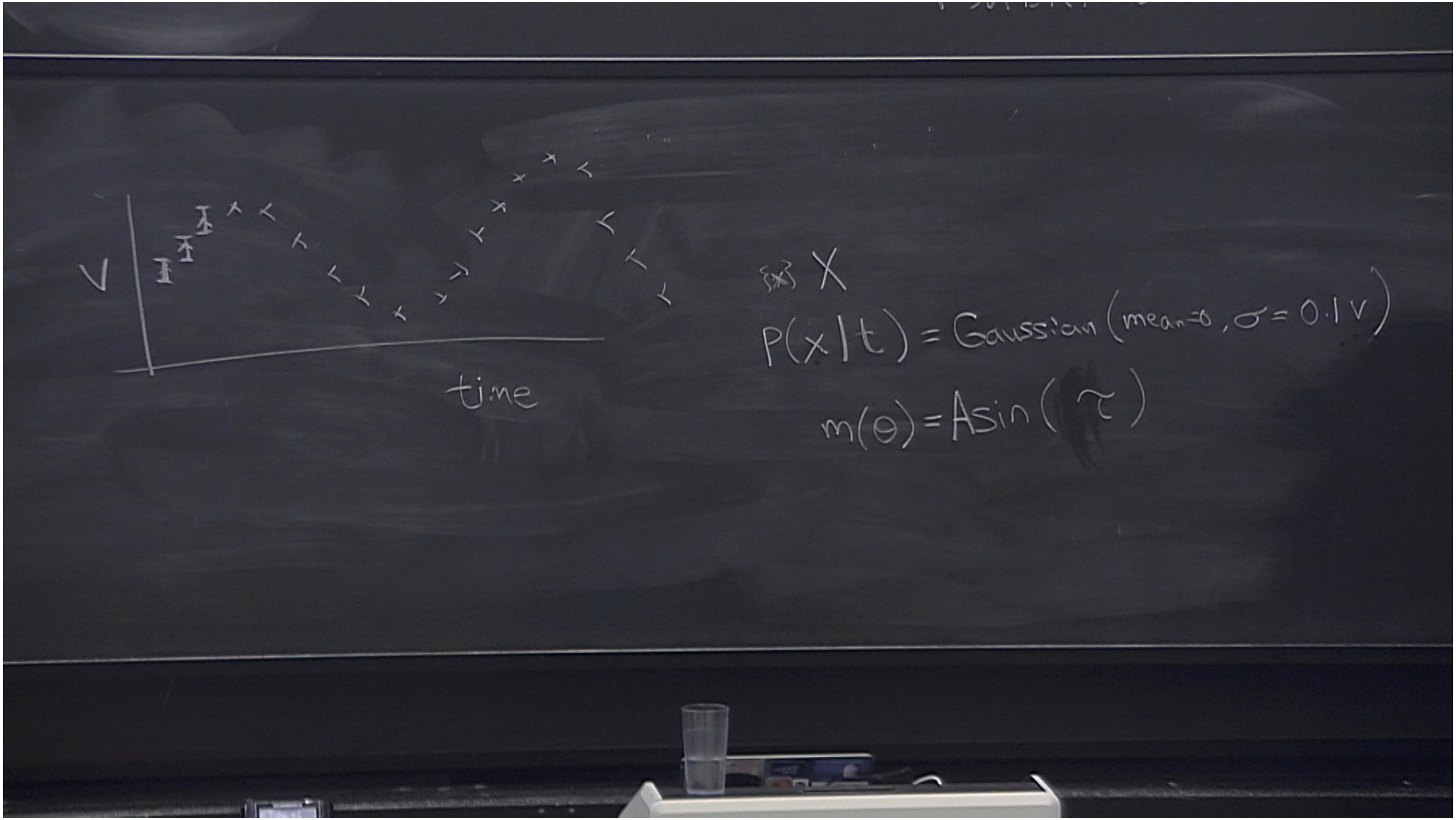
Markov Chain Monte Carlo for data analysis

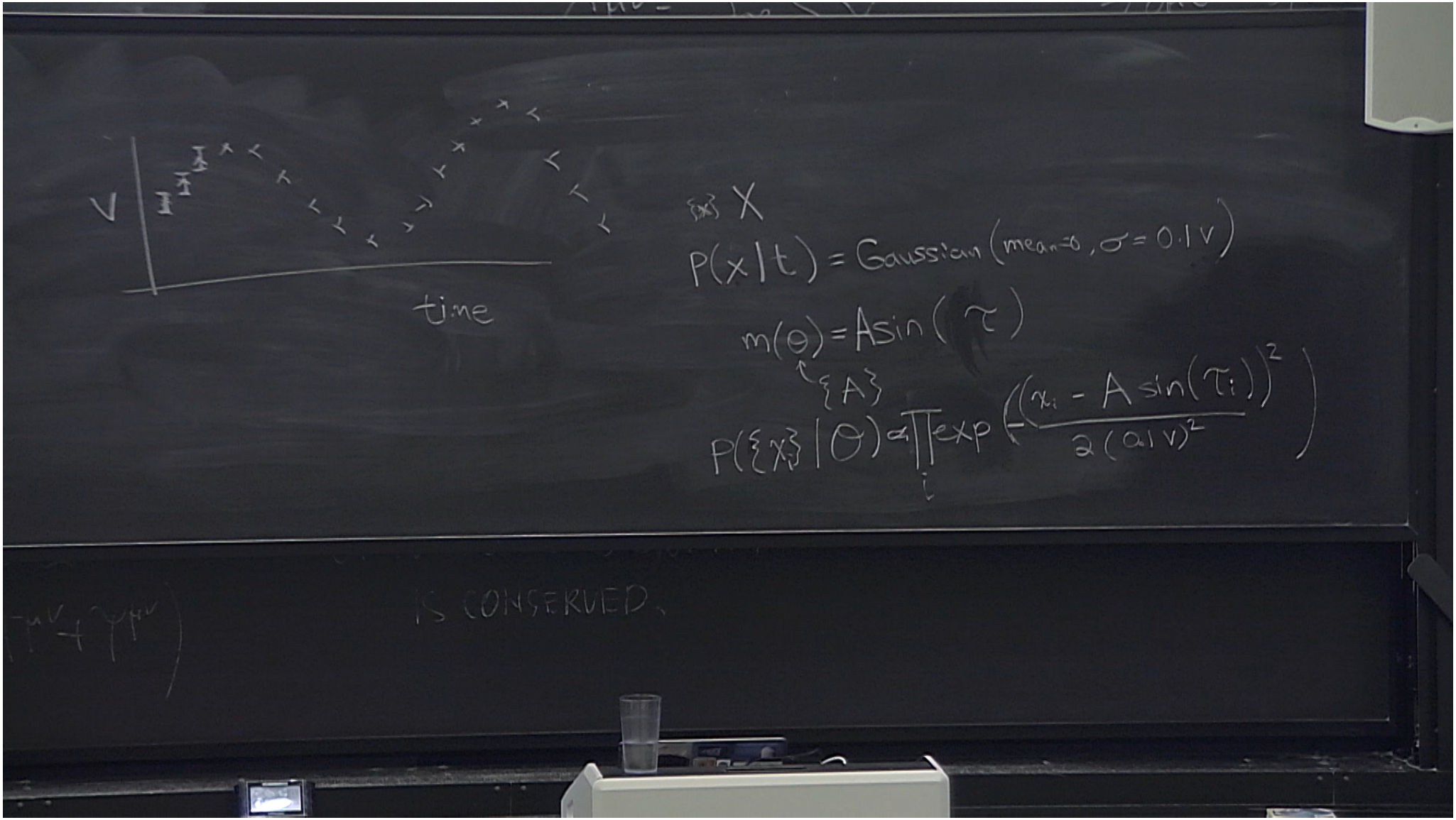
- MCMC generates samples from a probability distribution function
- For data analysis: you have some *measurements* X, with uncertainties
- have a *model* for the thing you're measuring
- with unknown *parameters* θ
- the model is *generative* – predicts measurements given parameters
- MCMC can put constraints on the parameters given your measurements

Markov Chain Monte Carlo for data analysis

- can think of it as: *model* is a function mapping parameters to observables t: $m(\theta) \to t$
- and *measurement process* gives probability of observing X given t: $P(X|t)$
- (eg, maybe $P(X|t)$ is a Gaussian with zero mean and standard deviation σ)
- Then, can chain $P(X|t) = P(X|m(\theta))$
- Usually, we fold m and P together to write $P(X|\theta)$


V
time
X
P(x|t) = Gaussian (mean=0, σ = 0.1 V)


V
time
X
P(x|t) = Gaussian (mean=0, σ = 0.1 V)
m(θ) = Asin(τ)


V
time
{x} X
P(x|t) = Gaussian (mean=0, σ = 0.1 V)
m(θ) = A sin(τ)
{A}
P({x}|θ) ∝ ∏ exp(-((x_i - A sin(τ_i))^2 / (2(0.1 V)^2)))
i
IS CONSERVED.

Markov Chain Monte Carlo for data analysis

- $P(X|\theta)$ (the "likelihood") is not what you want!
- You want $P(\theta|X)$ – the "posterior" – constraints on / measurements of the parameters
- Good thing you're Bayesian!
- "posterior" $P(\theta|X) = \frac{P(X|\theta)P(\theta)}{P(X)}$
- $P(\theta)$ is the "prior"
- $P(X)$ is the "evidence", and in this setting is constant (ignored)

Markov Chain Monte Carlo for data analysis

- We'll use MCMC to draw samples from the *posterior* probability distribution
- $P(\theta|X) = P(X|\theta)P(\theta)$
- The samples are a set of θ values drawn from your parameter constraints distribution

What's in a name

- *Monte Carlo* – a famous casino in Monaco – alluding to the use of *randomness* in the algorithm
- *Chain* – a list (here, a list of samples / steps θ_1, θ_2, θ_3)
- *Markov Chain* – a list of "steps" where the decision to step from θ_i to θ_{i+1} depends only on the value of θ_i

- *Metropolis–Hastings* – first authors of 1953 and 1970 papers (resp) with the simplest, most commonly taught algorithm

Using MCMC

- "Just write down the probability distributions"
- hah, the "model to predictions" part, $m(\theta) \to t$, can be *very* difficult
- eg, in cosmology, the software *CAMB* does this, in 25,000 lines of FORTRAN
- writing down the *likelihood* might take some work too!

MCMC: the idea

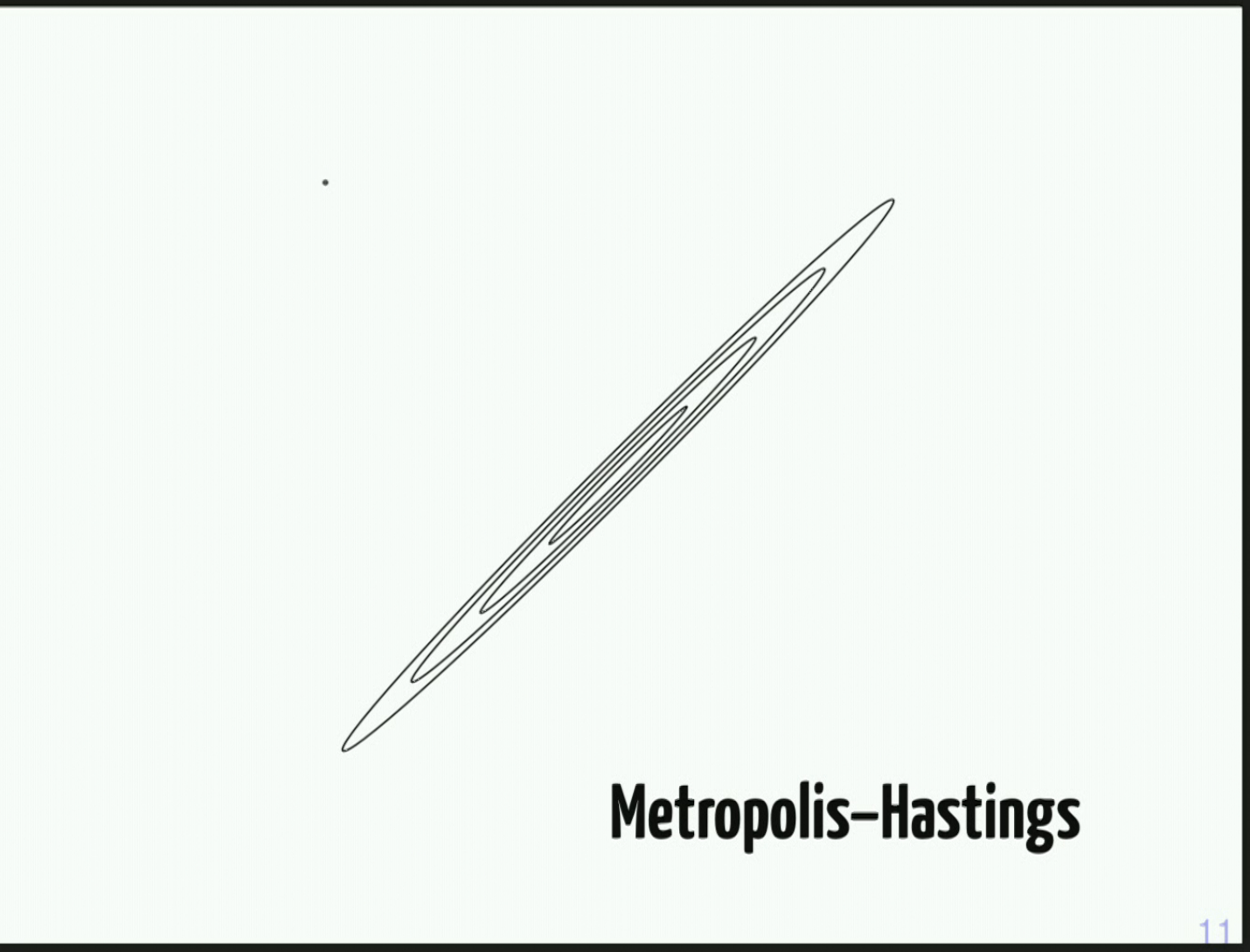
- move a “particle” or “walker” around in the parameter space θ, based on the value of the probability function (for us: the posterior probability $P(\theta|X)$)
- choose new parameters according to a *proposal distribution* – often Gaussian: $\theta_{new} = \theta + \mathcal{N}(0, \Sigma)$
- compute probability at new θ_{new} and decide whether to move (“jump”) to new parameters
- *always* keep improvements, *sometimes* keep decreases
- in the long run, the set of walker positions $\{\theta_1, \theta_2, \theta_3, ...\}$ are your samples

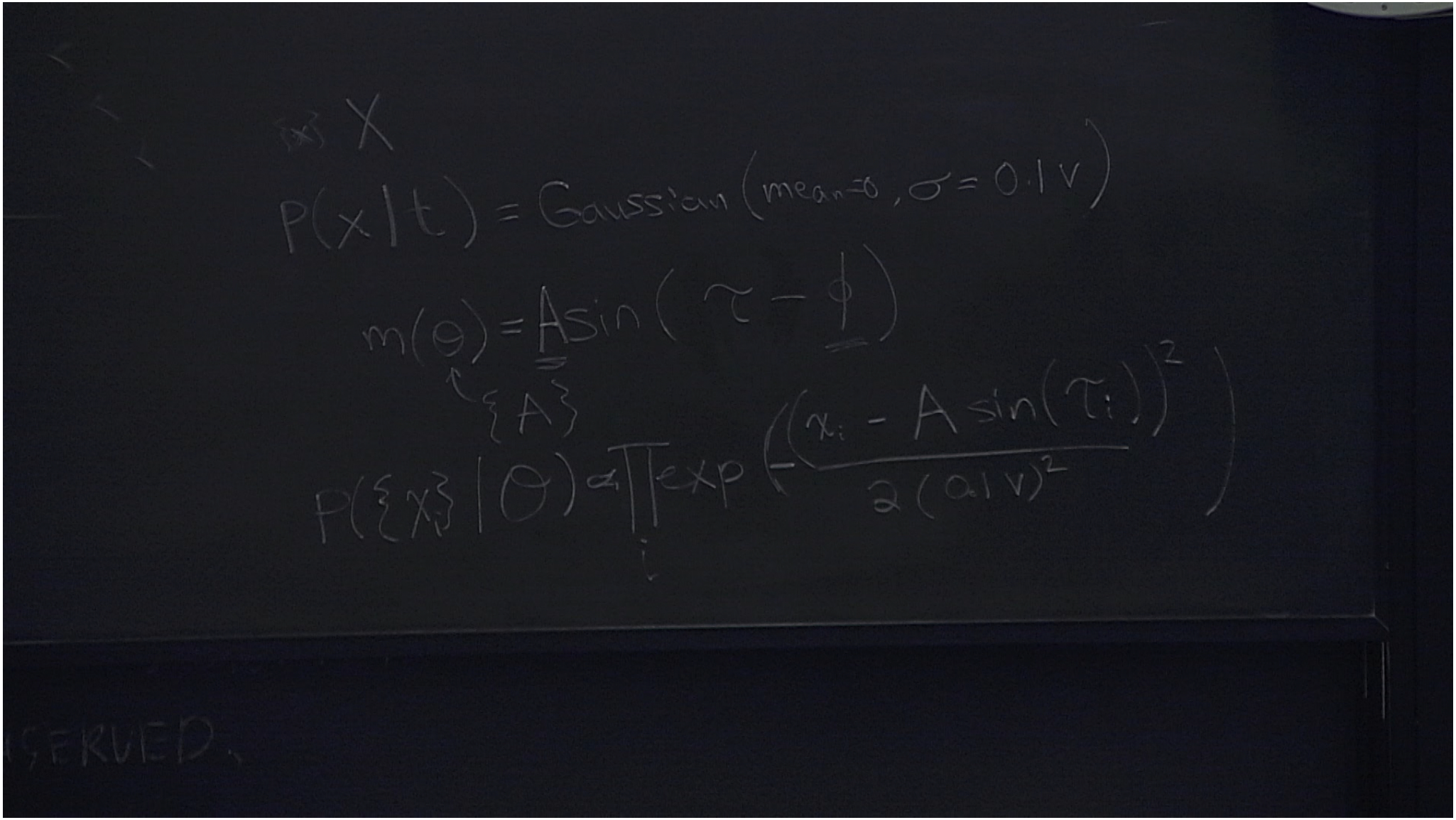
Finally, M–H MCMC code

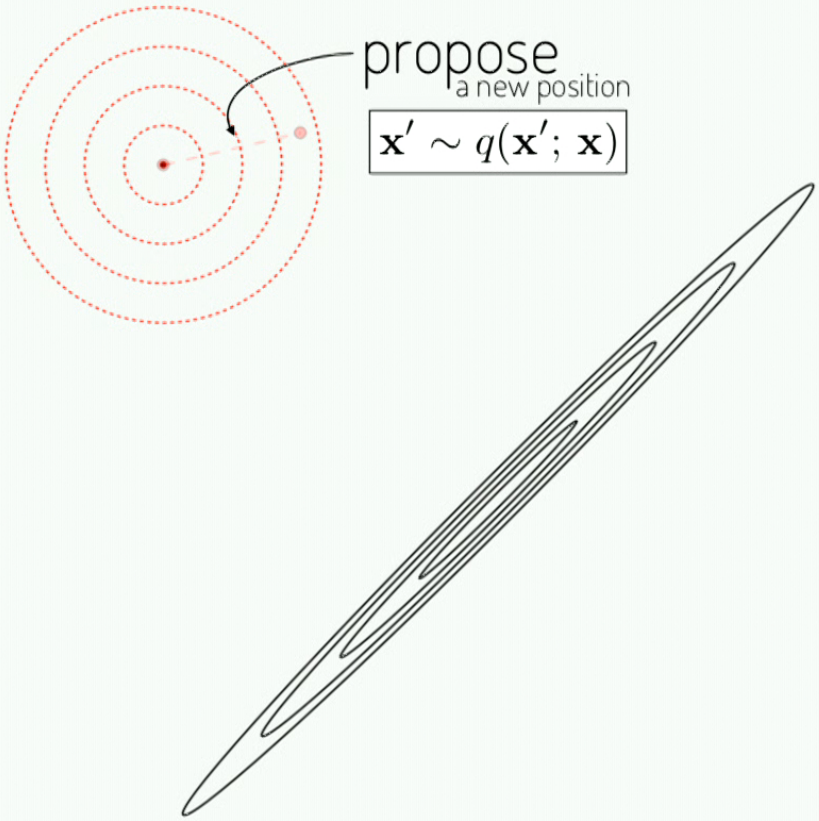
```
fun MCMC(prob_function, sample_proposal, params, Nsteps):
    chain = []
    prob = prob_function(params)
    for i in 1 to Nsteps:
        # sample from proposal distribution
        params_new = sample_proposal(params)
        prob_new = prob_function(params_new)
        # accept?
        if ((prob_new > prob) OR
            (prob_new / prob) > uniform_random()):
            params = params_new
            prob = prob_new
        chain.append(params)
    return chain
```

Demo

Credit: Dan Foreman–Mackey
https://speakerdeck.com/dfm/data-analysis-with-mcmc


Metropolis–Hastings
11

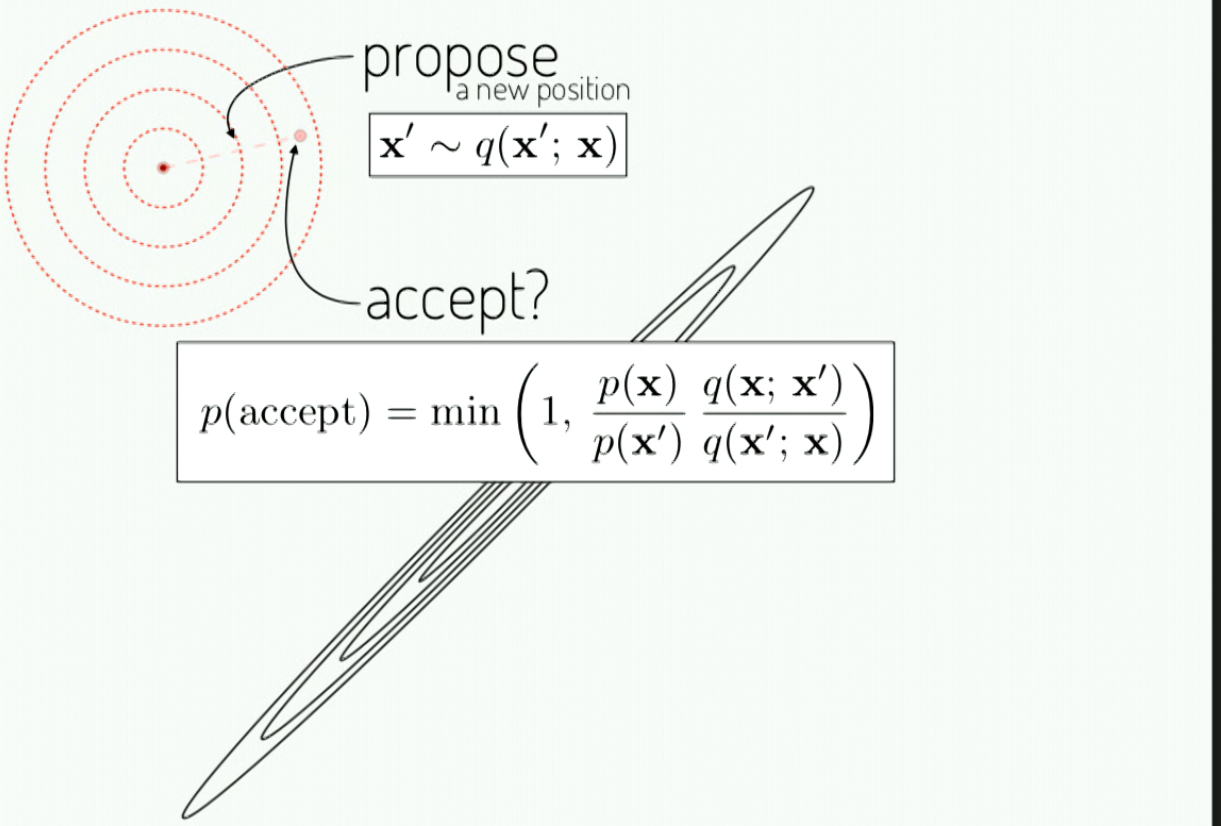

X
P(x|t) = Gaussian (mean=0, σ = 0.1 V)
m(θ) = A sin(τ − φ)
{A}
P({x}|θ) ∝ ∏ exp(−((x_i − A sin(τ_i))^2 / 2(0.1 V)^2))
i



Metropolis–Hastings

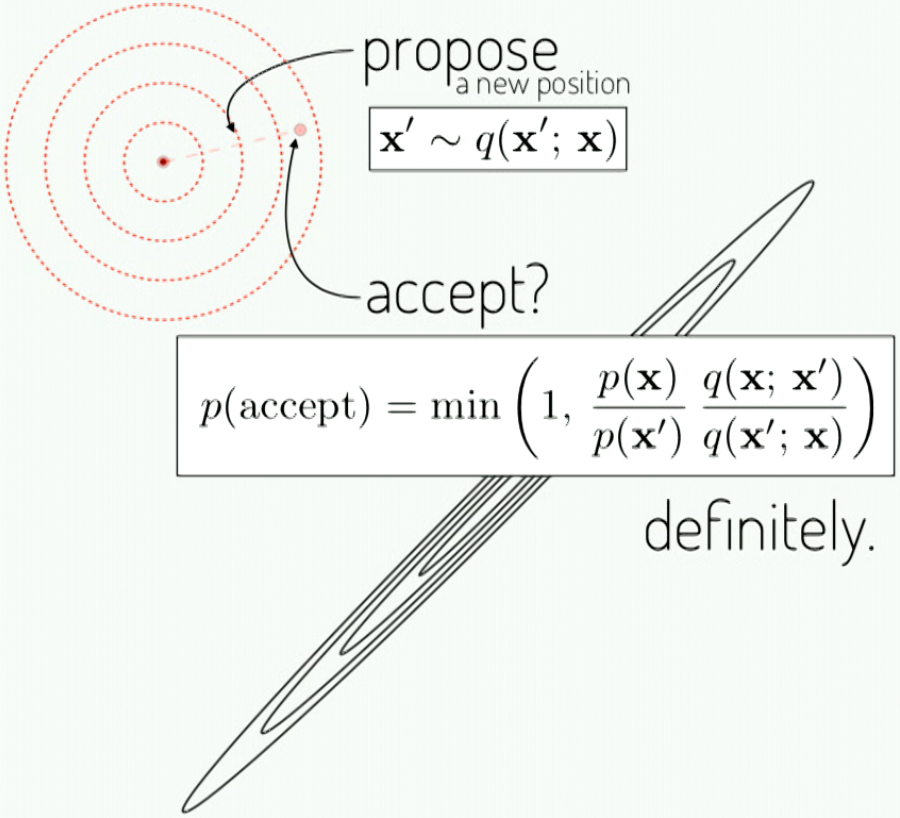
in an ideal world

14



Metropolis–Hastings

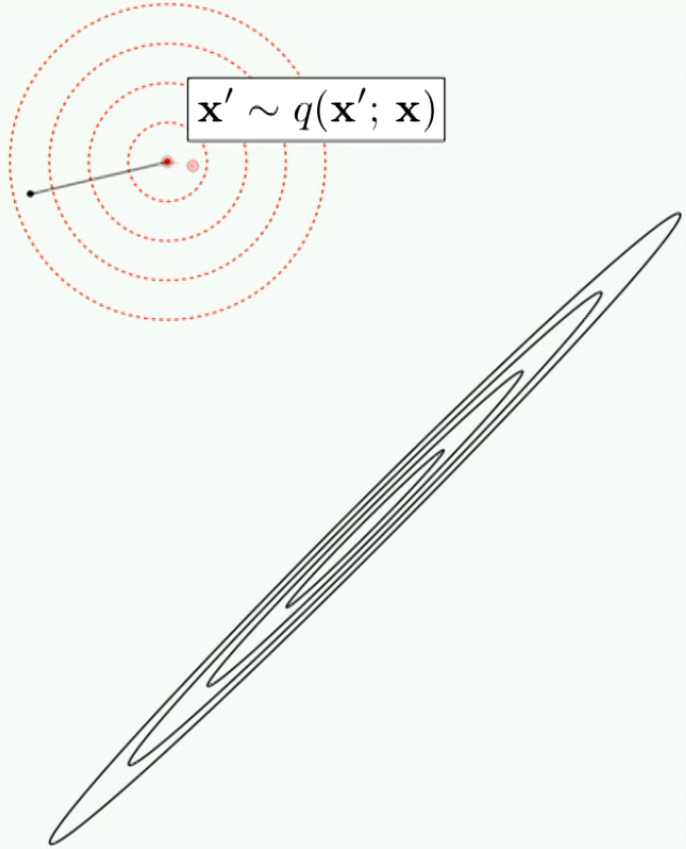
in an ideal world



Metropolis–Hastings

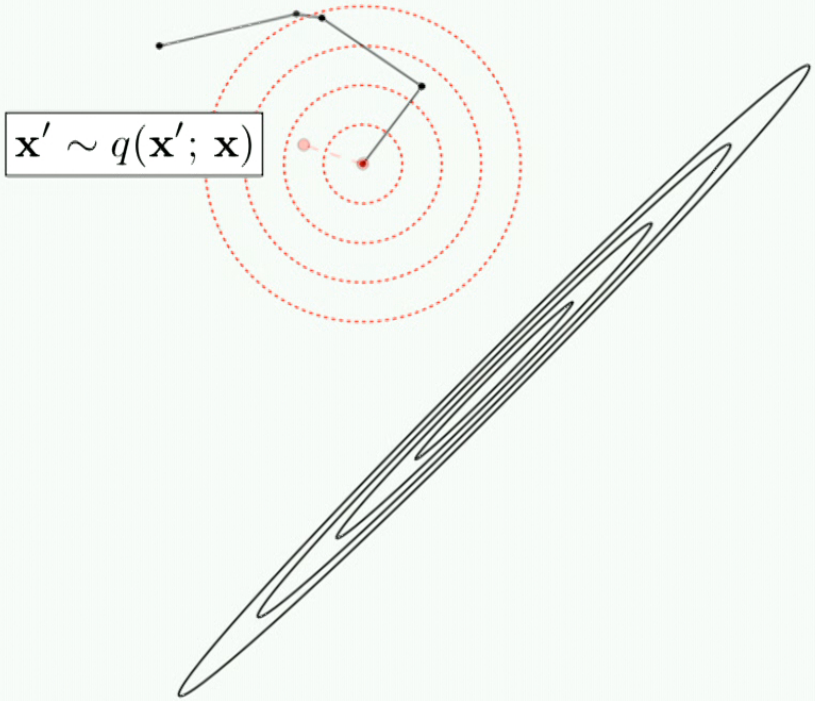
in an ideal world

16



Metropolis–Hastings

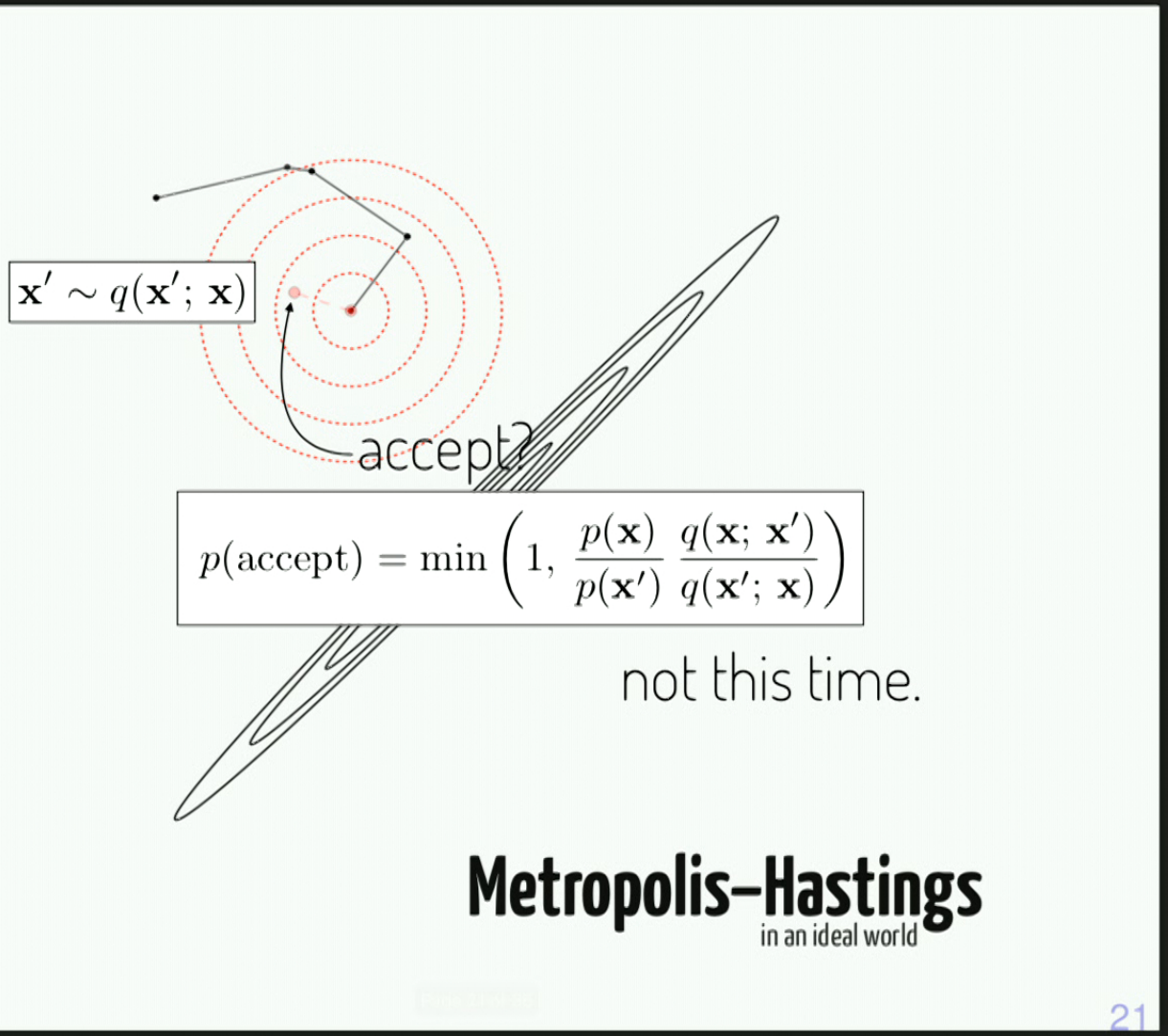
in an ideal world

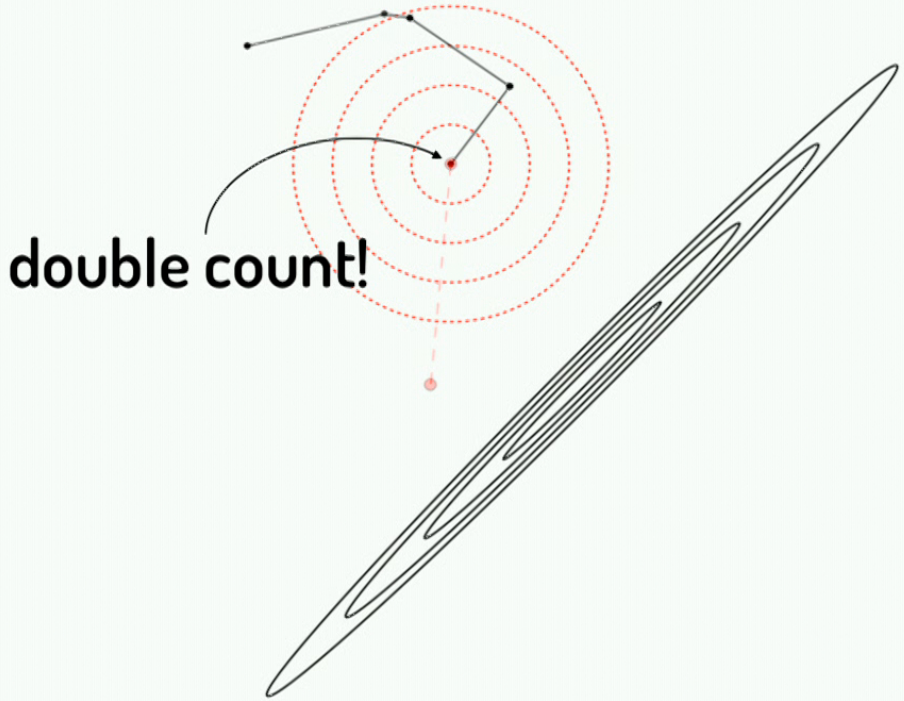


Metropolis–Hastings

in an ideal world

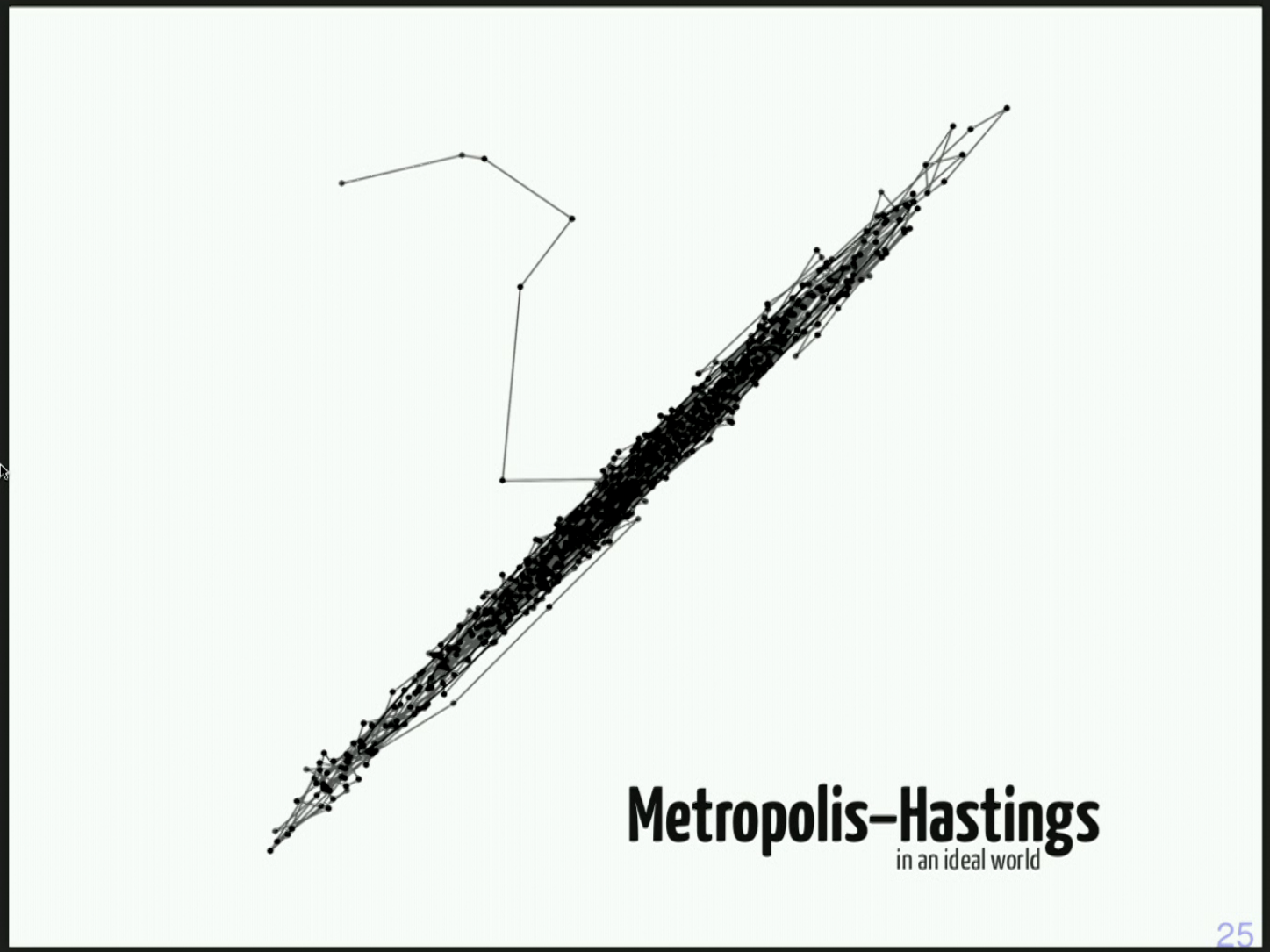
19

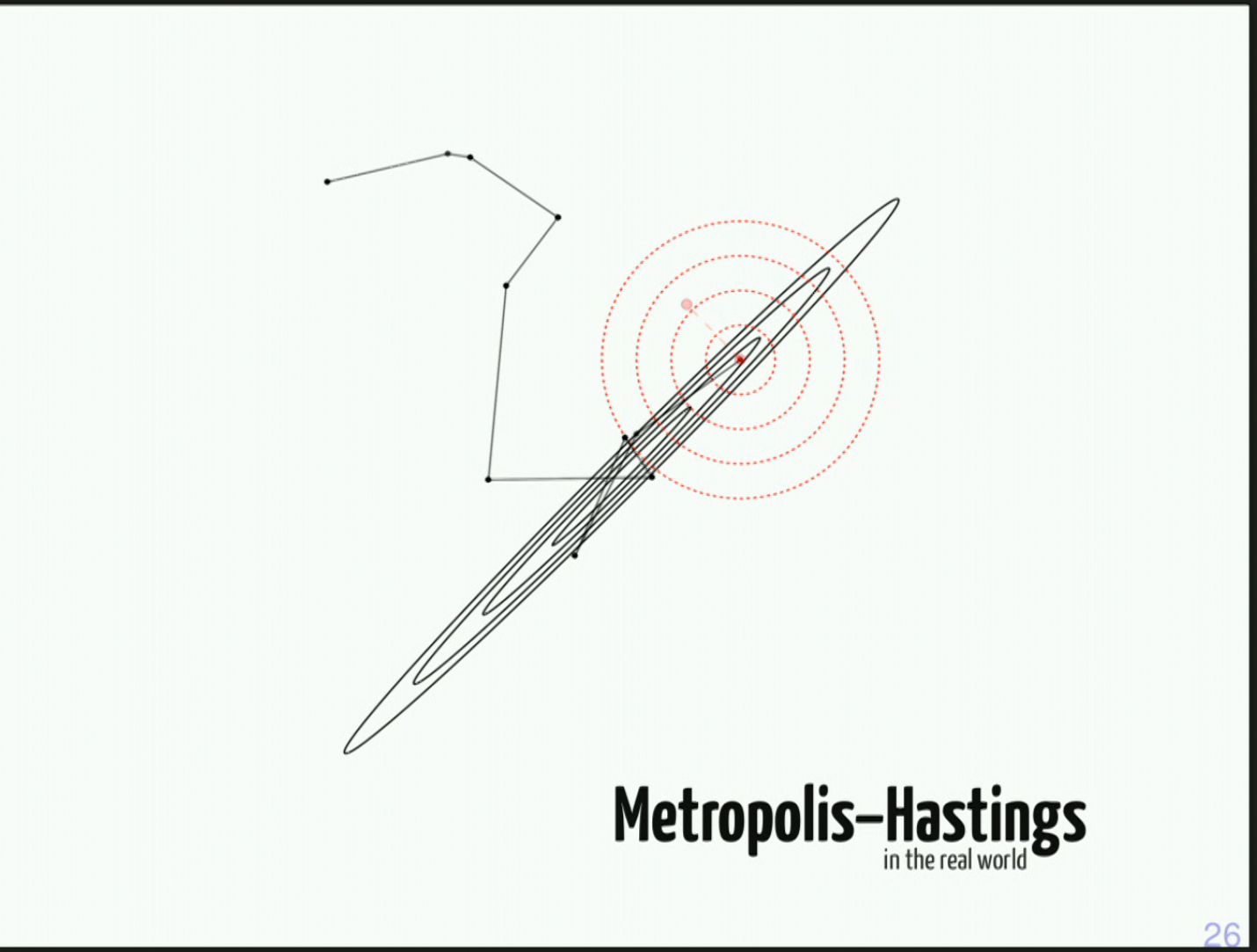

$\mathbf{x}' \sim q(\mathbf{x}'; \mathbf{x})$
accept?
$p(\text{accept}) = \min\left(1, \frac{p(\mathbf{x})}{p(\mathbf{x}')} \frac{q(\mathbf{x}; \mathbf{x}')}{q(\mathbf{x}'; \mathbf{x})}\right)$
not this time.
Metropolis–Hastings
in an ideal world
21

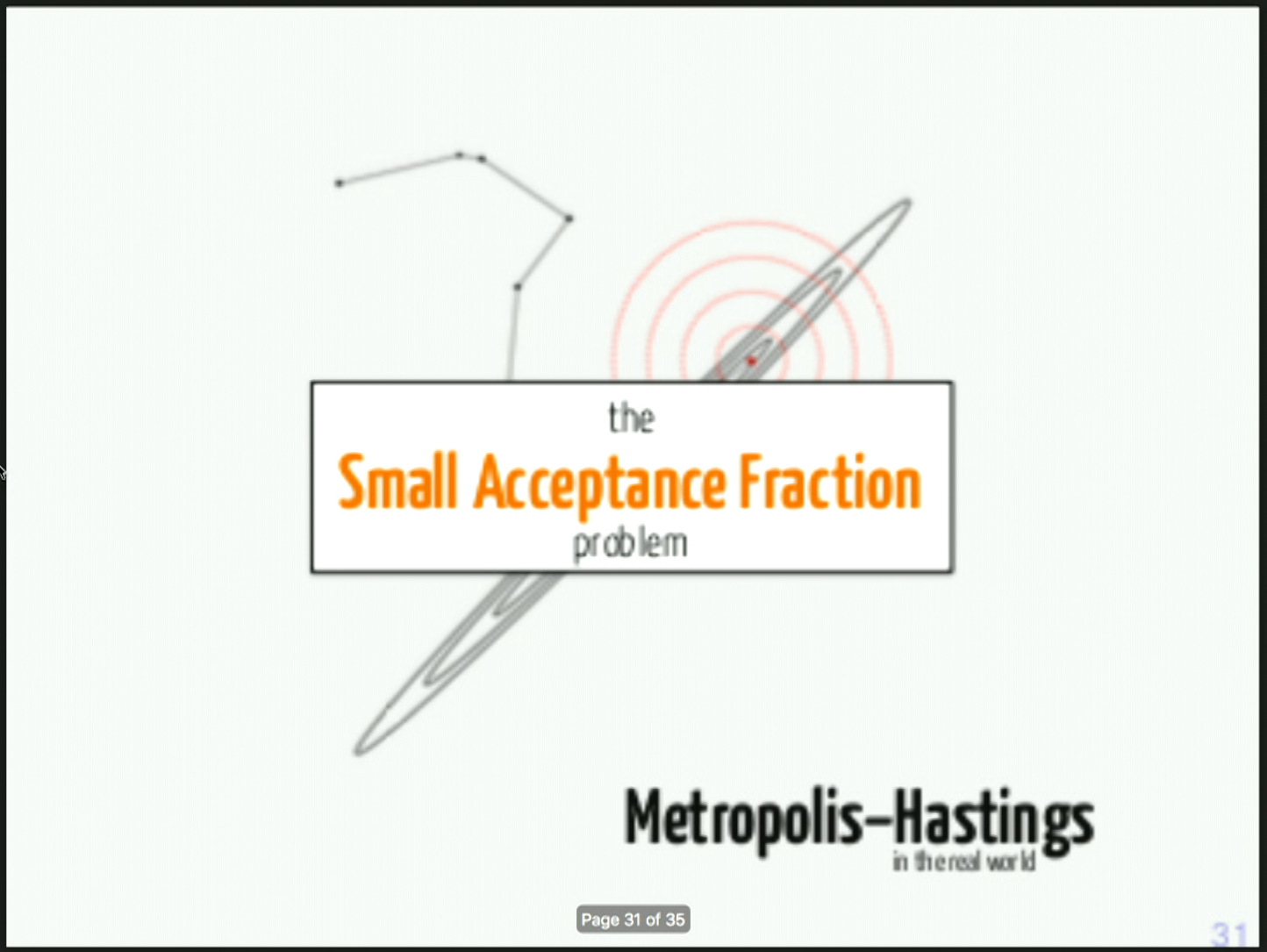


Metropolis–Hastings

in an ideal world

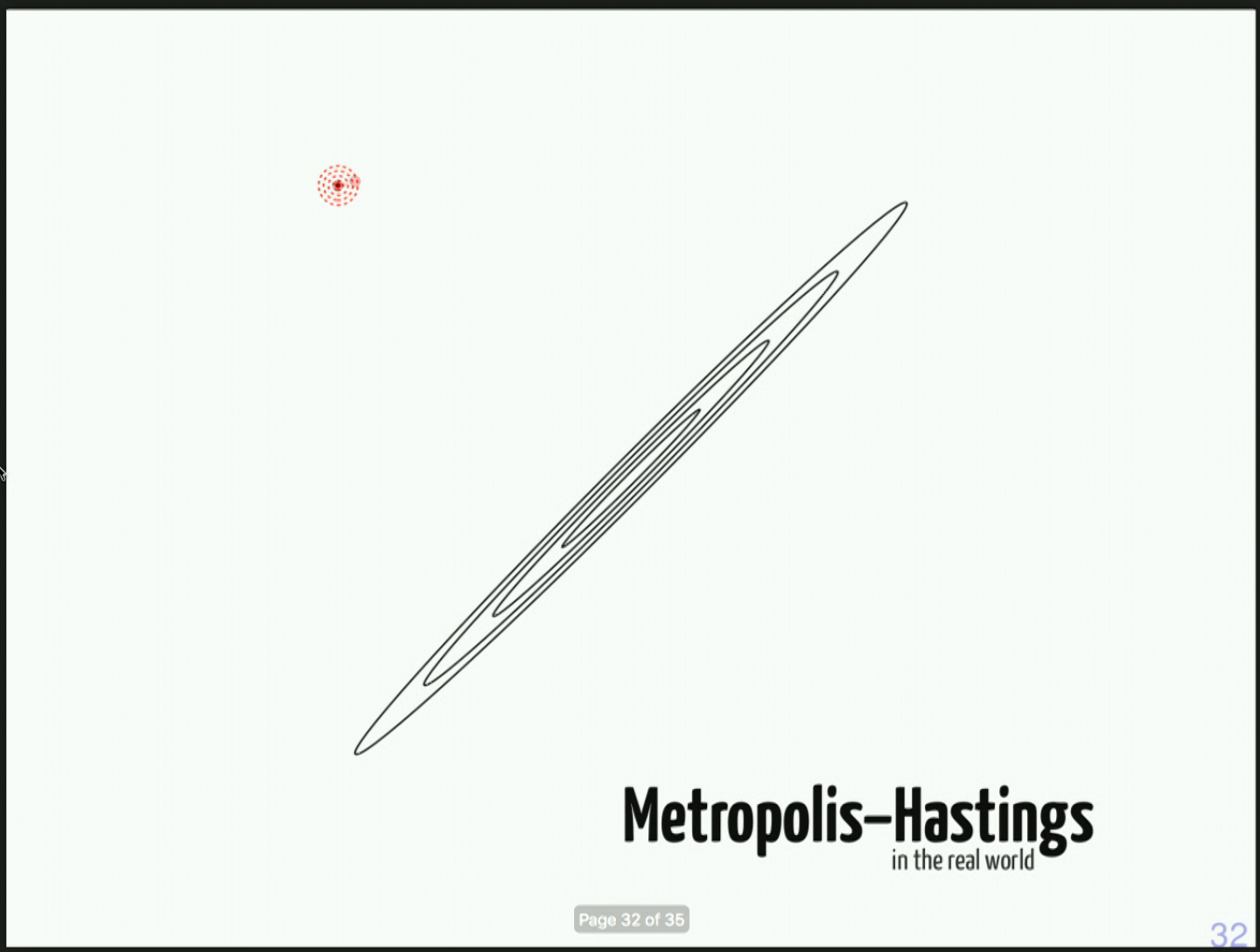

Metropolis–Hastings
in an ideal world
25

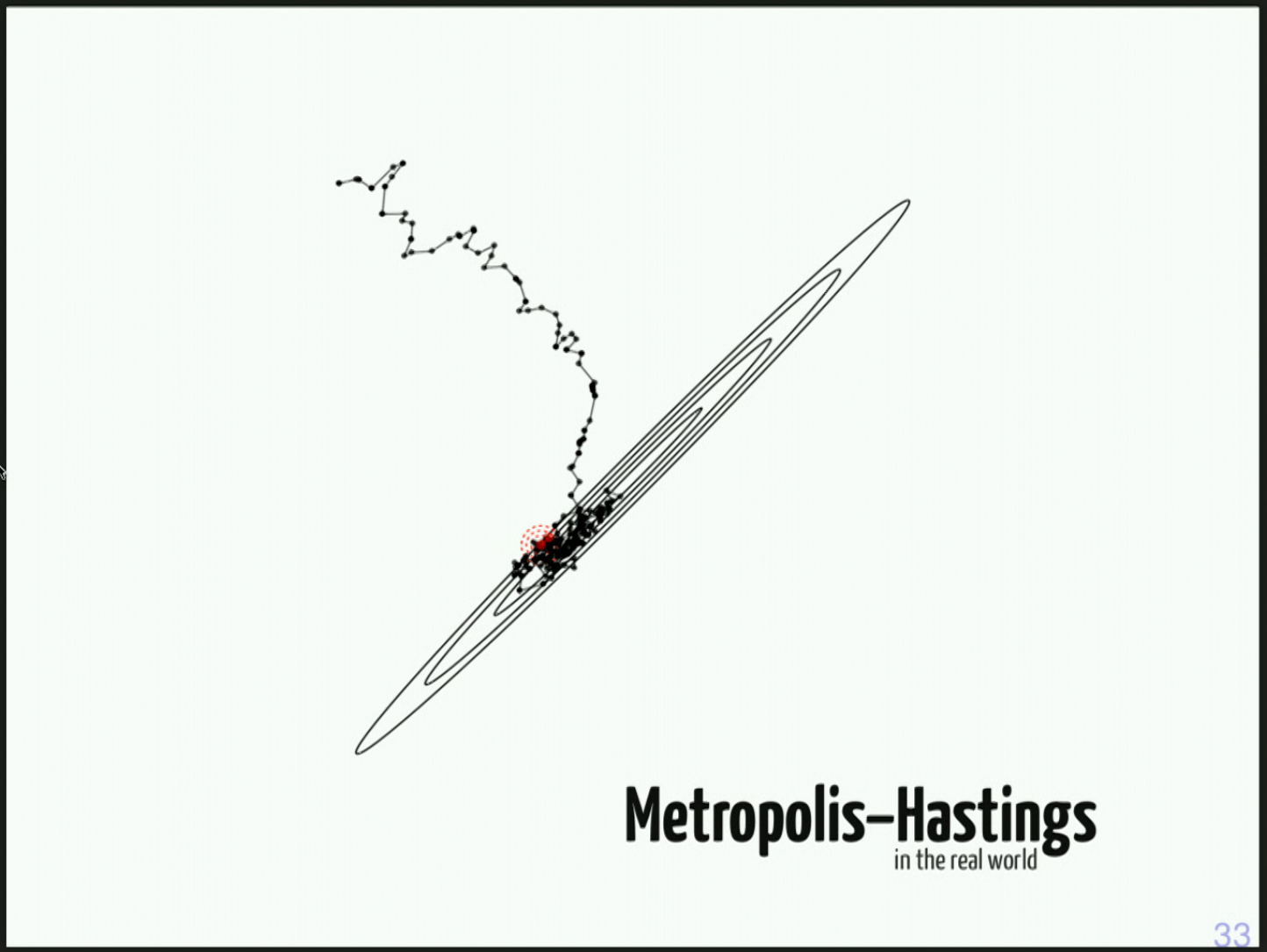

Metropolis–Hastings
in the real world
26


the
Small Acceptance Fraction
problem
Metropolis–Hastings
in the real world
Page 31 of 35
31

Metropolis–Hastings
in the real world

31


Metropolis–Hastings
in the real world
Page 32 of 35
32


Metropolis–Hastings
in the real world
33

Metropolis–Hastings
in the real world

34

https://symmetry1/user/dlang/lab

File Edit View Run Kernel Tabs Settings Help

/ ··· / dlang / mcmc /

Name	Last Modified
mcmc.ipynb	an hour ago
p99-data.txt	an hour ago
perlmutter99-julia-fill...	an hour ago
Untitled.ipynb	an hour ago

mcmc.ipynb | perlmutter99-julia-filled.ip | Untitled.ipynb

Markdown Julia 1.2.0

Imagine we are measuring voltages of a sine wave, and we want to constrain the amplitude and phase.

Our measurements have Gaussian noise of known standard deviation, and occur at known times (assume we can measure time so well that there is practically no uncertainty).

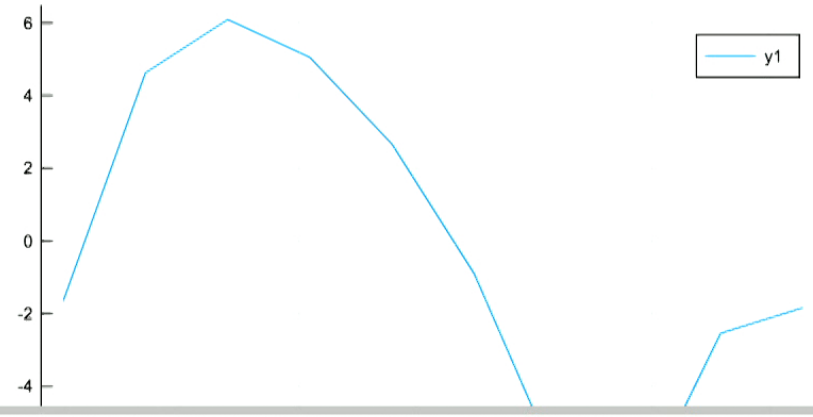
[18]:
```
# "y" are our measurements
y = Array([-1.64, 4.63, 6.09, 5.05, 2.66, -0.90, -6.09, -7.16, -2.54, -1.84])
# "noise_std" is the standard deviation of the Gaussian noise on our measurements
noise_std = 1.0

nsamples = length(y)
# "times" are when the measurements were taken (evenly spaced)
times = collect(LinRange(0.0, 2.0*pi, nsamples));
```

[6]:
```
using Plots
```

[7]:
```
plot(times, y)
```

[7]:



0 5 Julia 1.2.0 | Idle Mode: Command Ln 1, Col 103 mcmc.ipynb

Set status

Dustin Lang

dstndstn

Edit profile

Perimeter Institute (@PerimeterInsti...

Waterloo, Canada

http://dstn.astrometry.net

Organizations

Overview | **Repositories 71** | Projects 0 | Packages 0 | Stars 9 | Followers 123 | Following 7

Find a repository... | Type: All | Language: All | New

mcmc-notes — Star

TeX Updated 20 hours ago

unwise.me — Star

Web service powering unwise.me

Python 1 Updated yesterday

tractor — Star

The Tractor: measuring astronomical sources via probabilistic inference

Python 37 14 Other Updated 4 days ago

astrometry.net — Star

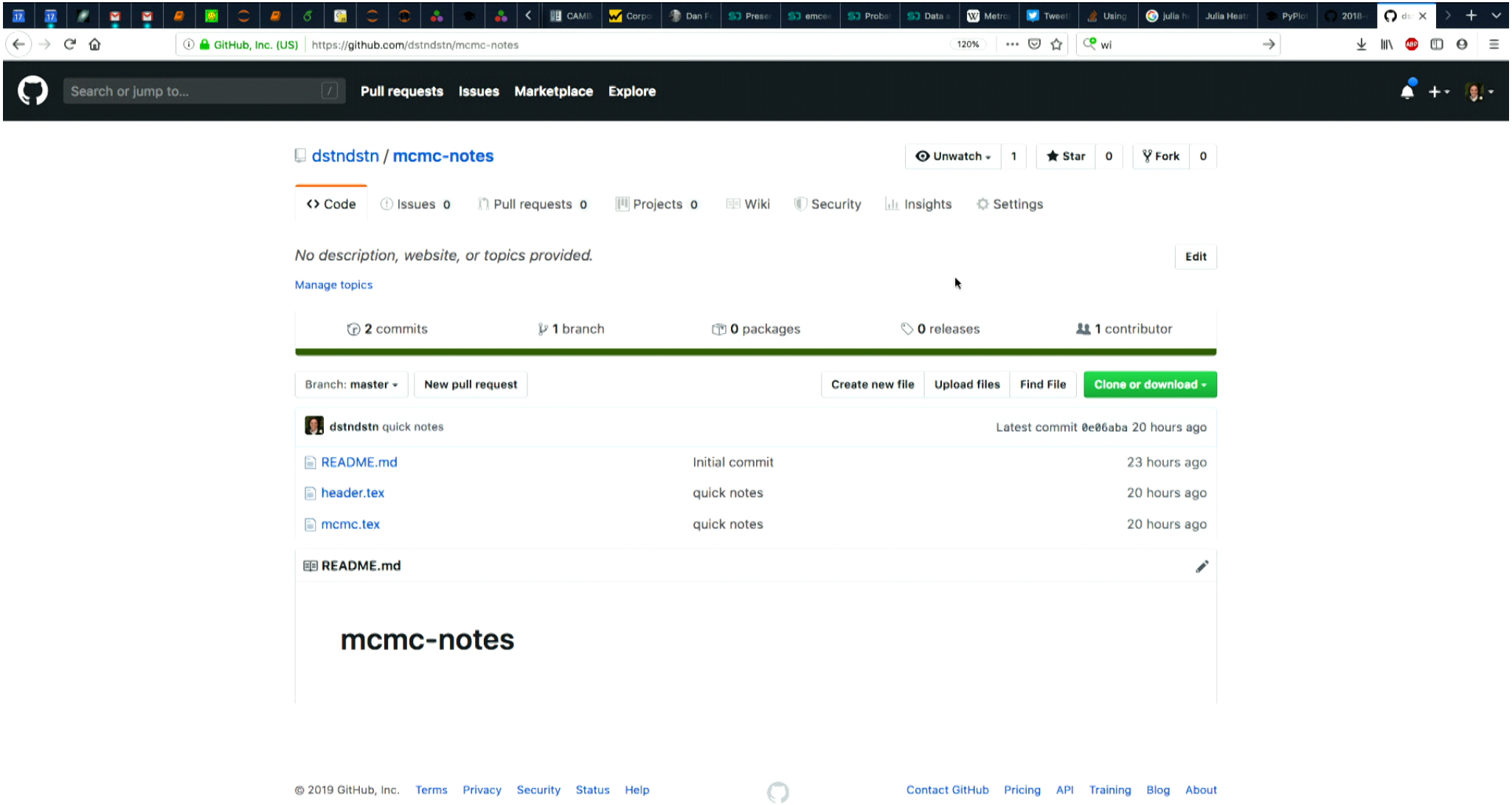
Astrometry.net -- automatic recognition of astronomical images

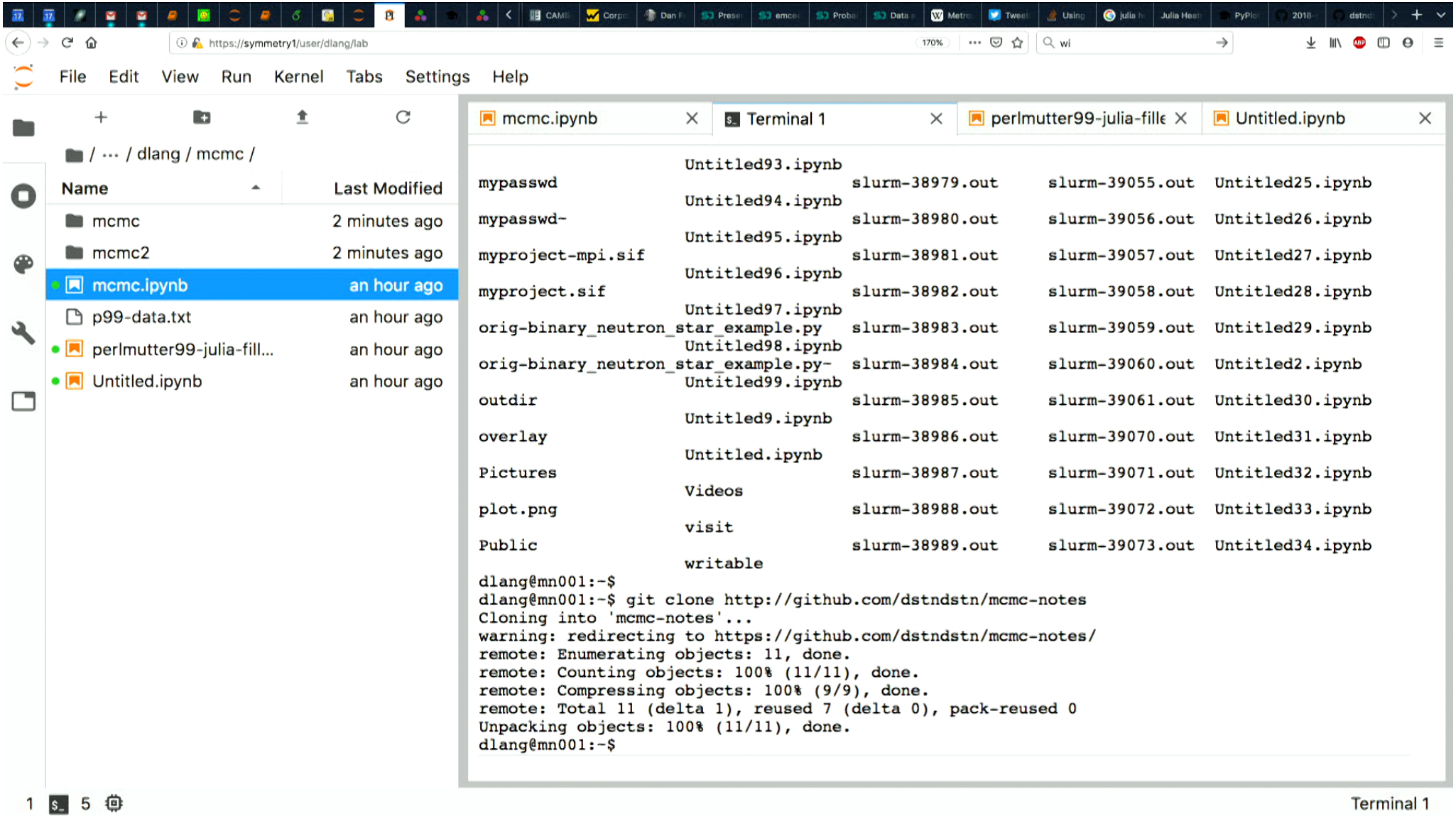
C 270 98 BSD 3-Clause "New" or "Revised" License Updated 11 days ago

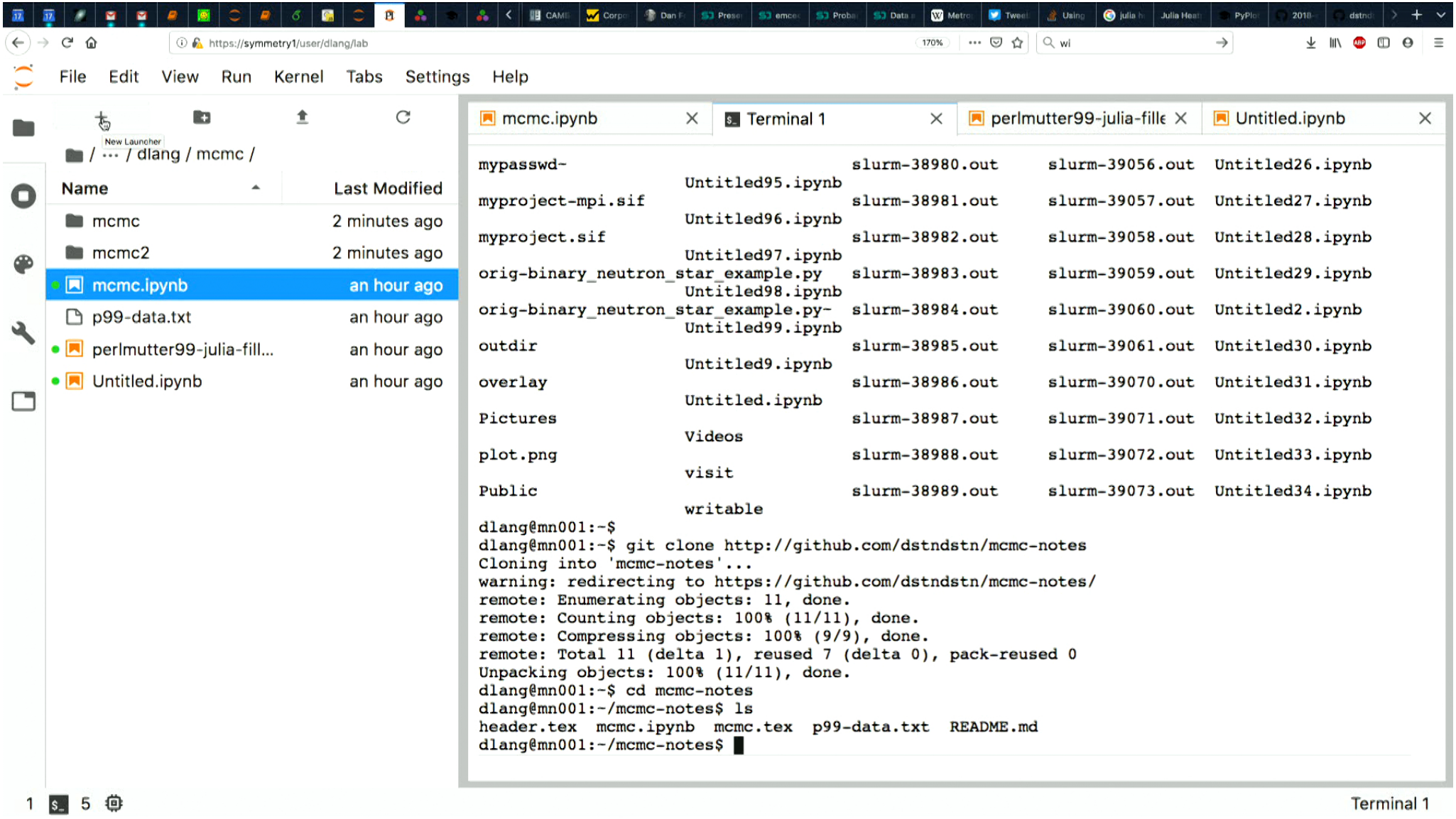
Leaflet — Star

Forked from Leaflet/Leaflet

JavaScript library for mobile-friendly interactive maps


GitHub, Inc. (US)
https://github.com/dstndstn/mcmc-notes
Search or jump to...
Pull requests
Issues
Marketplace
Explore
dstndstn / mcmc-notes
Unwatch 1
Star 0
Fork 0
Code
Issues 0
Pull requests 0
Projects 0
Wiki
Security
Insights
Settings
No description, website, or topics provided.
Edit
Manage topics
2 commits
1 branch
0 packages
0 releases
1 contributor
Branch: master
New pull request
Create new file
Upload files
Find File
Clone or download
dstndstn quick notes
Latest commit 0e06aba 20 hours ago
README.md
Initial commit
23 hours ago
header.tex
quick notes
20 hours ago
mcmc.tex
quick notes
20 hours ago
README.md
mcmc-notes
© 2019 GitHub, Inc.
Terms
Privacy
Security
Status
Help
Contact GitHub
Pricing
API
Training
Blog
About


https://symmetry1/user/dlang/lab
File Edit View Run Kernel Tabs Settings Help
/ ··· / dlang / mcmc /
Name
Last Modified
mcmc 2 minutes ago
mcmc2 2 minutes ago
mcmc.ipynb an hour ago
p99-data.txt an hour ago
perlmutter99-julia-fill... an hour ago
Untitled.ipynb an hour ago
mcmc.ipynb
Terminal 1
perlmutter99-julia-fille
Untitled.ipynb
Untitled93.ipynb
mypasswd
Untitled94.ipynb
mypasswd-
Untitled95.ipynb
myproject-mpi.sif
Untitled96.ipynb
myproject.sif
Untitled97.ipynb
orig-binary_neutron_star_example.py
Untitled98.ipynb
orig-binary_neutron_star_example.py-
Untitled99.ipynb
outdir
Untitled9.ipynb
overlay
Untitled.ipynb
Pictures
Videos
plot.png
visit
Public
writable
slurm-38979.out
slurm-38980.out
slurm-38981.out
slurm-38982.out
slurm-38983.out
slurm-38984.out
slurm-38985.out
slurm-38986.out
slurm-38987.out
slurm-38988.out
slurm-38989.out
slurm-39055.out
slurm-39056.out
slurm-39057.out
slurm-39058.out
slurm-39059.out
slurm-39060.out
slurm-39061.out
slurm-39070.out
slurm-39071.out
slurm-39072.out
slurm-39073.out
Untitled25.ipynb
Untitled26.ipynb
Untitled27.ipynb
Untitled28.ipynb
Untitled29.ipynb
Untitled2.ipynb
Untitled30.ipynb
Untitled31.ipynb
Untitled32.ipynb
Untitled33.ipynb
Untitled34.ipynb
dlang@mn001:~$
dlang@mn001:~$ git clone http://github.com/dstndstn/mcmc-notes
Cloning into 'mcmc-notes'...
warning: redirecting to https://github.com/dstndstn/mcmc-notes/
remote: Enumerating objects: 11, done.
remote: Counting objects: 100% (11/11), done.
remote: Compressing objects: 100% (9/9), done.
remote: Total 11 (delta 1), reused 7 (delta 0), pack-reused 0
Unpacking objects: 100% (11/11), done.
dlang@mn001:~$
1 5
Terminal 1


https://symmetry1/user/dlang/lab
File Edit View Run Kernel Tabs Settings Help
New Launcher
/ ··· / dlang / mcmc /
Name
Last Modified
mcmc 2 minutes ago
mcmc2 2 minutes ago
mcmc.ipynb an hour ago
p99-data.txt an hour ago
perlmutter99-julia-fill... an hour ago
Untitled.ipynb an hour ago
mcmc.ipynb
Terminal 1
perlmutter99-julia-fille
Untitled.ipynb
mypasswd- Untitled95.ipynb slurm-38980.out slurm-39056.out Untitled26.ipynb
myproject-mpi.sif Untitled96.ipynb slurm-38981.out slurm-39057.out Untitled27.ipynb
myproject.sif Untitled97.ipynb slurm-38982.out slurm-39058.out Untitled28.ipynb
orig-binary_neutron_star_example.py Untitled98.ipynb slurm-38983.out slurm-39059.out Untitled29.ipynb
orig-binary_neutron_star_example.py- Untitled99.ipynb slurm-38984.out slurm-39060.out Untitled2.ipynb
outdir Untitled9.ipynb slurm-38985.out slurm-39061.out Untitled30.ipynb
overlay Untitled.ipynb slurm-38986.out slurm-39070.out Untitled31.ipynb
Pictures Videos slurm-38987.out slurm-39071.out Untitled32.ipynb
plot.png visit slurm-38988.out slurm-39072.out Untitled33.ipynb
Public writable slurm-38989.out slurm-39073.out Untitled34.ipynb
dlang@mn001:~$
dlang@mn001:~$ git clone http://github.com/dstndstn/mcmc-notes
Cloning into 'mcmc-notes'...
warning: redirecting to https://github.com/dstndstn/mcmc-notes/
remote: Enumerating objects: 11, done.
remote: Counting objects: 100% (11/11), done.
remote: Compressing objects: 100% (9/9), done.
remote: Total 11 (delta 1), reused 7 (delta 0), pack-reused 0
Unpacking objects: 100% (11/11), done.
dlang@mn001:~$ cd mcmc-notes
dlang@mn001:~/mcmc-notes$ ls
header.tex mcmc.ipynb mcmc.tex p99-data.txt README.md
dlang@mn001:~/mcmc-notes$
1 5
Terminal 1

https://symmetry1/user/dlang/lab

File Edit View Run Kernel Tabs Settings Help

/ ··· / dlang / mcmc-notes /

Name	Last Modified
header.tex	seconds ago
mcmc.ipynb	seconds ago
mcmc.tex	seconds ago
p99-data.txt	seconds ago
README.md	seconds ago

mcmc.ipynb | Terminal 1 | mcmc.ipynb | perlmutter99-ju | Untitled.ipynb

Markdown Julia 1.2.0

Imagine we are measuring voltages of a sine wave, and we want to constrain the amplitude and phase.

Our measurements have Gaussian noise of known standard deviation, and occur at known times (assume we can measure time so well that there is practically no uncertainty).

[18]:
```
# "y" are our measurements
y = Array([-1.64, 4.63, 6.09, 5.05, 2.66, -0.90, -6.09, -7.16, -2.54, -1.84])
# "noise_std" is the standard deviation of the Gaussian noise on our measurements
noise_std = 1.0

nsamples = length(y)
# "times" are when the measurements were taken (evenly spaced)
times = collect(LinRange(0.0, 2.0*pi, nsamples));
```

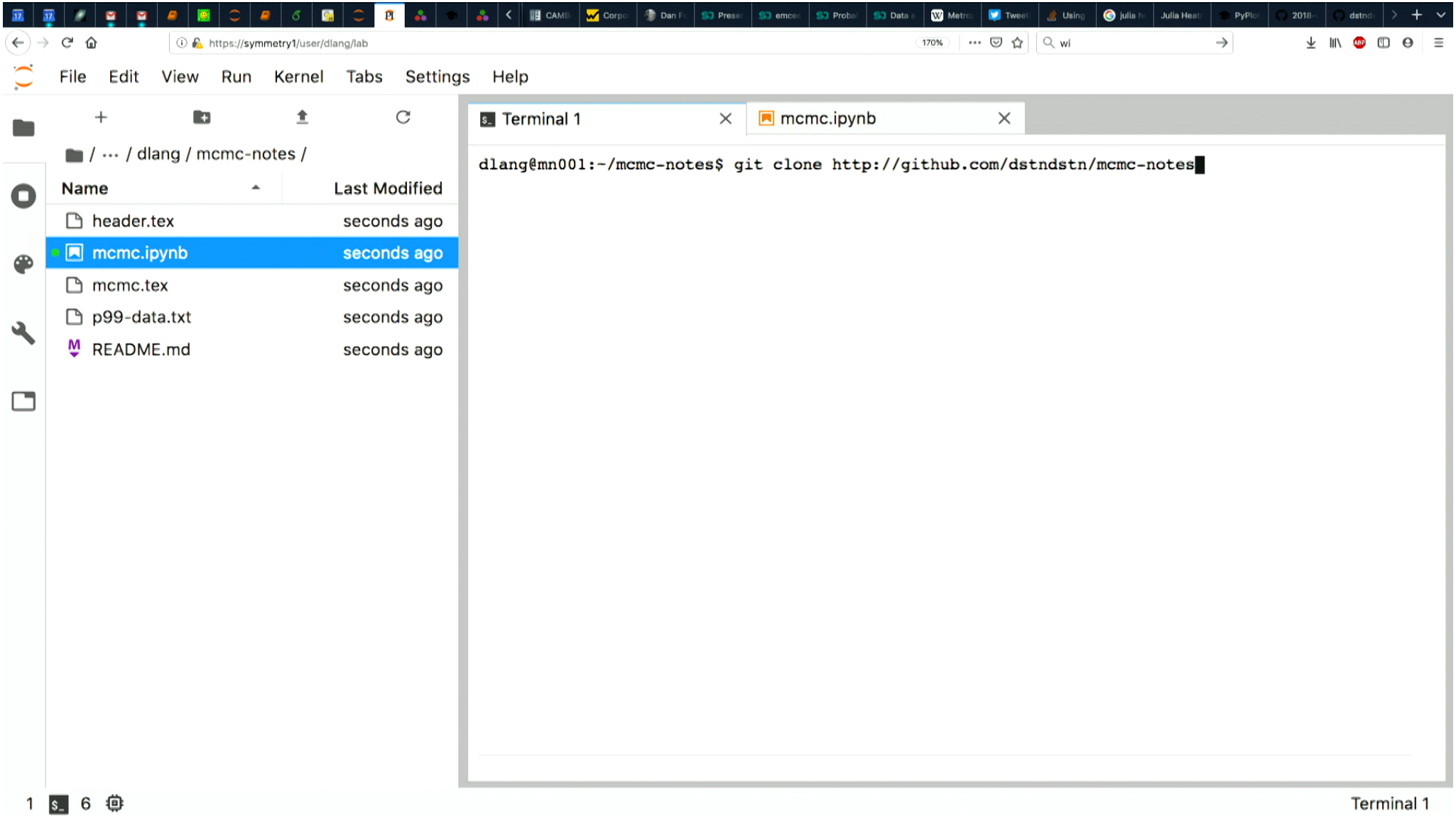
[6]:
```
using Plots
```

[7]:
```
plot(times, y)
```

[7]:

1 6 No Kernel! | Starting

Mode: Command Ln 1, Col 1 mcmc.ipynb


https://symmetry1/user/dlang/lab
File Edit View Run Kernel Tabs Settings Help
/ ··· / dlang / mcmc-notes /
Name
Last Modified
header.tex seconds ago
mcmc.ipynb seconds ago
mcmc.tex seconds ago
p99-data.txt seconds ago
README.md seconds ago
Terminal 1
mcmc.ipynb
dlang@mn001:~/mcmc-notes$ git clone http://github.com/dstndstn/mcmc-notes
1 6
Terminal 1

https://symmetry1/user/dlang/lab

File Edit View Run Kernel Tabs Settings Help

/ ··· / dlang / mcmc-notes /

Name	Last Modified
header.tex	2 minutes ago
mcmc.ipynb	2 minutes ago
mcmc.tex	2 minutes ago
p99-data.txt	2 minutes ago
README.md	2 minutes ago

Terminal 1 | mcmc.ipynb

Markdown | Julia 1.2.0

Imagine we are measuring voltages of a sine wave, and we want to constrain the amplitude and phase.

Our measurements have Gaussian noise of known standard deviation, and occur at known times (assume we can measure time so well that there is practically no uncertainty).

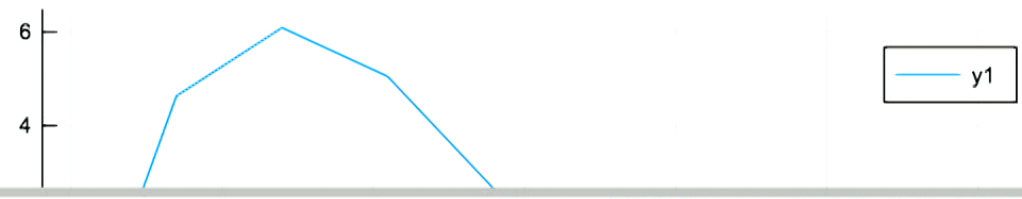
[18]:
```
# "y" are our measurements
y = Array([-1.64, 4.63, 6.09, 5.05, 2.66, -0.90, -6.09, -7.16, -2.54, -1.84])
# "noise_std" is the standard deviation of the Gaussian noise on our measurements
noise_std = 1.0

nsamples = length(y)
# "times" are when the measurements were taken (evenly spaced)
times = collect(LinRange(0.0, 2.0*pi, nsamples));
```

[6]:
```
using Plots
```

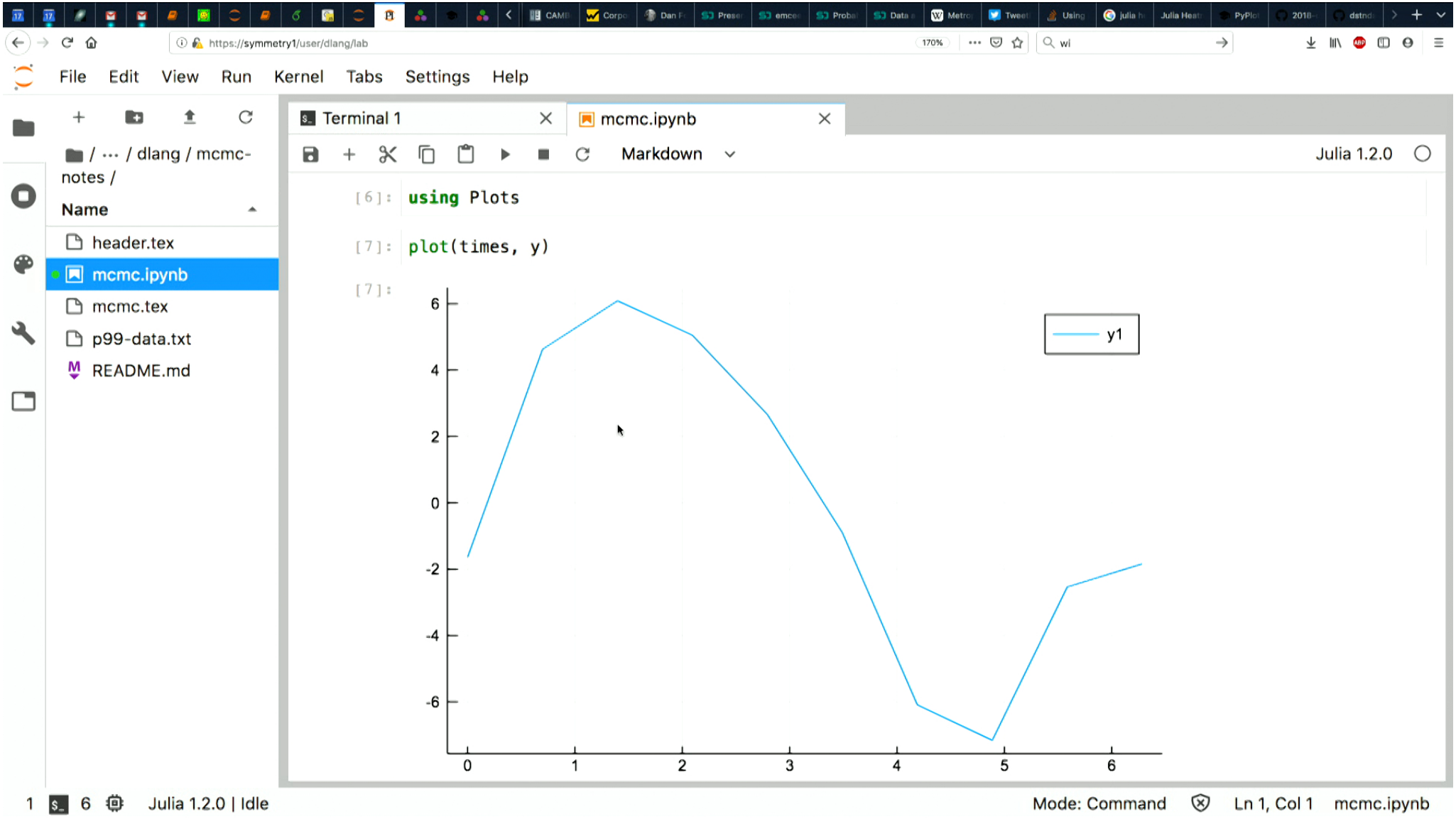
[7]:
```
plot(times, y)
```

[7]:



1 6 Julia 1.2.0 | Idle

Mode: Command Ln 1, Col 1 mcmc.ipynb


https://symmetry1/user/dlang/lab
File Edit View Run Kernel Tabs Settings Help
Terminal 1
mcmc.ipynb
/ ··· / dlang / mcmc-notes /
Name
header.tex
mcmc.ipynb
mcmc.tex
p99-data.txt
README.md
Markdown
Julia 1.2.0
[6]: using Plots
[7]: plot(times, y)
[7]:
y1
6
4
2
0
-2
-4
-6
0
1
2
3
4
5
6
1 6 Julia 1.2.0 | Idle
Mode: Command
Ln 1, Col 1
mcmc.ipynb

https://symmetry1/user/dlang/lab

File Edit View Run Kernel Tabs Settings Help

/ ··· / dlang / mcmc-notes /

Name

- header.tex
- mcmc.ipynb
- mcmc.tex
- p99-data.txt
- README.md

Terminal 1 | mcmc.ipynb

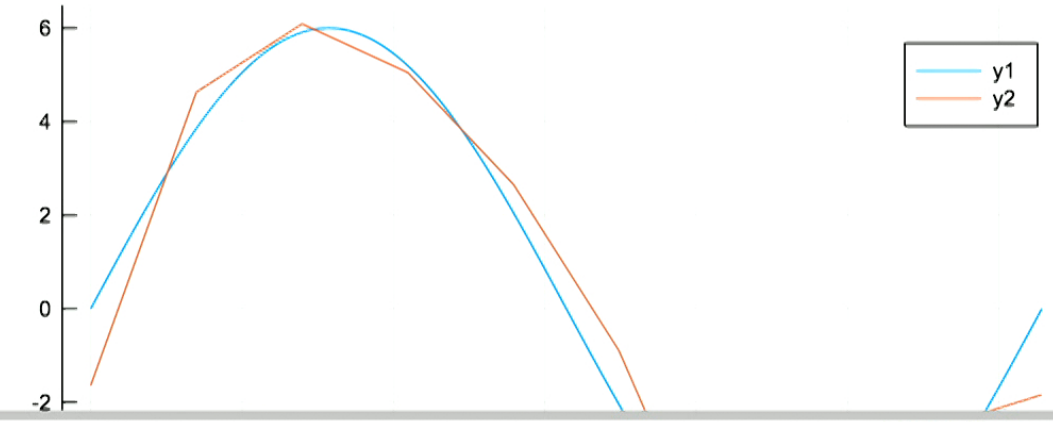
Markdown | Julia 1.2.0

Now, we have to write our *model* function. Our model is that this is a sine wave with parameters of *phase offset* and *amplitude*.

[15]:
```
function sine_model(t, phase, amplitude)
    return amplitude * sin.(t .- phase)
end;
```

[17]:
```
tplot = collect(LinRange(0.0, 2.0*pi, 200));
# I'm going to guess at parameters: phase = 0, amplitude = 6
plot(tplot, sine_model(tplot, 0., 6))
plot!(times, y)
```

[17]:



1 6 Julia 1.2.0 | Idle

Mode: Command Ln 1, Col 1 mcmc.ipynb

https://symmetry1/user/dlang/lab

File Edit View Run Kernel Tabs Settings Help

/ ··· / dlang / mcmc-notes /

Name

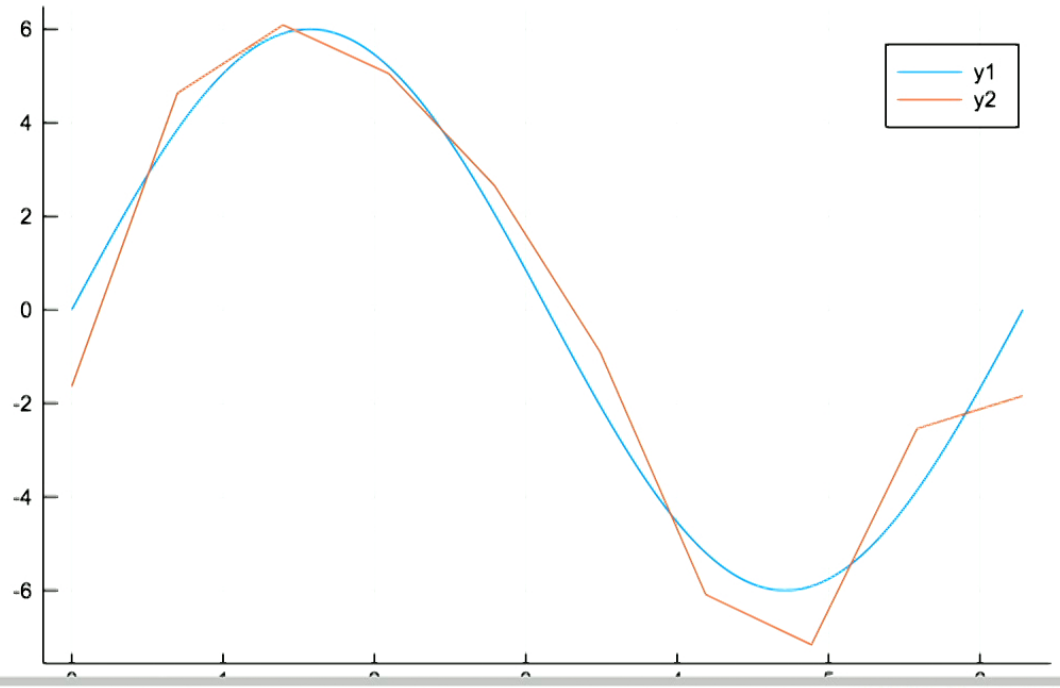
- header.tex
- mcmc.ipynb
- mcmc.tex
- p99-data.txt
- README.md

Terminal 1 | mcmc.ipynb

Markdown | Julia 1.2.0

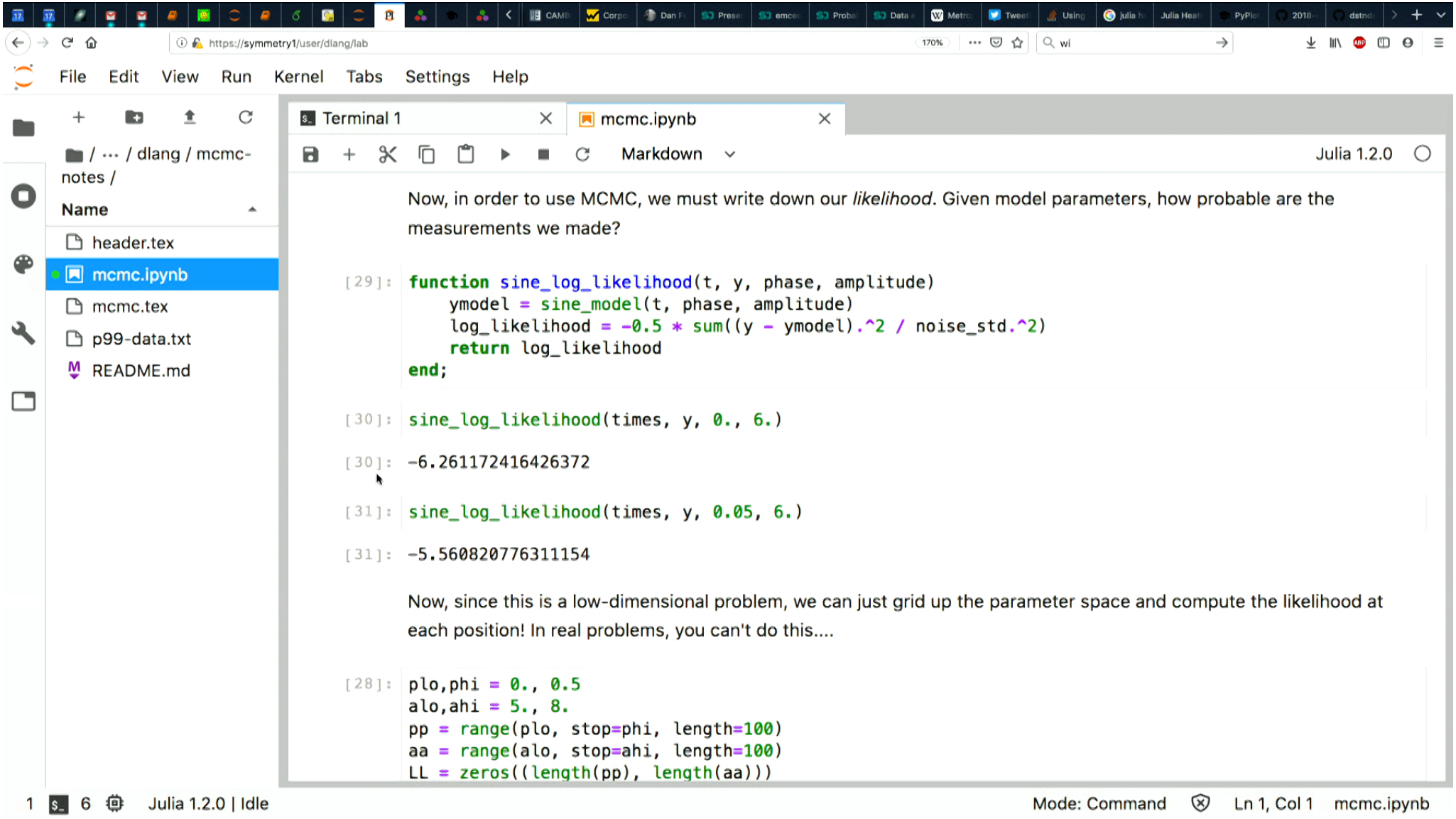
[17]:
```
tplot = collect(LinRange(0.0, 2.0*pi, 200));
# I'm going to guess at parameters: phase = 0, amplitude = 6
plot(tplot, sine_model(tplot, 0., 6))
plot!(times, y)
```

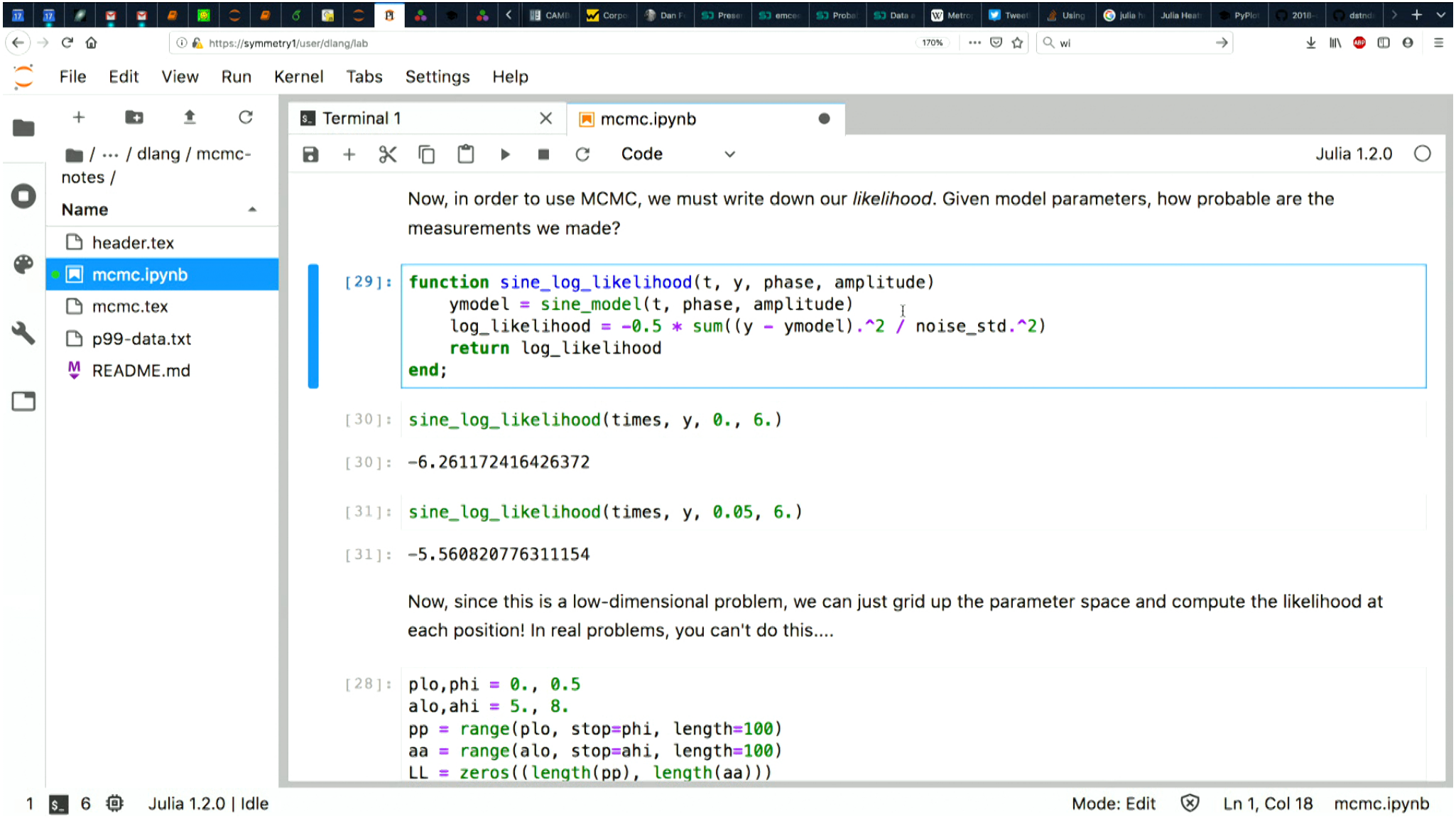
[17]:

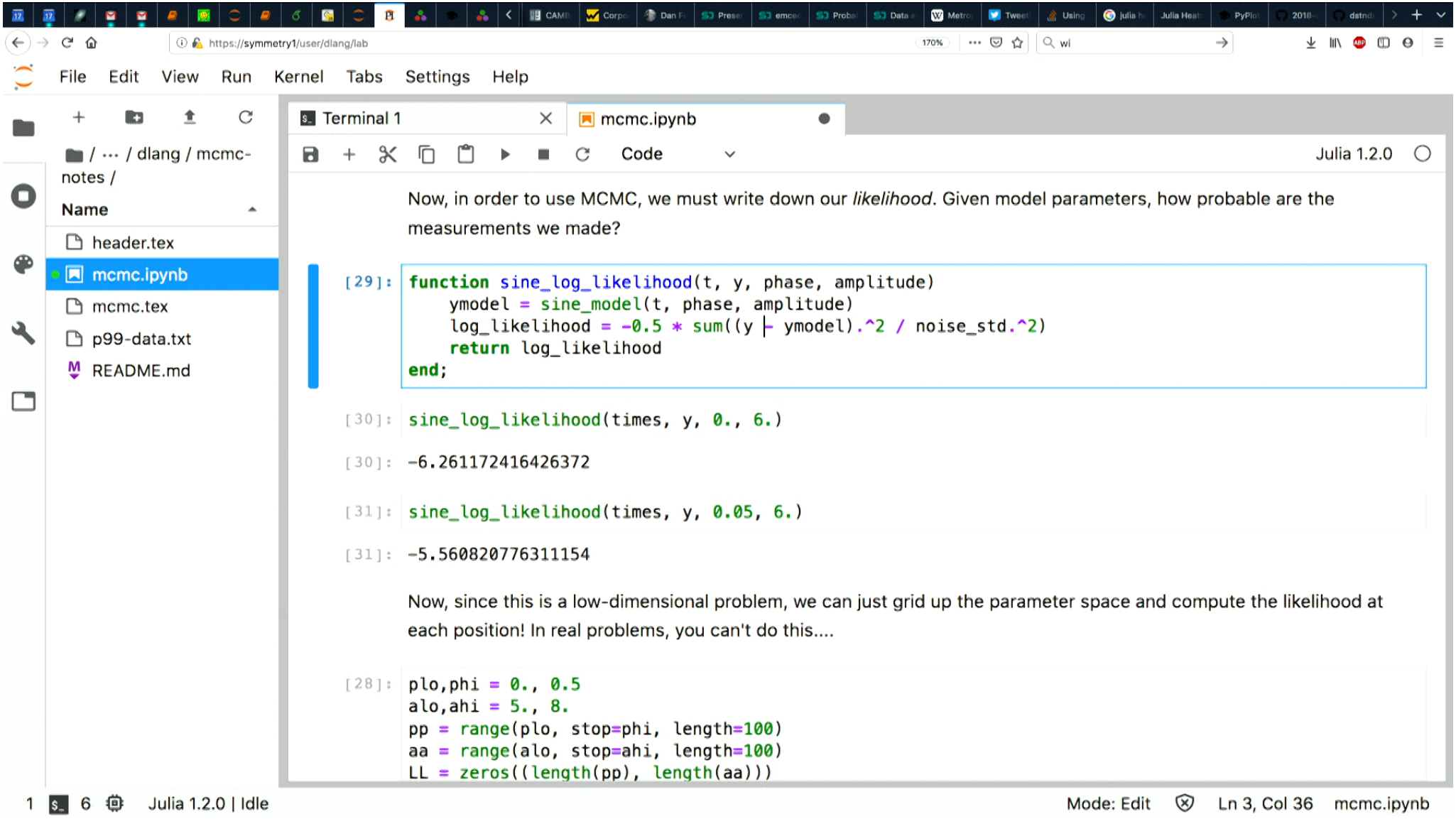


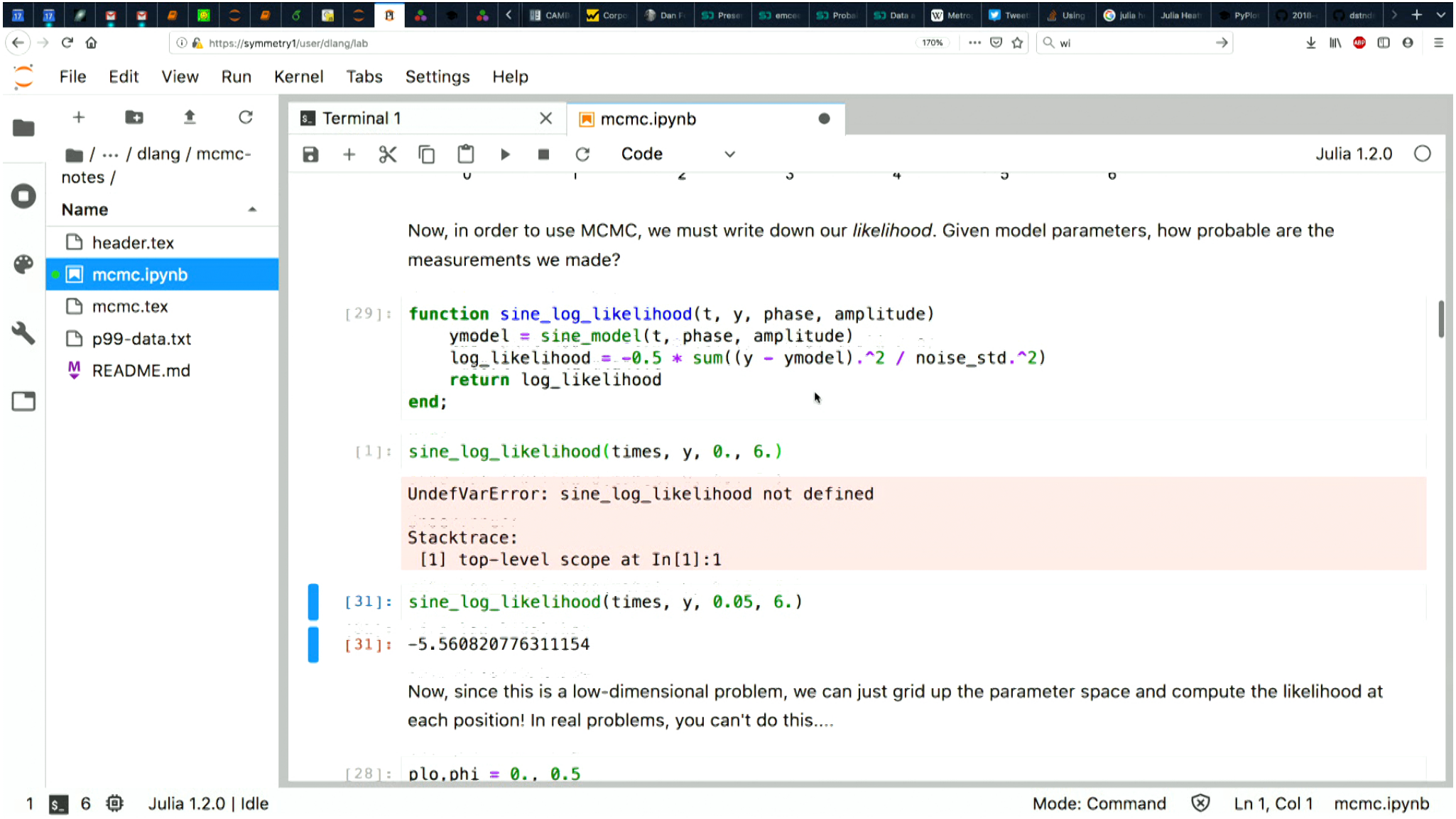
1 6 Julia 1.2.0 | Idle

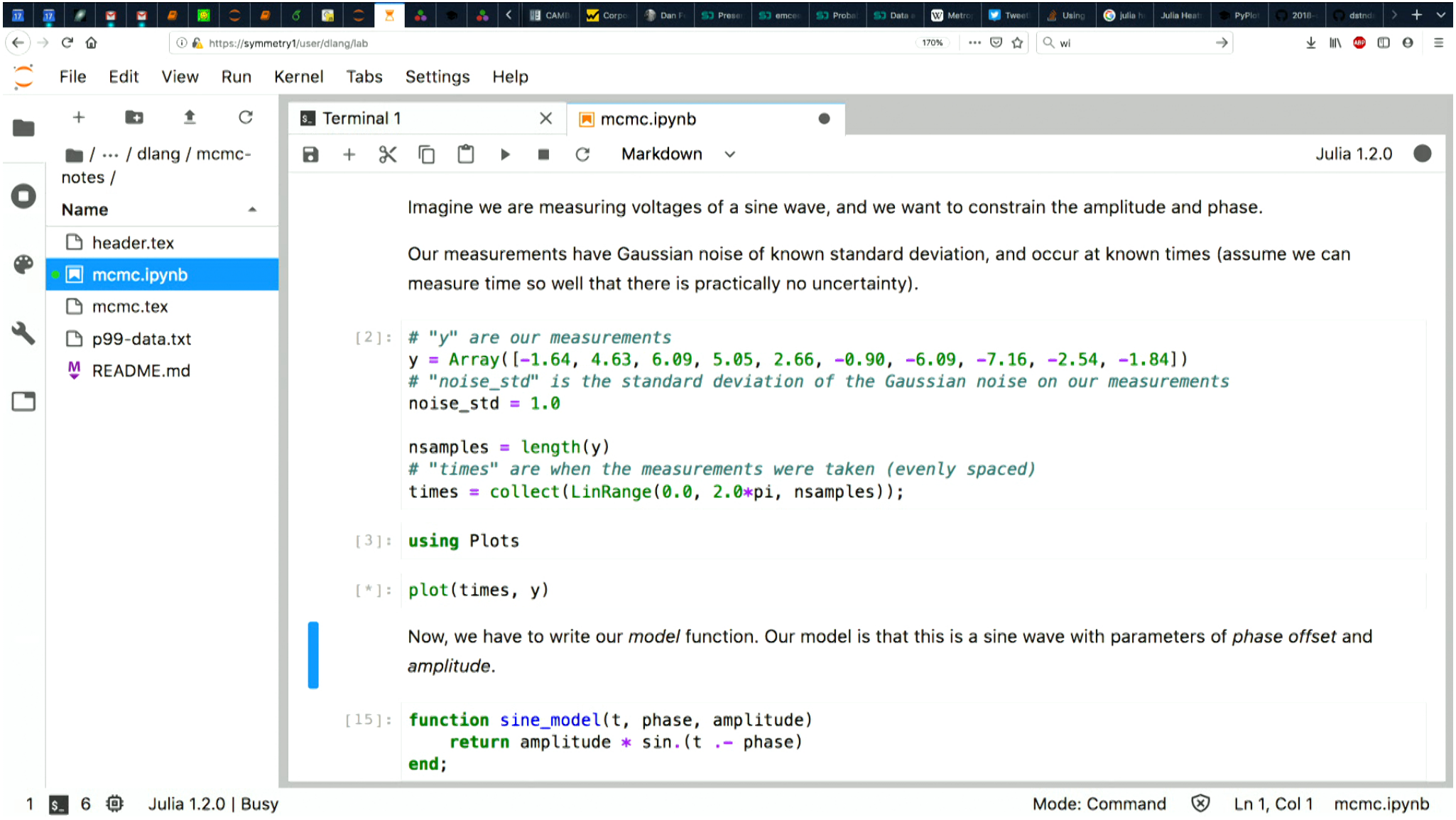
Mode: Command Ln 1, Col 1 mcmc.ipynb

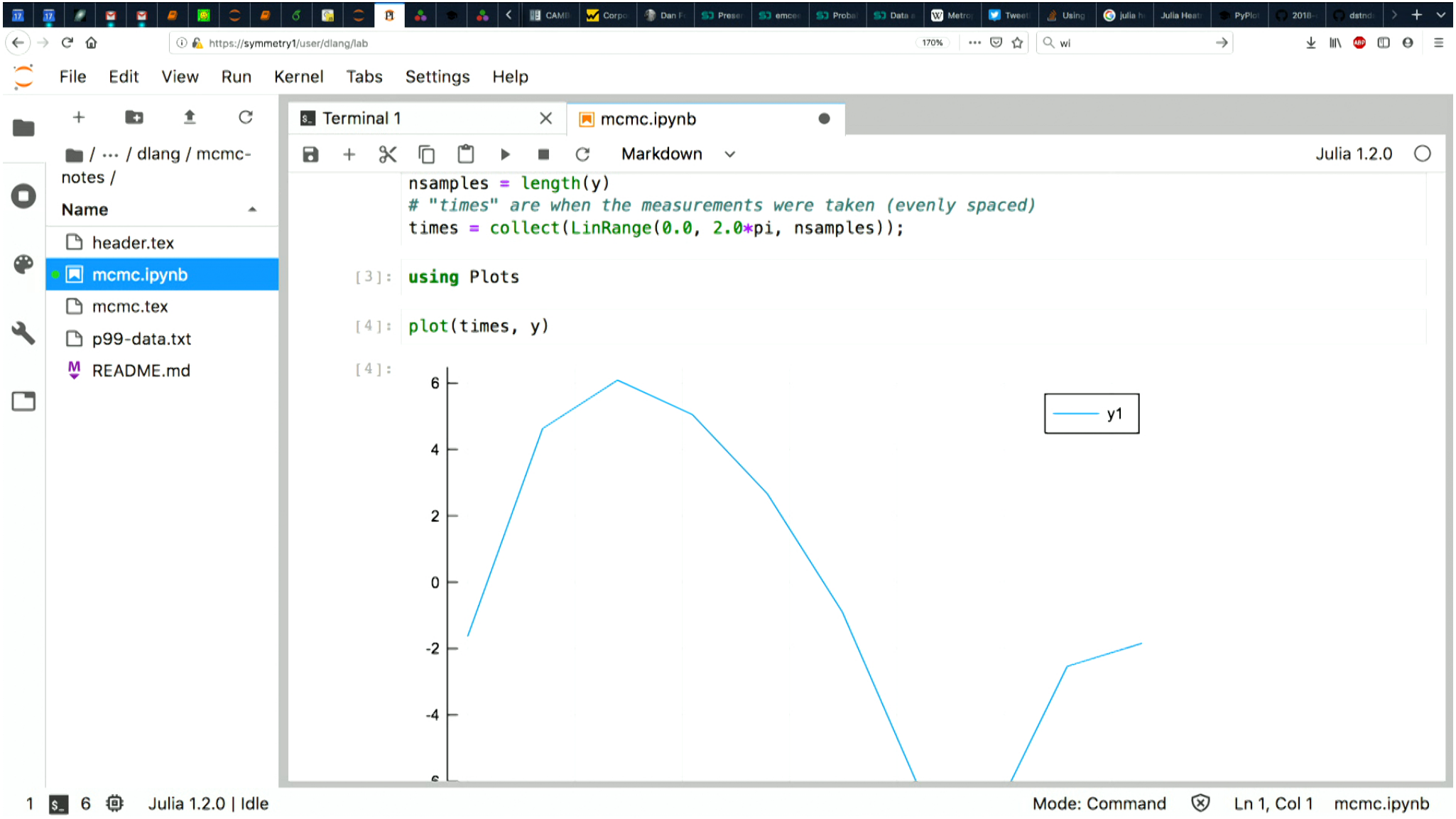

https://symmetry1/user/dlang/lab
File Edit View Run Kernel Tabs Settings Help
Terminal 1
mcmc.ipynb
/ ··· / dlang / mcmc-notes /
Name
header.tex
mcmc.ipynb
mcmc.tex
p99-data.txt
README.md
Markdown
Julia 1.2.0
Now, in order to use MCMC, we must write down our likelihood. Given model parameters, how probable are the measurements we made?
[29]:
function sine_log_likelihood(t, y, phase, amplitude)
ymodel = sine_model(t, phase, amplitude)
log_likelihood = -0.5 * sum((y - ymodel).^2 / noise_std.^2)
return log_likelihood
end;
[30]:
sine_log_likelihood(times, y, 0., 6.)
[30]:
-6.261172416426372
[31]:
sine_log_likelihood(times, y, 0.05, 6.)
[31]:
-5.560820776311154
Now, since this is a low-dimensional problem, we can just grid up the parameter space and compute the likelihood at each position! In real problems, you can't do this....
[28]:
plo,phi = 0., 0.5
alo,ahi = 5., 8.
pp = range(plo, stop=phi, length=100)
aa = range(alo, stop=ahi, length=100)
LL = zeros((length(pp), length(aa)))
1 6 Julia 1.2.0 | Idle
Mode: Command
Ln 1, Col 1
mcmc.ipynb

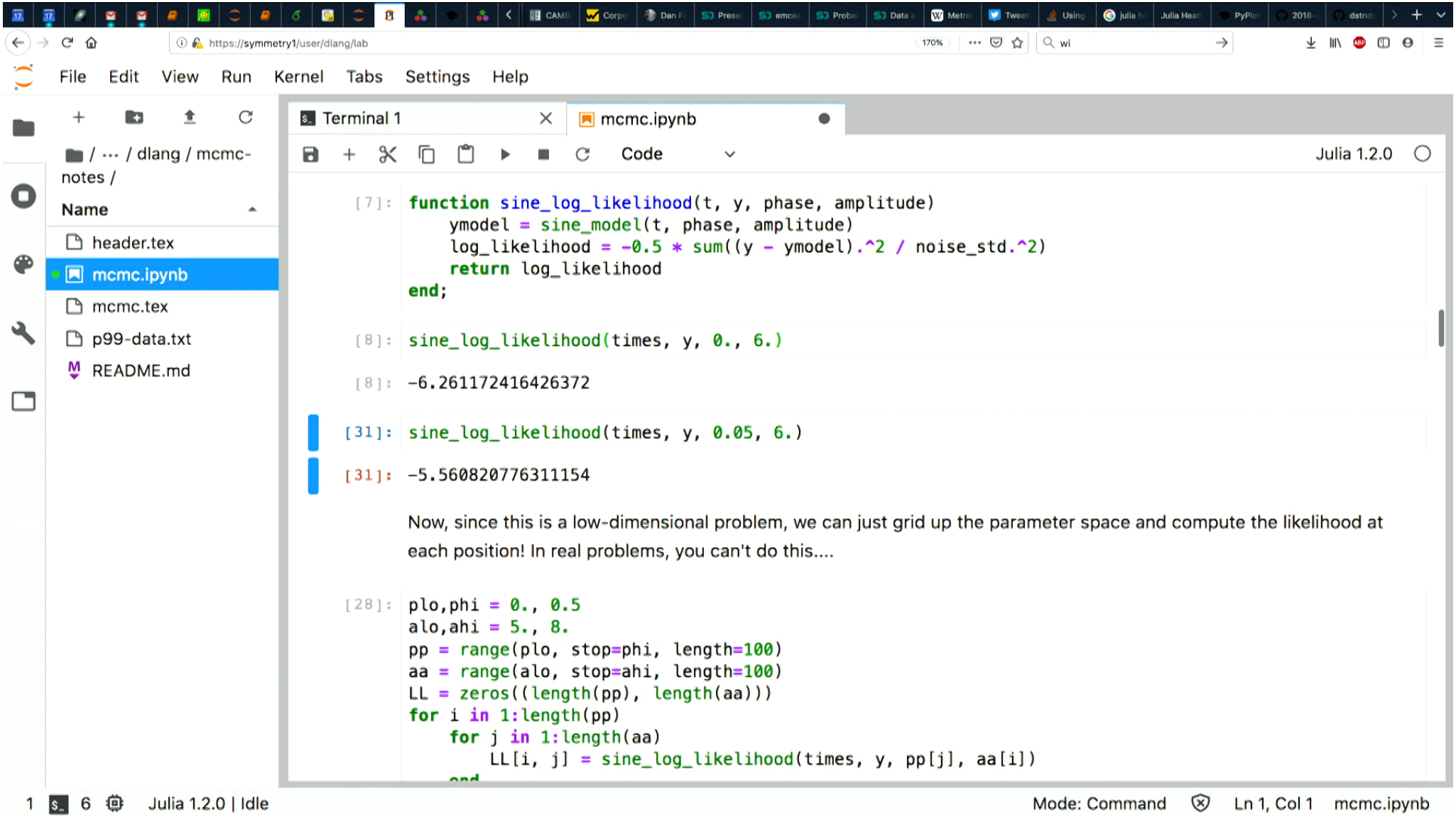

https://symmetry1/user/dlang/lab
File Edit View Run Kernel Tabs Settings Help
Terminal 1
mcmc.ipynb
/ ··· / dlang / mcmc-notes /
Name
header.tex
mcmc.ipynb
mcmc.tex
p99-data.txt
README.md
Code
Julia 1.2.0
Now, in order to use MCMC, we must write down our likelihood. Given model parameters, how probable are the measurements we made?
[29]:
function sine_log_likelihood(t, y, phase, amplitude)
ymodel = sine_model(t, phase, amplitude)
log_likelihood = -0.5 * sum((y - ymodel).^2 / noise_std.^2)
return log_likelihood
end;
[30]:
sine_log_likelihood(times, y, 0., 6.)
[30]:
-6.261172416426372
[31]:
sine_log_likelihood(times, y, 0.05, 6.)
[31]:
-5.560820776311154
Now, since this is a low-dimensional problem, we can just grid up the parameter space and compute the likelihood at each position! In real problems, you can't do this....
[28]:
plo,phi = 0., 0.5
alo,ahi = 5., 8.
pp = range(plo, stop=phi, length=100)
aa = range(alo, stop=ahi, length=100)
LL = zeros((length(pp), length(aa)))
1 6 Julia 1.2.0 | Idle
Mode: Edit
Ln 1, Col 18
mcmc.ipynb


https://symmetry1/user/dlang/lab
File Edit View Run Kernel Tabs Settings Help
Terminal 1
mcmc.ipynb
/ ··· / dlang / mcmc-notes /
Name
header.tex
mcmc.ipynb
mcmc.tex
p99-data.txt
README.md
Code
Julia 1.2.0
Now, in order to use MCMC, we must write down our *likelihood*. Given model parameters, how probable are the measurements we made?
[29]:
function sine_log_likelihood(t, y, phase, amplitude)
ymodel = sine_model(t, phase, amplitude)
log_likelihood = -0.5 * sum((y - ymodel).^2 / noise_std.^2)
return log_likelihood
end;
[30]:
sine_log_likelihood(times, y, 0., 6.)
[30]:
-6.261172416426372
[31]:
sine_log_likelihood(times, y, 0.05, 6.)
[31]:
-5.560820776311154
Now, since this is a low-dimensional problem, we can just grid up the parameter space and compute the likelihood at each position! In real problems, you can't do this....
[28]:
plo,phi = 0., 0.5
alo,ahi = 5., 8.
pp = range(plo, stop=phi, length=100)
aa = range(alo, stop=ahi, length=100)
LL = zeros((length(pp), length(aa)))
1 6 Julia 1.2.0 | Idle
Mode: Edit
Ln 3, Col 36
mcmc.ipynb


https://symmetry1/user/dlang/lab
File Edit View Run Kernel Tabs Settings Help
Terminal 1
mcmc.ipynb
/ ··· / dlang / mcmc-notes /
Name
header.tex
mcmc.ipynb
mcmc.tex
p99-data.txt
README.md
Code
Julia 1.2.0
Now, in order to use MCMC, we must write down our *likelihood*. Given model parameters, how probable are the measurements we made?
[29]:
function sine_log_likelihood(t, y, phase, amplitude)
ymodel = sine_model(t, phase, amplitude)
log_likelihood = -0.5 * sum((y - ymodel).^2 / noise_std.^2)
return log_likelihood
end;
[1]:
sine_log_likelihood(times, y, 0., 6.)
UndefVarError: sine_log_likelihood not defined
Stacktrace:
[1] top-level scope at In[1]:1
[31]:
sine_log_likelihood(times, y, 0.05, 6.)
[31]:
-5.560820776311154
Now, since this is a low-dimensional problem, we can just grid up the parameter space and compute the likelihood at each position! In real problems, you can't do this....
[28]:
plo.phi = 0., 0.5
1 6 Julia 1.2.0 | Idle
Mode: Command
Ln 1, Col 1
mcmc.ipynb


https://symmetry1/user/dlang/lab
File Edit View Run Kernel Tabs Settings Help
Terminal 1
mcmc.ipynb
/ ··· / dlang / mcmc-notes /
Name
header.tex
mcmc.ipynb
mcmc.tex
p99-data.txt
README.md
Markdown
Julia 1.2.0
Imagine we are measuring voltages of a sine wave, and we want to constrain the amplitude and phase.
Our measurements have Gaussian noise of known standard deviation, and occur at known times (assume we can measure time so well that there is practically no uncertainty).
[2]:
# "y" are our measurements
y = Array([-1.64, 4.63, 6.09, 5.05, 2.66, -0.90, -6.09, -7.16, -2.54, -1.84])
# "noise_std" is the standard deviation of the Gaussian noise on our measurements
noise_std = 1.0
nsamples = length(y)
# "times" are when the measurements were taken (evenly spaced)
times = collect(LinRange(0.0, 2.0*pi, nsamples));
[3]:
using Plots
[*]:
plot(times, y)
Now, we have to write our model function. Our model is that this is a sine wave with parameters of phase offset and amplitude.
[15]:
function sine_model(t, phase, amplitude)
return amplitude * sin.(t .- phase)
end;
1 6 Julia 1.2.0 | Busy
Mode: Command
Ln 1, Col 1
mcmc.ipynb


https://symmetry1/user/dlang/lab
File Edit View Run Kernel Tabs Settings Help
Terminal 1
mcmc.ipynb
/ ··· / dlang / mcmc-notes /
Name
header.tex
mcmc.ipynb
mcmc.tex
p99-data.txt
README.md
Markdown
Julia 1.2.0
nsamples = length(y)
# "times" are when the measurements were taken (evenly spaced)
times = collect(LinRange(0.0, 2.0*pi, nsamples));
[3]:
using Plots
[4]:
plot(times, y)
[4]:
y1
6
4
2
0
-2
-4
Julia 1.2.0 | Idle
Mode: Command
Ln 1, Col 1
mcmc.ipynb


https://symmetry1/user/dlang/lab
File Edit View Run Kernel Tabs Settings Help
/ ··· / dlang / mcmc-notes /
Name
header.tex
mcmc.ipynb
mcmc.tex
p99-data.txt
README.md
Terminal 1
mcmc.ipynb
Code
Julia 1.2.0
[7]:
function sine_log_likelihood(t, y, phase, amplitude)
ymodel = sine_model(t, phase, amplitude)
log_likelihood = -0.5 * sum((y - ymodel).^2 / noise_std.^2)
return log_likelihood
end;
[8]:
sine_log_likelihood(times, y, 0., 6.)
[8]:
-6.261172416426372
[31]:
sine_log_likelihood(times, y, 0.05, 6.)
[31]:
-5.560820776311154
Now, since this is a low-dimensional problem, we can just grid up the parameter space and compute the likelihood at each position! In real problems, you can't do this....
[28]:
plo,phi = 0., 0.5
alo,ahi = 5., 8.
pp = range(plo, stop=phi, length=100)
aa = range(alo, stop=ahi, length=100)
LL = zeros((length(pp), length(aa)))
for i in 1:length(pp)
for j in 1:length(aa)
LL[i, j] = sine_log_likelihood(times, y, pp[j], aa[i])
1 6 Julia 1.2.0 | Idle
Mode: Command
Ln 1, Col 1
mcmc.ipynb

https://symmetry1/user/dlang/lab

File Edit View Run Kernel Tabs Settings Help

/ ··· / dlang / mcmc-notes /

Name

- header.tex
- mcmc.ipynb
- mcmc.tex
- p99-data.txt
- README.md

Terminal 1 | mcmc.ipynb

Code | Julia 1.2.0

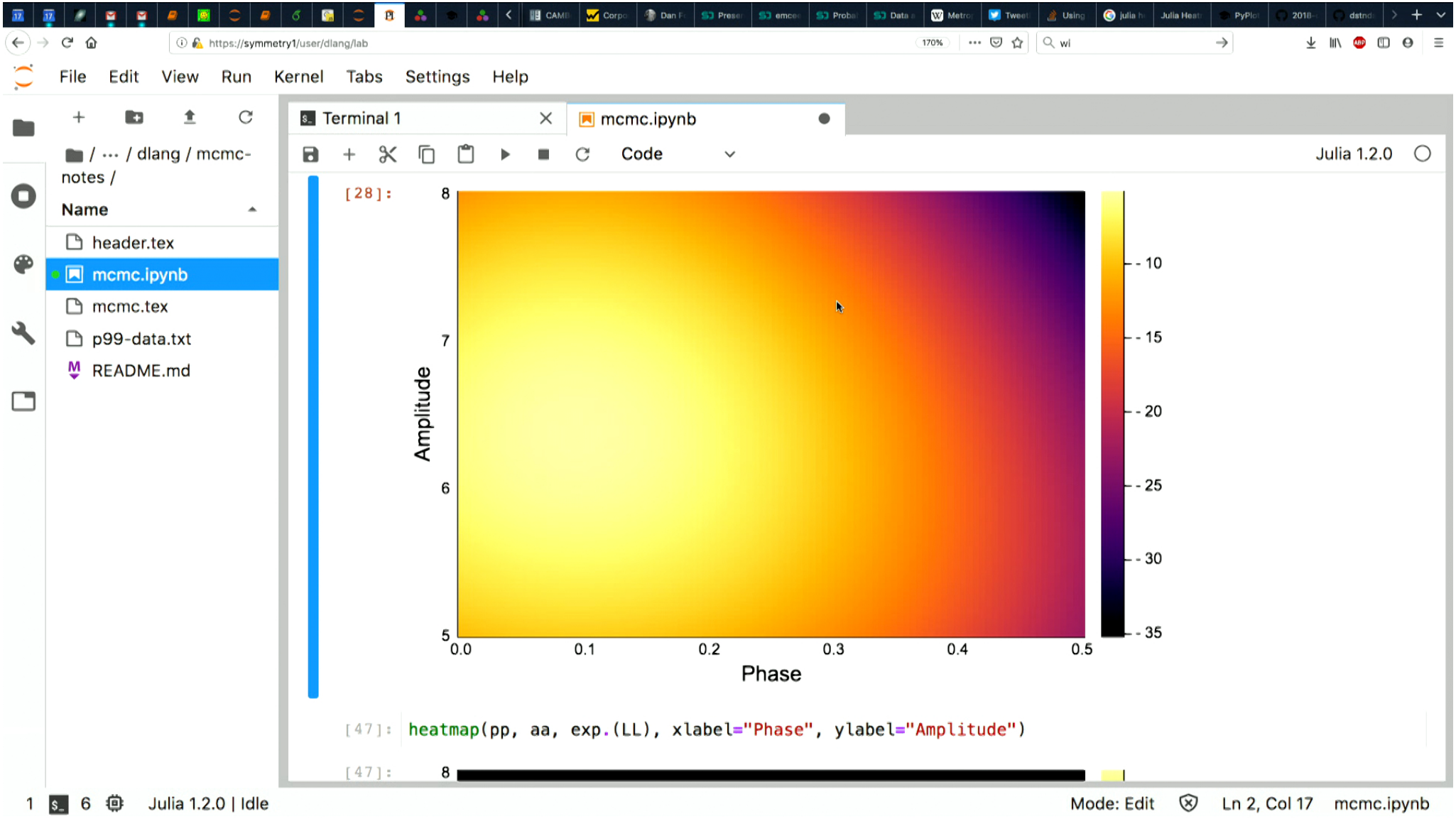
Now, since this is a low-dimensional problem, we can just grid up the parameter space and compute the likelihood at each position! In real problems, you can't do this....

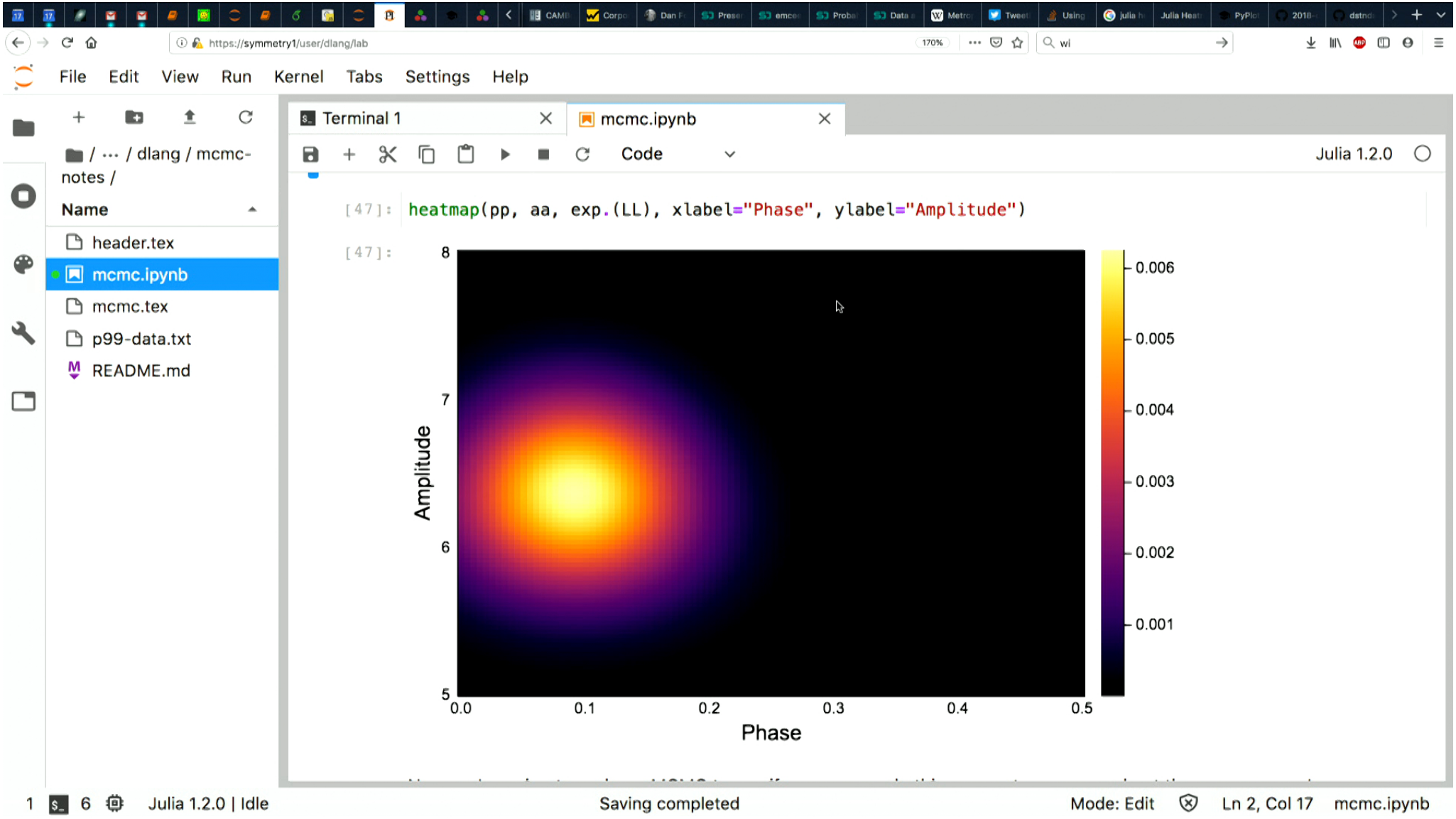
[28]:

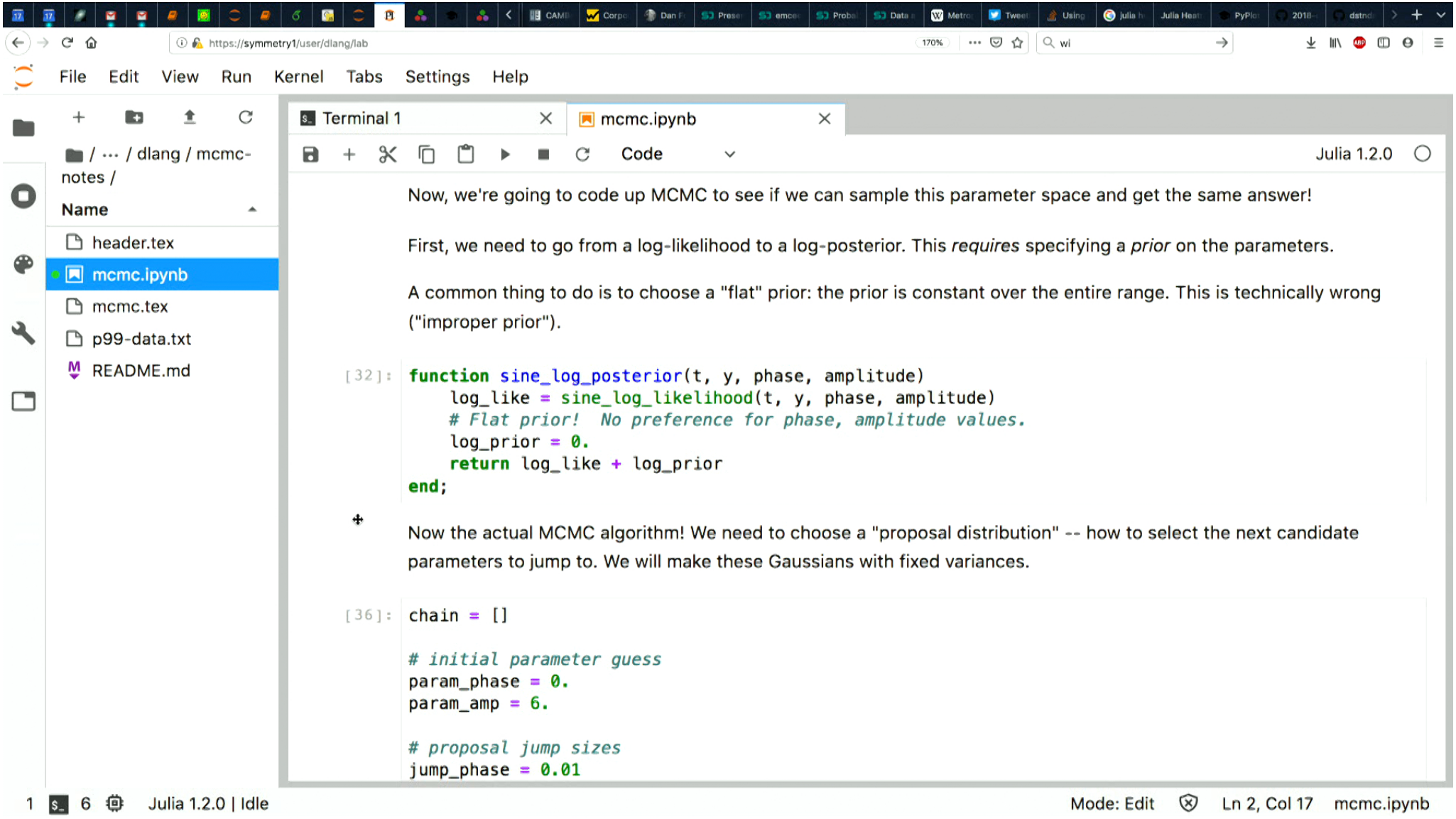
```
plo,phi = 0., 0.5
alo,ahi = 5., 8.
pp = range(plo, stop=phi, length=100)
aa = range(alo, stop=ahi, length=100)
LL = zeros((length(pp), length(aa)))
for i in 1:length(pp)
    for j in 1:length(aa)
        LL[i, j] = sine_log_likelihood(times, y, pp[j], aa[i])
    end
end
heatmap(pp, aa, LL, xlabel="Phase", ylabel="Amplitude")
```

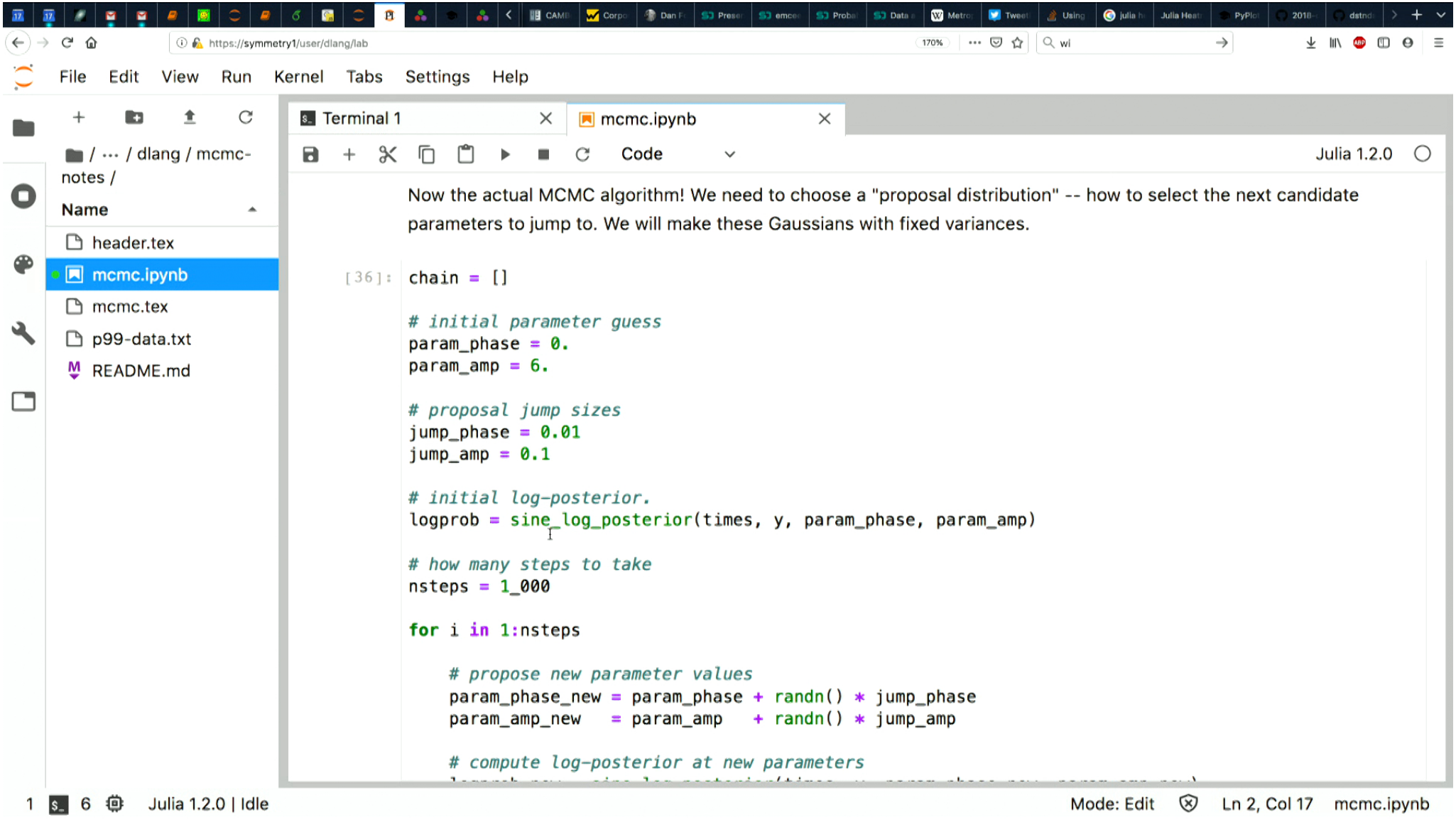
[28]:

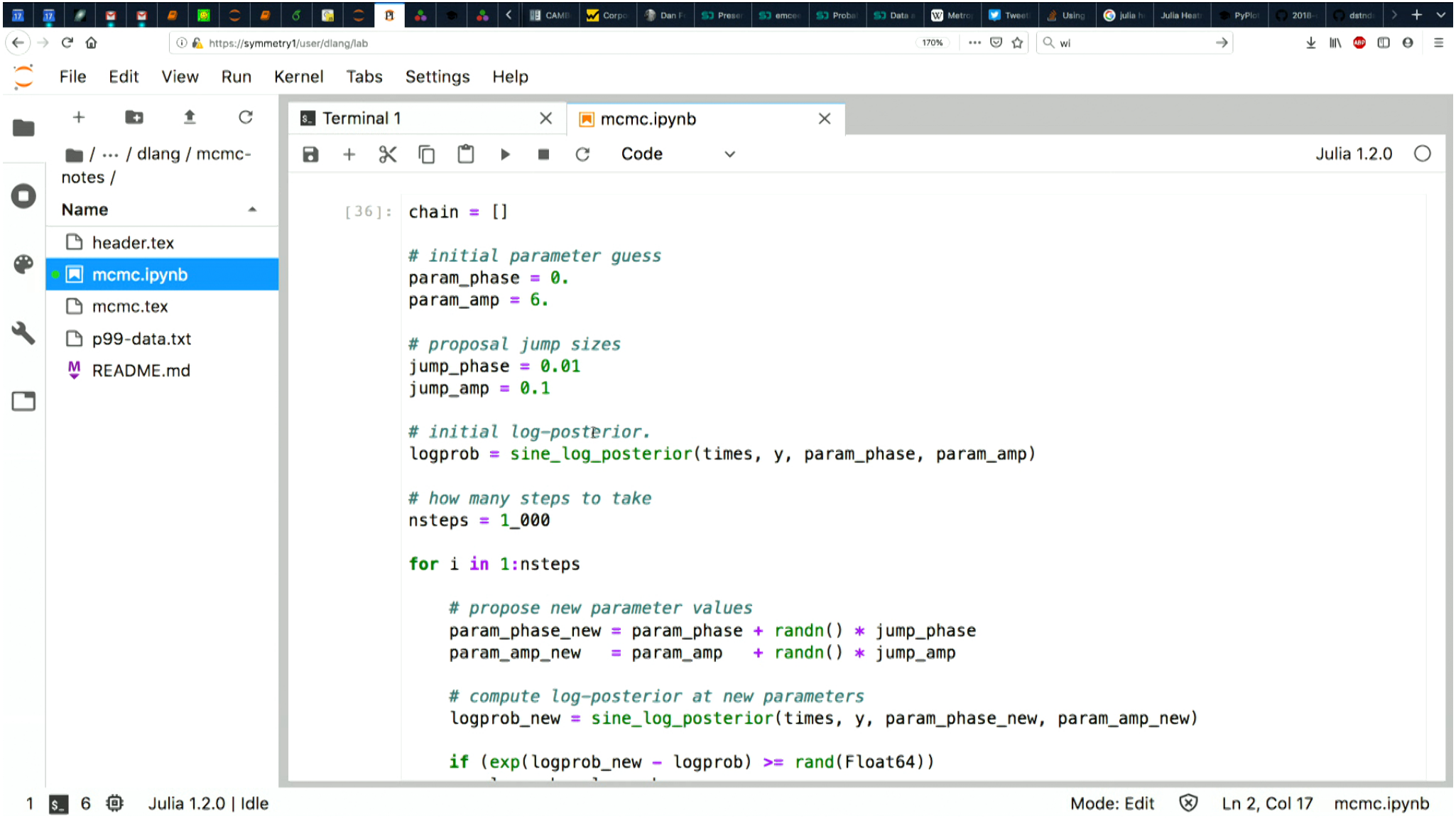
1 6 Julia 1.2.0 | Idle Mode: Edit Ln 2, Col 17 mcmc.ipynb

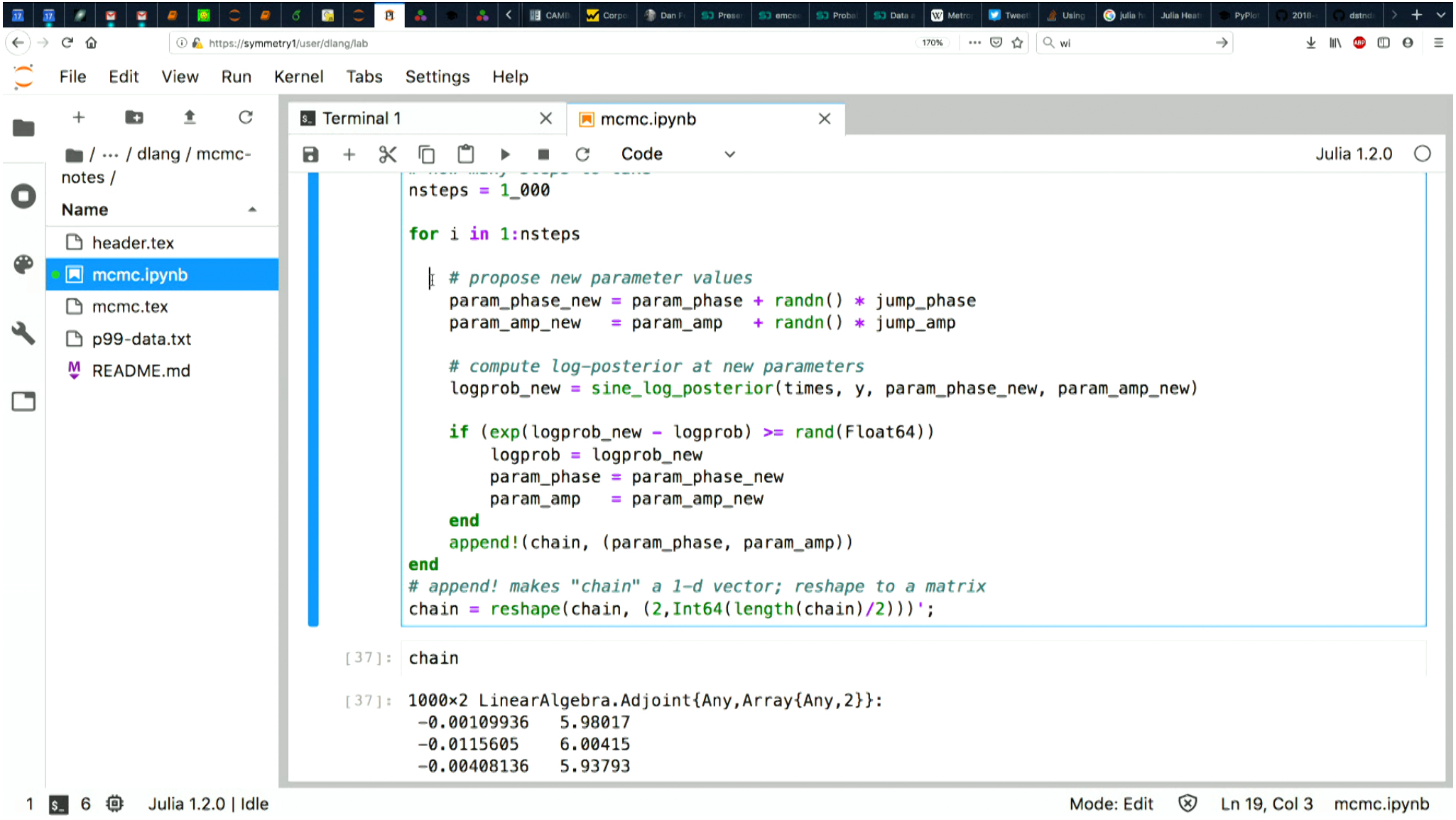

https://symmetry1/user/dlang/lab
File Edit View Run Kernel Tabs Settings Help
Terminal 1
mcmc.ipynb
/ ··· / dlang / mcmc-notes /
Name
header.tex
mcmc.ipynb
mcmc.tex
p99-data.txt
README.md
Code
Julia 1.2.0
[28]:
Amplitude
Phase
[47]: heatmap(pp, aa, exp.(LL), xlabel="Phase", ylabel="Amplitude")
[47]:
Julia 1.2.0 | Idle
Mode: Edit
Ln 2, Col 17
mcmc.ipynb

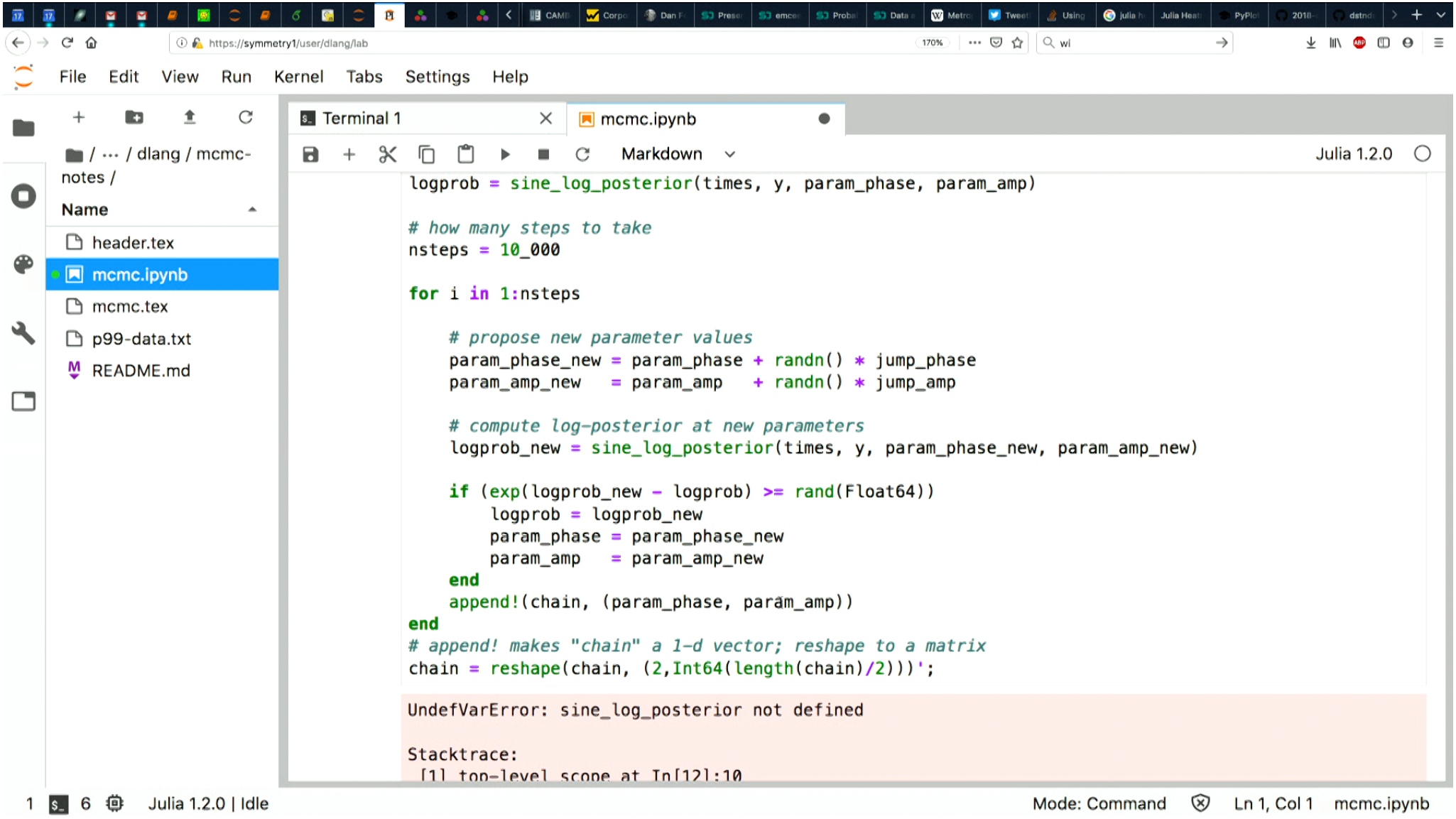

https://symmetry1/user/dlang/lab
File Edit View Run Kernel Tabs Settings Help
Terminal 1
mcmc.ipynb
/ ··· / dlang / mcmc-notes /
Name
header.tex
mcmc.ipynb
mcmc.tex
p99-data.txt
README.md
Code
Julia 1.2.0
[47]: heatmap(pp, aa, exp.(LL), xlabel="Phase", ylabel="Amplitude")
[47]:
Amplitude
Phase
8
7
6
5
0.0
0.1
0.2
0.3
0.4
0.5
0.006
0.005
0.004
0.003
0.002
0.001
1 6 Julia 1.2.0 | Idle
Saving completed
Mode: Edit
Ln 2, Col 17
mcmc.ipynb

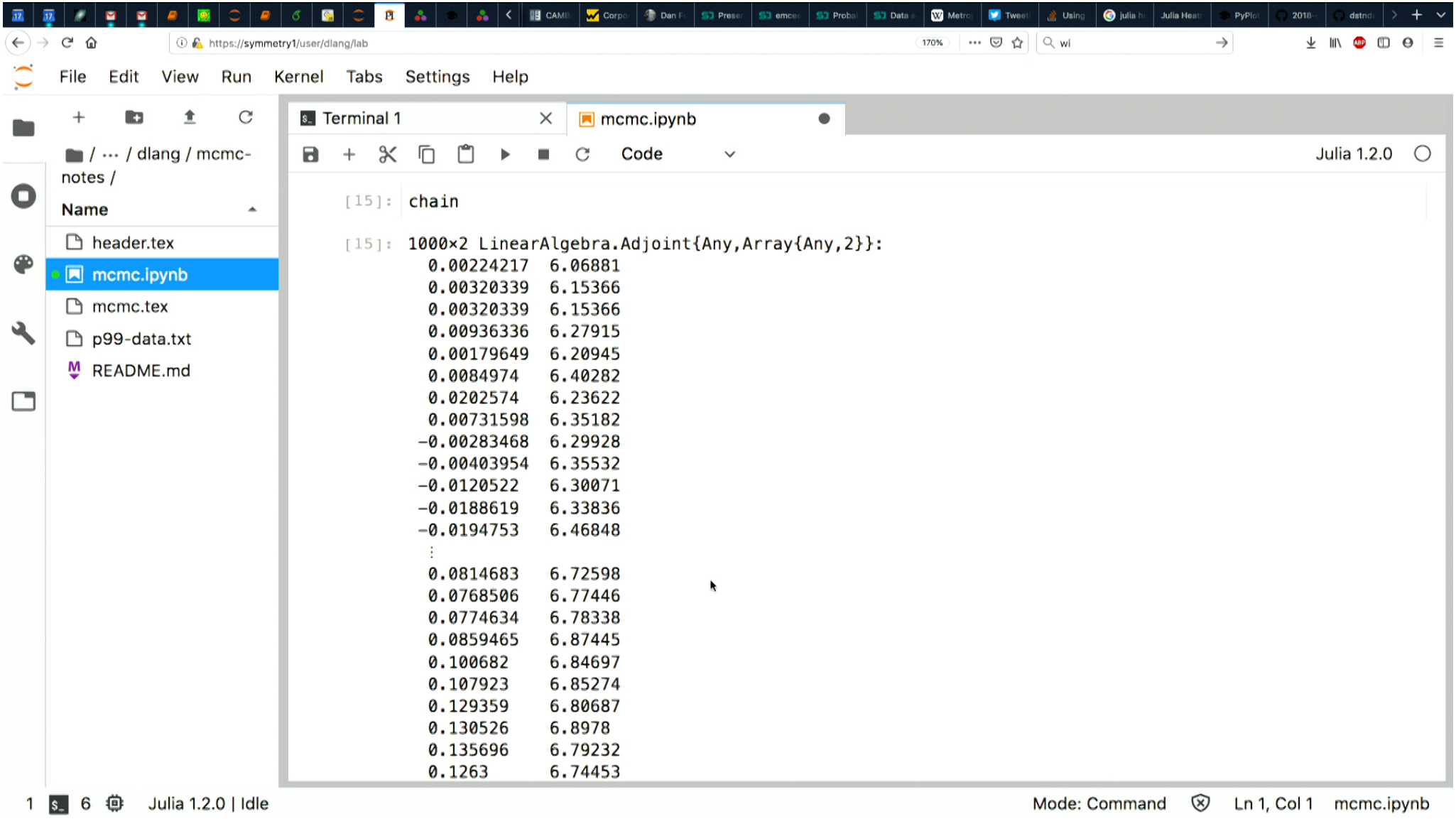

https://symmetry1/user/dlang/lab
File Edit View Run Kernel Tabs Settings Help
Terminal 1
mcmc.ipynb
/ ··· / dlang / mcmc-notes /
Name
header.tex
mcmc.ipynb
mcmc.tex
p99-data.txt
README.md
Code
Julia 1.2.0
Now, we're going to code up MCMC to see if we can sample this parameter space and get the same answer!
First, we need to go from a log-likelihood to a log-posterior. This requires specifying a prior on the parameters.
A common thing to do is to choose a "flat" prior: the prior is constant over the entire range. This is technically wrong ("improper prior").
[32]:
function sine_log_posterior(t, y, phase, amplitude)
log_like = sine_log_likelihood(t, y, phase, amplitude)
# Flat prior! No preference for phase, amplitude values.
log_prior = 0.
return log_like + log_prior
end;
Now the actual MCMC algorithm! We need to choose a "proposal distribution" -- how to select the next candidate parameters to jump to. We will make these Gaussians with fixed variances.
[36]:
chain = []
# initial parameter guess
param_phase = 0.
param_amp = 6.
# proposal jump sizes
jump_phase = 0.01
1 6 Julia 1.2.0 | Idle
Mode: Edit
Ln 2, Col 17
mcmc.ipynb


https://symmetry1/user/dlang/lab
File Edit View Run Kernel Tabs Settings Help
Terminal 1
mcmc.ipynb
/ ··· / dlang / mcmc-notes /
Name
header.tex
mcmc.ipynb
mcmc.tex
p99-data.txt
README.md
Code
Julia 1.2.0
Now the actual MCMC algorithm! We need to choose a "proposal distribution" -- how to select the next candidate parameters to jump to. We will make these Gaussians with fixed variances.
[36]:
chain = []
# initial parameter guess
param_phase = 0.
param_amp = 6.
# proposal jump sizes
jump_phase = 0.01
jump_amp = 0.1
# initial log-posterior.
logprob = sine_log_posterior(times, y, param_phase, param_amp)
# how many steps to take
nsteps = 1_000
for i in 1:nsteps
# propose new parameter values
param_phase_new = param_phase + randn() * jump_phase
param_amp_new = param_amp + randn() * jump_amp
# compute log-posterior at new parameters
1 6 Julia 1.2.0 | Idle
Mode: Edit
Ln 2, Col 17
mcmc.ipynb


https://symmetry1/user/dlang/lab
File Edit View Run Kernel Tabs Settings Help
Terminal 1
mcmc.ipynb
/ ··· / dlang / mcmc-notes /
Name
header.tex
mcmc.ipynb
mcmc.tex
p99-data.txt
README.md
Code
Julia 1.2.0
[36]:
chain = []
# initial parameter guess
param_phase = 0.
param_amp = 6.
# proposal jump sizes
jump_phase = 0.01
jump_amp = 0.1
# initial log-posterior.
logprob = sine_log_posterior(times, y, param_phase, param_amp)
# how many steps to take
nsteps = 1_000
for i in 1:nsteps
# propose new parameter values
param_phase_new = param_phase + randn() * jump_phase
param_amp_new = param_amp + randn() * jump_amp
# compute log-posterior at new parameters
logprob_new = sine_log_posterior(times, y, param_phase_new, param_amp_new)
if (exp(logprob_new - logprob) >= rand(Float64))
1 6 Julia 1.2.0 | Idle
Mode: Edit
Ln 2, Col 17
mcmc.ipynb


https://symmetry1/user/dlang/lab
File Edit View Run Kernel Tabs Settings Help
Terminal 1
mcmc.ipynb
/ ··· / dlang / mcmc-notes /
Name
header.tex
mcmc.ipynb
mcmc.tex
p99-data.txt
README.md
Code
Julia 1.2.0
nsteps = 1_000
for i in 1:nsteps
# propose new parameter values
param_phase_new = param_phase + randn() * jump_phase
param_amp_new = param_amp + randn() * jump_amp
# compute log-posterior at new parameters
logprob_new = sine_log_posterior(times, y, param_phase_new, param_amp_new)
if (exp(logprob_new - logprob) >= rand(Float64))
logprob = logprob_new
param_phase = param_phase_new
param_amp = param_amp_new
end
append!(chain, (param_phase, param_amp))
end
# append! makes "chain" a 1-d vector; reshape to a matrix
chain = reshape(chain, (2,Int64(length(chain)/2)))';
[37]: chain
[37]: 1000×2 LinearAlgebra.Adjoint{Any,Array{Any,2}}:
-0.00109936 5.98017
-0.0115605 6.00415
-0.00408136 5.93793
1 6 Julia 1.2.0 | Idle
Mode: Edit
Ln 19, Col 3
mcmc.ipynb


https://symmetry1/user/dlang/lab
File Edit View Run Kernel Tabs Settings Help
Terminal 1
mcmc.ipynb
Markdown
Julia 1.2.0
/ ··· / dlang / mcmc-notes /
Name
header.tex
mcmc.ipynb
mcmc.tex
p99-data.txt
README.md
logprob = sine_log_posterior(times, y, param_phase, param_amp)
# how many steps to take
nsteps = 10_000
for i in 1:nsteps
# propose new parameter values
param_phase_new = param_phase + randn() * jump_phase
param_amp_new = param_amp + randn() * jump_amp
# compute log-posterior at new parameters
logprob_new = sine_log_posterior(times, y, param_phase_new, param_amp_new)
if (exp(logprob_new - logprob) >= rand(Float64))
logprob = logprob_new
param_phase = param_phase_new
param_amp = param_amp_new
end
append!(chain, (param_phase, param_amp))
end
# append! makes "chain" a 1-d vector; reshape to a matrix
chain = reshape(chain, (2,Int64(length(chain)/2)))';
UndefVarError: sine_log_posterior not defined
Stacktrace:
[1] top-level scope at In[12]:10
1 6 Julia 1.2.0 | Idle
Mode: Command
Ln 1, Col 1
mcmc.ipynb


https://symmetry1/user/dlang/lab
File Edit View Run Kernel Tabs Settings Help
Terminal 1
mcmc.ipynb
/ ··· / dlang / mcmc-notes /
Name
header.tex
mcmc.ipynb
mcmc.tex
p99-data.txt
README.md
Code
Julia 1.2.0
[15]: chain
[15]: 1000×2 LinearAlgebra.Adjoint{Any,Array{Any,2}}:
0.00224217 6.06881
0.00320339 6.15366
0.00320339 6.15366
0.00936336 6.27915
0.00179649 6.20945
0.0084974 6.40282
0.0202574 6.23622
0.00731598 6.35182
-0.00283468 6.29928
-0.00403954 6.35532
-0.0120522 6.30071
-0.0188619 6.33836
-0.0194753 6.46848
0.0814683 6.72598
0.0768506 6.77446
0.0774634 6.78338
0.0859465 6.87445
0.100682 6.84697
0.107923 6.85274
0.129359 6.80687
0.130526 6.8978
0.135696 6.79232
0.1263 6.74453
1 6 Julia 1.2.0 | Idle
Mode: Command
Ln 1, Col 1
mcmc.ipynb

https://symmetry1/user/dlang/lab

File Edit View Run Kernel Tabs Settings Help

/ ··· / dlang / mcmc-notes /

Name

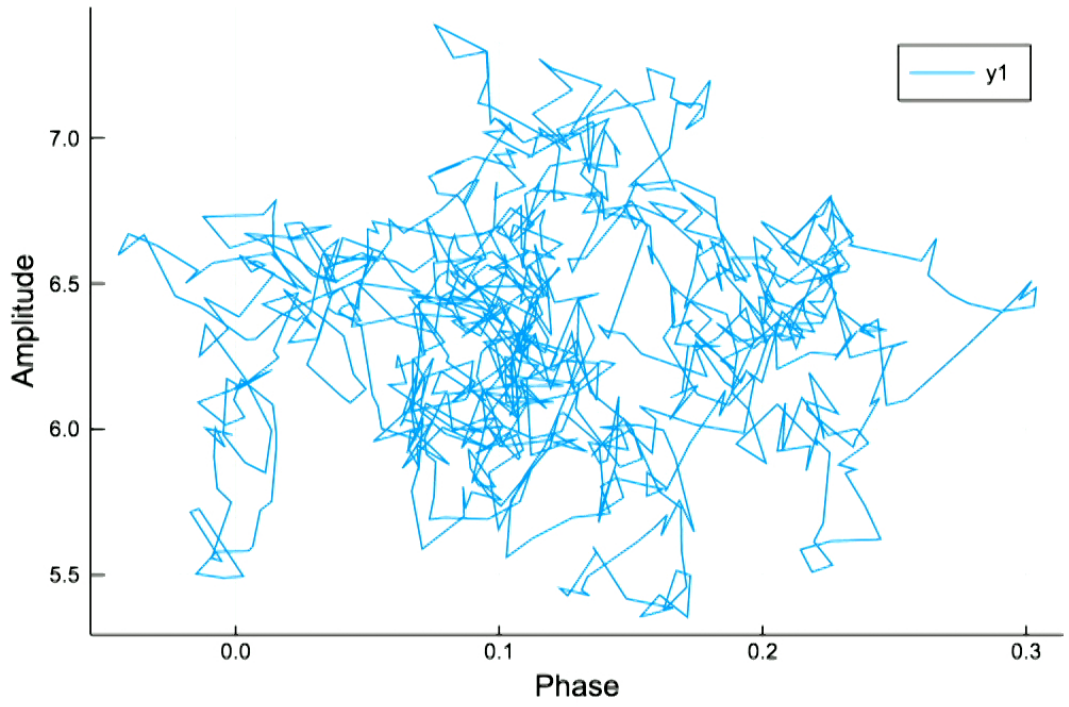
- header.tex
- mcmc.ipynb
- mcmc.tex
- p99-data.txt
- README.md

Terminal 1 | mcmc.ipynb

Code | Julia 1.2.0

```
[43]: plot(chain[:,1], chain[:,2], xlabel="Phase", ylabel="Amplitude")
```

[43]:



1 6 Julia 1.2.0 | Idle

Mode: Edit Ln 1, Col 65 mcmc.ipynb

https://symmetry1/user/dlang/lab

File Edit View Run Kernel Tabs Settings Help

Terminal 1 | mcmc.ipynb

/ ··· / dlang / mcmc-notes /

Name

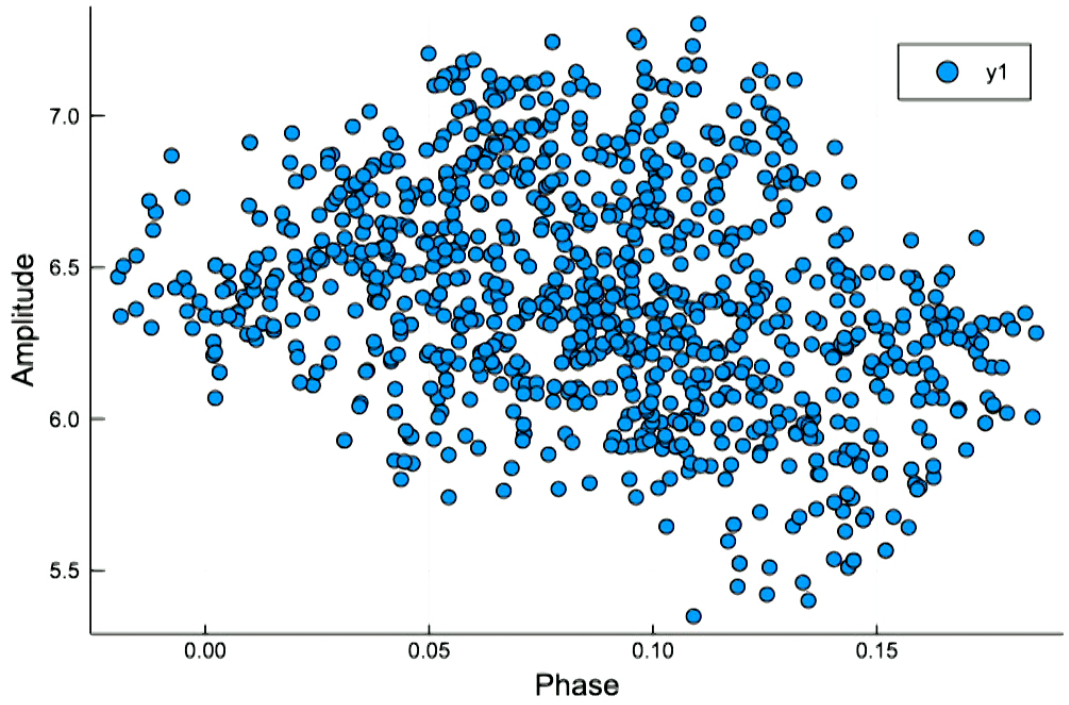
- header.tex
- mcmc.ipynb
- mcmc.tex
- p99-data.txt
- README.md

Code | Julia 1.2.0

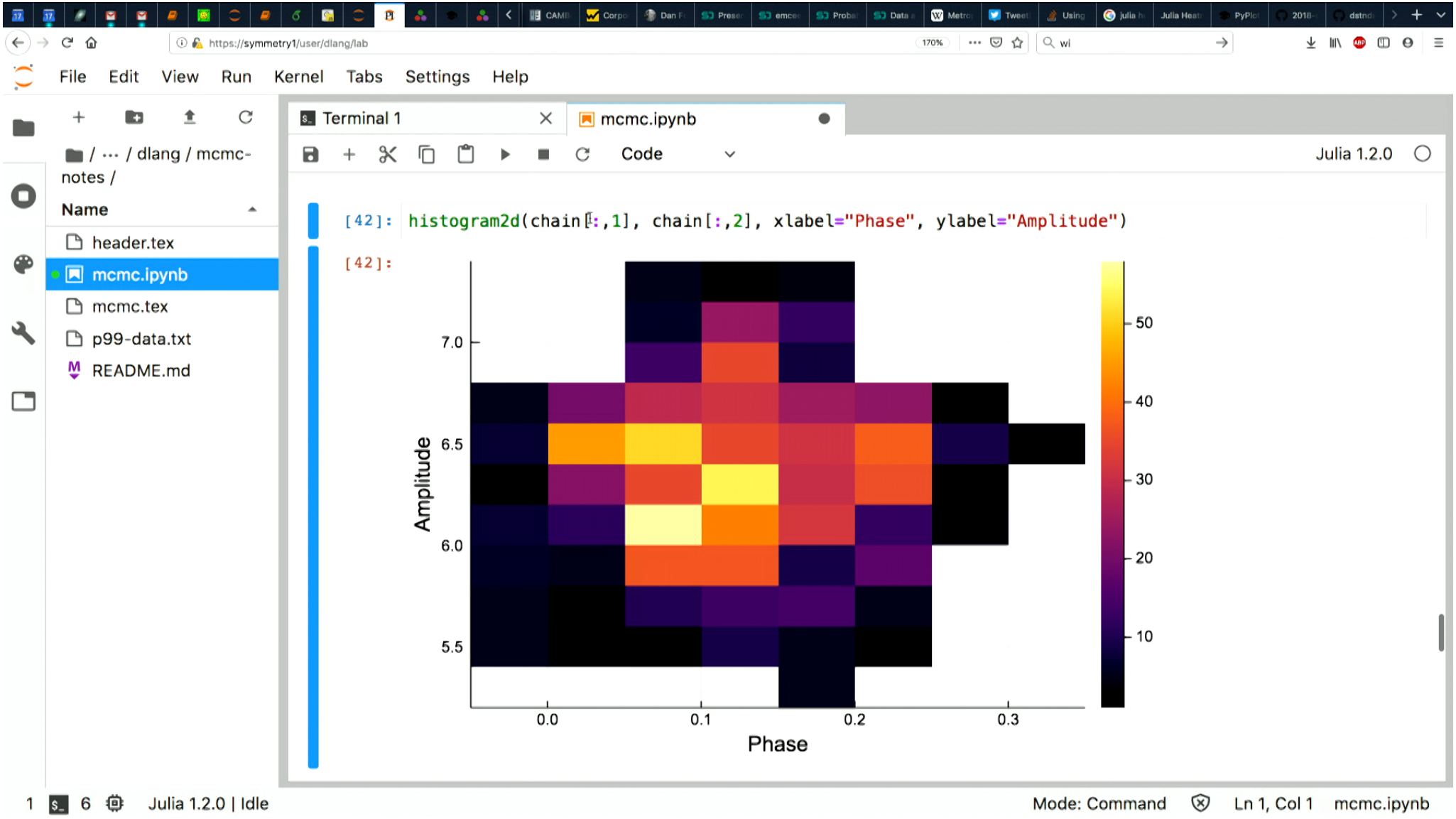
0.104783 6.89053

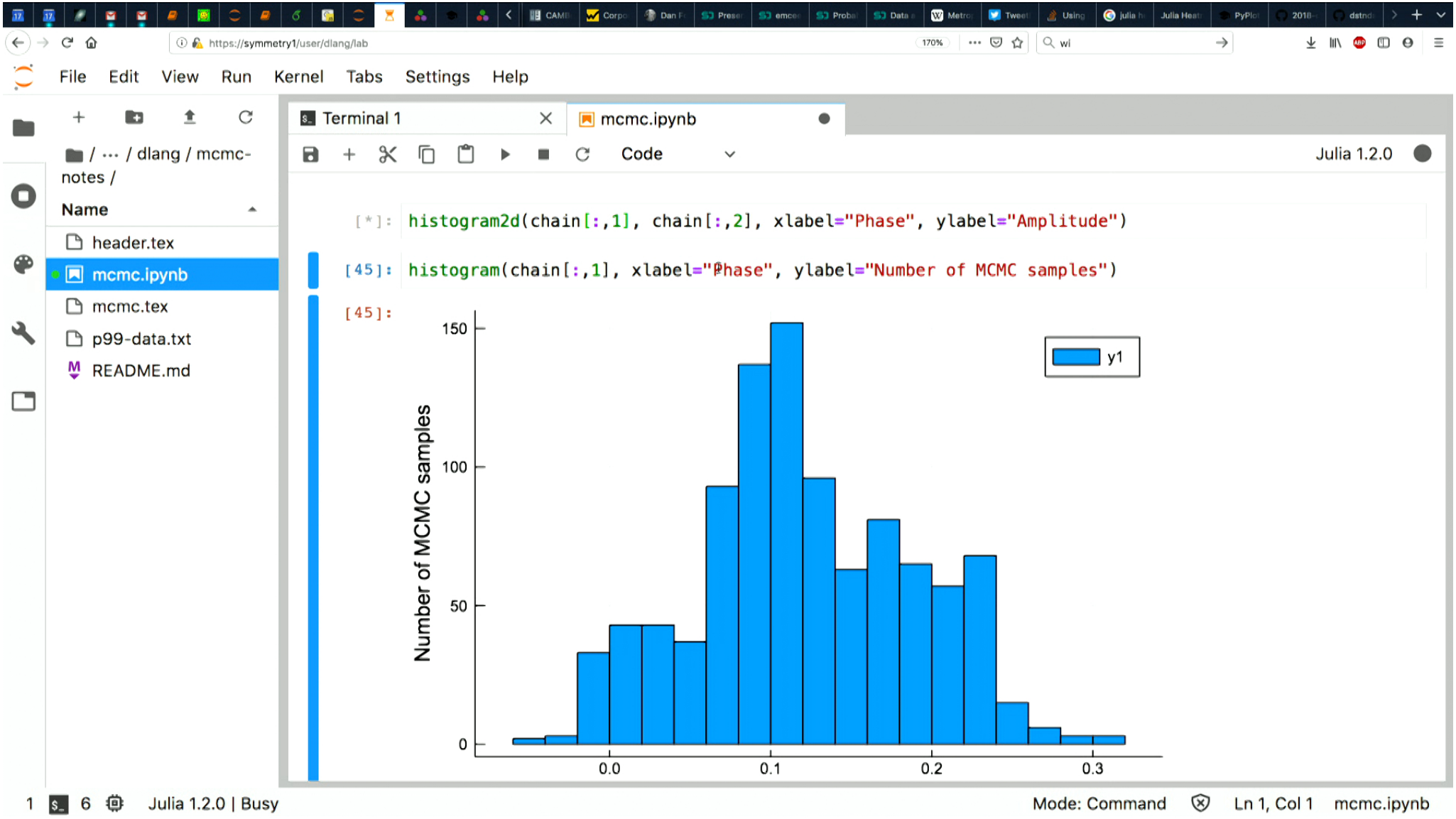
[17]:
```
scatter(chain[:,1], chain[:,2], xlabel="Phase", ylabel="Amplitude")
```

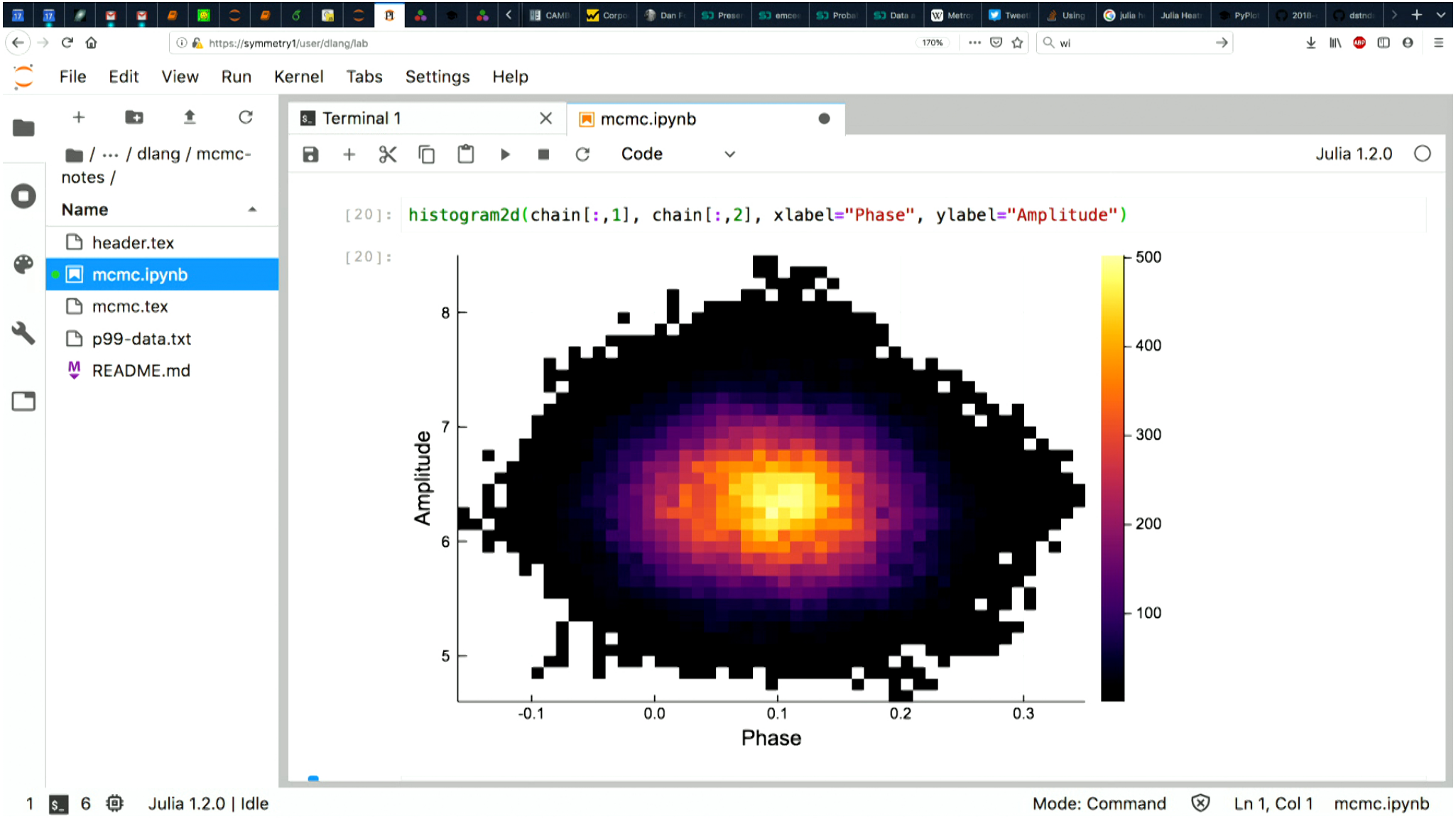
[17]:

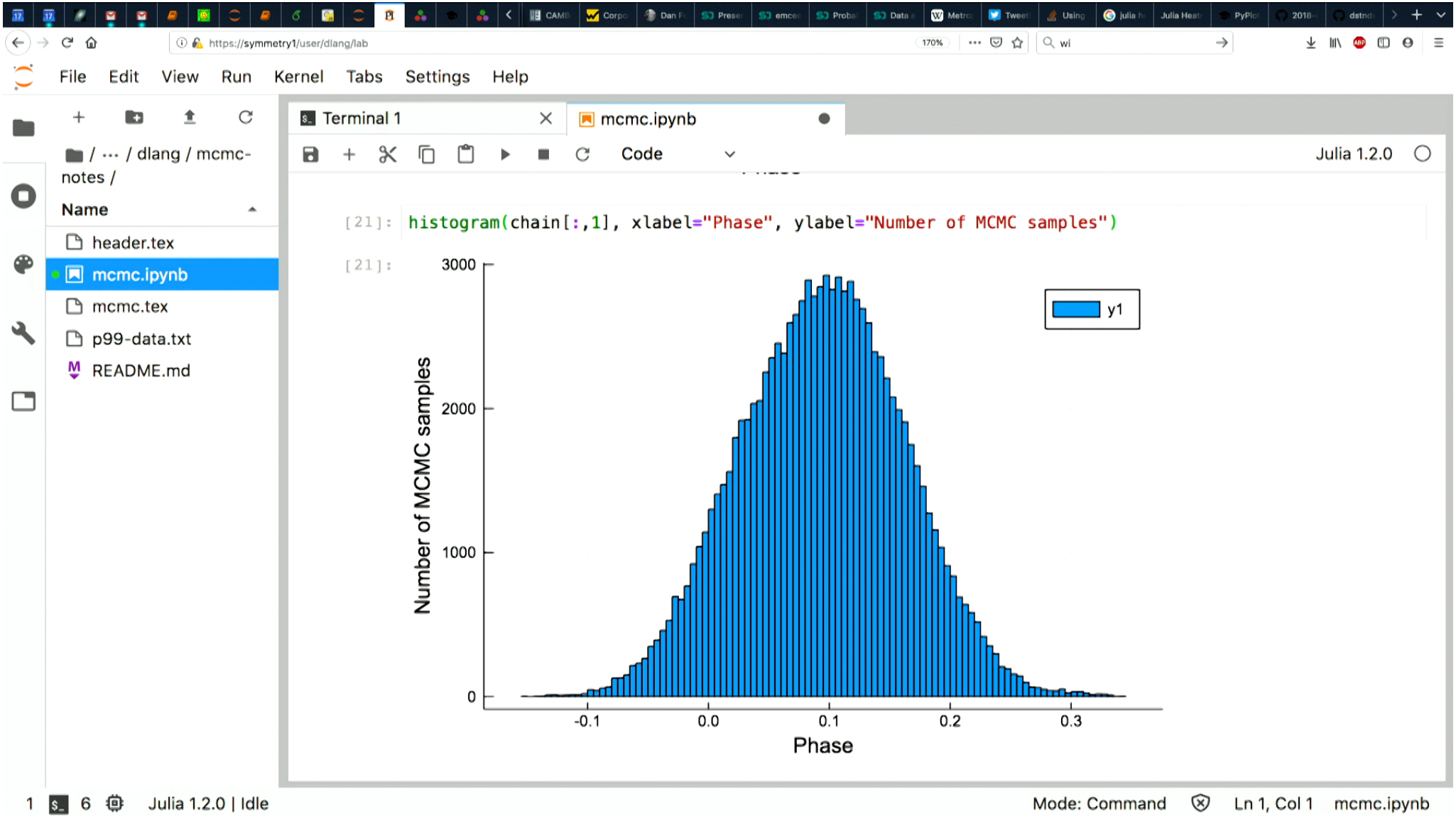


1 6 Julia 1.2.0 | Idle Mode: Command Ln 1, Col 1 mcmc.ipynb


https://symmetry1/user/dlang/lab
File Edit View Run Kernel Tabs Settings Help
Terminal 1
mcmc.ipynb
/ ··· / dlang / mcmc-notes /
Name
header.tex
mcmc.ipynb
mcmc.tex
p99-data.txt
README.md
Code
Julia 1.2.0
[42]: histogram2d(chain[:,1], chain[:,2], xlabel="Phase", ylabel="Amplitude")
[42]:
Amplitude
7.0
6.5
6.0
5.5
0.0
0.1
0.2
0.3
Phase
50
40
30
20
10
1 6 Julia 1.2.0 | Idle
Mode: Command
Ln 1, Col 1
mcmc.ipynb


https://symmetry1/user/dlang/lab
File Edit View Run Kernel Tabs Settings Help
Terminal 1
mcmc.ipynb
/ ··· / dlang / mcmc-notes /
Name
header.tex
mcmc.ipynb
mcmc.tex
p99-data.txt
README.md
Code
Julia 1.2.0
[*]: histogram2d(chain[:,1], chain[:,2], xlabel="Phase", ylabel="Amplitude")
[45]: histogram(chain[:,1], xlabel="Phase", ylabel="Number of MCMC samples")
[45]:
y1
Number of MCMC samples
150
100
50
0
0.0
0.1
0.2
0.3
1 6 Julia 1.2.0 | Busy
Mode: Command
Ln 1, Col 1
mcmc.ipynb


File Edit View Run Kernel Tabs Settings Help
Terminal 1
mcmc.ipynb
Code
Julia 1.2.0
/ ··· / dlang / mcmc-notes /
Name
header.tex
mcmc.ipynb
mcmc.tex
p99-data.txt
README.md
[20]: histogram2d(chain[:,1], chain[:,2], xlabel="Phase", ylabel="Amplitude")
[20]:
Amplitude
Phase
8
7
6
5
-0.1
0.0
0.1
0.2
0.3
500
400
300
200
100
1 6 Julia 1.2.0 | Idle
Mode: Command
Ln 1, Col 1
mcmc.ipynb


https://symmetry1/user/dlang/lab
File Edit View Run Kernel Tabs Settings Help
Terminal 1
mcmc.ipynb
/ ··· / dlang / mcmc-notes /
Name
header.tex
mcmc.ipynb
mcmc.tex
p99-data.txt
README.md
Code
Julia 1.2.0
[21]: histogram(chain[:,1], xlabel="Phase", ylabel="Number of MCMC samples")
[21]:
y1
Number of MCMC samples
3000
2000
1000
0
-0.1
0.0
0.1
0.2
0.3
Phase
1 6 Julia 1.2.0 | Idle
Mode: Command
Ln 1, Col 1
mcmc.ipynb

https://symmetry1/user/dlang/lab

File Edit View Run Kernel Tabs Settings Help

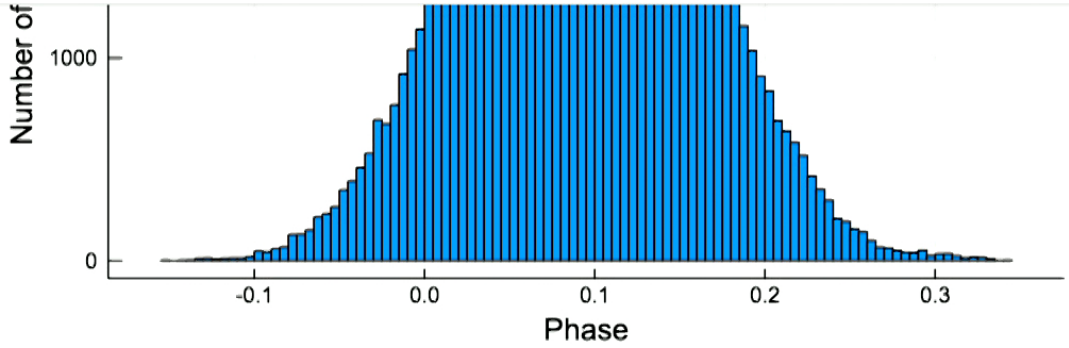
/ ··· / dlang / mcmc-notes /

Name

- header.tex
- mcmc.ipynb
- mcmc.tex
- p99-data.txt
- README.md

Terminal 1 | mcmc.ipynb

Code | Julia 1.2.0




```
[22]: phase_samples = chain[:,1];
```

```
[22]: 100000-element Array{Any,1}:
 -0.00529563841029898
 -0.005535057765940251
  0.0037756773269296664
  0.002952215400349176
  0.018064279802153575
  0.03245155569517563
  0.03455935993776585
  0.02970500128417867
  0.02215830430322016
  0.01680371432401418
  0.026077909314019345
  0.015089532364753406
```

1 6 Julia 1.2.0 | Idle

Mode: Edit Ln 1, Col 28 mcmc.ipynb

https://symmetry1/user/dlang/lab

File Edit View Run Kernel Tabs Settings Help

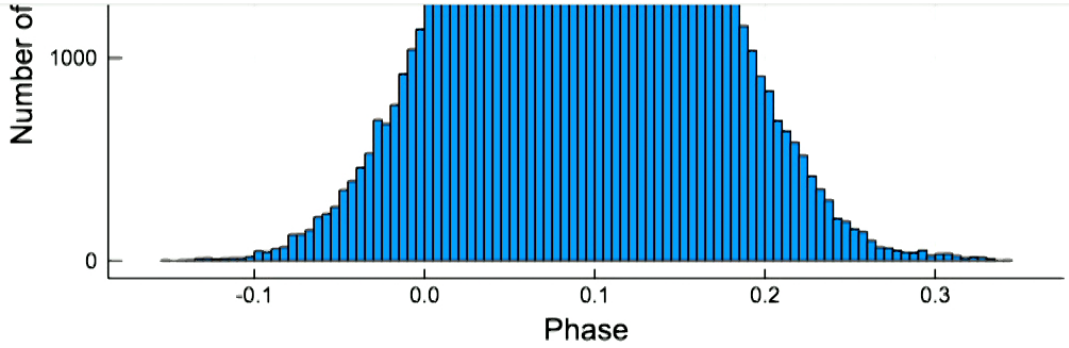
/ ··· / dlang / mcmc-notes /

Name

- header.tex
- mcmc.ipynb
- mcmc.tex
- p99-data.txt
- README.md

Terminal 1 | mcmc.ipynb

Code | Julia 1.2.0



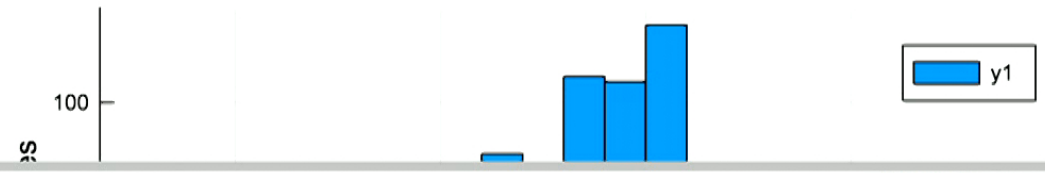
[23]:
```
phase_samples = chain[:,1];
median(phase_samples)
```

```
UndefVarError: median not defined

Stacktrace:
 [1] top-level scope at In[23]:2
```

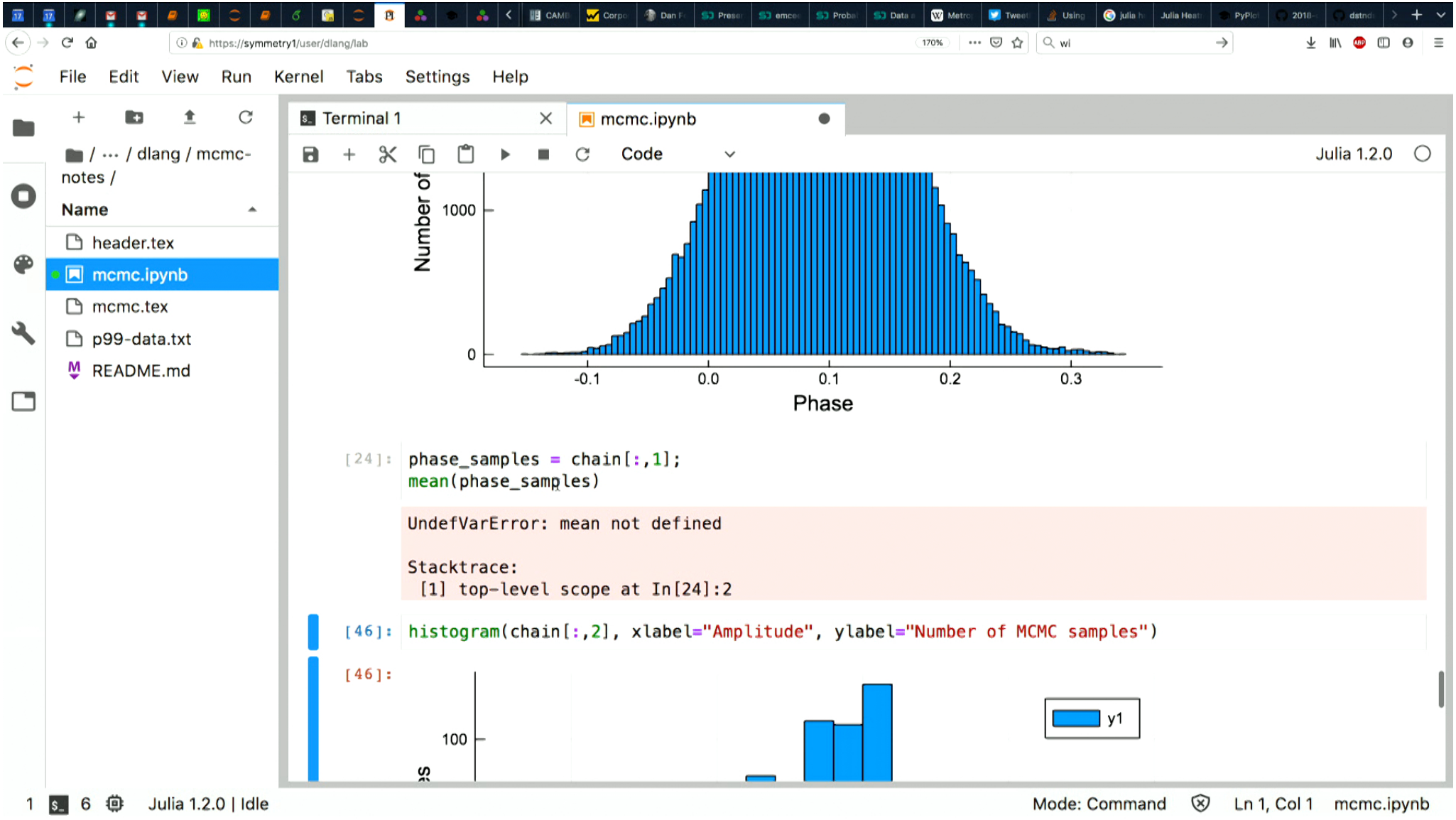
[46]:
```
histogram(chain[:,2], xlabel="Amplitude", ylabel="Number of MCMC samples")
```

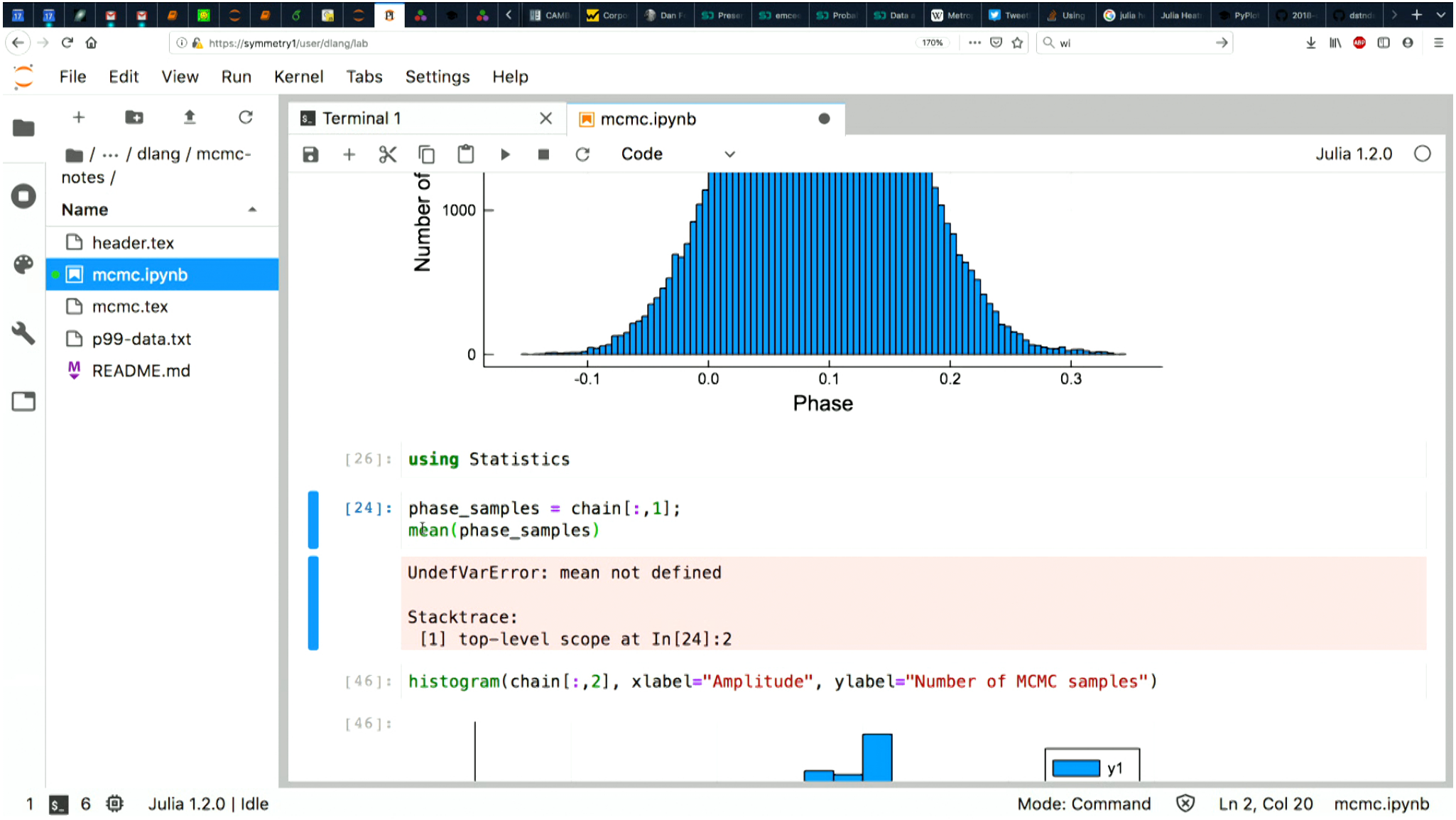
[46]:

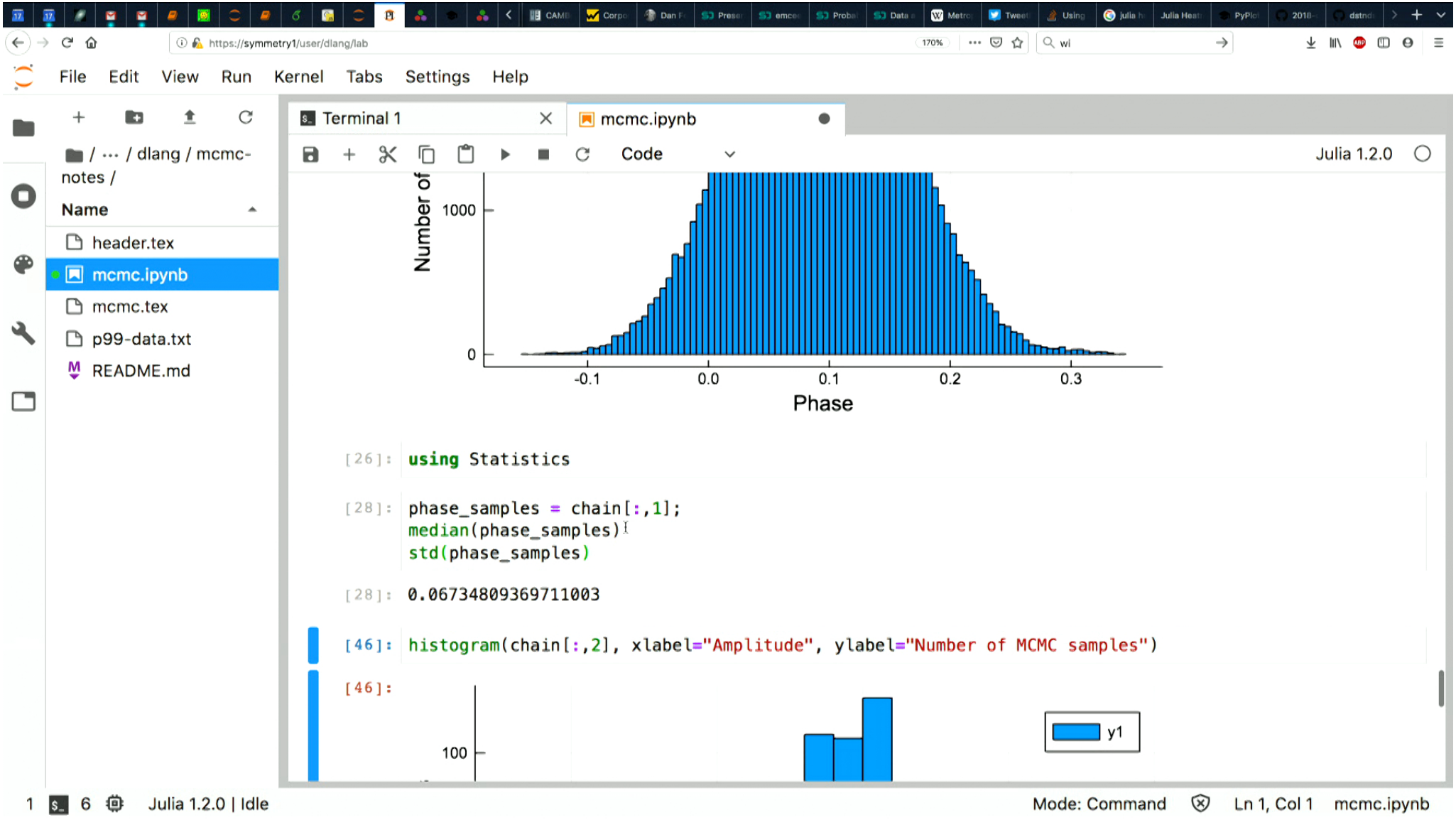


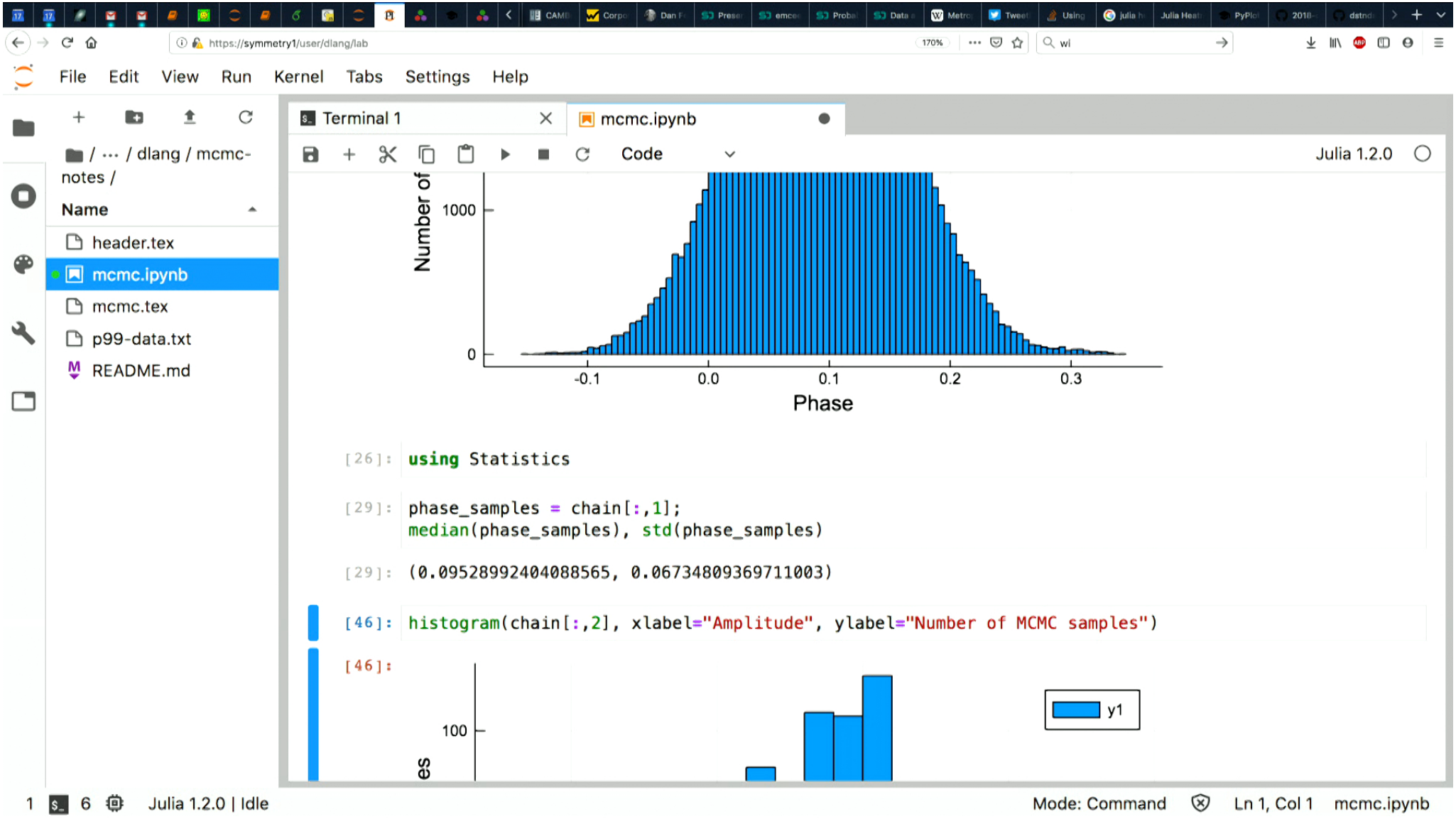
1 6 Julia 1.2.0 | Idle

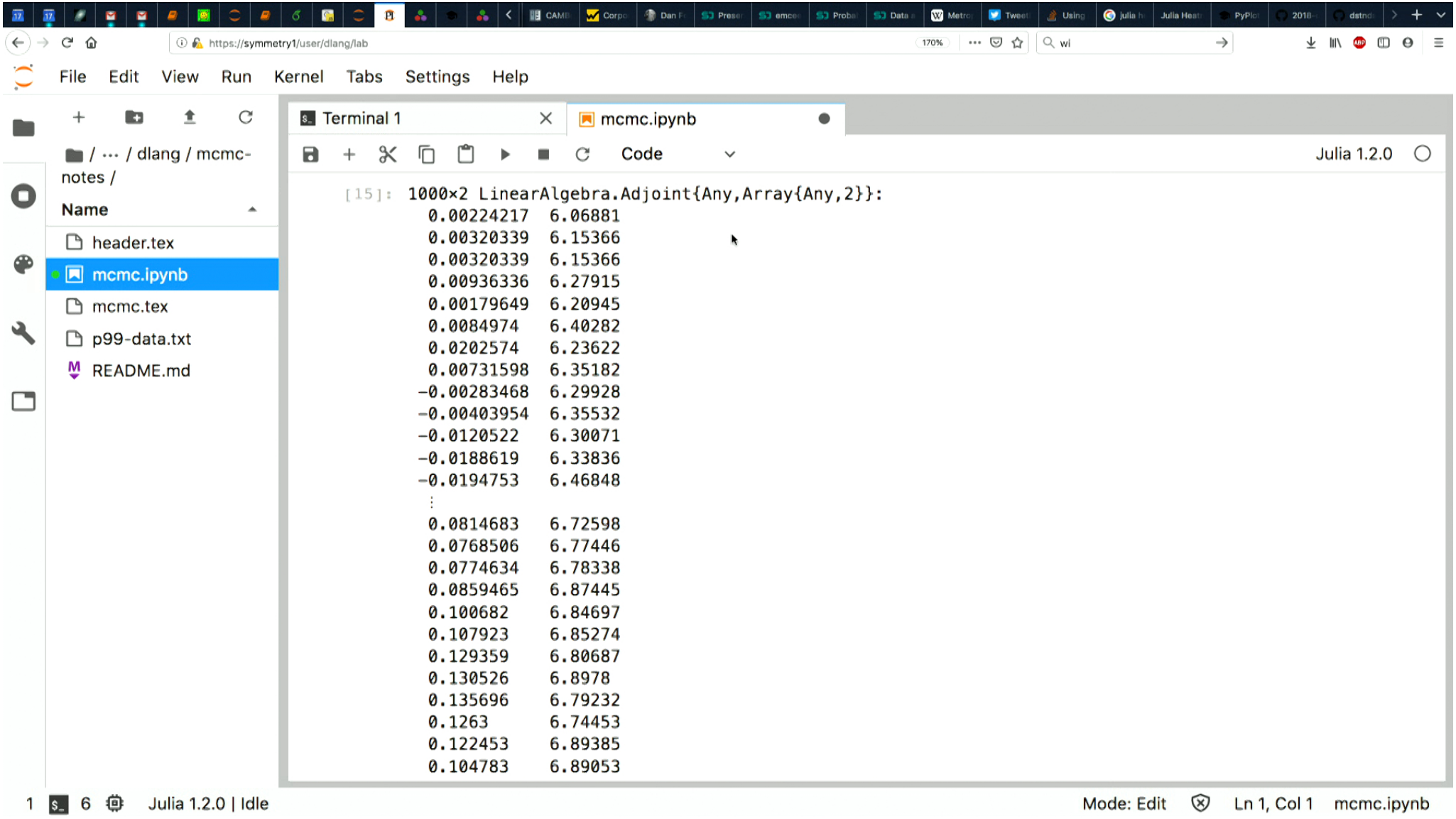
Mode: Command Ln 1, Col 1 mcmc.ipynb

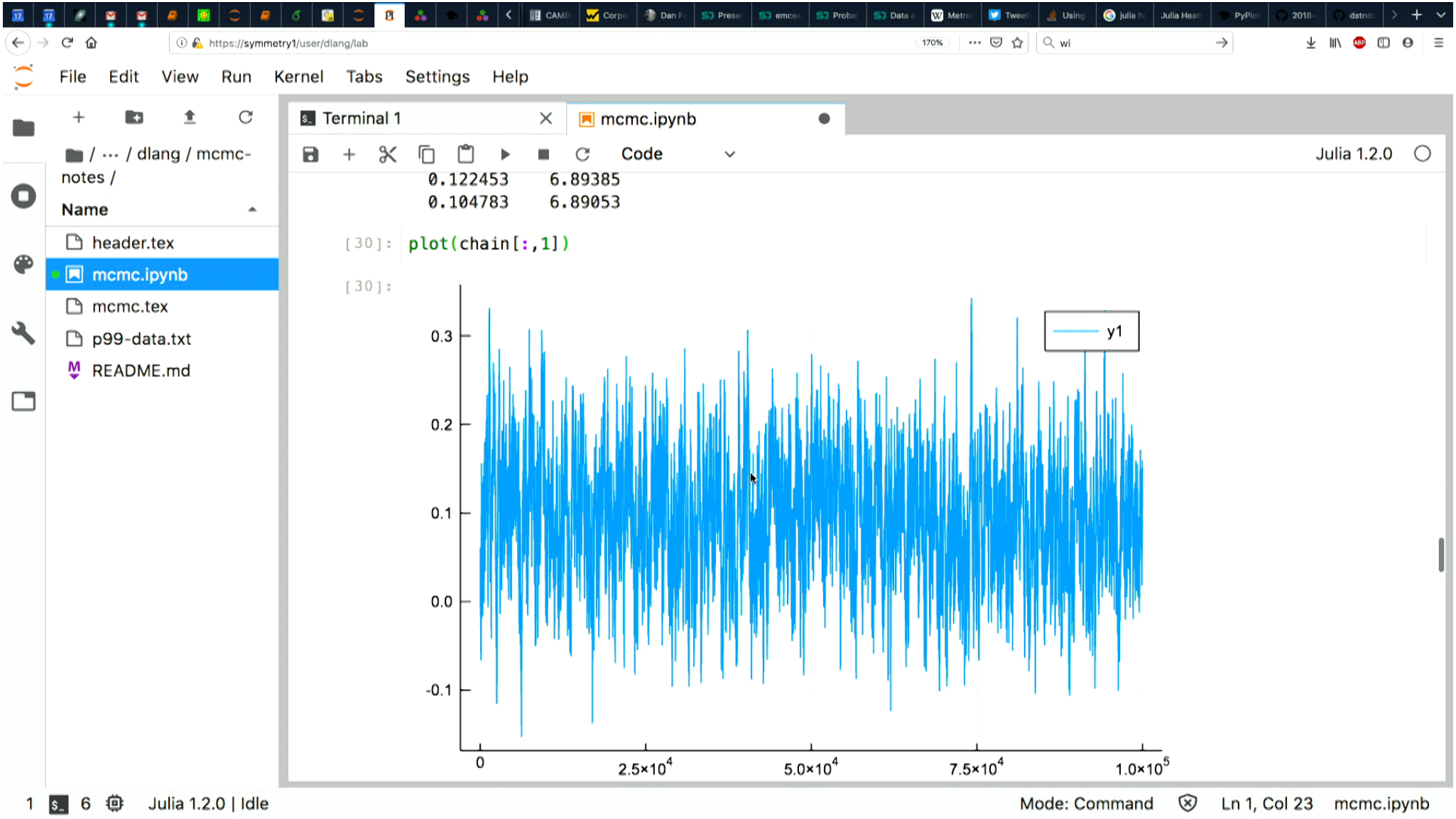

https://symmetry1/user/dlang/lab
File Edit View Run Kernel Tabs Settings Help
Terminal 1
mcmc.ipynb
Code
Julia 1.2.0
/ ··· / dlang / mcmc-notes /
Name
header.tex
mcmc.ipynb
mcmc.tex
p99-data.txt
README.md
Number of
1000
0
-0.1
0.0
0.1
0.2
0.3
Phase
[24]:
phase_samples = chain[:,1];
mean(phase_samples)
UndefVarError: mean not defined
Stacktrace:
[1] top-level scope at In[24]:2
[46]:
histogram(chain[:,2], xlabel="Amplitude", ylabel="Number of MCMC samples")
[46]:
100
y1
1 6 Julia 1.2.0 | Idle
Mode: Command
Ln 1, Col 1
mcmc.ipynb


https://symmetry1/user/dlang/lab
File Edit View Run Kernel Tabs Settings Help
Terminal 1
mcmc.ipynb
/ ··· / dlang / mcmc-notes /
Name
header.tex
mcmc.ipynb
mcmc.tex
p99-data.txt
README.md
Code
Julia 1.2.0
Number of
1000
0
-0.1
0.0
0.1
0.2
0.3
Phase
[26]: using Statistics
[24]: phase_samples = chain[:,1];
mean(phase_samples)
UndefVarError: mean not defined
Stacktrace:
[1] top-level scope at In[24]:2
[46]: histogram(chain[:,2], xlabel="Amplitude", ylabel="Number of MCMC samples")
[46]:
y1
1 6 Julia 1.2.0 | Idle
Mode: Command
Ln 2, Col 20
mcmc.ipynb


https://symmetry1/user/dlang/lab
File Edit View Run Kernel Tabs Settings Help
Terminal 1
mcmc.ipynb
Code
Julia 1.2.0
/ ··· / dlang / mcmc-notes /
Name
header.tex
mcmc.ipynb
mcmc.tex
p99-data.txt
README.md
Number of
1000
0
-0.1
0.0
0.1
0.2
0.3
Phase
[26]: using Statistics
[28]: phase_samples = chain[:,1];
median(phase_samples)
std(phase_samples)
[28]: 0.06734809369711003
[46]: histogram(chain[:,2], xlabel="Amplitude", ylabel="Number of MCMC samples")
[46]:
100
y1
1 6 Julia 1.2.0 | Idle
Mode: Command
Ln 1, Col 1
mcmc.ipynb


https://symmetry1/user/dlang/lab
File Edit View Run Kernel Tabs Settings Help
Terminal 1
mcmc.ipynb
Code
Julia 1.2.0
/ ··· / dlang / mcmc-notes /
Name
header.tex
mcmc.ipynb
mcmc.tex
p99-data.txt
README.md
Number of
1000
0
-0.1
0.0
0.1
0.2
0.3
Phase
[26]: using Statistics
[29]: phase_samples = chain[:,1];
median(phase_samples), std(phase_samples)
[29]: (0.09528992404088565, 0.06734809369711003)
[46]: histogram(chain[:,2], xlabel="Amplitude", ylabel="Number of MCMC samples")
[46]:
100
es
y1
1 6 Julia 1.2.0 | Idle
Mode: Command
Ln 1, Col 1
mcmc.ipynb


https://symmetry1/user/dlang/lab
File Edit View Run Kernel Tabs Settings Help
Terminal 1
mcmc.ipynb
/ ··· / dlang / mcmc-notes /
Name
header.tex
mcmc.ipynb
mcmc.tex
p99-data.txt
README.md
Code
Julia 1.2.0
[15]: 1000×2 LinearAlgebra.Adjoint{Any,Array{Any,2}}:
0.00224217 6.06881
0.00320339 6.15366
0.00320339 6.15366
0.00936336 6.27915
0.00179649 6.20945
0.0084974 6.40282
0.0202574 6.23622
0.00731598 6.35182
-0.00283468 6.29928
-0.00403954 6.35532
-0.0120522 6.30071
-0.0188619 6.33836
-0.0194753 6.46848
0.0814683 6.72598
0.0768506 6.77446
0.0774634 6.78338
0.0859465 6.87445
0.100682 6.84697
0.107923 6.85274
0.129359 6.80687
0.130526 6.8978
0.135696 6.79232
0.1263 6.74453
0.122453 6.89385
0.104783 6.89053
1 6 Julia 1.2.0 | Idle
Mode: Edit
Ln 1, Col 1
mcmc.ipynb


https://symmetry1/user/dlang/lab
File Edit View Run Kernel Tabs Settings Help
Terminal 1
mcmc.ipynb
/ ··· / dlang / mcmc-notes /
Name
header.tex
mcmc.ipynb
mcmc.tex
p99-data.txt
README.md
Code
Julia 1.2.0
0.122453 6.89385
0.104783 6.89053
[30]: plot(chain[:,1])
[30]:
y1
0.3
0.2
0.1
0.0
-0.1
0
2.5×10^4
5.0×10^4
7.5×10^4
1.0×10^5
1 6 Julia 1.2.0 | Idle
Mode: Command
Ln 1, Col 23
mcmc.ipynb

https://symmetry1/user/dlang/lab

File Edit View Run Kernel Tabs Settings Help

Terminal 1 | mcmc.ipynb

/ ··· / dlang / mcmc-notes /

Name

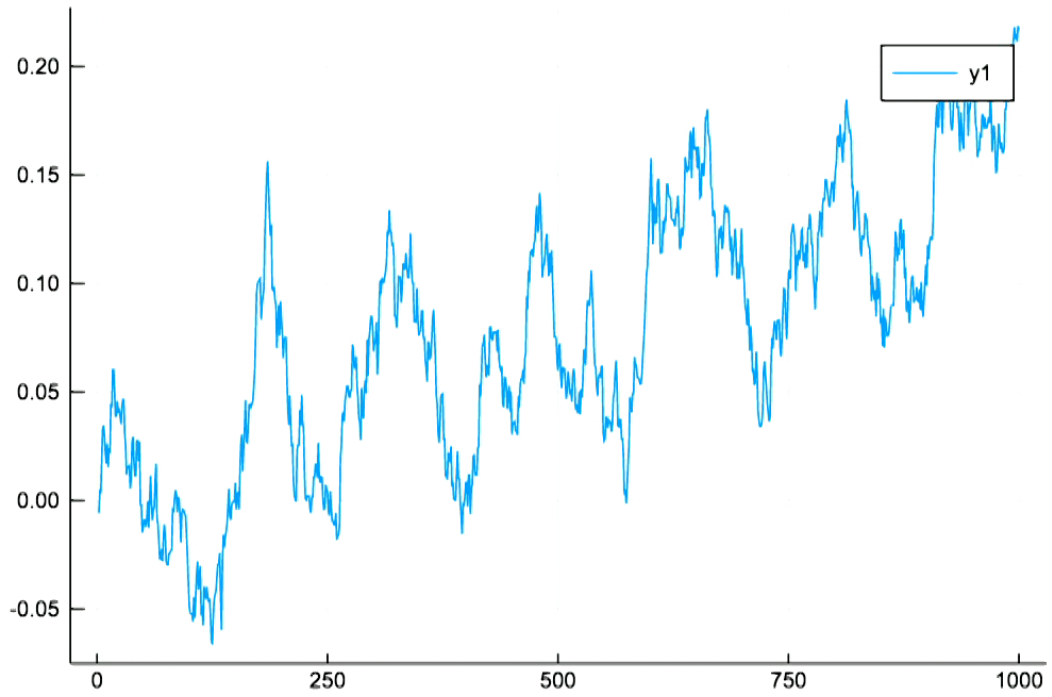
- header.tex
- mcmc.ipynb
- mcmc.tex
- p99-data.txt
- README.md

Code | Julia 1.2.0

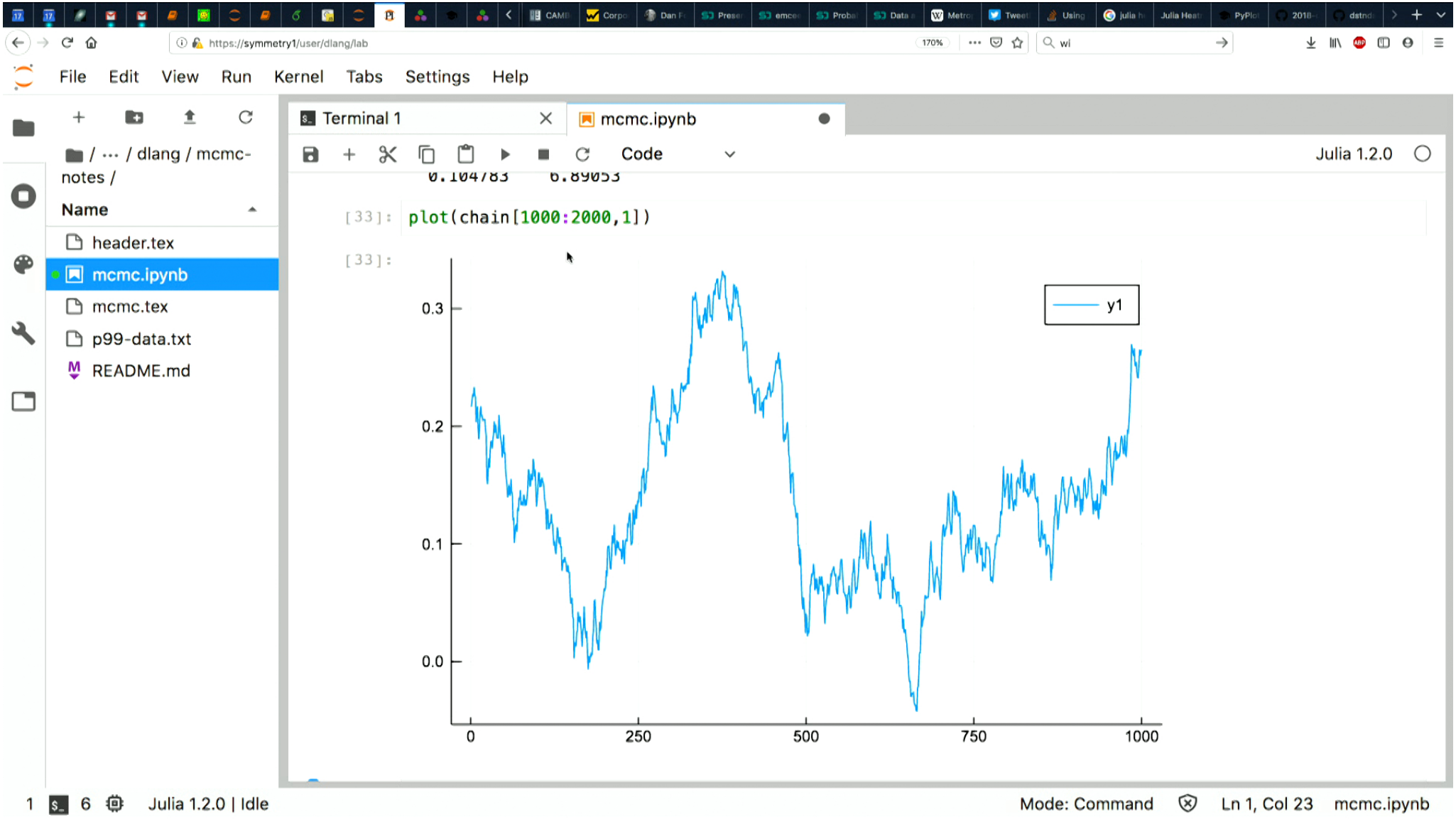
0.104783 6.89053

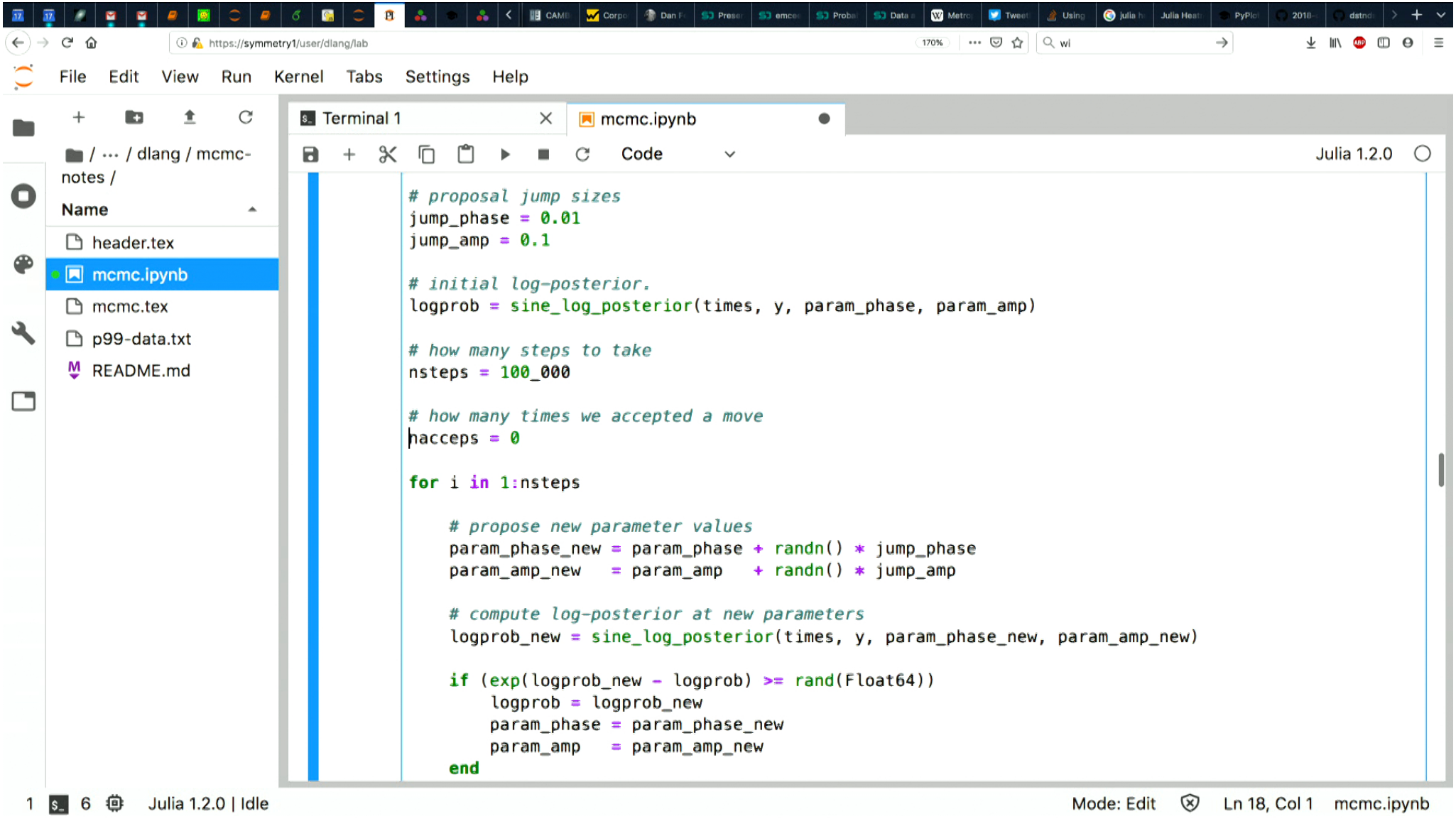
[32]:
```
plot(chain[1:1000,1])
```

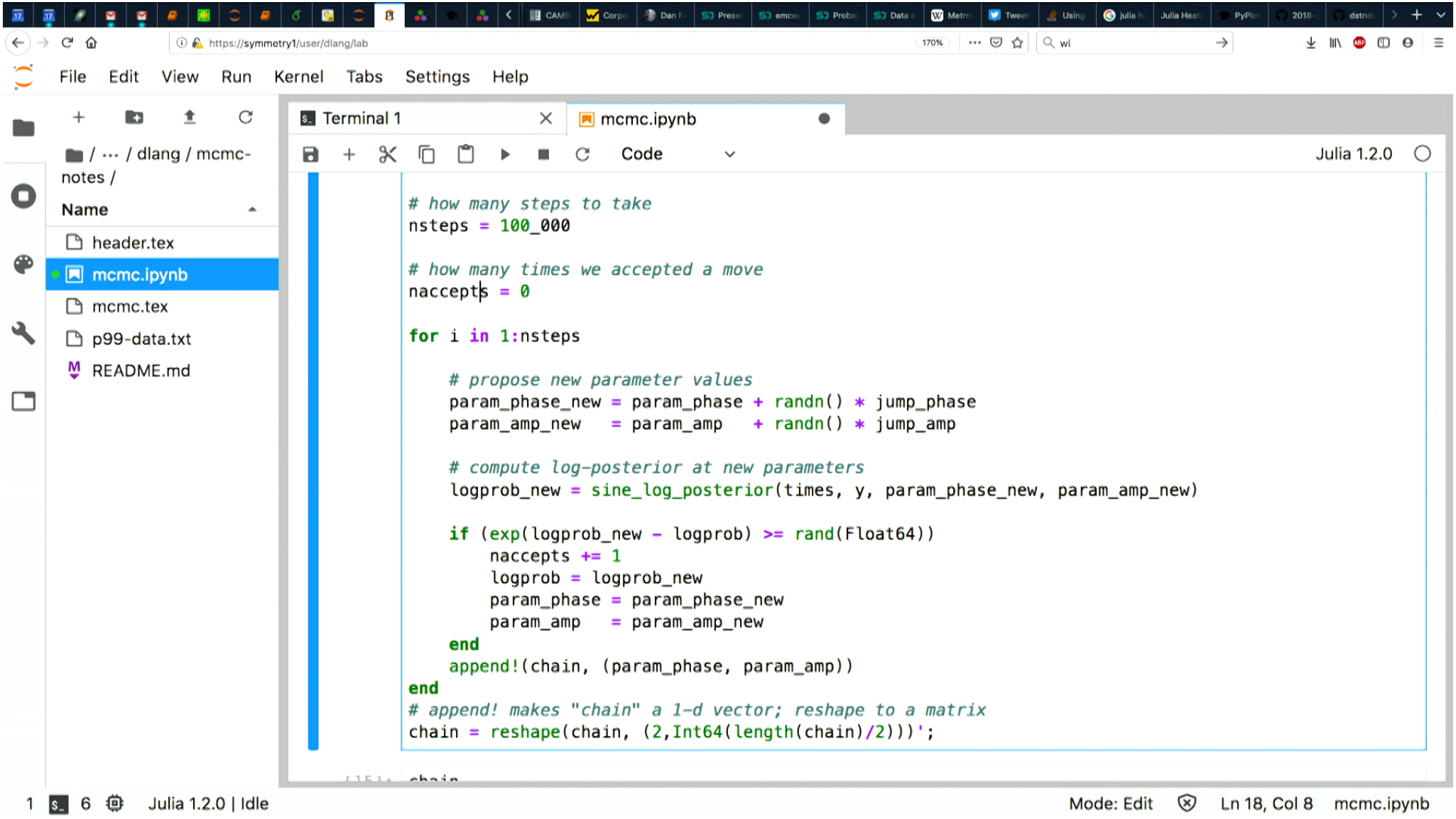
[32]:

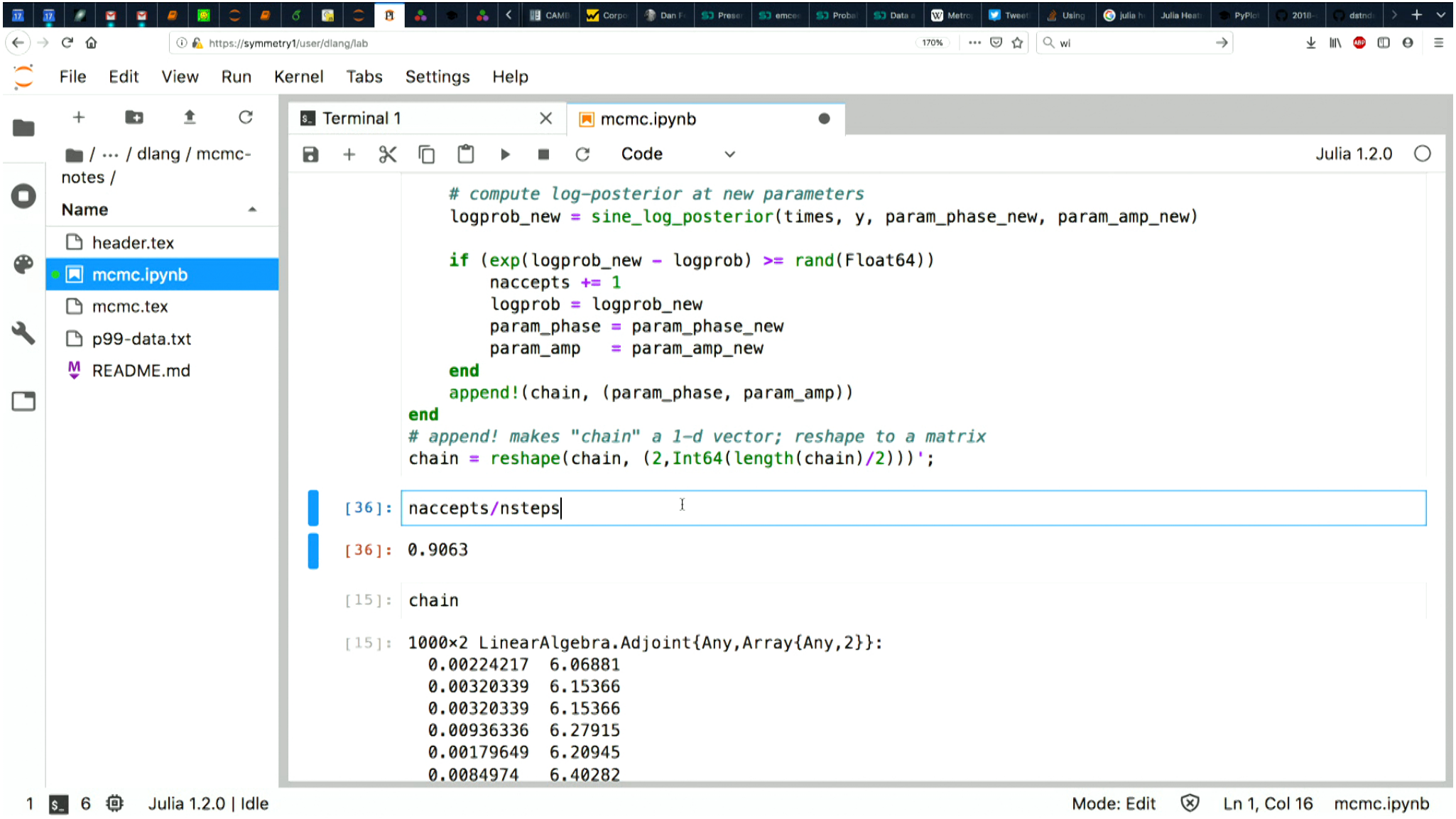


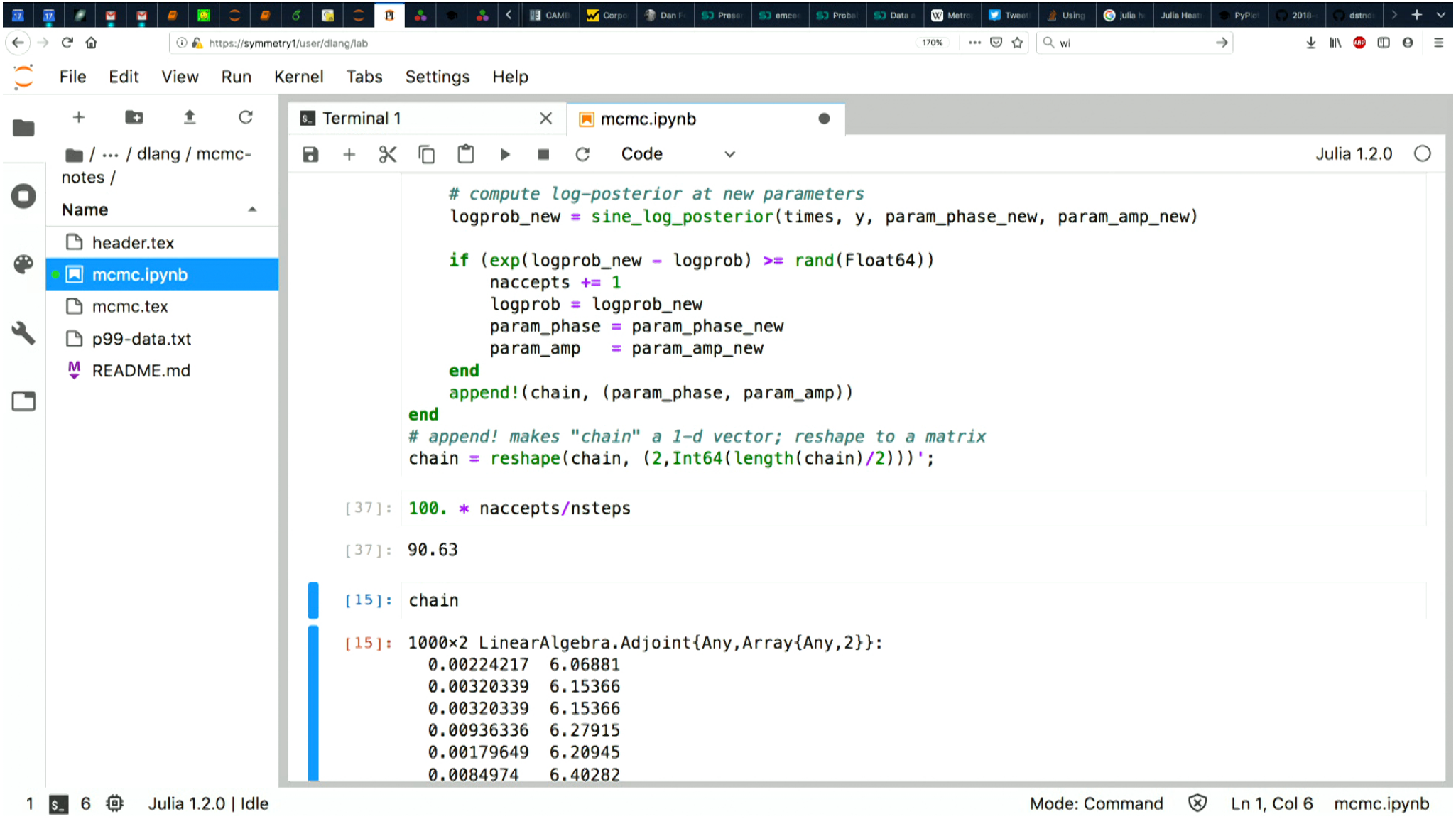
1 6 Julia 1.2.0 | Idle Mode: Command Ln 1, Col 23 mcmc.ipynb

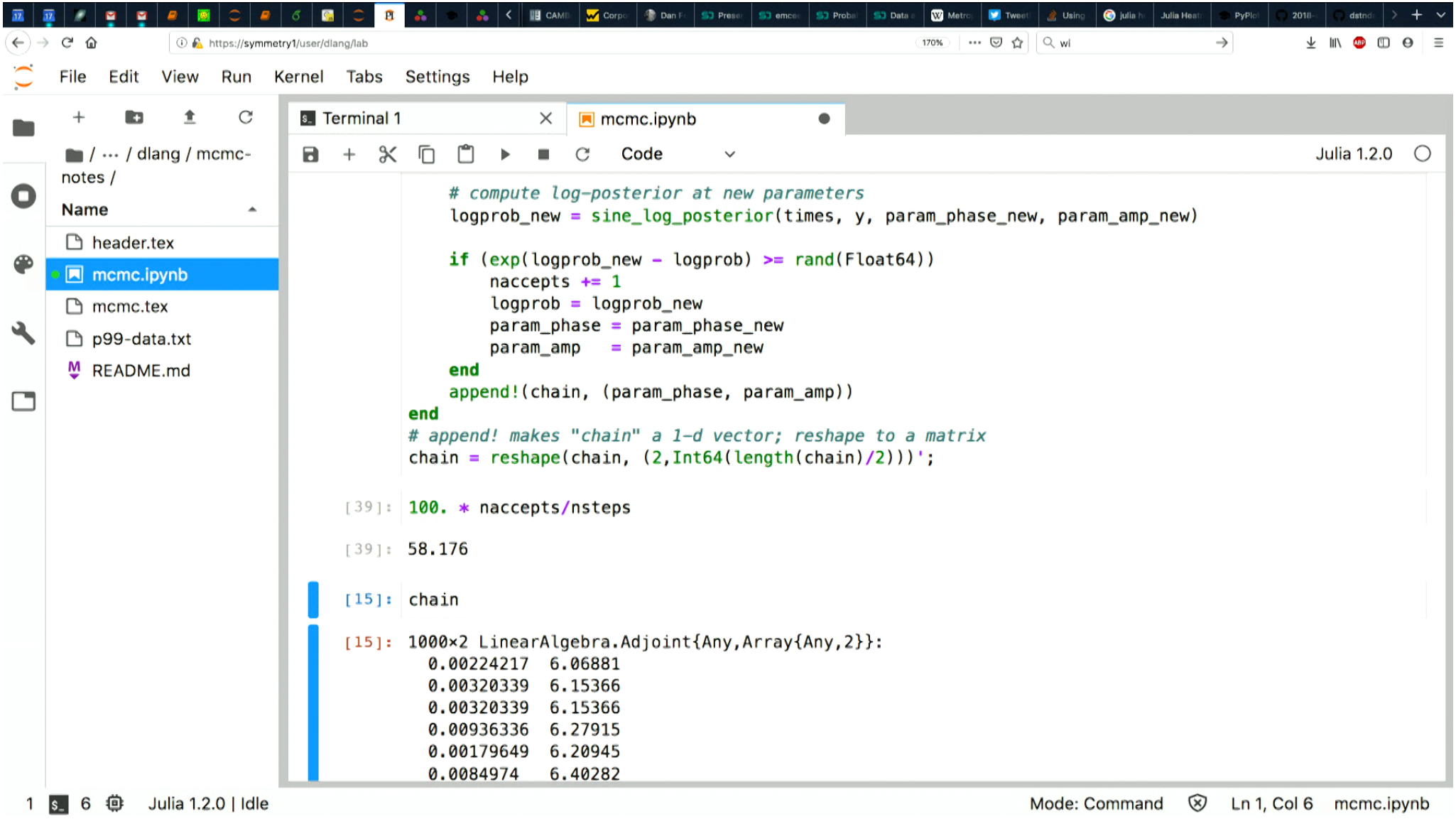

https://symmetry1/user/dlang/lab
File Edit View Run Kernel Tabs Settings Help
Terminal 1
mcmc.ipynb
Code
Julia 1.2.0
/ ··· / dlang / mcmc-notes /
Name
header.tex
mcmc.ipynb
mcmc.tex
p99-data.txt
README.md
[33]: plot(chain[1000:2000,1])
[33]:
y1
0.3
0.2
0.1
0.0
0
250
500
750
1000
1 6 Julia 1.2.0 | Idle
Mode: Command
Ln 1, Col 23
mcmc.ipynb

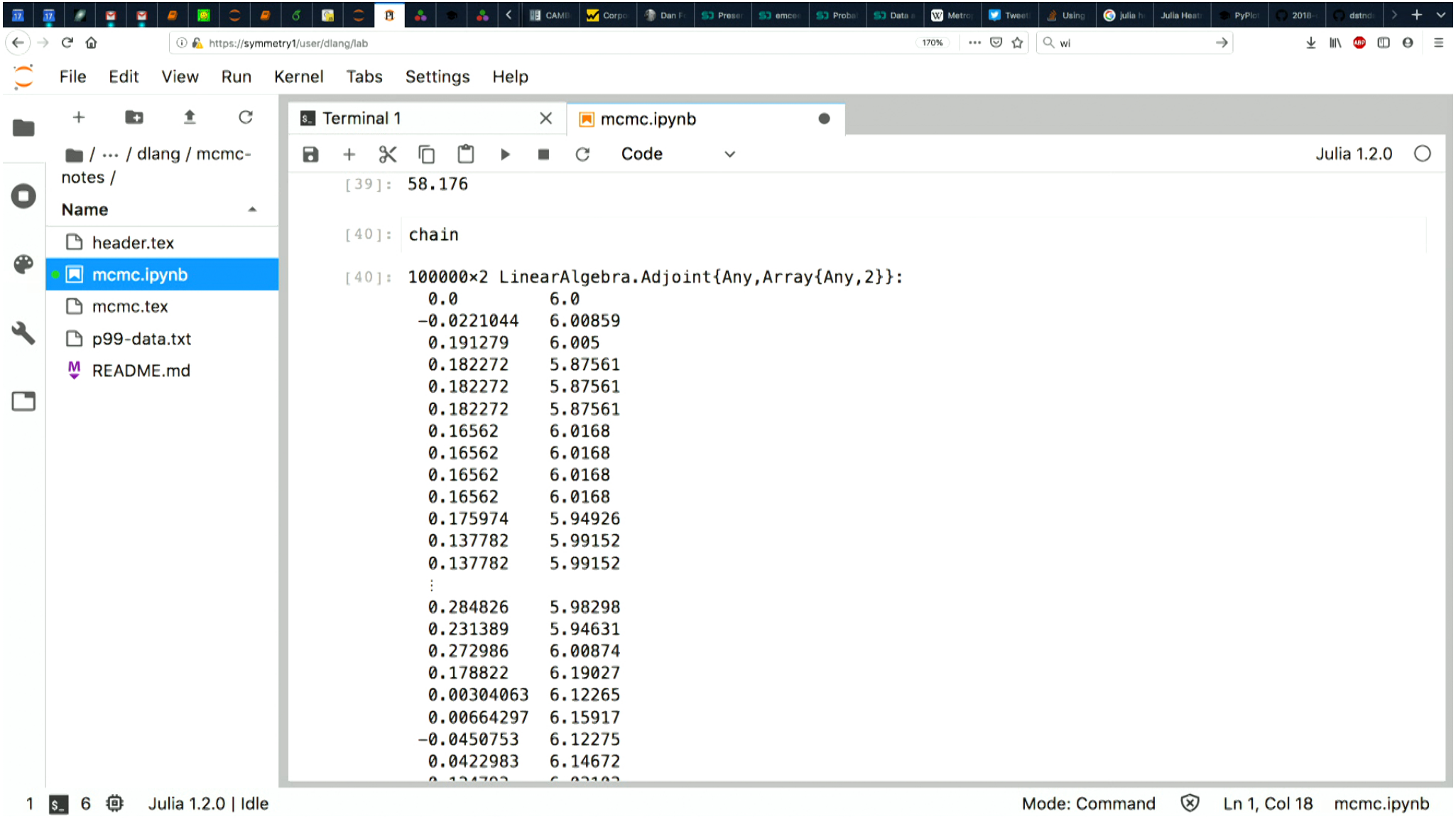

https://symmetry1/user/dlang/lab
File Edit View Run Kernel Tabs Settings Help
Terminal 1
mcmc.ipynb
/ ··· / dlang / mcmc-notes /
Name
header.tex
mcmc.ipynb
mcmc.tex
p99-data.txt
README.md
Code
Julia 1.2.0
# proposal jump sizes
jump_phase = 0.01
jump_amp = 0.1
# initial log-posterior.
logprob = sine_log_posterior(times, y, param_phase, param_amp)
# how many steps to take
nsteps = 100_000
# how many times we accepted a move
nacceps = 0
for i in 1:nsteps
# propose new parameter values
param_phase_new = param_phase + randn() * jump_phase
param_amp_new = param_amp + randn() * jump_amp
# compute log-posterior at new parameters
logprob_new = sine_log_posterior(times, y, param_phase_new, param_amp_new)
if (exp(logprob_new - logprob) >= rand(Float64))
logprob = logprob_new
param_phase = param_phase_new
param_amp = param_amp_new
end
1 6 Julia 1.2.0 | Idle
Mode: Edit
Ln 18, Col 1
mcmc.ipynb

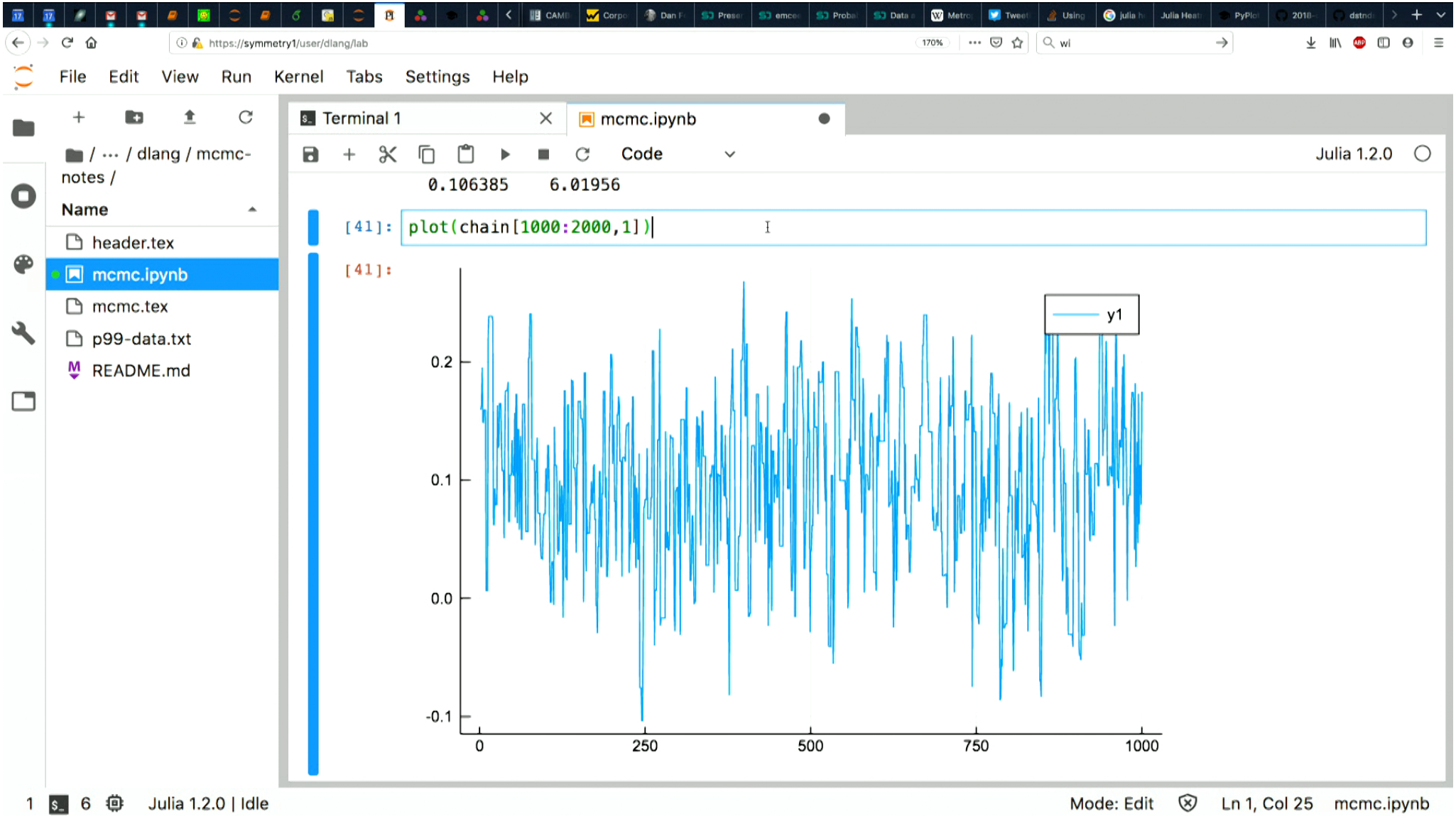

https://symmetry1/user/dlang/lab
File Edit View Run Kernel Tabs Settings Help
Terminal 1
mcmc.ipynb
/ ··· / dlang / mcmc-notes /
Name
header.tex
mcmc.ipynb
mcmc.tex
p99-data.txt
README.md
Code
Julia 1.2.0
# how many steps to take
nsteps = 100_000
# how many times we accepted a move
naccepts = 0
for i in 1:nsteps
# propose new parameter values
param_phase_new = param_phase + randn() * jump_phase
param_amp_new = param_amp + randn() * jump_amp
# compute log-posterior at new parameters
logprob_new = sine_log_posterior(times, y, param_phase_new, param_amp_new)
if (exp(logprob_new - logprob) >= rand(Float64))
naccepts += 1
logprob = logprob_new
param_phase = param_phase_new
param_amp = param_amp_new
end
append!(chain, (param_phase, param_amp))
end
# append! makes "chain" a 1-d vector; reshape to a matrix
chain = reshape(chain, (2,Int64(length(chain)/2)))';
chain
1 6 Julia 1.2.0 | Idle
Mode: Edit
Ln 18, Col 8
mcmc.ipynb

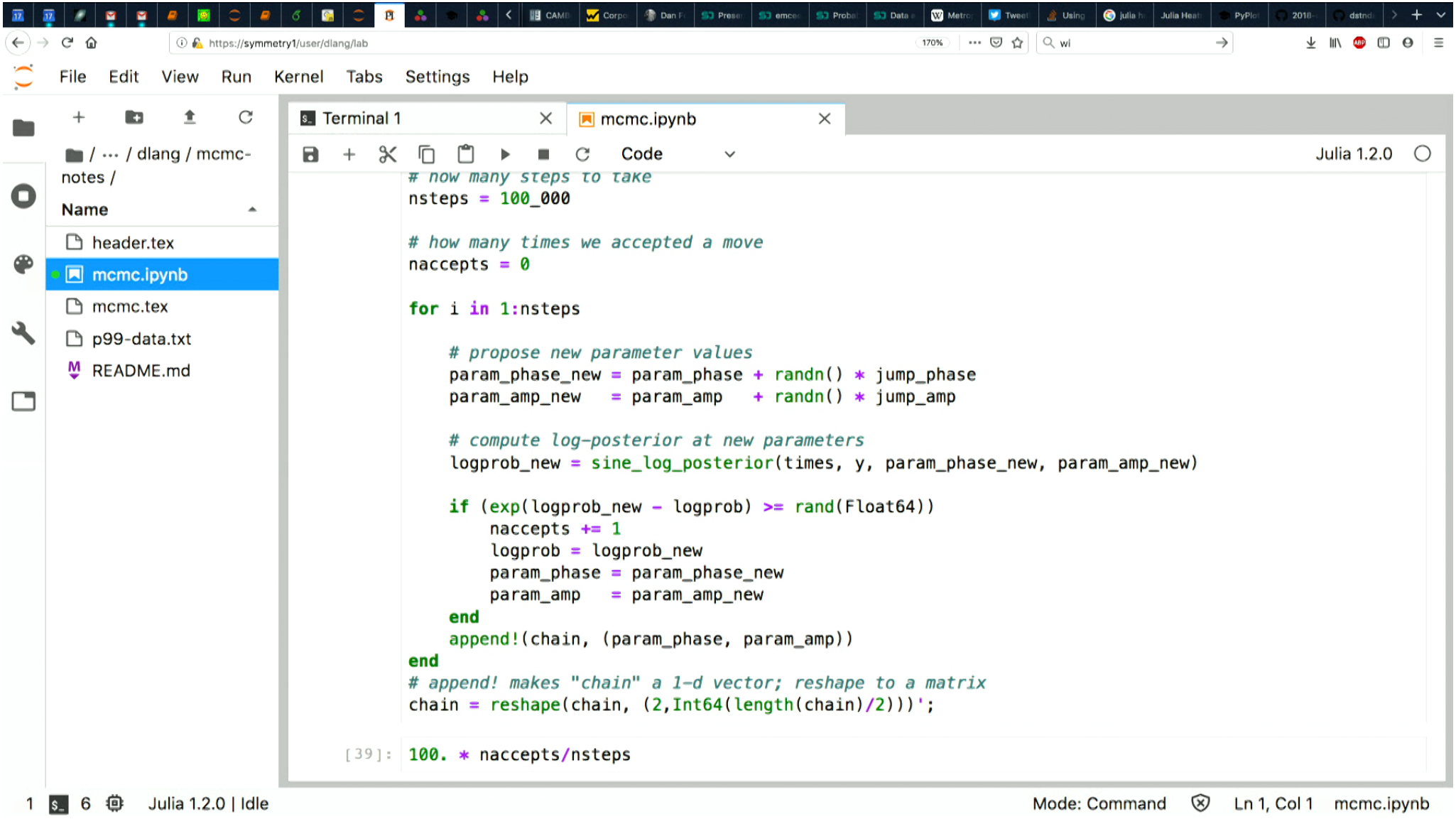

https://symmetry1/user/dlang/lab
File Edit View Run Kernel Tabs Settings Help
Terminal 1
mcmc.ipynb
/ ··· / dlang / mcmc-notes /
Name
header.tex
mcmc.ipynb
mcmc.tex
p99-data.txt
README.md
Code
Julia 1.2.0
# compute log-posterior at new parameters
logprob_new = sine_log_posterior(times, y, param_phase_new, param_amp_new)
if (exp(logprob_new - logprob) >= rand(Float64))
naccepts += 1
logprob = logprob_new
param_phase = param_phase_new
param_amp = param_amp_new
end
append!(chain, (param_phase, param_amp))
end
# append! makes "chain" a 1-d vector; reshape to a matrix
chain = reshape(chain, (2,Int64(length(chain)/2)))';
[36]: naccepts/nsteps
[36]: 0.9063
[15]: chain
[15]: 1000×2 LinearAlgebra.Adjoint{Any,Array{Any,2}}:
0.00224217 6.06881
0.00320339 6.15366
0.00320339 6.15366
0.00936336 6.27915
0.00179649 6.20945
0.0084974 6.40282
1 6 Julia 1.2.0 | Idle
Mode: Edit
Ln 1, Col 16
mcmc.ipynb


https://symmetry1/user/dlang/lab
File Edit View Run Kernel Tabs Settings Help
Terminal 1
mcmc.ipynb
/ ··· / dlang / mcmc-notes /
Name
header.tex
mcmc.ipynb
mcmc.tex
p99-data.txt
README.md
Code
Julia 1.2.0
# compute log-posterior at new parameters
logprob_new = sine_log_posterior(times, y, param_phase_new, param_amp_new)
if (exp(logprob_new - logprob) >= rand(Float64))
naccepts += 1
logprob = logprob_new
param_phase = param_phase_new
param_amp = param_amp_new
end
append!(chain, (param_phase, param_amp))
end
# append! makes "chain" a 1-d vector; reshape to a matrix
chain = reshape(chain, (2,Int64(length(chain)/2)))';
[37]: 100. * naccepts/nsteps
[37]: 90.63
[15]: chain
[15]: 1000×2 LinearAlgebra.Adjoint{Any,Array{Any,2}}:
0.00224217 6.06881
0.00320339 6.15366
0.00320339 6.15366
0.00936336 6.27915
0.00179649 6.20945
0.0084974 6.40282
1 6 Julia 1.2.0 | Idle
Mode: Command
Ln 1, Col 6
mcmc.ipynb


https://symmetry1/user/dlang/lab
File Edit View Run Kernel Tabs Settings Help
Terminal 1
mcmc.ipynb
/ ··· / dlang / mcmc-notes /
Name
header.tex
mcmc.ipynb
mcmc.tex
p99-data.txt
README.md
Code
Julia 1.2.0
# compute log-posterior at new parameters
logprob_new = sine_log_posterior(times, y, param_phase_new, param_amp_new)
if (exp(logprob_new - logprob) >= rand(Float64))
naccepts += 1
logprob = logprob_new
param_phase = param_phase_new
param_amp = param_amp_new
end
append!(chain, (param_phase, param_amp))
end
# append! makes "chain" a 1-d vector; reshape to a matrix
chain = reshape(chain, (2,Int64(length(chain)/2)))';
[39]: 100. * naccepts/nsteps
[39]: 58.176
[15]: chain
[15]: 1000×2 LinearAlgebra.Adjoint{Any,Array{Any,2}}:
0.00224217 6.06881
0.00320339 6.15366
0.00320339 6.15366
0.00936336 6.27915
0.00179649 6.20945
0.0084974 6.40282
1 6 Julia 1.2.0 | Idle
Mode: Command
Ln 1, Col 6
mcmc.ipynb


https://symmetry1/user/dlang/lab
File Edit View Run Kernel Tabs Settings Help
/ ··· / dlang / mcmc-notes /
Name
header.tex
mcmc.ipynb
mcmc.tex
p99-data.txt
README.md
Terminal 1
mcmc.ipynb
Code
Julia 1.2.0
[39]: 58.176
[40]: chain
[40]: 100000×2 LinearAlgebra.Adjoint{Any,Array{Any,2}}:
0.0 6.0
-0.0221044 6.00859
0.191279 6.005
0.182272 5.87561
0.182272 5.87561
0.182272 5.87561
0.16562 6.0168
0.16562 6.0168
0.16562 6.0168
0.16562 6.0168
0.175974 5.94926
0.137782 5.99152
0.137782 5.99152
0.284826 5.98298
0.231389 5.94631
0.272986 6.00874
0.178822 6.19027
0.00304063 6.12265
0.00664297 6.15917
-0.0450753 6.12275
0.0422983 6.14672
1 6 Julia 1.2.0 | Idle
Mode: Command Ln 1, Col 18 mcmc.ipynb


https://symmetry1/user/dlang/lab
File Edit View Run Kernel Tabs Settings Help
Terminal 1
mcmc.ipynb
/ ··· / dlang / mcmc-notes /
Name
header.tex
mcmc.ipynb
mcmc.tex
p99-data.txt
README.md
Code
Julia 1.2.0
0.106385 6.01956
[41]: plot(chain[1000:2000,1])
[41]:
y1
0.2
0.1
0.0
-0.1
0
250
500
750
1000
1 6 Julia 1.2.0 | Idle
Mode: Edit
Ln 1, Col 25
mcmc.ipynb


https://symmetry1/user/dlang/lab
File Edit View Run Kernel Tabs Settings Help
/ ··· / dlang / mcmc-notes /
Name
header.tex
mcmc.ipynb
mcmc.tex
p99-data.txt
README.md
Terminal 1
mcmc.ipynb
Code
Julia 1.2.0
# how many steps to take
nsteps = 100_000
# how many times we accepted a move
naccepts = 0
for i in 1:nsteps
# propose new parameter values
param_phase_new = param_phase + randn() * jump_phase
param_amp_new = param_amp + randn() * jump_amp
# compute log-posterior at new parameters
logprob_new = sine_log_posterior(times, y, param_phase_new, param_amp_new)
if (exp(logprob_new - logprob) >= rand(Float64))
naccepts += 1
logprob = logprob_new
param_phase = param_phase_new
param_amp = param_amp_new
end
append!(chain, (param_phase, param_amp))
end
# append! makes "chain" a 1-d vector; reshape to a matrix
chain = reshape(chain, (2,Int64(length(chain)/2)))';
[39]:
100. * naccepts/nsteps
1 6 Julia 1.2.0 | Idle
Mode: Command
Ln 1, Col 1
mcmc.ipynb

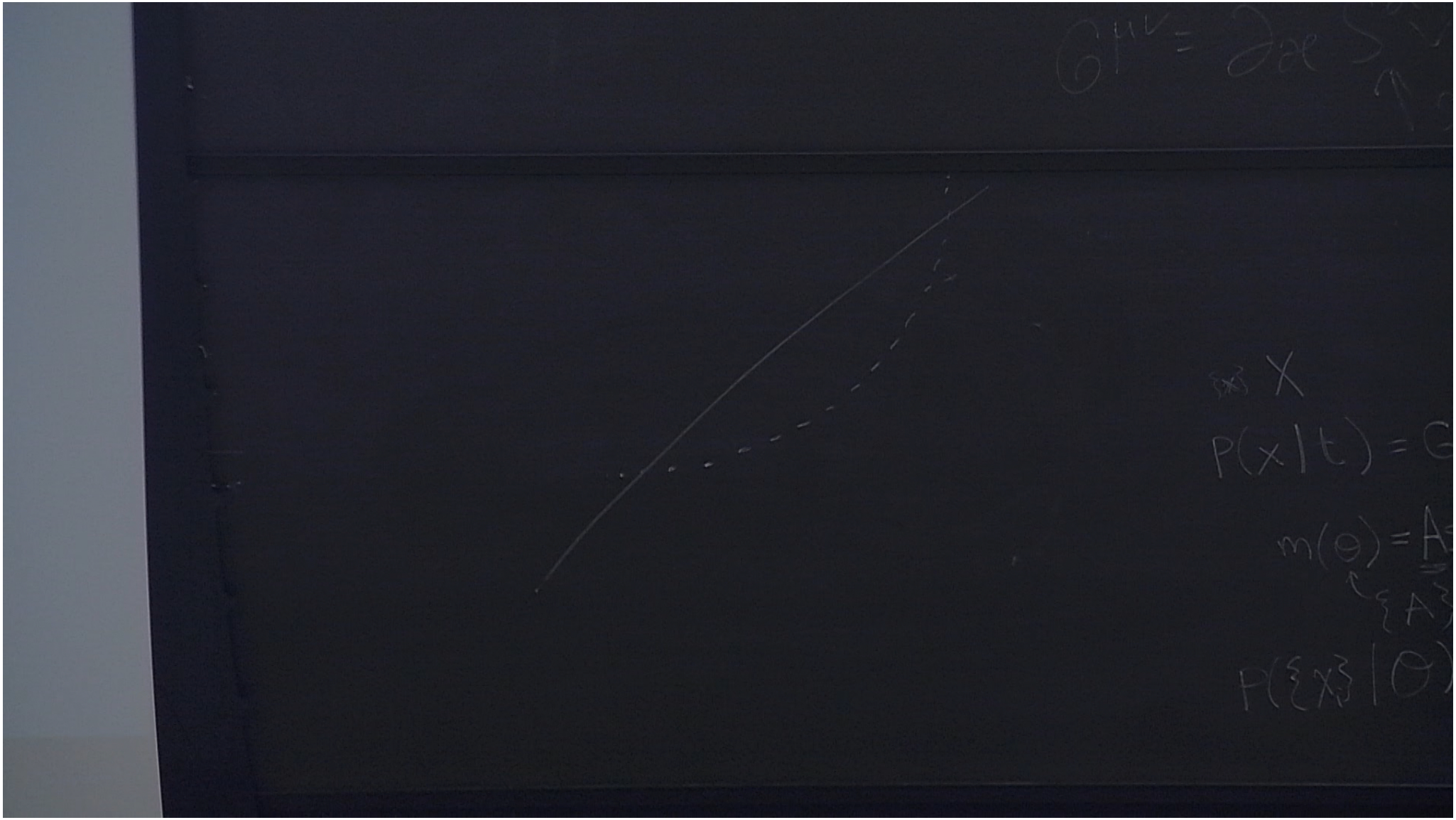
MCMC: summary

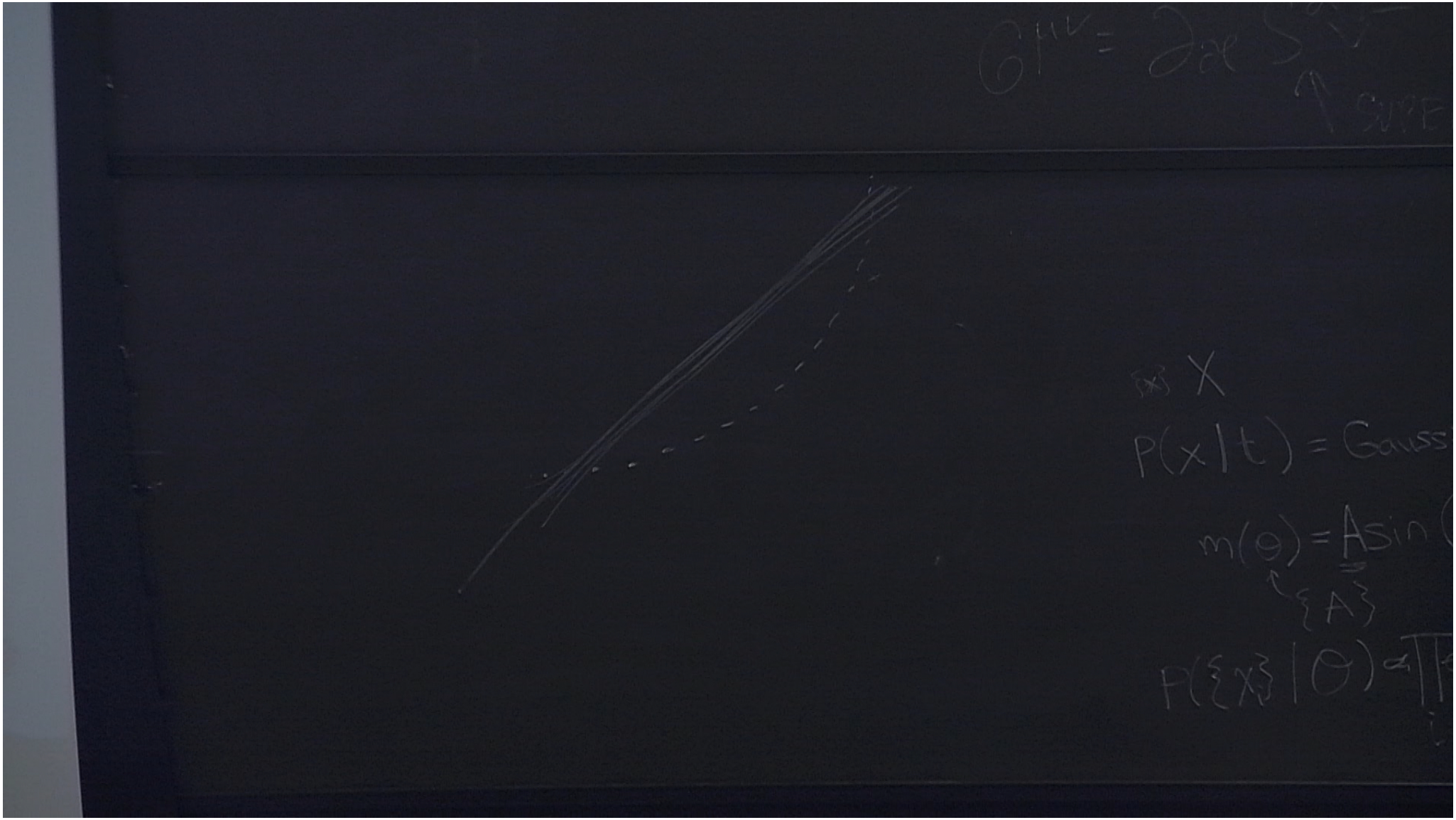
Things MCMC can do for you

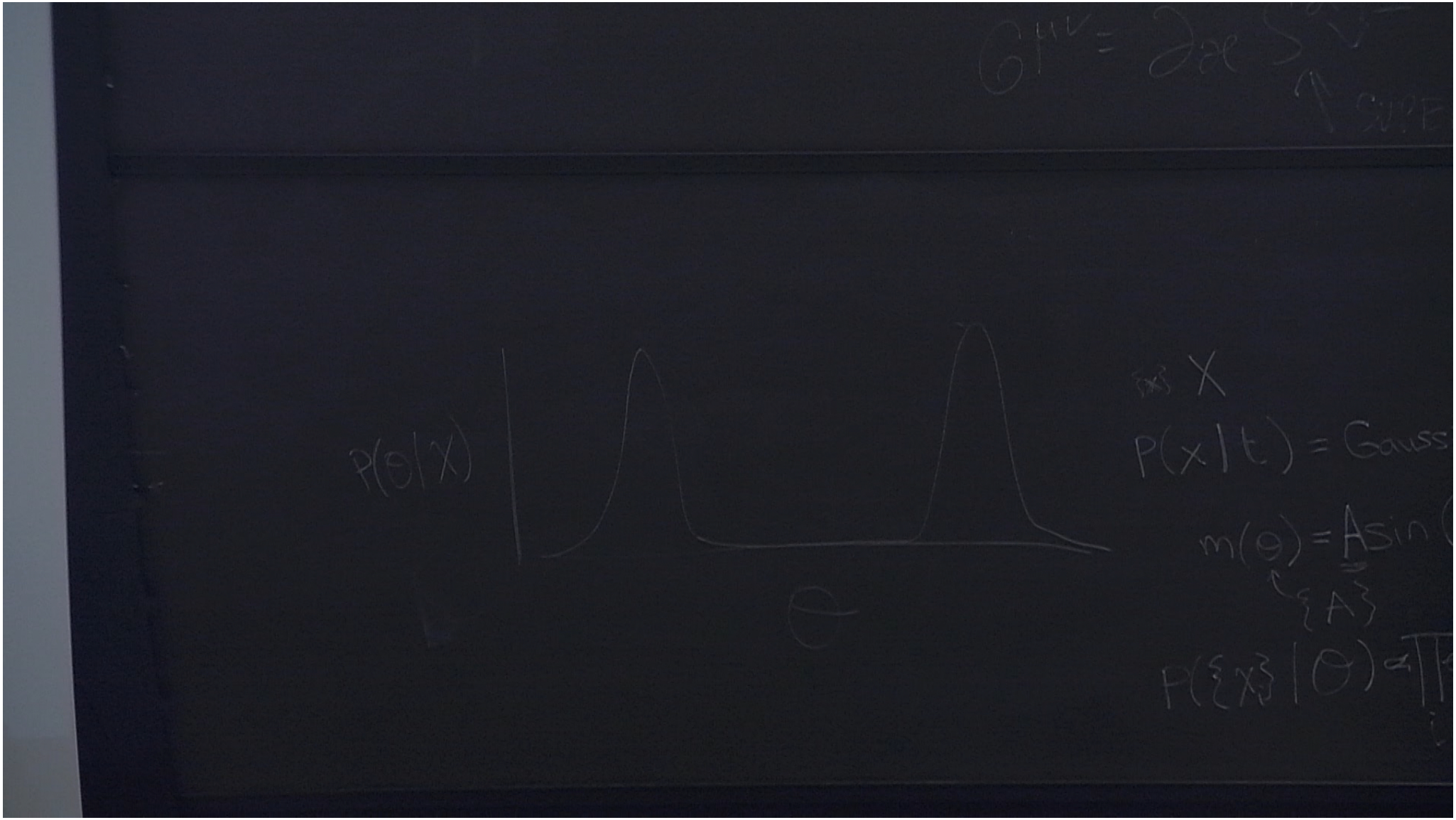
- produce constraints on model parameters, including covariances
- include (but ignore) *nuisance parameters* in your data analysis

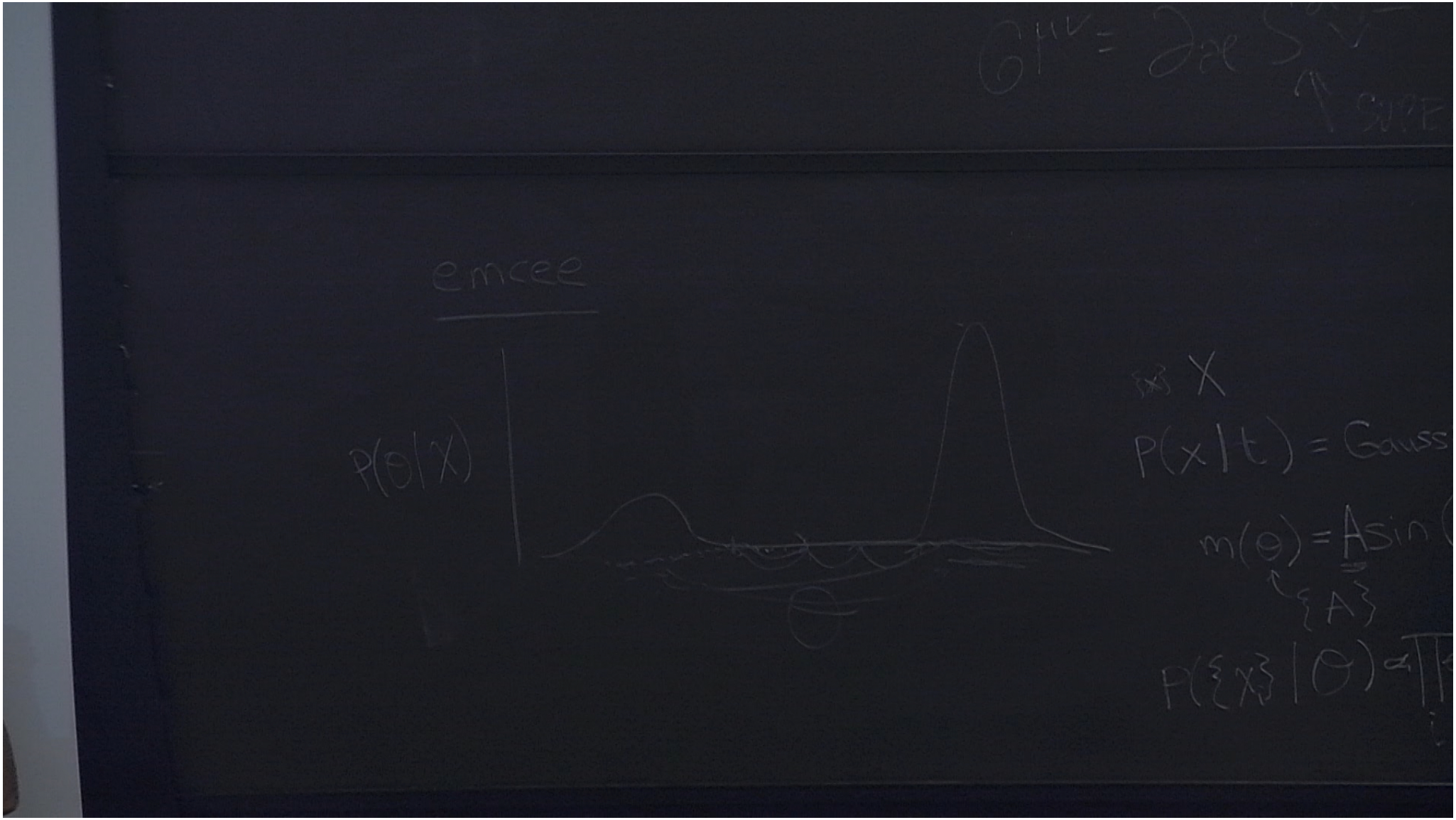
Things MCMC doesn't do for you

- tell you if it has converged!
- tell you if your model is good
- (easily) allow comparisons between models
- handle multi-modal distributions (try *nested sampling*)
- Dan Foreman–Mackey has good notes on advanced sampling methods


X
P(x|t) = G
m(θ) = A
P({x}|θ)


X
P(x|t) = Gauss
m(θ) = A sin
{A}
P({x}|θ) ∝


P(θ|x)
x
P(x|t) = Gauss
m(θ) = A sin
{A}
P({x}|θ) ∝


emcee
P(θ|X)
{x} X
P(x|t) = Gauss
m(θ) = A sin
{A}
P({x}|θ) ∝