Performance of Gibbs v.s. NUTS on MoGs

1. Load packages & helper functions

- `Distributions.jl` is for distribution supports and `Turing.jl` is our PPL.
- `gmm.helper.jl` contains functions to build histogram plots

[1]
```
using Distributions, Turing

TPATH = Pkg.dir("Turing")

include(TPATH*"/example-models/nips-2017/gmm.helper.jl");
```

Function below takes x from Gibbs and NUTS chains and make a histogram plot with exact density.

[2]
```
make_plot(x_gibbs, x_nuts, xmin=-5, xmax=20) = begin
    x, y_g = make_vec(x_gibbs)
    gibbs_layer = layer(x=x, y=y_g, Geom.bar,
Theme(default_color=colors[1]))
    x, y_n = make_vec(x_nuts)
    nuts_layer = layer(x=x, y=y_n, Geom.bar,
Theme(default_color=colors[2]))
    contour_layer = layer([make_norm_pdf(p, μ, σ)], xmin, xmax,
Theme(default_color=colors[3]))

    layers = [gibbs_layer, nuts_layer, contour_layer]
    labels = ["Gibbs", "NUTS", "Exact"]

    order = [3,1]
    plot_g = plot(layers[order]..., Guide.manual_color_key("",
labels[order], colors[order]),
                  Coord.cartesian(xmin=xmin, xmax=xmax, ymin=0,
ymax=1.0),
```

```
                        Guide.xlabel(nothing), Guide.ylabel("Density"),
Guide.title("NUTS v.s. Gibbs"))

    order = [3,2]
    plot_n = plot(layers[order]..., Guide.manual_color_key("",
labels[order], colors[order]),
                        Coord.cartesian(xmin=xmin, xmax=xmax, ymin=0,
ymax=1.0),
                        Guide.xlabel(nothing), Guide.ylabel("Density"),
Guide.title("NUTS v.s. Gibbs"))

    vstack(plot_g, plot_n)
end
```

2. Define MoG(s)

[3]

```
@model GMM(p, μ, σ) = begin
  z ~ Categorical(p)
  x ~ Normal(μ[z], σ[z])
end

# @model cGMM(p, μ, σ) = begin
#   x ~ UnivariateGMM(μ, σ, Categorical(p));
# end

M = 5
p = [ 0.2,   0.2,    0.2,  0.2,   0.2]
μ = [   0,     1,      2,  3.5,  4.25] + 2.5 * collect(0:4)

s = [-0.5,  -1.5,  -0.75,   -2,  -0.5]
σ = exp(s);

# TODO: Remove blow when PR is accepted

make_norm_pdf(p, μ, σ) =
  x -> (pdf(Normal(μ[1], σ[1]), x) * p[1] + pdf(Normal(μ[2], σ[2]), x)
* p[2] +
        pdf(Normal(μ[3], σ[3]), x) * p[3] + pdf(Normal(μ[4], σ[4]), x)
* p[4] +
        pdf(Normal(μ[5], σ[5]), x) * p[5])

vn = Turing.VarName(gensym(), :x, "", 0)
@model cGMM(p, μ, σ) = begin
  if isempty(vi)
    Turing.push!(vi, vn, 0, Normal(0,1), 0)
    x = rand(Uniform(-20,20))
  else
```

```
        x = vi[vn]
    end
    Turing.acclogp!(vi, log(make_norm_pdf(p, μ, σ)(x)))
end
```

3. Sample from MoG(s) using Gibbs and NUTS

3.0 Some parameter for the experiment

[4]

```
N = 10000
K = 500;
```

3.1 Sampling from MoG(s) with setting 1

[5]

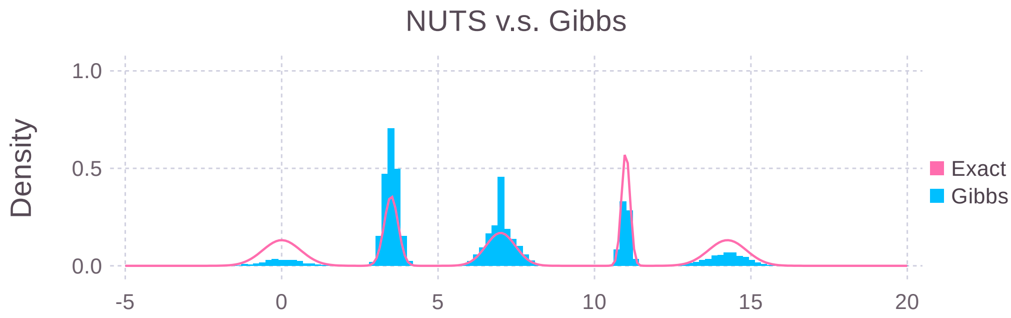
```
println("Running Gibbs")
chain_gibbs = sample(GMM(p, μ, σ), Gibbs(round(Int,N/K), PG(10, 1, :z),
HMC(K-1, 0.2, 4, :x); thin=false))
x_gibbs = map(x_arr -> x_arr[1], chain_gibbs[:x]);

println("Running NUTS")
chain_nuts = sample(cGMM(p, μ, σ), NUTS(N, 0.65))
x_nuts = map(x_arr -> x_arr[1], chain_nuts[:x]);
```

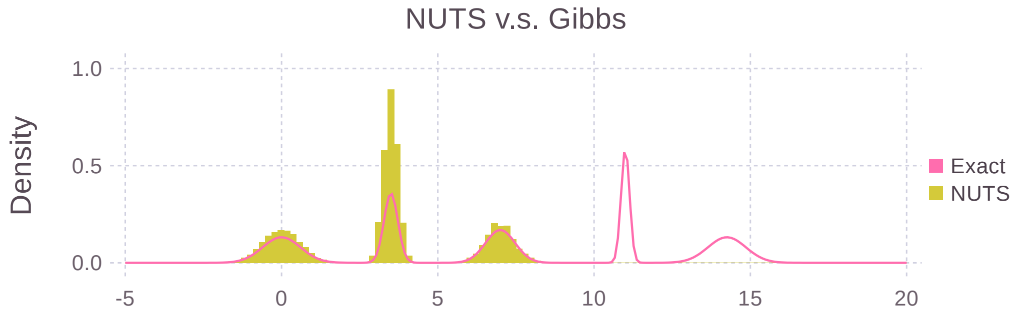
```
[Gibbs] Finished with
  Running time    = 5.898401496999998;
Running NUTS
[Turing] looking for good initial eps...
[Turing.NUTS] found initial ϵ: 2.0
[Turing.WARNING]: Numerical error has been found in gradients.
  in verifygrad(::Array{Float64,1}) at ad.jl:100
[Turing.WARNING]: Numerical error has been found in gradients.
  in verifygrad(::Array{Float64,1}) at ad.jl:100
[Turing.WARNING]: Numerical error has been found in gradients.
  in verifygrad(::Array{Float64,1}) at ad.jl:100
[Turing.WARNING]: Numerical error has been found in gradients.
  in verifygrad(::Array{Float64,1}) at ad.jl:100
[Turing.WARNING]: Numerical error has been found in gradients.
  in verifygrad(::Array{Float64,1}) at ad.jl:100
[Turing.WARNING]: Numerical error has been found in gradients.
  in verifygrad(::Array{Float64,1}) at ad.jl:100
[Turing.WARNING]: Numerical error has been found in gradients.
  in verifygrad(::Array{Float64,1}) at ad.jl:100
[Turing]:  Adapted ϵ = 0.016520651907651517, 1000 HMC iterations is used
for adaption.
  in step(::Function, ::Turing.Sampler{Turing.NUTS}, ::Turing.VarInfo,
::Bool) at nuts.jl:135
[NUTS] Finished with
  Running time        = 28.542914961000044;
  #lf / sample        = 41.9492;
  #evals / sample     = 41.9494;
```

3.2 Visualization of setting 1

[6]

```
make_plot(x_gibbs, x_nuts)
```





3.3 Sampling from MoG(s) with setting 2

[8]

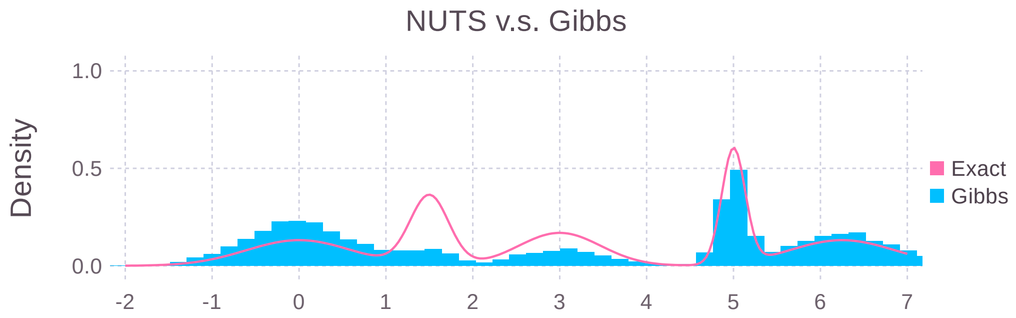
```
μ = [    0,     1,      2, 3.5, 4.25] + 0.5 * collect(0:4)

println("Running Gibbs")
chain_gibbs = sample(GMM(p, μ, σ), Gibbs(round(Int,N/K), PG(10, 1, :z),
HMC(K-1, 0.2, 4, :x); thin=false))
x_gibbs = map(x_arr -> x_arr[1], chain_gibbs[:x]);

println("Running NUTS")
chain_nuts = sample(cGMM(p, μ, σ), NUTS(N, 0.65))
x_nuts = map(x_arr -> x_arr[1], chain_nuts[:x]);
```

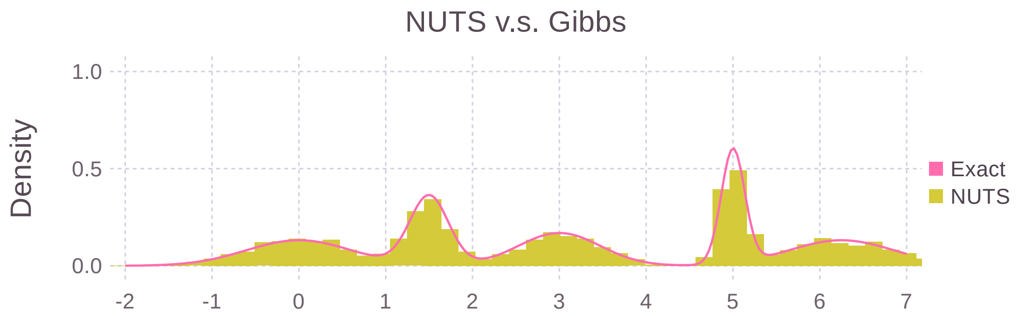
```
[Gibbs] Finished with
  Running time    = 3.4638651700000005;
Running NUTS
[Turing] looking for good initial eps...
[Turing.NUTS] found initial ϵ: 2.0
[Turing.WARNING]: Numerical error has been found in gradients.
  in verifygrad(::Array{Float64,1}) at ad.jl:100
[Turing.WARNING]: Numerical error has been found in gradients.
  in verifygrad(::Array{Float64,1}) at ad.jl:100
[Turing.WARNING]: Numerical error has been found in gradients.
  in verifygrad(::Array{Float64,1}) at ad.jl:100
[Turing.WARNING]: Numerical error has been found in gradients.
  in verifygrad(::Array{Float64,1}) at ad.jl:100
[Turing.WARNING]: Numerical error has been found in gradients.
  in verifygrad(::Array{Float64,1}) at ad.jl:100
[Turing]:  Adapted ϵ = 0.03350258731495017, 1000 HMC iterations is used
for adaption.
  in step(::Function, ::Turing.Sampler{Turing.NUTS}, ::Turing.VarInfo,
::Bool) at nuts.jl:135
[NUTS] Finished with
  Running time        = 30.433535454999983;
  #lf / sample        = 39.7549;
  #evals / sample     = 39.7551;
  pre-cond. diag mat  = [2.37182].
WARNING: Method definition GMM_model_##280() in module Main at In[3]:2
overwritten at In[3]:2.
WARNING: Method definition #GMM_model_##280(Array{Any, 1},
```

3.4 Visualization of setting 2

[9]

```
make_plot(x_gibbs, x_nuts, -2, 7)
```


NUTS v.s. Gibbs
Density
1.0
0.5
0.0
-2
-1
0
1
2
3
4
5
6
7
Exact
Gibbs


NUTS v.s. Gibbs
Density
1.0
0.5
0.0
-2
-1
0
1
2
3
4
5
6
7
Exact
NUTS