Chap 14: Adventures in Covariance

```
md"# Chap 14: Adventures in Covariance"
```

```
versioninfo()
```

```
Julia Version 1.10.2
Commit bd47eca2c8a (2024-03-01 10:14 UTC)
Build Info:
  Official https://julialang.org/ release
Platform Info:
  OS: Linux (x86_64-linux-gnu)
  CPU: 32 × Intel(R) Xeon(R) CPU E5-2630 v3 @ 2.40GHz
  WORD_SIZE: 64
  LIBM: libopenlibm
  LLVM: libLLVM-15.0.7 (ORCJIT, haswell)
Threads: 16 default, 0 interactive, 8 GC (on 32 virtual cores)
Environment:
  JULIA_PKG_SERVER = https://mirrors.tuna.tsinghua.edu.cn/julia
  JULIA_REVISE_WORKER_ONLY = 1
```

```
html"""
<style>
    main {
        margin: 0 auto;
        max-width: max(1800px, 75%);
        padding-left: max(5px, 1%);
        padding-right: max(350px, 10%);
    }
</style>
"""
```

Table of Contents

Chap 14: Adventures in Covariance
14.1 Varying slopes by construction.
Code 14.5 - 14.8 Simulate a_cafe (intercept) and b_cafe(slope)
Code 14.9 Plot a_cafe(intercepts) vs b_cafe(slopes) and its contour
Code 14.10 Simulate observations
Code 14.11 LKJ prior on ρ
Code 14.12 m14_1 multilevel covariance model
Code 14.13 Posterior vs prior of ρ
Code 14.14 Draw posterior estimates of intercept vs slope
Code 14.15 Draw the contour of posterior estimates
Code 14.16 Draw morning wait vs afternoon wait
Code 14.17 Draw contour , but failed.
14.2 Advanced varying slopes.
Code 14.18 Load data and fit model m14_2
Code 14.19 m14_3 non-centered approach via Cholesky Decomposition
Code 14.20 Fig 14.6 Compare ess of m14_2 vs m14_3
Code 14.21 range of σ for each actor by m14_3
Code 14.22 Posterior predictions (black) vs raw data (blue)
14.3 Instruments and causal designs.
Code 14.23 Simulate: Education has no effect on Wage and U(Unknown) has positive effect on W.
Code 14.24 m14_4 shows Education has strong effect on Wage!
Code 14.25 m14_5: Including Q amplifies the false effect of E on W.
Code 14.26 Generative model to statistical model. m14_6: Model residual covariance between W an…
Code 14.28 Simulate2: E has a positive effect on W.
Code 14.29 Find out which one is instrumental variable via dagitty.jl (NOT Implemented)
14.4 Social relations as correlated varying effects.
Code 14.30 Load the data
Code 14.31 m14_7 a model with dyad covariance
Code 14.33 generated giving vs receiving
Code 14.35 Estimates of dyads
Code 14.36 Residual gifts are strongly correlated within dyads.
14.5 Continuous categories and the Gaussian process.
Code 14.37 Load the distance matrix of 10 islands in the Kline dataset
Code 14.38 Covariance function of linear or squared distance
Code 14.39 Tools vs pop and interaction
Code 14.40 Posterior estimates
Code 14.41 Posterior covariance function
Code 14.42 median estimates of K
Code 14.43 Convert K to correlation matrix
Code 14.44 Plot the islands on the map, dot size by log(pop), edge alpha by correlation between isla…
Code 14.45 Plot #Tools vs log(pop), dot sized by log(pop), edge alpha by correlation between islands
Code 14.46 Non-centered Gaussian Process
14.47 Phylogeny regression via Gaussian Process
14.47.1 Load the data
14.48 Trim the missing data
14.49 Ordinary regression: Brain size ~ Mass + Group size
Code 14.50 Plot the implied covariance matrix vs the distance matrix
14.51 m14_10: Gaussian process using the Quadratic kernel (the covariance matrix).
14.52 OU process kernel (=Exponential distance kernel)
14.53 Posterior estimates vs prior

```
begin
  using Pkg, DrWatson
  using PlutoUI
  TableOfContents()
end
```

```
begin
    using Turing
    using Turing
    using DataFrames
    using CSV
    using Random
    using Dagitty
    using Distributions
    using StatisticalRethinking
    using StatisticalRethinking: link
    using StatisticalRethinkingPlots
    using StatsPlots
    using StatsBase
    using Logging
    using LinearAlgebra
end
```

```
  7. eval @ boot.jl:385
  8. eval @ StatisticalRethinking.jl:1
  9. (::StatisticalRethinking.var"#3#12")() @ require.jl:101
 10. macro expansion @ timing.jl:395
 11. err(f::Any, listener::Module, modname::String, file::String,
     line::Any) @ require.jl:47
 12. (::StatisticalRethinking.var"#2#11")() @ require.jl:100
 13. withpath(f::Any, path::String) @ require.jl:37
 14. (::StatisticalRethinking.var"#1#10")() @ require.jl:99
 15. listenpkg(f::Any, pkg::Base.PkgId) @ require.jl:20
 16. macro expansion @ require.jl:98
 17. __init__() @ StatisticalRethinking.jl:31
 18. run_module_init(mod::Module, i::Int64) @ loading.jl:1134
 19. register_restored_modules(sv::Core.SimpleVector, pkg::Base.PkgId,
     path::String) @ loading.jl:1122
 20. _include_from_serialized(pkg::Base.PkgId, path::String,
     ocachepath::String, depmods::Vector{Any}) @ loading.jl:1067
 21. _require_search_from_serialized(pkg::Base.PkgId,
     sourcepath::String, build_id::UInt128) @ loading.jl:1581
 22. _require(pkg::Base.PkgId, env::String) @ loading.jl:1938
 23. __require_prelocked(uuidkey::Base.PkgId, env::String)
     @ loading.jl:1812
 24. #invoke_in_world#3 @ essentials.jl:926
 25. invoke_in_world @ essentials.jl:923
 26. _require_prelocked(uuidkey::Base.PkgId, env::String)
     @ loading.jl:1803
 27. macro expansion @ loading.jl:1790
 28. macro expansion @ lock.jl:267
 29. __require(into::Module, mod::Symbol) @ loading.jl:1753
 30. #invoke_in_world#3 @ essentials.jl:926
 31. invoke_in_world @ essentials.jl:923
 32. require(into::Module, mod::Symbol) @ loading.jl:1746
 33. Other cell: line 9
```

```
using Dagitty
using Distributions
```

```
begin
    Plots.default(label=false);
    #Logging.disable_logging(Logging.Warn)
end;
```

14.1 Varying slopes by construction.

```
md" # 14.1 Varying slopes by construction."
```

Code 14.1 - 14.4

```
md" #### Code 14.1 - 14.4"
```

> Note

Julia has similar `column-first` matix order.

```
2×2 reshape(::UnitRange{Int64}, 2, 2) with eltype Int64:
 1  3
 2  4
```

```
reshape(1:4, (2,2))
```

Code 14.5 - 14.8 Simulate `a_cafe` (intercept) and `b_cafe`(slope)

```
md" ## Code 14.5 - 14.8 Simulate `a_cafe` (intercept) and `b_cafe`(slope)"
```

```
[-0.793374, -1.65825, -0.708108, -0.770274, -1.81915, -0.413504, -0.471515, -0.899277, -1
```

```
let
    a = 3.5    # average morning wait time
    b = -1     # average difference afternoon wait time
    σ_a = 1    # std dev in intercepts
    σ_b = 0.5  # std dev in slopes
    ρ = -0.7;  # correlation between intercepts and slopes

    global μ = [a, b]
    cov_ab = σ_a * σ_b * ρ
    global Σ₂ = [[σ_a^2, cov_ab] [cov_ab, σ_b^2]]
    sigmas = [σ_a, σ_b]
    P = [[1, ρ] [ρ, 1]]

    global Σ₃ = Diagonal(sigmas) * P * Diagonal(sigmas)
    N_cafes = 20
    Random.seed!(5)
    vary_effect = rand(MvNormal(μ, Σ₃), N_cafes)
    global a_cafe = vary_effect[1,:]
    global b_cafe = vary_effect[2,:];
end
```

```
2×2 Matrix{Float64}:
  1.0   -0.35
 -0.35   0.25
```

```
Σ₂
```

```
2×2 Matrix{Float64}:
  1.0   -0.35
 -0.35   0.25
```

```
Σ₃
```

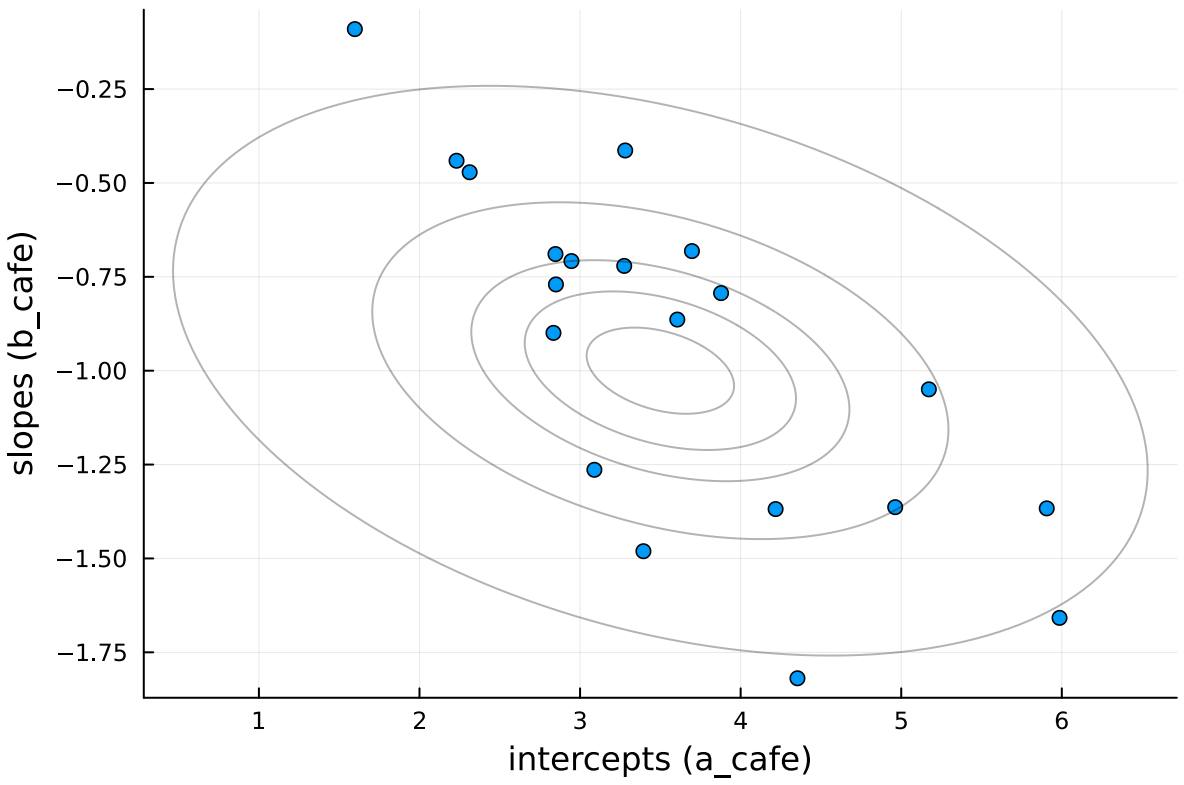
```
[3.87817, 5.98586, 2.94622, 2.8496, 4.35373, 3.28105, 2.31243, 2.83429, 3.0891, 5.17199,
```

```
a_cafe
```

```
Enter cell code...
```

Code 14.9 Plot `a_cafe`(intercepts) vs `b_cafe`(slopes) and its contour

```
md" ## Code 14.9 Plot `a_cafe`(intercepts) vs `b_cafe`(slopes) and its contour"
```




```
let
    p = scatter(a_cafe, b_cafe, xlab="intercepts (a_cafe)", ylab="slopes (b_cafe)")

    d = acos(Σ₂[1,2])
    chi = Chisq(2)

    for l ∈ (0.1, 0.3, 0.5, 0.8, 0.99)
        scale = sqrt(quantile(chi, l))
        xₜ(t) = scale*Σ₂[1,1]*cos(t + d/2) + μ[1]
        yₜ(t) = scale*Σ₂[2,2]*cos(t - d/2) + μ[2]

        plot!(xₜ, yₜ, 0, 2π, c=:black, alpha=0.3)
    end
    p
end
```

Code 14.10 Simulate observations

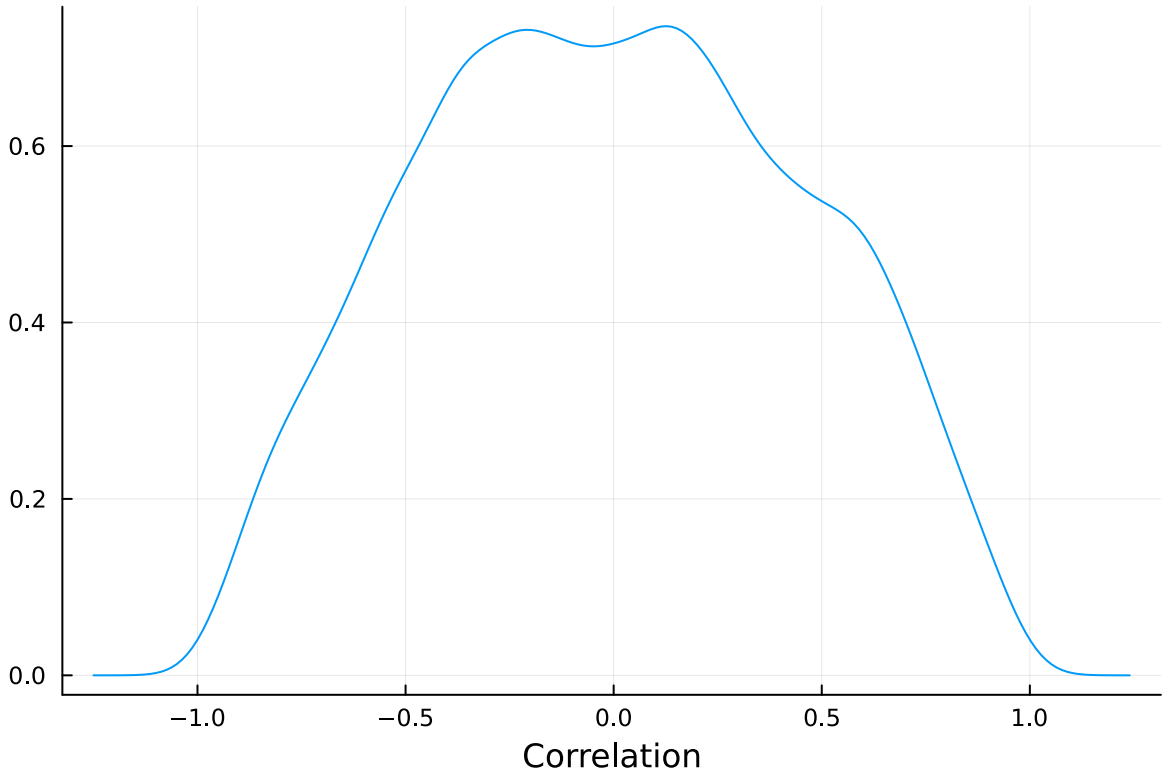
```
md" ## Code 14.10 Simulate observations"
```

	cafe	afternoon	wait
1	1	0	3.84288
2	1	1	3.35054
3	1	0	3.47475
4	1	1	4.31329
5	1	0	4.46061
6	1	1	3.21858
7	1	0	4.75314
8	1	1	2.67179
9	1	0	3.3568
10	1	1	2.92023
	more		
200	20	1	1.39984

```
let
    Random.seed!(1)
    N_cafes = 20
    N_visits = 10

    afternoon = repeat(0:1, N_visits*N_cafes ÷ 2)
    cafe_id = repeat(1:N_cafes, inner=N_visits)
    μ = a_cafe[cafe_id] + b_cafe[cafe_id] .* afternoon
    σ = 0.5
    wait = rand.(Normal.(μ, σ))
    global cafes = DataFrame(cafe=cafe_id, afternoon=afternoon, wait=wait);
end
```

Code 14.11 LKJ prior on ρ

```
md" ## Code 14.11 LKJ prior on ρ "
```




```
let
    R = rand(LKJ(2, 2), 10^4);
    density(getindex.(R, 2), xlab="Correlation")
end
```

Code 14.12 `m14_1` multilevel covariance model

```
md" ## Code 14.12 `m14_1` multilevel covariance model"
```

```
m14_1 (generic function with 2 methods)
```

```
@model function m14_1(cafe, afternoon, wait)
    a ~ Normal(5, 2)
    b ~ Normal(-1, 0.5)
    σ_cafe ~ filldist(Exponential(), 2)
    Rho ~ LKJ(2, 2)
    # build sigma matrix manually, to avoid numerical errors
#     (σ₁, σ₂) = σ_cafe
#     sc = [[σ₁^2, σ₁*σ₂] [σ₁*σ₂, σ₂^2]]
#     Σ = Rho .* sc
    # the same as above, but shorter and generic
    Σ = (σ_cafe .* σ_cafe') .* Rho
    ab ~ filldist(MvNormal([a,b], Σ), 20)
    a = ab[1,cafe]
    b = ab[2,cafe]
    μ = @. a + b * afternoon
    σ ~ Exponential()
    for i ∈ eachindex(wait)
        wait[i] ~ Normal(μ[i], σ)
    end
end
```

	Rho[1,1]	Rho[1,2]	Rho[2,1]	Rho[2,2]	a	ab[1,10]	ab[1,11]	ab[1,12]
1	1.0	-0.522936	-0.522936	1.0	3.60627	5.01258	4.45344	1.79449
2	1.0	-0.643782	-0.643782	1.0	3.25136	5.26433	4.20388	1.71686
3	1.0	-0.705805	-0.705805	1.0	3.71874	5.53875	3.97945	2.1904
4	1.0	-0.645296	-0.645296	1.0	3.61088	5.23887	4.31106	2.18245
5	1.0	-0.362138	-0.362138	1.0	3.54838	5.3092	3.94371	2.11748
6	1.0	-0.220197	-0.220197	1.0	3.72025	5.42255	4.54472	2.16845
7	1.0	-0.741257	-0.741257	1.0	3.73357	5.35305	3.88705	2.20009
8	1.0	-0.50097	-0.50097	1.0	4.14349	5.18516	4.58703	1.90888
9	1.0	-0.835074	-0.835074	1.0	4.06352	5.69799	4.03862	2.38005
10	1.0	-0.723432	-0.723432	1.0	4.22396	5.99279	4.07209	1.7115
	more							
1000	1.0	-0.72519	-0.72519	1.0	3.49954	5.26343	3.91567	2.14063

```
begin
    Random.seed!(1)
    @time m14_1_ch = sample(m14_1(cafes.cafe, cafes.afternoon, cafes.wait), NUTS(), 1000)
    m14_1_df = DataFrame(m14_1_ch);
end
```

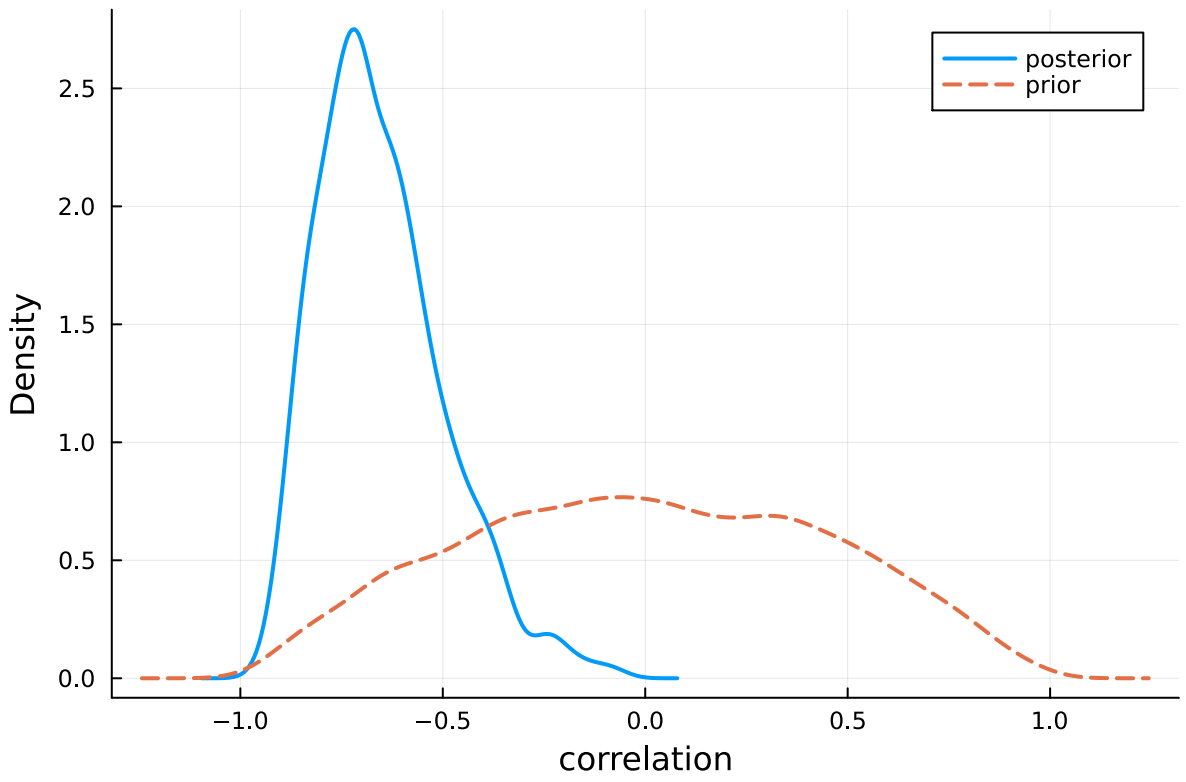
```
100%
```

```
Found initial step size
ϵ: 0.025
```

```
 16.115599 seconds (22.45 M allocations: 10.670 GiB, 5.83% gc time, 44.44% compilation time)
```

Code 14.13 Posterior vs prior of ρ

```
md" ## Code 14.13 Posterior vs prior of ρ"
```




```
let
    density(m14_1_df."Rho[1,2]", lab="posterior", lw=2)
    @show LKJ(2,2)
    R = rand(LKJ(2, 2), 10^4);
    density!(getindex.(R, 2), lab="prior", ls=:dash, lw=2)
    plot!(xlab="correlation", ylab="Density")
end
```

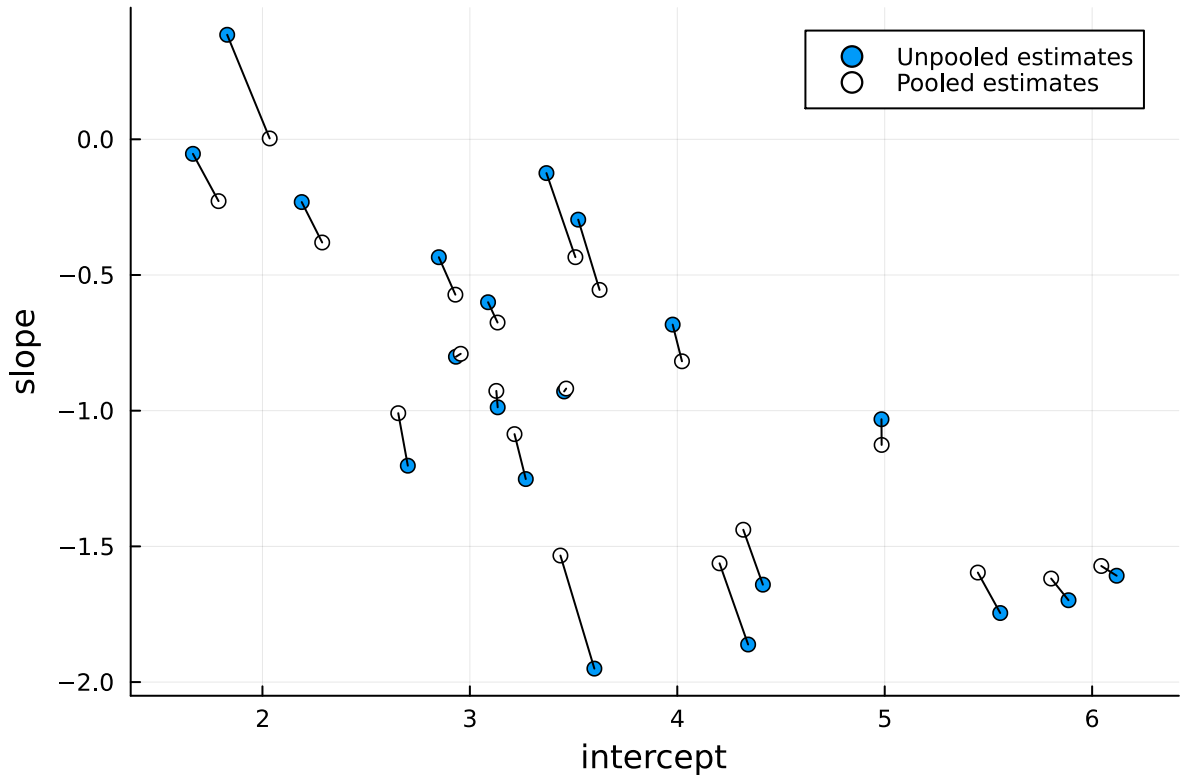
```
LKJ(2, 2) = Distributions.LKJ{Float64, Int64}(
d: 2
η: 2.0
)
```

Code 14.14 Draw posterior estimates of intercept vs slope

```
md" ## Code 14.14 Draw posterior estimates of intercept vs slope"
```

Note

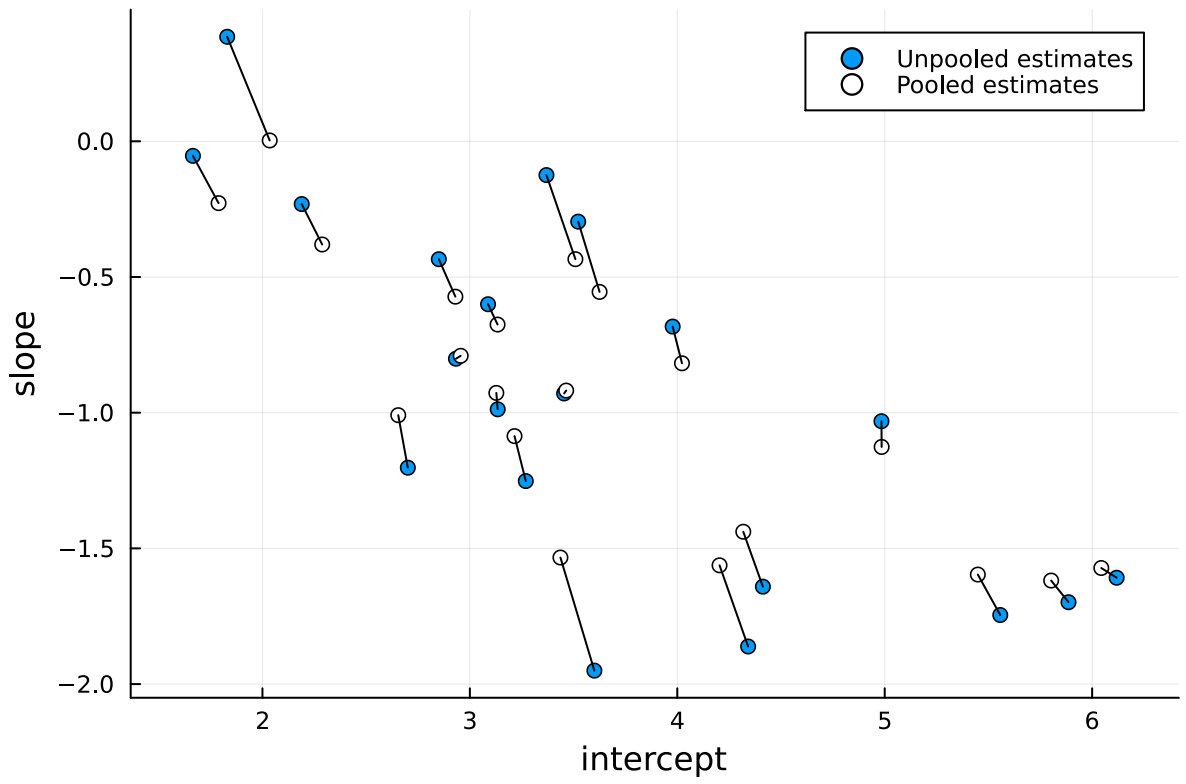
Plot differs from presented in the book due to different seed in data generation.




```
let
    N_cafes = 20
    gb = groupby(cafes[cafes.afternoon .== 0,:], :cafe)
    global a1 = combine(gb, :wait => mean).wait_mean

    gb = groupby(cafes[cafes.afternoon .== 1,:], :cafe)
    global b1 = combine(gb, :wait => mean).wait_mean .- a1

    global a2 = [mean(m14_1_df[:, "ab[1,$i]"]) for i ∈ 1:N_cafes]
    global b2 = [mean(m14_1_df[:, "ab[2,$i]"]) for i ∈ 1:N_cafes]

    xlim = extrema(a1) .+ (-0.3, 0.3)
    ylim = extrema(b1) .+ (-0.1, 0.1)

    global p_cafe = scatter(a1, b1,
        label="Unpooled estimates",
        xlab="intercept", ylab="slope",
        xlim=xlim, ylim=ylim, legend=:topright)

    scatter!(a2, b2, mc=:white, label="Pooled estimates")

    for (x1,y1,x2,y2) ∈ zip(a1, b1, a2, b2)
        plot!([x1,x2], [y1,y2], c=:black)
    end
    p_cafe
end
```

Code 14.15 Draw the contour of posterior estimates

```
md" ## Code 14.15 Draw the contour of posterior estimates"
```

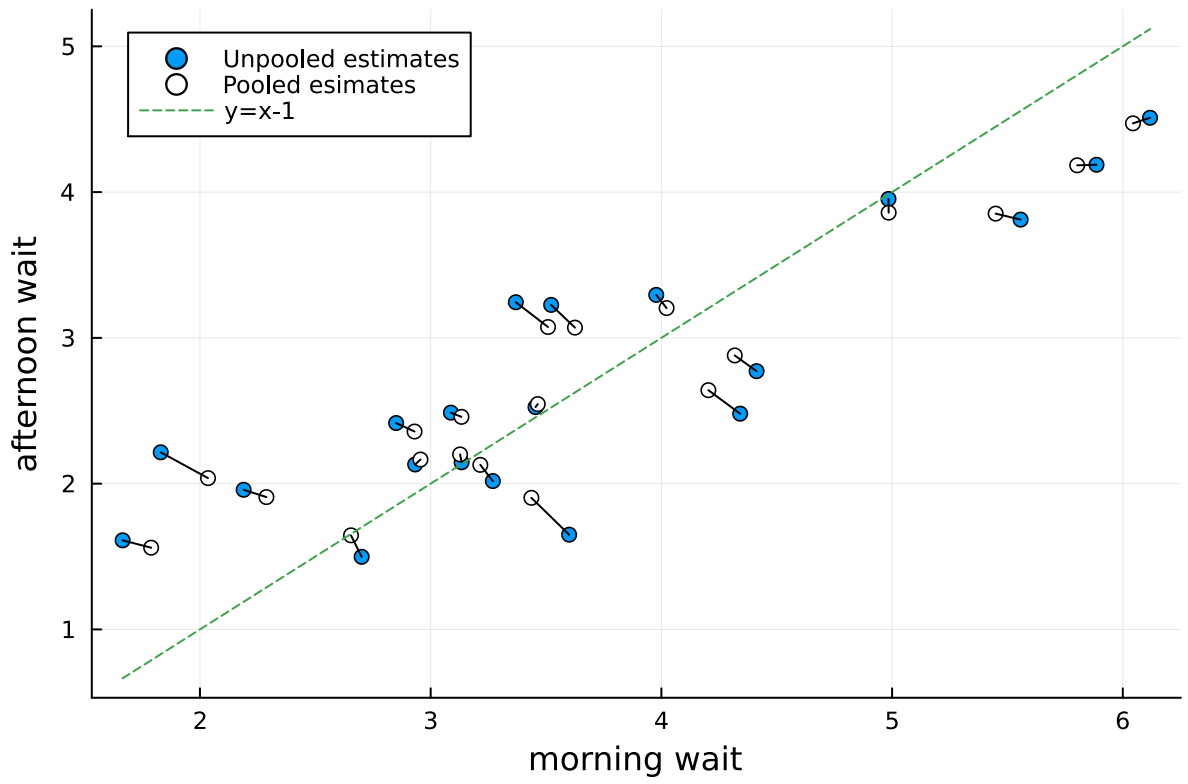



```
let
    # posterior mean
    @show ρ = mean(m14_1_df."Rho[1,2]")
    @show μ_a = mean(m14_1_df.a)
    @show μ_b = mean(m14_1_df.b)
    @show σ₁ = mean(m14_1_df."σ_cafe[1]")
    @show σ₂ = mean(m14_1_df."σ_cafe[2]")

    # draw ellipses
    @show ρ*σ₁*σ₂
    @show dt = acos(ρ*σ₁*σ₂)
    chi = Chisq(2)

    for l ∈ (0.1, 0.3, 0.5, 0.8, 0.99)
        scale = sqrt(quantile(chi, l))
        xₜ(t) = scale*σ₁^2*cos(t + dt/2) + μ_a
        yₜ(t) = scale*σ₂^2*cos(t - dt/2) + μ_b

        @show typeof(xₜ)
        plot!(xₜ, yₜ, 0, 2π, c=:black, alpha=0.3)
    end

    p_cafe
end
```

```
ρ = mean(m14_1_df.:("Rho[1,2]")) = -0.6522465771101927
μ_a = mean(m14_1_df.a) = 3.6822784498860353
μ_b = mean(m14_1_df.b) = -0.9563569801205808
σ₁ = mean(m14_1_df.:("σ_cafe[1]")) = 1.2340157814627257
σ₂ = mean(m14_1_df.:("σ_cafe[2]")) = 0.5798965785335839
ρ * σ₁ * σ₂ = -0.46674864820859646
dt = acos(ρ * σ₁ * σ₂) = 2.056407145226051
typeof(xₜ) = Main.var"workspace#138".var"#xₜ#7"{Float64, Float64, Float64, Flo
at64}
typeof(xₜ) = Main.var"workspace#138".var"#xₜ#7"{Float64, Float64, Float64, Flo
at64}
typeof(xₜ) = Main.var"workspace#138".var"#xₜ#7"{Float64, Float64, Float64, Flo
at64}
typeof(xₜ) = Main.var"workspace#138".var"#xₜ#7"{Float64, Float64, Float64, Flo
at64}
typeof(xₜ) = Main.var"workspace#138".var"#xₜ#7"{Float64, Float64, Float64, Flo
at64}
```

Code 14.16 Draw morning wait vs afternoon wait

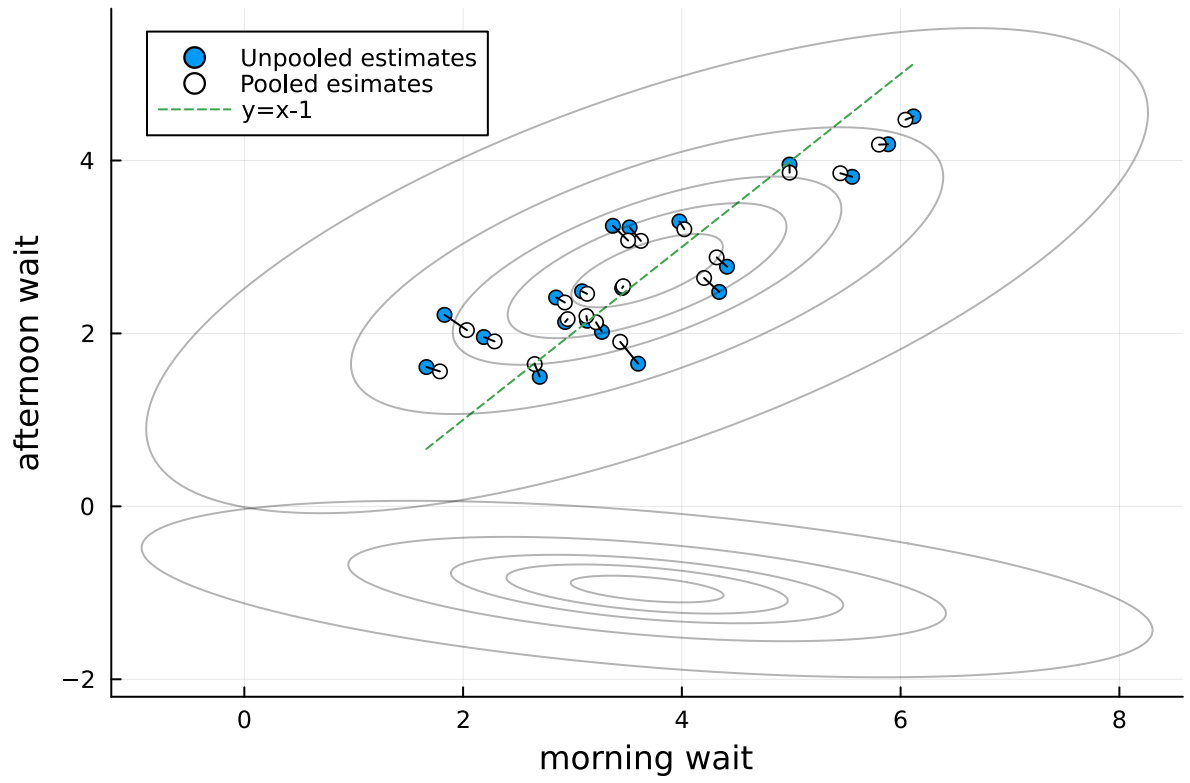
```
md" ## Code 14.16 Draw morning wait vs afternoon wait"
```




```
let
    wait_morning_1 = a1
    wait_afternoon_1 = a1 .+ b1
    wait_morning_2 = a2
    wait_afternoon_2 = a2 .+ b2

    global p = scatter(wait_morning_1, wait_afternoon_1,
        label="Unpooled estimates",
        xlab="morning wait", ylab="afternoon wait", legend=true)
    scatter!(wait_morning_2, wait_afternoon_2,
        label="Pooled esimates",
        mc=:white)

    plot!(x -> x-1, label="y=x-1", s=:dash)

    #connect the two estimates
    for (x1,y1,x2,y2) ∈ zip(wait_morning_1, wait_afternoon_1, wait_morning_2,
    wait_afternoon_2)
        plot!([x1,x2], [y1,y2], c=:black)
    end
    p
end
```

Code 14.17 Draw contour , but failed.

```
md" ## Code 14.17 Draw contour , but failed."
```




```
let
    Random.seed!(1)

    # posterior mean
    ρ = mean(m14_1_df."Rho[1,2]")
    μ_a = mean(m14_1_df.a)
    μ_b = mean(m14_1_df.b)
    σ₁ = mean(m14_1_df."σ_cafe[1]")
    σ₂ = mean(m14_1_df."σ_cafe[2]")

    Σ = [[σ₁^2, σ₁*σ₂*ρ] [σ₁*σ₂*ρ, σ₂^2]]
    μ = [μ_a, μ_b]
    v = rand(MvNormal(μ, Σ), 10^4)
    v[2,:] += v[1,:]
    @show Σ₂ = cov(v')
    μ₂ = [μ_a, μ_a+μ_b]

    # draw ellipses
    #dt = acos(Σ₂[1,2])
    dt = acos(-ρ)
    chi = Chisq(2)

    for l ∈ (0.1, 0.3, 0.5, 0.8, 0.99)
        scale = sqrt(quantile(chi, l))
        xₜ(t) = scale*Σ₂[1,1]*cos(t + dt/2) + μ₂[1]
        yₜ(t) = scale*Σ₂[2,2]*cos(t - dt/2) + μ₂[2]

        plot!(xₜ, yₜ, 0, 2π, c=:black, alpha=0.3)
    end

    p
end
```

```
Σ₂ = cov(v') = [1.5089577739474576 1.047441280622814; 1.047441280622814 0.9243859719067469]
```

14.2 Advanced varying slopes.

```
md"# 14.2 Advanced varying slopes."
```

Code 14.18 Load data and fit model m14_2

```
md" ## Code 14.18 Load data and fit model m14_2"
```

```
begin
    chimpanzees = CSV.read(sr_datadir("chimpanzees.csv"), DataFrame)
    chimpanzees.treatment = 1 .+ chimpanzees.prosoc_left .+ 2*chimpanzees.condition;
    chimpanzees.block_id = chimpanzees.block;
end;
```

```
m14_2 (generic function with 2 methods)
```

```
@model function m14_2(L, tid, actor, block_id)
    tid_len = length(levels(tid))
    act_len = length(levels(actor))
    blk_len = length(levels(block_id))
    g ~ filldist(Normal(), tid_len)

    σ_actor ~ filldist(Exponential(), tid_len)
    ρ_actor ~ LKJ(tid_len, 2)
    Σ_actor = (σ_actor .* σ_actor') .* ρ_actor
    alpha ~ filldist(MvNormal(zeros(tid_len), Σ_actor), act_len)

    σ_block ~ filldist(Exponential(), tid_len)
    ρ_block ~ LKJ(tid_len, 2)
    Σ_block = (σ_block .* σ_block') .* ρ_block
    beta ~ filldist(MvNormal(zeros(tid_len), Σ_block), blk_len)

    for i ∈ eachindex(L)
        p = logistic(g[tid[i]] + alpha[tid[i], actor[i]] + beta[tid[i],
block_id[i]])
        L[i] ~ Bernoulli(p)
    end
end
```

	alpha[1,1]	alpha[1,2]	alpha[1,3]	alpha[1,4]	alpha[1,5]	alpha[1,6]	alpha[1,7]	al
1	-0.417707	-0.314063	-0.916674	-0.229096	-0.504657	1.17184	-2.13538	0.
2	-0.476854	-0.16158	-0.88613	-0.143451	-0.521008	1.16193	-2.01501	0.
3	-0.621353	0.0456847	-1.01	0.0130383	-0.517569	1.1296	-1.99259	0.
4	-0.474075	0.0639516	-1.00353	0.0499248	-0.452563	1.10109	-2.02357	0.
5	-0.571126	0.206455	-0.864771	0.26421	-0.44174	1.10781	-1.84904	0.
6	-0.487028	0.167715	-0.9494	0.119357	-0.474304	1.10679	-1.82779	0.
7	-0.447684	0.282566	-0.867375	-0.0182152	-0.47895	1.09035	-1.99392	0.
8	-0.330291	0.278504	-0.74836	0.0818598	-0.528977	1.08922	-1.98568	0.
9	-0.261654	0.17458	-0.864757	0.113052	-0.481614	1.14328	-1.80429	0.
10	-0.363133	0.266128	-0.599063	0.220736	-0.473274	1.20038	-1.69584	0.
	more							
1000	-1.76428	3.37145	-1.34892	-0.229719	-0.69239	0.926537	0.217575	-0

```
begin
    Random.seed!(123)
    @time m14_2_ch = sample(m14_2(chimpanzees.pulled_left, chimpanzees.treatment,
    chimpanzees.actor, chimpanzees.block_id),
        Turing.HMC(0.01, 10), 1000);
    m14_2_df = DataFrame(m14_2_ch);
end
```

```
100%
```

```
 35.317963 seconds (40.75 M allocations: 6.273 GiB, 3.38% gc time, 64.47% c
ompilation time)
```

	variable	mean	min	median	max	nmissing	eltype
1	Symbol("σ_actor[1]")	1.23729	0.483076	1.18119	3.00203	0	Float64
2	Symbol("σ_actor[2]")	0.623552	0.085594	0.591163	2.16501	0	Float64
3	Symbol("σ_actor[3]")	1.48661	0.174945	1.45986	2.83405	0	Float64
4	Symbol("σ_actor[4]")	1.38866	0.442452	1.10594	7.55954	0	Float64
5	Symbol("σ_block[1]")	0.861144	0.145118	0.627499	4.66835	0	Float64
6	Symbol("σ_block[2]")	0.712646	0.198766	0.606631	2.59925	0	Float64
7	Symbol("σ_block[3]")	0.230892	0.0223226	0.128093	2.29431	0	Float64
8	Symbol("σ_block[4]")	1.1752	0.072619	0.825017	5.25396	0	Float64

```
describe(m14_2_df[:, r"σ"])
```

```
[0.445566, 0.337794, 0.43427, 1.01894, 0.741829, 0.436681, 0.302691, 1.0843]
```

```
std.(eachcol(m14_2_df[:, r"σ"]))
```

Code 14.19 `m14_3` non-centered approach via Cholesky Decomposition

```
md" ## Code 14.19 `m14_3` non-centered approach via Cholesky Decomposition"
```

```
m14_3 (generic function with 2 methods)
```

```
@model function m14_3(L, tid, actor, block_id)
    tid_len = length(levels(tid))
    act_len = length(levels(actor))
    blk_len = length(levels(block_id))
    g ~ filldist(Normal(), tid_len)

    σ_actor ~ filldist(Exponential(), tid_len)
    # LKJCholesky is not usable in Turing: https://github.com/TuringLang/Turing.jl/issues/1629
    ρ_actor ~ LKJ(tid_len, 2)
    ρ_actor_L = cholesky(Symmetric(ρ_actor)).L
    z_actor ~ filldist(MvNormal(zeros(tid_len), 1), act_len)
    alpha = (σ_actor .* ρ_actor_L) * z_actor

    σ_block ~ filldist(Exponential(), tid_len)
    ρ_block ~ LKJ(tid_len, 2)
    ρ_block_L = cholesky(Symmetric(ρ_block)).L
    z_block ~ filldist(MvNormal(zeros(tid_len), 1), blk_len)
    beta = (σ_block .* ρ_block_L) * z_block

    for i ∈ eachindex(L)
        p = logistic(g[tid[i]] + alpha[tid[i], actor[i]] + beta[tid[i], block_id[i]])
        L[i] ~ Bernoulli(p)
    end
end
```

Note

Hm, this is less stable and slower than m14_2... So if you know how to improve it - PR or open the issue in the repo

```
Chains MCMC chain (1000×106×1 Array{Float64, 3}):

Iterations        = 1:1:1000
Number of chains  = 1
Samples per chain = 1000
Wall duration     = 29.06 seconds
Compute duration  = 29.06 seconds
parameters        = g[1], g[2], g[3], g[4], σ_actor[1], σ_actor[2], σ_actor[3], σ_actor
internals         = lp, n_steps, is_accept, acceptance_rate, log_density, hamiltonian_e

Summary Statistics
    parameters      mean       std      mcse   ess_bulk   ess_tail      rhat   ess_per_
        Symbol   Float64   Float64   Float64    Float64    Float64   Float64       Floa

          g[1]    0.1060    0.2869    0.0578    25.7588    59.3582    1.0131        0.8
          g[2]    0.6767    0.3139    0.0709    18.9982    76.7602    1.0111        0.6
          g[3]   -0.1875    0.4386    0.1664     7.1864    47.1005    1.0569        0.2
          g[4]    1.5824    1.5151    0.8923     2.9944    15.2337    1.5871        0.1
    σ_actor[1]    1.2373    0.4456    0.2444     3.2382    24.4370    1.6385        0.1
    σ_actor[2]    0.6236    0.3378    0.1150     8.2140    21.2645    1.0559        0.2
    σ_actor[3]    1.4866    0.4343    0.1151    15.7506    12.2639    1.0289        0.5
    σ_actor[4]    1.3887    1.0189    0.4697     3.9312    13.8524    1.3697        0.1
  ρ_actor[1,1]    1.0000    0.0000       NaN        NaN        NaN       NaN
  ρ_actor[2,1]    0.3321    0.2932    0.0671    18.7593    29.8972    1.0401        0.6
  ρ_actor[3,1]    0.4403    0.2525    0.0549    13.5494    16.6620    1.0845        0.4
  ρ_actor[4,1]    0.3658    0.2119    0.0405    28.0090    80.0800    1.0699        0.9
  ρ_actor[1,2]    0.3321    0.2932    0.0671    18.7593    29.8972    1.0401        0.6
  ρ_actor[2,2]    1.0000    0.0000    0.0000   854.5574   888.8420    0.9990       29.4
  ρ_actor[3,2]    0.3935    0.3453    0.1977     3.1071    19.4583    1.6338        0.1
  ρ_actor[4,2]    0.3576    0.3341    0.1216     7.8928    13.0007    1.1744        0.2
  ρ_actor[1,3]    0.4403    0.2525    0.0549    13.5494    16.6620    1.0845        0.4
  ρ_actor[2,3]    0.3935    0.3453    0.1977     3.1071    19.4583    1.6338        0.1
```

```
begin
    Random.seed!(123)
    @time m14_3_ch = sample(m14_3(chimpanzees.pulled_left, chimpanzees.treatment,
    chimpanzees.actor, chimpanzees.block_id),
        Turing.HMC(0.01, 10), 1000)
    CHNS(m14_2_ch)
end
```

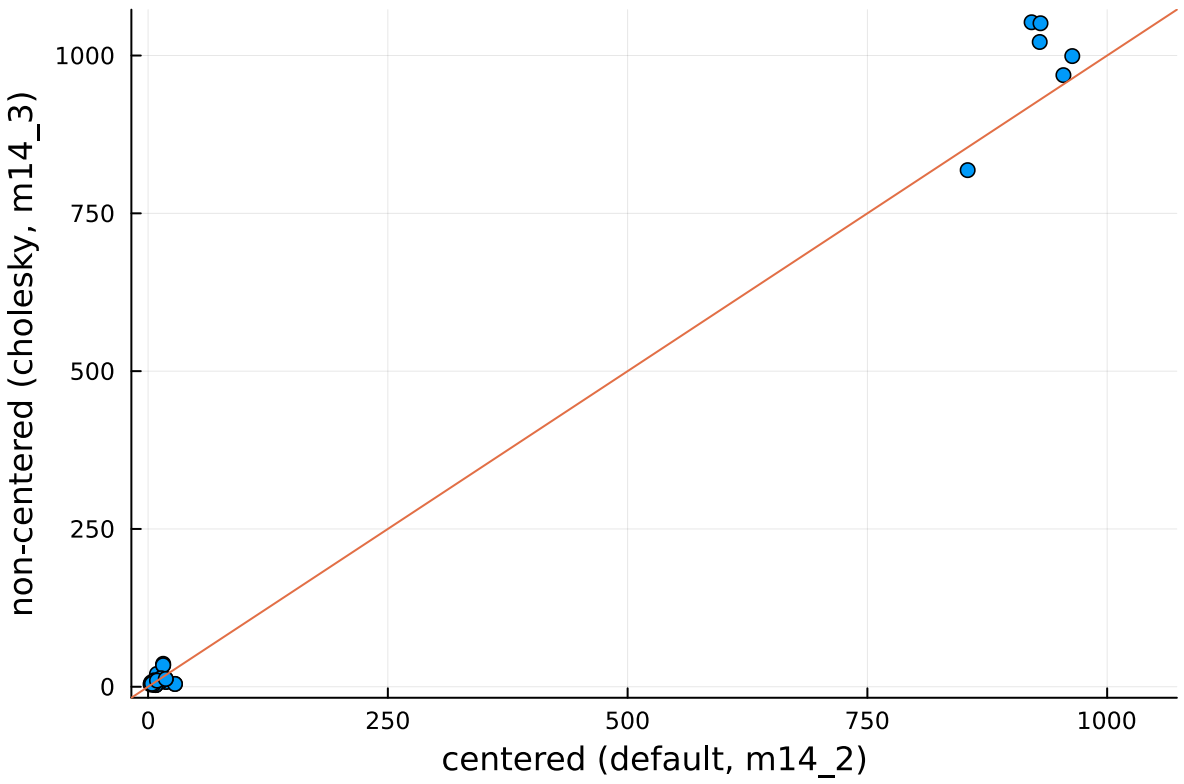
```
100%
```

```
 26.989298 seconds (26.14 M allocations: 8.244 GiB, 3.26% gc time, 51.79% c
ompilation time)
```

Code 14.20 Fig 14.6 Compare ess of `m14_2` vs `m14_3`

- not much improvement, unlike the figure in the book.

```
md" ## Code 14.20 Fig 14.6 Compare ess of `m14_2` vs `m14_3`
- not much improvement, unlike the figure in the book."
```




```
let
    t = DataFrame(ess_rhat(m14_2_ch));
    ess_2 = t.ess[occursin.(r"σ|ρ", string.(t.parameters))]
    ess_2 = filter(v -> !isnan(v), ess_2);
    t = DataFrame(ess_rhat(m14_3_ch));
    ess_3 = t.ess[occursin.(r"σ|ρ", string.(t.parameters))]
    ess_3 = filter(v -> !isnan(v), ess_3);

    bounds = extrema([ess_2; ess_3]) .+ (-20, 20)
    # xaxis=:log10, yaxis=:log10,
    scatter(ess_2, ess_3, xlim=bounds, ylim=bounds,xlab="centered (default,
    m14_2)", ylab="non-centered (cholesky, m14_3)")
    plot!(identity)
end
```

Code 14.21 range of σ for each actor by `m14_3`

```
md" ## Code 14.21 range of $$\sigma$$ for each actor by `m14_3`"
```

	variable	mean	min	median	max	nmissing	eltype
1	Symbol("σ_actor[1]")	1.31584	0.540223	1.19047	3.21493	0	Float64
2	Symbol("σ_actor[2]")	0.726816	0.0517818	0.670125	3.02735	0	Float64
3	Symbol("σ_actor[3]")	1.83558	0.183491	1.81545	3.79541	0	Float64
4	Symbol("σ_actor[4]")	1.45863	0.633517	1.37547	3.69217	0	Float64
5	Symbol("σ_block[1]")	0.60381	0.170483	0.571687	1.94563	0	Float64
6	Symbol("σ_block[2]")	0.625586	0.150397	0.550898	2.23285	0	Float64
7	Symbol("σ_block[3]")	0.15929	0.017754	0.0947639	1.52265	0	Float64
8	Symbol("σ_block[4]")	0.444059	0.0486484	0.402553	1.56902	0	Float64

```
begin
    m14_3_df = DataFrame(m14_3_ch)
    describe(m14_3_df[:, r"σ"])
end
```

```
[0.493607, 0.588746, 0.518553, 0.507875, 0.228682, 0.321564, 0.166934, 0.279272]
```

```
std.(eachcol(m14_3_df[:, r"σ"]))
```

Code 14.22 Posterior predictions (black) vs raw data (blue)

```
md" ## Code 14.22 Posterior predictions (black) vs raw data (blue)"
```

Note

The results for both models 2 and 3 are weird and mismatch with the book. So, something is wrong here. Put below both link functions for experimentations.

Plot is from model 2, because model 3 is totally off.

	actor	1	2	3	4
1	1	0.333333	0.5	0.277778	0.555556
2	2	1.0	1.0	1.0	1.0
3	3	0.277778	0.611111	0.166667	0.333333
4	4	0.333333	0.5	0.111111	0.444444
5	5	0.333333	0.555556	0.277778	0.5
6	6	0.777778	0.611111	0.555556	0.611111
7	7	0.777778	0.833333	0.944444	1.0

```
let
    gd = groupby(chimpanzees, [:actor, :treatment])
    c = combine(gd, :pulled_left => mean => :val)
    global pl = unstack(c, :actor, :treatment, :val);
end
```

```
l_fun = #11 (generic function with 1 method)
```

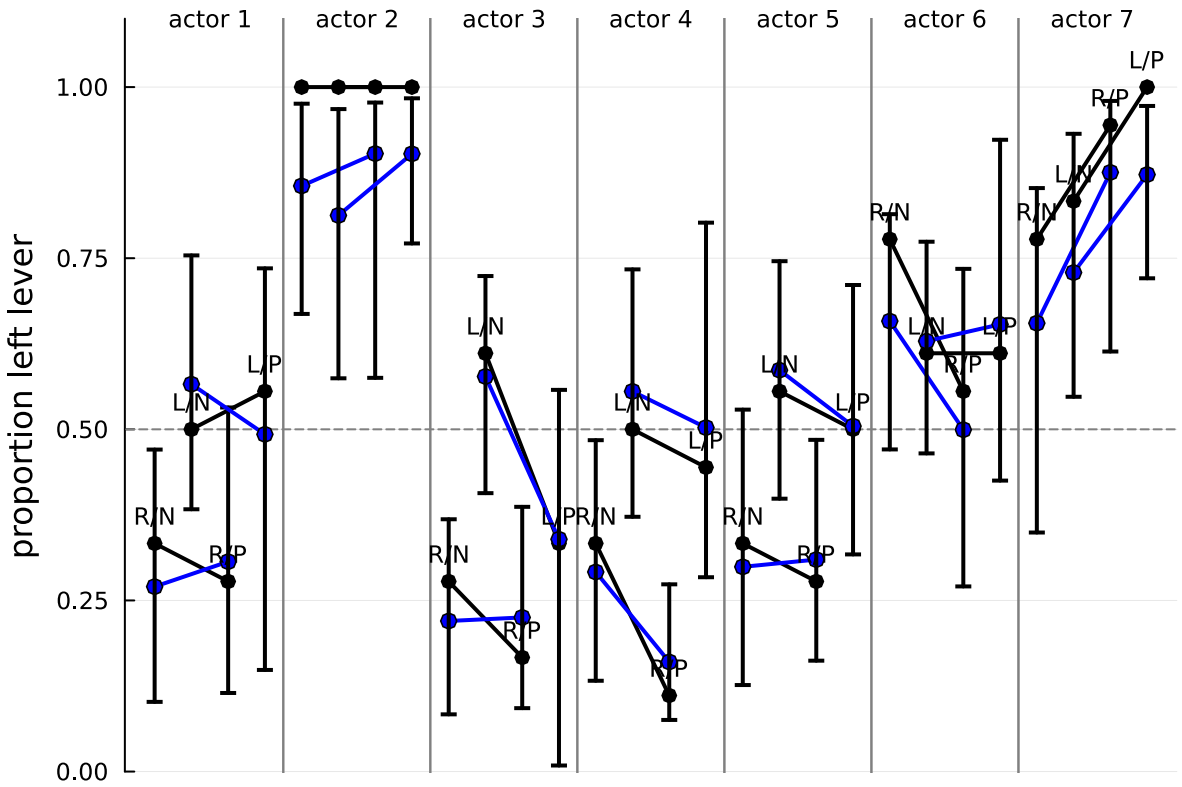
```
l_fun = (r, (ai, ti)) -> begin
    bi = 5
    g = get(r, "g[$ti]", missing)

    σ_actor = get(r, "σ_actor[$ti]", missing)
    ρ_actor = reshape(collect(r[r"ρ_actor"]), (4, 4))
    ρ_actor_L = cholesky(Symmetric(ρ_actor)).L
    z_actor = reshape(collect(r[r"z_actor"]), (4, 7))
    alpha = (σ_actor .* ρ_actor_L) * z_actor
    a = alpha[ti, ai]

    σ_block = get(r, "σ_block[$ti]", missing)
    ρ_block = reshape(collect(r[r"ρ_block"]), (4, 4))
    ρ_block_L = cholesky(Symmetric(ρ_block)).L
    z_block = reshape(collect(r[r"z_block"]), (4, 6))
    beta = (σ_block .* ρ_block_L) * z_block
    b = beta[ti, bi]

    logistic(g + a + b)
end
```

```
l_fun2 = #13 (generic function with 1 method)
```

```
#p_post = link(m14_3_df, l_fun, Iterators.product(1:7, 1:4))

l_fun2 = (r, (ai, ti)) -> begin
    bi = 5
    g = get(r, "g[$ti]", missing)
    a = get(r, "alpha[$ti,$ai]", missing)
    b = get(r, "beta[$ti,$bi]", missing)
    logistic(g + a + b)
end
```

```
p_post =
7×4 Matrix{Vector{Float64}}:
 [0.124631, 0.110113, 0.123545, 0.166727, 0.16432, 0.172442, 0.201527, 0.248427, 0.210971
 [0.136385, 0.145006, 0.215473, 0.255218, 0.299674, 0.286249, 0.343772, 0.377959, 0.29258
 [0.0795664, 0.0759377, 0.0872307, 0.105415, 0.127853, 0.116008, 0.142281, 0.178714, 0.12
 [0.146706, 0.147269, 0.210006, 0.252561, 0.311934, 0.276471, 0.279426, 0.332951, 0.28001
 [0.11545, 0.10586, 0.135229, 0.169737, 0.182867, 0.174266, 0.196543, 0.213207, 0.176675,
 [0.411022, 0.389169, 0.448105, 0.491543, 0.513123, 0.506349, 0.540227, 0.577495, 0.52145
 [0.0249174, 0.0258881, 0.0345388, 0.0407577, 0.0519391, 0.0517014, 0.0510276, 0.05939, 0
```

```
p_post = link(m14_2_df, l_fun2, Iterators.product(1:7, 1:4))
```




```
let
    p_μ = map(mean, p_post)
    p_ci = map(PI, p_post);
    rel_ci = map(idx -> (p_μ[idx]-p_ci[idx][1], p_ci[idx][2]-p_μ[idx]),
    CartesianIndices(p_ci));

    n_names = ["R/N", "L/N", "R/P", "L/P"]
    p = plot(ylims=(0, 1.1), ylab="proportion left lever", showaxis=:y,
    xticks=false)
    hline!([0.5], c=:gray, s=:dash)

    # raw data
    for actor in 1:7
        ofs = (actor-1)*4
        actor > 1 && vline!([ofs+0.5], c=:gray)
        plot!([ofs+1,ofs+3], collect(pl[actor,["1","3"]]), lw=2, m=:o, c=:black)
        plot!([ofs+2,ofs+4], collect(pl[actor,["2","4"]]), lw=2, m=:o, c=:black)
        anns = [
            (ofs+idx, pl[actor,string(idx)]+.04, (name, 8))
            for (idx,name) ∈ enumerate(n_names)
        ]
        actor != 2 && annotate!(anns)
    end

    annotate!([
        (2.5 + (idx-1)*4, 1.1, ("actor $idx", 8))
        for idx ∈ 1:7
    ])

    # posterior predictions
    for actor in 1:7
        ofs = (actor-1)*4
        actor > 1 && vline!([ofs+0.5], c=:gray)
        err = [rel_ci[actor,1], rel_ci[actor,3]]
        plot!([ofs+1,ofs+3], collect(p_μ[actor,[1,3]]), err=err, lw=2, m=:o,
    c=:blue)
        err = [rel_ci[actor,2], rel_ci[actor,4]]
        plot!([ofs+2,ofs+4], collect(p_μ[actor,[2,4]]), err=err, lw=2, m=:o,
    c=:blue)
    end

    p
end
```

14.3 Instruments and causal designs.

```
md" # 14.3 Instruments and causal designs."
```

Code 14.23 Simulate: Education has no effect on Wage and U(Unknown) has positive effect on W.

```
md" ## Code 14.23 Simulate: Education has no effect on Wage and U(Unknown) has positive effect on W."
```

	W	E	Q
1	0.939656	0.024442	-1.30672
2	0.0728279	0.55812	0.478416
3	0.0973456	0.539391	0.478416
4	0.920423	-0.13165	0.478416
5	0.390072	0.707761	-0.414151
6	-1.44529	-1.8591	-0.414151
7	-0.275619	-1.28112	-0.414151
8	-0.808482	0.450688	-0.414151
9	-0.155278	1.10511	0.478416
10	-0.656739	0.303932	0.478416
	more		
500	0.253405	0.203403	0.478416

```
let
    Random.seed!(73)
    N = 500
    U_sim = rand(Normal(), N)
    Q_sim = rand(1:4, N)
    E_sim = [rand(Normal(μ)) for μ ∈ U_sim .+ Q_sim]
    W_sim = [rand(Normal(μ)) for μ ∈ U_sim .+ 0*E_sim]

    global dat_sim1 = DataFrame(
        W=standardize(ZScoreTransform, W_sim),
        E=standardize(ZScoreTransform, E_sim),
        Q=standardize(ZScoreTransform, float.(Q_sim)),
    )
end
```

Code 14.24 `m14_4` shows Education has strong effect on Wage!

```
md" ## Code 14.24 `m14_4` shows Education has strong effect on Wage!"
```

```
m14_4 (generic function with 2 methods)
```

```
@model function m14_4(W, E)
    σ ~ Exponential()
    aW ~ Normal(0, 0.2)
    bEW ~ Normal(0, 0.5)
    μ = @. aW + bEW * E
    W ~ MvNormal(μ, σ)
end
```

	variable	mean	min	median	max	nmissing	eltype
1	:aW	0.00134619	-0.12032	0.00225114	0.123929	0	Float64
2	:bEW	0.368166	0.227714	0.367469	0.501682	0	Float64
3	:σ	0.931602	0.852432	0.930905	1.0094	0	Float64

```
begin
    m14_4_ch = sample(m14_4(dat_sim1.W, dat_sim1.E),
        NUTS(), 1000)
    m14_4_df = DataFrame(m14_4_ch)
    describe(m14_4_df)
end
```

100%

```
Found initial step size
ϵ: 0.05
```

Code 14.25 `m14_5`: Including Q amplifies the false effect of E on W.

```
md" ## Code 14.25 `m14_5`: Including Q amplifies the false effect of E on W."
```

```
m14_5 (generic function with 2 methods)
```

```
@model function m14_5(W, E, Q)
    σ ~ Exponential()
    aW ~ Normal(0, 0.2)
    bEW ~ Normal(0, 0.5)
    bQW ~ Normal(0, 0.5)
    μ = @. aW + bEW * E + bQW * Q
    W ~ MvNormal(μ, σ)
end
```

	variable	mean	min	median	max	nmissing	eltype
1	:aW	-4.47795e-5	-0.111266	-0.000409709	0.117655	0	Float64
2	:bEW	0.601418	0.437991	0.60136	0.836314	0	Float64
3	:bQW	-0.367757	-0.583443	-0.367212	-0.199085	0	Float64
4	:σ	0.88433	0.800927	0.884014	0.972633	0	Float64

```
begin
    m14_5_ch = sample(m14_5(dat_sim1.W, dat_sim1.E, dat_sim1.Q),
        NUTS(), 1000)
    m14_5_df = DataFrame(m14_5_ch)
    describe(m14_5_df)
end
```

100%

```
Found initial step size
ϵ: 0.05
```

Code 14.26 Generative model to statistical model. `m14_6`: Model residual covariance between W and E.

- The effect of E on W is almost nil/zero.

```
md" ## Code 14.26 Generative model to statistical model. `m14_6`: Model residual
covariance between W and E.
- The effect of E on W is almost nil/zero."
```

```
m14_6 (generic function with 2 methods)
```

```
@model function m14_6(W, E, Q, WE)
    σ ~ filldist(Exponential(), 2)
    ρ ~ LKJ(2, 2)
    aW ~ Normal(0, 0.2)
    aE ~ Normal(0, 0.2)
    bEW ~ Normal(0, 0.5)
    bQE ~ Normal(0, 0.5)
    μW = @. aW + bEW*E
    μE = @. aW + bQE*Q
    Σ = (σ .* σ') .* ρ
    for i ∈ eachindex(WE)
        WE[i] ~ MvNormal([μW[i], μE[i]], Σ)
    end
end
```

	aE	aW	bEW	bQE	ρ[1,1]	ρ[1,2]	ρ[2,1]	ρ[2
1	-0.275388	-0.00455866	0.0217678	0.617142	1.0	0.397875	0.397875	1.0
2	0.336914	0.0219614	0.108115	0.671408	1.0	0.46236	0.46236	1.0
3	-0.0924645	0.00669586	-0.0260834	0.553916	1.0	0.483434	0.483434	1.0
4	0.0640475	-0.0044254	0.0758127	0.724402	1.0	0.410268	0.410268	1.0
5	0.0948245	-0.0404549	0.133099	0.733443	1.0	0.424989	0.424989	1.0
6	0.237508	0.0354321	0.0656408	0.61204	1.0	0.457784	0.457784	1.0
7	-0.130972	-0.00618881	-0.0208404	0.65776	1.0	0.461704	0.461704	1.0
8	-0.0892549	-0.0250028	0.0629245	0.639256	1.0	0.454715	0.454715	1.0
9	0.145087	0.0590502	-0.0709248	0.623526	1.0	0.479662	0.479662	1.0
10	0.0674607	-0.0190673	-0.11767	0.579361	1.0	0.563057	0.563057	1.0
	more							
1000	-0.27975	-0.0112401	0.0218391	0.624	1.0	0.502842	0.502842	1.0

```
begin
    Random.seed!(1)
    # need to combine W and E here (Turing vars limitation)
    WE = [[w,e] for (w,e) ∈ zip(dat_sim1.W, dat_sim1.E)]
    m14_6_ch = sample(m14_6(dat_sim1.W, dat_sim1.E, dat_sim1.Q, WE),
        NUTS(200, 0.65, init_ε=0.003), 1000)
    m14_6_df = DataFrame(m14_6_ch);
end
```

```
100%
```

	variable	mean	min	median	max	nmissing	eltyp
1	:aE	0.00250696	-0.676259	0.00502105	0.691043	0	Float6
2	:aW	-0.000122586	-0.113971	-0.000478248	0.10315	0	Float6
3	:bEW	0.0233552	-0.186585	0.0246894	0.225334	0	Float6
4	:bQE	0.63236	0.50675	0.630933	0.754378	0	Float6
5	Symbol("ρ[1,2]")	0.453372	0.236312	0.454441	0.60589	0	Float6
6	Symbol("ρ[2,1]")	0.453372	0.236312	0.454441	0.60589	0	Float6
7	Symbol("σ[1]")	0.993746	0.851014	0.990533	1.12996	0	Float6
8	Symbol("σ[2]")	0.775185	0.710199	0.775005	0.861344	0	Float6

```
let
    # Drop cols with zero variance
    df = m14_6_df[:, Not("ρ[1,1]")][:, Not("ρ[2,2]")]
    describe(df)
end
```

Code 14.28 Simulate2: E has a positive effect on W.

```
md" ## Code 14.28 Simulate2: E has a positive effect on W."
```

	W	E	Q
1	-0.831296	0.024442	-1.30672
2	0.125797	0.55812	0.478416
3	-0.778276	0.539391	0.478416
4	2.95296	-0.13165	0.478416
5	-0.0860009	0.707761	-0.414151
6	0.811718	-1.8591	-0.414151
7	0.181046	-1.28112	-0.414151
8	-1.06115	0.450688	-0.414151
9	-0.355391	1.10511	0.478416
10	0.123301	0.303932	0.478416
	more		
500	-0.564515	0.203403	0.478416

```
let
    Random.seed!(73)

    N = 500
    U_sim = rand(Normal(), N)
    Q_sim = rand(1:4, N)
    E_sim = [rand(Normal(μ)) for μ ∈ U_sim .+ Q_sim]
    W_sim = [rand(Normal(μ)) for μ ∈ -U_sim .+ 0.2*E_sim]

    global dat_sim2 = DataFrame(
        W=standardize(ZScoreTransform, W_sim),
        E=standardize(ZScoreTransform, E_sim),
        Q=standardize(ZScoreTransform, float.(Q_sim)),
    );
end
```

> **PosDefException: matrix is not positive definite; Cholesky factorization failed.**

Stack trace

Here is what happened, the most recent locations are first:

1. **checkpositivedefinite** @ *factorization.jl:67*
2. **#cholesky!#140** @ *cholesky.jl:269*
3. **cholesky!** @ *cholesky.jl:267*
4. **cholesky!** (A::Matrix{ForwardDiff.Dual{ForwardDiff.Tag{DynamicPPL.DynamicPPLTag, Float64}, Float64, 7}}, ::LinearAlgebra.NoPivot; check::Bool) @ *cholesky.jl:301*
5. **cholesky!** @ *cholesky.jl:295*
6. **cholesky** @ *cholesky.jl:401*
7. **PDMats.PDMat**(mat::Matrix{ForwardDiff.Dual{ForwardDiff.Tag{DynamicPPL.DynamicPPLTag, Float64}, Float64, 7}}) @ *pdmat.jl:19*
8. **MvNormal** @ *mvnormal.jl:201*
9. **macro expansion** @ *Other cell: line 555*
10. Show more...

```
begin
    Random.seed!(1)
    # need to combine W and E here (Turing vars limitation)
    WE_sim2 = [[w,e] for (w,e) ∈ zip(dat_sim2.W, dat_sim2.E)]
    @time m14_6_sim2_ch = sample(m14_6(dat_sim2.W, dat_sim2.E, dat_sim2.Q,
    WE_sim2),
        NUTS(200, 0.65, init_ϵ=0.003), 1000)
    m14_6_sim2_df = DataFrame(m14_6_sim2_ch);
end
```

100%

> *Another cell defining **m14_6_sim2_df** contains errors.*

```
let
    # Drop cols with zero variance
    df = m14_6_sim2_df[:, Not("ρ[1,1]")][:, Not("ρ[2,2]")]
    describe(df)
end
```

Code 14.29 Find out which one is instrumental variable via dagitty.jl (NOT Implemented)

```
md" ## Code 14.29 Find out which one is instrumental variable via dagitty.jl (NOT
Implemented)"
```

Note

Not implemented in dagitty.jl yet.

```
DAG: {4, 3} directed simple Int64 graph with labels [:E, :Q, :U, :W])
```

```
let
    g = DAG(:Q => :E, :U => :E, :E => :W, :E => :W)
end
```

14.4 Social relations as correlated varying effects.

```
md" # 14.4 Social relations as correlated varying effects."
```

Code 14.30 Load the data

```
md" ## Code 14.30 Load the data"
```

	variable	mean	min	median	max	nmissing	eltype
1	:hidA	8.66667	1	8.0	24	0	Int64
2	:hidB	17.3333	2	18.0	25	0	Int64
3	:did	150.5	1	150.5	300	0	Int64
4	:giftsAB	3.86667	0	1.0	75	0	Int64
5	:giftsBA	5.70333	0	2.0	110	0	Int64
6	:offset	-0.12636	-1.236	-0.022	0.0	0	Float64
7	:drel1	0.0633333	0	0.0	1	0	Int64
8	:drel2	0.12	0	0.0	1	0	Int64
9	:drel3	0.213333	0	0.0	1	0	Int64
10	:drel4	0.256667	0	0.0	1	0	Int64
11	:dlndist	-2.25849	-4.068	-2.178	-1.031	0	Float64
12	:dass	0.0416867	0.0	0.015	0.552	0	Float64
13	:d0125	0.00333333	0	0.0	1	0	Int64

```
begin
    kl_dyads = CSV.read(sr_datadir("KosterLeckie.csv"), DataFrame)
    describe(kl_dyads)
end
```

	hidA	hidB	did	giftsAB	giftsBA	offset	drel1	drel2	drel3	drel4	dlndist
1	1	2	1	0	4	0.0	0	0	1	0	-2.79
2	1	3	2	6	31	-0.003	0	1	0	0	-2.817
3	1	4	3	2	5	-0.019	0	1	0	0	-1.886

```
first(kl_dyads,3)
```

Code 14.31 `m14_7` a model with dyad covariance

```
md" ## Code 14.31 `m14_7` a model with dyad covariance"
```

```
kl_data =
  (N = 300, N_households = 25, did = [1, 2, 3, 4, 5, 6, 7, 8, 9,    more ,300], hidA = [1, 1
```

```
# +
kl_data = (
    N = nrow(kl_dyads),
    N_households = maximum(kl_dyads.hidB),
    did = kl_dyads.did,
    hidA = kl_dyads.hidA,
    hidB = kl_dyads.hidB,
    giftsAB = kl_dyads.giftsAB,
    giftsBA = kl_dyads.giftsBA,
)
```

```
m14_7 (generic function with 2 methods)
```

```
@model function m14_7(N, N_households, hidA, hidB, did, giftsAB, giftsBA)
    a ~ Normal()
    #2,4 controls how flat the \rho distribution is .
    ρ_gr ~ LKJ(2, 4)
    σ_gr ~ filldist(Exponential(), 2)
    Σ = (σ_gr .* σ_gr') .* ρ_gr
    gr ~ filldist(MvNormal(Σ), N_households)

    # dyad effects (use 2 z values)
    z₁ ~ filldist(Normal(), N)
    z₂ ~ filldist(Normal(), N)
    z = [z₁ z₂]'
    σ_d ~ Exponential()
    #2,8 controls how flat the \rho distribution is
    ρ_d ~ LKJ(2, 4)
    L_ρ_d = cholesky(Symmetric(ρ_d)).L
    d = (σ_d .* L_ρ_d) * z

    λ_AB = exp.(a .+ gr[1, hidA] .+ gr[2, hidB] .+ d[1, did])
    λ_BA = exp.(a .+ gr[1, hidB] .+ gr[2, hidA] .+ d[2, did])
    for i ∈ eachindex(giftsAB)
        giftsAB[i] ~ Poisson(λ_AB[i])
        giftsBA[i] ~ Poisson(λ_BA[i])
    end
    return d
end
```

> **PosDefException: matrix is not positive definite; Cholesky factorization failed.**

Stack trace

Here is what happened, the most recent locations are first:

1. **checkpositivedefinite** @ *factorization.jl:67*
2. **#cholesky!#140** @ *cholesky.jl:269*
3. **cholesky!** @ *cholesky.jl:267*
4. **cholesky**(A::LinearAlgebra.Symmetric{ForwardDiff.Dual{ForwardDiff.Tag{DynamicPPL.DynamicPPLTag, Float64}, Float64, 12}, Matrix{ForwardDiff.Dual{ForwardDiff.Tag{DynamicPPL.DynamicPPLTag, Float64}, Float64, 12}}}, ::LinearAlgebra.NoPivot; check::Bool) @ *cholesky.jl:401*
5. **cholesky** @ *cholesky.jl:401*
6. **m14_7**(__model__::DynamicPPL.Model{typeof(Main.var"workspace#173".m14_7), (:N, :N_households, :hidA, :hidB, :did, :giftsAB, :giftsBA), (), (), Tuple{Int64, Int64, Vararg{Vector{Int64}, 5}}, Tuple{}, DynamicPPL.DefaultContext}, __varinfo__::DynamicPPL.ThreadSafeVarInfo{DynamicPPL.TypedVarInfo{@NamedTuple{a::DynamicPPL.Metadata{Dict{AbstractPPL.VarName{:a, Setfield.IdentityLens}, Int64}, Vector{Distributions.Normal{Float64}}, Vector{AbstractPPL.VarName{:a, Setfield.IdentityLens}}, Vector{ForwardDiff.Dual{ForwardDiff.Tag{DynamicPPL.DynamicPPLTag, Float64}, Float64, 12}}, Vector{Set{DynamicPPL.Selector}}}, ρ_gr::DynamicPPL.Metadata{Dict{AbstractPPL.VarName{:ρ_gr, Setfield.IdentityLens}, Int64}, Vector{Distributions.LKJ{Float64, Int64}}, Vector{AbstractPPL.VarName{:ρ_gr, Setfield.IdentityLens}}, Vector{ForwardDiff.Dual{ForwardDiff.Tag{DynamicPPL.DynamicPPLTag, Float64}, Float64, 12}}, Vector{Set{DynamicPPL.Selector}}}, σ_gr::DynamicPPL.Metadata{Dict{AbstractPPL.VarName{:σ_gr, Setfield.IdentityLens}, Int64}, Vector{Distributions.Product{Distributions.Continuous, Distributions.Exponential{Float64}, FillArrays.Fill{Distributions.Exponential{Float64}, 1, Tuple{Base.OneTo{Int64}}}}}, Vector{AbstractPPL.VarName{:σ_gr, Setfield.IdentityLens}}, Vector{ForwardDiff.Dual{ForwardDiff.Tag{DynamicPPL.DynamicPPLTag, Float64}, Float64, 12}}, Vector{Set{DynamicPPL.Selector}}}, gr::DynamicPPL.Metadata{Dict{AbstractPPL.VarName{:gr, Setfield.IdentityLens}, Int64}, Vector{DistributionsAD.VectorOfMultivariate{Distributions.Continuous, Distributions.ZeroMeanFullNormal{Tuple{Base.OneTo{Int64}}}, FillArrays.Fill{Distributions.ZeroMeanFullNormal{Tuple{Base.OneTo{Int64}}}, 1, Tuple{Base.OneTo{Int64}}}}}, Vector{AbstractPPL.VarName{:gr, Setfield.IdentityLens}}, Vector{ForwardDiff.Dual{ForwardDiff.Tag{DynamicPPL.DynamicPPLTag, Float64}, Float64, 12}}, Vector{Set{DynamicPPL.Selector}}}, z₁::DynamicPPL.Metadata{Dict{AbstractPPL.VarName{:z₁, Setfield.IdentityLens}, Int64}, Vector{DistributionsAD.TuringScalMvNormal{Vector{Float64}, Float64}}, Vector{AbstractPPL.VarName{:z₁, Setfield.IdentityLens}}, Vector{ForwardDiff.Dual{ForwardDiff.Tag{DynamicPPL.DynamicPPLTag, Float64}, Float64, 12}}, Vector{Set{DynamicPPL.Selector}}}, z₂::DynamicPPL.Metadata{Dict{AbstractPPL.VarName{:z₂, Setfield.IdentityLens}, Int64}, Vector{DistributionsAD.TuringScalMvNormal{Vector{Float64}, Float64}}, Vector{AbstractPPL.VarName{:z₂, Setfield.IdentityLens}}, Vector{ForwardDiff.Dual{ForwardDiff.Tag{DynamicPPL.DynamicPPLTag, Float64}, Float64, 12}}, Vector{Set{DynamicPPL.Selector}}}, σ_d::DynamicPPL.Metadata{Dict{AbstractPPL.VarName{:σ_d, Setfield.IdentityLens}, Int64}, Vector{Distributions.Exponential{Float64}}, Vector{AbstractPPL.VarName{:σ_d, Setfield.IdentityLens}}, Vector{ForwardDiff.Dual{ForwardDiff.Tag{DynamicPPL.DynamicPPLTag, Float64}, Float64, 12}}, Vector{Set{DynamicPPL.Selector}}}, ρ_d::DynamicPPL.Metadata{Dict{AbstractPPL.VarName{:ρ_d, Setfield.IdentityLens}, Int64}, Vector{Distributions.LKJ{Float64, Int64}}, Vector{AbstractPPL.VarName{:ρ_d, Setfield.IdentityLens}}, Vector{ForwardDiff.Dual{ForwardDiff.Tag{DynamicPPL.DynamicPPLTag, Float64}, Float64, 12}}, Vector{Set{DynamicPPL.Selector}}}}, ForwardDiff.Dual{ForwardDiff.Tag{DynamicPPL.DynamicPPLTag, Float64}, Float64, 12}},

```
Vector{Base.RefValue{ForwardDiff.Dual{ForwardDiff.Tag{DynamicPPL.DynamicPPLTa
g, Float64}, Float64, 12}}}},
__context__::DynamicPPL.SamplingContext{DynamicPPL.Sampler{Turing.Inference.N
UTS{ADTypes.AutoForwardDiff{0, Nothing}, (),
AdvancedHMC.DiagEuclideanMetric}}, DynamicPPL.DefaultContext,
Random.TaskLocalRNG}, N::Int64, N_households::Int64, hidA::Vector{Int64},
hidB::Vector{Int64}, did::Vector{Int64}, giftsAB::Vector{Int64},
giftsBA::Vector{Int64}) @ Other cell: line 16
```

```
#2,8 controls how flat the \rho distribution is
    ρ_d ~ LKJ(2, 4)
    L_ρ_d = cholesky(Symmetric(ρ_d)).L
    d = (σ_d .* L_ρ_d) * z
```

cell preview

7. Show more...

```
begin
    model = m14_7(
        kl_data.N, kl_data.N_households, kl_data.hidA, kl_data.hidB,
        kl_data.did, kl_data.giftsAB, kl_data.giftsBA
    )
    m14_7_ch = sample(model,
        NUTS(1000, 0.65, init_ϵ=0.025),
        1000)
    m14_7_df = DataFrame(m14_7_ch);
end
```

100%

Code 14.32 Check posterior estimates of giving vs receiving

```
md" #### Code 14.32 Check posterior estimates of giving vs receiving"
```

Another cell defining m14_7_df contains errors.

```
describe(m14_7_df[:, r"_gr\[(1,2|2,1|1|2)\]"])
```

Code 14.33 generated giving vs receiving

```
md" ## Code 14.33 generated giving vs receiving"
```

> *Another cell defining m14_7_df contains errors.*

```
let
    g = [
        m14_7_df.a .+ m14_7_df[!,"gr[1,$i]"]
        for i ∈ 1:25
    ]
    r = [
        m14_7_df.a .+ m14_7_df[!,"gr[2,$i]"]
        for i ∈ 1:25
    ]
    g = hcat(g...)'
    r = hcat(r...)';
    Eg_μ = mean(eachcol(exp.(g)))
    Er_μ = mean(eachcol(exp.(r)));

    # Code 14.34

    # +
    plot(xlim=(0, 8.6), ylim=(0,8.6), xlab="generalized giving", ylab="generalized receiving")
    plot!(x -> x, c=:black, s=:dash)

    for i ∈ 1:25
        gi = exp.(g[i,:])
        ri = exp.(r[i,:])
        Σ = cov([gi ri])
        μ = [mean(gi), mean(ri)]

        dt = acos(Σ[1,2])
        xₜ(t) = Σ[1,1]*cos(t + dt/2) + μ[1]
        yₜ(t) = Σ[2,2]*cos(t - dt/2) + μ[2]

        plot!(xₜ, yₜ, 0, 2π, c=:black, lw=1)
    end

    scatter!(Eg_μ, Er_μ, c=:white, msw=1.5)
end
```

Code 14.35 Estimates of dyads

```
md" ## Code 14.35 Estimates of dyads"
```

> *Another cell defining m14_7_df contains errors.*

```
describe(m14_7_df[:, r"_d"])
```

Code 14.36 Residual gifts are strongly correlated within dyads.

```
md" ## Code 14.36 Residual gifts are strongly correlated within dyads."
```

> *Another cell defining* ***model*** *and* ***m14_7_ch*** *contains errors.*

```
#
# Illustrates `generated_quantities` trick to extract values returned from the model

let
    ch = Turing.MCMCChains.get_sections(m14_7_ch, :parameters)
    d_vals = generated_quantities(model, ch)

    d_y1 = [r[1,:] for r in d_vals]
    d_y1 = hcat(d_y1...)
    d_y1 = mean.(eachrow(d_y1))

    d_y2 = [r[2,:] for r in d_vals]
    d_y2 = hcat(d_y2...)
    d_y2 = mean.(eachrow(d_y2))

    scatter(d_y1, d_y2)
end
```

14.5 Continuous categories and the Gaussian process.

```
md" # 14.5 Continuous categories and the Gaussian process."
```

Code 14.37 Load the distance matrix of 10 islands in the Kline dataset

```
md"## Code 14.37 Load the distance matrix of 10 islands in the Kline dataset"
```

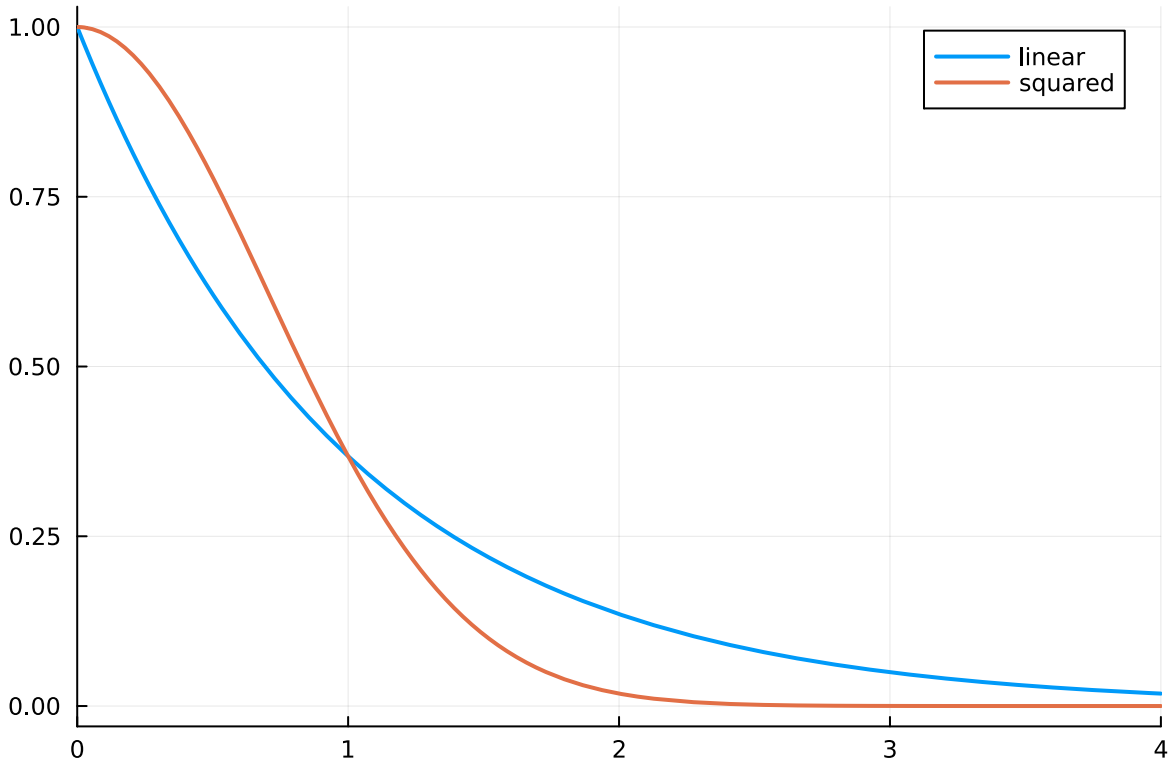
```
begin
    islandsDistMatrix = DataFrame(CSV.File("data/islandsDistMatrix.csv"))
    # drop index column
    select!(islandsDistMatrix, Not(:Column1))

    # round distances
    show(mapcols(c -> round.(c, digits=1), islandsDistMatrix), allcols=true)
end
```

10×10 DataFrame

Row	Malekula	Tikopia	Santa Cruz	Yap	Lau Fiji	Trobriand	Chuuk	M
anus	Tonga	Hawaii						
	Float64	Float64	Float64	Float64	Float64	Float64	Float64	F
loat64	Float64	Float64						
1	0.0	0.5	0.6	4.4	1.2	2.0	3.2	
2.8	1.9	5.7						
2	0.5	0.0	0.3	4.2	1.2	2.0	2.9	
2.7	2.0	5.3						
3	0.6	0.3	0.0	3.9	1.6	1.7	2.6	
2.4	2.3	5.4						
4	4.4	4.2	3.9	0.0	5.4	2.5	1.6	
1.6	6.1	7.2						
5	1.2	1.2	1.6	5.4	0.0	3.2	4.0	
3.9	0.8	4.9						
6	2.0	2.0	1.7	2.5	3.2	0.0	1.8	
0.8	3.9	6.7						
7	3.2	2.9	2.6	1.6	4.0	1.8	0.0	
1.2	4.8	5.8						
8	2.8	2.7	2.4	1.6	3.9	0.8	1.2	
0.0	4.6	6.7						
9	1.9	2.0	2.3	6.1	0.8	3.9	4.8	
4.6	0.0	5.0						
10	5.7	5.3	5.4	7.2	4.9	6.7	5.8	
6.7	5.0	0.0						

Code 14.38 Covariance function of linear or squared distance

```
md"## Code 14.38 Covariance function of linear or squared distance "
```




```
let
    plot(x -> exp(-x), xlim=(0, 4), label="linear", lw=2)
    plot!(x -> exp(-(x^2)), label="squared", lw=2)
end
```

Code 14.39 Tools vs pop and interaction

- Adapted from example here:
- https://discourse.julialang.org/t/gaussian-process-model-with-turing/42453

```
md"## Code 14.39 Tools vs pop and interaction
- Adapted from example here:
- https://discourse.julialang.org/t/gaussian-process-model-with-turing/42453"
```

(10, 10)

```
begin
    d_kline = DataFrame(CSV.File("data/Kline2.csv"))
    d_kline[:, "society"] = 1:10;
    size(d_kline)
end
```

	culture	population	contact	total_tools	mean_TU	lat	lon	lon2
1	"Malekula"	1100	"low"	13	3.2	-16.3	167.5	-12.5
2	"Tikopia"	1500	"low"	22	4.7	-12.3	168.8	-11.2
3	"Santa Cruz"	3600	"low"	24	4.0	-10.7	166.0	-14.0
4	"Yap"	4791	"high"	43	5.0	9.5	138.1	-41.9
5	"Lau Fiji"	7400	"high"	33	5.0	-17.7	178.1	-1.9

```
first(d_kline, 5)
```

	a	b	g	k[10]	k[1]	k[2]	k[3]	
1	2.60299	0.261837	0.885529	0.033361	-0.140898	0.00415115	0.0801916	0.
2	2.8613	0.262905	0.863808	-0.266686	-0.111073	0.106206	-0.104712	0.
3	1.69077	0.355334	1.28627	-0.387867	-0.129019	-0.0560739	-0.0836371	0.
4	1.17769	0.319865	0.642677	-0.29785	-0.1119	-0.0967	-0.06303	0.
5	2.54099	0.3093	1.2883	-0.159887	-0.00754474	0.259647	0.0193543	0.
6	2.7249	0.289148	1.15538	-0.175397	-0.262286	-0.171157	-0.170504	0.
7	1.66217	0.31373	0.812349	-0.393222	0.0775516	0.25637	-0.077708	0.
8	2.42316	0.289077	1.0573	-0.136535	-0.257896	0.16586	0.0617369	0.
9	2.21222	0.320821	1.33025	-0.229484	-0.0854074	-0.145393	0.136849	0.
10	1.84364	0.321997	1.07361	-0.280911	-0.0171653	-0.0128215	0.123424	0.
	more							
1000	0.986526	0.361661	1.03537	0.0156037	-0.140819	0.424612	0.270094	0.

```
begin
    d_kline_list = (
        T = d_kline.total_tools,
        P = d_kline.population,
        society = d_kline.society,
        Dmat = Matrix(islandsDistMatrix),
    )

    @model function m14_8(T, P, society, Dmat)
        η² ~ Exponential(2)
        ρ² ~ Exponential(0.5)
        a ~ Exponential()
        b ~ Exponential()
        g ~ Exponential()

        Σ = η² * exp.(-ρ² * Dmat^2) + LinearAlgebra.I * (0.01 + η²)
        k ~ MvNormal(zeros(10), Σ)
        λ = @. (a*P^b/g)*exp(k[society])
        @. T ~ Poisson(λ)
    end

    Random.seed!(1)
    @time m14_8_ch = sample(m14_8(d_kline_list.T, d_kline_list.P,
        d_kline_list.society, d_kline_list.Dmat),
        NUTS(), 1000)
    m14_8_df = DataFrame(m14_8_ch);
end
```

```
100%
```

```
Found initial step size
ϵ: 3.0517578125e-6
```

```
 16.386254 seconds (30.05 M allocations: 15.179 GiB, 7.67% gc time, 31.42%
compilation time)
```

	parameters	ess	rhat	ess_per_sec
1	:η²	269.577	1.0031	18.6044
2	:ρ²	589.179	1.0	40.6611
3	:a	328.083	1.00212	22.642
4	:b	290.992	1.00955	20.0823
5	:g	384.076	1.00226	26.5063
6	Symbol("k[1]")	445.434	1.00256	30.7408
7	Symbol("k[2]")	418.713	1.00025	28.8967
8	Symbol("k[3]")	378.894	1.00813	26.1487
9	Symbol("k[4]")	426.557	1.00145	29.4381
10	Symbol("k[5]")	442.705	1.00319	30.5525
11	Symbol("k[6]")	470.381	1.00215	32.4625
12	Symbol("k[7]")	403.4	1.00072	27.8399
13	Symbol("k[8]")	275.982	1.01063	19.0464
14	Symbol("k[9]")	329.392	1.00174	22.7323
15	Symbol("k[10]")	275.872	1.00381	19.0388

```
ess_rhat(m14_8_ch)
```

Code 14.40 Posterior estimates

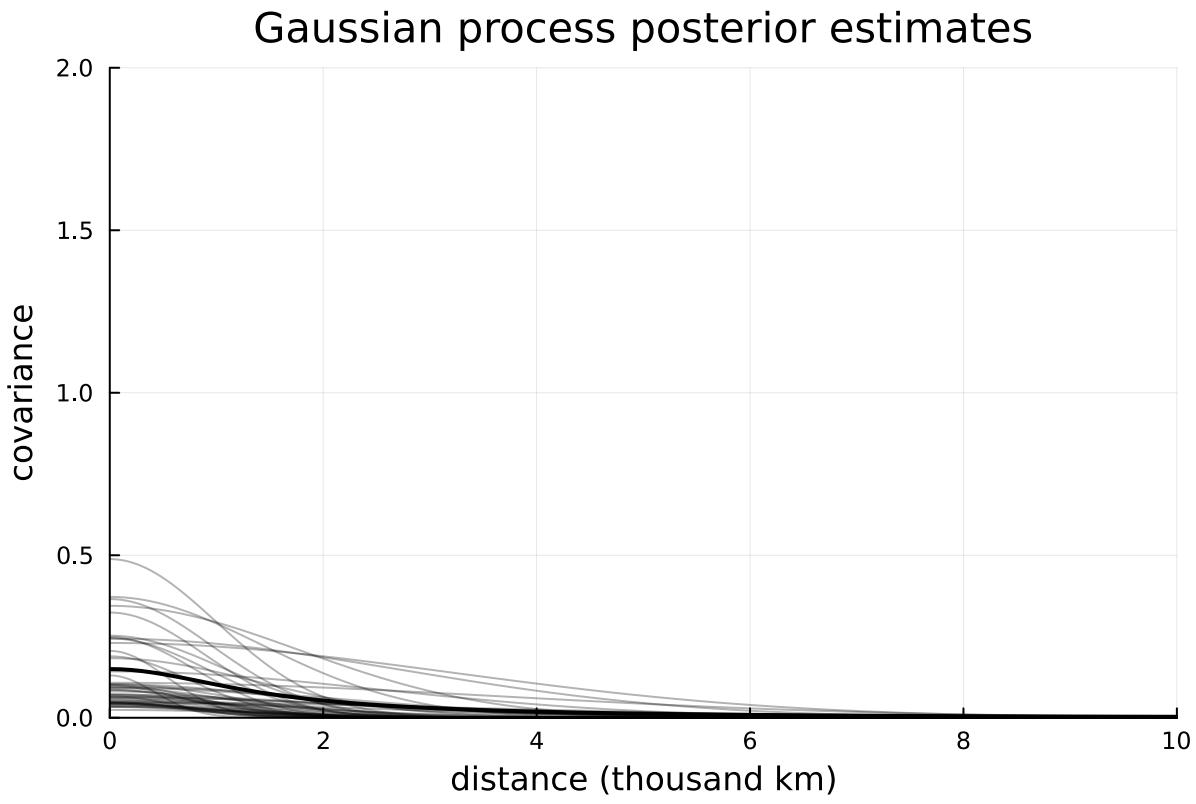
```
md"## Code 14.40 Posterior estimates"
```

	variable	mean	min	median	max	nmissing	eltyp
1	:a	1.36782	0.0373188	1.12611	6.65822	0	Float6
2	:b	0.282734	0.0590402	0.273076	0.640827	0	Float6
3	:g	0.610625	0.00868177	0.433815	5.33763	0	Float6
4	Symbol("k[10]")	-0.165326	-1.56209	-0.135623	0.664269	0	Float6
5	Symbol("k[1]")	-0.200451	-1.1554	-0.188539	0.992125	0	Float6
6	Symbol("k[2]")	0.0828482	-1.01051	0.0788567	0.968134	0	Float6
7	Symbol("k[3]")	-0.0268136	-0.725135	-0.0264817	0.809023	0	Float6
8	Symbol("k[4]")	0.362367	-0.377694	0.356998	1.15993	0	Float6
9	Symbol("k[5]")	0.0526015	-0.591652	0.0535401	0.855128	0	Float6
10	Symbol("k[6]")	-0.354165	-1.41438	-0.337611	0.498054	0	Float6
11	Symbol("k[7]")	0.157425	-0.495427	0.158401	0.742411	0	Float6
12	Symbol("k[8]")	-0.186911	-0.867707	-0.180207	0.709823	0	Float6
13	Symbol("k[9]")	0.286442	-0.329755	0.2843	0.970938	0	Float6
14	:η²	0.149839	0.0018906	0.104921	1.18721	0	Float6
15	:ρ²	0.497266	0.000293882	0.336686	3.47485	0	Float6

```
describe(m14_8_df)
```

Code 14.41 Posterior covariance function

```
md"## Code 14.41 Posterior covariance function"
```




```
begin
    x_seq = range(0, 10, length=100)
    rx_link = (r, x) -> r.η²*exp(-r.ρ²*x^2)
    pmcov = link(m14_8_df, rx_link, x_seq)
    pmcov = hcat(pmcov...)
    pmcov_μ = mean.(eachcol(pmcov))

    p_cov_vs_dist = plot(xlab="distance (thousand km)", ylab="covariance",
    title="Gaussian process posterior estimates",
        xlim=(0,10), ylim=(0,2))
    plot!(x_seq, pmcov_μ, c=:black, lw=2)

    for r ∈ first(eachrow(m14_8_df), 50)
        plot!(x -> rx_link(r, x), c=:black, alpha=0.3)
    end

    p_cov_vs_dist
end
```

Code 14.42 median estimates of K

```
md"## Code 14.42 median estimates of K"
```

```
10×10 Matrix{Float64}:
 0.219842     0.0972458   0.0917577    …  0.00757526   0.0327338    2.02649e-6
 0.0972458    0.219842    0.101474        0.00951628   0.0285937    8.70606e-6
 0.0917577    0.101474    0.219842        0.0161893    0.0182563    5.69481e-6
 0.000172756  0.00029825  0.000697199     0.0435524    3.2777e-7    3.06915e-9
 0.0628352    0.0627308   0.0467268       0.000616584  0.0862455    3.41182e-5
 0.0259856    0.0270319   0.0392913    …  0.0822654    0.000637995  3.53809e-8
 0.00350007   0.00646505  0.011003        0.0639318    4.64894e-5   1.3305e-6
 0.00757526   0.00951628  0.0161893       0.219842     7.89151e-5   2.59314e-8
 0.0327338    0.0285937   0.0182563       7.89151e-5   0.219842     2.04661e-5
 2.02649e-6   8.70606e-6  5.69481e-6      2.59314e-8   2.04661e-5   0.219842
```

```
begin
    @show η²_kline = median(m14_8_df.η²)
    @show ρ²_kline = median(m14_8_df.ρ²)
    @show K_kline = map(d -> η²_kline * exp(-ρ²_kline*d^2),
    Matrix(islandsDistMatrix))
    K_kline += LinearAlgebra.I * (0.01 + η²_kline);
    K_kline
end
```

```
η²_kline = median(m14_8_df.η²) = 0.10492089972090332
ρ²_kline = median(m14_8_df.ρ²) = 0.33668624969357663
K_kline = map((d->begin
                #= /y/home/huangyu/src/SR2TuringPluto.jl/notebooks/Chapter_14.
jl#==#f2f97686-af08-48a9-a125-e2da8ac13e18:4 =#
                η²_kline * exp(-ρ²_kline * d ^ 2)
            end), Matrix(islandsDistMatrix)) = [0.10492089972090332 0.09724580
984843191 0.0917576990166079 0.00017275602423562483 0.0628352186888312 0.02598
560072026867 0.003500066615155294 0.007575255422380179 0.03273376064143446 2.0
264869187737267e-6; 0.09724580984843191 0.10492089972090332 0.1014736376479247
3 0.0002982498511225473 0.06273079612687886 0.027031909969047453 0.00646505299
2543495 0.009516281034322862 0.02859367003246023 8.706055475627651e-6; 0.09175
76990166079 0.10147363764792473 0.10492089972090332 0.00069719869681535 0.0467
2680651853192 0.03929133671448891 0.011003005046093011 0.016189295356874106 0.
01825633024218832 5.694805365888118e-6; 0.00017275602423562483 0.0002982498511
225473 0.00069719869681535 0.10492089972090332 5.905532362214767e-6 0.01363175
3427859767 0.04648320020271371 0.04355242580637653 3.2777027794405896e-7 3.069
1521532389685e-9; 0.0628352186888312 0.06273079612687886 0.04672680651853192
5.905532362214767e-6 0.10492089972090332 0.003204246437122960 7 0.0004463093582
5300277 0.0006165843377981074 0.08624554763605904 3.4118213651646404e-5; 0.025
98560072026867 0.027031909969047453 0.03929133671448891 0.013631753427859767
0.003204246437122960 7 0.10492089972090332 0.03520298478627004 0.08226542539536
912 0.0006379951765283693 3.5380923815186285e-8; 0.003500066615155294 0.006465
052992543495 0.011003005046093011 0.04648320020271371 0.0004463093582530027 7
0.03520298478627004 0.10492089972090332 0.06393180828319123 4.648939142836051e
-5 1.3304962008933076e-6; 0.007575255422380179 0.009516281034322862 0.01618929
5356874106 0.04355242580637653 0.0006165843377981074 0.08226542539536912 0.063
93180828319123 0.10492089972090332 7.891510959709572e-5 2.593137114672748e-8;
0.03273376064143446 0.02859367003246023 0.01825633024218832 3.2777027794405896
e-7 0.08624554763605904 0.0006379951765283693 4.648939142836051e-5 7.891510959
709572e-5 0.10492089972090332 2.046612194460281e-5; 2.0264869187737267e-6 8.70
6055475627651e-6 5.694805365888118e-6 3.0691521532389685e-9 3.4118213651646404
e-5 3.5380923815186285e-8 1.3304962008933076e-6 2.593137114672748e-8 2.0466121
94460281e-5 0.10492089972090332]
```

Code 14.43 Convert K to correlation matrix

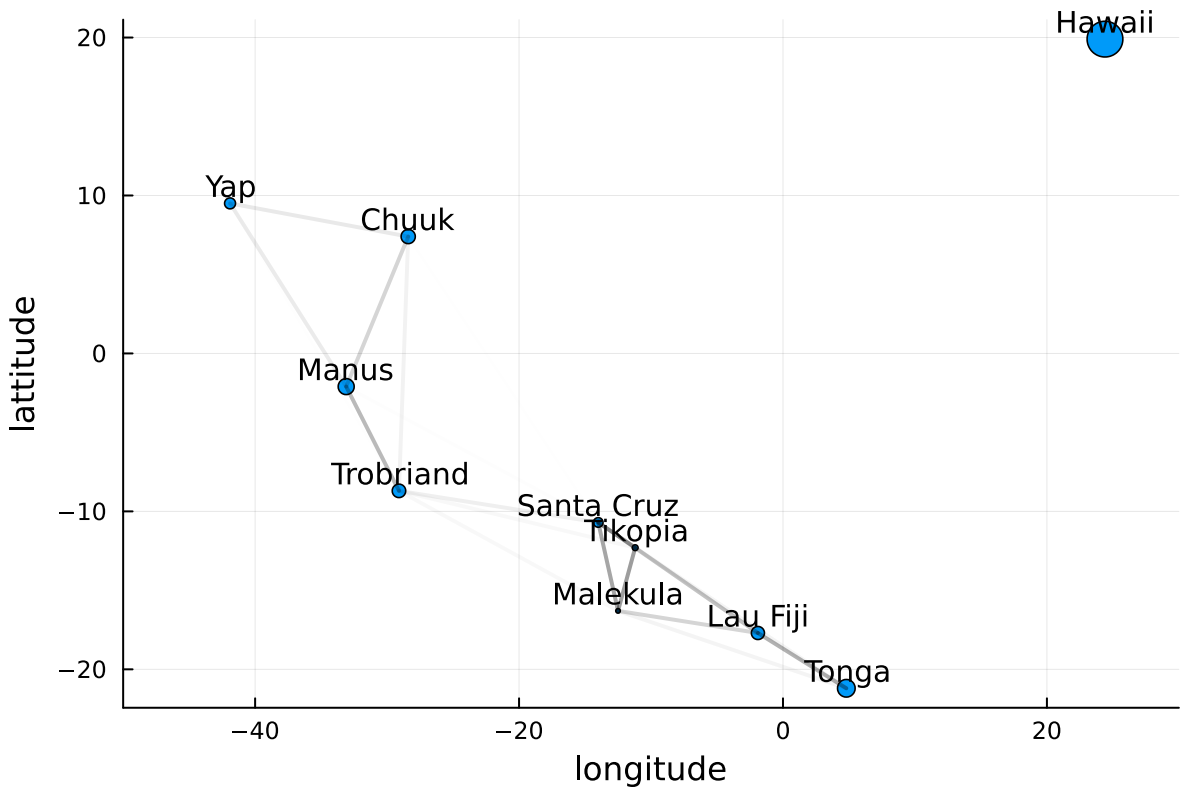
```
md"## Code 14.43 Convert K to correlation matrix"
```

```
begin
    @show Rho = round.(cov2cor(K_kline, sqrt.(diag(K_kline))), digits=2)
    cnames = ["Ml","Ti","SC","Ya","Fi","Tr","Ch","Mn","To","Ha"]
    Rho = DataFrame(Rho, :auto)
    rename!(Rho, cnames)
    show(Rho, allcols=true)
end
```

```
Rho = round.(cov2cor(K_kline, sqrt.(diag(K_kline))), digits = 2) = [1.0 0.4
4 0.42 0.0 0.29 0.12 0.02 0.03 0.15 0.0; 0.44 1.0 0.46 0.0 0.29 0.12 0.03 0.04
0.13 0.0; 0.42 0.46 1.0 0.0 0.21 0.18 0.05 0.07 0.08 0.0; 0.0 0.0 0.0 1.0 0.0
0.06 0.21 0.2 0.0 0.0; 0.29 0.29 0.21 0.0 1.0 0.01 0.0 0.0 0.39 0.0; 0.12 0.12
0.18 0.06 0.01 1.0 0.16 0.37 0.0 0.0; 0.02 0.03 0.05 0.21 0.0 0.16 1.0 0.29 0.
0 0.0; 0.03 0.04 0.07 0.2 0.0 0.37 0.29 1.0 0.0 0.0; 0.15 0.13 0.08 0.0 0.39
0.0 0.0 0.0 1.0 0.0; 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0]
10×10 DataFrame
 Row │ Ml       Ti       SC       Ya       Fi       Tr       Ch       Mn
To       Ha
     │ Float64  Float64  Float64  Float64  Float64  Float64  Float64  Float64
Float64  Float64
─────┼──────────────────────────────────────────────────────────────────────
──────────────────
   1 │     1.0      0.44     0.42     0.0      0.29     0.12     0.02     0.03
0.15     0.0
   2 │     0.44     1.0      0.46     0.0      0.29     0.12     0.03     0.04
0.13     0.0
   3 │     0.42     0.46     1.0      0.0      0.21     0.18     0.05     0.07
0.08     0.0
   4 │     0.0      0.0      0.0      1.0      0.0      0.06     0.21     0.2
0.0      0.0
   5 │     0.29     0.29     0.21     0.0      1.0      0.01     0.0      0.0
0.39     0.0
   6 │     0.12     0.12     0.18     0.06     0.01     1.0      0.16     0.37
0.0      0.0
   7 │     0.02     0.03     0.05     0.21     0.0      0.16     1.0      0.29
0.0      0.0
   8 │     0.03     0.04     0.07     0.2      0.0      0.37     0.29     1.0
0.0      0.0
   9 │     0.15     0.13     0.08     0.0      0.39     0.0      0.0      0.0
1.0      0.0
  10 │     0.0      0.0      0.0      0.0      0.0      0.0      0.0      0.0
0.0      1.0
```

Code 14.44 Plot the islands on the map, dot size by log(pop), edge alpha by correlation between islands

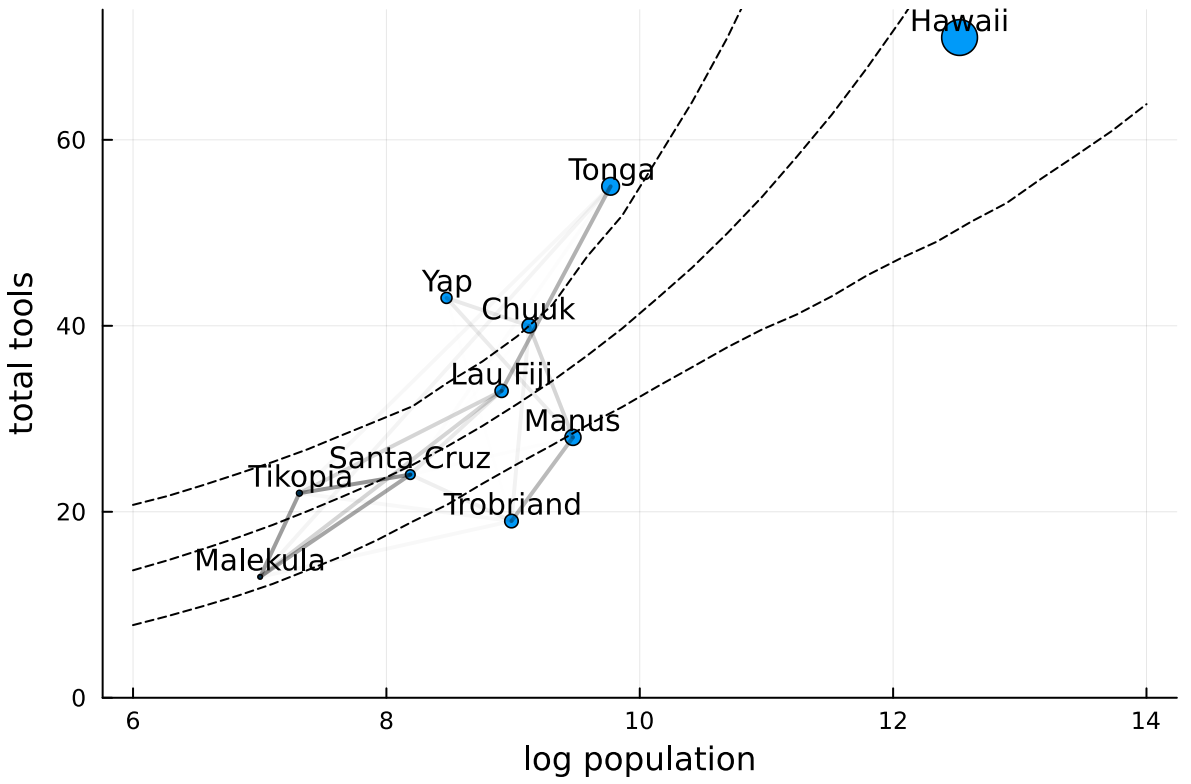
```
md"## Code 14.44 Plot the islands on the map, dot size by log(pop), edge alpha by
correlation between islands"
```




```
begin

    psize = d_kline.logpop ./ maximum(d_kline.logpop)
    psize = @. exp(psize * 1.5) - 2

    labels = map(s -> text(s, 10, :bottom), d_kline.culture)
    islands_on_map = scatter(d_kline.lon2, d_kline.lat, msize=psize*4, texts=labels,
        xlab="longitude", ylab="lattitude", xlim=(-50, 30))
    for (i, j) ∈ Base.Iterators.product(1:10, 1:10)
        i >= j && continue
        plot!(d_kline.lon2[[i,j]], d_kline.lat[[i, j]], c=:black, lw=2,
            alpha=2*(Rho[i,j]^2))
    end
islands_on_map
end
```

Code 14.45 Plot #Tools vs log(pop), dot sized by log(pop), edge alpha by log(pop), edge alpha by correlation between islands

```
md"## Code 14.45 Plot #Tools vs log(pop), dot sized by log(pop), edge alpha by
correlation between islands"
```




```
begin
    logpop_seq = range(6, 14, length=30)
    λ = link(m14_8_df,  (r, x) -> r.a * exp(x)^r.b / r.g, logpop_seq)
    λ = hcat(λ...)
    λ_median = median.(eachcol(λ))
    λ_pi = PI.(eachcol(λ))
    λ_pi = hcat(λ_pi...)'

    p_t_vs_p = scatter(d_kline.logpop, d_kline.total_tools, msize=psize*4,
        texts=labels,
        xlab="log population", ylab="total tools", ylim=(0, 74))
    plot!(logpop_seq, λ_median, c=:black, ls=:dash)
    plot!(logpop_seq, λ_pi[:,1], c=:black, ls=:dash)
    plot!(logpop_seq, λ_pi[:,2], c=:black, ls=:dash)

    # overlay correlation
    for (i, j) ∈ Base.Iterators.product(1:10, 1:10)
        i >= j && continue
        plot!(d_kline.logpop[[i,j]], d_kline.total_tools[[i, j]],
            c=:black, lw=2, alpha=2*(Rho[i,j]^2))
    end
    p_t_vs_p
end
```

Code 14.46 Non-centered Gaussian Process

```
md"## Code 14.46 Non-centered Gaussian Process"
```

	iteration	chain	η²	ρ²	a	b	g	z[1]
1	501	1	1.01645	0.0356481	0.241615	0.226015	0.0527304	-0.778275
2	502	1	1.12898	0.0300959	0.47726	0.255872	0.146684	-0.0854117
3	503	1	0.858487	1.61494	4.09854	0.19634	0.742286	-0.359778
4	504	1	0.646236	0.210158	1.34222	0.297491	0.599814	-0.652208
5	505	1	1.98839	0.29806	2.73087	0.164436	0.322229	-0.805039
6	506	1	0.25253	0.345608	2.42531	0.174716	0.330176	-0.497753
7	507	1	0.337914	0.605574	1.72533	0.175288	0.223283	-0.642683
8	508	1	7.75187	0.0397525	1.54999	0.167499	0.220966	-0.0598181
9	509	1	0.20858	0.0162421	2.97718	0.183331	0.450289	-0.708784
10	510	1	0.659069	0.0279394	1.26982	0.219457	0.255791	-1.35185

more

```
begin
    @model function m14_8nc(T, P, society, Dmat)
        η² ~ Exponential(2)
        ρ² ~ Exponential(0.5)
        a ~ Exponential()
        b ~ Exponential()
        g ~ Exponential()

        Σ = η² * exp.(-ρ² * Dmat^2) + LinearAlgebra.I * (0.01 + η²)
        L_Σ = cholesky(Σ).L
        z ~ filldist(Normal(0, 1), 10)
        k = L_Σ .* z
        λ = @. (a*P^b/g)*exp(k[society])
        @. T ~ Poisson(λ)
    end

    Random.seed!(1)
    @time m14_8nc_ch = sample(m14_8nc(d_kline_list.T, d_kline_list.P,
        d_kline_list.society, d_kline_list.Dmat),
        NUTS(), 1000);
end
```

```
100%
```

```
Found initial step size
ϵ: 3.0517578125e-6
```

```
 21.001216 seconds (27.74 M allocations: 13.825 GiB, 5.56% gc time, 53.96% compilation time)
```

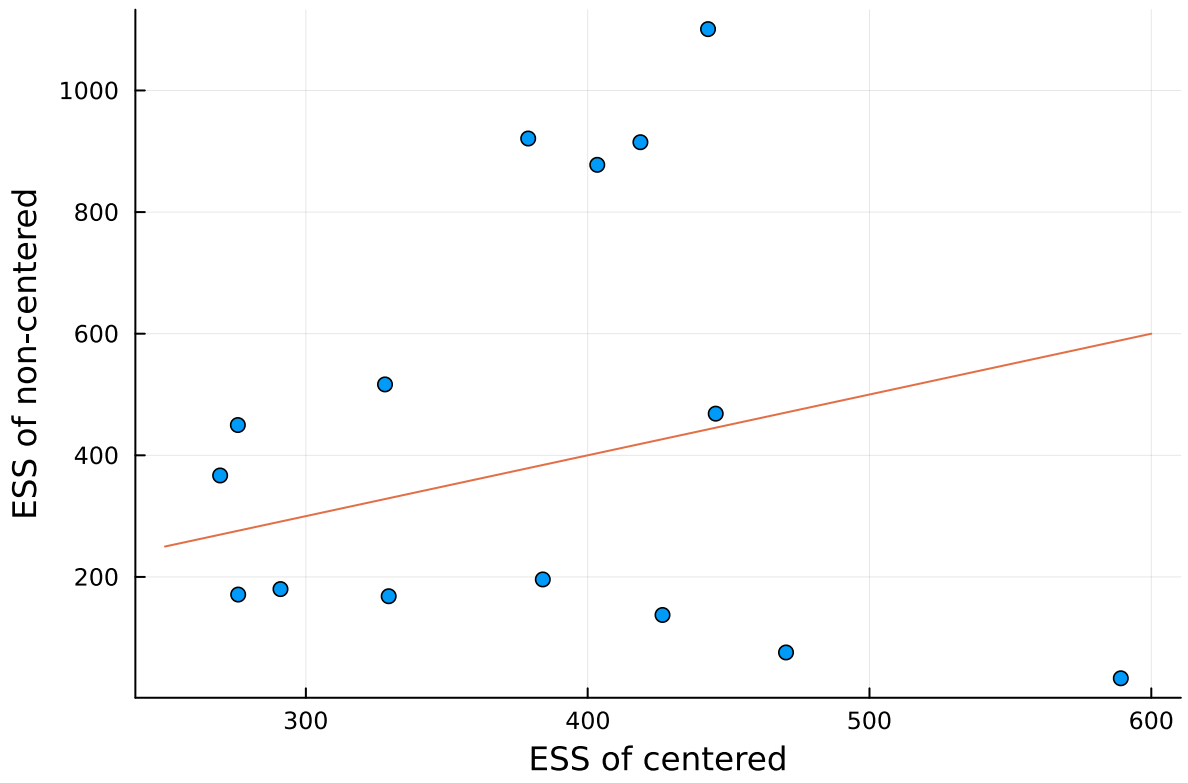
	variable	mean	min	median	max	nmissing	eltype
1	:a	1.5872	0.0424398	1.27807	6.80231	0	Float64
2	:b	0.23518	0.0822363	0.2314	0.447619	0	Float64
3	:g	0.433241	0.00760211	0.316795	3.13487	0	Float64
4	Symbol("z[10]")	-0.230531	-2.72328	-0.266017	2.75241	0	Float64
5	Symbol("z[1]")	-0.484565	-2.33059	-0.435598	0.911371	0	Float64
6	Symbol("z[2]")	-0.0367941	-2.84858	-0.0135021	3.48301	0	Float64
7	Symbol("z[3]")	-0.18049	-3.0554	-0.198315	2.72128	0	Float64
8	Symbol("z[4]")	0.636842	-3.33073	0.762636	3.34482	0	Float64
9	Symbol("z[5]")	0.034993	-2.27139	0.0572446	2.78206	0	Float64
10	Symbol("z[6]")	-0.756573	-3.79315	-0.863982	2.8377	0	Float64
11	Symbol("z[7]")	0.249736	-2.58276	0.278544	2.79834	0	Float64
12	Symbol("z[8]")	-0.411511	-3.06782	-0.484334	3.49382	0	Float64
13	Symbol("z[9]")	0.614787	-3.23198	0.70867	3.31601	0	Float64
14	:η²	1.42835	0.00744645	0.855348	11.464	0	Float64
15	:ρ²	0.218971	0.00014614	0.0266166	3.17854	0	Float64

```
1 describe(DataFrame(m14_8nc_ch))
```

m14_8nc_ch_ess_rhat =

	parameters	ess	rhat	ess_per_sec
1	:η²	366.832	1.00056	20.0334
2	:ρ²	33.3269	1.052	1.82005
3	:a	516.596	0.999557	28.2123
4	:b	180.001	1.01173	9.83019
5	:g	195.874	1.0051	10.6971
6	Symbol("z[1]")	468.503	1.00555	25.5859
7	Symbol("z[2]")	914.909	0.999489	49.965
8	Symbol("z[3]")	920.991	1.0046	50.2972
9	Symbol("z[4]")	137.426	1.01548	7.5051
10	Symbol("z[5]")	1100.9	1.00176	60.1221
11	Symbol("z[6]")	75.7905	1.03507	4.13907
12	Symbol("z[7]")	877.718	1.00483	47.9339
13	Symbol("z[8]")	171.004	1.01306	9.33887
14	Symbol("z[9]")	168.264	1.01601	9.18923
15	Symbol("z[10]")	449.791	1.01215	24.564

```
1 m14_8nc_ch_ess_rhat = ess_rhat(m14_8nc_ch)
```




```
let
    scatter(ess_rhat(m14_8_ch)[:, :ess], ess_rhat(m14_8nc_ch)[:, :ess],
        xlabel="ESS of centered", ylabel="ESS of non-centered")
    plot!([250, 600], [250, 600])
end
```

14.47 Phylogeny regression via Gaussian Process

```
md"## 14.47 Phylogeny regression via Gaussian Process"
```

14.47.1 Load the data

```
md" ### 14.47.1 Load the data"
```

d =

	name	genus	species	subspecies	spp_
1	"Allenopithecus_nigroviridis"	"Allenopithecus"	"nigroviridis"	missing	1
2	"Allocebus_trichotis"	"Allocebus"	"trichotis"	missing	2
3	"Alouatta_belzebul"	"Alouatta"	"belzebul"	missing	3
4	"Alouatta_caraya"	"Alouatta"	"caraya"	missing	4
5	"Alouatta_guariba"	"Alouatta"	"guariba"	missing	5
6	"Alouatta_palliata"	"Alouatta"	"palliata"	missing	6
7	"Alouatta_pigra"	"Alouatta"	"pigra"	missing	7
8	"Alouatta_sara"	"Alouatta"	"sara"	missing	8
9	"Alouatta_seniculus"	"Alouatta"	"seniculus"	missing	9
10	"Aotus_azarai"	"Aotus"	"azarai"	missing	10
	more				
301	"Varecia_variegata_variegata"	"Varecia"	"variegata"	"variegata"	301

```
d = DataFrame(CSV.File("data/Primates301.csv", missingstring="NA"))
```

	variable	mean	min	median	
1	:name	nothing	"Allenopithecus_nigroviridis"	nothing	"Varecia_var
2	:genus	nothing	"Allenopithecus"	nothing	"Varecia"
3	:species	nothing	"abelii"	nothing	"zaza"
4	:subspecies	nothing	"alaotrensis"	nothing	"verus"
5	:spp_id	151.0	1	151.0	301
6	:genus_id	34.186	1	36.0	68
7	:social_learning	2.30049	0	0.0	214
8	:research_effort	38.7634	1	16.0	755
9	:brain	68.4932	1.63	58.55	491.27
10	:body	6795.18	31.23	3553.5	130000.0
	more				
16	:maternal_investment	478.64	99.99	401.35	1492.3

```
describe(d)
```

14.48 Trim the missing data

```
md"## 14.48 Trim the missing data"
```

```
["Allenopithecus_nigroviridis", "Alouatta_belzebul", "Alouatta_caraya", "Alouatta_guarib
```

```
begin
    dstan = d[completecases(d, ["group_size", "body", "brain"]), :]
    spp_obs = dstan.name;
end
```

14.49 Ordinary regression: Brain size ~ Mass + Group size

- The σ_sq estimate (0.22) is higher than the book (0.05).
- Other estimates are similar.

```
md" ## 14.49 Ordinary regression: Brain size ~ Mass + Group size
- The σ_sq estimate (0.22) is higher than the book (0.05).
- Other estimates are similar."
```

	variable	mean	min	median	max	nmissing	eltype
1	:a	-5.5118e-5	-0.0525152	-4.54075e-5	0.0606643	0	Float64
2	:bG	0.123264	0.0538936	0.122837	0.192069	0	Float64
3	:bM	0.89316	0.816441	0.894246	0.96223	0	Float64
4	:σ_sq	0.215283	0.181315	0.215428	0.261569	0	Float64

```
begin
    dat_list = (
        N_spp = nrow(dstan),
        M = standardize(ZScoreTransform, log.(dstan.body)),
        B = standardize(ZScoreTransform, log.(dstan.brain)),
        G = standardize(ZScoreTransform, log.(dstan.group_size)),
    )

    @model function m14_9(N_spp, M, B, G)
        σ_sq ~ Exponential()
        bM ~ Normal(0, 0.5)
        bG ~ Normal(0, 0.5)
        a ~ Normal()
        μ = @. a + bM*M + bG * G
        B ~ MvNormal(μ, σ_sq)
    end

    Random.seed!(1)
    @time m14_9_ch = sample(m14_9(dat_list...), NUTS(), 1000)
    m14_9_df = DataFrame(m14_9_ch)
    describe(m14_9_df)
end
```

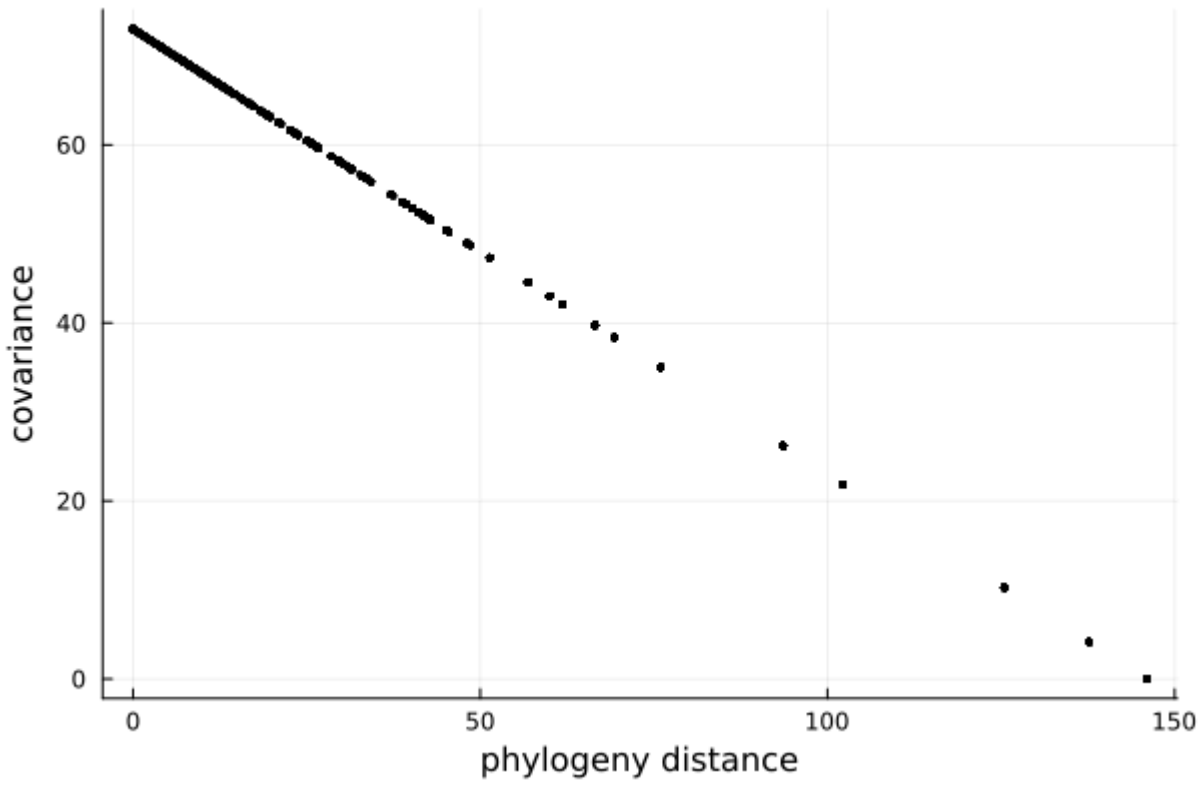
```
100%
```

```
Found initial step size
ϵ: 0.00625
```

```
  4.544652 seconds (3.74 M allocations: 409.716 MiB, 3.05% gc time, 77.41%
compilation time)
```

Code 14.50 Plot the implied covariance matrix vs the distance matrix

```
md"## Code 14.50 Plot the implied covariance matrix vs the distance matrix"
```




```
begin
    cov_mat = DataFrame(CSV.File("data/Primates301_vcov_matrix.csv"))
    dist_mat = DataFrame(CSV.File("data/Primates301_distance_matrix.csv"))

    # Drop index columns
    select!(cov_mat, Not(:Column1))
    select!(dist_mat, Not(:Column1));

    p_dist_vs_cov = scatter(Matrix(dist_mat), Matrix(cov_mat), c=:black, ms=2,
        xlabel="phylogeny distance", ylabel="covariance")
    #display("image/png", p_dist_vs_cov)
end
```

	Allenopithecus_nigroviridis	Cercopithecus_albogularis	Cercopithecus_ascanius	Cercopith
1	0.0	23.7898	23.7898	23.7898
2	23.7898	0.0	12.8584	15.7804
3	23.7898	12.8584	0.0	15.7804

```
first(dist_mat, 3)
```

	Allenopithecus_nigroviridis	Cercopithecus_albogularis	Cercopithecus_ascanius	Cercopith
1	73.003	61.1081	61.1081	61.1081
2	61.1081	73.003	66.5738	65.1128
3	61.1081	66.5738	73.003	65.1128
4	61.1081	65.1128	65.1128	73.003
5	61.1081	65.1128	65.1128	72.7287

```
first(cov_mat, 5)
```

14.51 `m14_10`: Gaussian process using the Quadratic kernel (the covariance matrix).

- All estimates are similar to the book.
- Effect of group size is now insignificant!

```
md"## 14.51 `m14_10`: Gaussian process using the Quadratic kernel (the covariance matrix).
- All estimates are similar to the book.
- Effect of group size is now insignificant!"
```

	variable	mean	min	median	max	nmissing	eltype
1	:a	-0.199958	-0.792692	-0.200044	0.431724	0	Float64
2	:bG	-0.0131494	-0.0761548	-0.013531	0.040789	0	Float64
3	:bM	0.699238	0.585403	0.700427	0.809172	0	Float64
4	:σ_sq	0.161629	0.11643	0.160304	0.248758	0	Float64

```
begin
    # reorder the covariance matrix so that the rows/columns match the rest of the
    data.
    V_inds = [
        findfirst(x -> x == n, names(cov_mat))
        for n in spp_obs
    ];

    V_dat = Matrix(cov_mat[V_inds, V_inds])
    #convert it into correlation matrix.
    R = V_dat ./ maximum(V_dat);

    @model function m14_10(N_spp, M, B, G, R)
        σ_sq ~ Exponential()
        bM ~ Normal(0, 0.5)
        bG ~ Normal(0, 0.5)
        a ~ Normal()
        μ = @. a + bM*M + bG * G
        Σ = R * σ_sq
        B ~ MvNormal(μ, Σ)
    end

    Random.seed!(1)
    @time m14_10_ch = sample(m14_10(dat_list..., R), NUTS(), 1000)
    m14_10_df = DataFrame(m14_10_ch)
    describe(m14_10_df)
end
```

```
100%
```

```
Found initial step size
ϵ: 0.025
```

```
 22.899474 seconds (3.68 M allocations: 14.041 GiB, 3.29% gc time, 15.73% c
ompilation time)
```

```
[1, 107, 108, 109, 110, 111, 113, 120, 128, 165, 114, 116, 117, 34, 35, 174, 199, 200, 130
```

```
V_inds
```

```
3×151 Matrix{Float64}:
 73.003   26.1912  26.1912  26.1912  …  51.5926  51.5926  51.5926  51.5926  0.0
 26.1912  73.003   69.1148  68.4338     26.1912  26.1912  26.1912  26.1912  0.0
 26.1912  69.1148  73.003   68.4338     26.1912  26.1912  26.1912  26.1912  0.0
```

```
V_dat[1:3, :]
```

```
["Allenopithecus_nigroviridis", "Alouatta_belzebul", "Alouatta_caraya", "Alouatta_guarib
```

```
spp_obs
```

14.52 OU process kernel (=Exponential distance kernel)

- the \eta estimate (1.01) is similar in PyMC3 (1.08), but both are higher than the book(0.03)

```
md"## 14.52 OU process kernel (=Exponential distance kernel)
- the \eta estimate (1.01) is similar in PyMC3 (1.08), but both are higher than the
book(0.03)."
```

```
151×151 Matrix{Float64}:
 0.0       0.641231   0.641231   0.641231   …  0.293281   0.293281   0.293281   1.0
 0.641231  0.0        0.053261   0.0625898     0.641231   0.641231   0.641231   1.0
 0.641231  0.053261   0.0        0.0625898     0.641231   0.641231   0.641231   1.0
 0.641231  0.0625898  0.0625898  0.0           0.641231   0.641231   0.641231   1.0
 0.641231  0.0468669  0.053261   0.0625898     0.641231   0.641231   0.641231   1.0
 0.641231  0.0468669  0.053261   0.0625898  …  0.641231   0.641231   0.641231   1.0
 0.641231  0.053261   0.0395391  0.0625898     0.641231   0.641231   0.641231   1.0
 ⋮                                          ⋱                                  ⋮
 0.293281  0.641231   0.641231   0.641231   …  0.156999   0.0640177  0.0640177  1.0
 0.293281  0.641231   0.641231   0.641231      0.0719557  0.156999   0.156999   1.0
 0.293281  0.641231   0.641231   0.641231      0.0        0.156999   0.156999   1.0
 0.293281  0.641231   0.641231   0.641231      0.156999   0.0        0.0478578  1.0
 0.293281  0.641231   0.641231   0.641231      0.156999   0.0478578  0.0        1.0
 1.0       1.0        1.0        1.0        …  1.0        1.0        1.0        0.0
```

```julia
begin
    # reorder the distance matrix so that the rows/columns match the rest of the
    data.
    D_inds = [
        findfirst(x -> x == n, names(dist_mat))
        for n in spp_obs
    ];

    D_dat = Matrix(dist_mat[D_inds, D_inds])
    # turn it into correlation matrix.
    D_dat ./= maximum(D_dat);
end
```

	variable	mean	min	median	max	nmissing	eltype
1	:a	-0.0675645	-1.2731	-0.0685745	1.25081	0	Float64
2	:bG	0.0972907	-0.567878	0.0993189	0.662547	0	Float64
3	:bM	0.752272	0.0751662	0.752781	1.32438	0	Float64
4	:η²	1.01405	1.0	1.00962	1.11629	0	Float64
5	:ρ²	3.13306	3.0	3.10447	3.71449	0	Float64

```julia
begin

    @model function m14_11(N_spp, M, B, G, D_dat)
        bM ~ Normal(0, 0.5)
        bG ~ Normal(0, 0.5)
        a ~ Normal()
        μ = @. a + bM*M + bG * G

        η² ~ truncated(Normal(1, 0.25), lower=1)
        ρ² ~ truncated(Normal(3, 0.25), lower=3)

        Σ = η² * exp.(-ρ² * D_dat) + LinearAlgebra.I * (0.01 + η²)
        B ~ MvNormal(μ, Σ)
    end

    Random.seed!(1)
    @time m14_11_ch = sample(m14_11(dat_list..., D_dat), NUTS(500, 0.65,
    init_ϵ=0.4), 4000)
    m14_11_df = DataFrame(m14_11_ch)
    describe(m14_11_df)
end
```

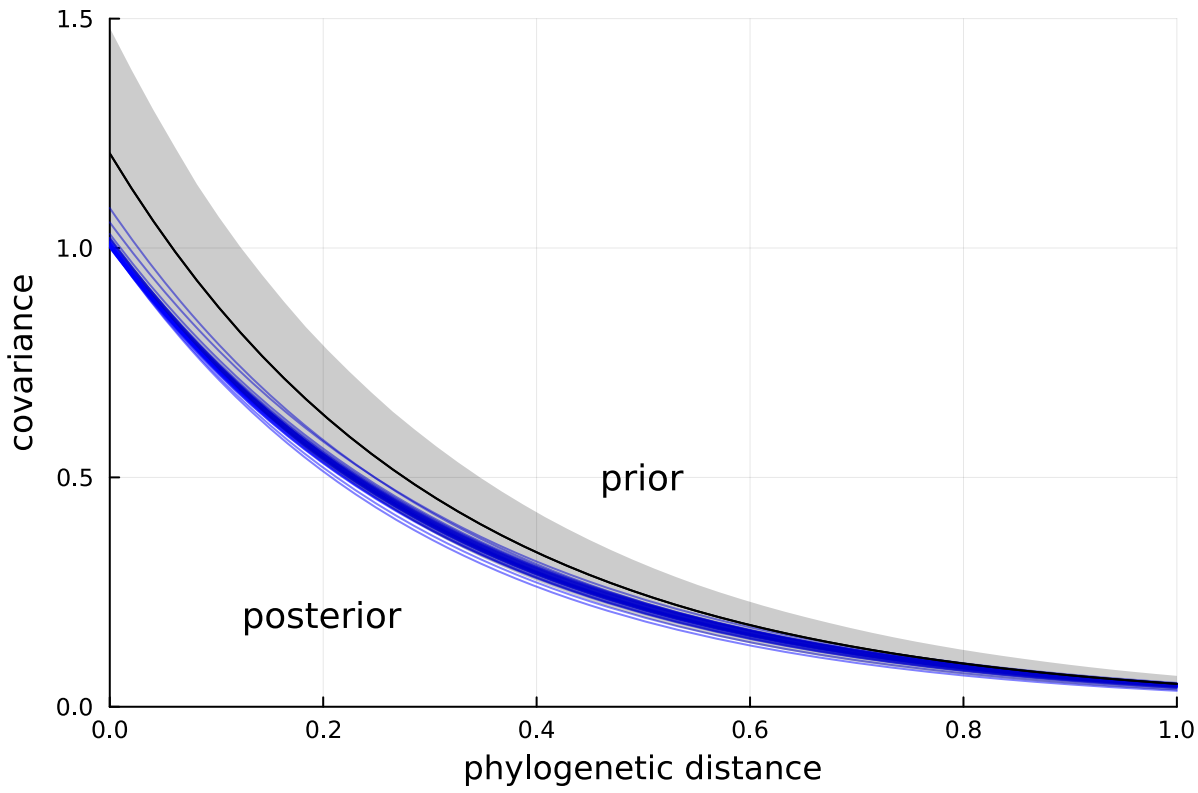
```
40%
```

	a	bG	bM	η²	ρ²
1	0.22797	0.0462125	0.882097	1.0123	3.03843
2	0.470245	0.161446	0.900867	1.05593	3.0121
3	0.417907	0.29114	0.507619	1.00533	3.19625
4	0.235574	0.662547	0.807119	1.00408	3.14357
5	-0.0101257	0.0209207	0.710142	1.00616	3.10534
6	-0.301817	-0.020667	0.58236	1.02259	3.10659
7	-0.501224	0.0662104	0.832912	1.00502	3.07052
8	-0.120246	-0.0412677	0.690122	1.01005	3.10297
9	-0.037833	-0.0630879	0.574009	1.00501	3.05086
10	-0.00906637	-0.107797	0.895472	1.00686	3.01282
	more				
30	-0.0255108	-0.102753	0.5992	1.00633	3.11237

```
first(m14_11_df, 30)
```

14.53 Posterior estimates vs prior

```
md"## 14.53 Posterior estimates vs prior"
```




```
begin
    plot(xlim=(0, maximum(D_dat)), ylim=(0, 1.5),
        xlab="phylogenetic distance", ylab="covariance")
    # posterior estimates
    for r in first(eachrow(m14_11_df), 30)
        plot!(x -> r.η² * exp(-r.ρ²*x), c=:blue, alpha=0.5)
    end

    # Prior sampling
    Random.seed!(1)
    η = rand(truncated(Normal(1, 0.25), lower=1), 1000)
    ρ = rand(truncated(Normal(3, 0.25), lower=3), 1000)
    d_seq = range(0, 1, length=50)

    K = [
        [
            η² * exp(-ρ²*d)
            for d ∈ d_seq
        ]
        for (η², ρ²) ∈ zip(η, ρ)
    ]
    K = hcat(K...)'

    K_μ = mean.(eachcol(K))
    K_pi = PI.(eachcol(K))
    K_pi = vcat(K_pi'...)

    plot!(d_seq, [K_μ K_μ], fillrange=K_pi, fillalpha=0.2, c=:black)
    annotate!([
            (0.5, 0.5, text("prior", 12)),
            (0.2, 0.2, text("posterior", 12))
    ])
end
```

```
r = DataFrameRow
 Row │ a        bG         bM        η²       ρ²
     │ Float64  Float64    Float64   Float64  Float64
─────┼──────────────────────────────────────────────
   1 │ 0.22797  0.0462125  0.882097   1.0123  3.03843
r = DataFrameRow
 Row │ a         bG        bM        η²       ρ²
     │ Float64   Float64   Float64   Float64  Float64
─────┼──────────────────────────────────────────────
   2 │ 0.470245  0.161446  0.900867  1.05593   3.0121
r = DataFrameRow
 Row │ a         bG       bM        η²       ρ²
     │ Float64   Float64  Float64   Float64  Float64
─────┼─────────────────────────────────────────────
   3 │ 0.417907  0.29114  0.507619  1.00533  3.19625
r = DataFrameRow
 Row │ a         bG        bM        η²       ρ²
     │ Float64   Float64   Float64   Float64  Float64
─────┼──────────────────────────────────────────────
   4 │ 0.235574  0.662547  0.807119  1.00408  3.14357
r = DataFrameRow
 Row │ a           bG         bM        η²       ρ²
     │ Float64     Float64    Float64   Float64  Float64
─────┼─────────────────────────────────────────────────
   5 │ -0.0101257  0.0209207  0.710142  1.00616  3.10534
r = DataFrameRow
 Row │ a          bG         bM       η²       ρ²
     │ Float64    Float64    Float64  Float64  Float64
─────┼──────────────────────────────────────────────
   6 │ -0.301817  -0.020667  0.58236  1.02259  3.10659
r = DataFrameRow
 Row │ a         bG         bM        η²       ρ²
```

```
      │ Float64    Float64    Float64   Float64  Float64
──────┼──────────────────────────────────────────────────
    7 │ -0.501224  0.0662104  0.832912  1.00502  3.07052
r = DataFrameRow
 Row  │ a          bG          bM        η²       ρ²
      │ Float64    Float64     Float64   Float64  Float64
──────┼──────────────────────────────────────────────────
    8 │ -0.120246  -0.0412677  0.690122  1.01005  3.10297
r = DataFrameRow
 Row  │ a          bG          bM        η²       ρ²
      │ Float64    Float64     Float64   Float64  Float64
──────┼──────────────────────────────────────────────────
    9 │ -0.037833  -0.0630879  0.574009  1.00501  3.05086
r = DataFrameRow
 Row  │ a            bG          bM        η²       ρ²
      │ Float64      Float64     Float64   Float64  Float64
```