[1]

```
using Distributions
using Turing
using Stan

# Load data; loaded data is a list of dict named `ldastandata`
include(Pkg.dir("Turing")*"/example-models/stan-models/lda-stan.data.jl")
topicdata = ldastandata[1]

# Load model
include(Pkg.dir("Turing")*"/example-models/stan-models/lda.model.jl")
#= NOTE: loaded model is defined as below
@model ldamodel(K, V, M, N, w, doc, beta, alpha) = begin
  theta = Vector{Vector{Real}}(M)
  for m = 1:M
    theta[m] ~ Dirichlet(alpha)
  end

  phi = Vector{Vector{Real}}(K)
  for k = 1:K
    phi[k] ~ Dirichlet(beta)
  end

  phi_dot_theta = [log([dot(map(p -> p[i], phi), theta[m]) for i = 1:V]) for m=1:M]
  for n = 1:N
    Turing.acclogp!(vi, phi_dot_theta[doc[n]][w[n]])
  end
end
=#
```

```
Environment variable JULIA_SVG_BROWSER not found.
WARNING: using Stan.CMDSTAN_HOME in module Main conflicts with an existing identifier.
ldamodel (generic function with 9 methods)
```

[2]

```
setchunksize(100)     # increase AD chunk-size to 100
```

```
[Turing]: AD chunk size is set as 100
100
```

[3]

```
samples = sample(ldamodel(data=topicdata), NUTS(1000, 0.65))
```

[4]

```
# Load visualization script for topic models; visualization function is
called `vis_topic_res`
include(Pkg.dir("Turing")*"/example-models/stan-
models/topic_model_vis_helper.jl")

@doc vis_topic_res  # show the usage of the visualization function
```

```
WARNING: using DataFrames.@~ in module Main conflicts with an existing
identifier.
```

Function for visualization topic models.

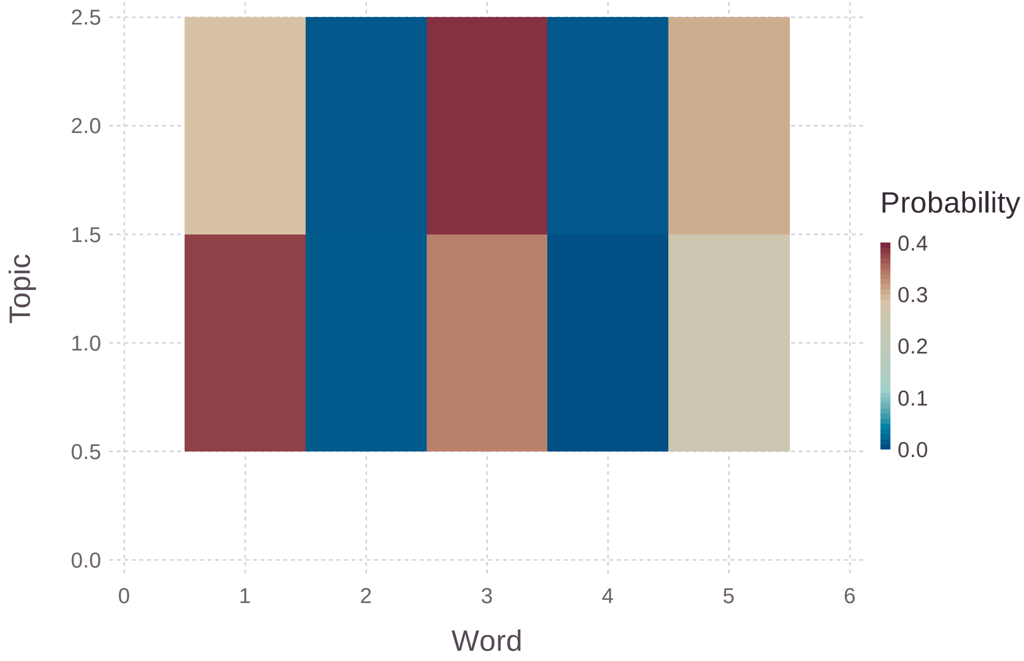
Usage:

`vis_topic_res(samples, K, V, avg_range)`

- `samples` is the chain return by `sample()`
- `K` is the number of topics
- `V` is the size of vocabulary
- `avg_range` is the end point of the running average

[5]

```
vis_topic_res(samples, topicdata["K"], topicdata["V"], 1000)
```



[6]

```
# Load data; loaded data is a list of dict named `nbstandata`
include(Pkg.dir("Turing")*"/example-models/stan-models/MoC-
stan.data.jl")
topicdata2 = nbstandata[1]
```

```
# Load model
include(Pkg.dir("Turing")*"/example-models/stan-models/MoC.model.jl")
#= NOTE: loaded model is defined as below
@model nbmodel(K, V, M, N, z, w, doc, alpha, beta) = begin
  theta ~ Dirichlet(alpha)

  phi = Array{Any}(K)
  for k = 1:K
    phi[k] ~ Dirichlet(beta)
  end

  log_theta = log(theta)
  Turing.acclogp!(vi, sum(log_theta[z[1:M]]))

  log_phi = map(x->log(x), phi)
  for n = 1:N
    Turing.acclogp!(vi, log_phi[z[doc[n]]][w[n]])
  end

  phi
end

=#
```

```
nbmodel (generic function with 10 methods)
```

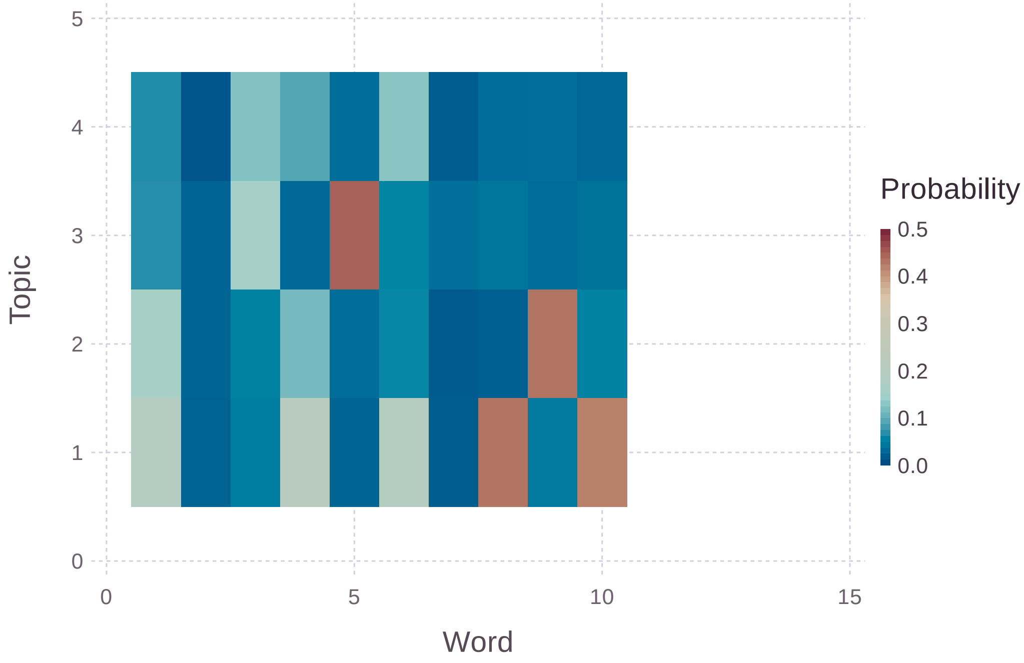
[7]

```
samples2 = sample(nbmodel(data=topicdata2), NUTS(1000, 0.65))
```

```
  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.NUTS] found initial ϵ: 0.25
[Turing.WARNING]: Numerical error has been found in gradients.
  in verifygrad(::Array{Float64,1}) at ad.jl:100
[Turing]:  Adapted ϵ = 0.14342147248586115, 200 HMC iterations is used
for adaption.
  in adapt_step_size(::Turing.Sampler{Turing.NUTS}, ::Float64, ::Float64)
at adapt.jl:17
[NUTS] Finished with
  Running time        = 351.79663569199965;
  #lf / sample        = 26.958;
  #evals / sample     = 26.96;
  pre-cond. diag mat  =
[1.61067,1.41474,1.80153,296.81,6.04135,11.9786,6.44429,1.0,16.0085,3.253
05,9.29968,2.12112,2.12419,159.515,4.88626,3.7894,3.25058,2.8811,5.91825,
2.07057,1.88234,2.10734,2.05558,170.528,17.1497,4.4157,3.13239,2.36315,1.
16818,5.28546,4.58307,6.46386,3.03735,261.914,2.88004,1.52325,42.8794,2.4
461,2.12965,3.80963,4.7378,7.079,3.55813,383.289].
[NUTS] Sampling...100% Time: 0:05:52
Object of type "Turing.Chain"

Iterations = 1:1000
Thinning interval = 1
Chains = 1
Samples per chain = 1000
```

```
[8]  vis_topic_res(samples2, topicdata2["K"], topicdata2["V"], 1000)
```


Topic
Word
Probability
0.5
0.4
0.3
0.2
0.1
0.0
0
1
2
3
4
5
0
5
10
15

[ ]