Statistical Rethinking 2024, Week 2

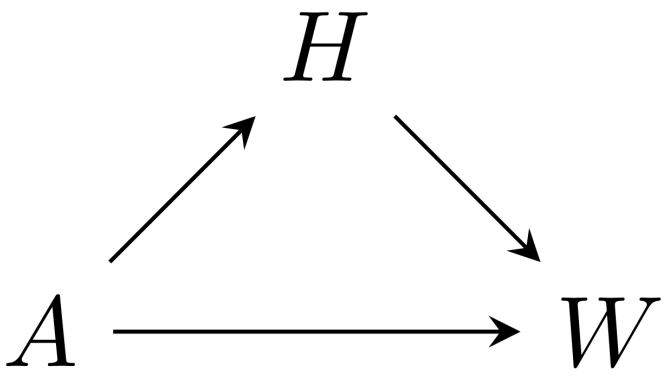
Neal Grantham
January 19, 2024

```
# Julia v1.9.3
using StatsBase
using StatsPlots
using Distributions
using Turing
using CSV
using DataFrames
```

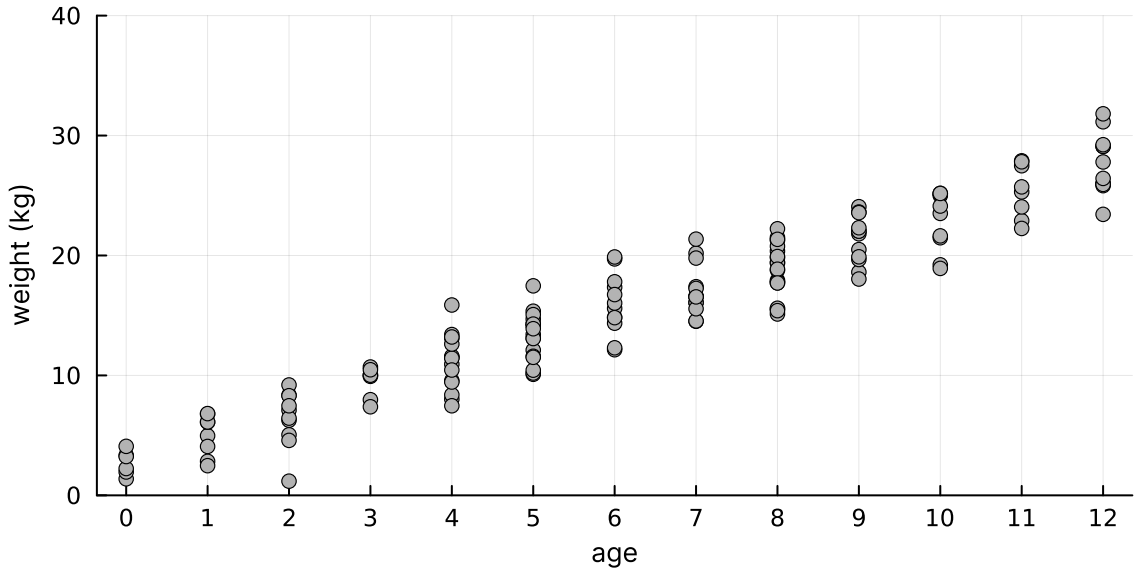
Q1. From the `Howell1` dataset, consider only the people younger than 13 years old. Estimate the causal association between age and weight. Assume that age influences weight through two paths:

1. age influences height and height influences weight, and
2. age directly influences weight through age-related changes in muscle growth and body proportions.

Draw the DAG that represents these causal relationships. And then write a generative simulation that takes age as an input and simulates height and weight, obeying the relationships in the DAG.




```
function generate_child(; α=0, β=0, γ=0)
    a = sample(0:12)
    h = 50 + 5 * a + rand(Normal(0, 2))  # 50 cm at birth + 5 cm / year + var.
    w = α + β * a + γ * h + rand(Normal(0, 2))
    (age=a, height=h, weight=w)
end

sim_params = Dict(:α => 3.3, :β => 2.0, :γ => 0.0)
sim_children = DataFrame(generate_child(; sim_params...) for _ in 1:150)

scatter(
    sim_children.age, sim_children.weight,
    xticks=0:12, ylims=(0, 40),
    ylab="weight (kg)", xlab="age"
)
```



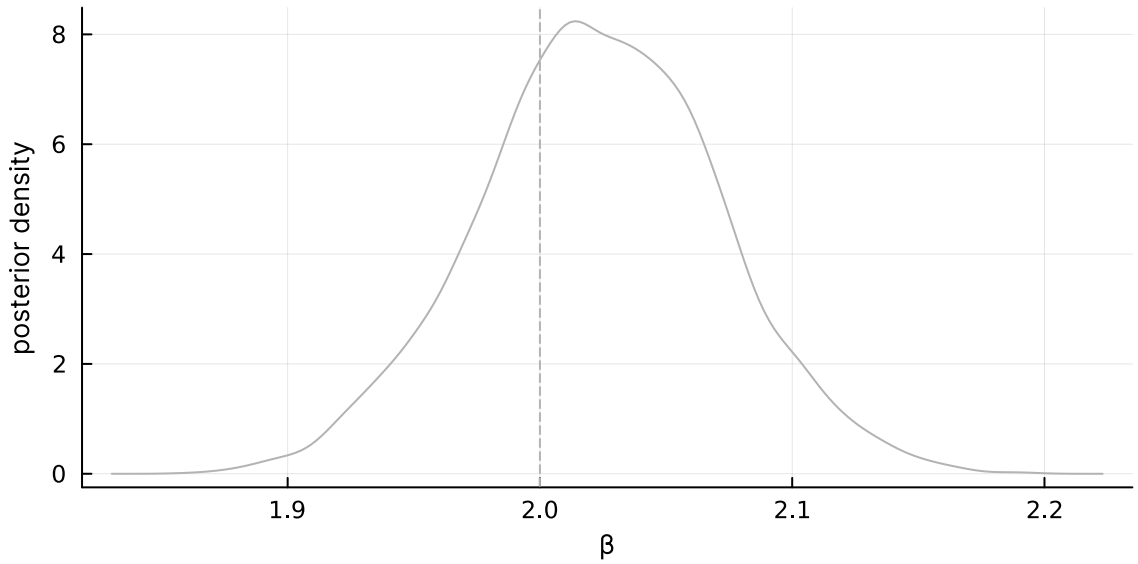
Q2. Estimate the total causal effect of each year of growth on weight.

We define the following `child_growth` model, where `a` is a child's age in years (0 to 12) and `w` is a child's weight in kilograms.

```
@model function child_growth(a, w)
    α ~ Normal(4, 1)
    β ~ LogNormal(0, 0.5)
    σ ~ Exponential(0.5)
    for i in eachindex(w)
        μ = α + β * a[i]
        w[i] ~ Normal(μ, σ)
    end
    w
end;
```

How does the model perform on the simulated data?

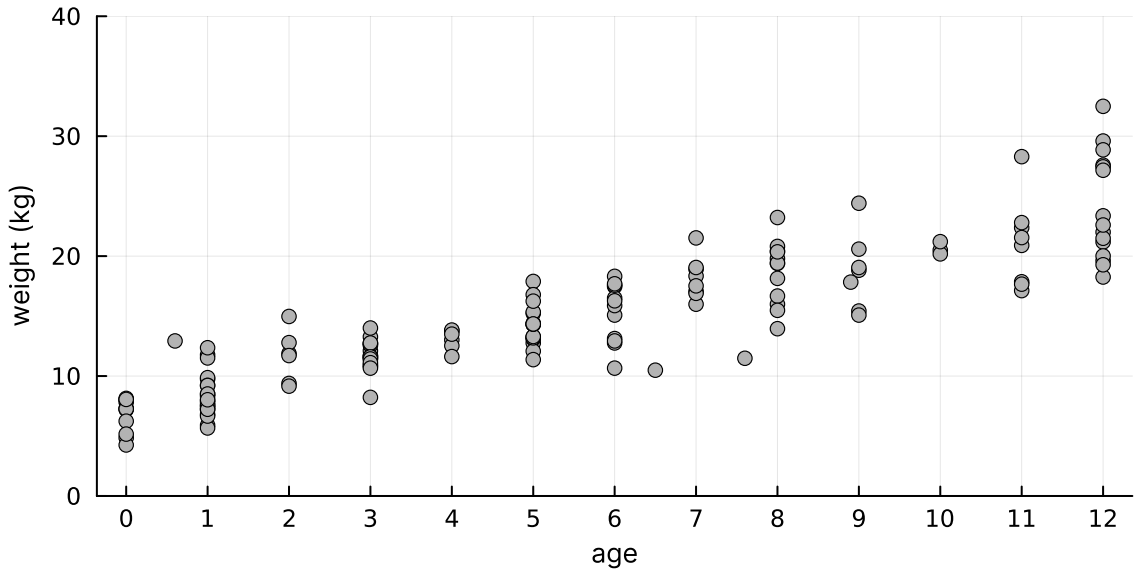
```
sim_model = child_growth(sim_children.age, sim_children.weight)
sim_chain = sample(sim_model, NUTS(), 5000)

density(sim_chain[:β], ylab="posterior density", xlab="β")
vline!([sim_params[:β]], linestyle=:dash)
```



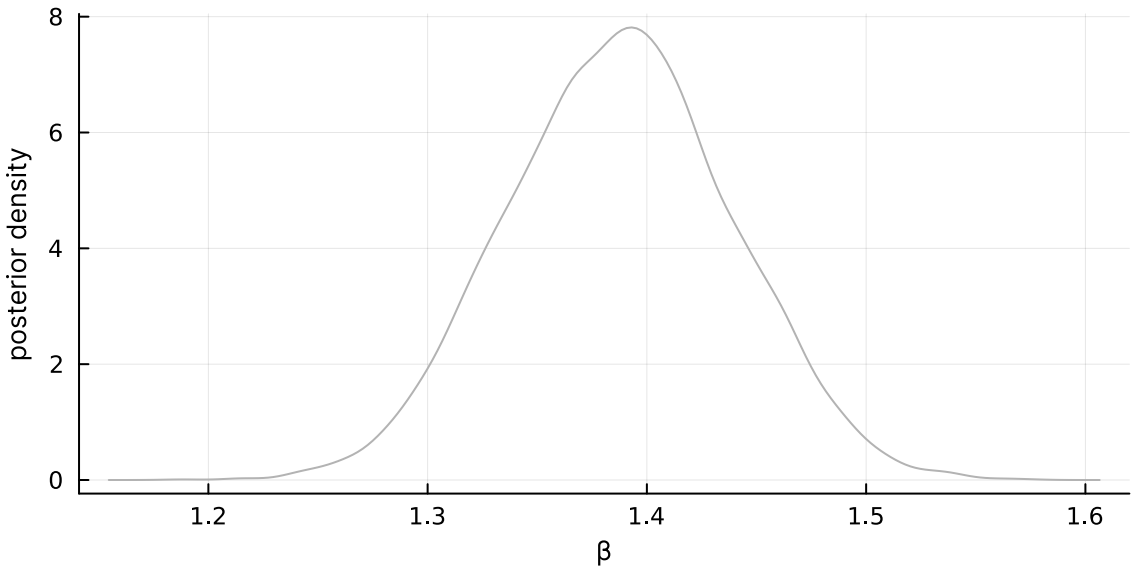
The model successfully recovered the simulated value of β.

Now we fit the model to Howell's Dobe !Kung census data.

```
url = "https://raw.githubusercontent.com/rmcelreath/rethinking/master/data/Howell1.csv"
howell = CSV.read(download(url), DataFrame; delim=';')

children = howell[howell.age .< 13, :]

scatter(
    children.age, children.weight,
    xticks=0:12, ylims=(0, 40),
    xlab="age", ylab="weight (kg)"
)
```



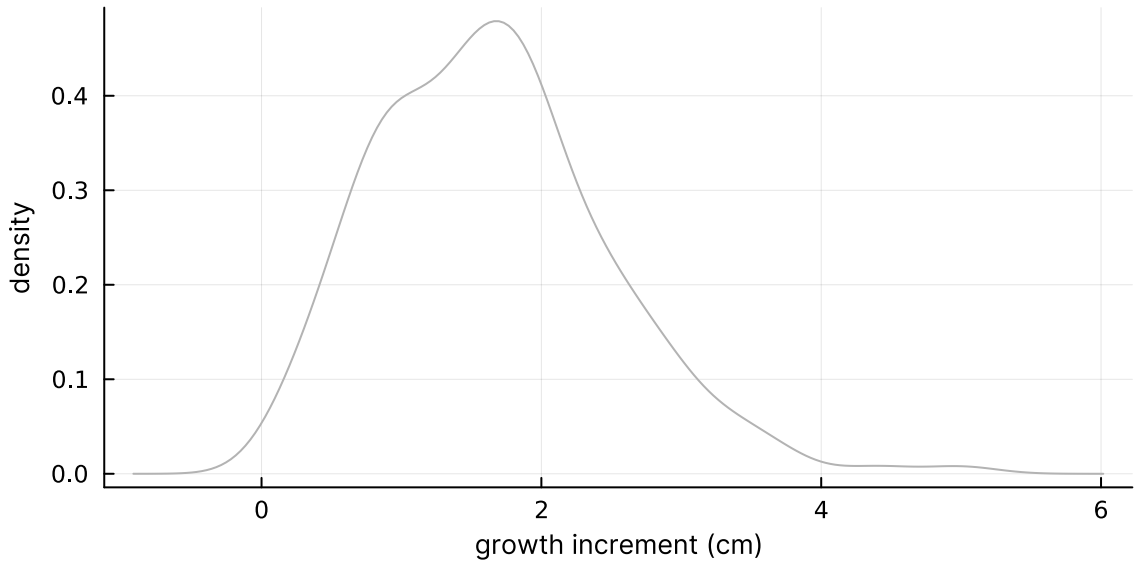
```
model = child_growth(children.age, children.weight)
chain = sample(model, NUTS(), 5000)

density(chain[:β], ylab="posterior density", xlab="β")
```



We estimate the total causal effect of age on weight for children under 13 years of age to be between 1.2 and 1.6 kilograms per year.

Q3. The data in `Oxboys` are growth records for 26 boys measured over 9 periods. I want you to model their growth. Specifically, model the increments in growth from one period (`Occasion` in the data table) to the next. Each increment is simply the difference between height in one occasion and height in the previous occasion. Since none of these boys shrunk during the study, all of the growth increments are greater than zero. Estimate the posterior distribution of these increments. Constrain the distribution so it is always positive — it should not be possible for the model to think that boys can shrink from year to year. Finally compute the posterior distribution of the total growth over all 9 occasions.

```
url = "https://raw.githubusercontent.com/rmcelreath/rethinking/master/data/Oxboys.csv"
oxboys = CSV.read(download(url), DataFrame; delim=';')

oxboys = combine(groupby(oxboys, :Subject), :height => diff => :diff);

density(oxboys.diff, ylab="density", xlab="growth increment (cm)")
```



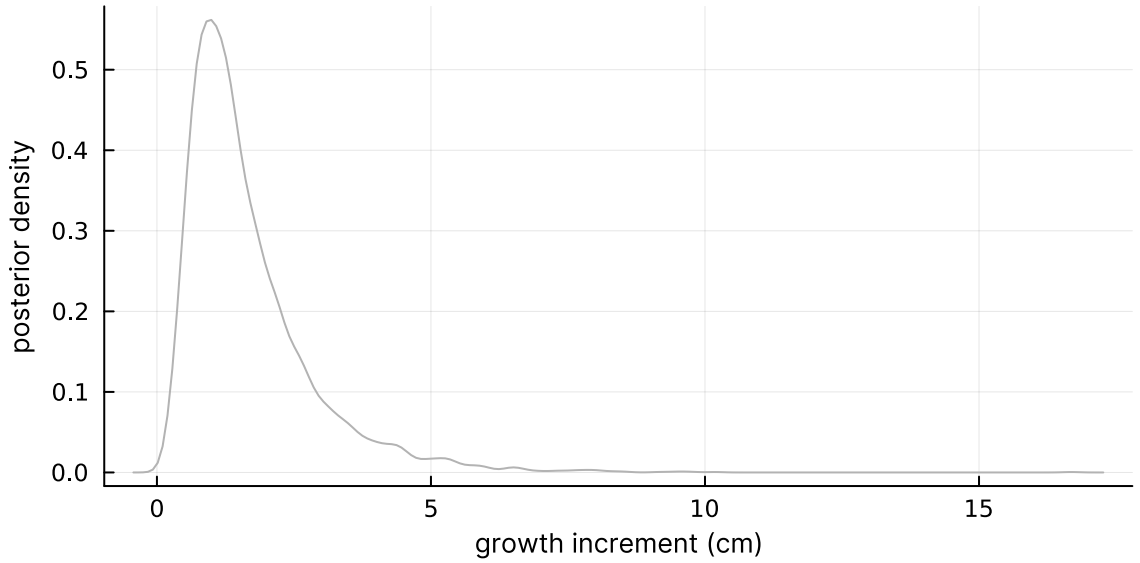
We define the following oxboy_growth model, where d is a boy's height increment (in cm) between two periods.

```
@model function oxboy_growth(d)
    σ ~ Exponential(0.2)
    μ ~ Normal(0, 0.2)
    for i in eachindex(d)
        d[i] ~ LogNormal(μ, σ)
    end
    d
end

model = oxboy_growth(oxboys.diff)
chain = sample(model, MH(), 5000)
```

```
Chains MCMC chain (5000×3×1 Array{Float64, 3}):

Iterations        = 1:1:5000
Number of chains  = 1
Samples per chain = 5000
Wall duration     = 0.33 seconds
Compute duration  = 0.33 seconds
parameters        = σ, μ
internals         = lp

Summary Statistics
  parameters      mean       std      mcse   ess_bulk   ess_tail       rhat   e ⋯
      Symbol   Float64   Float64   Float64    Float64    Float64    Float64     ⋯

           σ    0.6322    0.0478    0.0092    21.2664    19.9334     2.0343     ⋯
           μ    0.3151    0.0700    0.0154    16.2858    13.7129     2.1252     ⋯
                                                                1 column omitted

Quantiles
  parameters      2.5%     25.0%     50.0%     75.0%     97.5%
      Symbol   Float64   Float64   Float64   Float64   Float64
```

```
σ    0.5226    0.6196    0.6525    0.6525    0.6940
μ    0.1767    0.2795    0.3217    0.3342    0.4365
```

```
post = zip(chain[:μ], chain[:σ])
pred_post = rand.(LogNormal(μ, σ) for (μ, σ) in post)
density(pred_post, ylab="posterior density", xlab="growth increment (cm)")
```




```
pred_post = [sum(rand(LogNormal(μ, σ), 8)) for (μ, σ) in post]
density(pred_post, ylab="posterior density", xlab="total growth (cm)")
```