Chap 14: Adventures in Covariance

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

```
versioninfo()
```

```
Julia Version 1.11.0-rc4
Commit b4b9add84db (2024-09-25 11:03 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
  LLVM: libLLVM-16.0.6 (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


```
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
```

```
Error requiring `Turing` from `StatisticalRethinking`
exception:
```

LoadError: UndefVarError: `TuringOptimExt` not defined in `StatisticalRethinking`

Suggestion: check for spelling errors or missing imports.

in expression starting at /y/home/huangyu/.julia/packages/StatisticalRethinking/bmpWV/src/require/turing/turing_optim_sample.jl:3

in expression starting at /y/home/huangyu/.julia/packages/StatisticalRethinking/bmpWV/src/require/turing/turing.jl:7

Stack trace

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

1. **include**(mod::Module, _path::String) @ *Base.jl:557*
2. **include**(x::String) @ *StatisticalRethinking.jl:1*
3. *turing.jl:7*
4. **include**(mod::Module, _path::String) @ *Base.jl:557*
5. **include**(x::String) @ *StatisticalRethinking.jl:1*
6. *Requires.jl:40*
7. **eval** @ *boot.jl:430*
8. **eval** @ *StatisticalRethinking.jl:1*
9. (::StatisticalRethinking.var"#3#12")() @ *require.jl:101*
10. **macro expansion** @ *timing.jl:418*
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*

```
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, -0.899
```

```
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
```

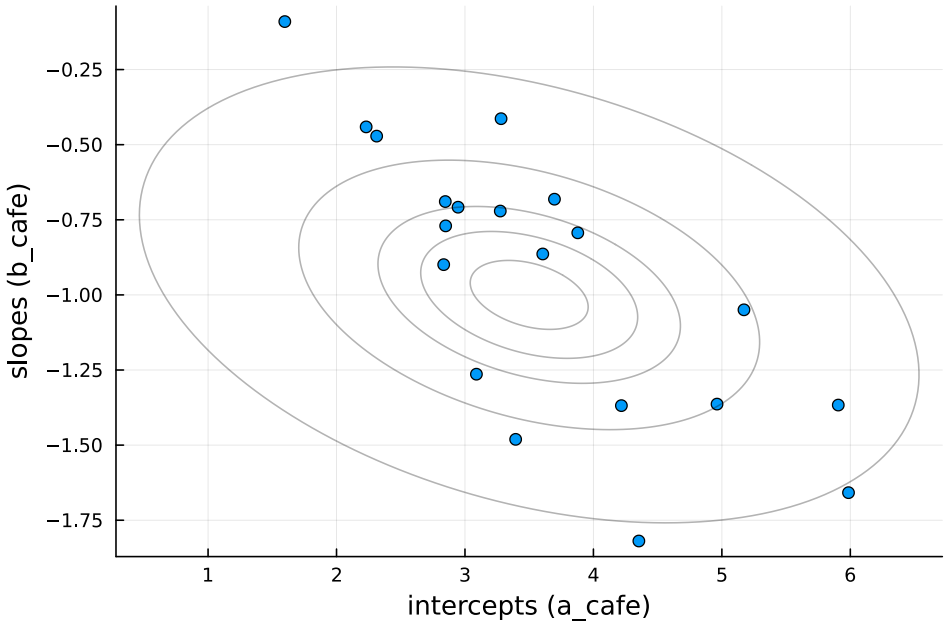
```
[3.87817, 5.98586, 2.94622, 2.8496, 4.35373, 3.28105, 2.31243, 2.83429, 3.0891, 5.
```

```
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

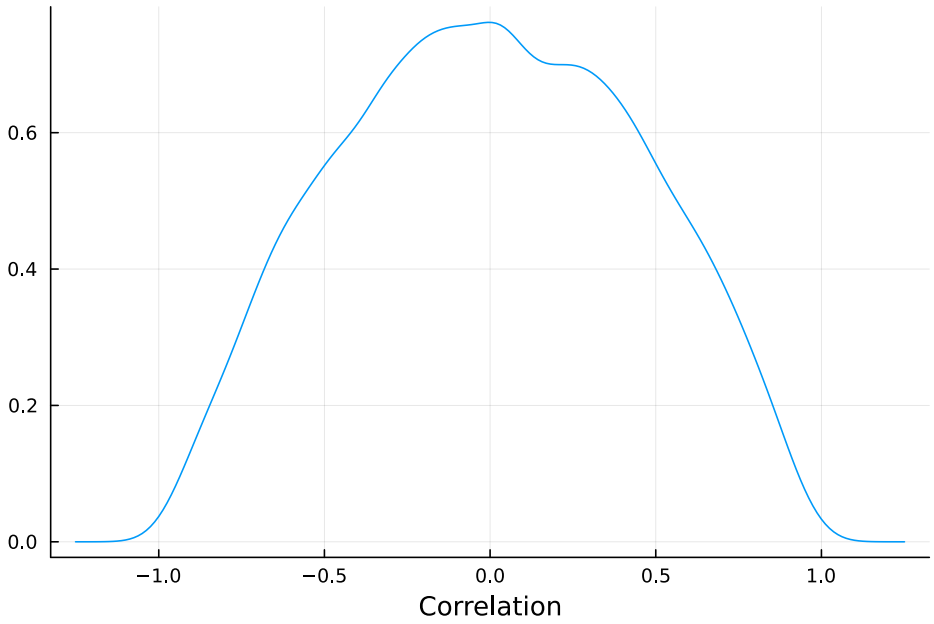
```
md" ## Code 14.10 Simulate observations"
```

	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
```

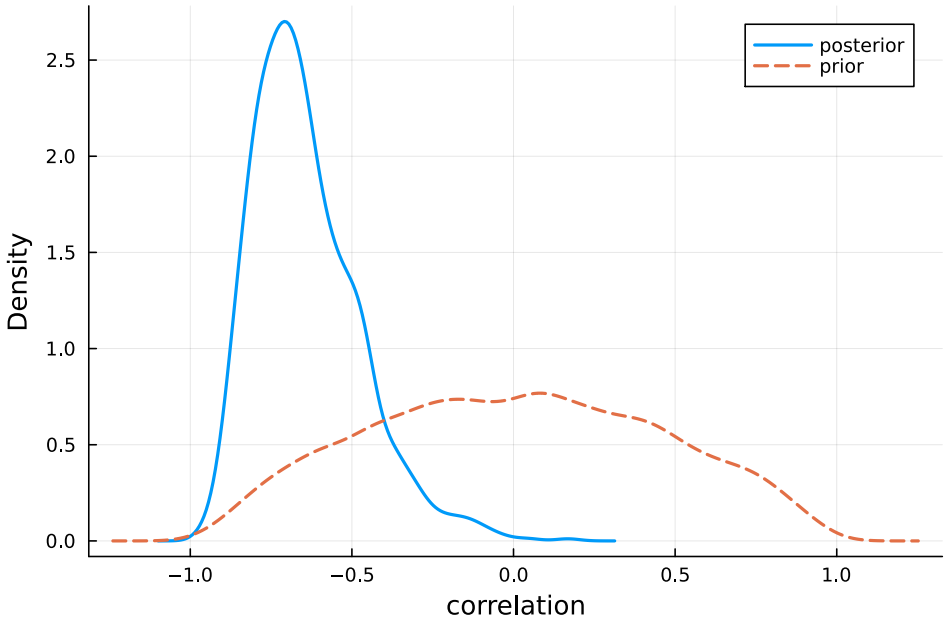
	Rho[1,1]	Rho[1,2]	Rho[2,1]	Rho[2,2]	a	ab[1,10]	ab[1,11]	ab[1,1
1	1.0	-0.657946	-0.657946	1.0	3.51383	5.53969	4.17799	2.114
2	1.0	-0.518042	-0.518042	1.0	3.69568	5.66638	4.07324	2.176
3	1.0	-0.207418	-0.207418	1.0	3.83465	5.40347	4.50334	1.940
4	1.0	-0.804438	-0.804438	1.0	3.86767	5.48815	4.19479	2.016
5	1.0	-0.640704	-0.640704	1.0	3.62613	5.31376	4.73209	1.952
6	1.0	-0.577758	-0.577758	1.0	3.70831	5.63534	4.42166	2.018
7	1.0	-0.70389	-0.70389	1.0	3.16901	5.08622	4.38331	2.010
8	1.0	-0.542787	-0.542787	1.0	3.58689	5.19034	4.07768	2.003
9	1.0	-0.763329	-0.763329	1.0	3.56206	5.35635	4.61547	2.000
10	1.0	-0.583904	-0.583904	1.0	3.49541	6.02328	4.21088	1.858
more								
1000	1.0	-0.811909	-0.811909	1.0	3.93622	5.46707	4.28705	2.381

```
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
```

```
Sampling 100%

Found initial step size
ϵ: 0.025

 31.508270 seconds (41.28 M allocations: 11.459 GiB, 7.39% gc time, 60.21% 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
```

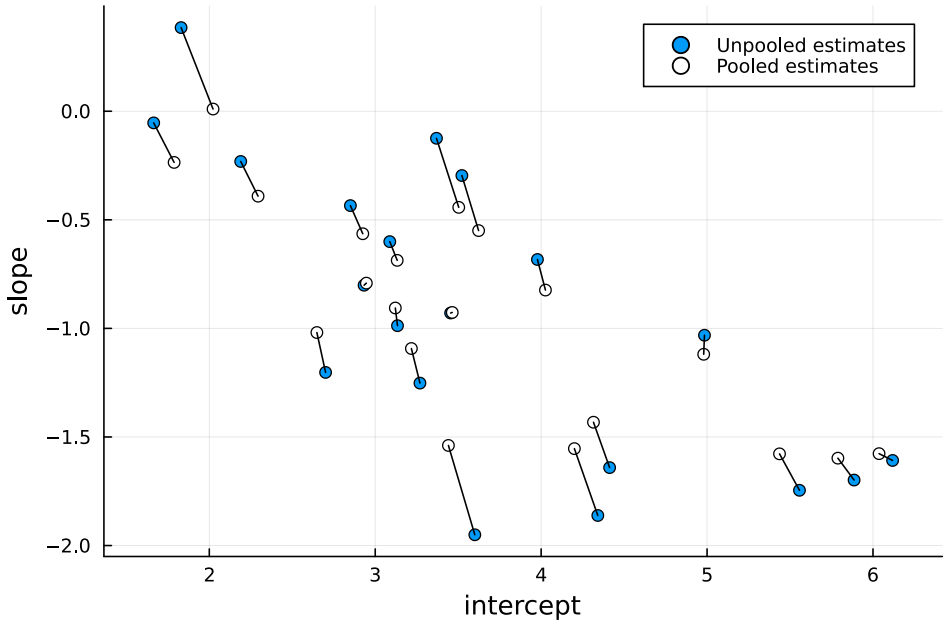
```
LKJ(2, 2) = Distributions.LKJ{Float64, Int64}(
d: 2
η: 2.0
)
```

Code 14.14 Draw posterior estimates of intercept vs slope

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

Note

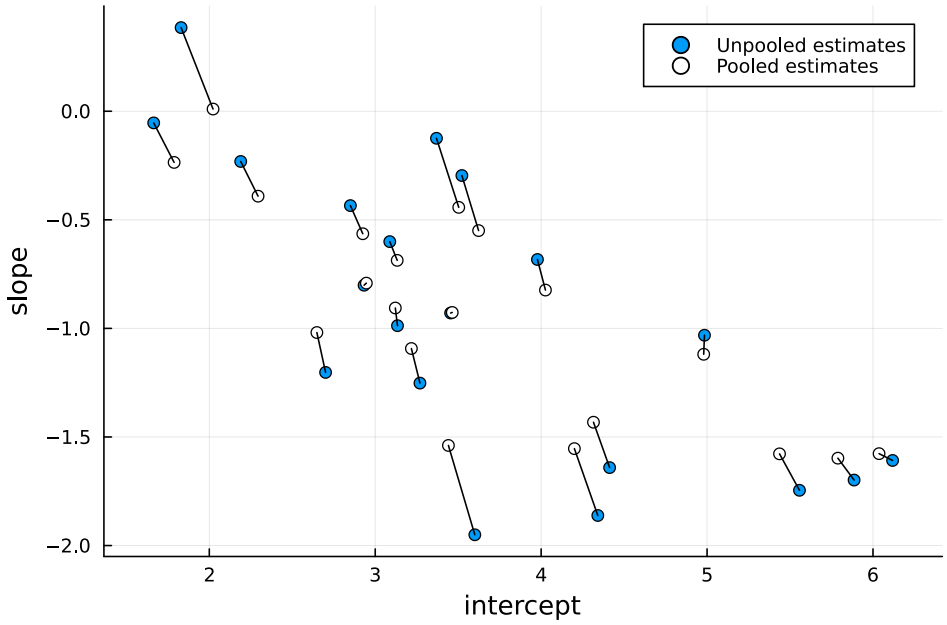
Plot differs from presented in the book due to different seed in data generation.




```
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"
```

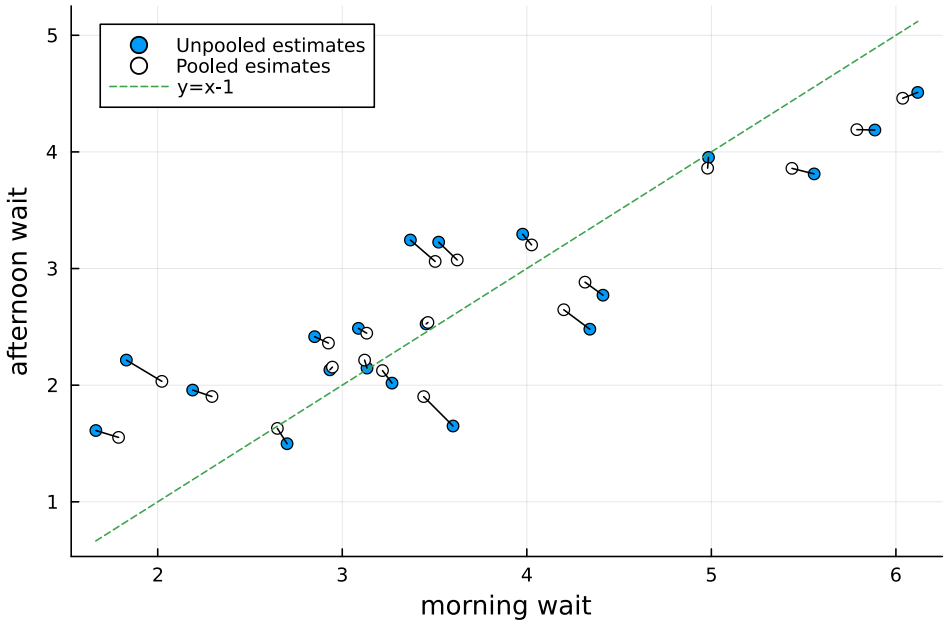



```
let
    # posterior mean
    @show ρ = mean(m14_1_df."Rho[1,2]")
    @show μ_a = mean(m14_1_df.a)
    @show μ_b = mean(m14_1_df.b)
    @show σ₁ = mean(m14_1_df."σ_cafe[1]")
    @show σ₂ = mean(m14_1_df."σ_cafe[2]")

    # 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]")) = -0.6427801748243955
μ_a = mean(m14_1_df.a) = 3.6692745226209116
μ_b = mean(m14_1_df.b) = -0.9522716534408012
σ₁ = mean(m14_1_df.:("σ_cafe[1]")) = 1.2181016699406042
σ₂ = mean(m14_1_df.:("σ_cafe[2]")) = 0.5740874582286424
ρ * σ₁ * σ₂ = -0.44949417821126425
dt = acos(ρ * σ₁ * σ₂) = 2.03699533495775
typeof(xₜ) = Main.var"workspace#4".var"#xₜ#25"{Float64, Float64, Float6
4, Float64}
typeof(xₜ) = Main.var"workspace#4".var"#xₜ#25"{Float64, Float64, Float6
4, Float64}
typeof(xₜ) = Main.var"workspace#4".var"#xₜ#25"{Float64, Float64, Float6
4, Float64}
typeof(xₜ) = Main.var"workspace#4".var"#xₜ#25"{Float64, Float64, Float6
4, Float64}
typeof(xₜ) = Main.var"workspace#4".var"#xₜ#25"{Float64, Float64, Float6
4, Float64}
```

Code 14.16 Draw morning wait vs afternoon wait

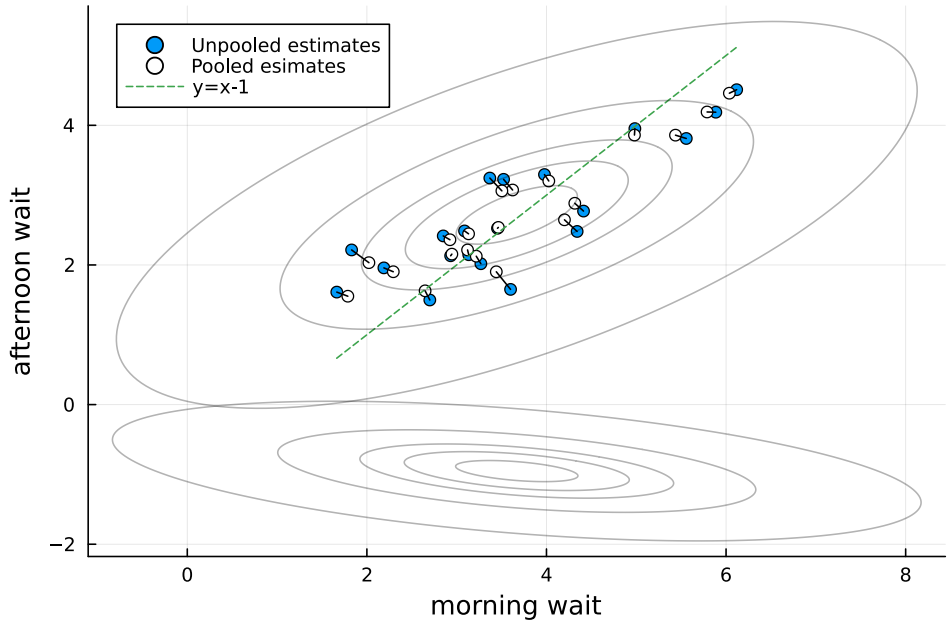
```
md" ## Code 14.16 Draw morning wait vs afternoon wait"
```




```
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') = [1.469347923410294 1.0252152354313935; 1.0252152354313935 0.9127738014717852]
```

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
1	-0.417707	-0.314063	-0.916674	-0.229096	-0.504657	1.17184	-2.135
2	-0.476854	-0.16158	-0.88613	-0.143451	-0.521008	1.16193	-2.015
3	-0.621353	0.0456847	-1.01	0.0130383	-0.517569	1.1296	-1.992
4	-0.474075	0.0639516	-1.00353	0.0499248	-0.452563	1.10109	-2.023
5	-0.571126	0.206455	-0.864771	0.26421	-0.44174	1.10781	-1.849
6	-0.487028	0.167715	-0.9494	0.119357	-0.474304	1.10679	-1.827
7	-0.447684	0.282566	-0.867375	-0.0182152	-0.47895	1.09035	-1.993
8	-0.330291	0.278504	-0.74836	0.0818598	-0.528977	1.08922	-1.985
9	-0.261654	0.17458	-0.864757	0.113052	-0.481614	1.14328	-1.804
10	-0.363133	0.266128	-0.599063	0.220736	-0.473274	1.20038	-1.695
more							
1000	-1.67881	3.73795	-1.24922	-0.12336	-0.592773	1.06977	0.3231

```
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
```

Sampling 100%

```
 47.790067 seconds (80.38 M allocations: 7.661 GiB, 5.39% gc time, 7
8.60% compilation time)
```

	variable	mean	min	median	max	nmissing	elty
1	Symbol("σ_actor[1]")	1.32384	0.525503	1.24591	3.08067	0	Floa
2	Symbol("σ_actor[2]")	0.694874	0.0856066	0.650226	2.17126	0	Floa
3	Symbol("σ_actor[3]")	1.51031	0.174945	1.50527	2.82788	0	Floa
4	Symbol("σ_actor[4]")	1.40991	0.465067	1.11895	7.55954	0	Floa
5	Symbol("σ_block[1]")	0.855117	0.0994363	0.608246	4.66835	0	Floa
6	Symbol("σ_block[2]")	0.673891	0.213082	0.592408	2.41527	0	Floa
7	Symbol("σ_block[3]")	0.222818	0.0152084	0.106654	2.29431	0	Floa
8	Symbol("σ_block[4]")	1.13863	0.0862642	0.71084	5.25396	0	Floa

```
describe(m14_2_df[:, r"σ"])
```

```
[0.506612, 0.399243, 0.436927, 1.01116, 0.747748, 0.366889, 0.303407, 1.08921]
```

```
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     = 33.73 seconds
Compute duration  = 33.73 seconds
parameters        = g[1], g[2], g[3], g[4], σ_actor[1], σ_actor[2], σ_actor[3],
internals         = lp, n_steps, is_accept, acceptance_rate, log_density, hamil

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

          g[1]    0.0478    0.3040    0.0663     20.8877    58.5749    1.0129
          g[2]    0.7048    0.3490    0.0902     15.3814    30.6052    1.0456
          g[3]   -0.2373    0.4206    0.1547      7.7083    49.8762    1.0333
          g[4]    1.5496    1.5286    0.8863      3.2328    15.2337    1.4875
    σ_actor[1]    1.3238    0.5066    0.2875      3.0347    25.4494    1.7344
    σ_actor[2]    0.6949    0.3992    0.1399      7.8104    20.3236    1.1528
    σ_actor[3]    1.5103    0.4369    0.1168     15.5721    12.2639    1.0138
    σ_actor[4]    1.4099    1.0112    0.4588      4.1419    13.8524    1.3447
  ρ_actor[1,1]    1.0000    0.0000       NaN         NaN         NaN       NaN
  ρ_actor[2,1]    0.3645    0.2753    0.0607     19.8644    38.4394    1.0226
  ρ_actor[3,1]    0.4126    0.2431    0.0494     22.4521    19.0494    1.0232
  ρ_actor[4,1]    0.3442    0.2203    0.0428     27.2323    74.6838    1.0511
  ρ_actor[1,2]    0.3645    0.2753    0.0607     19.8644    38.4394    1.0226
  ρ_actor[2,2]    1.0000    0.0000    0.0000    890.2848   785.2775    0.9999
  ρ_actor[3,2]    0.3893    0.3560    0.2083      3.0090    18.7786    1.6818
  ρ_actor[4,2]    0.3474    0.3181    0.1209      7.5244    13.0007    1.1906
  ρ_actor[1,3]    0.4126    0.2431    0.0494     22.4521    19.0494    1.0232
  ρ_actor[2,3]    0.3893    0.3560    0.2083      3.0090    18.7786    1.6818
```

```
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
```

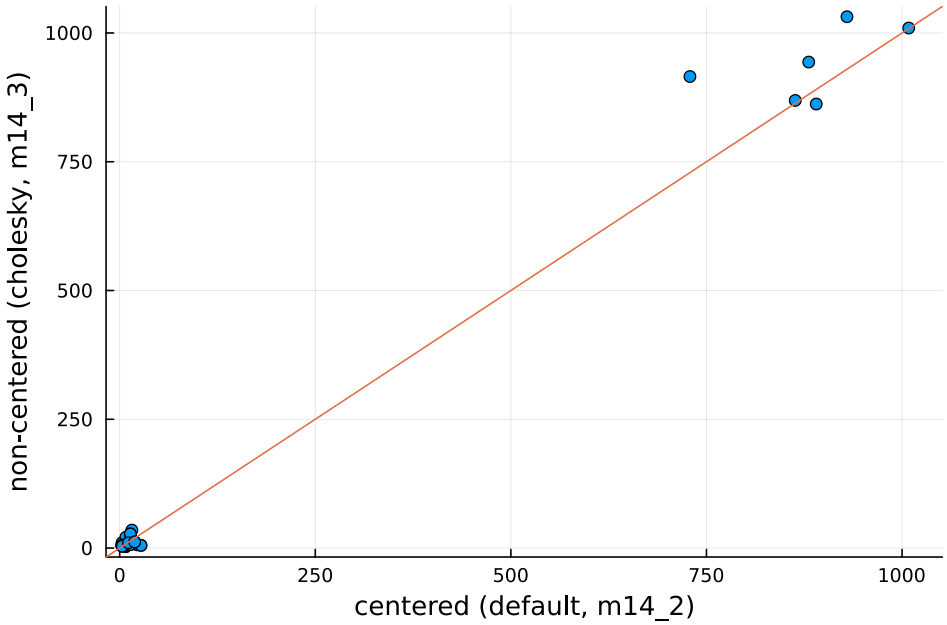
Sampling 100%

```
 27.408660 seconds (36.51 M allocations: 5.708 GiB, 9.09% gc time, 6
0.13% compilation 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	elt
1	Symbol("σ_actor[1]")	1.32317	0.540223	1.20628	3.21638	0	Flo
2	Symbol("σ_actor[2]")	0.835372	0.0517607	0.819267	2.96098	0	Flo
3	Symbol("σ_actor[3]")	1.83635	0.183491	1.79626	3.89039	0	Flo
4	Symbol("σ_actor[4]")	1.42224	0.660021	1.33257	3.51562	0	Flo
5	Symbol("σ_block[1]")	0.607012	0.131275	0.588839	1.94563	0	Flo
6	Symbol("σ_block[2]")	0.671062	0.154311	0.585296	2.30979	0	Flo
7	Symbol("σ_block[3]")	0.152836	0.0109841	0.0864434	1.52265	0	Flo
8	Symbol("σ_block[4]")	0.374972	0.0321135	0.353972	1.56902	0	Flo

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

```
[0.475901, 0.589065, 0.522226, 0.48601, 0.234996, 0.386329, 0.164525, 0.228482]
```

```
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)
```

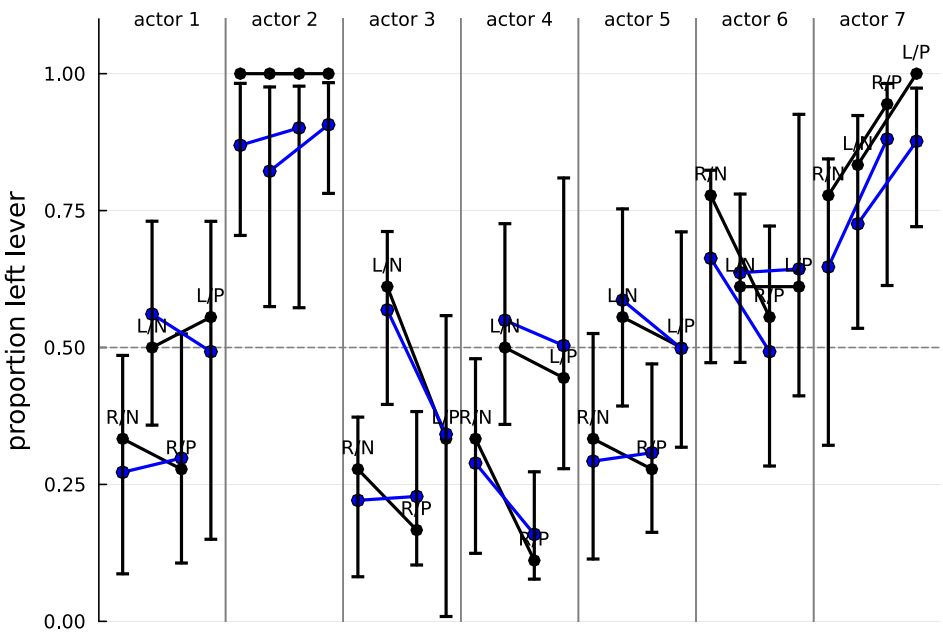
```
l_fun = (r, (ai, ti)) -> begin
    bi = 5
    g = get(r, "g[$ti]", missing)

    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,
 [0.136385, 0.145006, 0.215473, 0.255218, 0.299674, 0.286249, 0.343772, 0.377959,
 [0.0795664, 0.0759377, 0.0872307, 0.105415, 0.127853, 0.116008, 0.142281, 0.1787
 [0.146706, 0.147269, 0.210006, 0.252561, 0.311934, 0.276471, 0.279426, 0.332951,
 [0.11545, 0.10586, 0.135229, 0.169737, 0.182867, 0.174266, 0.196543, 0.213207, 0
 [0.411022, 0.389169, 0.448105, 0.491543, 0.513123, 0.506349, 0.540227, 0.577495,
 [0.0249174, 0.0258881, 0.0345388, 0.0407577, 0.0519391, 0.0517014, 0.0510276, 0.
```

```
1 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	0.939429	0.024442	-1.30672
2	0.0727574	0.55812	0.478416
3	0.0972706	0.539391	0.478416
4	0.9202	-0.13165	0.478416
5	0.389944	0.707761	-0.414151
6	-1.44508	-1.8591	-0.414151
7	-0.275627	-1.28112	-0.414151
8	-0.808393	0.450688	-0.414151
9	-0.155308	1.10511	0.478416
10	-0.656678	0.303932	0.478416
more			
500	0.253302	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	-0.000423836	-0.167881	-0.0014318	0.112602	0	Float64
2	:bEW	0.369349	0.235914	0.368061	0.495973	0	Float64
3	:σ	0.929404	0.84157	0.929294	1.04582	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
```

Sampling 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	-0.000432046	-0.125675	-0.000872481	0.124141	0	Float64
2	:bEW	0.600485	0.452118	0.599345	0.754245	0	Float64
3	:bQW	-0.366451	-0.539258	-0.366389	-0.213789	0	Float64
4	:σ	0.885765	0.803289	0.884955	0.977035	0	Float64

```
begin
    @time 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
```

Sampling 100%

Found initial step size
ϵ: 0.05

0.808449 seconds (2.31 M allocations: 532.007 MiB, 7.77% gc time)

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]
1	0.194709	0.0177445	-0.0353073	0.613553	1.0	0.506708	0.506708
2	-0.218594	-0.0328736	-0.0264742	0.651433	1.0	0.479301	0.479301
3	0.125035	0.0170733	-0.0584866	0.567135	1.0	0.525459	0.525459
4	-0.55263	0.000715925	0.0520584	0.640711	1.0	0.401065	0.401065
5	0.427718	-0.0099319	0.0172782	0.593023	1.0	0.459606	0.459606
6	-0.3327	0.0232276	0.0771693	0.667698	1.0	0.398754	0.398754
7	-0.32543	0.00793258	0.0270426	0.66583	1.0	0.437105	0.437105
8	-0.255165	-0.0220307	0.0723826	0.650154	1.0	0.407369	0.407369
9	0.0787043	0.0180333	0.0952352	0.594229	1.0	0.334419	0.334419
10	0.321241	0.00129076	0.00725325	0.692989	1.0	0.435716	0.435716
more							
1000	0.151718	0.0201554	-0.0660667	0.655863	1.0	0.532381	0.532381

```
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
```

```
Sampling 100%
```

	variable	mean	min	median	max	nmissing	e
1	:aE	0.00136959	-0.694988	-0.00754609	0.627908	0	Fl
2	:aW	7.25389e-5	-0.114637	0.00095929	0.109468	0	Fl
3	:bEW	0.0113427	-0.252964	0.0116683	0.247914	0	Fl
4	:bQE	0.628652	0.530339	0.628861	0.740342	0	Fl
5	Symbol("ρ[1,2]")	0.459903	0.244871	0.460696	0.629955	0	Fl
6	Symbol("ρ[2,1]")	0.459903	0.244871	0.460696	0.629955	0	Fl
7	Symbol("σ[1]")	1.00007	0.876256	0.99704	1.14776	0	Fl
8	Symbol("σ[2]")	0.77611	0.694477	0.775528	0.867317	0	Fl

```
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	-0.831215	0.024442	-1.30672
2	0.12572	0.55812	0.478416
3	-0.778204	0.539391	0.478416
4	2.95242	-0.13165	0.478416
5	-0.0860433	0.707761	-0.414151
6	0.811527	-1.8591	-0.414151
7	0.180959	-1.28112	-0.414151
8	-1.06103	0.450688	-0.414151
9	-0.355389	1.10511	0.478416
10	0.123224	0.303932	0.478416
	more		
500	-0.564479	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
```

	aE	aW	bEW	bQE	ρ[1,1]	ρ[1,2]	ρ[2,1]
1	-0.370683	0.0244453	0.300597	0.632723	1.0	-0.39708	-0.39708
2	0.203413	-0.027976	0.309913	0.665258	1.0	-0.384676	-0.384676
3	-0.0207131	-0.00374719	0.365355	0.60713	1.0	-0.504714	-0.504714
4	0.0319747	0.0072924	0.268568	0.645847	1.0	-0.420281	-0.420281
5	0.376645	0.0719597	0.335492	0.619595	1.0	-0.48527	-0.48527
6	0.447579	0.0423264	0.292015	0.634723	1.0	-0.487633	-0.487633
7	0.281063	0.0382402	0.326411	0.631751	1.0	-0.450584	-0.450584
8	0.0787139	8.71504e-6	0.297312	0.603145	1.0	-0.497854	-0.497854
9	-0.069086	-0.00723928	0.327991	0.591819	1.0	-0.459172	-0.459172
10	0.00939848	-0.00100974	0.362213	0.631014	1.0	-0.464422	-0.464422
	more						
1000	-0.0281268	-0.00356083	0.349405	0.630152	1.0	-0.420665	-0.420665

```
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
```

Sampling 100%

```
6.472083 seconds (46.26 M allocations: 5.512 GiB, 18.39% gc time)
```

	variable	mean	min	median	max	nmissing
1	:aE	0.00906687	-0.600167	0.00853107	0.745003	0
2	:aW	0.00062472	-0.0823971	0.000584539	0.0807928	0
3	:bEW	0.277566	0.0429785	0.275282	0.50516	0
4	:bQE	0.63316	0.526421	0.633598	0.748166	0
5	Symbol("ρ[1,2]")	-0.445249	-0.59736	-0.445285	-0.232784	0
6	Symbol("ρ[2,1]")	-0.445249	-0.59736	-0.445285	-0.232784	0
7	Symbol("σ[1]")	1.06466	0.924235	1.06127	1.2043	0
8	Symbol("σ[2]")	0.77422	0.693757	0.773588	0.861869	0

```
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	dln
1	1	2	1	0	4	0.0	0	0	1	0	-2.
2	1	3	2	6	31	-0.003	0	1	0	0	-2.
3	1	4	3	2	5	-0.019	0	1	0	0	-1.

```
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 = [
```

```
# +
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
```

	a	gr[1,10]	gr[1,11]	gr[1,12]	gr[1,13]	gr[1,14]	gr[1,15]
1	0.668223	1.36135	-1.09191	-0.908941	0.921675	0.189233	-1.26064
2	0.593859	1.54545	-0.812823	-0.586245	0.838229	0.496047	-1.22364
3	0.816155	1.13454	-0.701965	-0.783108	0.816233	0.0520201	-1.05138
4	0.774737	1.27193	-1.35479	-0.760177	0.761668	0.0262124	-1.18852
5	0.616663	1.07588	-0.978685	-0.718429	0.93517	0.207962	-0.972934
6	0.804864	1.13899	-1.216	-0.549003	0.665814	0.337514	-0.499462
7	0.652274	1.31269	-1.07505	-1.24125	0.648005	-0.446581	-0.477389
8	0.718287	1.20613	-0.880436	-0.876667	0.779502	0.0827148	-0.502941
9	0.647207	1.24235	-0.877803	-0.707964	0.787835	-0.0278827	-0.67318
10	0.744655	1.36999	-0.873163	-0.858496	0.927935	0.306286	-0.983463
	more						
1000	0.357471	1.99194	-0.72998	-0.196948	1.25014	0.332222	-1.06044

```
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
```

Sampling 100%

Code 14.32 Check posterior estimates of giving vs receiving

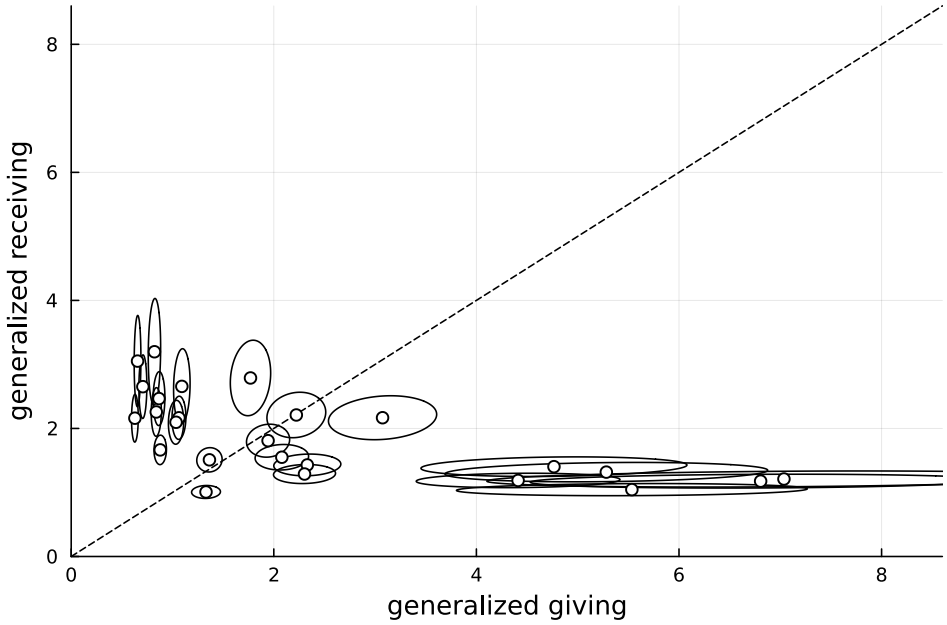
```
md" #### Code 14.32 Check posterior estimates of giving vs receiving"
```

	variable	mean	min	median	max	nmissing	e
1	Symbol("ρ_gr[1,2]")	-0.410239	-0.910125	-0.419703	0.288652	0	Fl
2	Symbol("ρ_gr[2,1]")	-0.410239	-0.910125	-0.419703	0.288652	0	Fl
3	Symbol("σ_gr[1]")	0.832868	0.448778	0.815751	1.42691	0	Fl
4	Symbol("σ_gr[2]")	0.415959	0.181183	0.40714	0.797566	0	Fl

```
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"
```




```
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

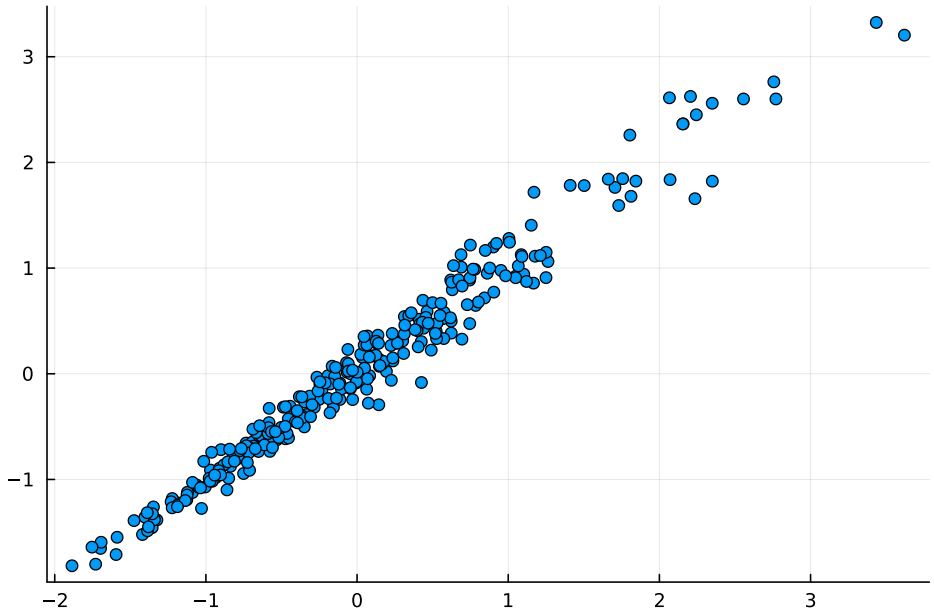
```
md" ## Code 14.35 Estimates of dyads"
```

	variable	mean	min	median	max	nmissing	eltype
1	Symbol("ρ_d[1,1]")	1.0	1.0	1.0	1.0	0	Float6
2	Symbol("ρ_d[1,2]")	0.914839	0.785041	0.918662	0.982436	0	Float6
3	Symbol("ρ_d[2,1]")	0.914839	0.785041	0.918662	0.982436	0	Float6
4	Symbol("ρ_d[2,2]")	1.0	1.0	1.0	1.0	0	Float6
5	:σ_d	1.10678	0.883299	1.10584	1.29037	0	Float6

```
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."
```




```
#
# 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

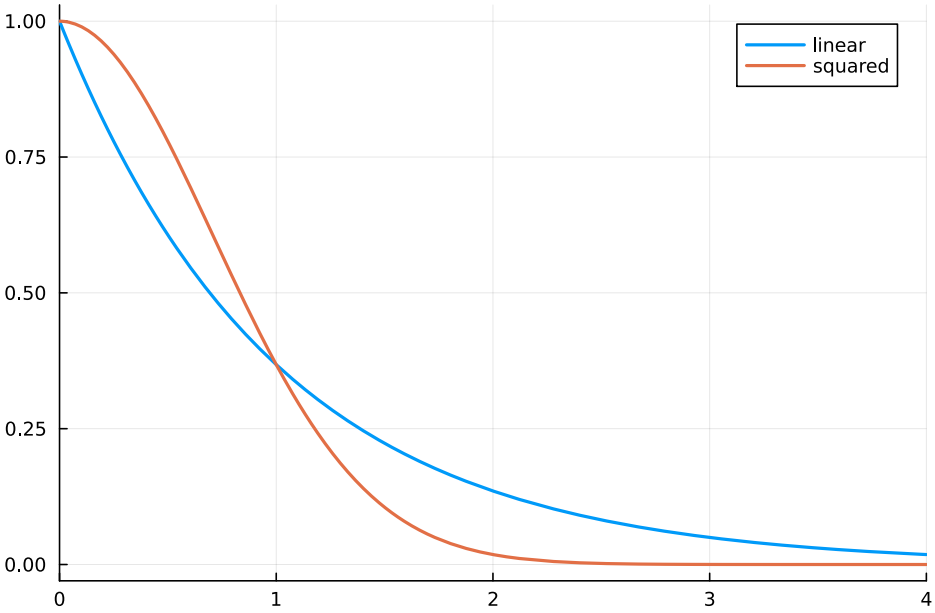
```
md"## Code 14.37 Load the distance matrix of 10 islands in the Kline
dataset"
```

```
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    Manus    Tonga    Hawaii
     │ Float64   Float64  Float64     Float64  Float64   Float64    Float64  Float64  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	lo
1	"Malekula"	1100	"low"	13	3.2	-16.3	167.5	-1
2	"Tikopia"	1500	"low"	22	4.7	-12.3	168.8	-1
3	"Santa Cruz"	3600	"low"	24	4.0	-10.7	166.0	-1
4	"Yap"	4791	"high"	43	5.0	9.5	138.1	-4
5	"Lau Fiji"	7400	"high"	33	5.0	-17.7	178.1	-1

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

	a	b	g	k[10]	k[1]	k[2]	k[3]
1	1.45238	0.324096	0.842851	-0.226995	-0.626395	0.275408	0.184
2	0.571851	0.41119	0.727426	-0.561549	0.245026	0.159697	0.0984115
3	0.81418	0.418264	1.26837	-0.52074	-0.37118	0.411883	0.192045
4	0.895721	0.285301	0.353218	-0.267641	-0.33972	0.193477	-0.036233
5	1.97988	0.328158	1.06905	-0.477676	-0.102479	0.103301	-0.316058
6	0.725774	0.321829	0.752886	0.187549	0.310349	0.79282	0.184478
7	0.497805	0.306187	0.412734	0.242201	0.416451	0.512626	0.407967
8	0.945026	0.282373	0.517388	0.0810263	0.0983937	0.392046	0.0618341
9	1.05809	0.264327	0.453589	0.108235	-0.150324	0.14857	0.363435
10	1.031	0.286022	0.48499	0.101321	-0.269605	0.0470595	0.396791
more							
1000	0.343189	0.280821	0.137591	-0.0250691	-0.164921	0.221518	-0.046793

```
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
```

Sampling 100%

Found initial step size
ϵ: 3.0517578125e-6

43.234221 seconds (62.16 M allocations: 15.552 GiB, 7.31% gc time, 51.69% compilation time)

	parameters	ess	rhat	ess_per_sec
1	:η²	298.816	1.00617	8.11118
2	:ρ²	597.1	0.999008	16.2079
3	:a	555.96	1.0009	15.0912
4	:b	280.425	1.01252	7.61198
5	:g	332.042	0.999967	9.01308
6	Symbol("k[1]")	369.556	1.00478	10.0314
7	Symbol("k[2]")	311.218	1.00158	8.44782
8	Symbol("k[3]")	344.999	1.00135	9.36479
9	Symbol("k[4]")	342.83	1.00039	9.30592
10	Symbol("k[5]")	373.875	1.00208	10.1486
11	Symbol("k[6]")	366.316	1.00284	9.94344
12	Symbol("k[7]")	355.974	1.00071	9.66271
13	Symbol("k[8]")	322.916	1.00416	8.76537
14	Symbol("k[9]")	376.923	1.00047	10.2313
15	Symbol("k[10]")	273.135	1.00805	7.4141

```
ess_rhat(m14_8_ch)
```

Code 14.40 Posterior estimates

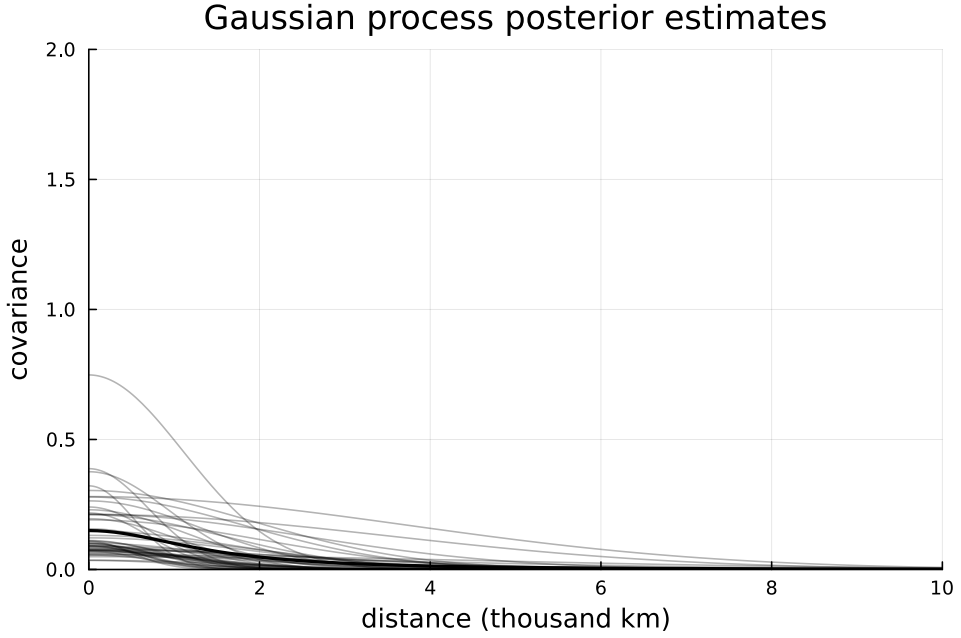
```
md"## Code 14.40 Posterior estimates"
```

	variable	mean	min	median	max	nmissing	
1	:a	1.37546	0.0471188	1.13546	9.13792	0	
2	:b	0.278803	0.0651048	0.278105	0.553101	0	
3	:g	0.592731	0.00809031	0.450591	3.85064	0	
4	Symbol("k[10]")	-0.130693	-1.52471	-0.118441	0.790872	0	
5	Symbol("k[1]")	-0.195035	-1.14865	-0.196343	0.559436	0	
6	Symbol("k[2]")	0.0835375	-0.646748	0.0782683	0.828934	0	
7	Symbol("k[3]")	-0.0186044	-1.21942	-0.0224099	0.709516	0	
8	Symbol("k[4]")	0.373466	-0.210505	0.367295	1.11371	0	
9	Symbol("k[5]")	0.065331	-0.820611	0.0632858	0.775987	0	
10	Symbol("k[6]")	-0.333055	-1.6791	-0.316699	0.384583	0	
11	Symbol("k[7]")	0.169819	-0.814528	0.161563	0.8403	0	
12	Symbol("k[8]")	-0.175578	-1.42058	-0.168713	0.563473	0	
13	Symbol("k[9]")	0.294285	-0.553318	0.289529	1.08295	0	
14	:η²	0.149638	0.00130013	0.103427	1.99472	0	
15	:ρ²	0.48709	0.00131721	0.349215	3.30259	0	

```
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}:
 0.216854     0.0955904    0.0900009    …  0.0067716   0.0308988    1.3338e-6
 0.0955904    0.216854     0.0999043       0.00857922  0.0268553    6.04959e-6
 0.0900009    0.0999043    0.216854        0.0148866   0.0168625    3.89514e-6
 0.000134161  0.000236373  0.000570291     0.0415502   2.01592e-7   1.58648e-9
 0.0607699    0.0606651    0.0446956       0.00050205  0.0843995    2.49439e-5
 0.0243191    0.0253355    0.0373417    …  0.0803631   0.000520144  2.00307e-8
 0.00304013   0.0057452    0.00997328      0.0618702   3.4382e-5    8.62104e-7
 0.0067716    0.00857922   0.0148866       0.216854    5.95237e-5   1.45121e-8
 0.0308988    0.0268553    0.0168625       5.95237e-5  0.216854     1.46809e-5
 1.3338e-6    6.04959e-6   3.89514e-6      1.45121e-8  1.46809e-5   0.216854
```

```
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) = 0.10342675067592842
ρ²_kline = median(m14_8_df.ρ²) = 0.34921525380676
K_kline = map((d->begin
                #= /y/home/huangyu/src/SR2TuringPluto.jl/notebooks/Chap
ter_14.jl#==#f2f97686-af08-48a9-a125-e2da8ac13e18:4 =#
                η²_kline * exp(-ρ²_kline * d ^ 2)
            end), Matrix(islandsDistMatrix)) = [0.10342675067592842 0.0
9559035707688367 0.0900009066836664 0.00013416065710907343 0.0607698644
85506385 0.024319124188640736 0.0030401289363306072 0.00677159806580275
75 0.030898837045996927 1.3337966338528576e-6; 0.09559035707688367 0.10
342675067592842 0.09990430278565482 0.0002363725785181301 0.06066511934
819337 0.025335523082057734 0.00574519546768209 0.008579220137901888
0.026855347706719043 6.0495875991601655e-6; 0.0900009066836664 0.099904
30278565482 0.10342675067592842 0.0005702910902487524 0.044695552138751
96 0.03734169575688237 0.009973277521544924 0.014886607126300187 0.0168
62549148063892 3.895142419038476e-6; 0.00013416065710907343 0.000236372
5785181301 0.0005702910902487524 0.10342675067592842 4.0447411400721885
e-6 0.01245491014501979 0.04445388790781411 0.04155024298788562 2.01592
16686122402e-7 1.5864801371717848e-9; 0.060769864485506385 0.0606651193
4819337 0.04469555213875196 4.0447411400721885e-6 0.10342675067592842
0.0027740512550336184 0.0003590601002145368 0.0005020496715800892 0.084
39948875007561 2.4943873151895074e-5; 0.024319124188640736 0.0253355230
82057734 0.03734169575688237 0.01245491014501979 0.0027740512550336184
0.10342675067592842 0.03331969597144541 0.08036314040088664 0.000520143
6113427674 2.003069414098879e-8; 0.0030401289363306072 0.00574519546768
0209 0.009973277521544924 0.04445388790781411 0.0003590601002145368 0.0
3331969597144541 0.10342675067592842 0.06187023083072714 3.438204038597
371e-5 8.621038180757191e-7; 0.0067715980658027575 0.008579220137901888
0.014886607126300187 0.04155024298788562 0.0005020496715800892 0.080363
14040088664 0.06187023083072714 0.10342675067592842 5.952366473340678e-
5 1.4512114404560638e-8; 0.030898837045996927 0.026855347706719043 0.01
6862549148063892 2.0159216686122402e-7 0.08439948875007561 0.0005201436
113427674 3.438204038597371e-5 5.952366473340678e-5 0.10342675067592842
1.4680937057964831e-5; 1.3337966338528576e-6 6.0495875991601655e-6 3.89
5142419038476e-6 1.5864801371717848e-9 2.4943873151895074e-5 2.00306941
4098879e-8 8.621038180757191e-7 1.4512114404560638e-8 1.468093705796483
1e-5 0.10342675067592842]
```

Code 14.43 Convert K to correlation matrix

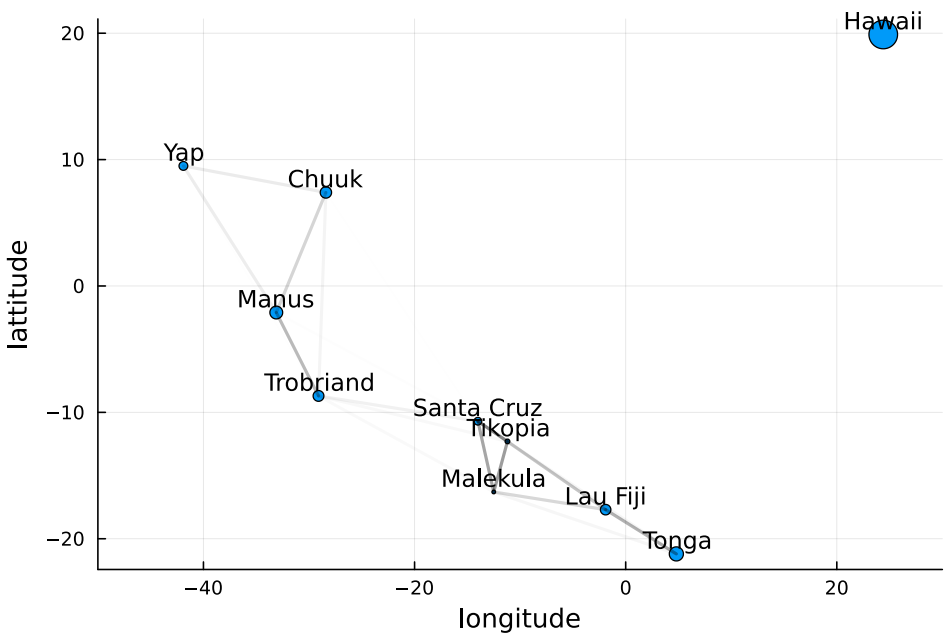
```
md"## Code 14.43 Convert K to correlation matrix"
```

```
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 0.44 0.42 0.0 0.28 0.11 0.01 0.03 0.14 0.0; 0.44 1.0 0.46 0.0 0.28
0.12 0.03 0.04 0.12 0.0; 0.42 0.46 1.0 0.0 0.21 0.17 0.05 0.07 0.08 0.
0; 0.0 0.0 0.0 1.0 0.0 0.06 0.2 0.19 0.0 0.0; 0.28 0.28 0.21 0.0 1.0 0.
01 0.0 0.0 0.39 0.0; 0.11 0.12 0.17 0.06 0.01 1.0 0.15 0.37 0.0 0.0; 0.
01 0.03 0.05 0.2 0.0 0.15 1.0 0.29 0.0 0.0; 0.03 0.04 0.07 0.19 0.0 0.3
7 0.29 1.0 0.0 0.0; 0.14 0.12 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       M
n       To       Ha
     │ Float64  Float64  Float64  Float64  Float64  Float64  Float64  F
loat64  Float64  Float64
─────┼──────────────────────────────────────────────────────────────────
──────────────────────
   1 │     1.0     0.44     0.42     0.0      0.28     0.11     0.01
0.03    0.14     0.0
   2 │     0.44    1.0      0.46     0.0      0.28     0.12     0.03
0.04    0.12     0.0
   3 │     0.42    0.46     1.0      0.0      0.21     0.17     0.05
0.07    0.08     0.0
   4 │     0.0     0.0      0.0      1.0      0.0      0.06     0.2
0.19    0.0      0.0
   5 │     0.28    0.28     0.21     0.0      1.0      0.01     0.0
0.0     0.39     0.0
   6 │     0.11    0.12     0.17     0.06     0.01     1.0      0.15
0.37    0.0      0.0
   7 │     0.01    0.03     0.05     0.2      0.0      0.15     1.0
0.29    0.0      0.0
   8 │     0.03    0.04     0.07     0.19     0.0      0.37     0.29
1.0     0.0      0.0
   9 │     0.14    0.12     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

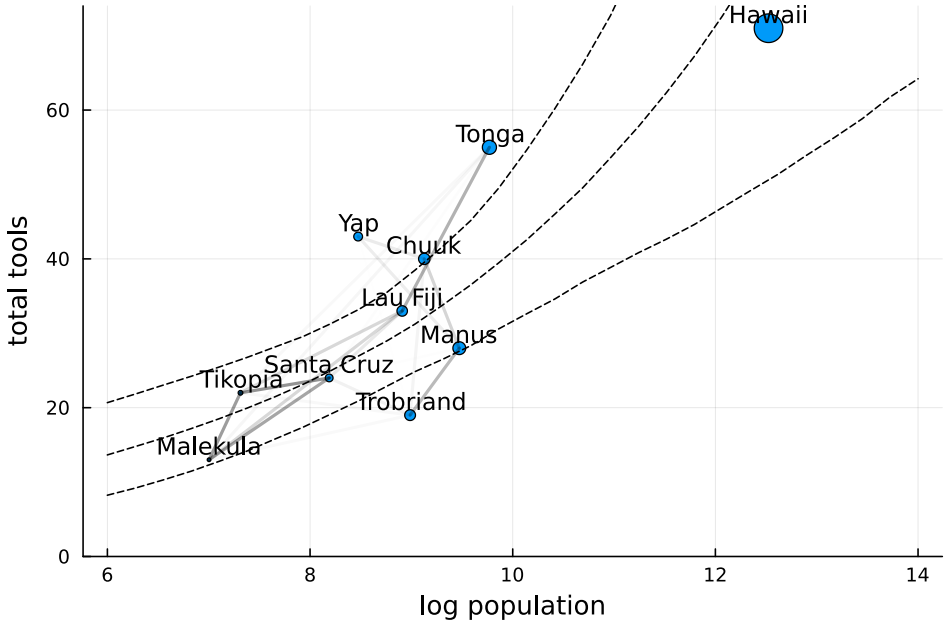
```
md"## Code 14.44 Plot the islands on the map, dot size by log(pop), edge
alpha by correlation between islands"
```




```
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 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	0.481969	0.0463536	0.84227	0.242065	0.232693	-1.0606
2	502	1	0.323372	0.880733	1.05629	0.16327	0.124255	-0.7064
3	503	1	1.53131	0.185273	0.83273	0.273921	0.314443	-0.0405
4	504	1	0.2123	1.12949	0.275797	0.233066	0.0699889	-0.5189
5	505	1	1.71934	0.185299	0.27643	0.214314	0.0555047	-0.6423
6	506	1	0.116134	0.276676	0.147608	0.228659	0.0349072	-0.5856
7	507	1	3.66782	0.244592	2.67432	0.226488	0.632001	-0.3906
8	508	1	0.78838	0.555603	1.82213	0.205261	0.343938	-0.3885
9	509	1	4.54856	0.367759	0.862188	0.258985	0.258494	-0.2811
10	510	1	1.13521	0.497189	3.63896	0.239607	0.962615	-0.1192

more

```
begin
    @model function m14_8nc(T, P, society, Dmat)
        # truncated to prevent cholesky decomposition (requires positive
    definite matrix) failure
        # and improve ssampling efficiency
        η² ~ truncated(Exponential(2), lower=0.01, upper=30)
        ρ² ~ truncated(Exponential(0.5), lower=0.01, upper=30)
        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
```

Sampling 100%

Found initial step size
ϵ: 3.0517578125e-6

29.175985 seconds (44.50 M allocations: 17.081 GiB, 9.72% gc time, 45.43% compilation time)

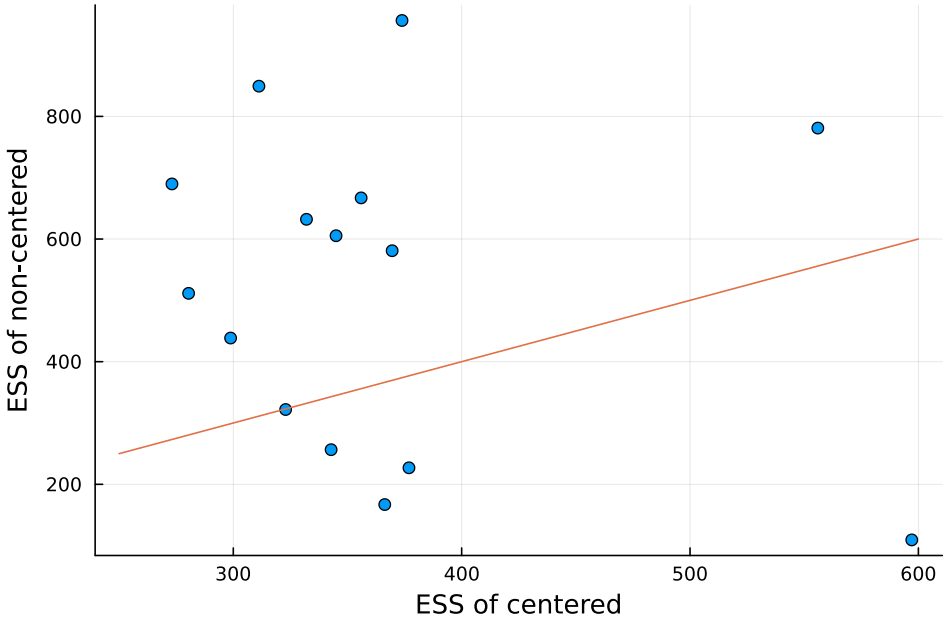
	variable	mean	min	median	max	nmissing
1	:a	1.59602	0.0595689	1.32287	7.41462	0
2	:b	0.227127	0.0621353	0.226613	0.375309	0
3	:g	0.391572	0.0075018	0.300138	2.40027	0
4	Symbol("z[10]")	-0.140941	-3.58073	-0.119874	2.98258	0
5	Symbol("z[1]")	-0.449823	-2.45787	-0.388147	0.701448	0
6	Symbol("z[2]")	-0.0419409	-2.99693	-0.0393957	4.76235	0
7	Symbol("z[3]")	-0.150084	-2.81428	-0.178995	3.22079	0
8	Symbol("z[4]")	0.426019	-3.03173	0.507833	3.22049	0
9	Symbol("z[5]")	0.0102977	-2.58729	-0.000788183	3.56601	0
10	Symbol("z[6]")	-0.50356	-3.58936	-0.541079	3.05754	0
11	Symbol("z[7]")	0.181929	-2.84422	0.213996	3.34966	0
12	Symbol("z[8]")	-0.29832	-3.05767	-0.362369	3.73932	0
13	Symbol("z[9]")	0.375471	-2.84759	0.454157	3.38227	0
14	:η²	1.74776	0.0145153	1.20798	13.164	0
15	:ρ²	0.314976	0.0103645	0.111551	4.03866	0

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

m14_8nc_ch_ess_rhat =

	parameters	ess	rhat	ess_per_sec
1	:η²	438.551	1.00544	16.6061
2	:ρ²	109.35	1.01291	4.14062
3	:a	780.936	1.00115	29.5708
4	:b	511.419	1.00032	19.3653
5	:g	632.148	1.00254	23.9368
6	Symbol("z[1]")	580.944	1.00571	21.9979
7	Symbol("z[2]")	849.275	1.00428	32.1585
8	Symbol("z[3]")	605.392	1.00056	22.9237
9	Symbol("z[4]")	256.476	1.00537	9.7117
10	Symbol("z[5]")	956.4	1.00759	36.2149
11	Symbol("z[6]")	167.003	1.0097	6.32371
12	Symbol("z[7]")	667.093	1.00083	25.26
13	Symbol("z[8]")	321.984	1.005	12.1922
14	Symbol("z[9]")	226.917	1.00649	8.59242
15	Symbol("z[10]")	689.844	0.999445	26.1215

```
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
1	"Allenopithecus_nigroviridis"	"Allenopithecus"	"nigroviridis"	missing
2	"Allocebus_trichotis"	"Allocebus"	"trichotis"	missing
3	"Alouatta_belzebul"	"Alouatta"	"belzebul"	missing
4	"Alouatta_caraya"	"Alouatta"	"caraya"	missing
5	"Alouatta_guariba"	"Alouatta"	"guariba"	missing
6	"Alouatta_palliata"	"Alouatta"	"palliata"	missing
7	"Alouatta_pigra"	"Alouatta"	"pigra"	missing
8	"Alouatta_sara"	"Alouatta"	"sara"	missing
9	"Alouatta_seniculus"	"Alouatta"	"seniculus"	missing
10	"Aotus_azarai"	"Aotus"	"azarai"	missing

more

301	"Varecia_variegata_variegata"	"Varecia"	"variegata"	"variegata

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

	variable	mean	min	median	
1	:name	nothing	"Allenopithecus_nigroviridis"	nothing	"Vare
2	:genus	nothing	"Allenopithecus"	nothing	"Vare
3	:species	nothing	"abelii"	nothing	"zaza
4	:subspecies	nothing	"alaotrensis"	nothing	"veru
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	491.2
10	:body	6795.18	31.23	3553.5	13000
	more				
16	:maternal_investment	478.64	99.99	401.35	1492.

```
describe(d)
```

14.48 Trim the missing data

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

```
["Allenopithecus_nigroviridis", "Alouatta_belzebul", "Alouatta_caraya", "Alouatta
```

```
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 & PyMC3 (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 & PyMC3 (0.05).
- Other estimates are similar."
```

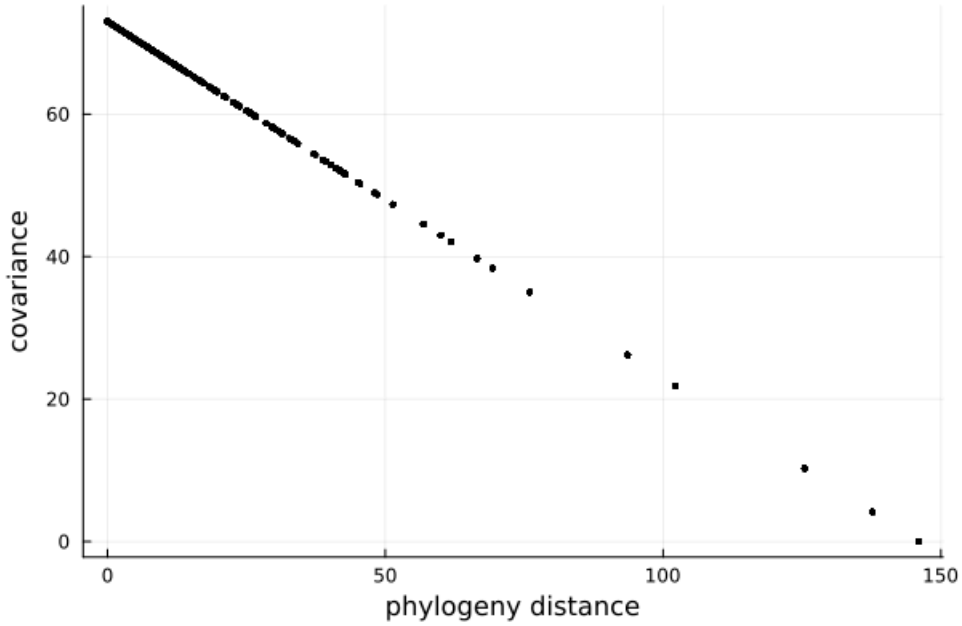
	variable	mean	min	median	max	nmissing	eltype
1	:a	-5.5118e-5	-0.0525152	-4.54075e-5	0.0606643	0	Float64
2	:bG	0.123264	0.0538936	0.122837	0.192069	0	Float64
3	:bM	0.89316	0.816441	0.894246	0.96223	0	Float64
4	:σ_sq	0.215283	0.181315	0.215428	0.261569	0	Float64

```
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
```

```
Sampling 100%

Found initial step size
ϵ: 0.00625

 13.224867 seconds (14.62 M allocations: 901.864 MiB, 3.18% gc time,
94.33% 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	C
1	0.0	23.7898	23.7898	2
2	23.7898	0.0	12.8584	1
3	23.7898	12.8584	0.0	1

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

	Allenopithecus_nigroviridis	Cercopithecus_albogularis	Cercopithecus_ascanius	C
1	73.003	61.1081	61.1081	6
2	61.1081	73.003	66.5738	6
3	61.1081	66.5738	73.003	6
4	61.1081	65.1128	65.1128	7
5	61.1081	65.1128	65.1128	7

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

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

- All estimates are similar to the book or PyMC3.
- 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 or PyMC3.
- Effect of group size is now insignificant!"
```

	variable	mean	min	median	max	nmissing	eltype
1	:a	-0.197988	-0.830633	-0.196486	0.625432	0	Float64
2	:bG	-0.0126961	-0.0736146	-0.0124873	0.0525547	0	Float64
3	:bM	0.69926	0.571707	0.701472	0.796535	0	Float64
4	:σ_sq	0.161624	0.112051	0.16078	0.256818	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
```

```
Sampling 100%

Found initial step size
ϵ: 0.025

 40.932121 seconds (7.01 M allocations: 16.360 GiB, 5.30% gc time, 1
1.96% compilation time)
```

```
[1, 107, 108, 109, 110, 111, 113, 120, 128, 165, 114, 116, 117, 34, 35, 174, 199, 20(
```

```
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
```

```
spp_obs
```

14.52 OU process kernel (=Exponential distance kernel)

- After truncating the range in sampling η^2 and ρ^2 to [0.01, 50], η^2 estimate (0.0135, was 1.01) is similar to the book (0.03). PyMC3 estimate of η^2 (1.08) is higher, not sure why.

```
md"## 14.52 OU process kernel (=Exponential distance kernel)
- After truncating the range in sampling η² and ρ² to [0.01, 50] (was above 1 and 3 respectively), η² estimate (0.0135, was 1.01) is similar to the book (0.03). PyMC3 estimate of η² (1.08) is higher, not sure why."
```

```
151×151 Matrix{Float64}:
 0.0       0.641231   0.641231   0.641231   …  0.293281   0.293281   0.293281   1
 0.641231  0.0        0.053261   0.0625898     0.641231   0.641231   0.641231   1
 0.641231  0.053261   0.0        0.0625898     0.641231   0.641231   0.641231   1
 0.641231  0.0625898  0.0625898  0.0           0.641231   0.641231   0.641231   1
 0.641231  0.0468669  0.053261   0.0625898     0.641231   0.641231   0.641231   1
 0.641231  0.0468669  0.053261   0.0625898  …  0.641231   0.641231   0.641231   1
 0.641231  0.053261   0.0395391  0.0625898     0.641231   0.641231   0.641231   1
 ⋮                                          ⋱                                  ⋮
 0.293281  0.641231   0.641231   0.641231   …  0.156999   0.0640177  0.0640177  1
 0.293281  0.641231   0.641231   0.641231      0.0719557  0.156999   0.156999   1
 0.293281  0.641231   0.641231   0.641231      0.0        0.156999   0.156999   1
 0.293281  0.641231   0.641231   0.641231      0.156999   0.0        0.0478578  1
 0.293281  0.641231   0.641231   0.641231      0.156999   0.0478578  0.0        1
 1.0       1.0        1.0        1.0        …  1.0        1.0        1.0        0
```

```
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	-0.0377256	-0.251695	-0.0372137	0.127873	0	Float64
2	:bG	0.075701	-0.0157486	0.0757361	0.159126	0	Float64
3	:bM	0.866457	0.770994	0.866204	0.949883	0	Float64
4	:η²	0.0135211	0.0100215	0.0132357	0.0234383	0	Float64
5	:ρ²	2.96498	2.13669	2.96609	3.72954	0	Float64

```
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=0.01, upper=50)
        ρ² ~ truncated(Normal(3, 0.25), lower=0.01, upper=50)

        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
```

```
Sampling 100%
```

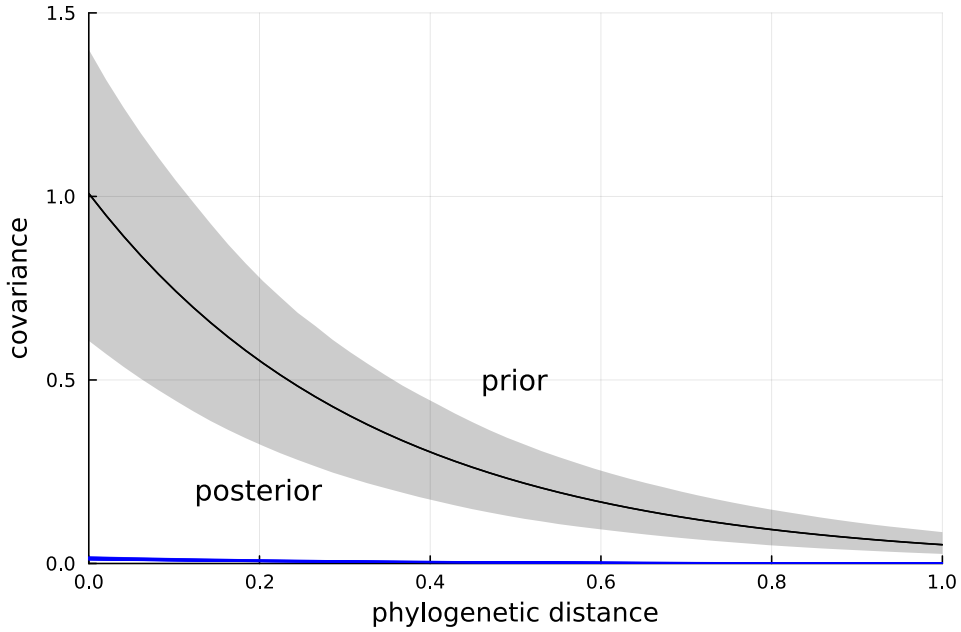
```
186.863054 seconds (16.81 M allocations: 179.516 GiB, 9.66% gc time,
5.01% compilation time)
```

	a	bG	bM	η²	ρ²
1	-0.0022878	0.0564412	0.878055	0.0108575	3.02875
2	0.0305023	0.094933	0.868039	0.0108652	2.97415
3	0.0615677	0.0900929	0.857875	0.0109256	3.08817
4	-0.119425	0.109866	0.84849	0.0116697	3.08448
5	0.00112701	0.103294	0.82943	0.0106868	2.2912
6	-0.103547	0.0819404	0.842405	0.0113144	3.07366
7	-0.0836759	0.0827252	0.852874	0.0125018	3.04564
8	-0.0794592	0.0907527	0.872432	0.0132997	3.04144
9	-0.0600428	0.0746468	0.845908	0.0117128	3.11801
10	-0.0450955	0.0588526	0.889034	0.0157301	3.34818
	more				
30	-0.0122219	0.00248618	0.890597	0.0106985	3.12447

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

14.53 Posterior estimates vs prior

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




```julia
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_vec = rand(Normal(1, 0.25), 1000)
    ρ_vec = rand(Normal(3, 0.25), 1000)
    d_seq = range(0, 1, length=50)

    K = [
        [
            η² * exp(-ρ²*d)
            for d ∈ d_seq
        ]
        for (η², ρ²) ∈ zip(η_vec, ρ_vec)
    ]
    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
```