Bayesian Statistics and Hierarchical Bayesian Modeling for Psychological Science

Lecture 09

Lei Zhang

Social, Cognitive and Affective Neuroscience Unit (SCAN-Unit)

Department of Basic Psychological Research and Research Methods

lei.zhang@univie.ac.at

lei-zhang.net

@lei_zhang_lz

https://github.com/lei-zhang/BayesCog_Wien

universität wien

Fakultät für Psychologie

Bayesian **warm-up**?

STAN PROGRAMMING LANGUAGE II

Why Use Stan?

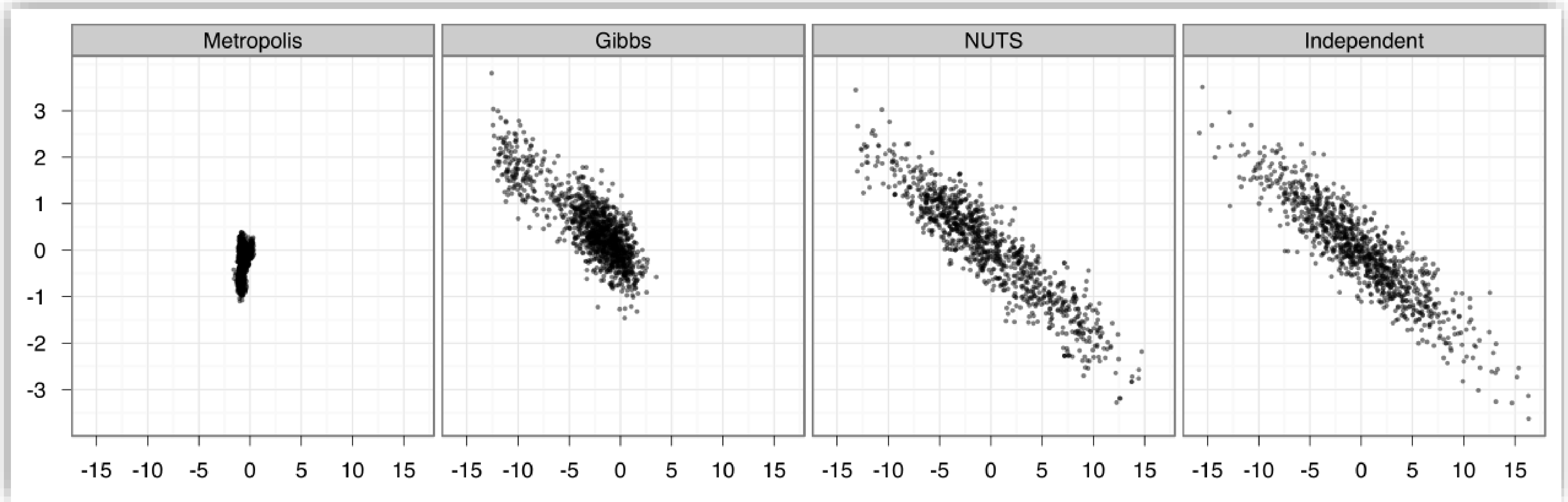
vs. BUGS and JAGS

- Time to converge and per effective sample size:
  0.5 - ∞ times faster
- Memory usage: 1 - 10%
- Language features
  - variable overwrite: `a = 4, then a = 5`
  - formal control flow
  - full support of vectorizing

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

NUTS vs. Gibbs and Metropolis

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

General Properties of Stan Language

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

Variable's Scope

	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

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)

Scalar Variables

`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


```
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

- 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)

cognitive model

statistics

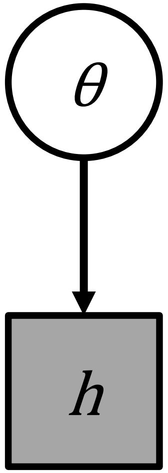
computing

Bernoulli Model

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

$N = 1$

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



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

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

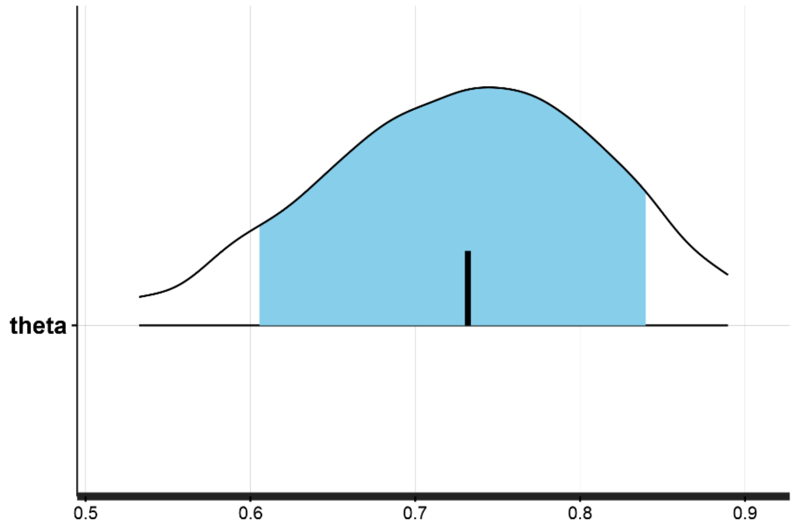
Exercise VIII

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?

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

61.59 secs*

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

53.25 secs*

Thinking before looping!

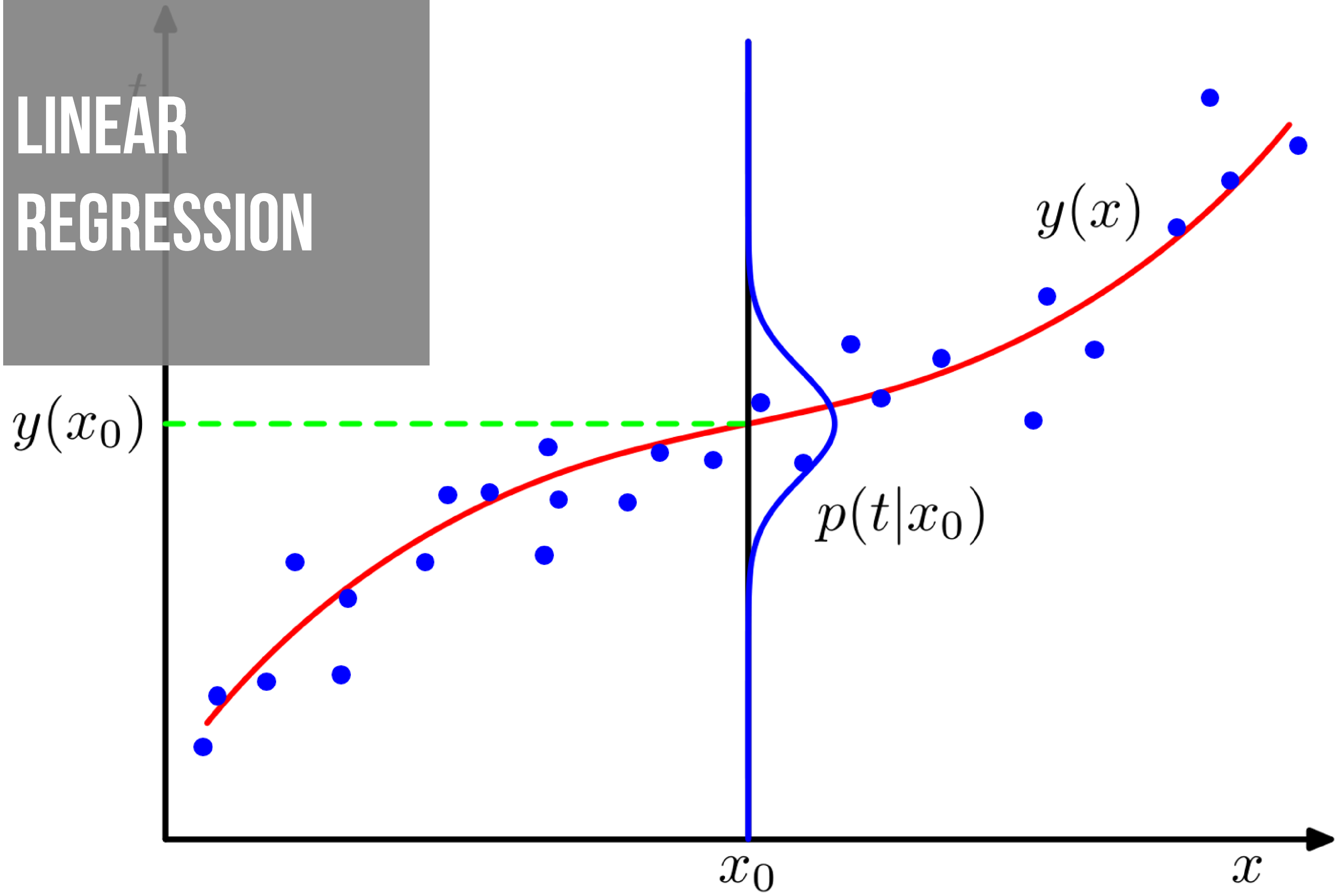
* compiling time included

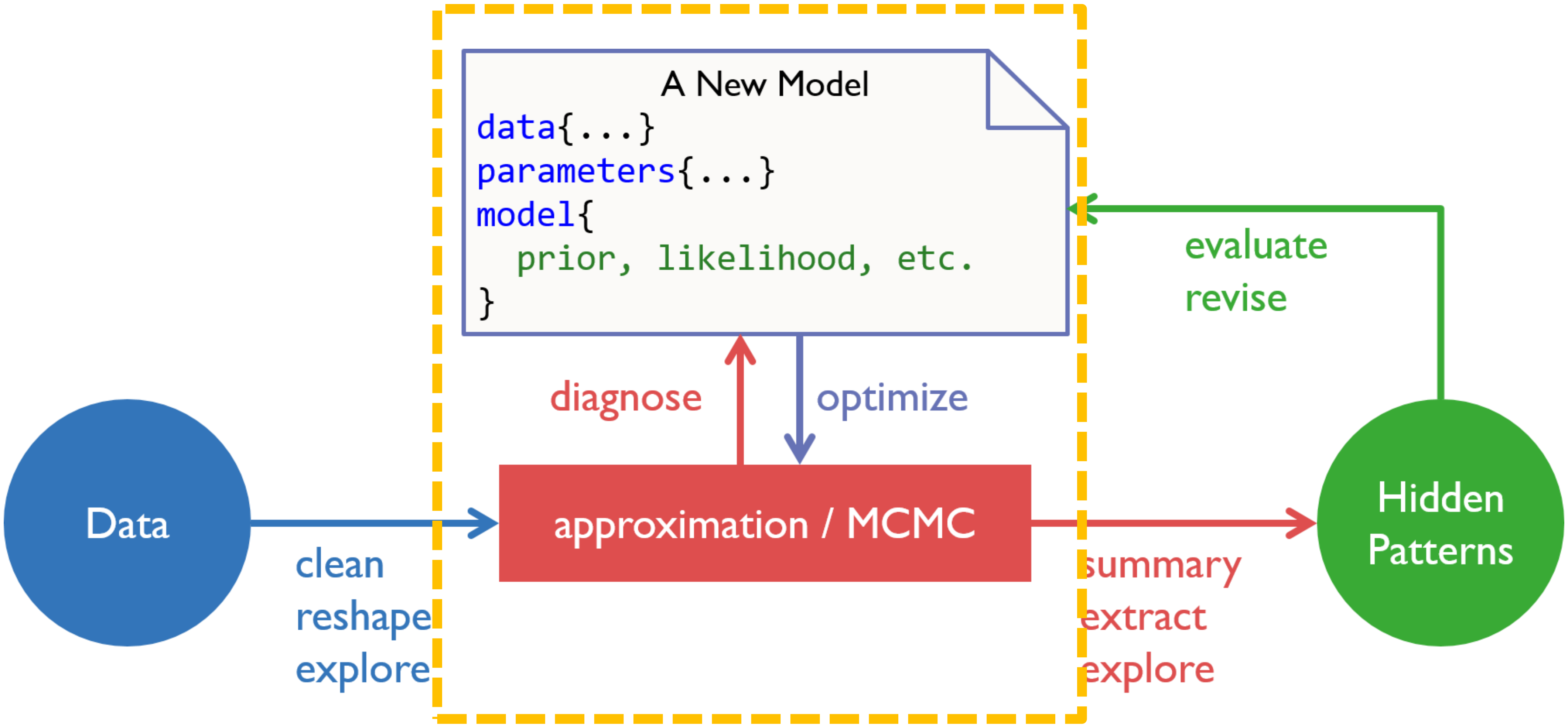
Recap

What we've learned…

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

LINEAR REGRESSION





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

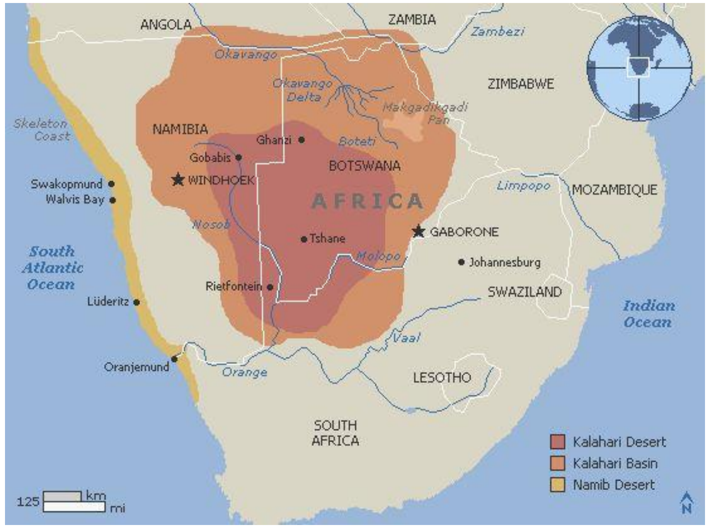
cognitive model
statistics
computing

Linear Regression: height ~ weight


```
…/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
```

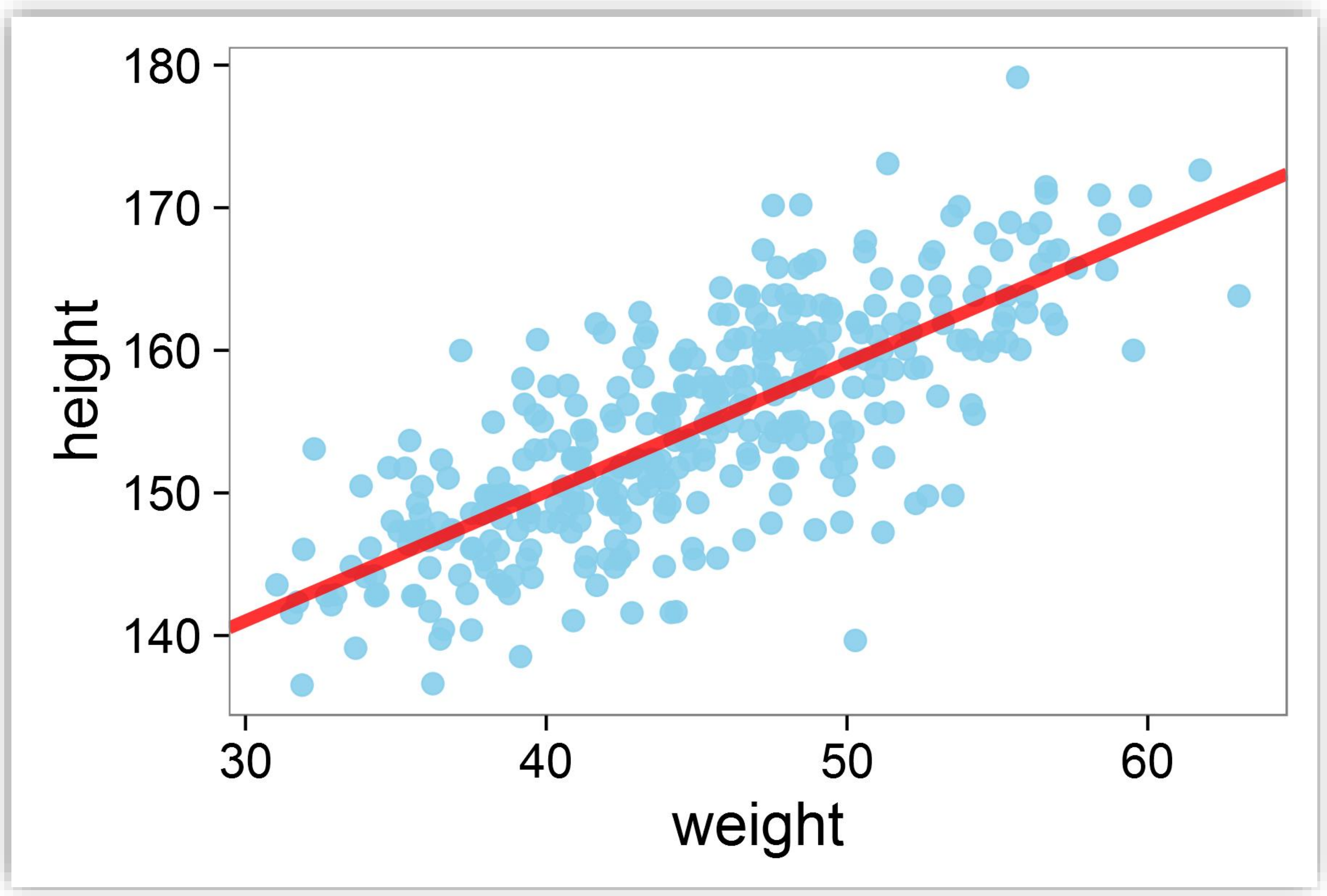


McElreath (2016)

Results with lm()


```
> 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



McElreath (2016)

Rethinking Regression Model

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

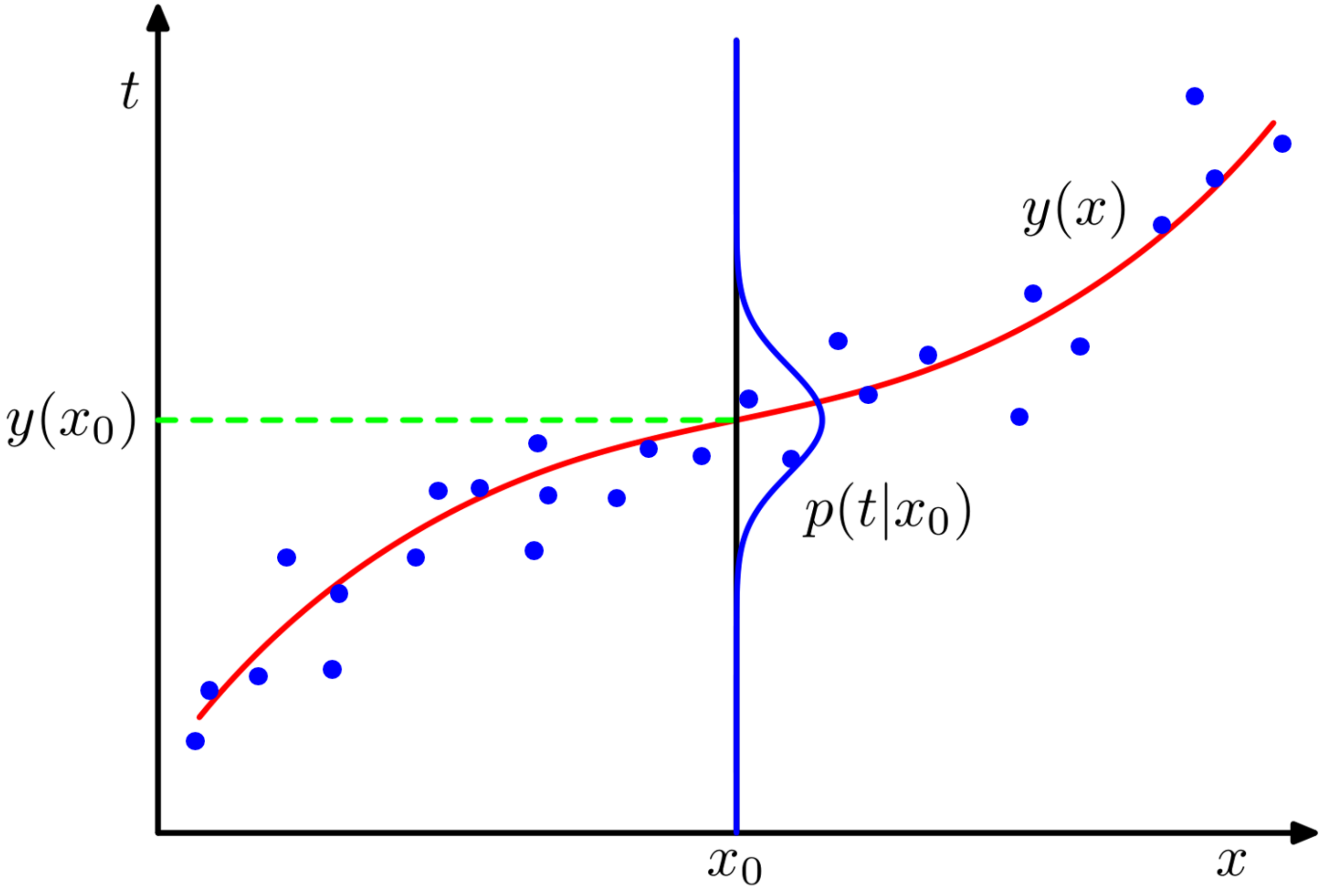
$$\cancel{y_i = \mu_i + \varepsilon}$$

$$\cancel{\varepsilon \sim Normal(0, \sigma)}$$

$$\boxed{y_i \sim Normal(\mu_i, \sigma)}$$

Rethinking Regression Model

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



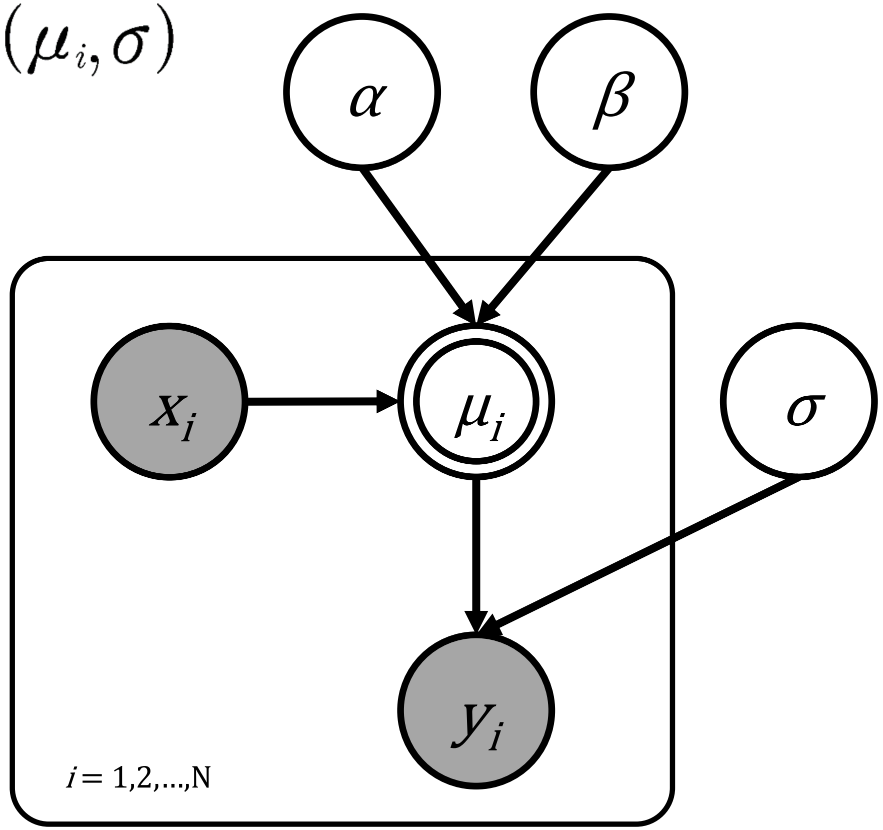
Bishop (2006)

Rethinking Regression Model

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?

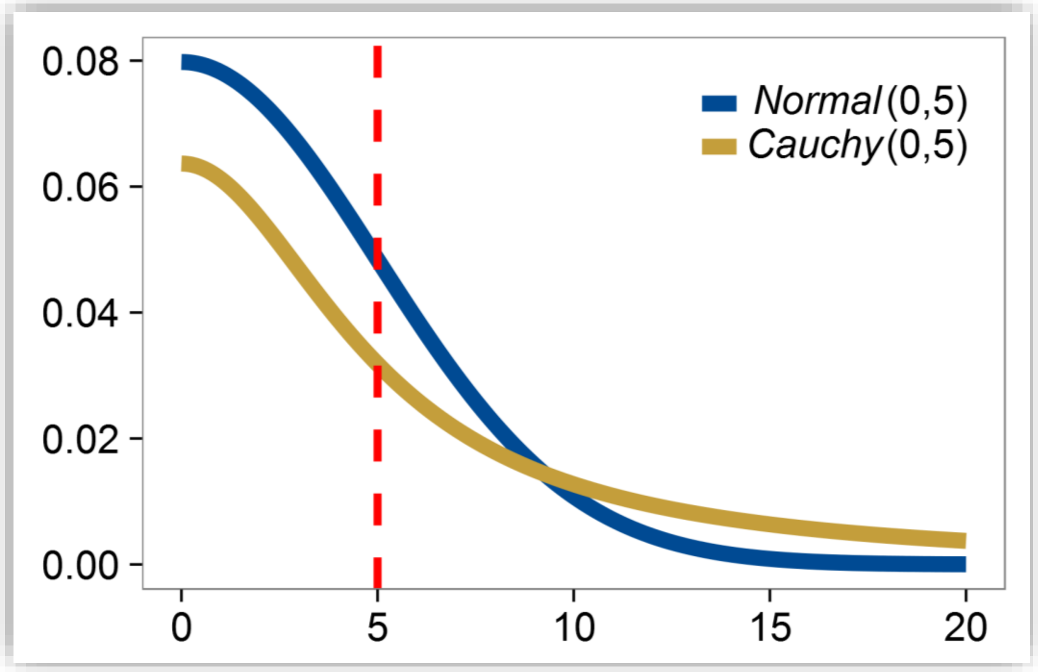
cognitive model
statistics
computing

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

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

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

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



Exercise VIII

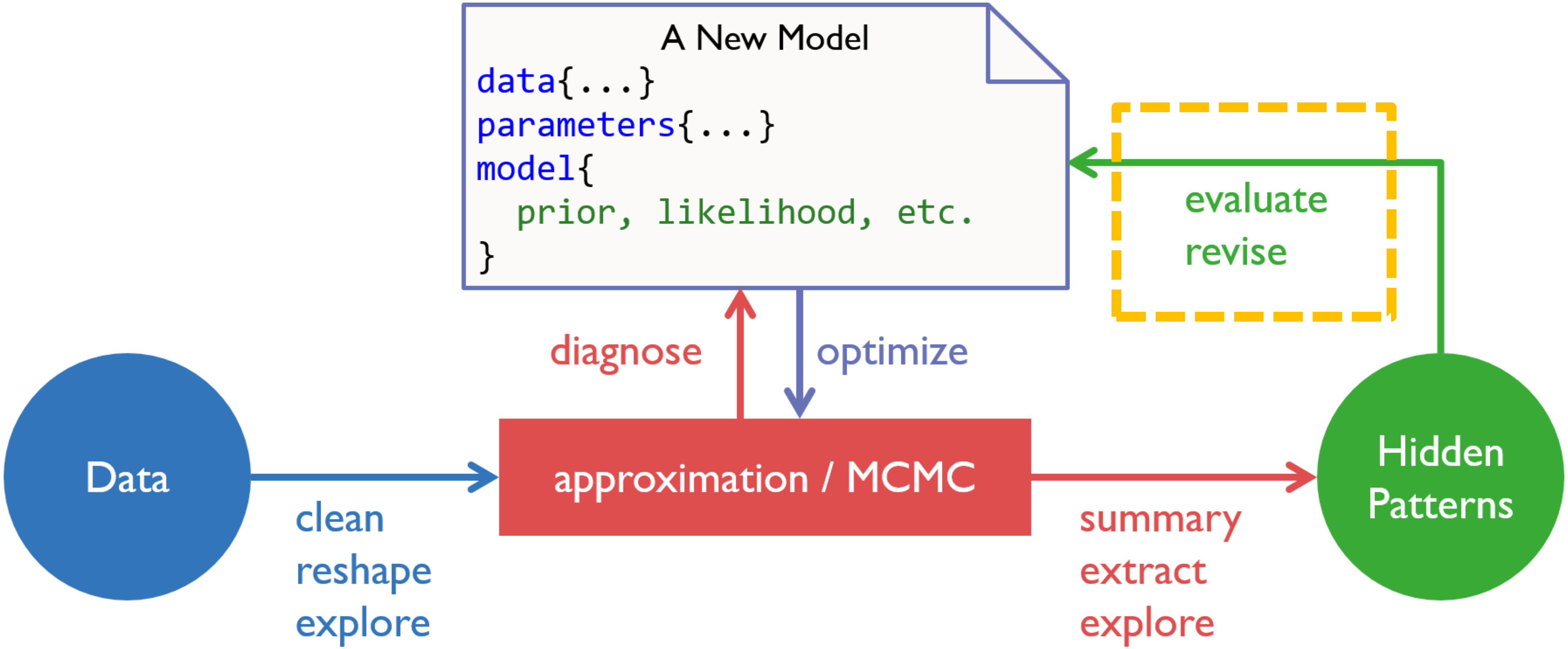
cognitive model
statistics
computing

```
…/04.regression_height/_scripts/regression_height_main.R
```

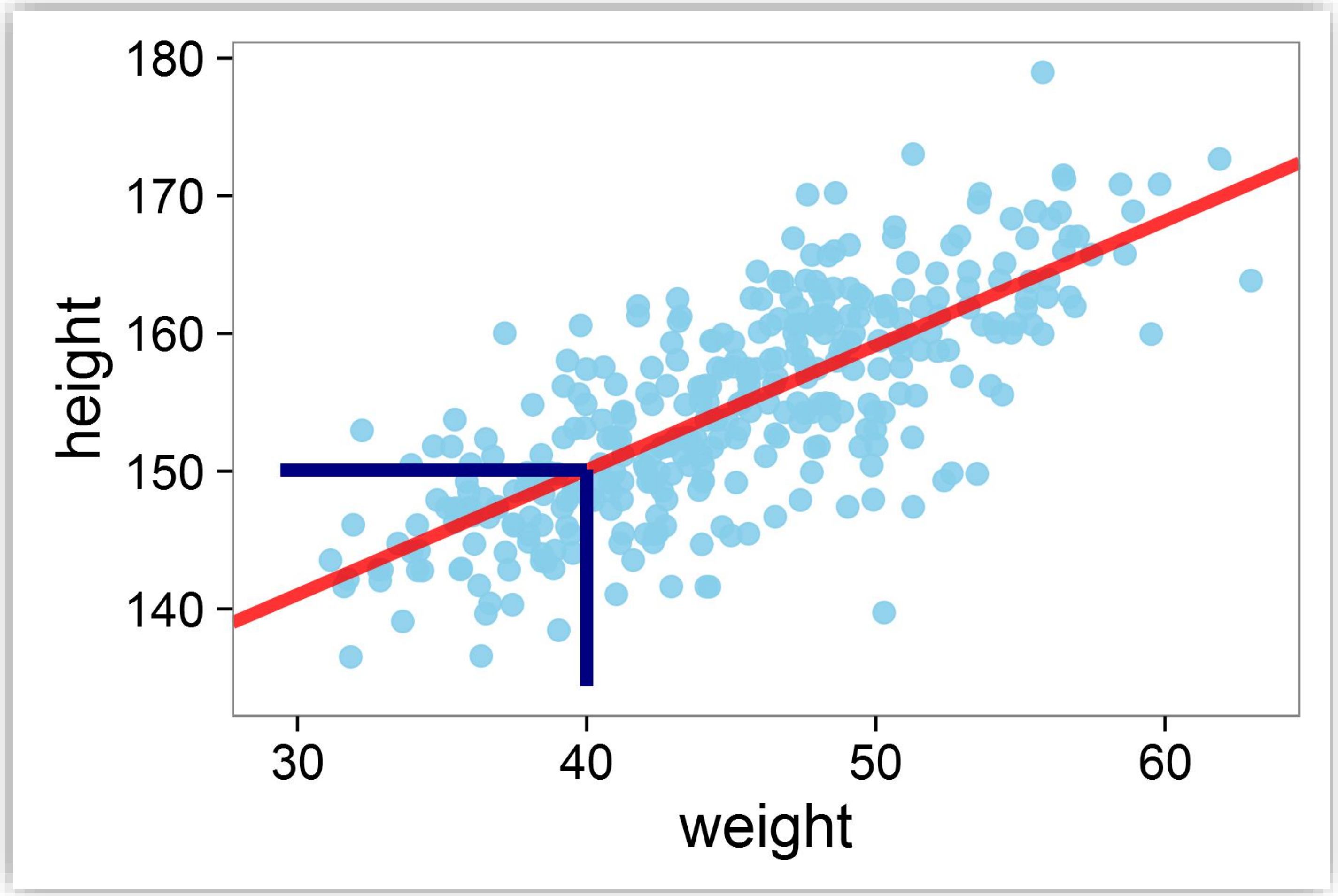
TASK: estimate the model and produce the results

```
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?



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

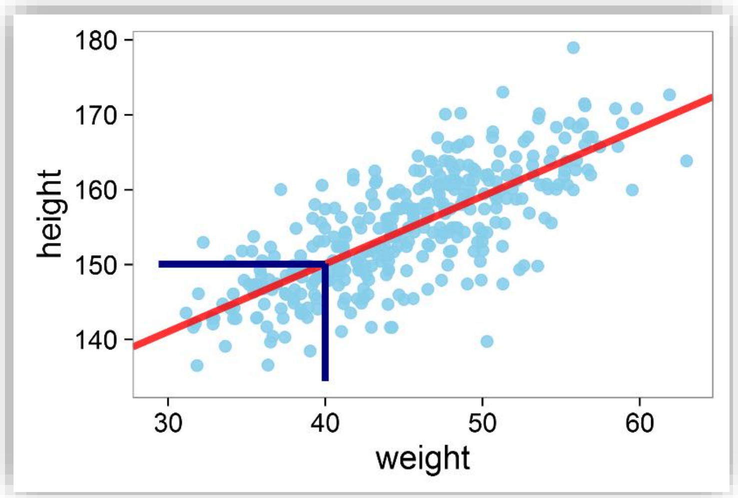
McElreath (2016)

Posterior Predictive Check (PPC)

cognitive model
statistics
computing

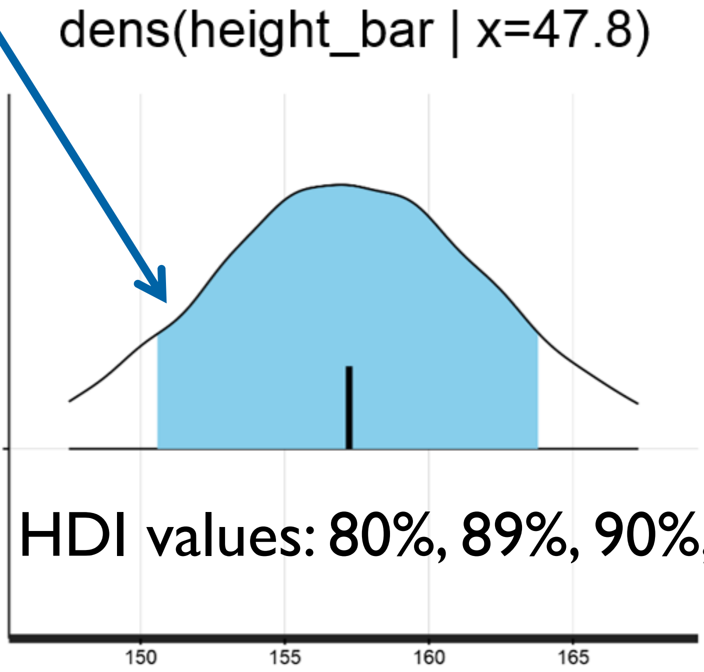
```
generated quantities {
  vector[N] height_bar;
  for (n in 1:N) {
      height_bar[n] = normal_rng(alpha + beta * weight[n], sigma);
  }
}
```

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

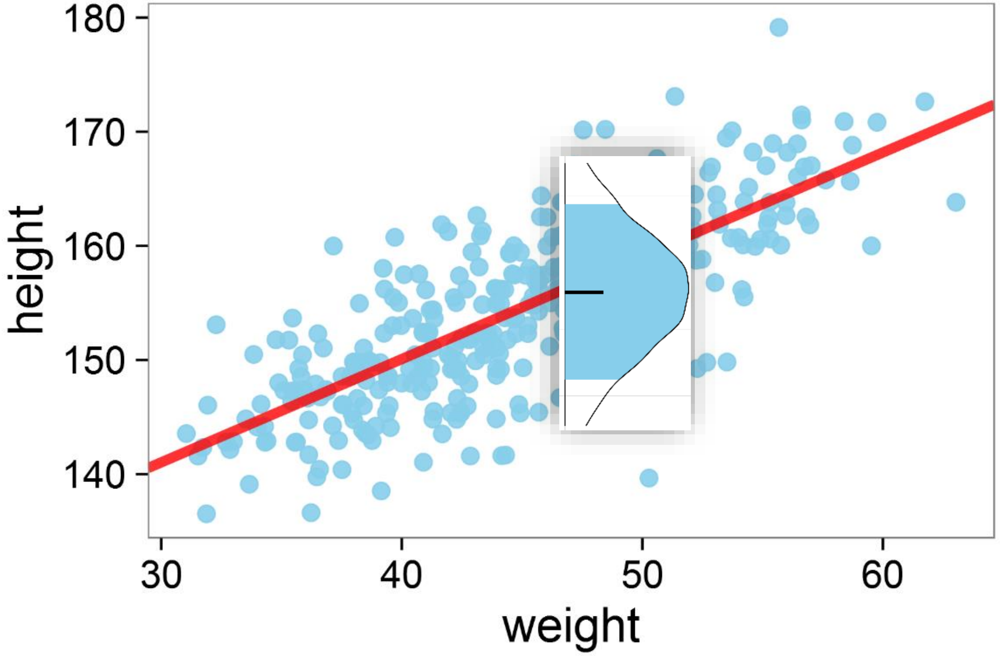


Posterior Predictive Check (PPC)

Highest density interval (HDI)



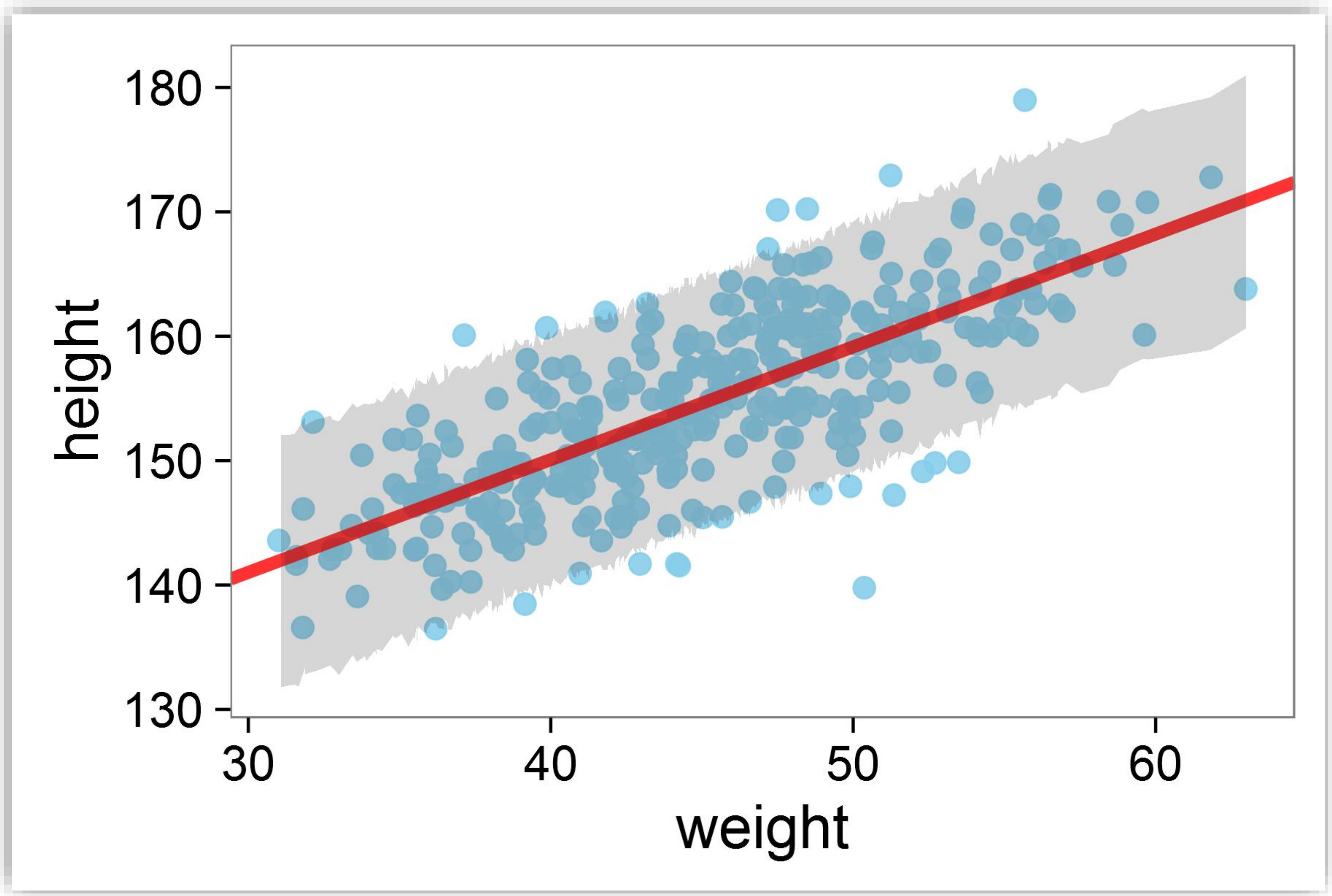
possible HDI values: 80%, 89%, 90%, 95%




```
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

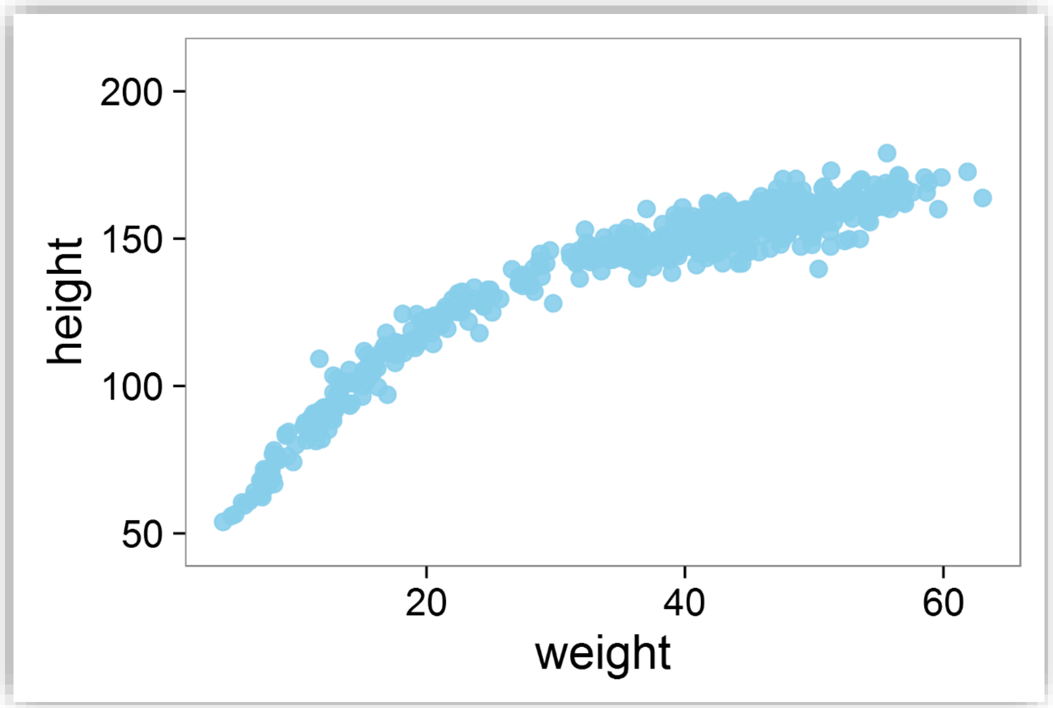


Exercise IX

```
…/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