

NRES 746, Week 9, Lecture 2: Getting Stan Running (take-home)

Name: _____

Due Monday. We'll start Monday's class by sorting out any installation problems, so bring your laptop. Start the installation early: it downloads and builds a fair amount of software and can take 20-30 minutes, most of it waiting.

Using AI: each coding question is labeled with a recommendation. After each one, circle how you actually used AI. Installation errors can be a good use for an AI assistant (or a web search): paste in the error message and ask what it means.

Part 1: Install Stan

We'll use **cmdstanr**, the R interface to Stan. The full instructions are at <https://mc-stan.org/cmdstanr/articles/cmdstanr.html>; the short version:

1. **Get a C++ toolchain.** Windows: install Rtools for your version of R (<https://cran.r-project.org/bin/windows/Rtools/>). Mac: in the Terminal, run `xcode-select --install`.
2. **Install the package and Stan itself**, in R:

```
install.packages("cmdstanr",  
                 repos = c("https://stan-dev.r-universe.dev", getOption("repos")))  
library(cmdstanr)  
check_cmdstan_toolchain(fix = TRUE) # checks (and tries to fix) the C++ setup  
install_cmdstan(cores = 2)         # downloads and builds Stan; be patient  
cmdstan_version()
```

1a. What does `cmdstan_version()` report? If the installation failed, write the error message you got stuck on.

CmdStan version: _____

Part 2: Fit Wednesday's myxo model

Here is the model from Wednesday's lecture (myxomatosis virus titers, Gamma distribution, flat priors on the positive values). Copy it into an R script and run it. Stan compiles the model to C++ the first time, which takes up to a minute.

```
library(cmdstanr)  
myxo <- read_csv(  
  "https://kevintshoemaker.github.io/NRES-746/data/MyxoTiter_sum.csv")  
titer <- myxo$titer[myxo$grade == 1]  
  
myxo_code <- "  
data {  
  int<lower=0> N;  
  vector<lower=0>[N] titer;  
}  
parameters {  
  real<lower=0> shape;
```

```

    real<lower=0> scale;
  }
  model {
    titer ~ gamma(shape, 1 / scale); // Stan's gamma uses shape and rate
  }
"
myxo_model <- cmdstan_model(write_stan_file(myxo_code))
fit <- myxo_model$sample(data = list(N = length(titer), titer = titer),
                        chains = 4, parallel_chains = 4, seed = 746)
fit$summary()
fit$diagnostic_summary()

```

2a. Record these from the output (yours may differ a little from a classmate's):

$\hat{R}$ for shape: _____ effective sample size (`ess_bulk`) for shape: _____

Number of divergent transitions (all chains): _____

95% credible interval for shape (hint: `quantile()` on `fit$draws("shape", format = "draws_matrix")`):
 _____ to _____

2b. (*Recommended: no AI.*) Now make the sampler work much less hard: rerun the `sample()` call with `iter_warmup = 20` and `iter_sampling = 50` added. How do $\hat{R}$ and the effective sample size change, and why would you not trust these results?

$\hat{R}$ for shape: _____ `ess_bulk` for shape: _____

AI use (circle one): no AI / AI helped (e.g., debugging) / AI wrote the code

Part 3: Reflection

Where did you use AI, if anywhere, and what did you have to check or fix? If you didn't use it, what was hardest to get working?
