Bayesian Statistics and Bayesian Cognitive Modeling (BayesCog)

Dr. Lei Zhang

Adaptive Learning Psychology & Neuroscience (ALPN) Lab
Centre for Human Brain Health, School of Psychology
University of Birmingham

lei-zhang.net
@lei_zhang_lz

UKE Hamburg, 10/2023

Schedule

cognitive model
statistics
computing

Oct. 2023 Statistical modeling

Introduction
R Basics
Probability Basics
Bayes' theorem
MCMC and Stan
Single-Parameter Model – Binomial Model
Multiple-Parameter Model – Linear Regression
Inference, Posterior Predictive Check

Dec. 2023 Cognitive modeling

Reinforcement Learning Model
Hierarchical Models
Optimizing Stan Codes
Model Comparison
Stan Style Tip and Debugging
Model-Based fMRI
Capstone Project: RLDDM modeling

Goal of this course

- Practical R programming

- Practical model-building in Stan, model diagnostics

- (Enough) theory to ground you

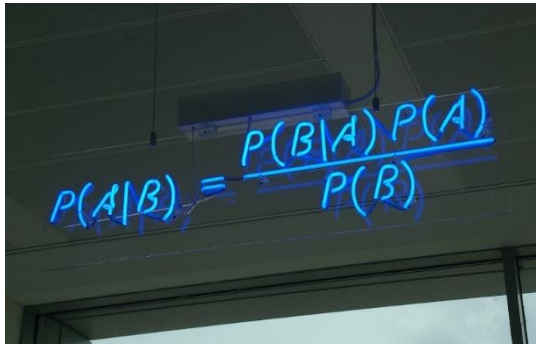
- Be comfortable to use R/Stan for your own work: model building/estimating/testing/validating, and/or model-based fMRI/EEG

Goal of this course

This course is NOT about…

… Bayes in the brain (e.g. predictive coding)

… Bayesian statistics to supersede classic statistics



However, Bayesian statistics offer great tools to analyze cognitive processes!

- Construct cognitive models
- Estimate posterior distributions of parameters
- Compare models: which is the best one, given the data
- Perform model-based analysis, e.g. model-based fMRI/EEG/eye-movement

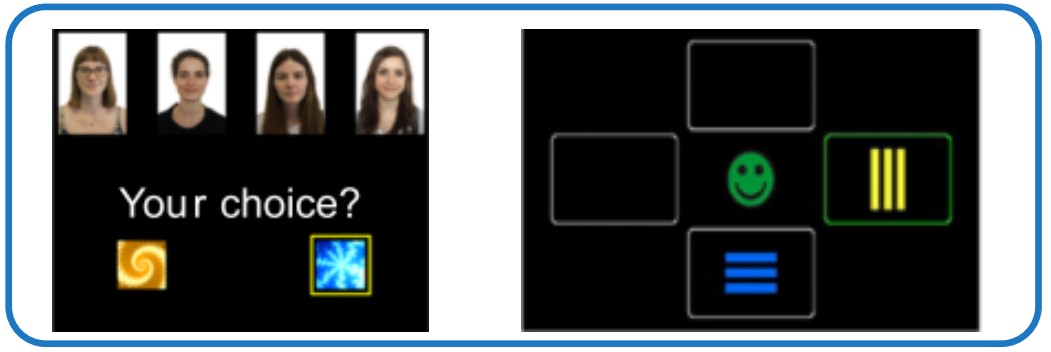
About me

- Current: Associate Professor
- Postdoc @ SCAN-Unit, w/ Claus Lamm
- Ph.D. Cognitive computational neuroscience, w/ Jan Gläscher
- M.Sc. Cognitive neuroscience
- B.Sc. Psychology

My research:

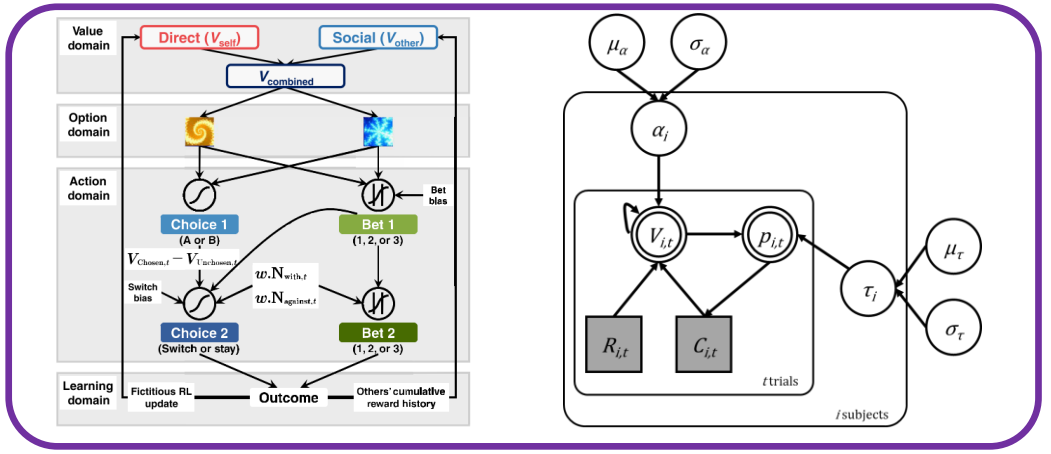
- I ask people to make decisions

Computation



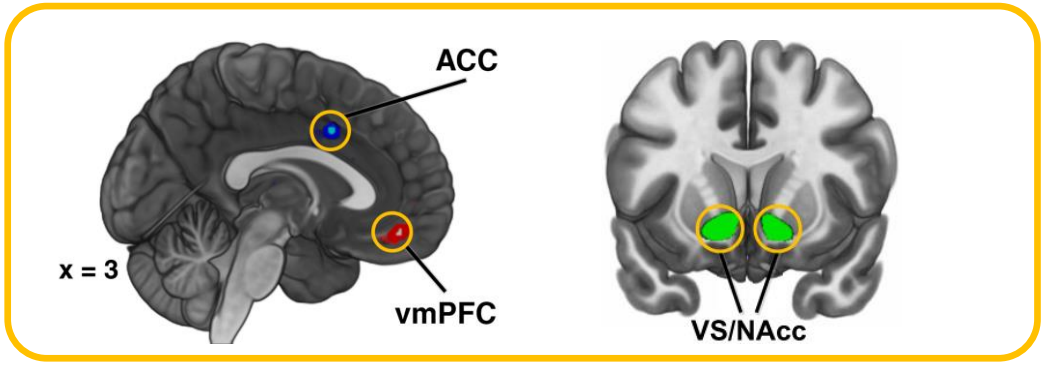
- I build computational models

Algorithm



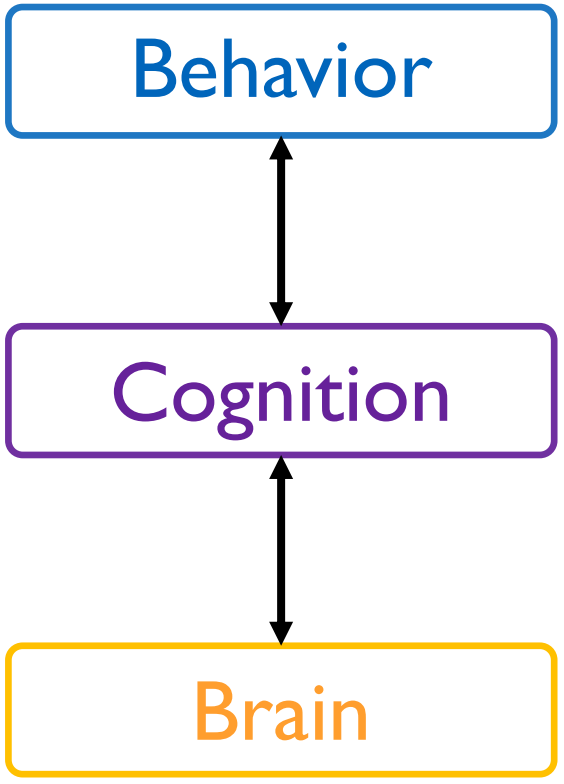
- I examine neural mechanisms

Implementation



Influential perspective: Marr's 3 levels of analysis

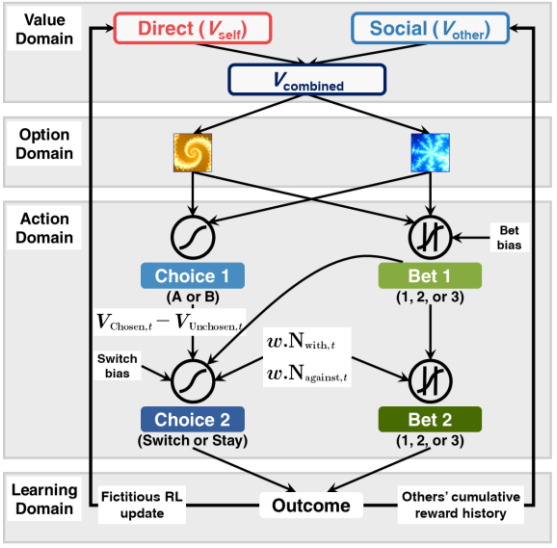
	LEVELS	
Computation	1	why (problem)
Algorithm	2	what (rules)
Implementation	3	how (physical)



David Marr, (1982)

My journey through computational modeling

- My journey through computational modeling
  - Started with MLE (@fminsearch in Matlab)
  - Switched to Bayesian: first JAGS, then Stan
  - Why switching to Stan? Look at Fig. 2A and Table S3 in [Zhang & Gläscher (2020)]



$$
\begin{aligned}
\mathbb{V}_{self,t} &= [V_{self,t}(A), V_{self,t}(B)] \\
\mathbb{V}_{other,t} &= [V_{other,t}(A), V_{other,t}(B)] \\
\mathbb{V}_t &= \beta_{vself}\,\mathbb{V}_{self,t} + \beta_{vother}\,\mathbb{V}_{other,t} \\
C1_t &\sim Categorical(Softmax(\mathbb{V}_t)) \\
U_{bet1,t} &= \beta_{bias,b} + \beta_{vdiff,b}(V_{chosen,C1,t} - V_{unchosen,C1,t}) \\
B1_t &\sim OrderedLogistic(U_{bet1,t} \mid \theta) \\
w.N_{against,t} &= \frac{\sum_{s=1}^{K} w_{s,t}}{\sum_{s=1}^{4} w_{s,t}},\ K = 0,1,...,4 \\
w.N_{with,t} &= \frac{\sum_{s=K}^{4} w_{s,t}}{\sum_{s=1}^{4} w_{s,t}} \\
V_t(\text{switch}) &= \beta_{bias,s} + \beta_{vdiff,s}(V_{chosen,C1,t} - V_{unchosen,C1,t}) + \beta_{against,s}\,w.N_{against,t} \\
C2 &\sim Bernoulli(V_t(\text{switch})) \\
U_{bet2,t} &= \begin{cases} U_{bet1,t} + \beta_{with,w}\,w.N_{with,t} + \beta_{against,w}\,w.N_{against,t}, & \text{if } C1 = C2 \\ U_{bet1,t} + \beta_{with,switch}\,w.N_{with,t} + \beta_{against,switch}\,w.N_{against,t}, & \text{if } C1 \neq C2 \end{cases} \\
B2_t &\sim OrderedLogistic(U_{bet2,t} \mid \theta) \\
\Phi(x) &= \frac{1}{1+e^{-x}} \\
\delta_{self,chosen,C2,t} &= R_{self,t} - V_{self,chosen,C2,t} \\
\delta_{self,unchosen,C2,t} &= -R_{self,t} - V_{self,unchosen,C2,t} \\
V_{self,chosen,C2,t+1} &= V_{self,chosen,C2,t} + \alpha\delta_{self,chosen,C2,t} \\
V_{self,unchosen,C2,t+1} &= V_{self,unchosen,C2,t} + \alpha\delta_{self,unchosen,C2,t}
\end{aligned}
$$

How about you?

Where are you from?

What do you work on?

How much programming have you done?

How much modeling have you done?

What motivated you to be here?

- gain knowledge of Bayesian stats?
- be able to read "computational modeling" section in papers?
- write your own model?

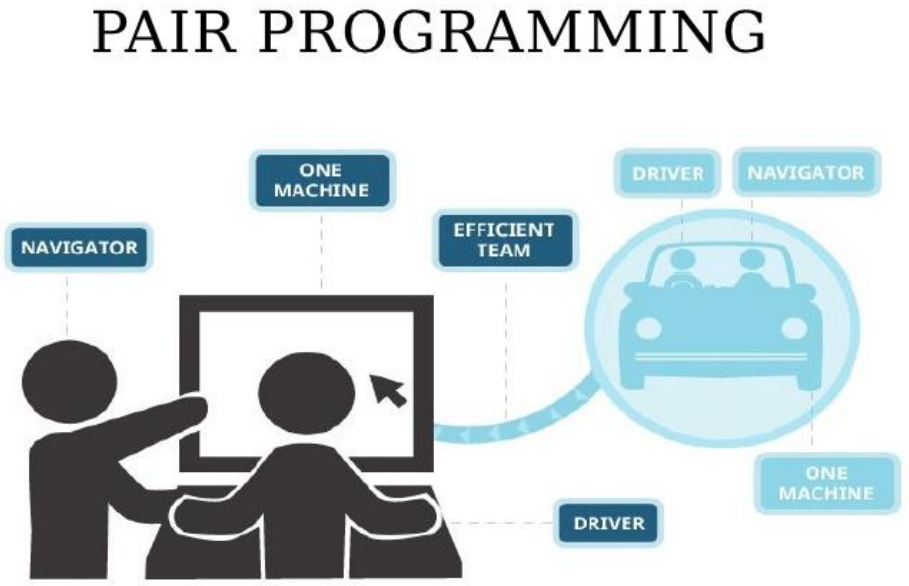
Richard McElreath
@rlmcelreath

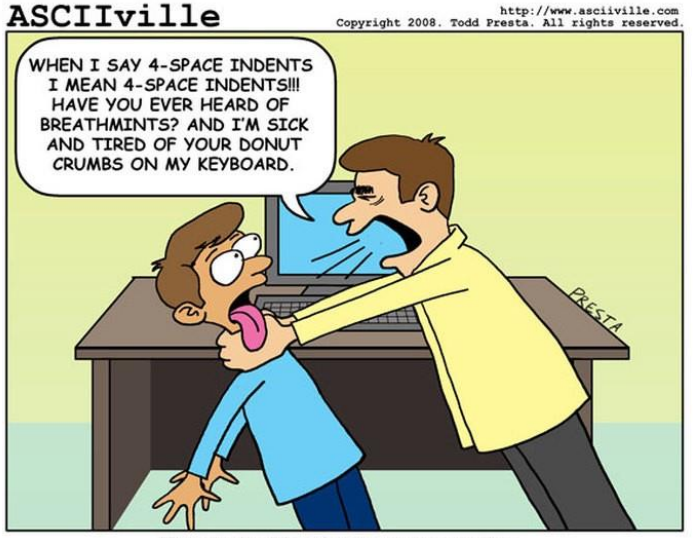
I say this a lot, bc I am also confused quite often.

> 
> **Anna Jacobson** @AnnaChingChing · Feb 21
>
> "If you are confused, it is only because you are trying to understand." - @rlmcelreath in Statistical Rethinking

How to Get the Most out of the course

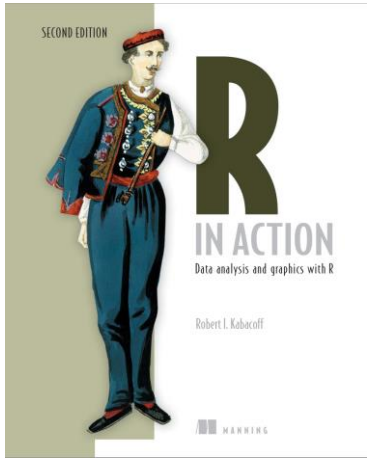
- Workshop structure: interleaved theory/demo + exercise
- Work in pairs: Talk to each other & help each other
- Ask questions
- Try the exercises

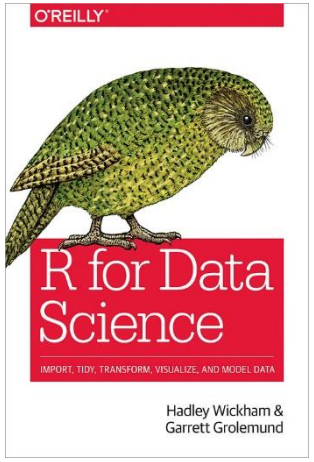


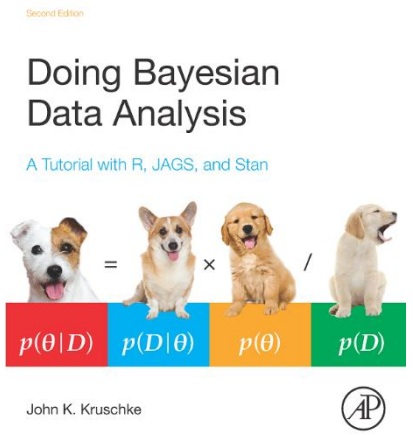


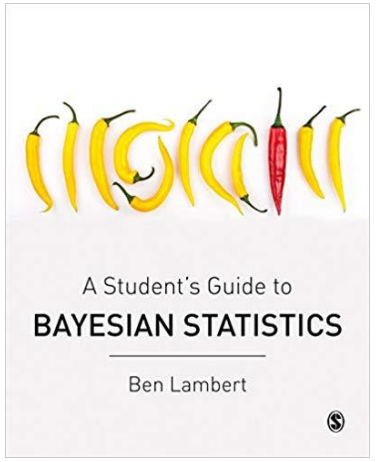
The dark side of pair programming.

Resources









Statistical Thinking for the 21st Century

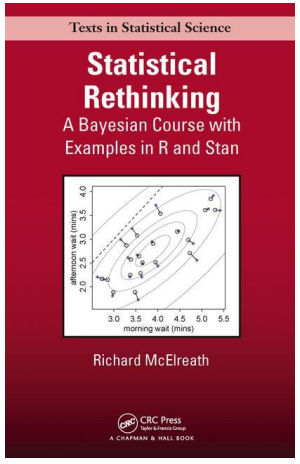
Copyright 2019 Russell A. Poldrack

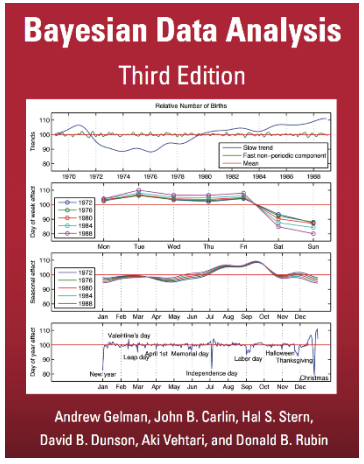
Draft: 2020-03-15

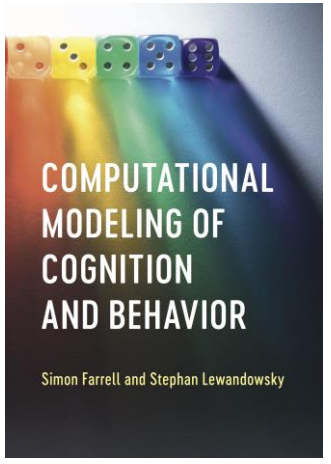
http://statsthinking21.org/

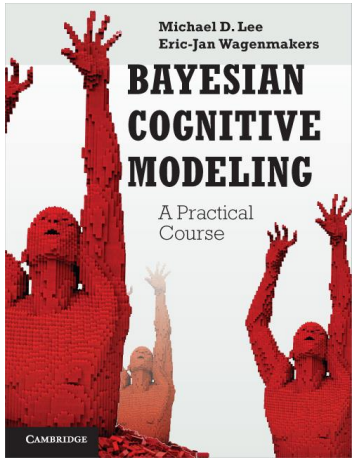
datacamp

https://www.datacamp.com/









JASP

https://jasp-stats.org/

Resources

Ten simple rules for the computational modeling of behavioral data

Robert C Wilson[1,2†*], Anne GE Collins[3,4†*]

https://elifesciences.org/articles/49547

Computational modelling of social cognition and behaviour—a reinforcement learning primer

Patricia L Lockwood ✉, Miriam C Klein-Flügge

Social Cognitive and Affective Neuroscience, nsaa040, https://doi.org/10.1093/scan/nsaa040

https://doi.org/10.1093/scan/nsaa040

Trial-by-trial data analysis using computational models

Nathaniel D. Daw

August 27, 2009

http://www.princeton.edu/~ndaw/d10.pdf

Using reinforcement learning models in social neuroscience: frameworks, pitfalls and suggestions of best practices

Lei Zhang ✉, Lukas Lengersdorff, Nace Mikus, Jan Gläscher, Claus Lamm Author Notes

Social Cognitive and Affective Neuroscience, Volume 15, Issue 6, June 2020, Pages 695–707, https://doi.org/10.1093/scan/nsaa089

https://doi.org/10.1093/scan/nsaa089

Now let's begin!

BASICS OF R PROGRAMMING

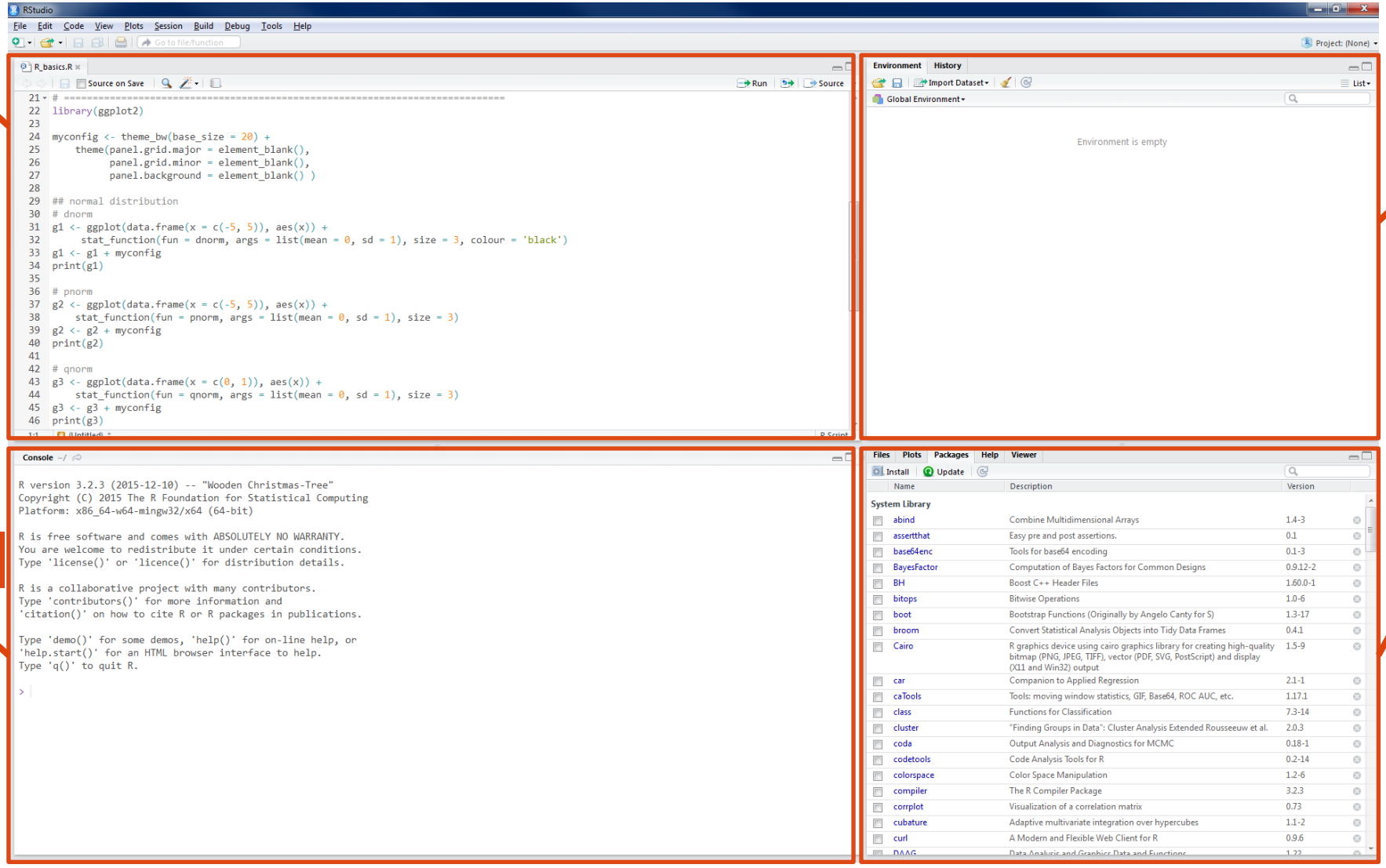
RStudio®

R Basics

- R
  - a programming language for statistical computing
  - R has its own user interface
  - freely available on Windows, Mac, and Linux

- R Studio
  - integrated development environment (IDE) for R
  - a more sophisticated R-friendly editor, with helpful syntax highlight

script editor



console

environment/
command history

file/pkg/img/
etc.

Know your R

```
>R.version

               _
platform       x86_64-w64-mingw32
arch           x86_64
os             mingw32
system         x86_64, mingw32
status
major          3
minor          5.1
year           2018
month          07
day            02
svn rev        74947
language       R
version.string R version 3.5.1 (2018-07-02)
nickname       Feather Spray
```

R Console as a Calculator

cognitive model
statistics
computing

Addition and Subtraction

```
> 3+2
[1] 5

> 3-2
[1] 1
```

Multiplication and Division

```
> 3*2
[1] 6

> 3/2
[1] 1.5
```

Exponents in R

```
> 3^2
[1] 9

> 2^3
[1] 8
```

Constants in R

```
> pi
[1] 3.141593

> exp(1)
[1] 2.718282
```

base of the natural logarithm

Special values

cognitive model
statistics
computing

Infinite Values

```
> Inf
[1] Inf

> 1+Inf
[1] Inf
```

Empty Values

```
> NULL
NULL

> 1+NULL
numeric(0)
```

Machine Epsilon

```
> .Machine$double.eps
[1] 2.220446e-16

> 0>.Machine$double.eps
[1] FALSE
```

Missing Values

```
> NA
[1] NA

> 1+NA
[1] NA
```

Storing and manipulating variables

cognitive model
statistics
computing

Define objects `x` and `y` with values of 3 and 2, respectively:

```
> x=3
> y=2
```

Some calculations with the defined objects `x` and `y`:

```
> x+y
[1] 5

> x*y
[1] 6
```

Warning: R is case sensitve, so `x` and `X` are not the same object.

Basic R functions

cognitive model
statistics
computing

Combine

```
> c(1,3,-2)
[1]  1  3 -2
```

```
> c("a","a","b","b","a")
[1] "a" "a" "b" "b" "a"
```

Variance and Std. Dev.

```
> var(c(1,3,-2))
[1] 6.333333
```

```
> sd(c(1,3,-2))
[1] 2.516611
```

Sum and Mean

```
> sum(c(1,3,-2))
[1] 2
```

```
> mean(c(1,3,-2))
[1] 0.6666667
```

Minimum and Maximum

```
> min(c(1,3,-2))
[1] -2
```

```
> max(c(1,3,-2))
[1] 3
```

Basic R functions (cont.)

cognitive model
statistics
computing

Define objects `x` and `y`:

```
> x=c(1,3,4,6,8)
> y=c(2,3,5,7,9)
```

Calculate the correlation:

```
> cor(x,y)
[1] 0.988765
```

Calculate the covariance:

```
> cov(x,y)
[1] 7.65
```

Combine as columns

```
> cbind(x,y)
     x y
[1,] 1 2
[2,] 3 3
[3,] 4 5
[4,] 6 7
[5,] 8 9
```

Combine as rows

```
> rbind(x,y)
  [,1] [,2] [,3] [,4] [,5]
x    1    3    4    6    8
y    2    3    5    7    9
```

Basic Commands

```
getwd()
setwd('E:/teaching/BayesCog_Wien/')
dir() # folders/files in the wd
ls()  # anything in the environment/workspace
print('Hello World!')
cat('Hello', 'World!')
paste0('C:/', 'Group1')
help(func)
? func # and Google!
a <- 5
a = 5
head(d) # first 6 entries
tail(d) # last 6 entries
save(varname, file = "pathname/varname.RData")
load("pathname/varname.RData")
rm(list = ls())
q()
```

RStudio - Shortcuts

cognitive model
statistics
computing

Ctrl + L: clean console

Ctrl + Shift + N: create a new script

↑: command history

Ctrl(hold) + ↑: command history with certain starts

Ctrl + Enter: execute selected codes (in a script)

Editor (WIN general) - Shortcuts

cognitive model
statistics
computing

Ctrl + home/Pos: go to the very top of a script

Ctrl + end/Ende: go to the very end of a script

Shift(hold) + ↑/↓: select line(s)

Ctrl(hold) + ←/→: select word(s)

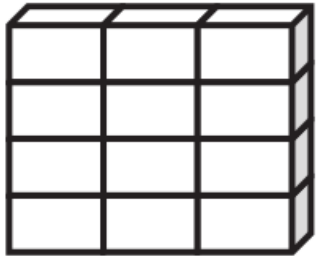
Data Classes


```
numeric: 1.1 2.0
integer: 1 2 3
character / string: "hello world!"
logical: TRUE FALSE
factors: "male" / "female"
(complex: 1+2i)
```

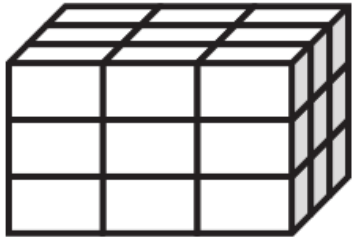
Data Types

(a) Vector

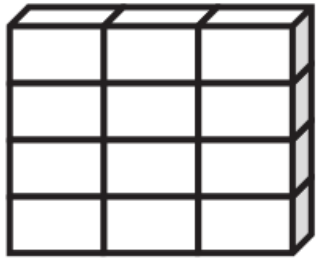
(b) Matrix



(c) Array



(d) Data frame



Columns can be different modes

(e) List

- Vectors
- Arrays
- Data frames
- Lists

Kabacoff (2015)

Exercise I

cognitive model
statistics
computing

```
…/01.R_basics/_scripts/R_basics.R

up to “Control Flow”
```

TASK: practise basic R commands and data type

TIP: `class()`, `str()`

Side note: folder structure

click this to start each exercise,
then no need to set directory

Logical Operators

Operator	Summary
$<$	Less than
$>$	Greater than
$<=$	Less than or equal to
$>=$	Greater than or equal to
$==$	Equal to
$!=$	Not equal to
$!x$	NOT x
$x \mid y$	x OR y
$x \& y$	x AND y

Control Flow

cognitive model
statistics
computing

- if-else

```
if (cond) {
    ..statement..
}
```

```
if (cond) {
    ..statement..
} else {
    ..statement..
}
```

```
if (cond) {
    ..statement..
} else if (cond) {
    ..statement..
} else {
    ..statement..
}
```

- for-loop

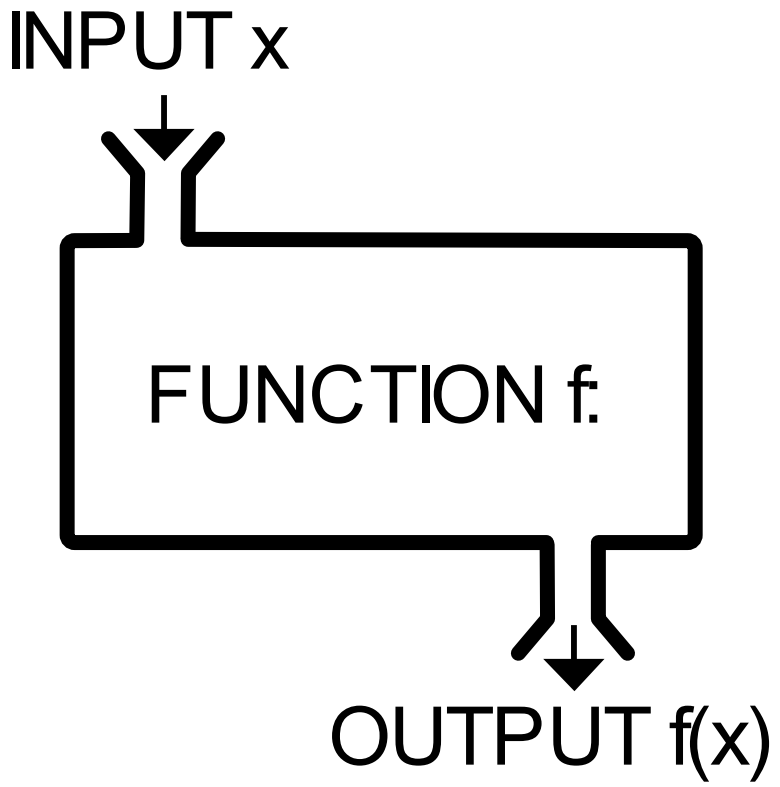
```
for ( j in 1:J) {
    ..statement..
}
```

```
for ( j in 1:J ) {
    for ( k in 1:K ) {
        ..statement..
    }
}
```

Functions

The operation(s) to obtain some quantity, based on another quantity.

- built-in functions
- external functions (packages)
- user-defined functions



User-defined Function

cognitive model
statistics
computing

```
funname <- function (input_arges) {
    .. function body ..
    .. function body ..
    return(output_arges)
}
```

$$sem = \sqrt{\frac{s^2}{n-1}}$$

```
sem <- function(x) {
    sqrt( var(x,na.rm=TRUE) / (length(na.omit(x))-1) )
}
```

Exercise II

cognitive model
statistics
computing

```
…/01.R_basics/_scripts/R_basics.R
```

TASK: practise control flow and user-defined function

Exercise II

cognitive model
statistics
computing

- Generate a random number between 0 and 1
- Compare it against 1/3 and 2/3
- Print the random number and its position relative to 1/3 and 2/3.

```r
# if-else
t <- runif(1) # random number between 0 and 1
if (t <= 1/3) {
    cat("t =", , ", t <= 1/3. \n")
} else if () {
    cat("t =", t, ", t > 2/3. \n")
} else {
    cat("t =", t, ", 1/3 < t <= 2/3. \n")
}
```

Example outcome:

```
t = 0.895 , t > 2/3.
```

- Get the name of each month
- Print it one by one

```r
# for-loop
month_name <- format(ISOdate(2018,1:12,1),"%B")
for (j in 1:length(month_name) ) {
    cat()
}
```

```
The month is January
The month is February
The month is March
The month is April
The month is May
The month is June
The month is July
The month is August
The month is September
The month is October
The month is November
The month is December
```

Packages in R

R packages are collections of functions and data sets developed by the community, to make your life a lot easier!

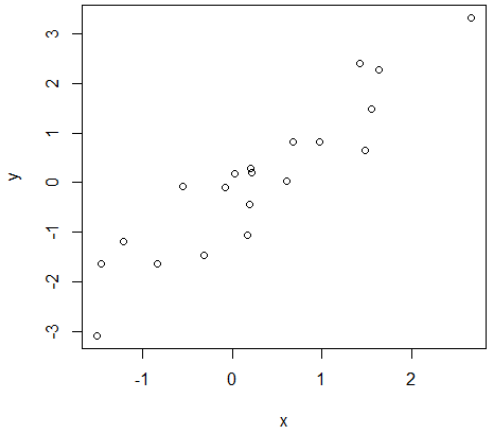
```
install.packages('ggplot2')
library(ggplot2)
detach('package:ggplot2')
```

Visualization

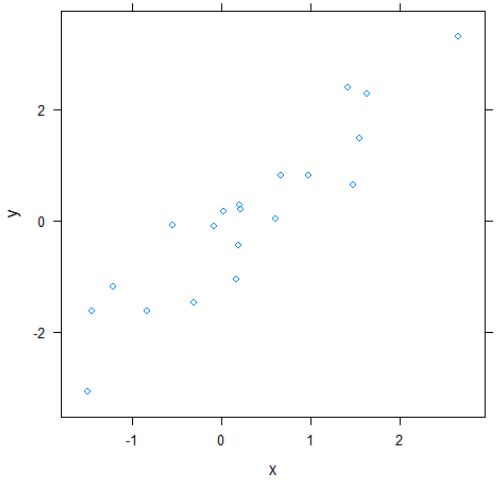
cognitive model
statistics
computing

- built-in plotting functions – first attempt / quick look / exploratory
- {lattice} – making nicer, similar to basic plotting functions (takes lm formulae)
- {ggplot2} – making nicer, a layering philosophy

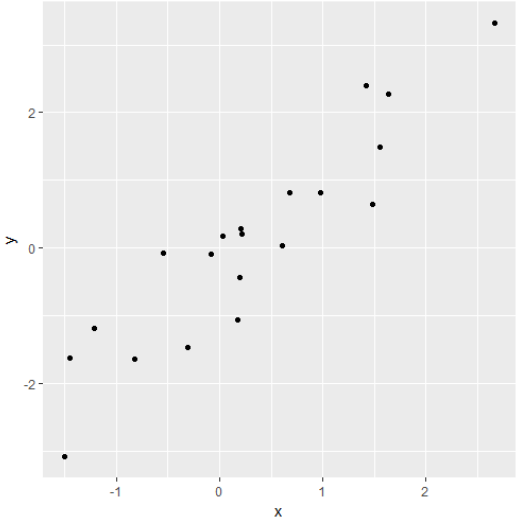
`plot(x,y)`



`lattice::xyplot(y~x)`



`ggplot2::qplot(x,y)`



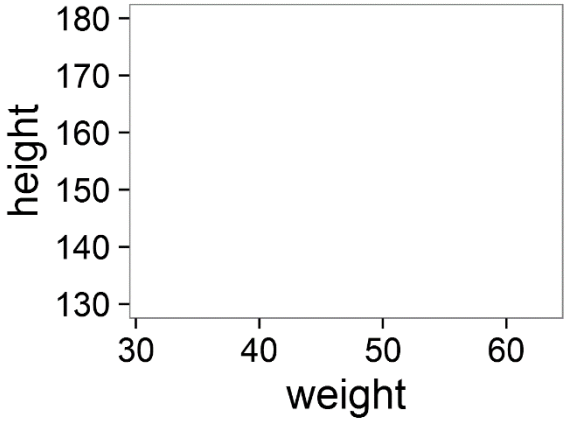
Brief Intro to ggplot2

cognitive model
statistics
computing

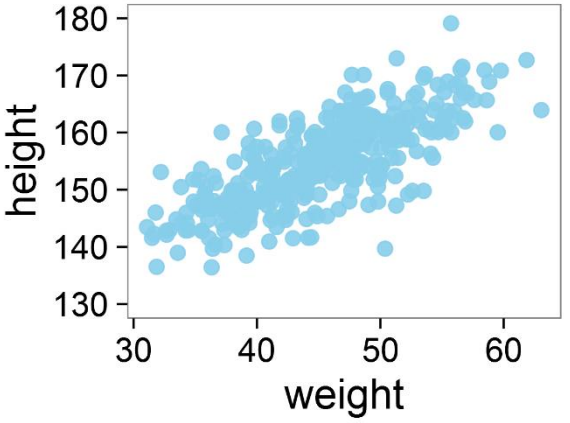
```
plot = geometric (points, lines, bars) + aesthetic (color, shape, size)
```

game of adding layers!

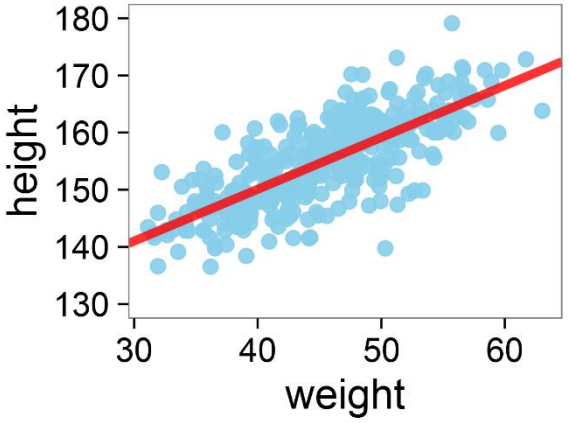
background



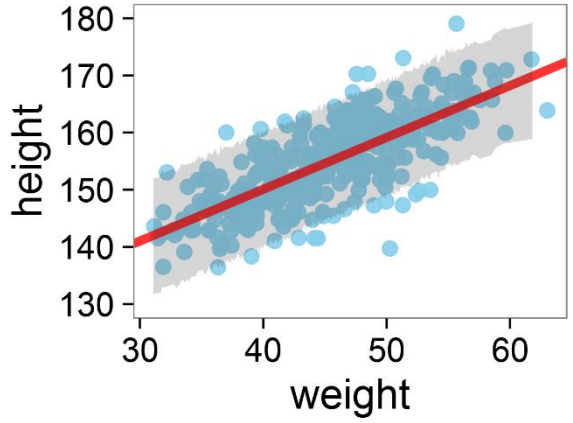
add scatters



add regression line

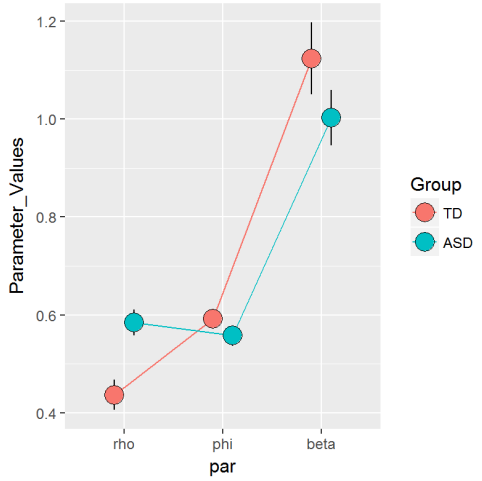


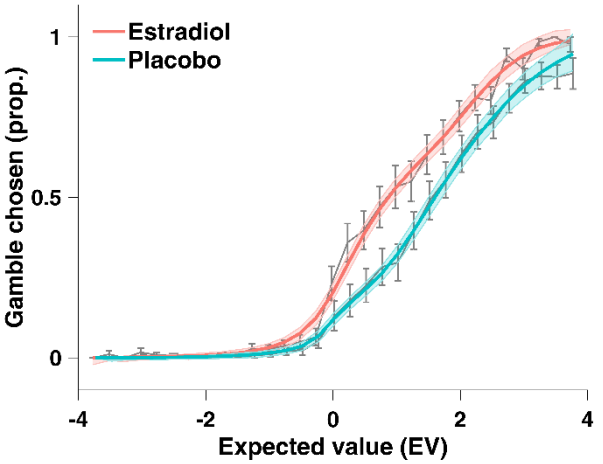
add uncertainty

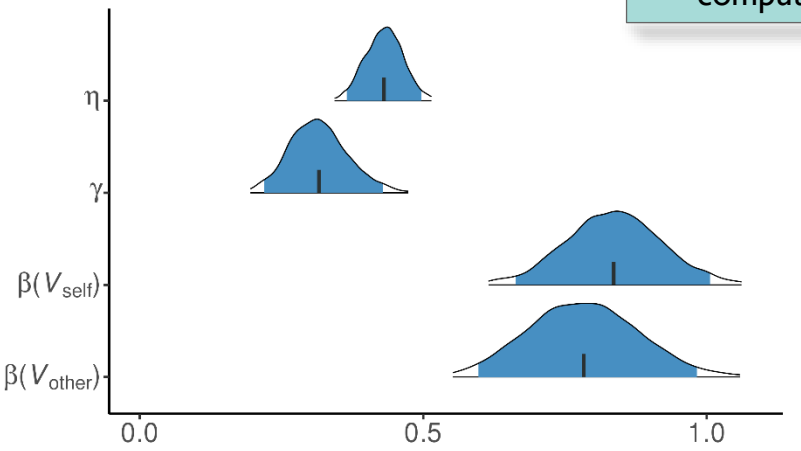


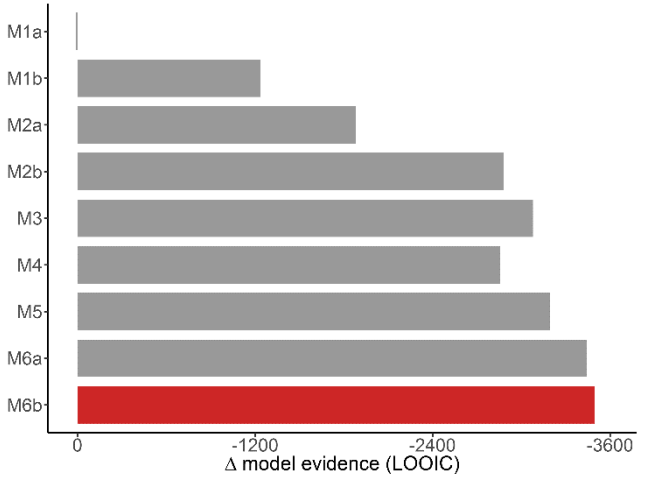
A taste of ggplot2

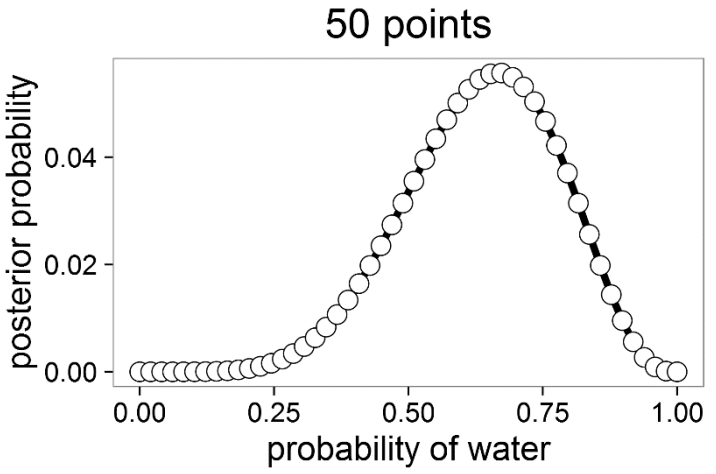
cognitive model
statistics
computing

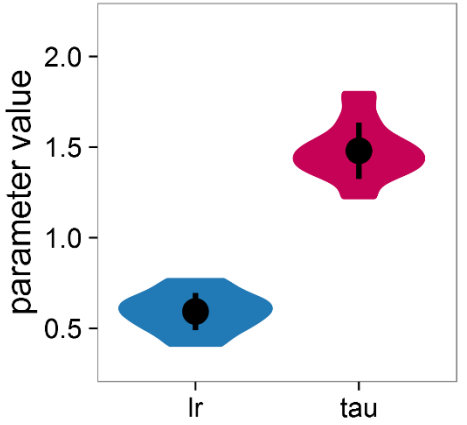






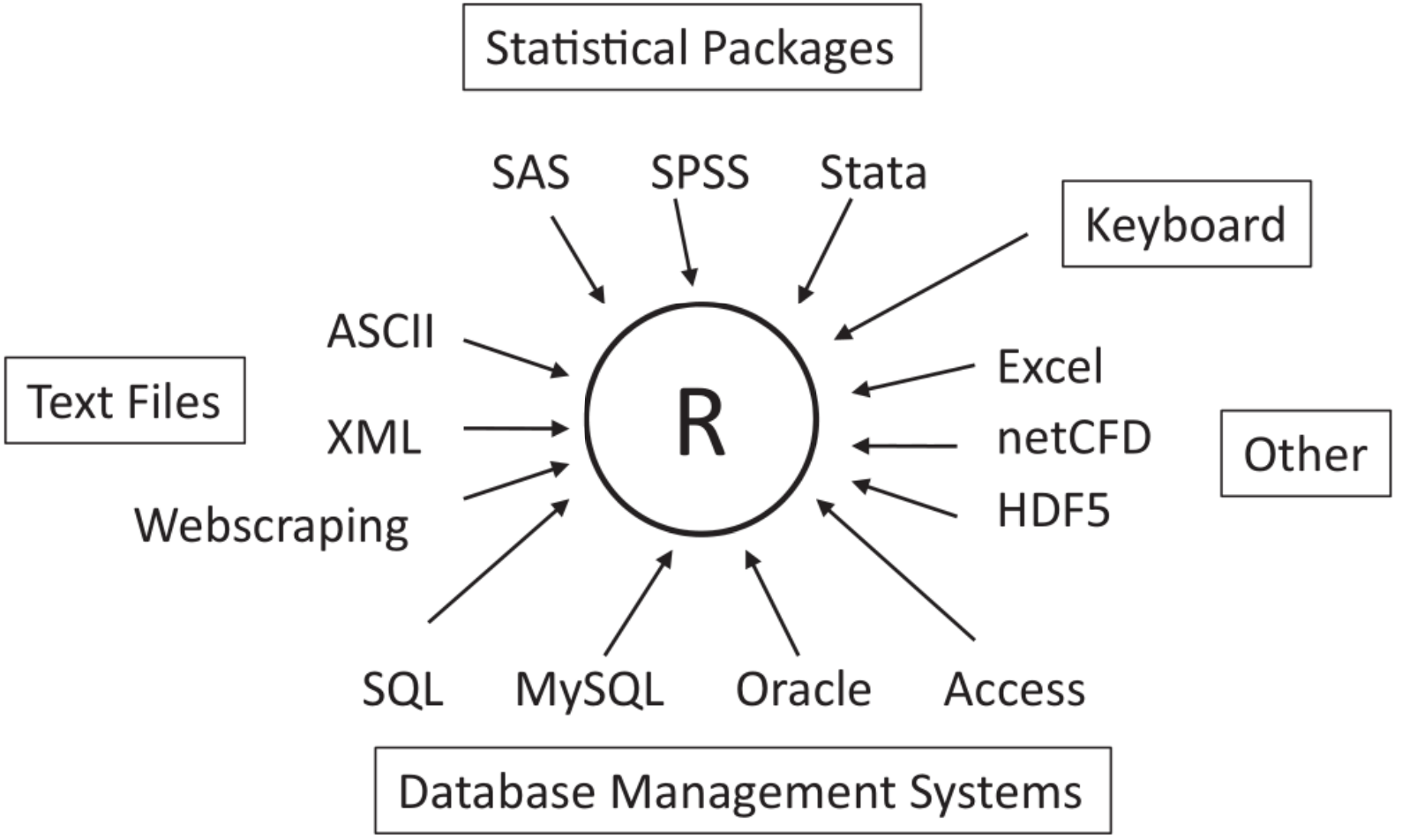






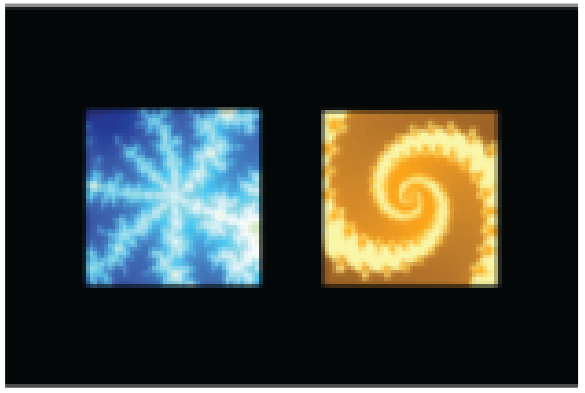
https://www.r-graph-gallery.com/

Data management

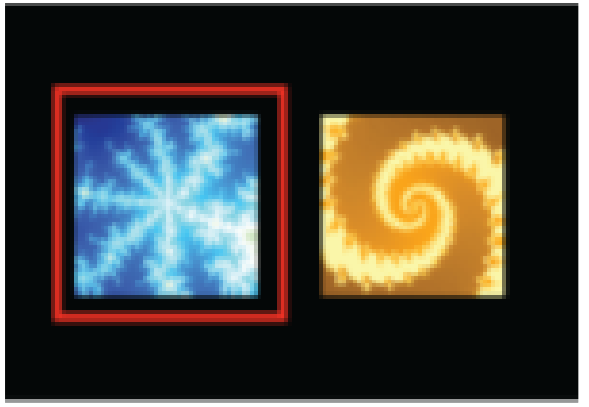


Kabacoff (2015)

One simple experiment



choice
presentation



action
selection

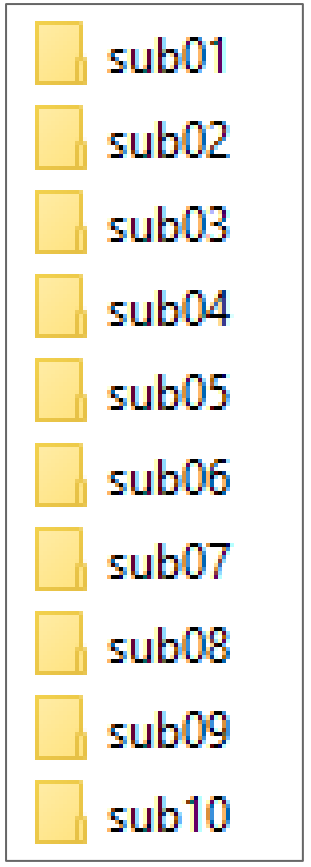
outcome

reward contingency – 80:20

The data

- nSub = 10
- nTrial = 80

./_data/_raw_data/sub01/raw_data_sub01.txt




```
subjID, trialID, choice, outcome, correct
1,1,2,-1,1
1,2,1,1,1
1,3,1,1,1
1,4,1,1,1
1,5,2,-1,1
1,6,1,1,1
1,7,1,1,1
1,8,1,1,1
1,9,1,-1,1
1,10,2,-1,1
1,11,1,1,1
1,12,1,1,1
1,13,1,-1,2
```

Import some data!

```
data_dir = ('_data/RL_raw_data/sub01/raw_data_sub01.txt')
data = read.table(data_dir, header = T, sep = ",")
head(data)
```

```
  subjID trialID choice outcome correct
1      1       1      1       1       1
2      1       2      1       1       1
3      1       3      1       1       1
4      1       4     NA       1       1
5      1       5      1      -1       1
6      1       6      2      -1       1
```

Indexing

```
data[1,1]
data[1,]
data[,1]
data[1:10,]
data[,1:2]
data[1:10, 1:2]
data[c(1,3,5,6), c(2,4)]

data$choice
```

```
> data
   subjID trialID choice outcome correct
1       1       1      1       1       1
2       1       2      1       1       1
3       1       3      1       1       1
5       1       5      1      -1       1
6       1       6      2      -1       1
7       1       7      1       1       1
8       1       8      1       1       1
9       1       9      1       1       1
10      1      10      1       1       1
11      1      11      1       1       1
```

Import some data!

```
data_dir = ('_data/RL_raw_data/sub01/raw_data_sub01.txt')
data = read.table(data_dir, header = T, sep = ",")
head(data)
```

```
  subjID trialID choice outcome correct
1      1       1      1       1       1
2      1       2      1       1       1
3      1       3      1       1       1
4      1       4     NA       1       1
5      1       5      1      -1       1
6      1       6      2      -1       1
```

```
sum(complete.cases(data))   # number of valid trials
data = data[complete.cases(data),]
dim(data[complete.cases(data),])
```

Exercise III

`…/01.R_basics/_scripts/R_basics.R`

TASK:

write a for loop

… which reads in each participant's raw data

… and reshape it in the "long format" by subj

TIP: complete line 173; consider `sprintf()`

```
for ( j in 1:n ) {
   read.table(file, header = T, sep = ",")
}
```

subID	Choice
sub01	1
sub01	2
…	
sub02	2
sub02	2
…	
sub10	2
sub10	1

Read all the data!

```
ns = 10
data_dir = '_data/RL_raw_data'

rawdata = c();
for (s in 1:ns) {
    sub_file = file.path(data_dir, sprintf('sub%02i/raw_data_sub%02i.txt',s,s))
    sub_data = read.table(sub_file, header = T, sep = ",")
    rawdata = rbind(rawdata, sub_data)
}
rawdata = rawdata[complete.cases(rawdata),]
rawdata$accuracy = (rawdata$choice == rawdata$correct) * 1.0

acc_mean = aggregate(rawdata$accuracy, by = list(rawdata$subjID), mean)[,2]
```

mean choice accuracy across trials, per participant.

Basic stats

```
> as.matrix(acc_mean, 10, 1)
          [,1]
 [1,] 0.8076923
 [2,] 0.7125000
 [3,] 0.6875000
 [4,] 0.6493506
 [5,] 0.7750000
 [6,] 0.7250000
 [7,] 0.7662338
 [8,] 0.8000000
 [9,] 0.7500000
[10,] 0.6750000
```

```
mean(acc_mean)
sd(acc_mean)
sem(acc_mean)

t.test(acc_mean, mu = 0.5) # one sample t-test

        One Sample t-test

data:  acc_mean
t = 13.788, df = 9, p-value = 2.34e-07
alternative hypothesis: true mean is not equal to 0.5
95 percent confidence interval:
 0.6962988 0.7733565
sample estimates:
mean of x
0.7348277
```

Basic correlation

```
load('_data/RL_descriptive.RData')
descriptive$acc = acc_mean
df = descriptive

cor.test(df$IQ, df$acc)

        Pearson's product-moment correlation

data:  df$IQ and df$acc
t = 4.8347, df = 8, p-value = 0.001297
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 0.5114810 0.9671586
sample estimates:
      cor
0.8631401
```

```
> descriptive
   subjID        IQ      Age    acc
1       1 123.98691 31.07218 0.8125
2       2  87.63187 30.13800 0.7125
3       3  89.39930 23.44219 0.6875
4       4  84.34607 27.44848 0.6500
5       5 134.72208 23.30624 0.7750
6       6  84.60797 25.67858 0.7250
7       7 111.10238 24.36375 0.7750
8       8 117.89599 32.74026 0.8000
9       9  96.88233 22.80211 0.7500
10     10  76.01652 30.44258 0.6750
```

Exercise IV

```
…/01.R_basics/_scripts/R_basics.R
```

TASK:

Read in the descriptive data: _data/descriptive.RData

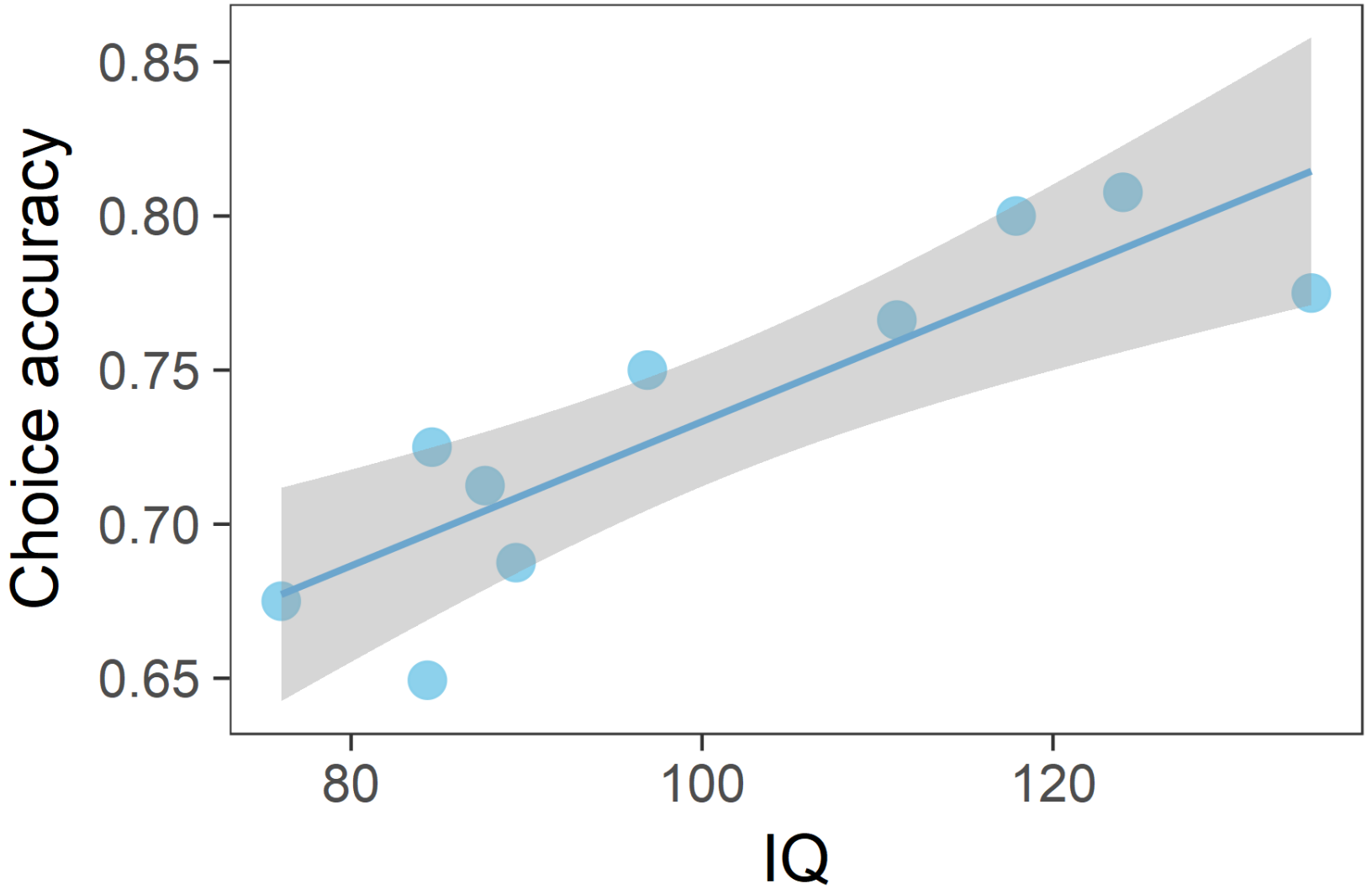
…include ‘acc_mean’ as a new column, and

…rename ‘descriptive’ as df.

Practice all the basic stats.

```
df$new_Col = new_Col
```

A simple linear regression



What is exactly the regression line in R?

$$\mu_i = \alpha + \beta x_i$$

$$y_i = \mu_i + \varepsilon$$

```
fit1 = lm(acc ~ IQ, data = df)
summary(fit1)

Call:
lm(formula = acc ~ IQ, data = df)

Residuals:
      Min        1Q    Median        3Q       Max
-0.047305 -0.016277  0.007562  0.022577  0.027731

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 0.499292   0.049565  10.073 8.04e-06 ***
IQ          0.002340   0.000484   4.835   0.0013 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.02885 on 8 degrees of freedom
Multiple R-squared:  0.745,  Adjusted R-squared:  0.7131
F-statistic: 23.37 on 1 and 8 DF,  p-value: 0.001297
```

Exercise V

```
…/01.R_basics/_scripts/R_basics.R
```

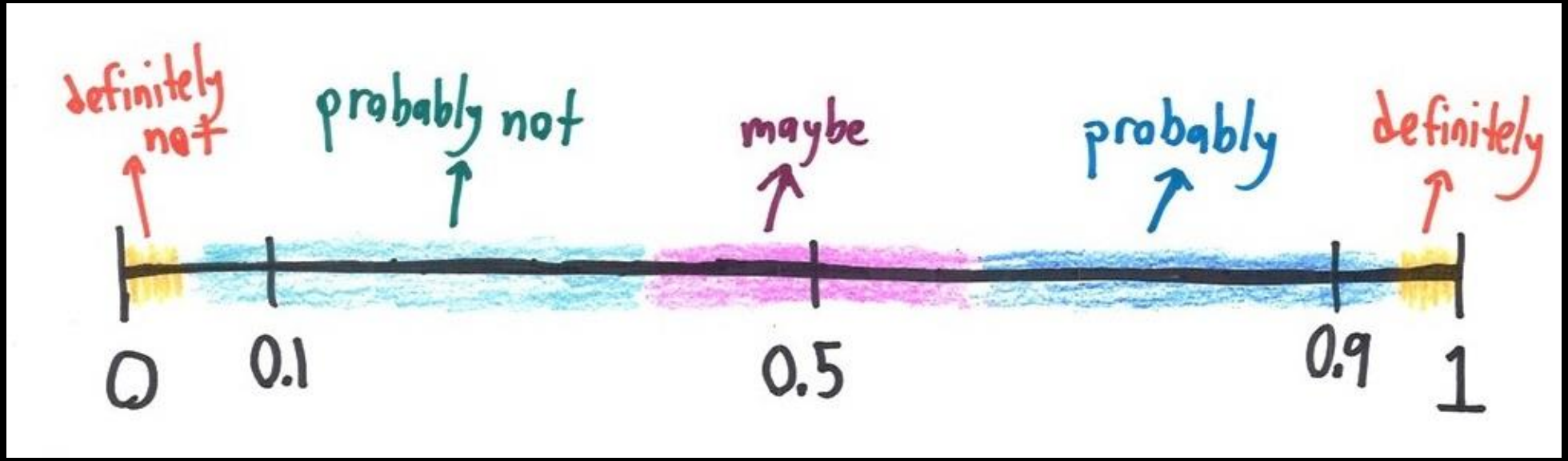
TASK:

Read and make sense of the ggplot functions,

… experiment make some adjustments (color marker size etc.), and

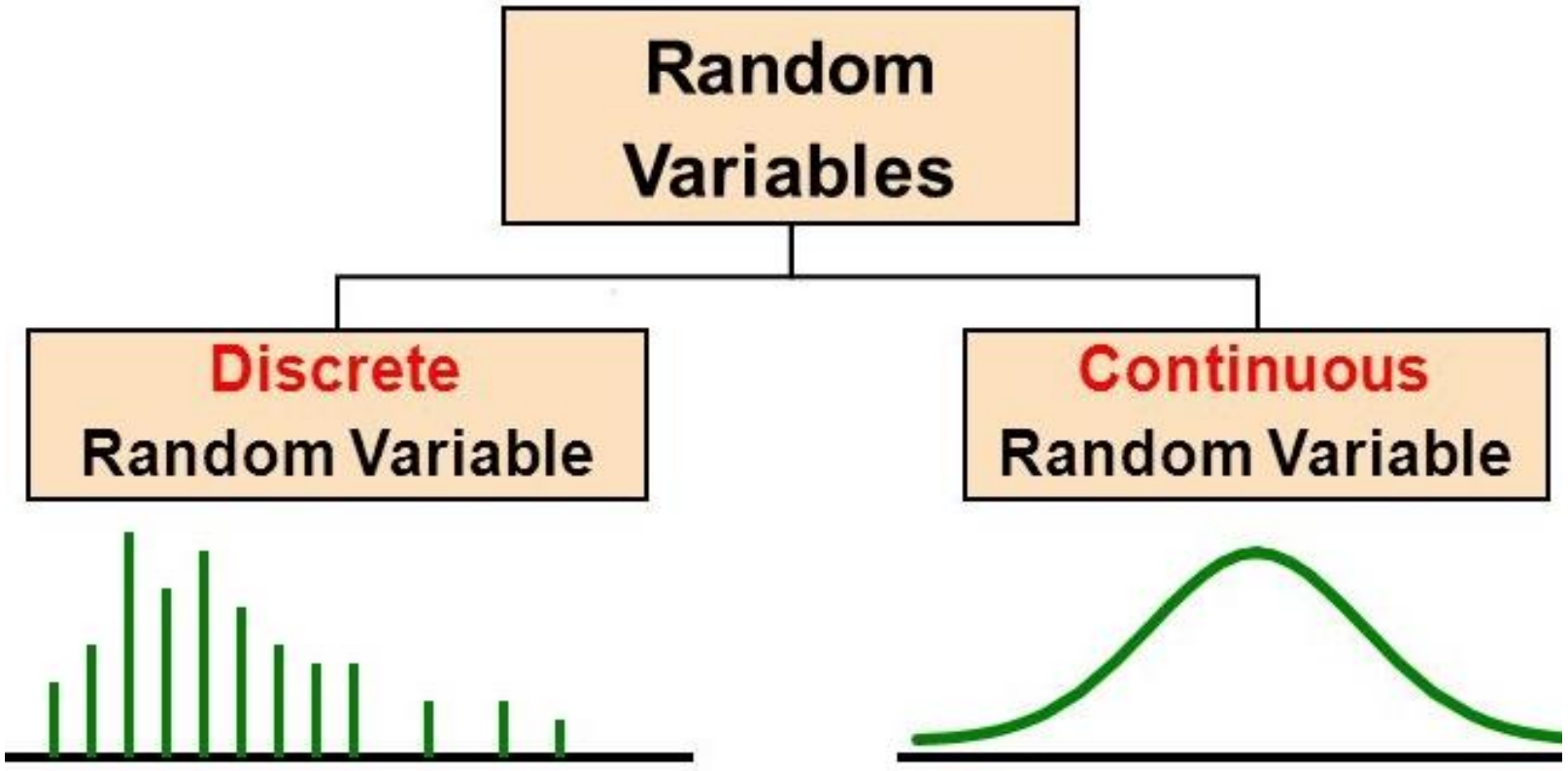
… run the `lm(acc ~ IQ)`

BASICS OF PROBABILITY



Probability Functions

cognitive model
statistics
computing



Probability

cognitive model
statistics
computing

…assigning numbers to a set of possibilities

Properties (Kolmogorov, 1956)

- $p \in [0,1]$
- $\Sigma p = 1$

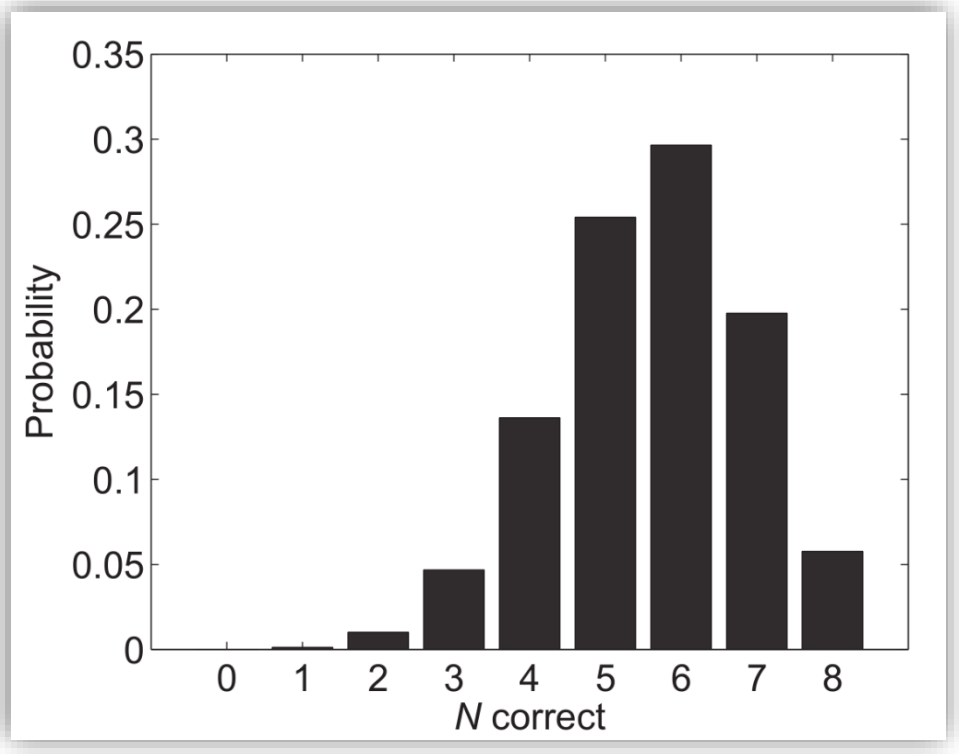
Probabilities are used to express uncertainty.

Probability Functions

cognitive model
statistics
computing

discrete variable – we talk about mass

- Run a test of 8 questions, and record each student's correct responses
- Count and plot the # of correct answers
- Then divide those counts by the total number of students
- The lengths of these bars sum up to 1



Lewandowsky & Farrell (2011)

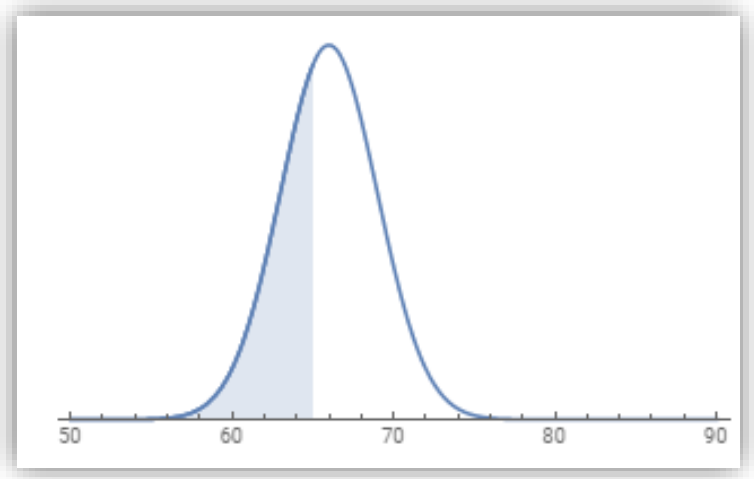
Probability Functions

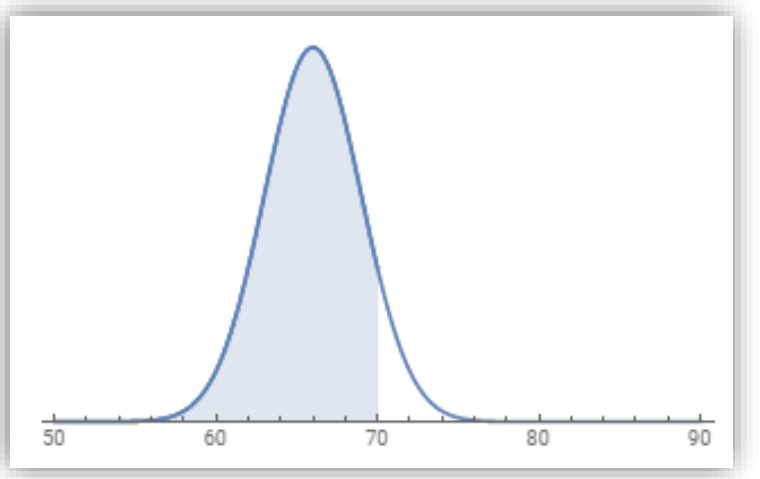
cognitive model
statistics
computing

continuous variable – we talk about density

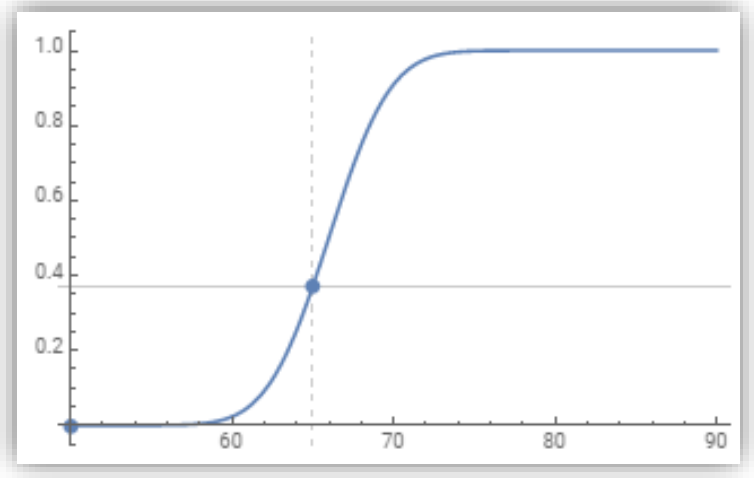
probability density function (PDF)

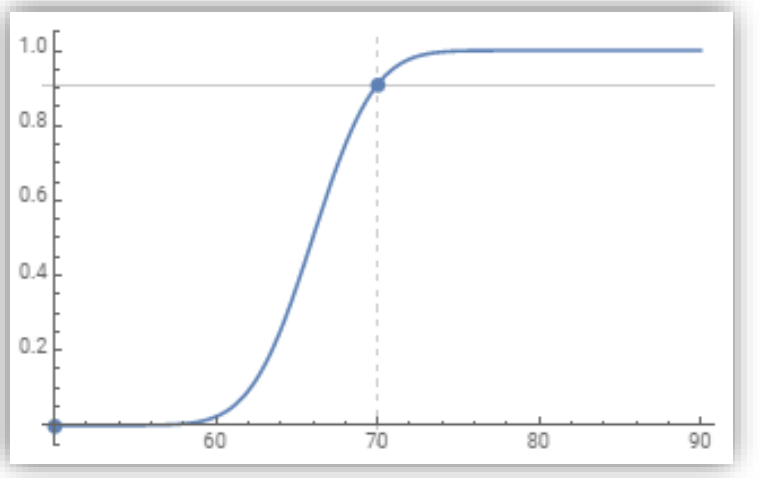
The entire area under the curve of PDF is 1.





cumulative distribution function (CDF)

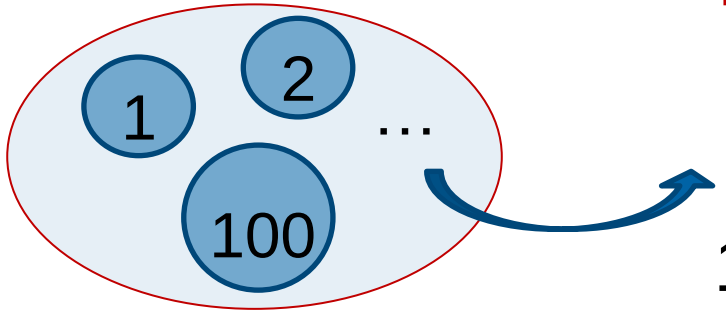




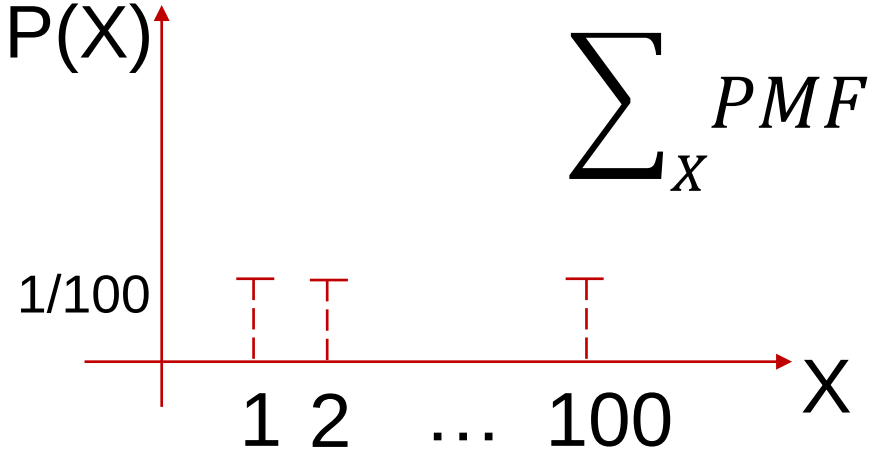
https://demonstrations.wolfram.com/ConnectingTheCDFAndThePDF/

Another example

Discrete



X	P(X)
1	1/100
2	1/100
…	…
100	1/100



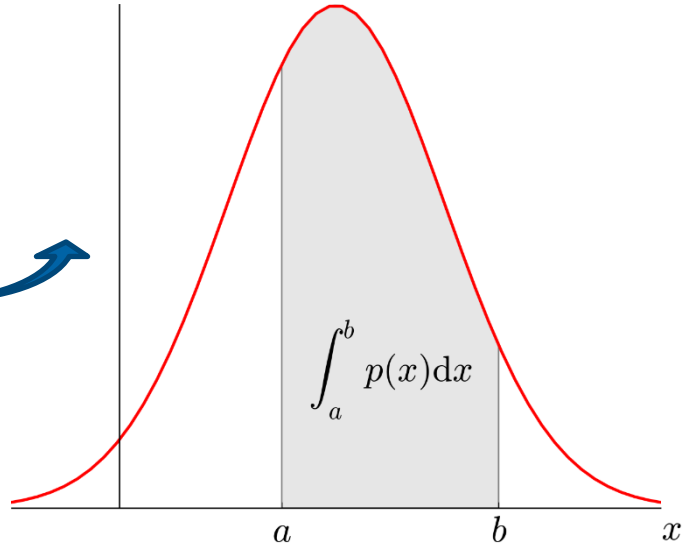
$$\sum_X PMF(X) = 1$$

Continuous

Height

X	P(X)
1.8 m	0

PDF



P given by the area

$$1.75 \leq X \leq 1.85$$

Playing with Probability Functions in R

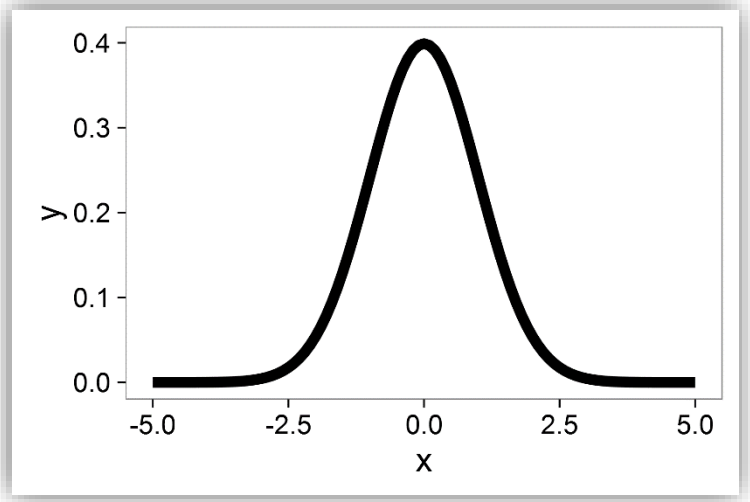
cognitive model
statistics
computing

```
dnorm() - PDF
pnorm() - CDF
qnorm() - quantile, inverse cdf
rnorm() - random number generator
```

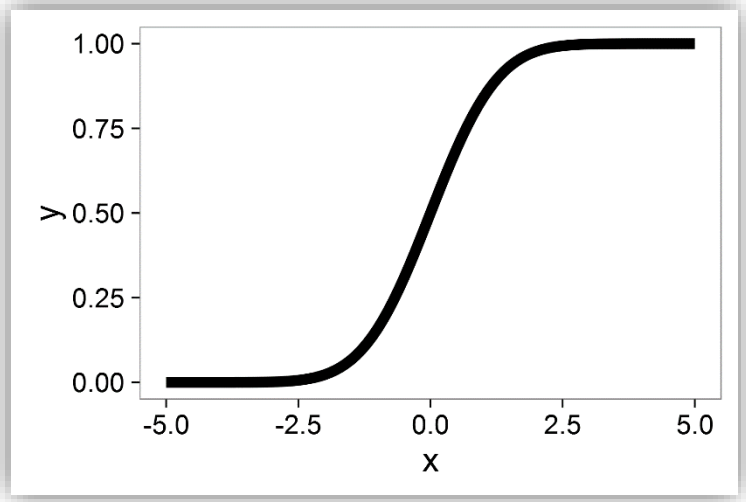
Example: Normal(0,1)

cognitive model
statistics
computing

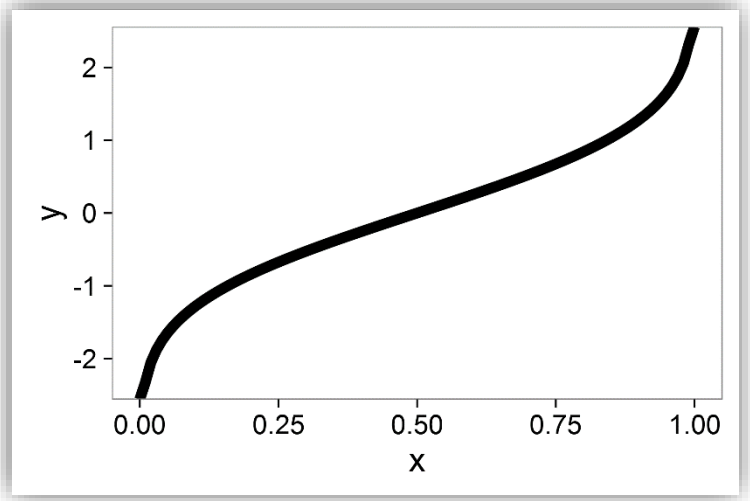
dnorm()



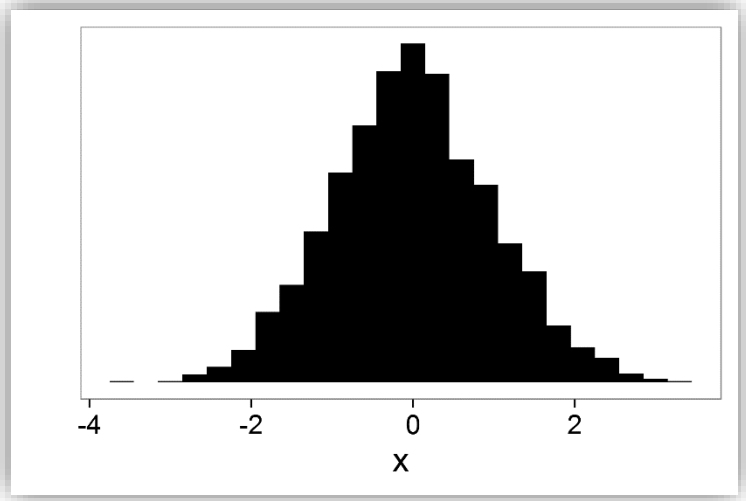
pnorm()



qnorm()



rnorm()



Joint, Marginal and Conditional Probability

Joint Probability

$p(A, B) = p(B, A)$

- e.g., $p(\text{rain, cold})$: $p(\text{rain})$ AND $p(\text{cold})$

Marginal Probability

$p(A)$ – 'p of A irrespective of B'

- e.g., $p(\text{rain})$: $p(\text{rain, cold}) + p(\text{rain, not cold})$

Conditional Probability

$p(A|B)$ – 'p of A given B' – event B is fixed, not uncertainty

$p(A,B) = p(A|B)p(B)$

- e.g., $p(\text{rain, cold}) = p(\text{rain}|\text{cold})p(\text{cold})$

Example I: discrete

cognitive model
statistics
computing

rain
x

cold Y	1	0
1	0.5	0.1
0	0.1	0.3

Joint probability :

$$P(X = 0, Y = 1) =$$

$$\sum_{x,y} P(X = x, Y = y) = 1$$

Marginal probability :

$$P(Y = 1) =$$

$$P(X = 0) =$$

$$P(X = x) = \sum_{y} P(X = x, Y = y)$$

Conditional probability :

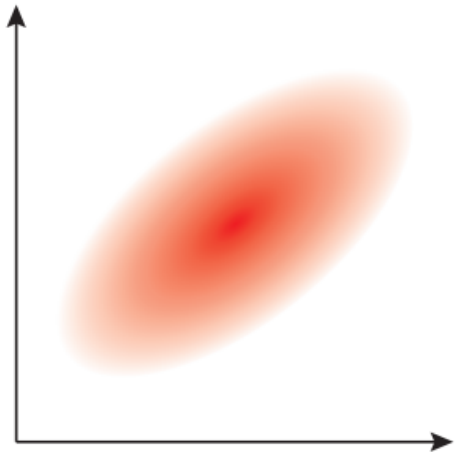
$$P(X = 1|Y = 1) =$$

$$P(X = x|Y = y) = \frac{P(X = x, Y = y)}{P(Y = y)} = \frac{P(X = x, Y = y)}{\sum_x P(X = x, Y = y)}$$

Example I: continuous

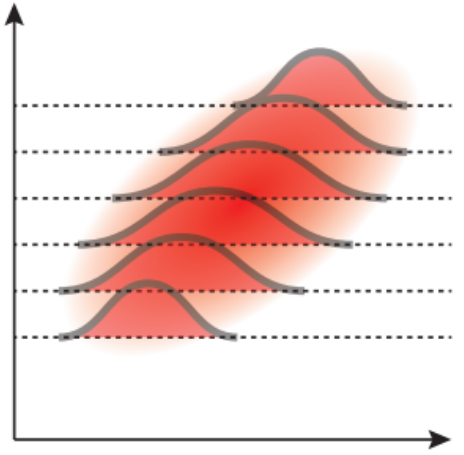
cognitive model
statistics
computing

joint distribution



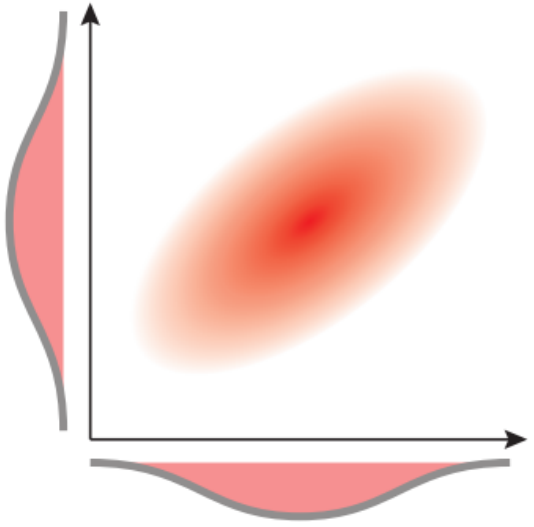
The "co-distribution" of x and y.

conditional distribution



The probability distribution of x, given that we know the value of y.

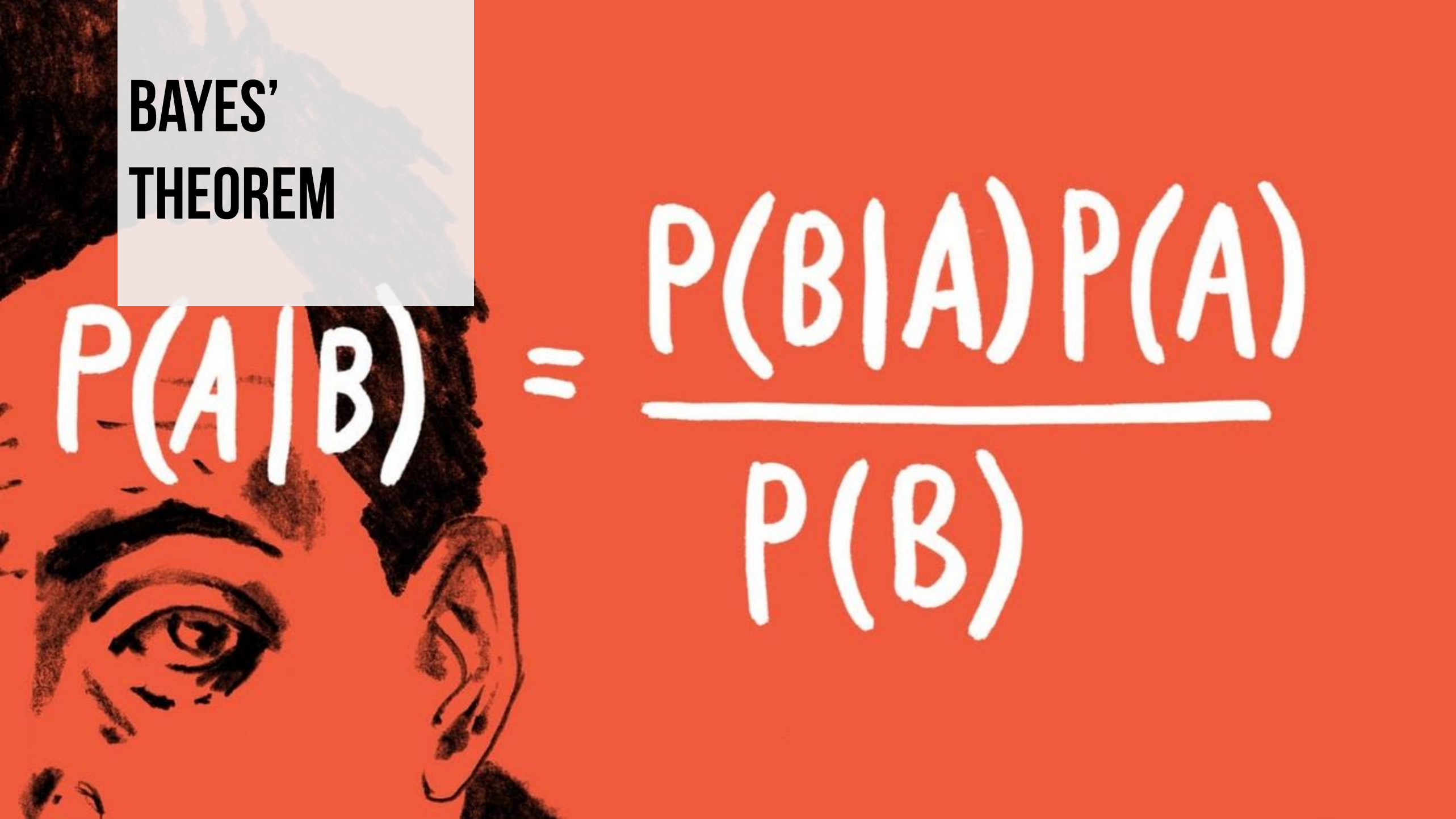
mariginal distribution



The density of x- (or y-) values, without knowing the other's value.

Credit: Eric Ma

BAYES' THEOREM



Bayes' theorem

$$p(A,B) = p(B,A)$$

$$p(A,B) = p(A|B)p(B)$$

$$p(B,A) = p(B|A)p(A)$$

$$p(A|B)p(B) = p(B|A)p(A)$$

$$p(A \mid B) = \frac{p(B \mid A)\,p(A)}{p(B)}$$

Second Example

	Column			
Row	...	c	...	Marginal
$\vdots$		$\vdots$		
r	...	$p(r, c) = p(r\|c)\, p(c)$	...	$p(r) = \sum_{c^*} p(r\|c^*)\, p(c^*)$
$\vdots$		$\vdots$		
Marginal		$p(c)$		

Kruschke (2015)

Second Example

cognitive model
statistics
computing

	Hair color				
Eye color	Black	Brunette	Red	Blond	Marginal (Eye color)
Brown	0.11	0.20	0.04	0.01	0.37
Blue	0.03	0.14	0.03	0.16	0.36
Hazel	0.03	0.09	0.02	0.02	0.16
Green	0.01	0.05	0.02	0.03	0.11
Marginal (hair color)	0.18	0.48	0.12	0.21	1.0

$$p(A \mid B) = \frac{p(B \mid A)\,p(A)}{\sum_{A^*} p(B \mid A^*)\,p(A^*)}$$

Kruschke (2015)

Exercise VI

Suppose that in the general population, the probability of having a rare disease is 1/1000. We denote the true presence or absence of the disease as the value of a parameter, ϑ, that can have the value ϑ = ☹ if disease is present in a person, or the value ϑ = ☺ if the disease is absent. The base rate of the disease is therefore denoted $p(\vartheta = ☹) = 0.001$.

Suppose(1): a test for the disease that has a 99% hit rate: $p(T = + | \vartheta = ☹) = 0.99$

Suppose(2): the test has a false alarm rate of 5%: $p(T = + | \vartheta = ☺) = 0.05$

Q: Suppose we sample a person at random from the population, administer the test, and it comes up positive. What is the posterior probability that the person has the disease?

Exercise VI

Q: What is the posterior probability that the person has the disease?

→ $p(\vartheta = ☹ \mid T = +)$

Exercise VI

Test result	Disease: $\theta =$ ☹ (present)	Disease: $\theta =$ ☺ (absent)	Marginal (test result)
$T = +$	$p(+\|$☹$)\, p($☹$) = 0.99 \cdot 0.001$	$p(+\|$☺$)\, p($☺$) = 0.05 \cdot (1 - 0.001)$	$\sum_\theta p(+\|\theta)\, p(\theta)$
$T = -$	$p(-\|$☹$)\, p($☹$) = (1 - 0.99) \cdot 0.001$	$p(-\|$☺$)\, p($☺$) = (1 - 0.05) \cdot (1 - 0.001)$	$\sum_\theta p(-\|\theta)\, p(\theta)$
Marginal (disease)	$p($☹$) = 0.001$	$p($☺$) = 1 - 0.001$	1.0

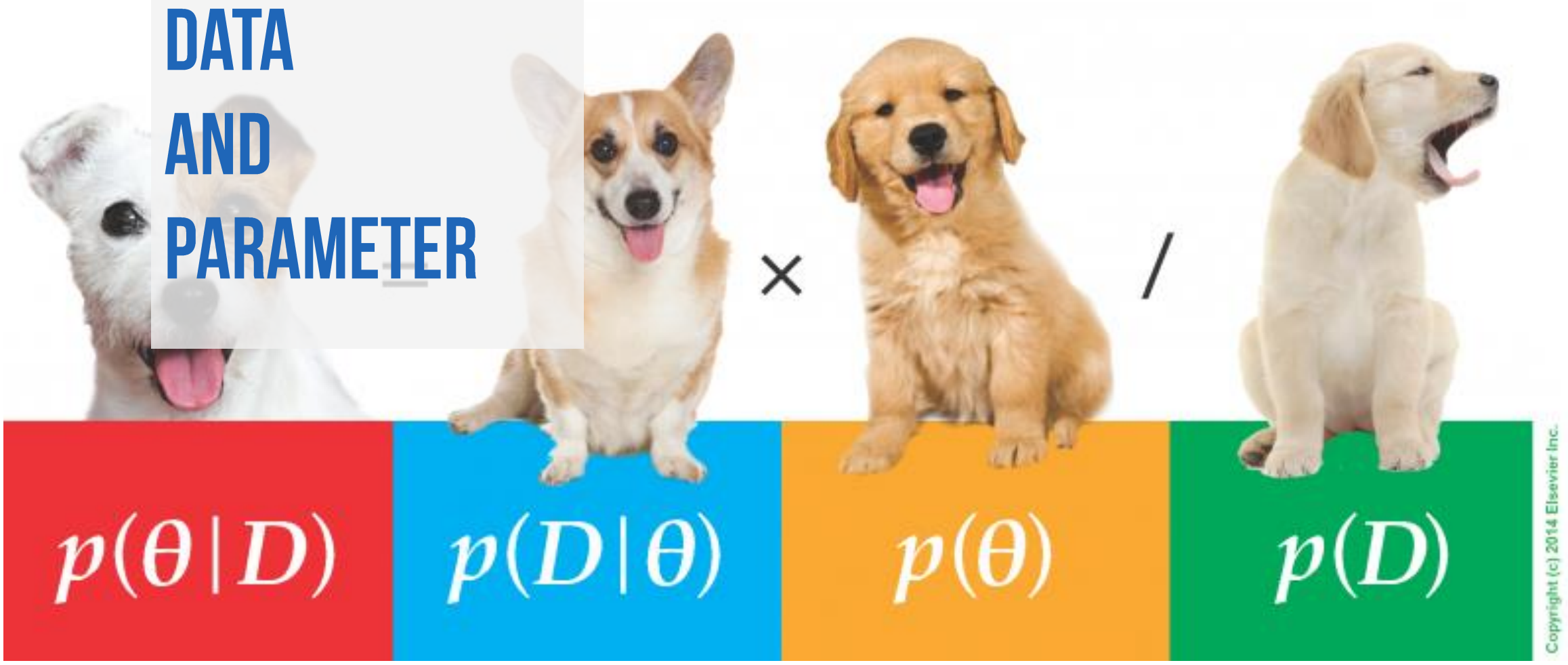
$$p(\theta = \text{☹}\,|\,T = +) = \frac{p(T = +|\theta = \text{☹})\, p(\theta = \text{☹})}{\sum_\theta p(T = +|\theta)\, p(\theta)}$$

$$= \frac{0.99 \cdot 0.001}{0.99 \cdot 0.001 + 0.05 \cdot (1 - 0.001)}$$

$$= 0.019$$

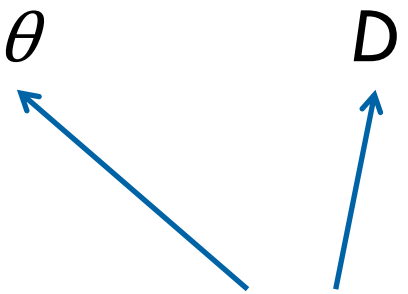
Kruschke (2015)

LINKING DATA AND PARAMETER



Linking Data and Parameter

cognitive model
statistics
computing



$$p(A|B) = \frac{p(B|A)p(A)}{p(B)}$$

Linking Data and Parameter

cognitive model
statistics
computing

$$p(\theta|D) = \frac{p(D|\theta)p(\theta)}{p(D)}$$

Linking Data and Parameter

cognitive model
statistics
computing

Likelihood

How plausible is the data given our parameter is true?

Prior

How plausible is our parameter before observing the data?

$$p(\theta|D) = \frac{p(D|\theta)p(\theta)}{p(D)}$$

Posterior

How plausible is our parameter given the observed data?

Evidence

How plausible is the data under all possible parameters?

What is $p(\text{Data}|\vartheta)$

cognitive model
statistics
computing

L(theta | Data)

- This is the "Model"
- Data is fixed, ϑ varies
- Not a probability distribution
  - the sum is not "one"

$$Pr(X=0|\theta)=Pr(T,T|\theta)=Pr(T|\theta)\times Pr(T|\theta)=(1-\theta)^2$$

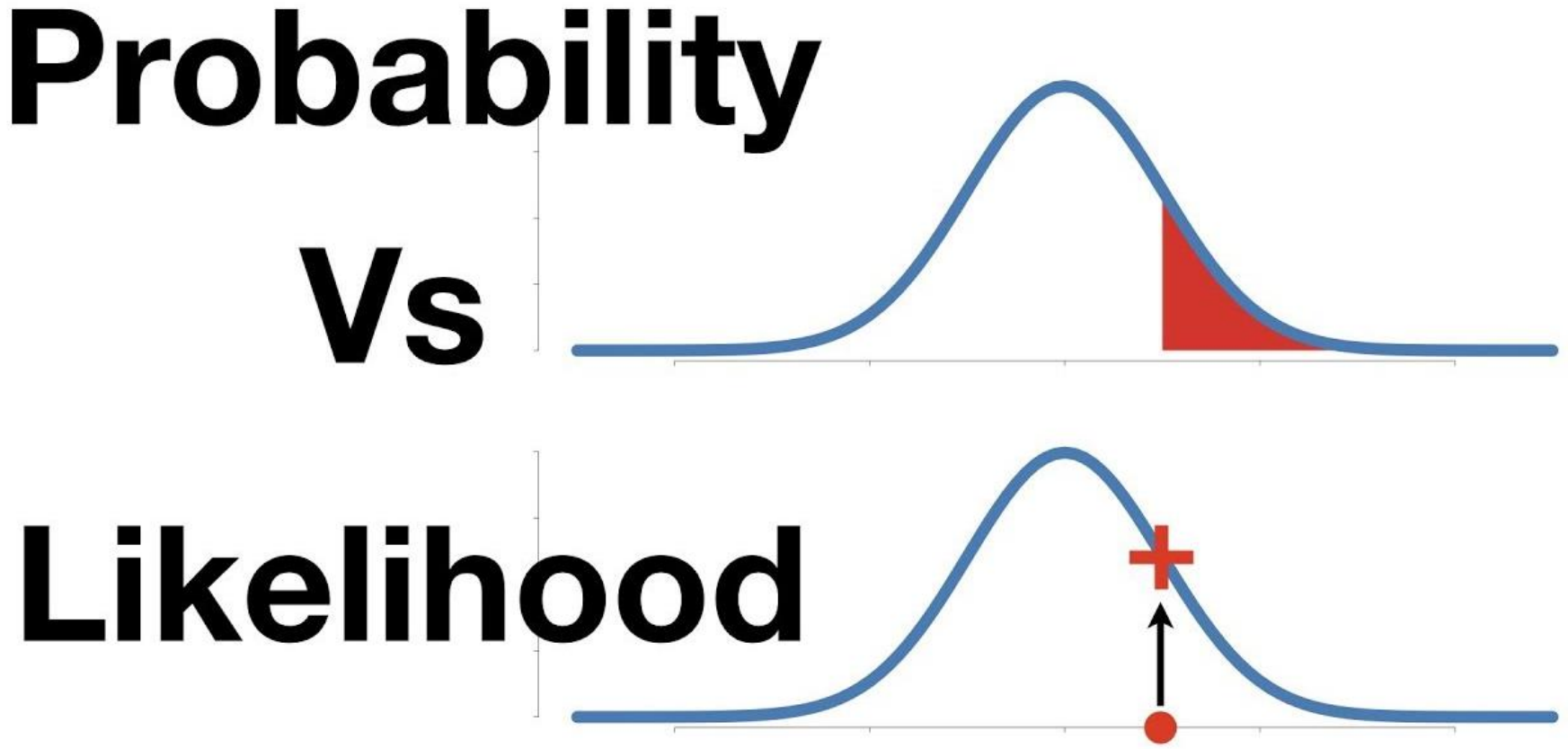
$$Pr(X=1|\theta)=Pr(H,T|\theta)+Pr(T,H|\theta)=2\times Pr(T|\theta)\times Pr(H|\theta)=2\theta(1-\theta)$$

$$Pr(X=2|\theta)=Pr(H,H|\theta)=Pr(H|\theta)\times Pr(H|\theta)=\theta^2.$$

Probability of coin landing heads up, θ	Number of heads, X 0	1	2	Total
0.0	1.00	0.00	0.00	1.00
0.2	0.64	0.32	0.04	1.00
0.4	0.36	0.48	0.16	1.00
0.6	0.16	0.48	0.36	1.00
0.8	0.04	0.32	0.64	1.00
1.0	0.00	0.00	1.00	1.00
Total	2.20	1.60	2.20	

Lambert (2018)

Watch this video!

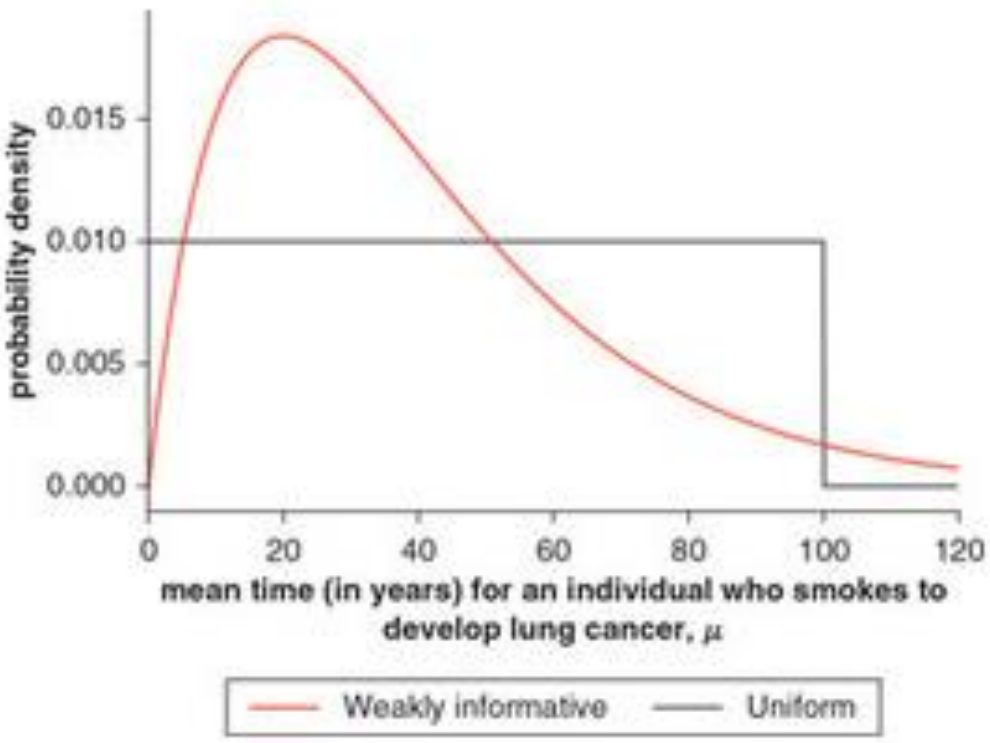


StatQuest with Josh Starmer
250K subscribers

https://youtu.be/pYxNSUDSFH4

What is $p(\vartheta)$?

cognitive model
statistics
computing



Lambert (2018)

What is *p*(*Data*)?

cognitive model
statistics
computing

discrete parameters

$$p(\theta \mid D) = \frac{p(D \mid \theta)\, p(\theta)}{\sum_{\theta^*} p(D \mid \theta^*)\, p(\theta^*)}$$

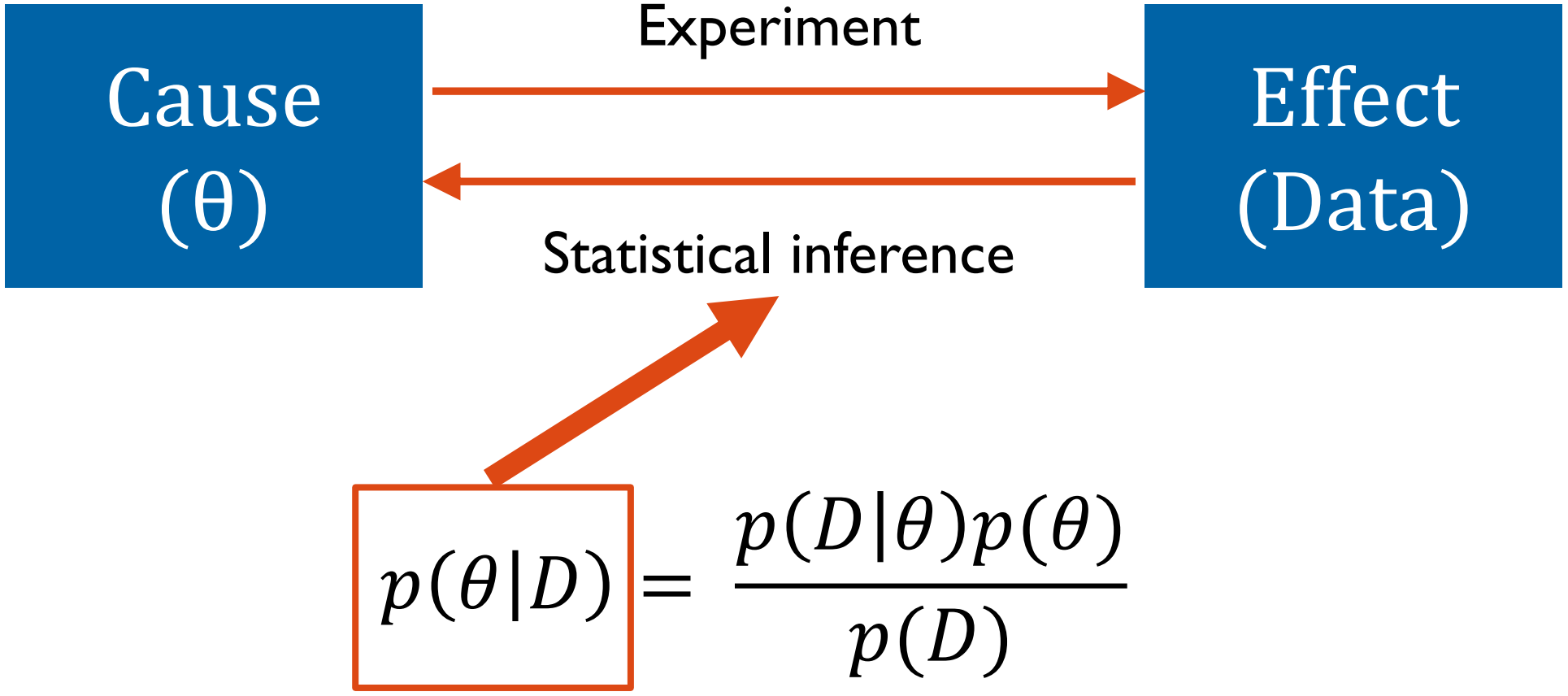
$$p(\theta|D) = \frac{p(D|\theta)p(\theta)}{p(D)}$$

continuous parameters

$$p(\theta \mid D) = \frac{p(D \mid \theta)\, p(\theta)}{\int p(D \mid \theta^*)\, p(\theta^*)\, d\theta^*}$$

Why the Bayes' theorem is important?

cognitive model
statistics
computing



BINOMIAL MODEL

Binomial Model

cognitive model
statistics
computing

- You are curious how much of the surface is covered in water.
- You will toss the globe up in the air.
- You will record whether or not the surface under your right index finger is water (W) or land (L).
- You might observe: W L W W W L W L W
- → 6/9 = 0.666667?
- Is it right? If not, what to do next?

McElreath (2016)

Steps of (Bayesian) Modeling?

cognitive model
statistics
computing

A data story

Think about how the data might arise.
It can be *descriptive* or even *causal*.

Update

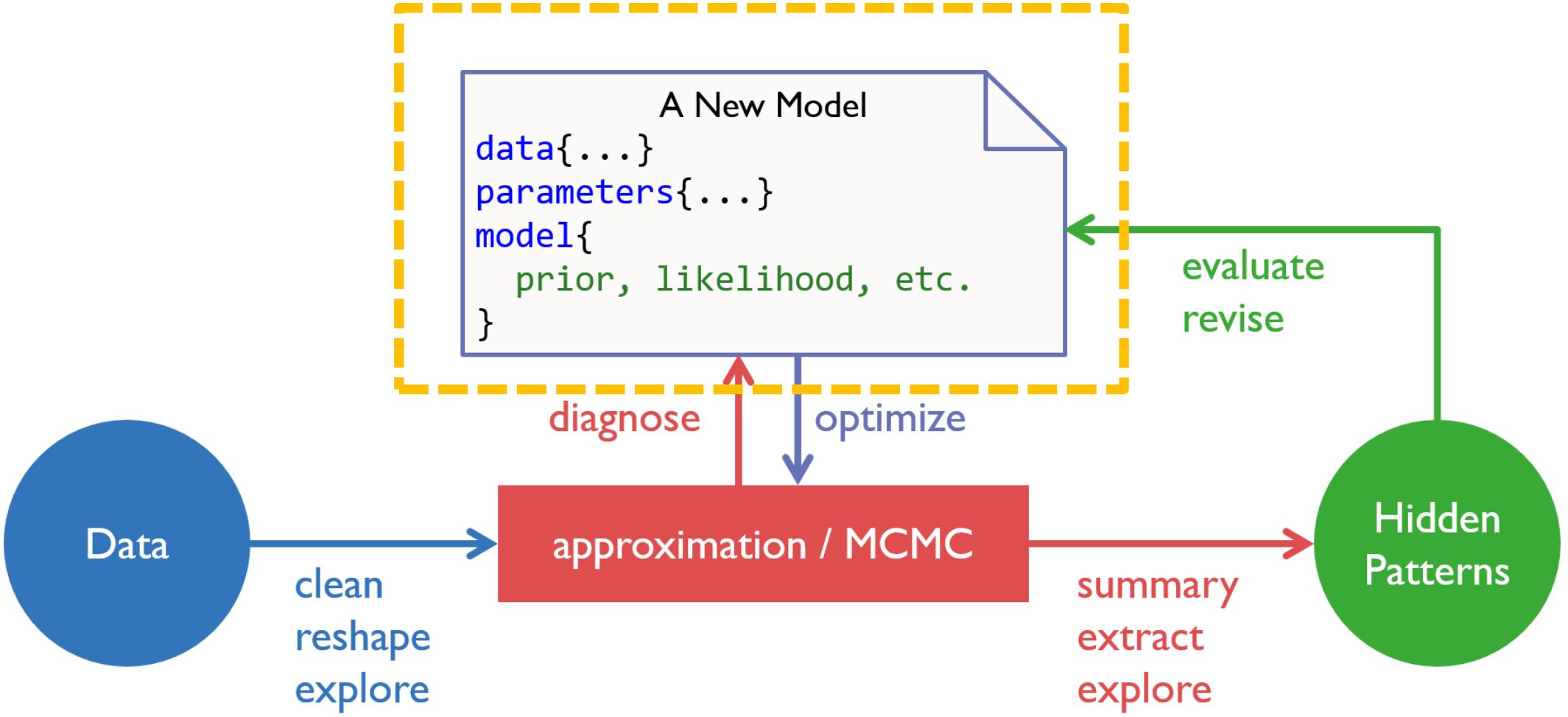
Educate your model by feeding it with data.
Bayesian Update:
update the prior, in light of data, to produce posterior
the updated posterior then becomes the prior of next update

Evaluate

Compare model with reality.
Revise your model.

McElreath (2016)

cognitive model
statistics
computing



A Data Story of the Globe

cognitive model
statistics
computing

- The true proportion of water covering the globe is ϑ.
- A single toss of the globe has a probability p of producing a water (W) observation.
- It has a probability $(1 - \vartheta)$ of producing a land (L) observation.
- Each toss of the globe is independent of the others.

McElreath (2016)

Components of a Model

cognitive model
statistics
computing

think about the likelihood function (of Binomial):

$$p(\theta \mid D) = \frac{p(D \mid \theta)\, p(\theta)}{\int p(D \mid \theta^*)\, p(\theta^*)\, d\theta^*}$$

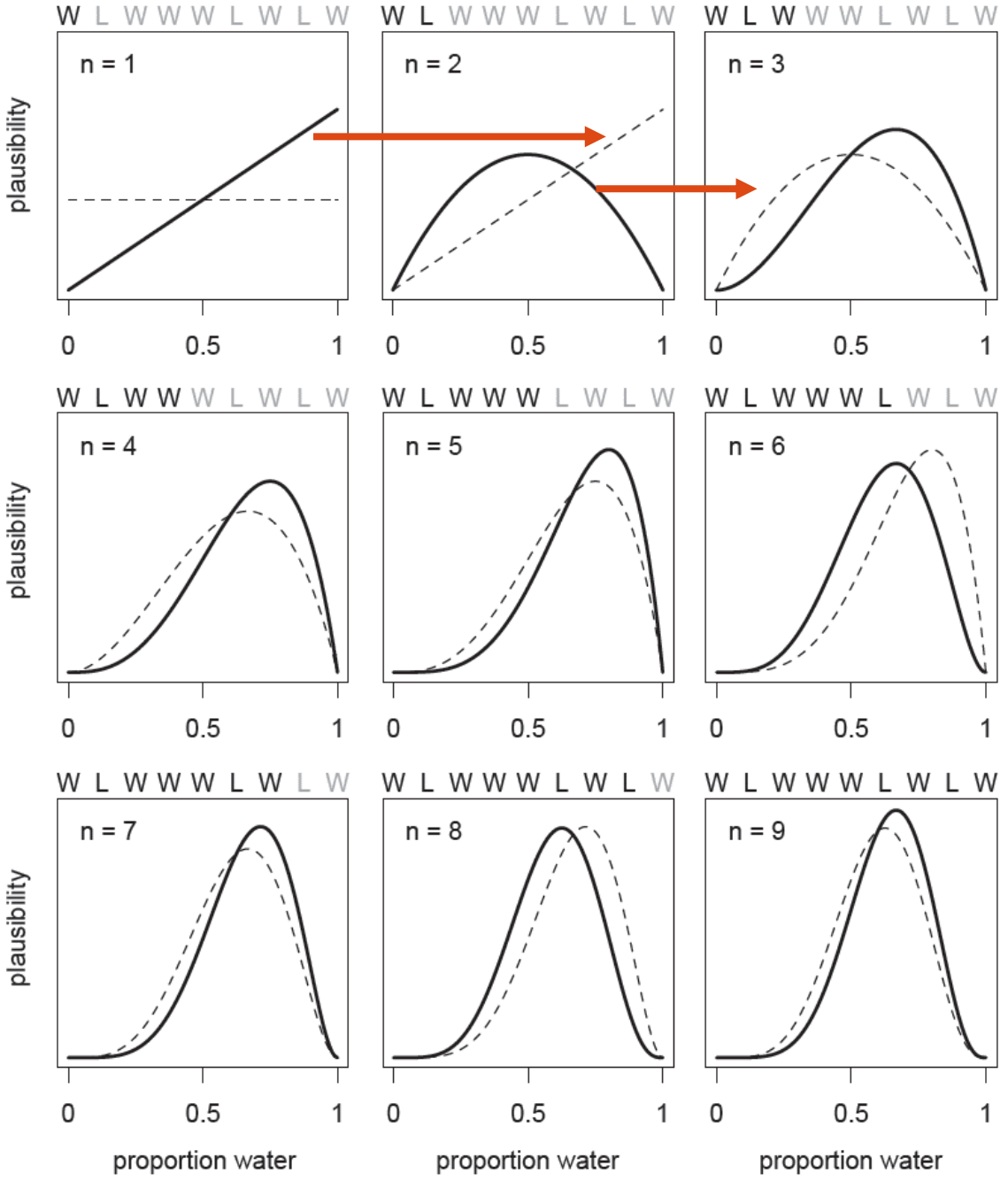
$$p(w \mid N, \theta) = \binom{N}{w} \theta^w (1-\theta)^{N-w}$$

N: total number of observations
w: number of water
ϑ : proportion of water

} known (data)

unknown (parameter)

Update



cognitive model

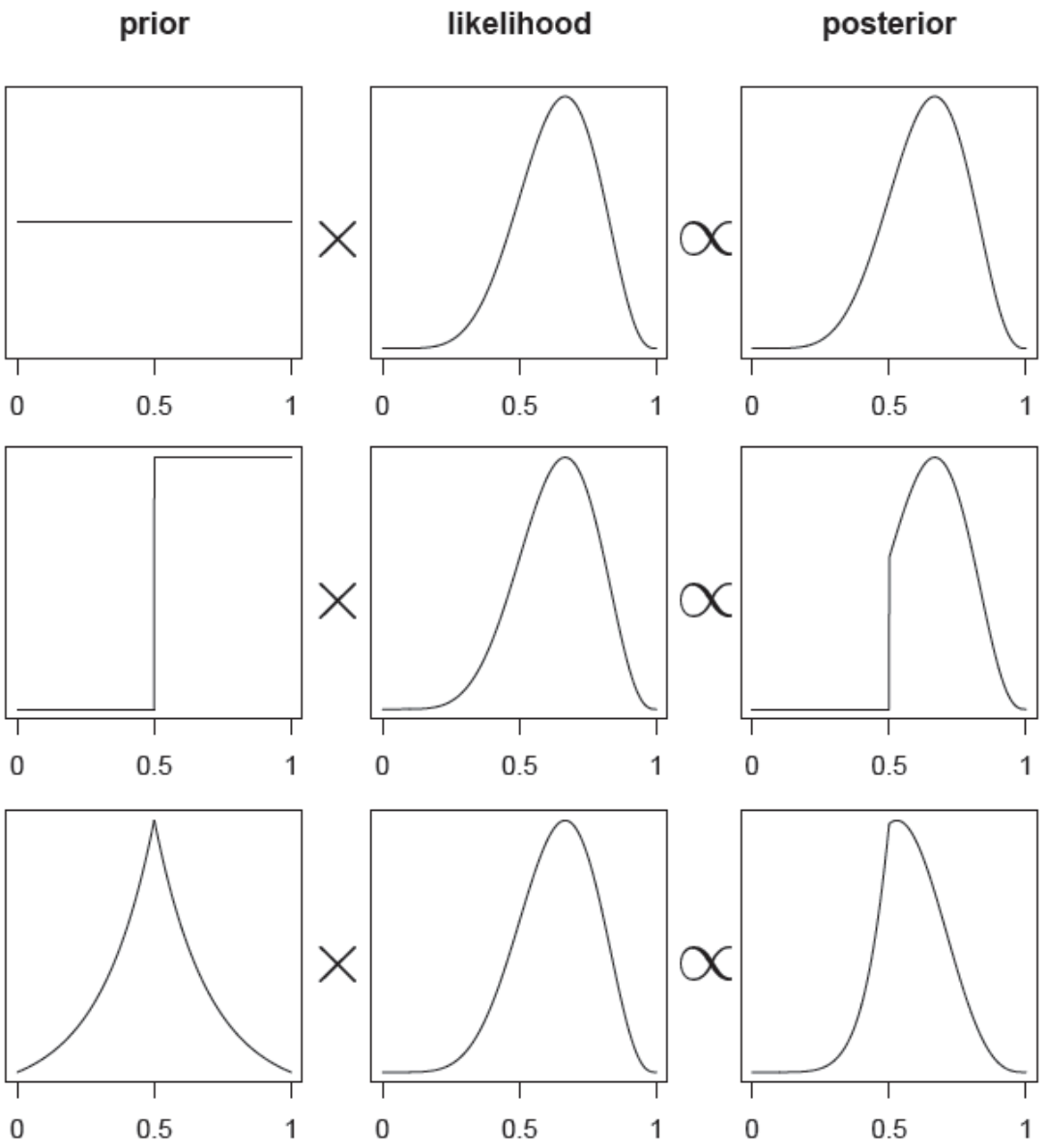
statistics

computing

McElreath (2016)

Impact of Prior

cognitive model
statistics
computing



McElreath (2016)

Solve it by Grid Approximation

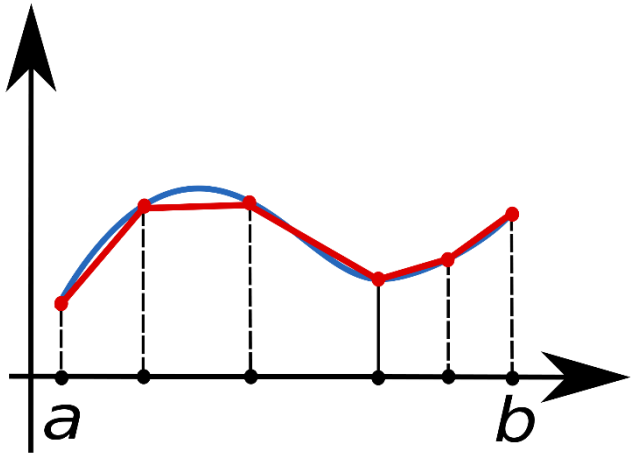
cognitive model
statistics
computing

discrete parameters

$$p(\theta \mid D) = \frac{p(D \mid \theta)p(\theta)}{\sum_{\theta^*} p(D \mid \theta^*)p(\theta^*)}$$

continuous parameters

$$p(\theta \mid D) = \frac{p(D \mid \theta)p(\theta)}{\int p(D \mid \theta^*)p(\theta^*)d\theta^*}$$



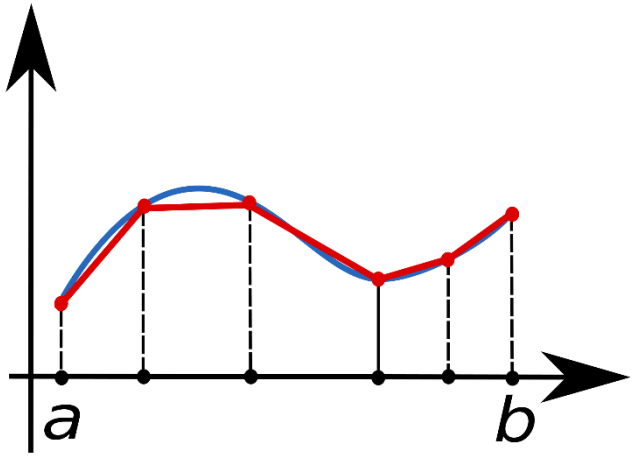
Binomial Model – Grid Approximation

cognitive model
statistics
computing

```
theta_start <- 0; theta_end <- 1; n_grid <- 20
w <- 6; N <- 9

# define grid
theta_grid <- seq(from = theta_start, to = theta_end,
                  length.out = n_grid)

# define prior
prior <- rep(1 , n_grid)

# compute likelihood at each value in grid
likelihood <- dbinom(w, size = N, prob = theta_grid)

# compute product of likelihood and prior
unstd.posterior <- likelihood * prior

# standardize the posterior, so it sums to 1
posterior <- unstd.posterior / sum(unstd.posterior)
```

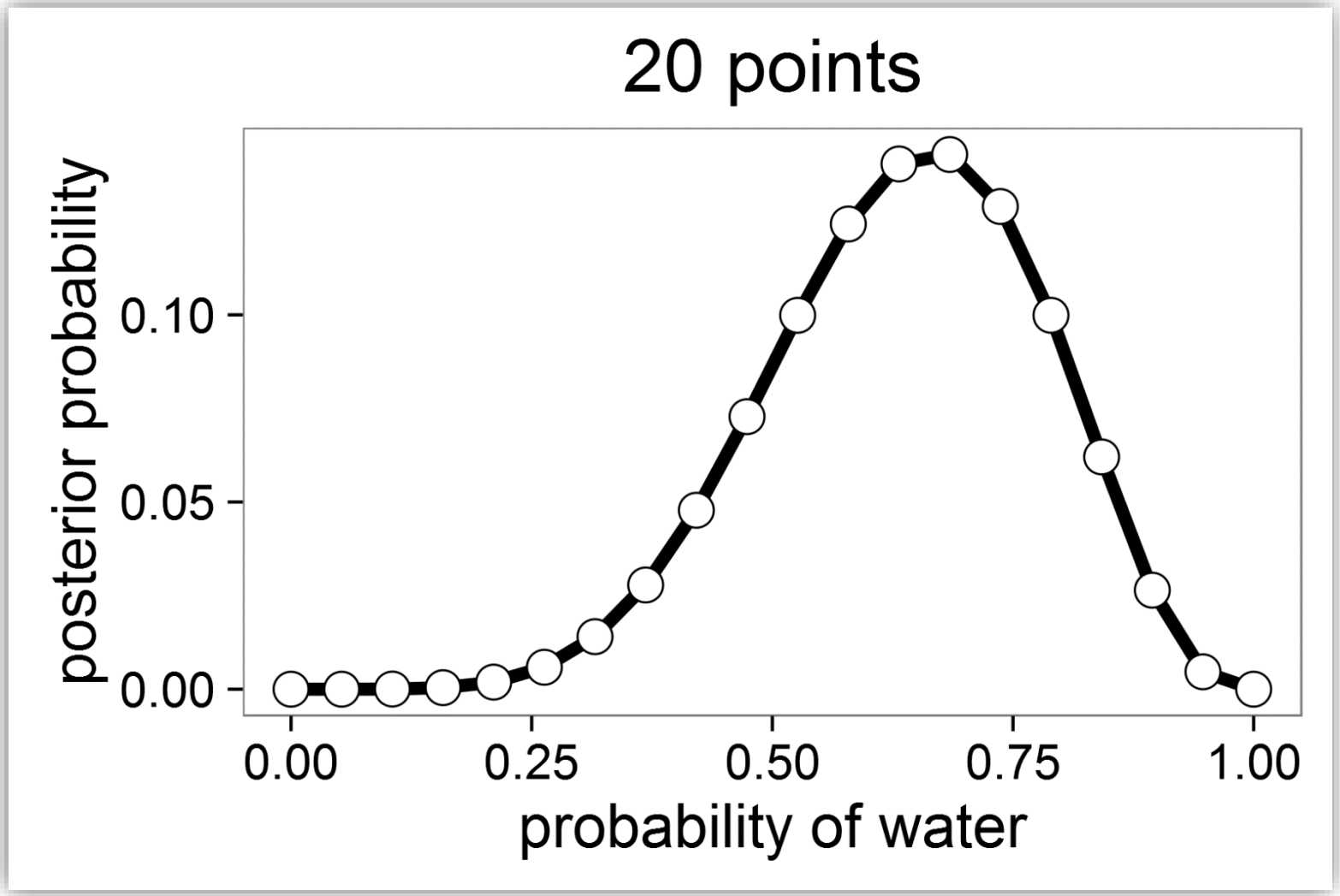
$$p(\theta \mid D) = \frac{p(D \mid \theta)p(\theta)}{\int p(D \mid \theta^*)p(\theta^*)d\theta^*}$$

$$p(w \mid N, \theta) = \binom{N}{w} \theta^w (1-\theta)^{N-w}$$



Binomial Model – Grid Approximation

cognitive model
statistics
computing



Exercise VII

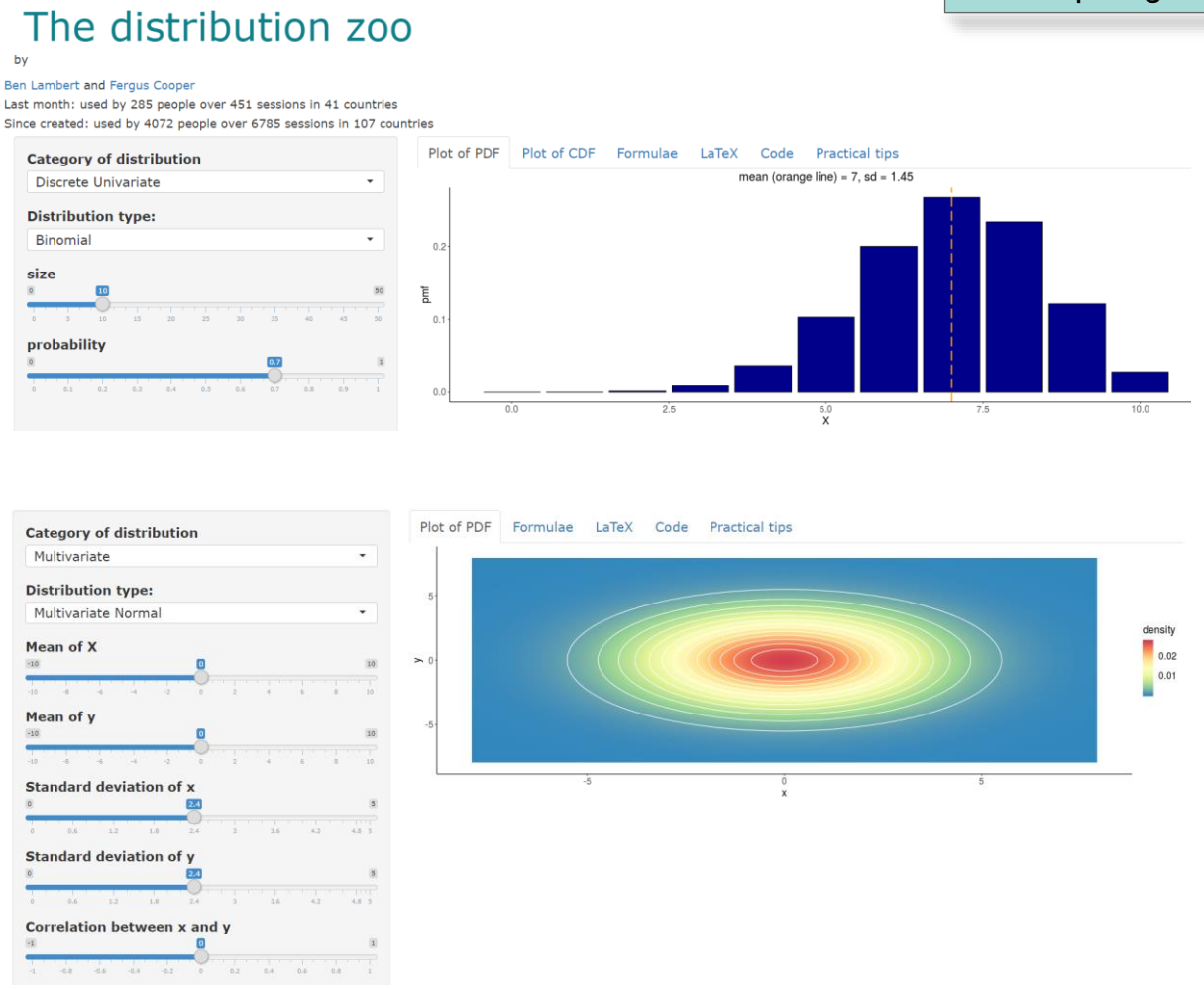
cognitive model
statistics
computing

`…/BayesCog/02.binomial_globe/_scripts/binomial_globe_grid.R`

TASK: run a grid approximation with `grid_size = 50`

How do I know which likelihood to use?

cognitive model
statistics
computing



Lambert (2018)

https://ben18785.shinyapps.io/distribution-zoo/

What if I have multiple parameters?

cognitive model
statistics
computing

grid approximation for
2 parameters?
5 parameters?
10 parameters?

$$p(\theta \mid D) = \frac{p(D \mid \theta)p(\theta)}{\int p(D \mid \theta^*)p(\theta^*)d\theta^*}$$

$$p(data) = \int_{All\theta_1} \int_{All\theta_2} p(data, \theta_1, \theta_2) \mathrm{d}\theta_1 \mathrm{d}\theta_2$$

$$p(data) = \int_{\mu_1} \int_{\sigma_1} \dots \int_{\mu_{100}} \int_{\sigma_{100}} \underbrace{p(data \mid \mu_1, \sigma_1, \dots, \mu_{100}, \sigma_{100})}_{\text{likelihood}} \times \underbrace{p(\mu_1, \sigma_1, \dots, \mu_{100}, \sigma_{100})}_{\text{prior}}$$
$$\mathrm{d}\mu_1 \mathrm{d}\sigma_1 \dots \mathrm{d}\mu_{100} \mathrm{d}\sigma_{100},$$

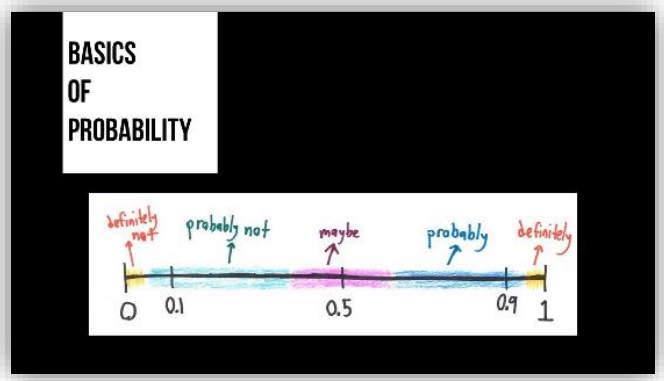
- Analytical solutions (often does not exist)
- Grid approximation (takes too long)

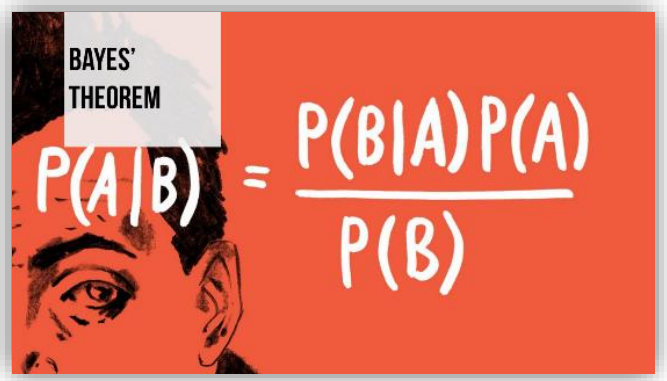
$$p(\theta \mid D) \propto p(D \mid \theta)p(\theta)$$

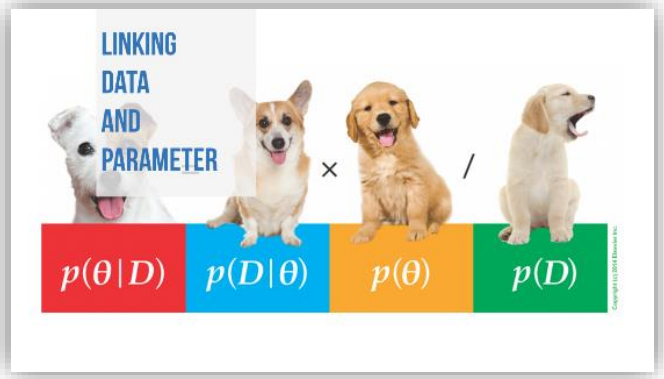
- solution: Markov Chain Monte Carlo

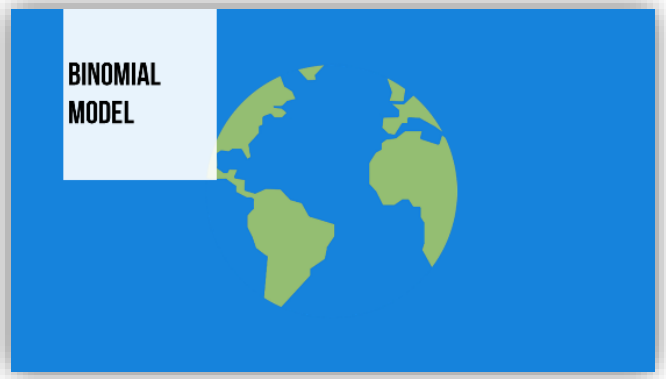
Recap


BASICS
OF
R
PROGRAMMING
R Studio


BASICS
OF
PROBABILITY
definitely not
probably not
maybe
probably
definitely
0
0.1
0.5
0.9
1


BAYES'
THEOREM
P(A|B) = P(B|A)P(A) / P(B)


LINKING
DATA
AND
PARAMETER
×
/
p(θ|D)
p(D|θ)
p(θ)
p(D)


BINOMIAL
MODEL

What if I have multiple parameters?

cognitive model
statistics
computing

grid approximation for
2 parameters?
5 parameters?
10 parameters?

$$p(\theta \mid D) = \frac{p(D \mid \theta)p(\theta)}{\int p(D \mid \theta^*)p(\theta^*)d\theta^*}$$

$$p(data) = \int_{All\theta_1} \int_{All\theta_2} p(data, \theta_1, \theta_2) \mathrm{d}\theta_1 \mathrm{d}\theta_2$$

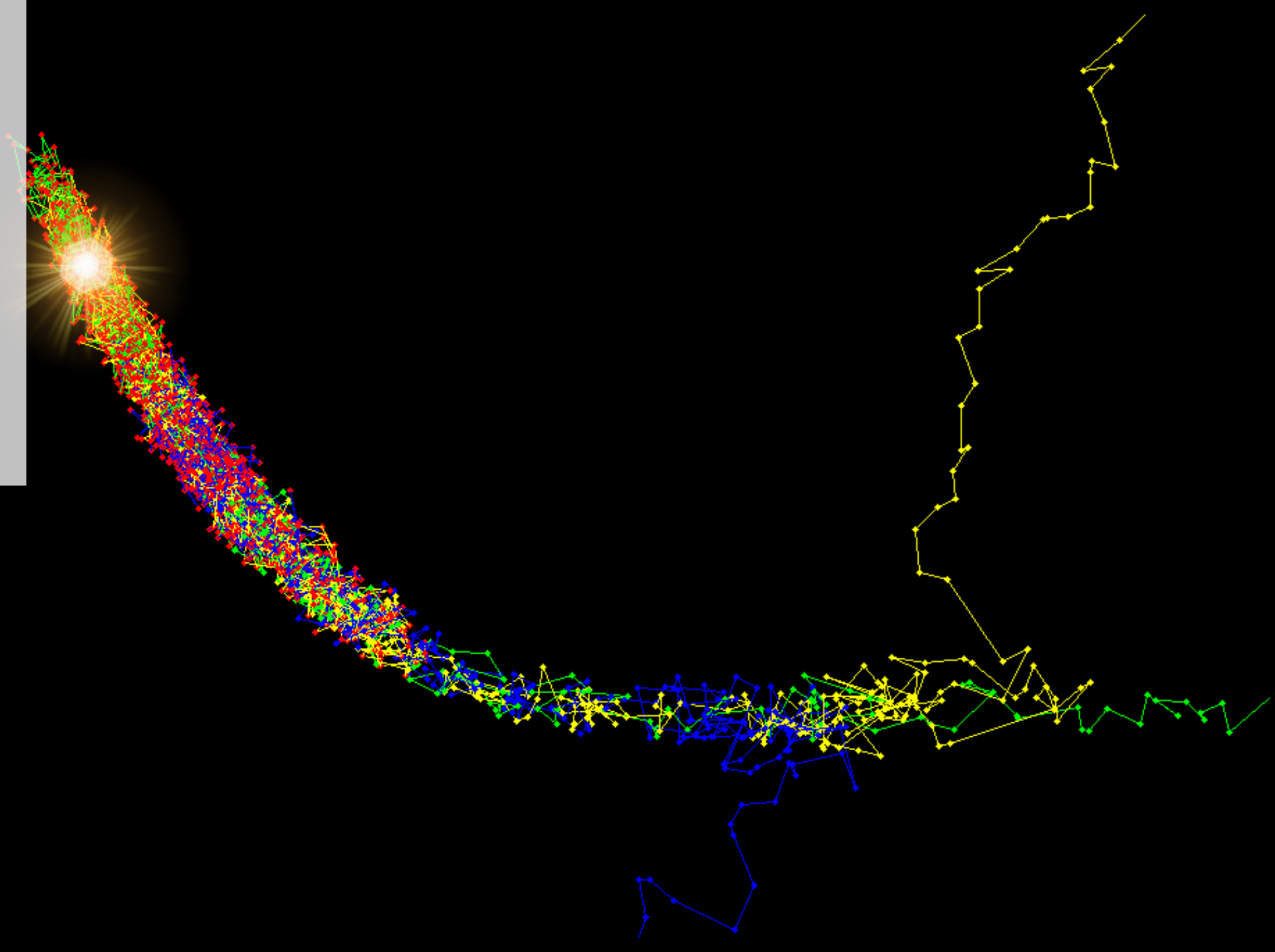
$$p(data) = \int_{\mu_1} \int_{\sigma_1} \dots \int_{\mu_{100}} \int_{\sigma_{100}} \underbrace{p(data \mid \mu_1, \sigma_1, \dots, \mu_{100}, \sigma_{100})}_{\text{likelihood}} \times \underbrace{p(\mu_1, \sigma_1, \dots, \mu_{100}, \sigma_{100})}_{\text{prior}}$$
$$\mathrm{d}\mu_1 \mathrm{d}\sigma_1 \dots \mathrm{d}\mu_{100} \mathrm{d}\sigma_{100},$$

- Analytical solutions (often does not exist)
- Grid approximation (takes too long)

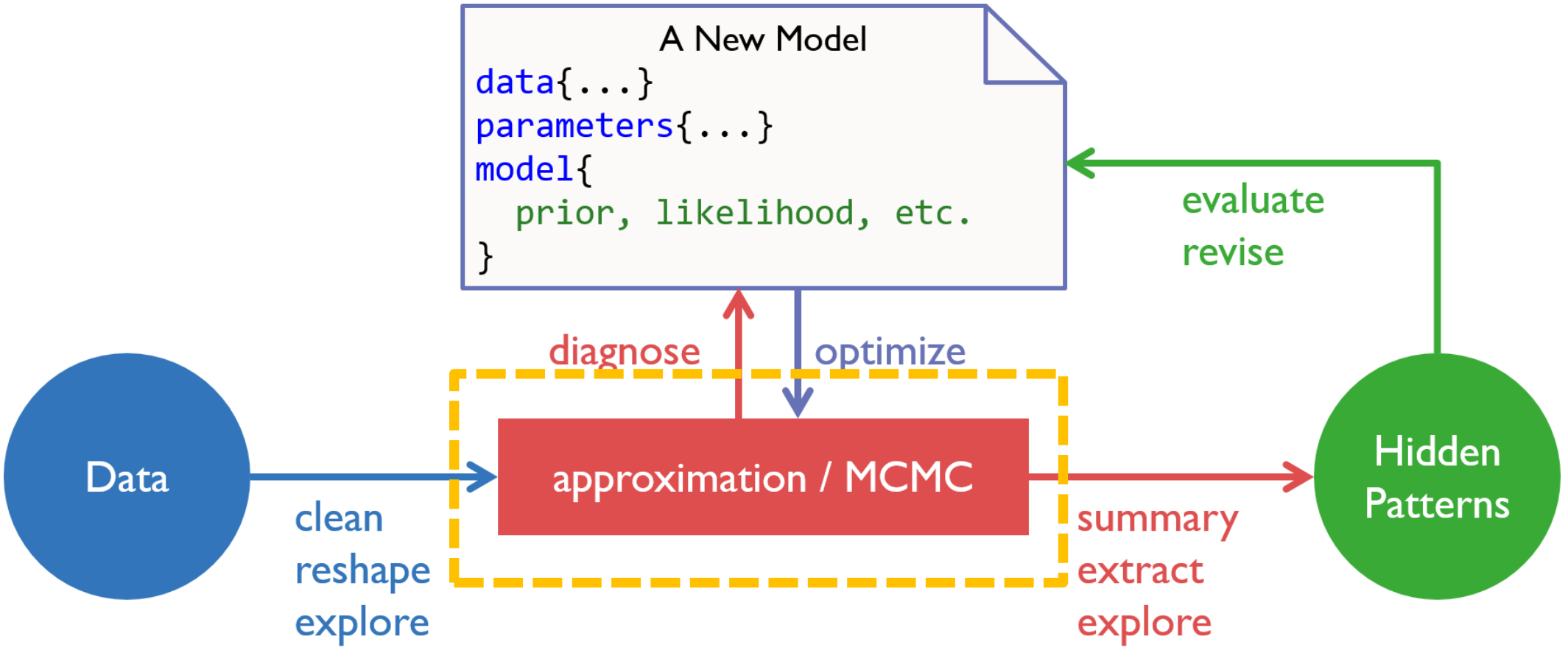
$$p(\theta \mid D) \propto p(D \mid \theta)p(\theta)$$

- solution: Markov Chain Monte Carlo

MARKOV CHAIN MONTE CARLO



cognitive model
statistics
computing



Solving the Problem by Approximation

cognitive model
statistics
computing

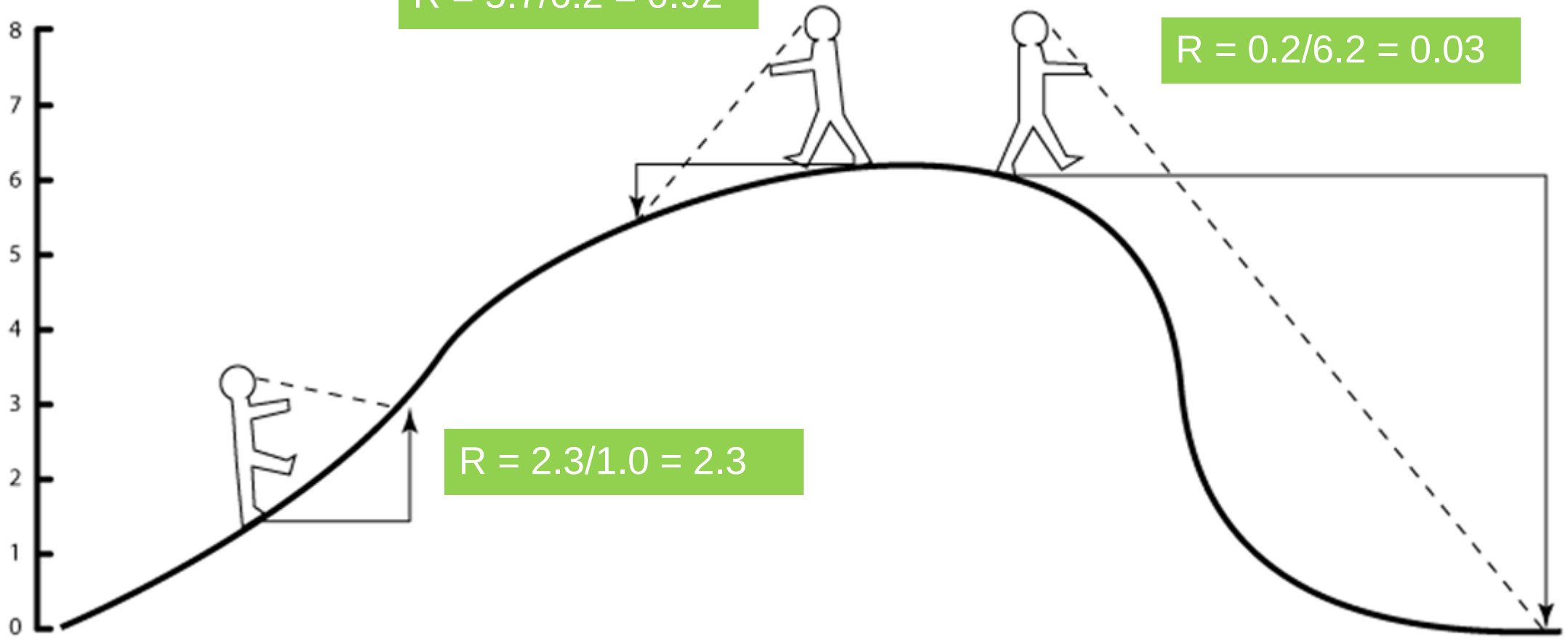
$$p(\theta \mid D) \propto p(D \mid \theta)\, p(\theta)$$

Deterministic Approximation	Stochastic Approximation
→ Variational Bayes	→ Sampling Methods

An MCMC Robot

cognitive model
statistics
computing

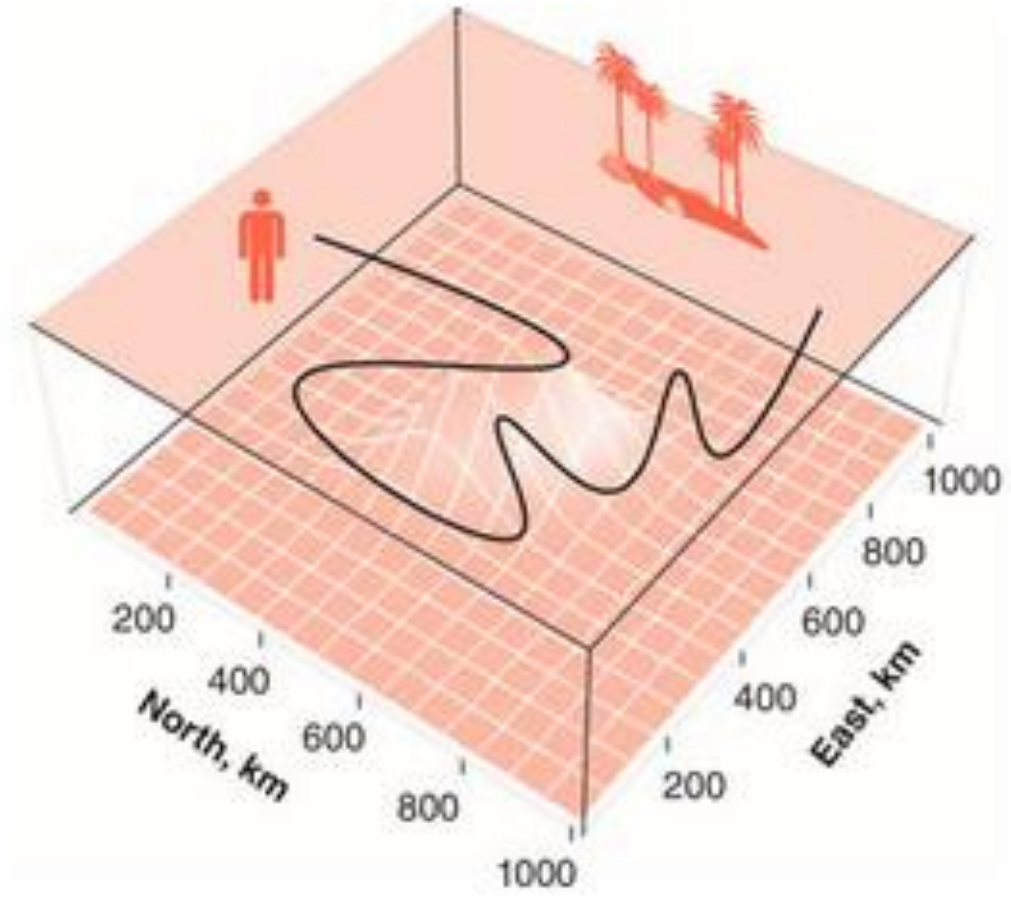
$$p(\theta \mid D) \propto p(D \mid \theta) p(\theta)$$



http://marple.eeb.uconn.edu/mcmcrobot/

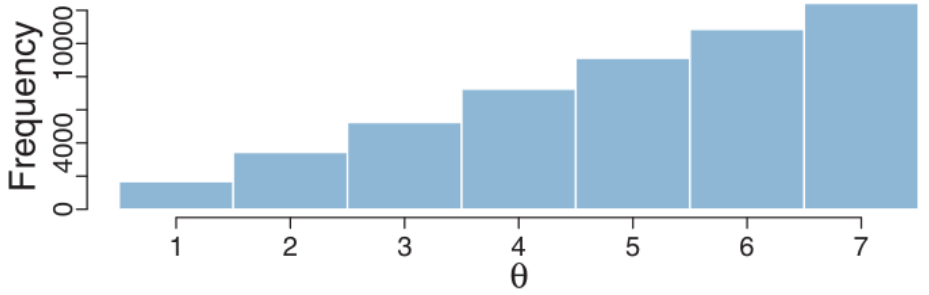
An MCMC Robert in 3D

cognitive model
statistics
computing

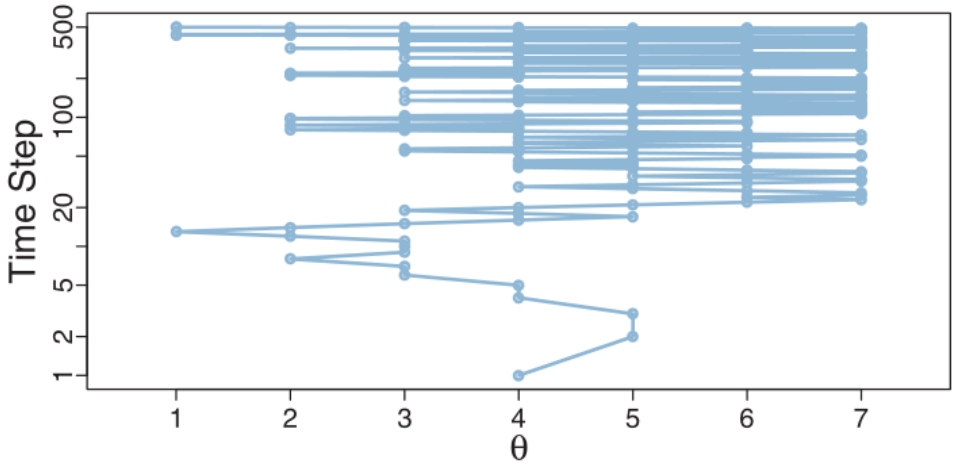


Lambert (2018)

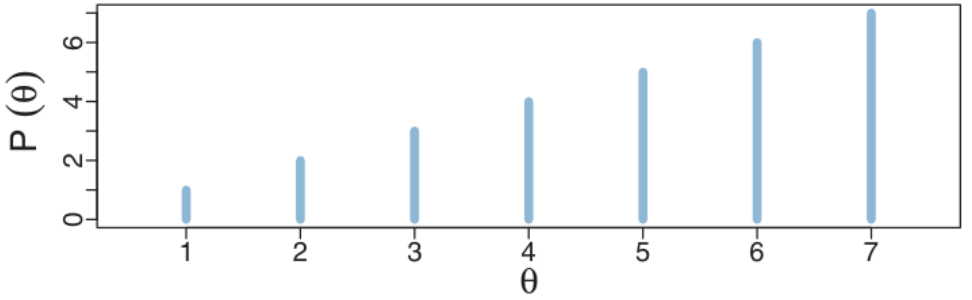
Sampling Example: Discrete



MCMC summary



MCMC trace

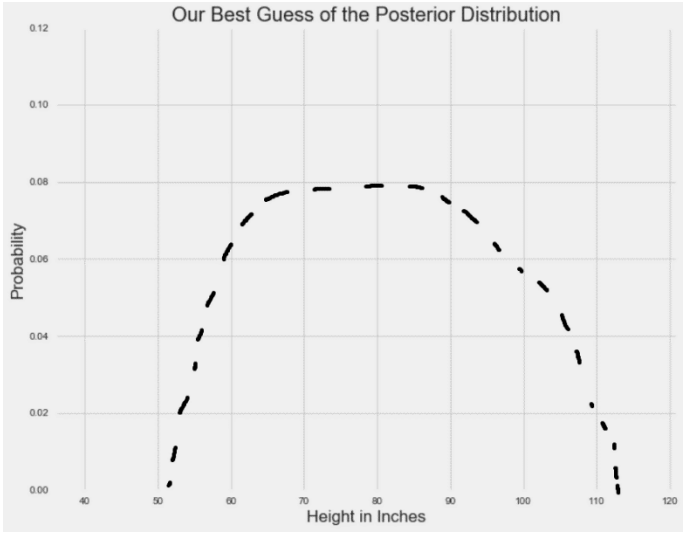


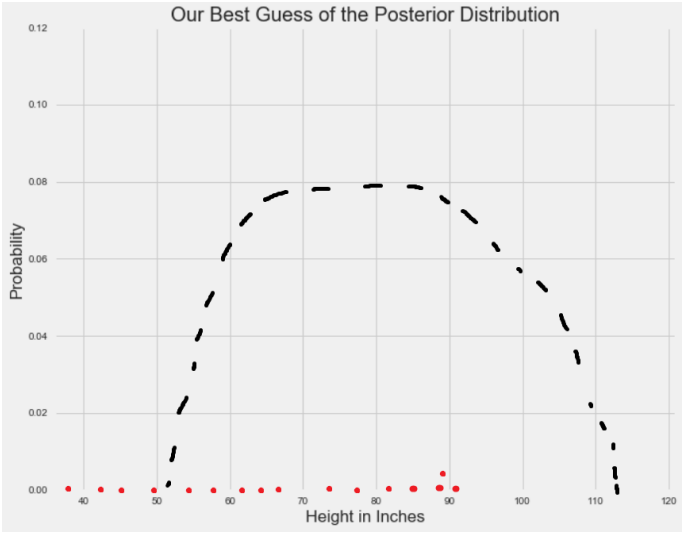
True distribution

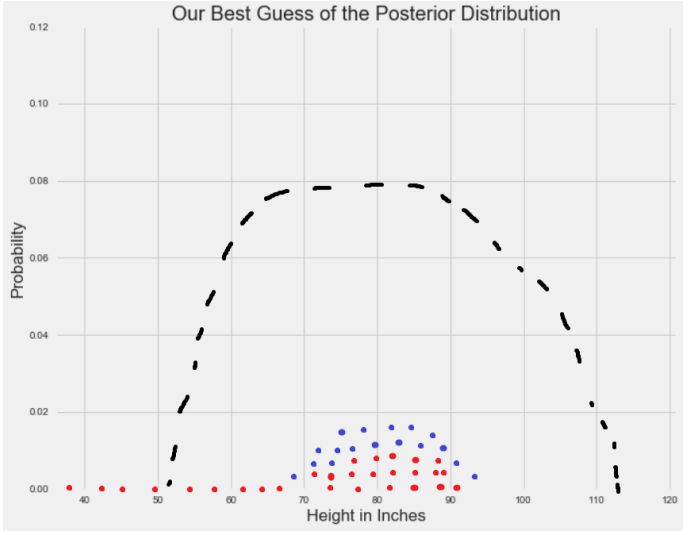
cognitive model
statistics
computing

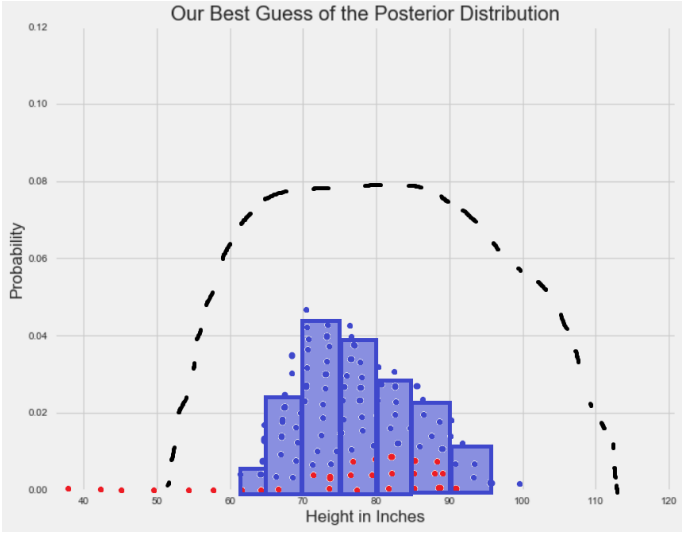
Kruschke (2015)

Sampling Example: Continuous









cognitive model

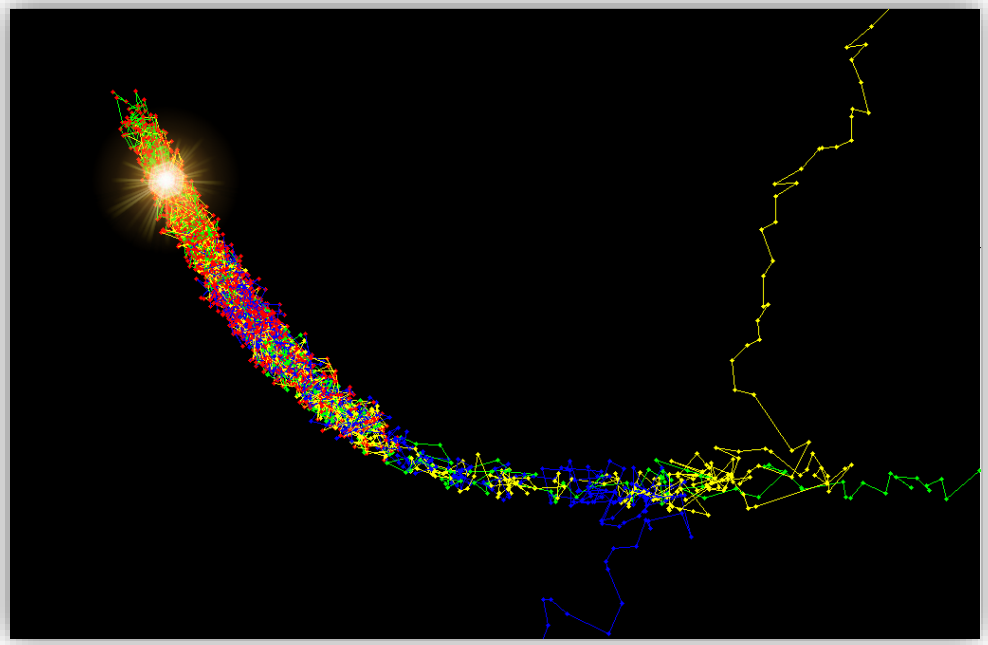
statistics

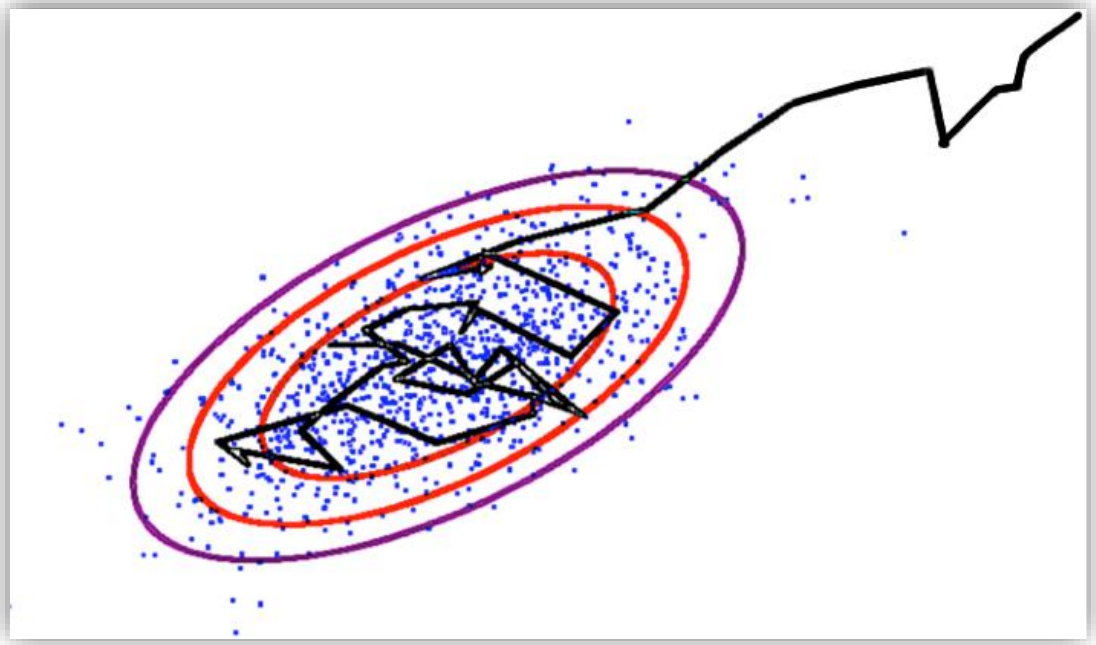
computing

[A Zero-Math Introduction to Markov Chain Monte Carlo Methods]

Visual Example

cognitive model
statistics
computing





Let's watch a video!

cognitive model
statistics
computing

Credit: Thomas Wiecki

MCMC Sampling Algorithms

cognitive model
statistics
computing

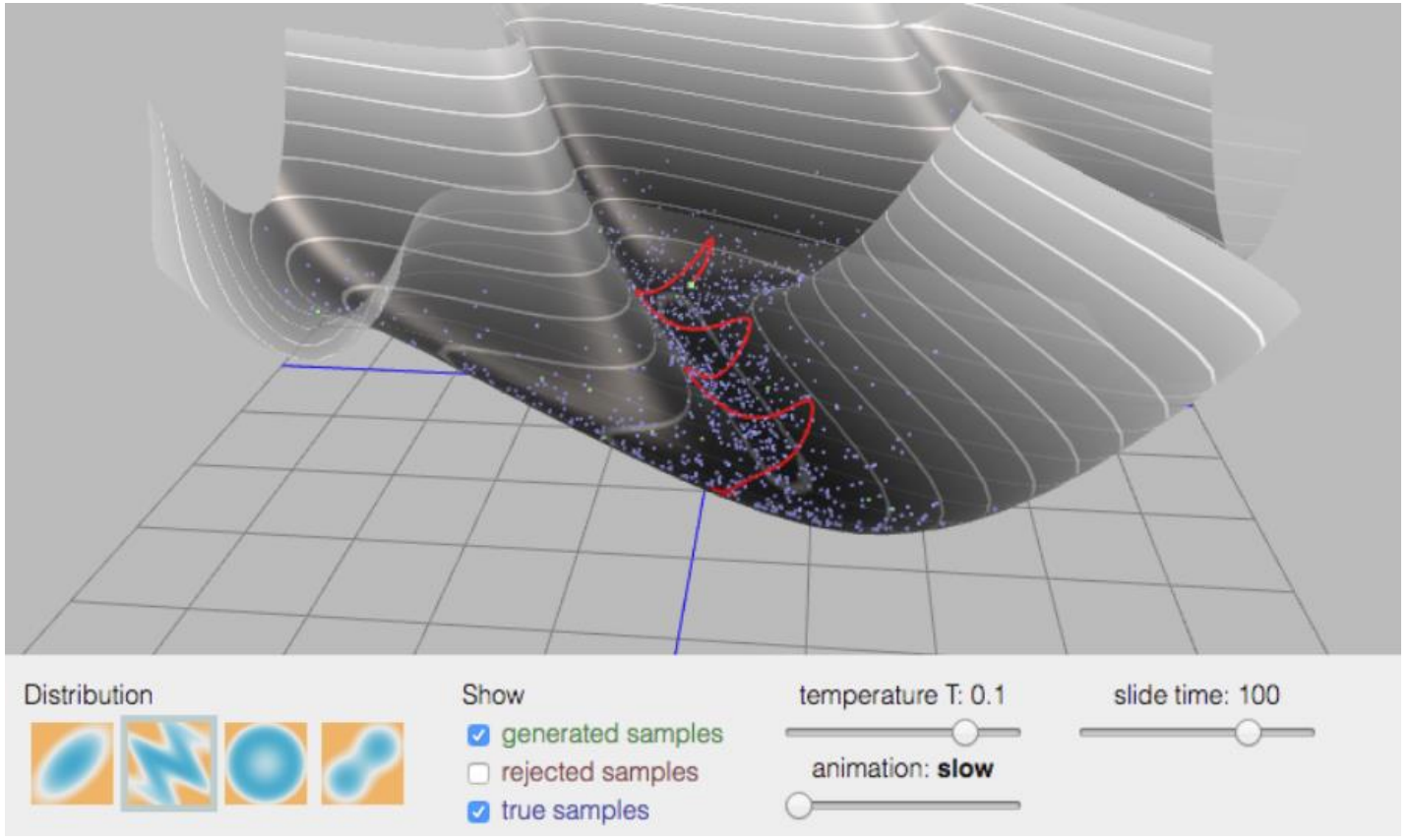
- Rejection sampling
- Importance sampling
- Metropolis algorithm
- Gibbs sampling → JAGS
- HMC sampling*

Stan!

*: also PyMC3

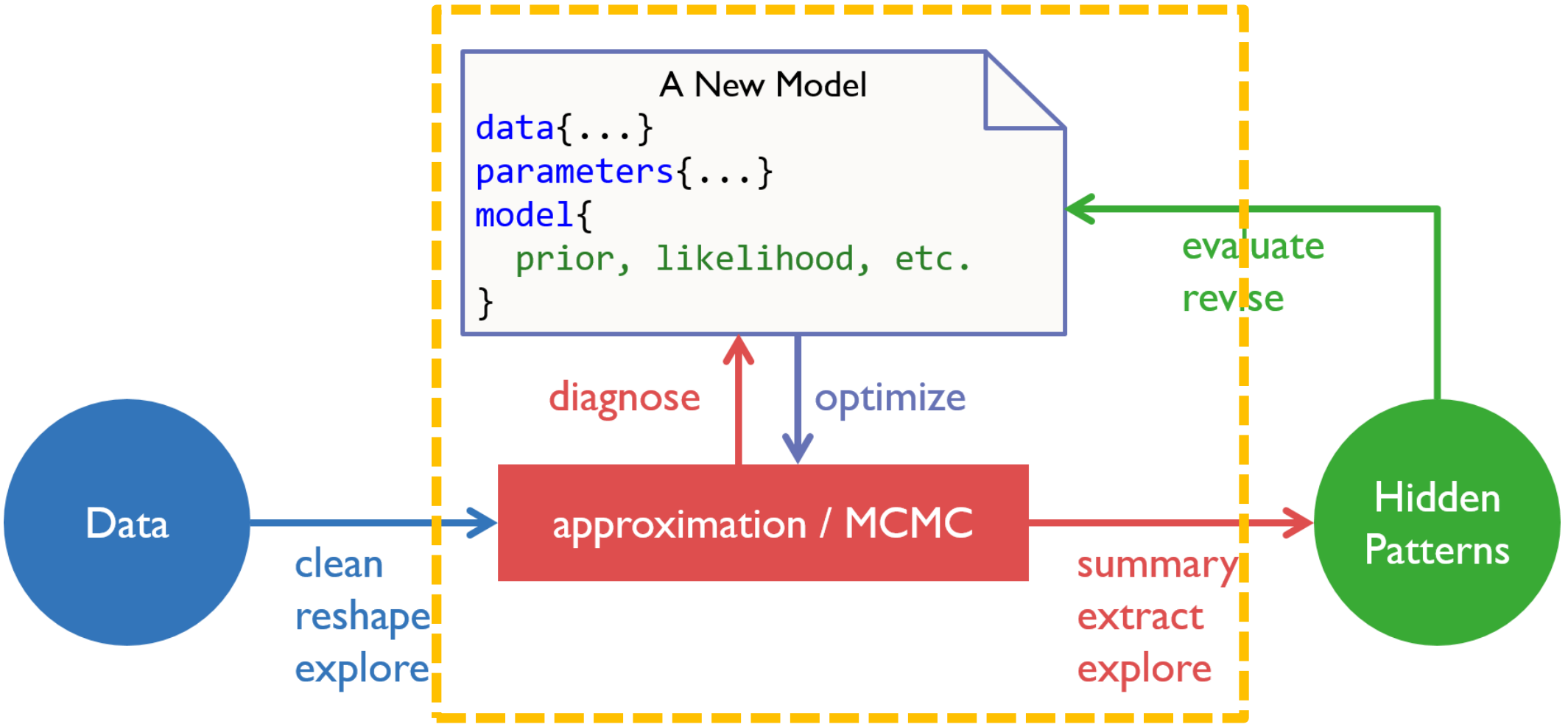
build some intuition

cognitive model
statistics
computing



https://arogozhnikov.github.io/2016/12/19/markov_chain_monte_carlo.html

STAN PROGRAMMING LANGUAGE I


cognitive model
statistics
computing
A New Model
data{...}
parameters{...}
model{
prior, likelihood, etc.
}
evaluate
revise
diagnose
optimize
Data
clean
reshape
explore
approximation / MCMC
summary
extract
explore
Hidden
Patterns

Is Stan popular?

2021 Summer

Stan: A Probabilistic Programming Language

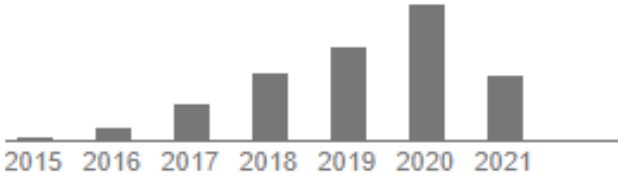
Carpenter, Bob; Gelman, Andrew; Hoffman, Matthew D.; Lee, Daniel; Goodrich, Ben; Betancourt, Michael; Brubaker, Marcus A.; Guo, Jiqiang; Li, Peter; Riddell, Allen

Grantee Submission, Journal of Statistical Software v76 n1 p1-32 Jan 2017

Stan is a probabilistic programming language for specifying statistical models. A Stan program imperatively defines a log probability function over parameters conditioned on specified data and constants. As of version 2.14.0, Stan provides full Bayesian inference for continuous-variable models through Markov chain Monte Carlo methods such as the No-U-Turn sampler, an adaptive form of Hamiltonian Monte Carlo sampling. Penalized maximum likelihood estimates are calculated using optimization methods such as the limited memory Broyden-Fletcher-Goldfarb-Shanno algorithm. Stan is also a platform for computing log densities and their gradients and Hessians, which can be used in alternative algorithms such as variational Bayes, expectation propagation, and marginal inference using approximate integration. To this end, Stan is set up so that the densities, gradients, and Hessians, along with intermediate quantities of the algorithm such as acceptance probabilities, are easily accessible. Stan can be called from the command line using the "cmdstan" package, through R using the "rstan" package, and through Python using the "pystan" package. All three interfaces support sampling and optimization-based inference with diagnostics and posterior analysis. "rstan" and "pystan" also provide access to log probabilities, gradients, Hessians, parameter transforms, and specialized plotting.

Descriptors: Programming Languages, Probability, Bayesian Statistics, Monte Carlo Methods, Statistical Inference, Maximum Likelihood Statistics, Computation, Statistical Distributions, Computer Software

Cited by 3989



[PDF] **JAGS**: A program for analysis of Bayesian graphical models using Gibbs sampling

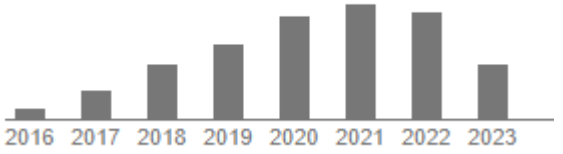
M Plummer - Proceedings of the 3rd international workshop on ..., 2003 - ci.tuwien.ac.at

JAGS is a program for Bayesian Graphical modelling which aims for compatibility with Classic BUGS. The program could eventually be developed as an R package. This article explains the motivations for this program, briefly describes the architecture and then ...

☆ 99 Cited by 4543 Related articles All 8 versions »

2023 Summer

Cited by 6594



brms: An R package for Bayesian multilevel **models** using Stan

PC Bürkner - Journal of statistical software, 2017 - jstatsoft.org

... The **brms** package does not fit **models** itself but uses Stan on the back-end. Accordingly, all samplers implemented in Stan can be used to fit **brms models**. Currently, these are the static ...

☆ Save 99 Cite Cited by 5534 Related articles All 29 versions »

[PDF] **JAGS**: A program for analysis of Bayesian graphical models using Gibbs sampling

M Plummer - Proceedings of the 3rd international workshop on ..., 2003 - r-project.org

... **JAGS** is a program for Bayesian Graphical modelling which aims for compatibility ... **JAGS** and so avoid having to write a new program for each application. A second motivation for **JAGS** ...

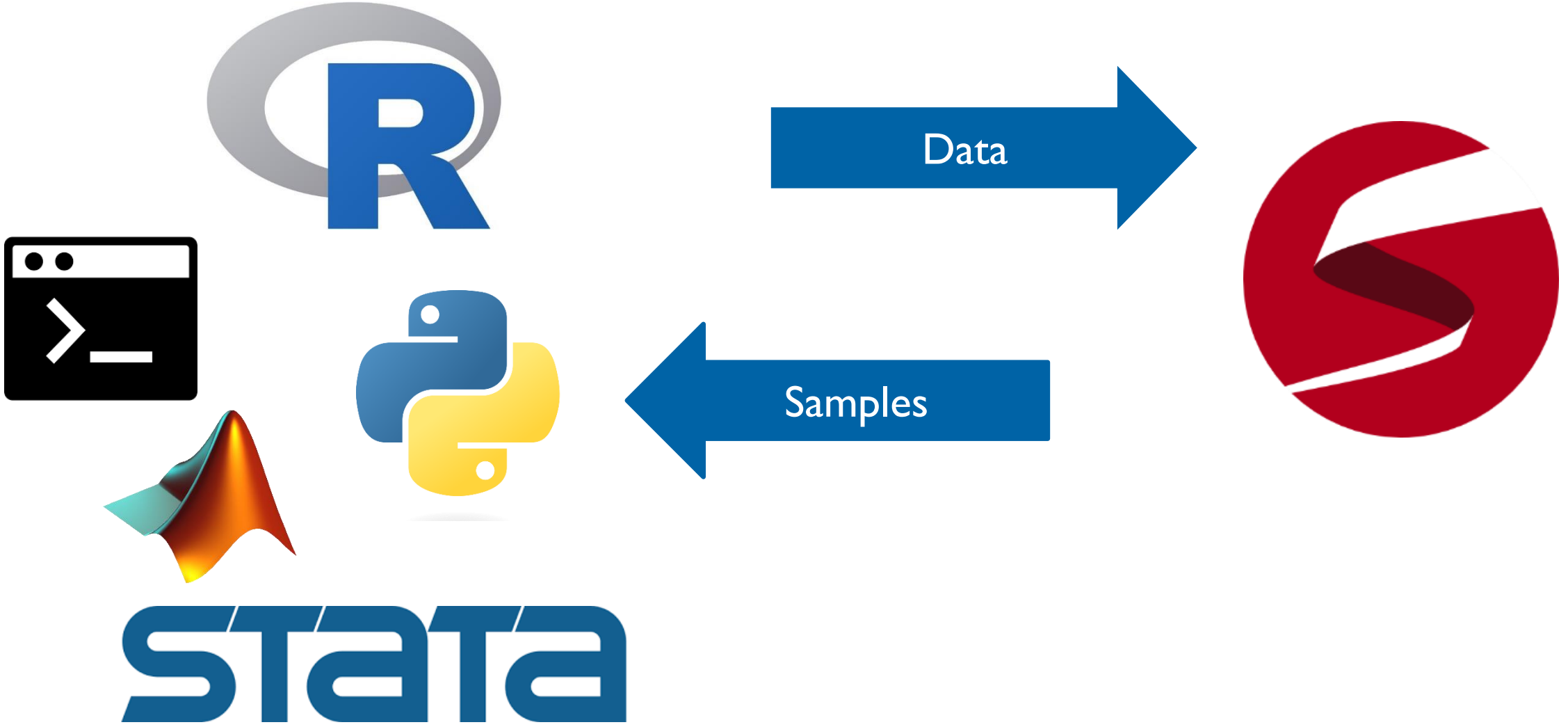
☆ Save 99 Cite Cited by 6277 Related articles All 7 versions »

Who are using Stan?

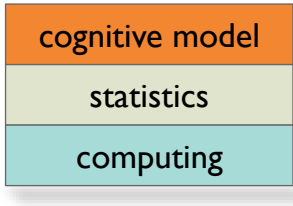
retrieved in 05/2021, Web of Science

Stan and Other Platforms

cognitive model
statistics
computing



Steps of Bayesian Modeling, with Stan



A data story	Think about how the data might arise. It can be *descriptive* or even *causal*. Write a Stan program (*.stan).
Update	Educate your model by feeding it the data. Bayesian Update: update the prior, in light of data, to produce posterior. Run Stan using RStan (PyStan, MatlabStan etc.)
Evaluate	Compare model with reality. Revise your model. Evaluate in RStan and ShinyStan.

McElreath (2016)

Steps of Using Stan

cognitive model
statistics
computing

1. Stan program read into memory
2. Source-to-source transformation into C++
3. C++ compiled and linked (takes a while)
4. Run Stan program
5. Posterior analysis / interface

```
data {
  int<lower=0> N;
  int<lower=0,upper=1> y[N];
}
parameters {
  real<lower=0,upper=1> theta;
}
model {
  y ~ bernoulli(theta);
}
```

Stan Language

model blocks

```
data {
  //... read in external data...
}

transformed data {
  //... pre-processing of data ...
}

parameters {
  //... parameters to be sampled by HMC ...
}

transformed parameters {
  //... pre-processing of parameters ...
}

model {
  //... statistical/cognitive model ...
}

generated quantities {
  //... post-processing of the model ...
}
```

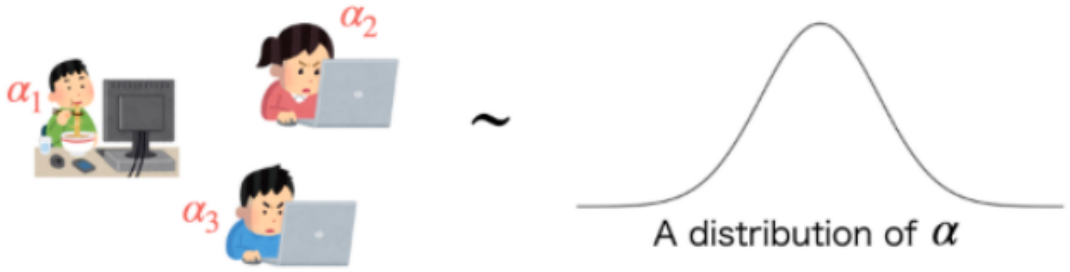
cognitive model
statistics
computing

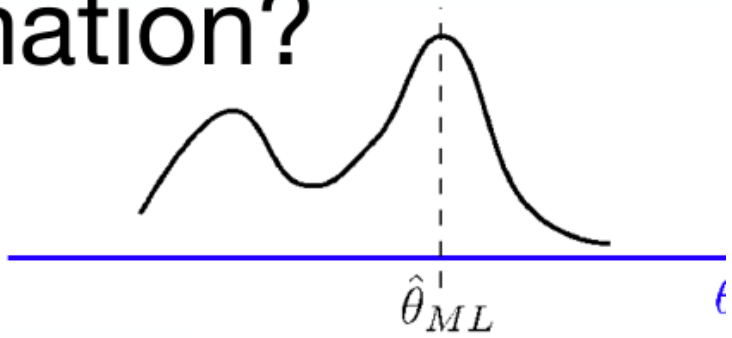
Why Bayesian model estimation?

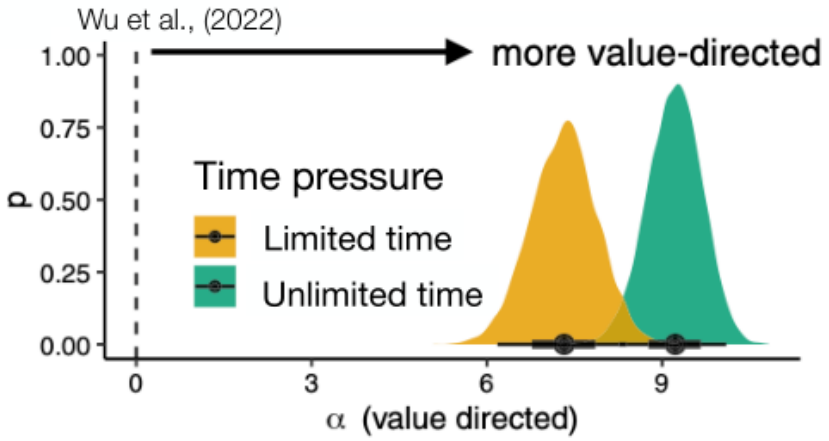
1. Not just a point estimate, but an entire **probability distribution over parameters**

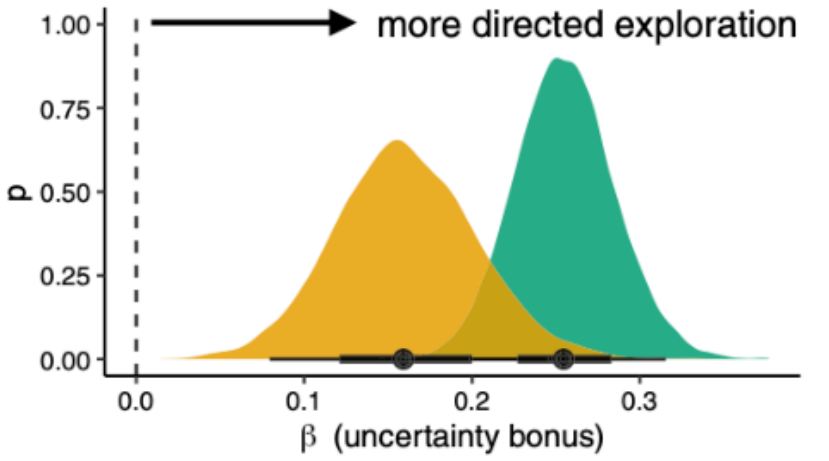
2. Rather than only assuming participants are independent samples, we can model **hierarchical relationships**

3. Naturally avoid overfitting through **Bayesian Occam's Razor**, since we evaluate the model across the entire range of parameters









But, any cons?

- HMC provides huge improvements in computational efficiency, but mathematical foundations are more difficult to follow, at least sometimes.
- Stan cannot sample from the posterior distribution of discrete parameters (e.g., [1, 2]).
  - → with additional effort, it can be achieved through marginalisation, see the Stan User Manual.
- There are a few unwritten tips and tricks.
  - Practise makes perfect and follow discussions on the forum.
- Block-based language → might seem rigid in the first place.

REVISIT BINOMIAL MODEL

Binomial Model

cognitive model
statistics
computing

W L W W W L W L W

$$p(w \mid N, \theta) = \binom{N}{w} \theta^{w} (1-\theta)^{N-w}$$

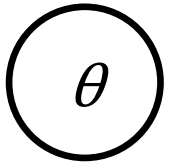
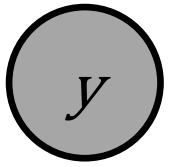
$$w \sim \text{Binomial}(N, \theta)$$

reads as:
w is distributed as a binomial distribution, with number of trials *N*, and success rate ϑ.

McElreath (2016)

Graphical Model Notations

cognitive model
statistics
computing

	continuous	discrete
unobserved		
observed		

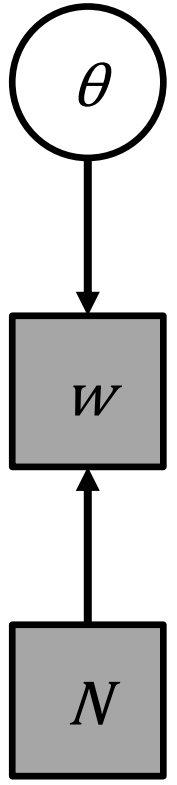
Lee & Wagenmakers (2013)

Binomial Model

cognitive model
statistics
computing

W L W W W L W L W

$$p(w \mid N, \theta) = \binom{N}{w} \theta^{w} (1-\theta)^{N-w}$$



$$\theta \sim \text{Uniform}(0, 1)$$

$$w \sim \text{Binomial}(N, \theta)$$

	continuous	discrete
unobserved	θ	δ
observed	y	N

McElreath (2016)

Binomial Model

cognitive model
statistics
computing

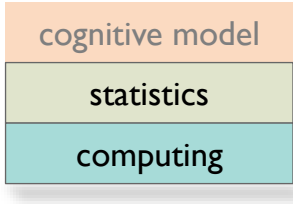
W L W W W L W L W

$$p(w \mid N, \theta) = \binom{N}{w} \theta^{w} (1-\theta)^{N-w}$$

```
data {
    int<lower=0> w;
    int<lower=0> N;
}

parameters {
    real<lower=0,upper=1> theta;
}

model {
    w ~ binomial(N, theta);
}
```

McElreath (2016)

Running Binomial Model with Stan




```
…/BayesCog/02.binomial_globe/_scripts/binomial_globe_main.R
```

```
> R.version
R version 3.5.1 (2018-07-02)

> stan_version()
[1] "2.18.0"
```

Model Summary

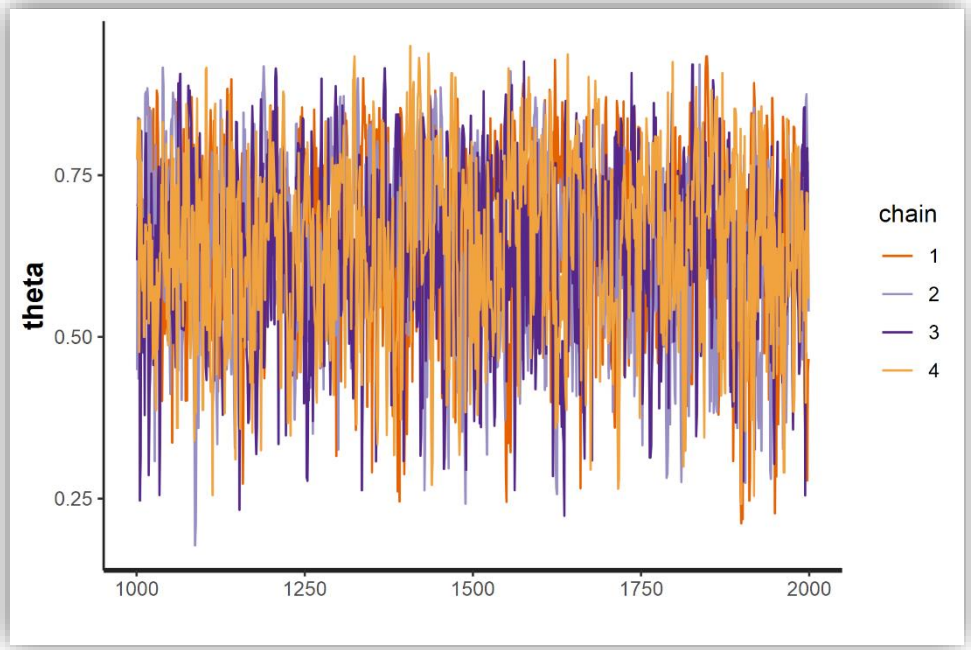
cognitive model
statistics
computing

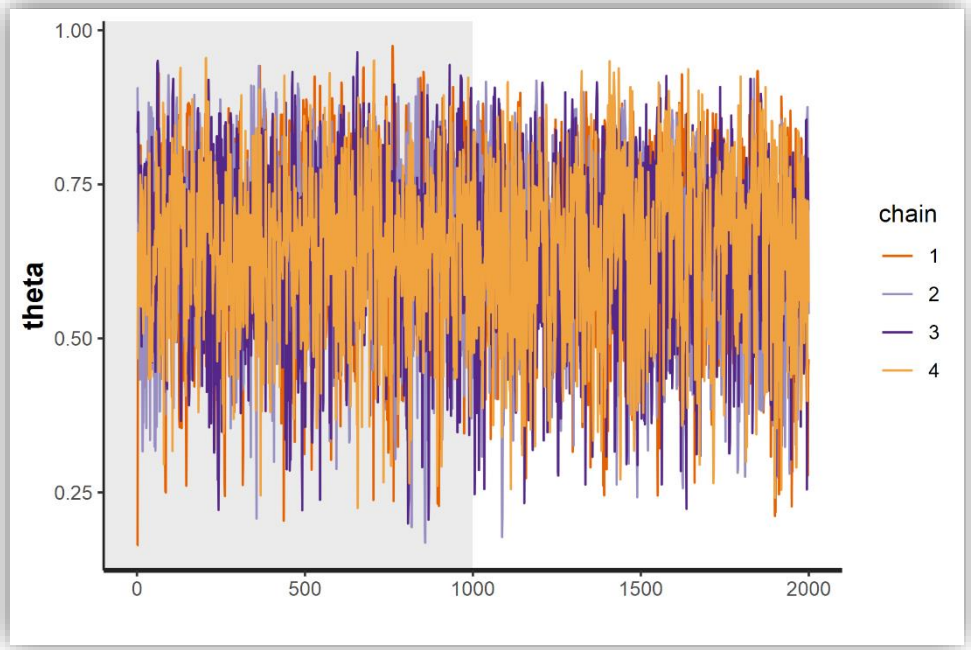
```
> print(fit_globe)
Inference for Stan model: binomial_globe_model.
4 chains, each with iter=2000; warmup=1000; thin=1;
post-warmup draws per chain=1000, total post-warmup draws=4000.

       mean se_mean   sd  2.5%   25%   50%   75% 97.5% n_eff Rhat
theta  0.64    0.00 0.14  0.35  0.54  0.65  0.74  0.87  1278    1
lp__  -7.72    0.02 0.69 -9.77 -7.89 -7.46 -7.27 -7.21  1824    1

Samples were drawn using NUTS(diag_e) at Tue Apr 09 12:44:04 2019.
For each parameter, n_eff is a crude measure of effective sample size,
and Rhat is the potential scale reduction factor on split chains (at
convergence, Rhat=1).
```

Gelman-Rubin convergence diagnostic
(Gelman & Rubin, 1992)

Diagnostics - traceplot

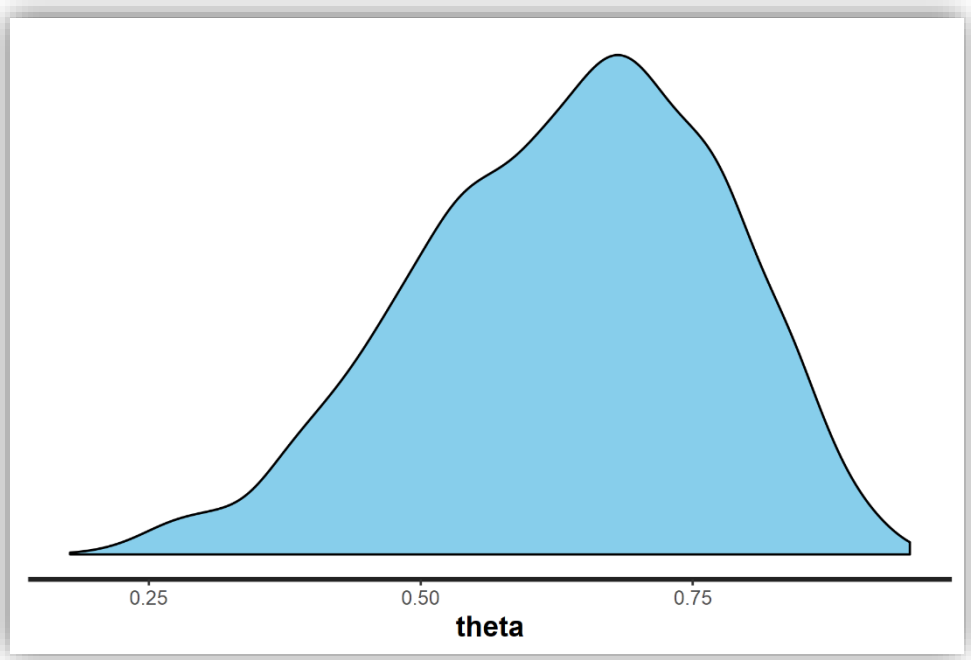
cognitive model
statistics
computing

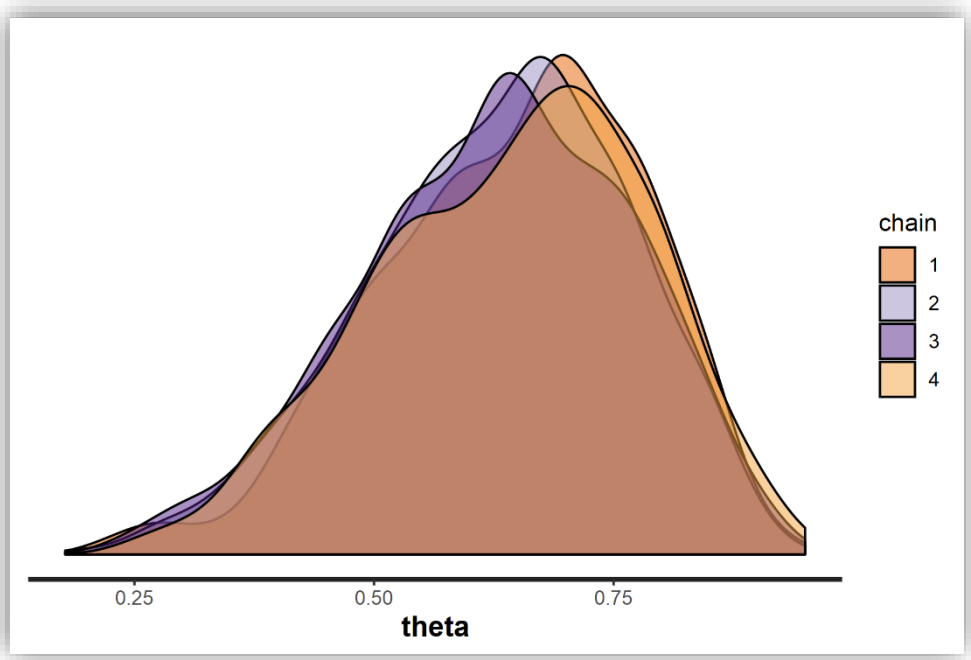




Diagnostics - density

cognitive model
statistics
computing

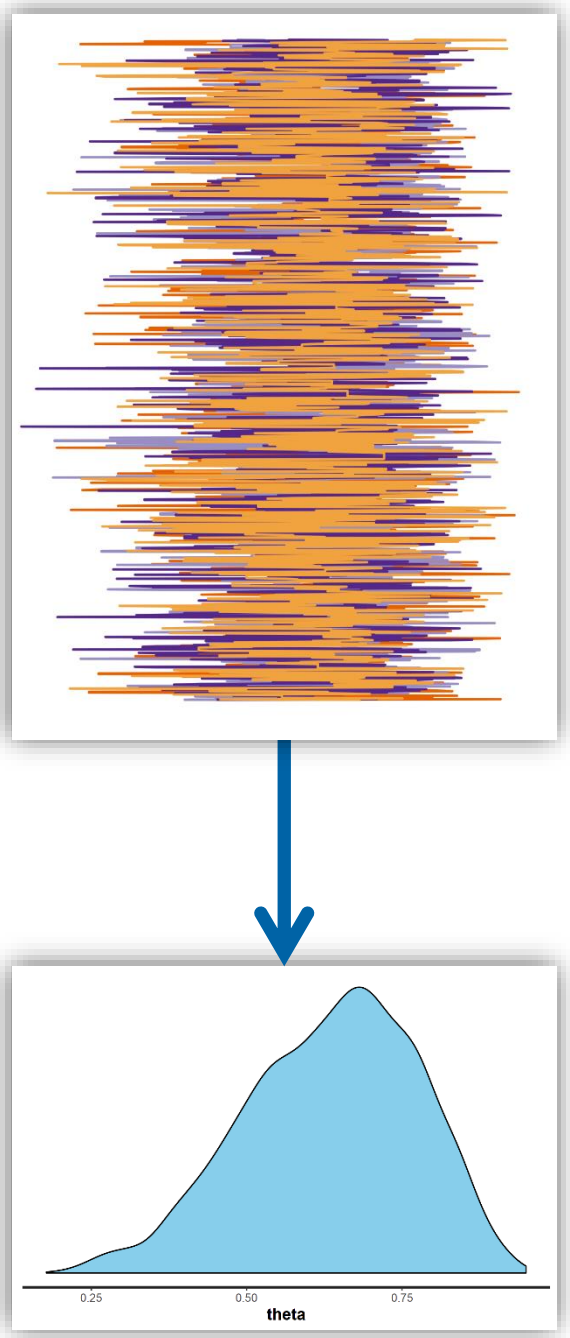




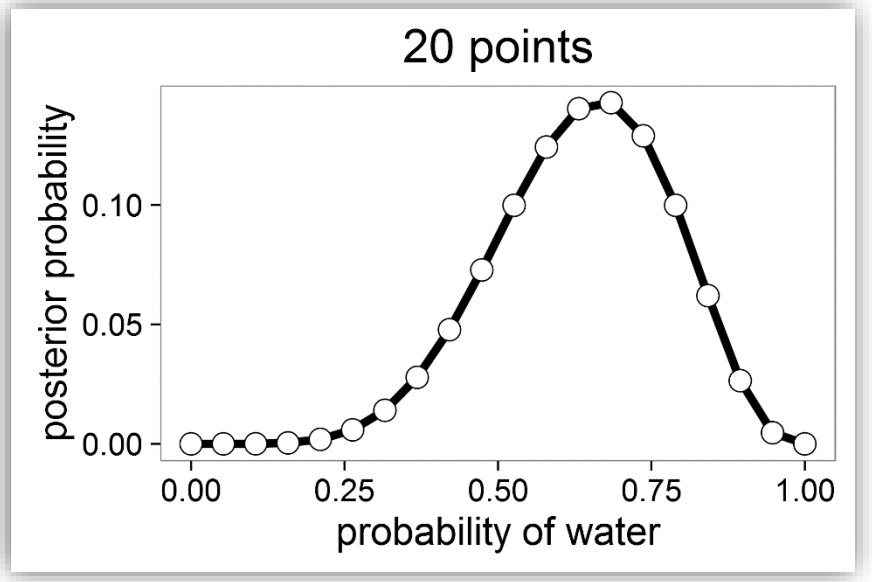
Diagnostics

cognitive model
statistics
computing

MCMC

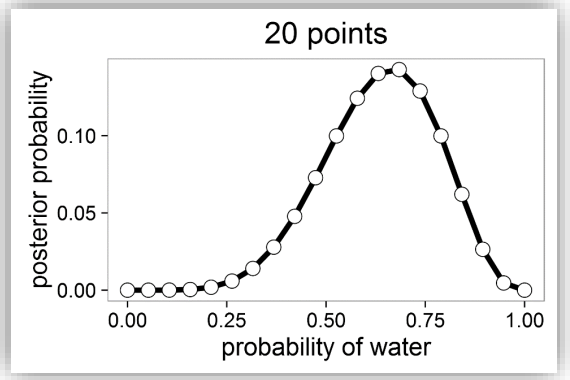


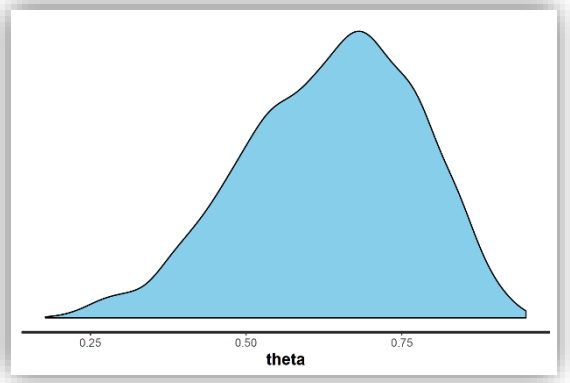
Grid Approximation



Draw a Conclusion?

- W = 6 out of N = 9
- uncertainty (relative plausibility) of all ϑ values
- the relative plausibility of ϑ = 0.64 is the highest, but it never rules out the possibility of ϑ being other values, e.g., 0.5, 0.75
- → when ϑ = 0.5, you may still observe 6W / 9 trials





Is Anything Missing? – NO

cognitive model
statistics
computing

```
data {
    int<lower=0> w;
    int<lower=0> N;
}

parameters {
    real<lower=0,upper=1> theta;
}

model {
    theta ~ uniform(0,1);
    w ~ binomial(N, theta);
}
```

```
data {
    int<lower=0> w;
    int<lower=0> N;
}

parameters {
    real<lower=0,upper=1> theta;
}

model {
    w ~ binomial(N, theta);
}
```

STAN PROGRAMMING LANGUAGE II

Why Use Stan?

cognitive model
statistics
computing

vs. BUGS / JAGS

- Less spatial correlation → effective samples
- Time to converge and per effective sample size:

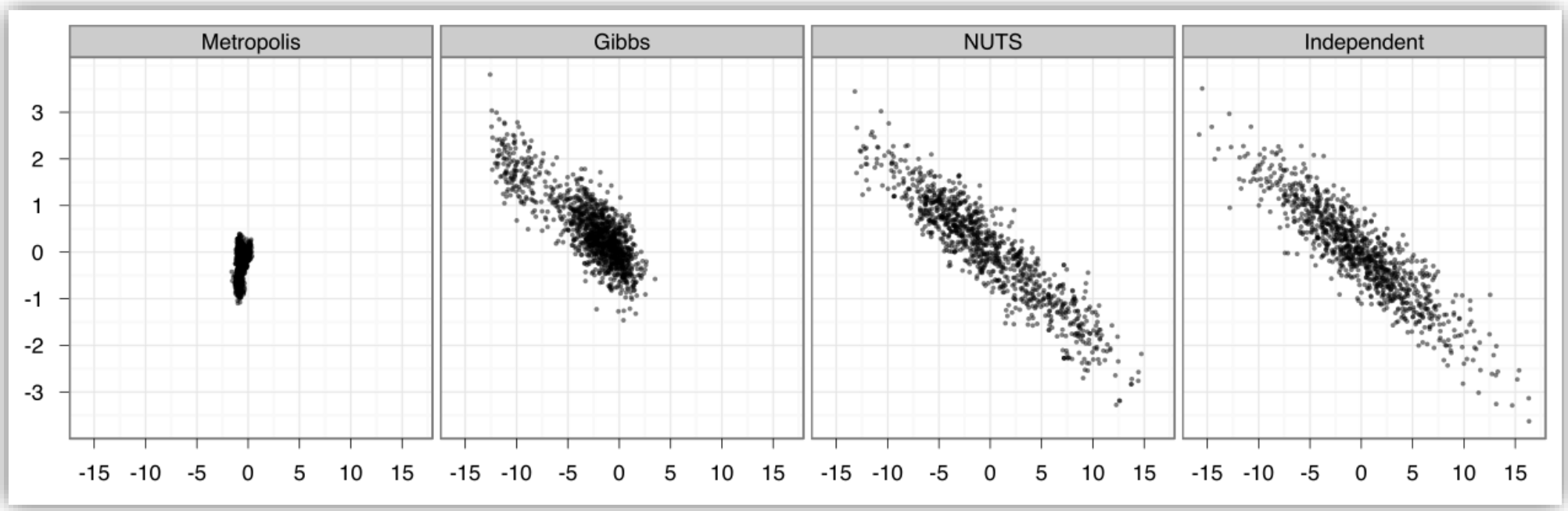
  1 - ∞ times faster
- Memory usage: 1–10%
- Language features
  - variable overwrite: `a = 4, then a = 5`
  - formal control flow (same as R)
  - full support of vectorizing & matrix calculation

http://mc-stan.org/
https://discourse.mc-stan.org/t/selling-stan/3693/6

NUTS vs. Gibbs and Metropolis

cognitive model
statistics
computing

Hamilton MC (HMC) implements No-U-Turn Sampler (NUTS)



- Two dimensions of highly correlated 250-dim normal
- 1,000,000 draws from Metropolis and Gibbs (thin to 1000)
- 1,000 draws from NUTS; 1000 independent draws

http://mc-stan.org/

General Properties of Stan Language

cognitive model
statistics
computing

- Whitespace does not matter
- Comments
  - //
  - /* … */
- Must use semicolon (;)
- Variables are typed and scoped

Variable Declaration

- Each variable has a type (static type; scalar, vector, matrix etc.)
- Only values of that type can be assigned to the variable
  - e.g. cannot assign [1 2 3] to a (declared as a scalar)
- Declaration of variables happen at the top of a block (including local blocks)

Variable's Scope

cognitive model
statistics
computing

	data	transformed data	parameters	transformed parameters	model	generated quantities
Variable Declarations	Yes	Yes	Yes	Yes	Yes	Yes
Variable Scope	Global	Global	Global	Global	Local	Local
Variables Saved?	No	No	Yes	Yes	No	Yes
Modify Posterior?	No	No	No	No	Yes	No
Random Variables	No	No	No	No	No	Yes

Scalar Variables

cognitive model
statistics
computing

`real`

- scalar
- continuous

```
data {
  real y;
}
```

`int`

- scalar
- integer
- can't be used in `parameters` or `transformed parameters` blocks

```
data {
  int n;
}
```

Constraining Scalar Variables

cognitive model
statistics
computing

```
data {
  int<lower=1> m;
  int<lower=0,upper=1> n;
  real<lower=0> x;
  real<upper=0> y;
  real<lower=-1,upper=1> rho;
}
```

Vector & Matrix


```
vector[3] a;
// column vector

row_vector[4] b;
// row vector

matrix[3,4] A;
// A is a 3x4 matrix
// A[1] returns a 4-element row vector

vector<lower=0,upper=1>[5] rhos;
row_vector<lower=0>[4] sigmas;
matrix<lower=-1, upper=1>[3,4] Sigma;
```

Control Flow

- if-else

```
if (cond) {
    ..statement..
}
```

```
if (cond) {
    ..statement..
} else {
    ..statement..
}
```

```
if (cond) {
    ..statement..
} else if (cond) {
    ..statement..
} else {
    ..statement..
}
```

- for-loop

```
for ( j in 1:J) {
    ..statement..
}
```

```
for ( j in 1:J ) {
    for ( k in 1:K ) {
        ..statement..
    }
}
```

same as the R syntax, but terminate each line with ;

BERNOULLI MODEL

Bernoulli Model

cognitive model
statistics
computing

- You are interested in if a coin is biased.
- You will flip the coin.
- You will record whether it comes up a head (h) or a tail (t).
- You might observe 15 heads out of 20 flips.
- What is your degree of belief about the biased parameter ϑ?

Kruschke (2015)

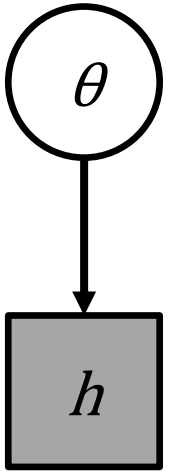
Bernoulli Model

cognitive model
statistics
computing

$$p(w \mid N, p) = \binom{N}{w} p^w (1-p)^{N-w}$$

$N = 1$

$$p(h \mid \theta) = \theta^h (1-\theta)^{1-h}$$



$\theta \sim \text{Uniform}(0, 1)$

$h \sim \text{Bernoulli}(\theta)$

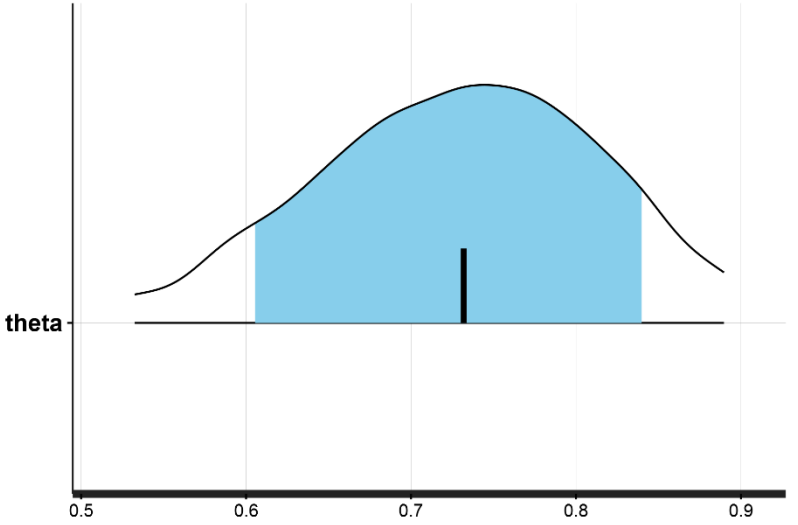
Exercise VIII

cognitive model
statistics
computing

```
…/BayesCog/03.bernoulli_coin/_scripts/bernoulli_coin_main.R
```

TASK: fit the Bernoulli model

```
> dataList
$`flip`
 [1] 1 1 1 0 1 1 1 1 1 0 0 1 1 0 1 1 1 1 0 1

$N
[1] 20
```



Possible Optimization?

cognitive model
statistics
computing

```
model {
  for (n in 1:N) {
    flip[n] ~ bernoulli(theta);
  }
}
```

61.59 secs*

```
model {
  flip ~ bernoulli(theta);
}
```

53.25 secs*

Thinking before looping!

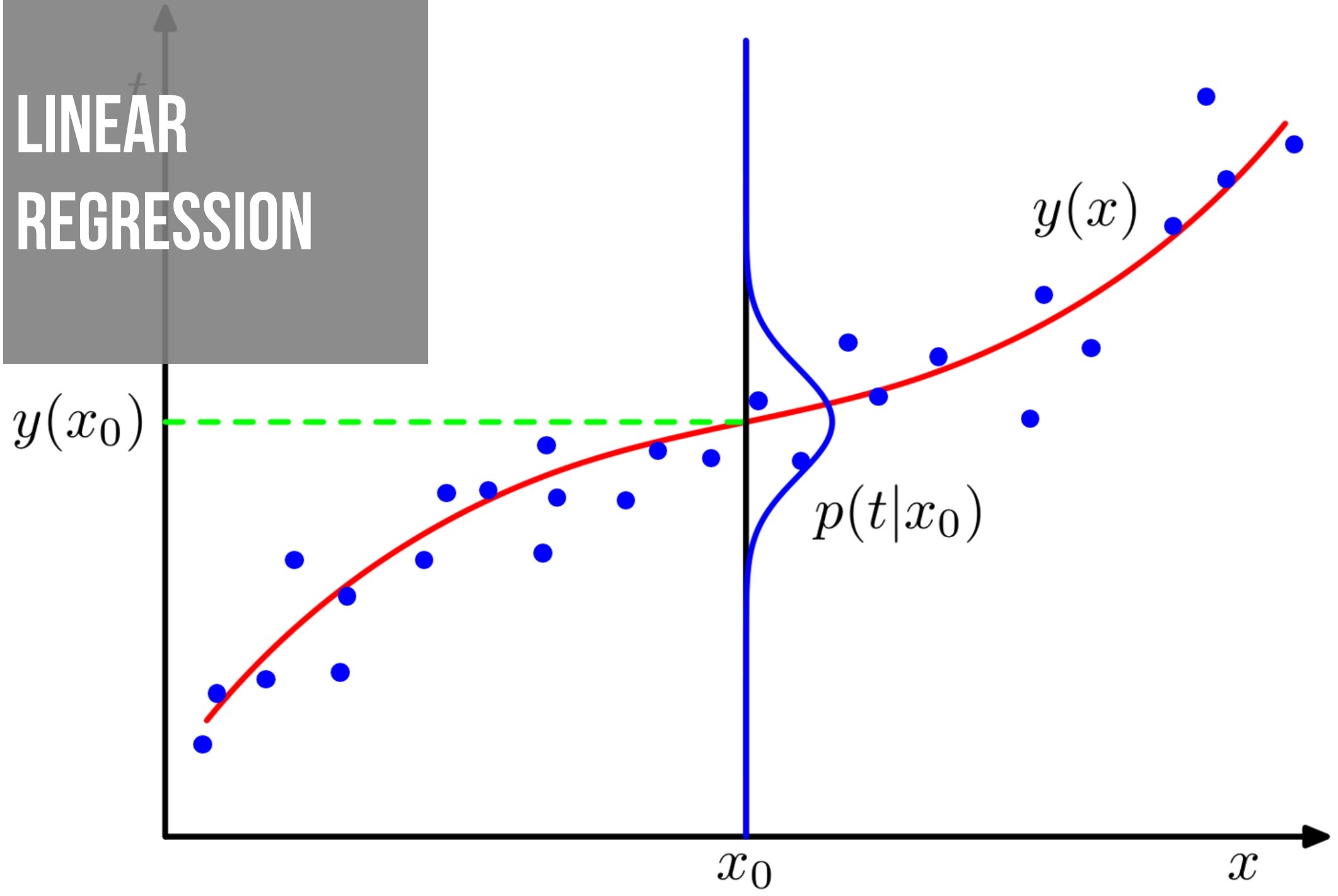
* compiling time included

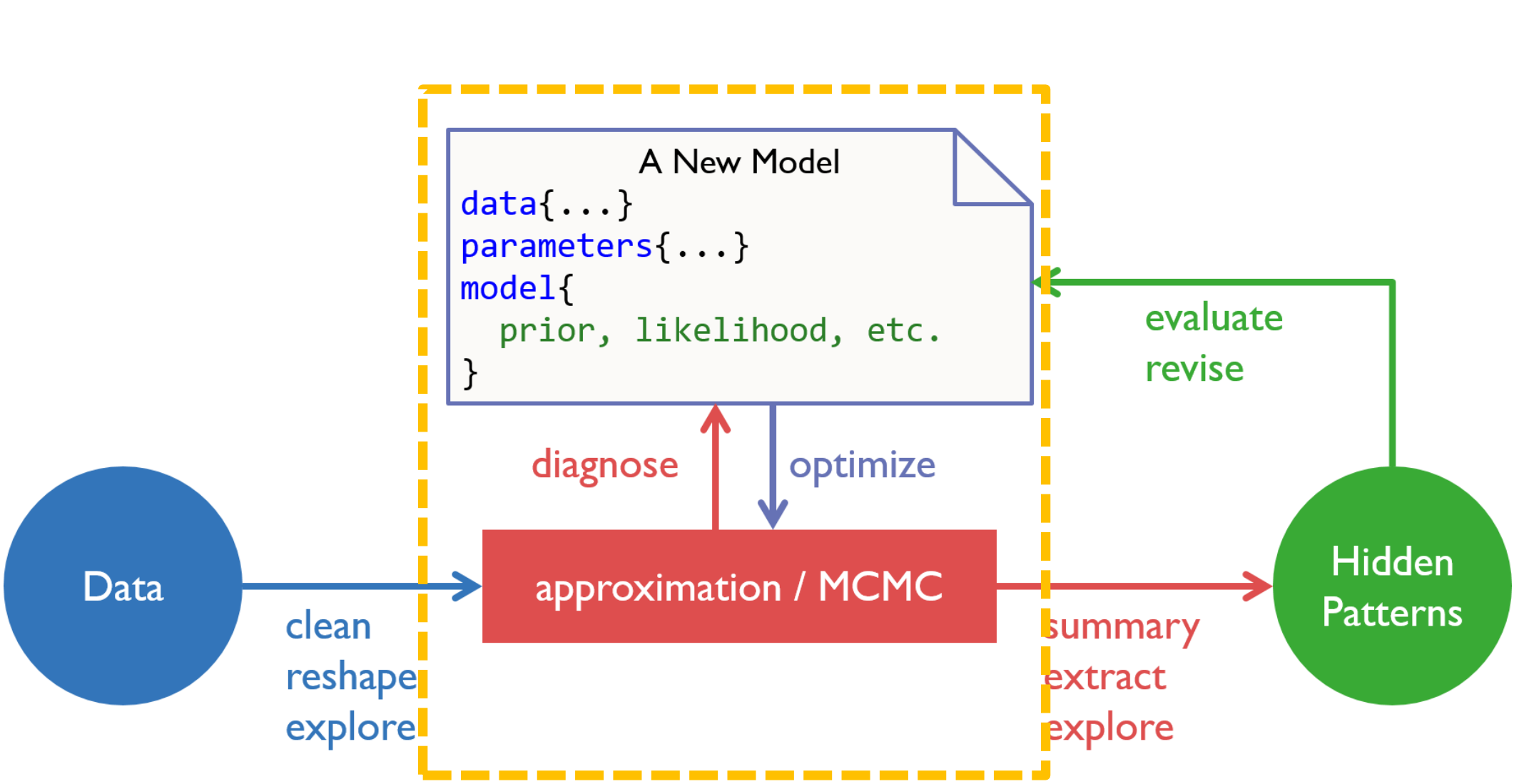
Recap

What we've learned…

- R Basics
- probability distributions
- Bayes' theorem, $p(\theta|D)$
- Binomial model
- MCMC and Stan

LINEAR REGRESSION




cognitive model
statistics
computing
A New Model
data{...}
parameters{...}
model{
prior, likelihood, etc.
}
diagnose
optimize
evaluate
revise
Data
clean
reshape
explore
approximation / MCMC
summary
extract
explore
Hidden
Patterns

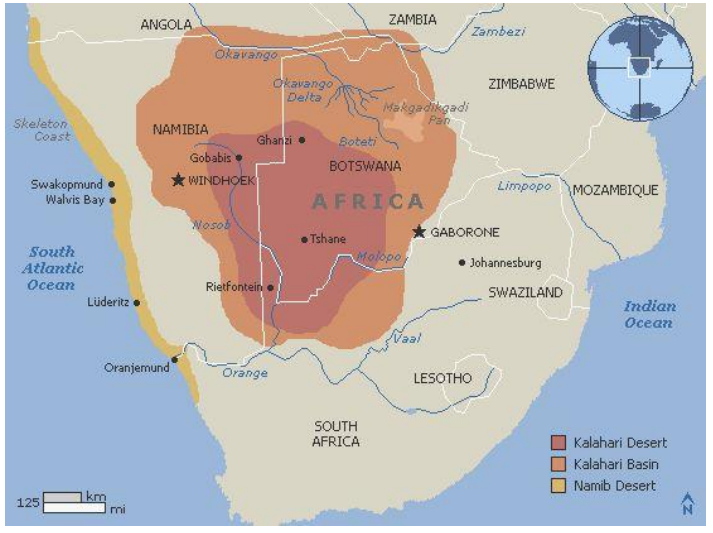
Linear Regression: height ~ weight

cognitive model
statistics
computing

…/04.regression_height/_scripts/regression_height_main.R

make scatter plot and fit the model with `lm()`

```
>load('_data/height.RData')
>d <- Howell1
>d <- d[ d$age >= 18 , ]
>head(d)
height   weight age male
1 151.765 47.82561  63    1
2 139.700 36.48581  63    0
3 136.525 31.86484  65    0
4 156.845 53.04191  41    1
5 145.415 41.27687  51    0
6 163.830 62.99259  35    1
```



McElreath (2016)

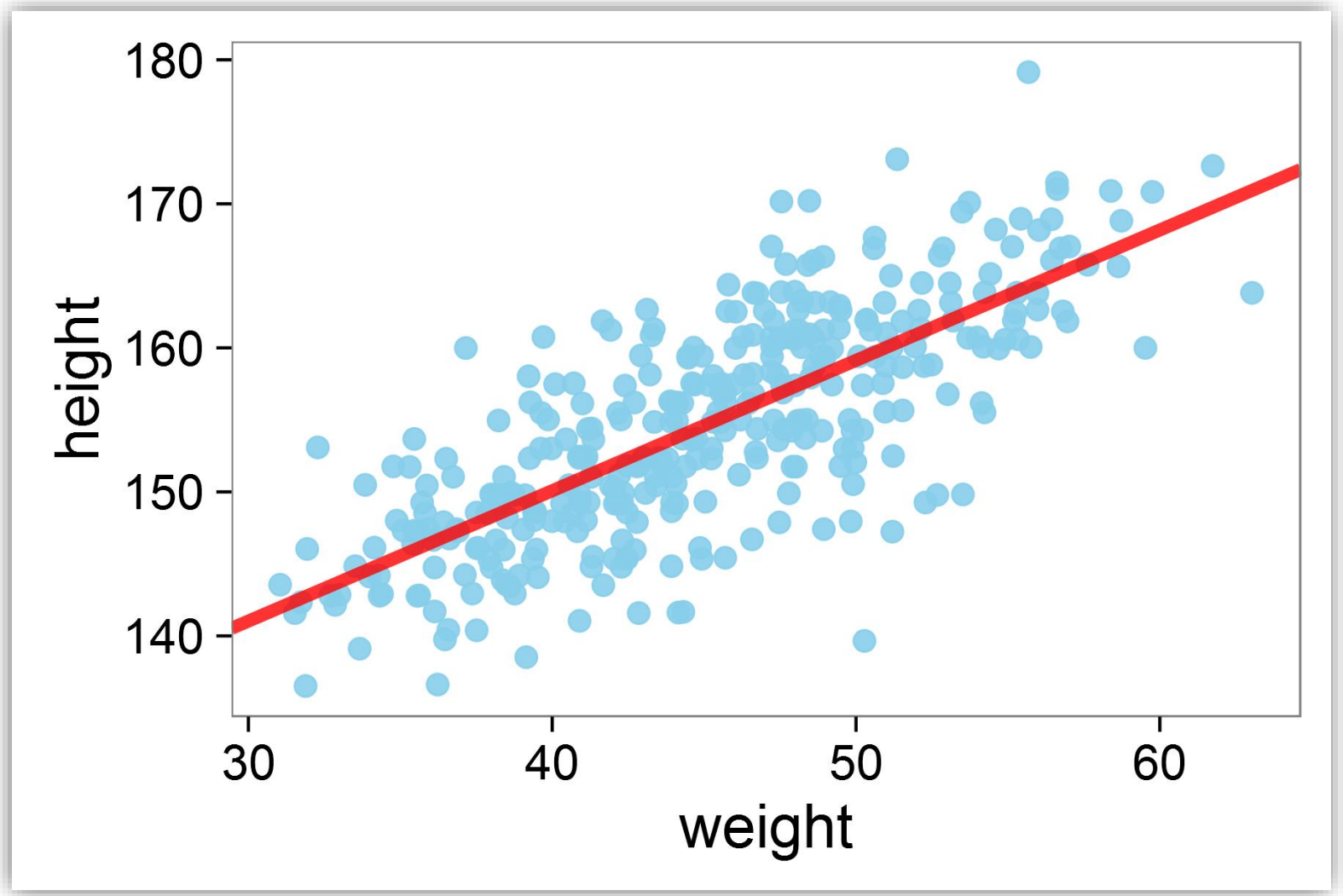
Results with lm()

cognitive model
statistics
computing

```
> L <- lm( height ~ weight, d) # estimate model by minimizing least squares errors
> summary(L)

Call:
lm(formula = height ~ weight, data = d)

Residuals:
     Min       1Q   Median       3Q      Max
-19.7464  -2.8835   0.0222   3.1424  14.7744

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 113.87939    1.91107   59.59   <2e-16 ***
weight        0.90503    0.04205   21.52   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 5.086 on 350 degrees of freedom
Multiple R-squared:  0.5696, Adjusted R-squared:  0.5684
F-statistic: 463.3 on 1 and 350 DF,  p-value: < 2.2e-16
```

height ~ weight

cognitive model
statistics
computing



McElreath (2016)

Rethinking Regression Model

cognitive model
statistics
computing

$$\mu_i = \alpha + \beta x_i$$

~~$$y_i = \mu_i + \varepsilon$$~~

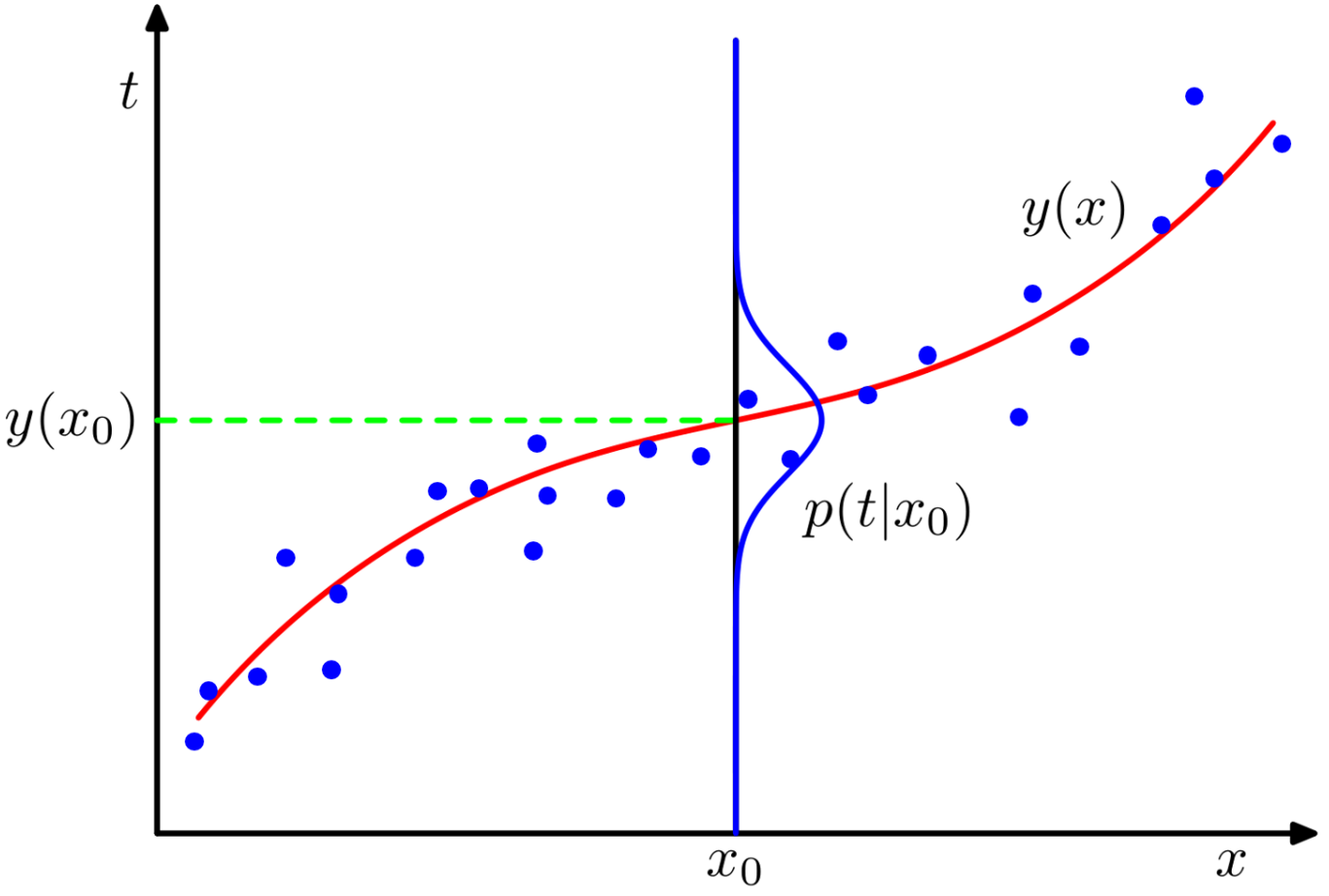
~~$$\varepsilon \sim Normal(0, \sigma)$$~~

$$y_i \sim Normal(\mu_i, \sigma)$$

Rethinking Regression Model

cognitive model
statistics
computing

$$\mu_i = \alpha + \beta x_i$$
$$y_i \sim Normal(\mu_i, \sigma)$$



Bishop (2006)

Demetri
@PhDemetri

I wish linear regression was never taught as

$y = a + bx + e$

and was instead taught as

$y \sim Normal(Xb, sigma)$

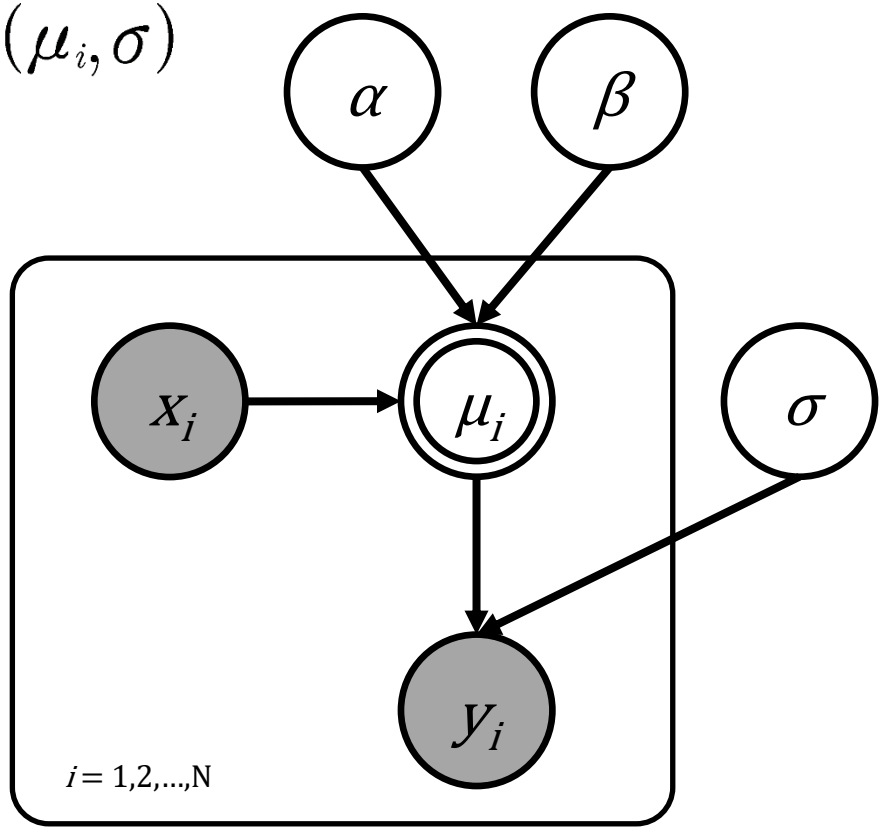
Because then explaining and learning GLM would be easier

1:19 AM · Apr 12, 2021 · Twitter Web App

from model to code

cognitive model
statistics
computing

$$\mu_i = \alpha + \beta x_i$$
$$y_i \sim Normal(\mu_i, \sigma)$$




```
model {
  vector[N] mu;
  for (i in 1:N) {
    mu[i] = alpha + beta * weight[i];
    height[i] ~ normal(mu[i], sigma);
  }
}
```

```
model {
  vector[N] mu;
  mu = alpha + beta * weight;
  height ~ normal(mu, sigma);
}
```

```
model {
  height ~ normal(alpha + beta * weight, sigma);
}
```

deterministic variable

Thinking about Priors?

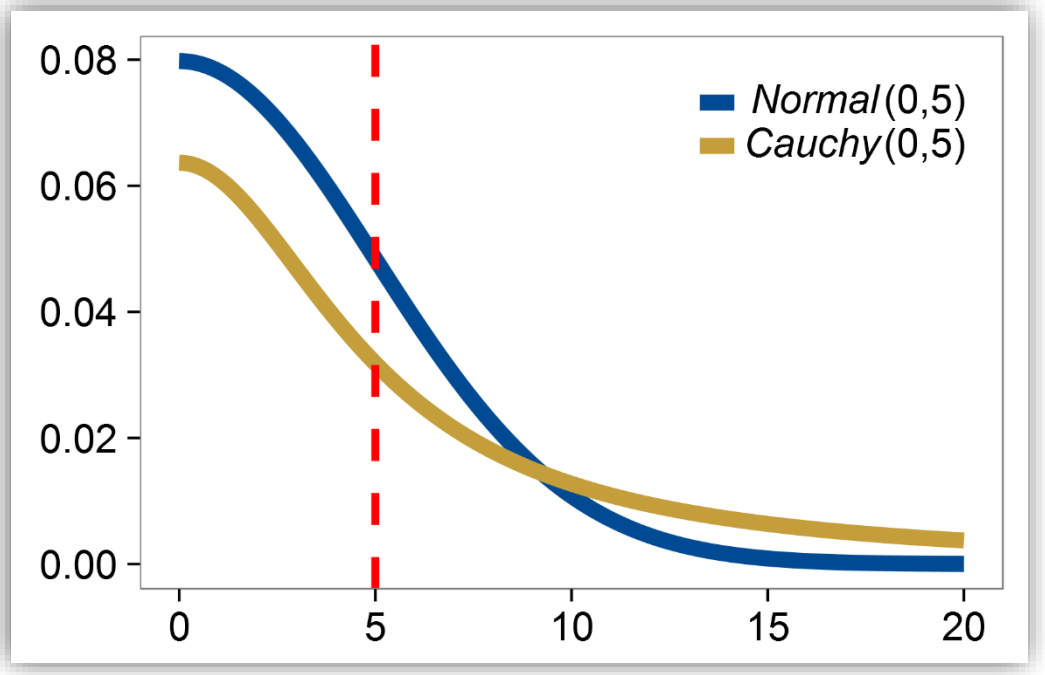
cognitive model
statistics
computing

$$\alpha \sim Normal(170, 100) \qquad \beta \sim Normal(0,20)$$

$$\overline{\text{height}} = \alpha + \beta * \text{weight}$$

$$\sigma \sim halfCauchy(0,20)$$

$$\text{height} \sim Normal(\overline{\text{height}}, \sigma)$$



Exercise VIII

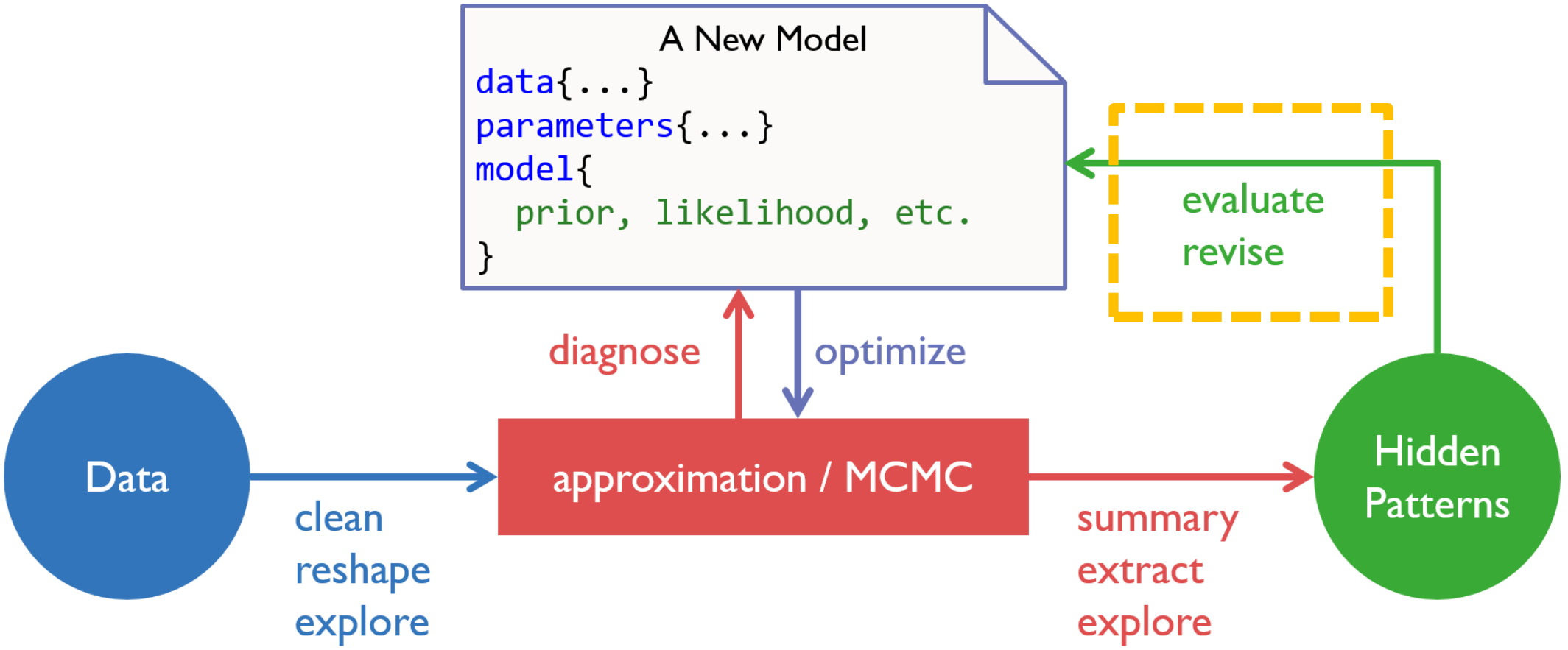
cognitive model
statistics
computing

…/04.regression_height/_scripts/regression_height_main.R

TASK: estimate the model and produce the results

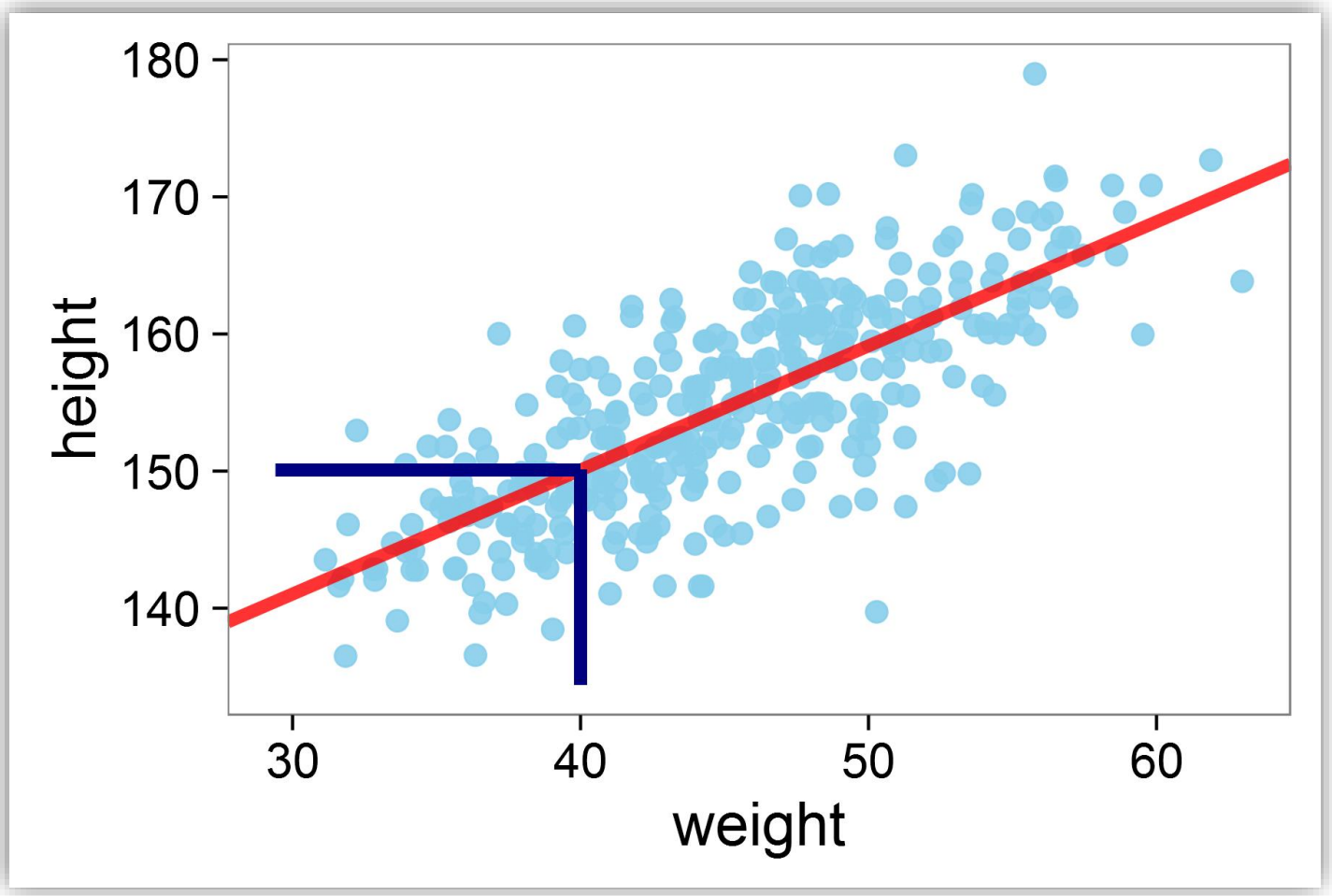
```
Inference for Stan model: regression_height_model.
4 chains, each with iter=2000; warmup=1000; thin=1;
post-warmup draws per chain=1000, total post-warmup draws=4000.

         mean se_mean   sd    2.5%     25%     50%     75%   97.5% n_eff Rhat
alpha  113.97    0.06 1.86  110.27  112.76  113.93  115.20  117.66   934    1
beta     0.90    0.00 0.04    0.82    0.88    0.90    0.93    0.99   922    1
sigma    5.11    0.01 0.19    4.74    4.97    5.10    5.24    5.50  1437    1
lp__  -747.61    0.04 1.23 -750.80 -748.15 -747.28 -746.72 -746.24   993    1
```

cognitive model
statistics
computing



What does the Model Predict?

cognitive model
statistics
computing



$$p(y_{rep} \mid y) = \int p(y_{rep} \mid \theta)\, p(\theta \mid y)\, d(\theta)$$

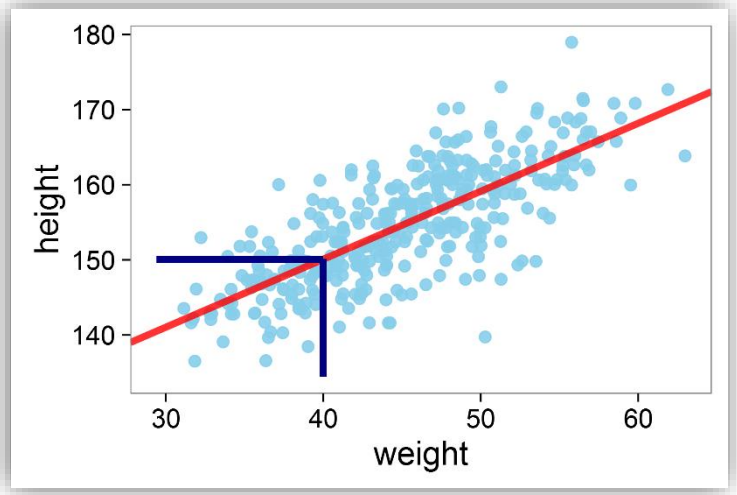
McElreath (2016)

Posterior Predictive Check (PPC)

cognitive model
statistics
computing

```
generated quantities {
  vector[N] height_bar;
  for (n in 1:N) {
    height_bar[n] = normal_rng(alpha + beta * weight[n], sigma);
  }
}
```

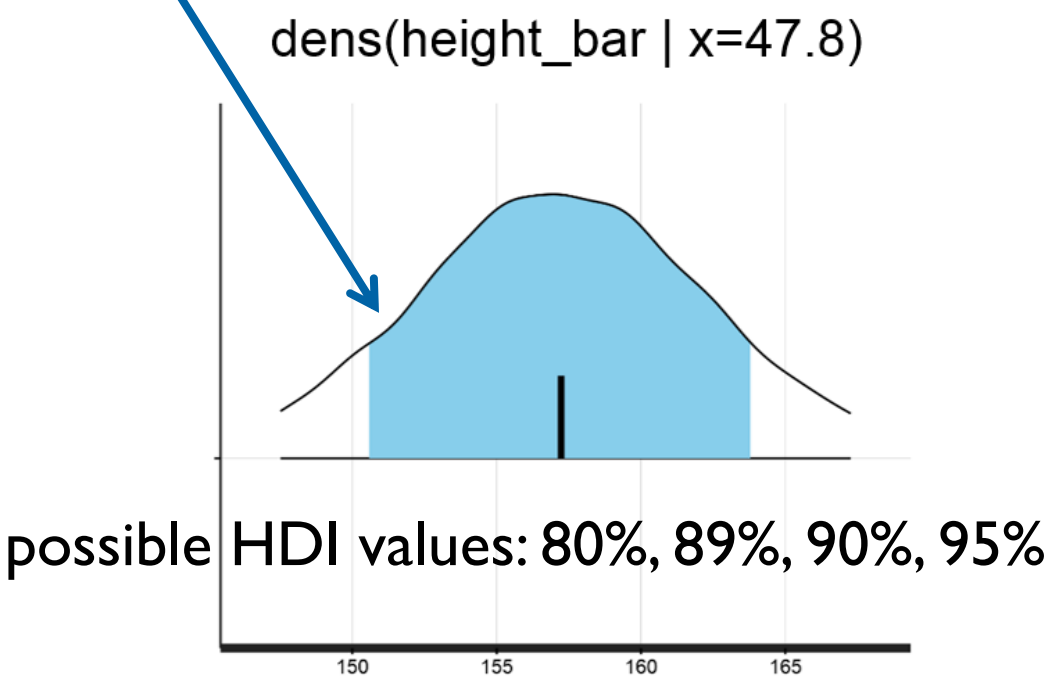
the generated quantities block runs only AFTER the sampling, and the time it costs can be essentially ignored!



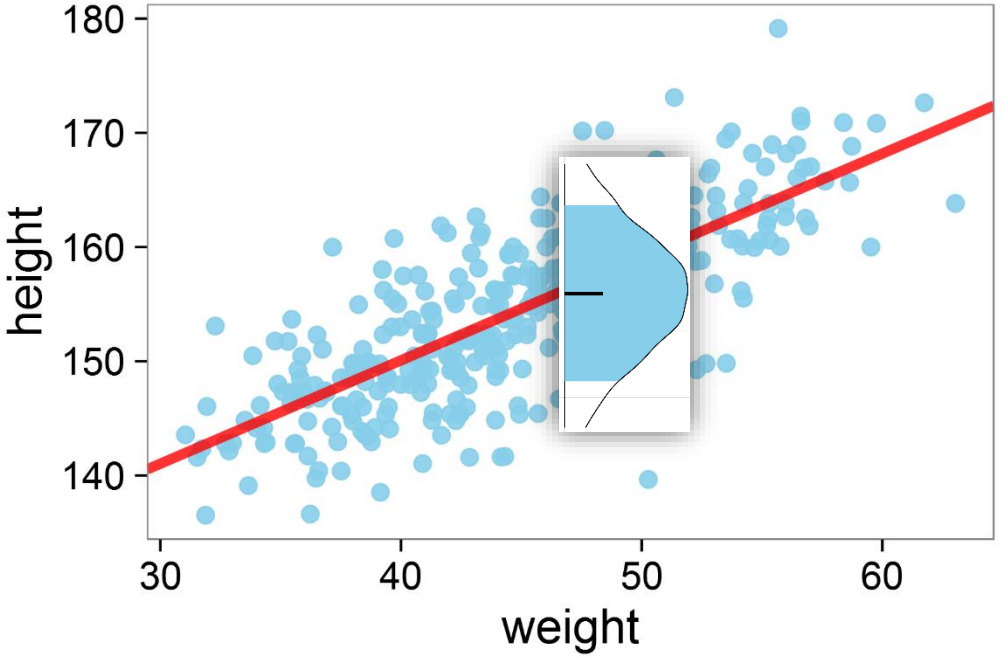
Posterior Predictive Check (PPC)

cognitive model
statistics
computing

Highest density interval (HDI)



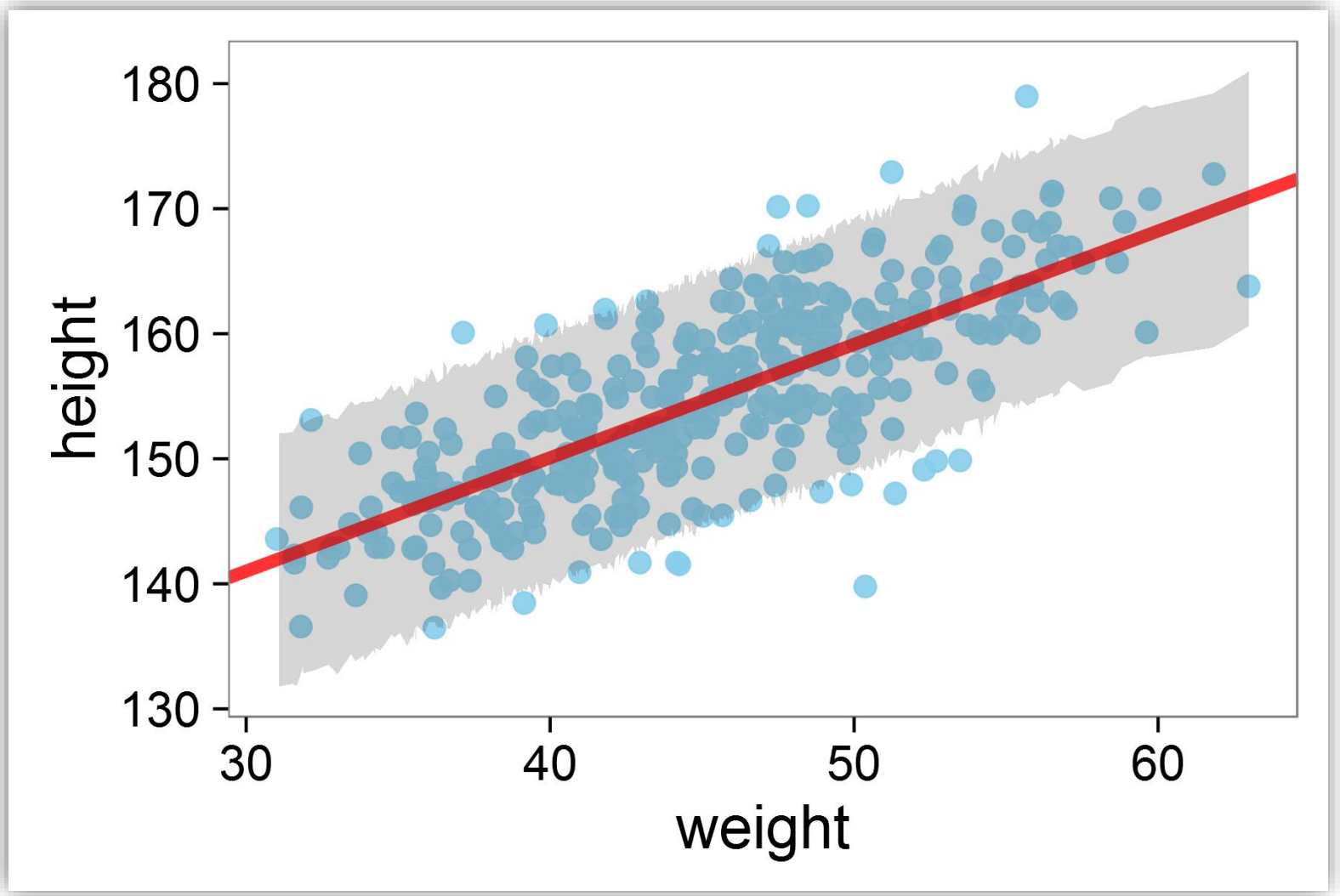
possible HDI values: 80%, 89%, 90%, 95%




```
height_bar <- extract(fit_reg_ppc, pars = 'height_bar',
                      permuted = FALSE)$height_bar
height_HDI <- apply(height_bar, 2, HDIofMCMC)
```

Posterior Predictive Check (PPC)

cognitive model
statistics
computing



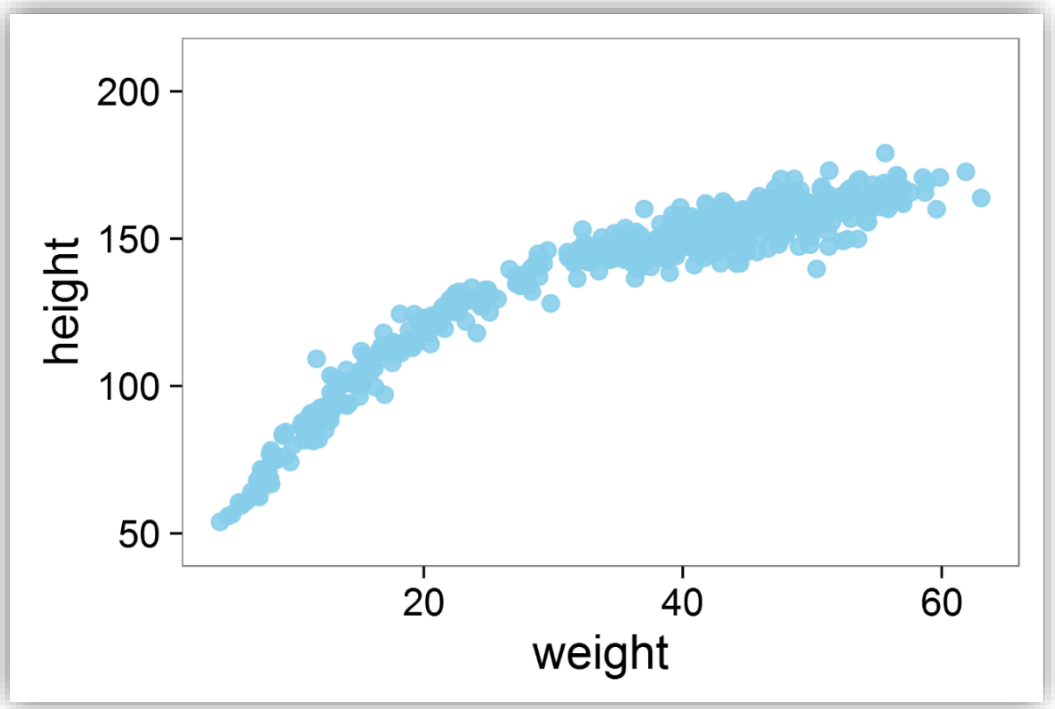
Exercise IX

cognitive model
statistics
computing

```
…/05.regression_height_poly/_scripts
  /regression_height_poly_main.R
```

TASK: (1) Complete "regression_height_poly2_model.stan"

(2) produce PPC plot for both 1st order and 2nd order polynomial fit



Exercise IX – Tips

cognitive model
statistics
computing

```
> source('_scripts/regression_height_poly_main.R')

> out1 <- reg_poly(poly_order = 1)
```

$$\overline{\text{height}} = \alpha + \beta 1 * \text{weight} + \beta 2 * \text{weight}^2$$

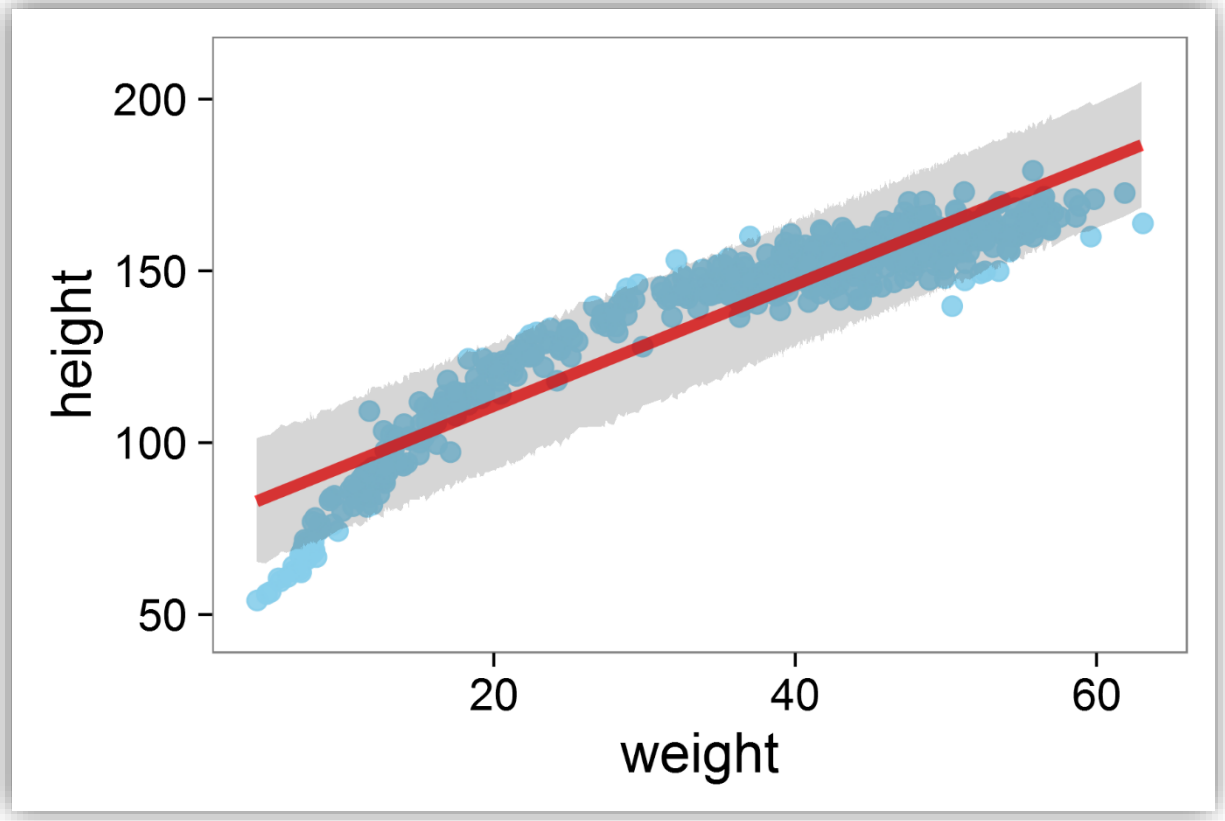
$$\text{height} \sim Normal(\overline{\text{height}}, \sigma)$$

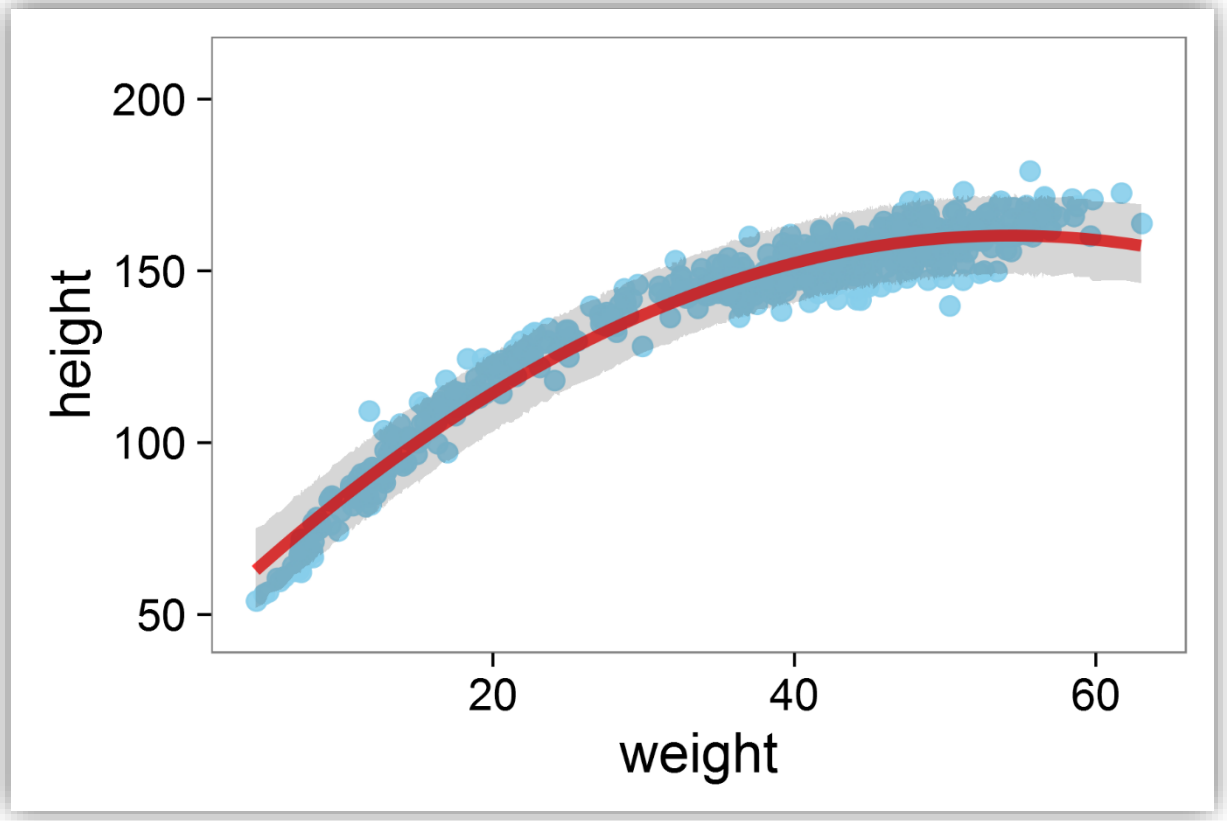
```
data {
  int<lower=0> N;
  vector<lower=0>[N] height;
  vector<lower=0>[N] weight;
  vector<lower=0>[N] weight_sq;
}
```

```
height ~ normal(alpha + beta1 * weight + beta2 * weight_sq, sigma);
```

Exercise IX – output2

cognitive model
statistics
computing





Complex GLMs in Stan?

Link

Want to learn more about Stan?

cognitive model
statistics
computing

Workshops

- StanCon 2019 Hierarchical Models
- PyData NYC 2019
- StanCon 2018 Intro Stan
- StanCon 2018 Hierarchical Models

https://mc-stan.org/workshops/

Twitter

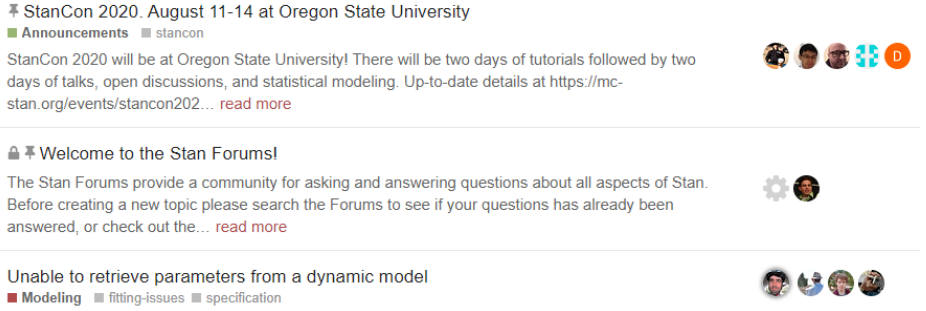
Richard McElreath
@rlmcelreath

Quiche Lorraine,
@dan_p_simpson

\mathfrak{Michael
@betanalpha

EJ Wagenmakers
@EJWagenmakers

Stan forum



https://discourse.mc-stan.org/


ANY
QUESTIONS
?

Happy Computing!