Stan Modeling with bbr.bayes : : CHEAT SHEET

Setup

bbr is an R interface for managing Stan and other modeling software, with a focus on reproducibility and traceability.

To use bbr with Stan, you need to have cmdstan installed. Follow the "Getting Started with CmdStanR" instructions to ensure you have Stan and CmdStan configured correctly.

Install CmdStan:

```
cmdstanr::check_cmdstan_toolchain()
cmdstanr::install_cmdstan()
```

You need to load both the core bbr package and the bbr.bayes package.

```
library(bbr)
library(bbr.bayes)
```

Creating a Model de novo

To submit or interact with models in bbr you need a model object. You can create a model object (and associated .yaml file) de novo using new_model().

```
MODEL_DIR <- here::here("model/stan")
mod1 <- new_model(file.path(MODEL_DIR, "mod1"),
                  .model_type = "stan")
```

This creates several "scaffold" files on disk. You can open them with these helper functions:

```
open_stanmod_file(mod1)
open_standata_file(mod1)
open_staninit_file(mod1)
```

See below for copying from a parent model.

You can read a previously created model with read_model().

```
mod2 <- read_model(file.path(MODEL_DIR, "mod2"))
```

Submitting a Model

Calling submit_model() ultimately triggers a call to cmdstanr$sample()

Pass arguments through to $sample()

```
mod1 <- set_stanargs(
  mod1, list(chains = 4, seed = 1234)
get_stanargs(mod1) # prints currently set args
```

Check that model is ready to submit (optional)

```
check_stan_model(mod1)
```

Preview the data set that will be used (optional)

```
standata <- build_data(mod1)
```

Load the cmdstanr::CmdStanModel object (optional)

```
stanmodel <- get_model_path(mod1) %>%
  cmdstanr::cmdstan_model(compile = FALSE)
```

Submit a model to be run

```
fit1 <- mod1 %>% submit_model()
```

submit_model() returns a CmdStanMCMC object.

Model Fit Object

The read_fit_model() function returns the cmdstan::CmdStanMCMC object, for a model that has finished running.

```
fit1 <- read_fit_model(mod1)

# use cmdstan methods, e.g.
fit1$cmdstan_diagnose()
fit1$summary()
```

Convert to a Draws Object

bbr.bayes has methods for the posterior::as_draws() family of functions. Pass a model or fit object.

```
draws1 <- as_draws(mod1)

# use posterior methods
summarize_draws(draws1)
rhat(draws1)
thin_draws(draws1)

# get draws as a tibble
draws_df1 <- as_draws_df(mod1)
```

Copying from a Parent Model

copy_model_from() creates a new model, based on an existing model. It begins as an exact copy, with only the name changed.

```
mod2 <- copy_model_from(mod1, "mod2")

# open any files you want to change
open_stanmod_file(mod2)
open_staninit_file(mod2)
# compare, once you have made changes
model_diff(mod2)
model_diff(mod2, .file = "init")
```

Standalone Generated Quantities

copy_model_as_stan_gq() creates a special class of model for running generated quantities without sampling.

```
mod_gq <- copy_model_as_stan_gq(mod2)
open_stanmod_file(mod_gq)
open_stan_fitted_params_file(mod_gq)

# runs $generate_quantities()
submit_model(mod_gq)
```

METRUM RESEARCH GROUP

Example Workflow

Create initial model

```
MODEL_DIR <- here::here("model/stan")
mod1 <- new_model(
 file.path(MODEL_DIR, 1001),
 .model_type = "stan") %>%
 add_tags("base models")
```

Submit model and view results

```
fit1 <- mod1 %>% submit_model()
fit1$cmdstan_diagnose()
fit1$summary(variables = c("lp__", "emax"))

mod1 <- mod1 %>% add_notes(
      "Divergent transitions, adjusting delta")
```

Create new model based on initial model

```
mod2 <- copy_model_from(
 mod1, "mod2", .inherit_tags = TRUE) %>%
 set_stanargs(list(adapt_delta = 0.98))
```

Edit stan file and compare to parent

```
open_stanmod_file(mod2)
model_diff(mod2)
```

Submit new model and view results

```
fit2 <- submit_model(mod2) ; ...
```

Add notes (be sure to reassign to object)

```
mod2 <- mod2 %>%
   add_notes("MCMC sampling looking good")
```

Create generated quantities model

```
mod_gq <- copy_model_as_stan_gq(mod2)
open_stanmod_file(mod_gq)
sims2 <- submit_model(mod_gq)
sims2$draws() # get gq for predictive checks
```

Continue to next model...

```
mod3 <- copy_model_from( mod2) %>%
 replace_tags("base models", "covariate mods")
```

Model Annotation: Tags and Notes and Description

The model object has tags, notes, and description fields, to annotate the model during development. Tags are concise and can be used for filtering and organizing your models, while notes are free form text to notate decisions and observations. The description field must be a single string.

Note: when adding or modifying these attributes, **you must reassign the modified model object**. You can pipe several modifications together.

```
mod4 <- mod4 %>%
  add_tags("covariate mods") %>%
  replace_tags("centered params", "non-centered params") %>%
  add_notes("First model to use non-centered parameterization")
```

Defining a glossary of tags

Tags are most useful when defined in a glossary. See “Details” of ?modify_tags for recommendations.

Modifying the model object

Helper functions exist to add, replace, or remove the tags, notes, description, and based_on fields.

```
mod1 <- mod1 %>% add_description("Base model")
mod1 <- mod1 %>% replace_based_on("1001", "1002")
```

Creating a Run Log

Pull all models in a given directory into a tibble with run_log().

```
log_df <- run_log(MODEL_DIR)
```

Use stan_summary_log() to pull simple diagnostics into a tibble, or add_stan_summary() to append this output to a run log tibble.

```
log_df <- run_log(MODEL_DIR) %>%
    add_stan_summary() %>%          # joins in diagnostics columns
    collapse_to_string(notes) %>%   # formatting for printing
    select(run, lp___median, lp___rhat, num_divergent, notes)
```

run	lp___median	lp___rhat	num_divergent	notes
1001	-114.065	1.00050	3	“Divergent transitions, adjusting delta”
1002	-111.832	1.00153	0	“MCMC sampling looking good”
1003	-109.239	1.00467	0	“Adding covariates, non-centered params”

Checking model and data are up to date

Pass a model object or run log tibble to check_up_to_date() to verify none of the control streams or data files on disk have changed since the models were run.

This checks if any model or data files have changed since the model was most recently submitted.

Assorted Helper Functions

```
check_stan_model(mod1)     # check if ready to submit
check_up_to_date(mod1)     # check if changed since submitted
get_model_id(log_df)       # return name of model as a string
res <- build_data(mod1)    # build Stan input data
add_stanmod_file(mod1,
     .source = "/path/to/source.stan")  # add external files
model_diff(mod2, .file = "standata")    # compare against parent

# get absolute file paths
mod_path <- get_model_path(mod1)
dir_path <- get_output_dir(mod1)
yaml_path <- build_path_from_model(mod1, "-param.yaml")
```

METRUM RESEARCH GROUP