Study 3: VAR(1) with Normal Margins — Gaussian vs. Clayton Copula

Table of contents

```r
suppressPackageStartupMessages({
  library(dplyr)
  library(tidyr)
  library(readr)
  library(ggplot2)
  library(stringr)
  library(knitr)
})
```

```
# paths
DATA_DIR   <- file.path("data")
RES_DIR    <- file.path("results")
EXPORT_DIR <- file.path(RES_DIR, "exported_tables")
dir.create(EXPORT_DIR, showWarnings = FALSE, recursive = TRUE)

files <- list(
  cond   = file.path(RES_DIR, "summary_conditions.csv"),
  rep    = file.path(RES_DIR, "summary_replications.csv"),
  design = file.path(DATA_DIR, "sim_conditions.rds")
)

if (!all(file.exists(unlist(files)))) {
  message("Missing required input files. Please run the Study 3 pipeline and analysis_singlel
  if (knitr::is_html_output() || knitr::is_latex_output()) knitr::knit_exit()
}
```

1. Design Overview

Table 1: Study 3 design factors.

Factor	Levels
Time Series Length (T)	50, 100, 200
Copula Parameter (Clayton theta)	0.5, 1, 2, 4, 8
VAR Parameters (Set A)	**Set A**: $\begin{pmatrix} 0.40 & 0.10 \\ 0.10 & 0.40 \end{pmatrix}$
Margins	Standard Normal (mean 0, variance 1)

2. Data Loading & Preparation

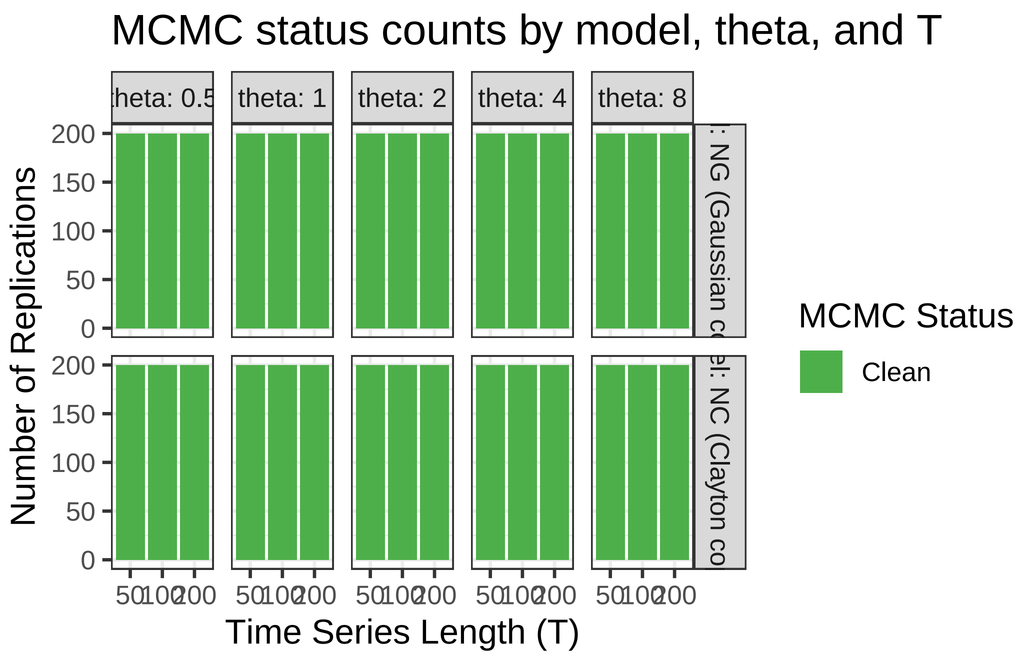
```
# condition-level summary (aggregated metrics) and replication-level summary
cond_raw <- read_csv(files$cond, show_col_types = FALSE)
rep_raw  <- read_csv(files$rep,  show_col_types = FALSE)

# attach design info (T, theta, VARset, direction; VARset is always 'A' here)
```

```r
design_min <- design |>
  dplyr::select(condition_id, T, theta, VARset, direction)

cond_raw <- cond_raw |>
  left_join(design_min, by = "condition_id")

rep_raw <- rep_raw |>
  filter(!is.na(param)) |>
  left_join(design_min, by = "condition_id")

# keep only NG (Gaussian copula) and NC (Clayton copula)
keep_models <- c("NG", "NC")
cond_raw <- cond_raw |> filter(model %in% keep_models)
rep_raw  <- rep_raw  |> filter(model %in% keep_models)

# parameter order
param_levels <- c(
  "theta",        # NC copula parameter
  "rho",          # NG copula parameter (no truth in this DGP)
  "sigma[1]", "sigma[2]",
  "mu[1]","mu[2]","phi11","phi12","phi21","phi22"
)

# clean labels and factors
prep_data <- function(df) {
  existing_params <- intersect(param_levels, unique(df$param))

  df |>
    mutate(
      param    = factor(param, levels = existing_params),
      T        = factor(T),
      theta_val= theta,
      theta    = factor(theta, levels = sort(unique(theta))),
      VARset   = factor(VARset, levels = sort(unique(VARset))),
      Model    = case_when(
        model == "NG" ~ "NG (Gaussian copula)",
        model == "NC" ~ "NC (Clayton copula)"
      ),
      Model = factor(Model, levels = c("NG (Gaussian copula)", "NC (Clayton copula)"))
    )
}
```

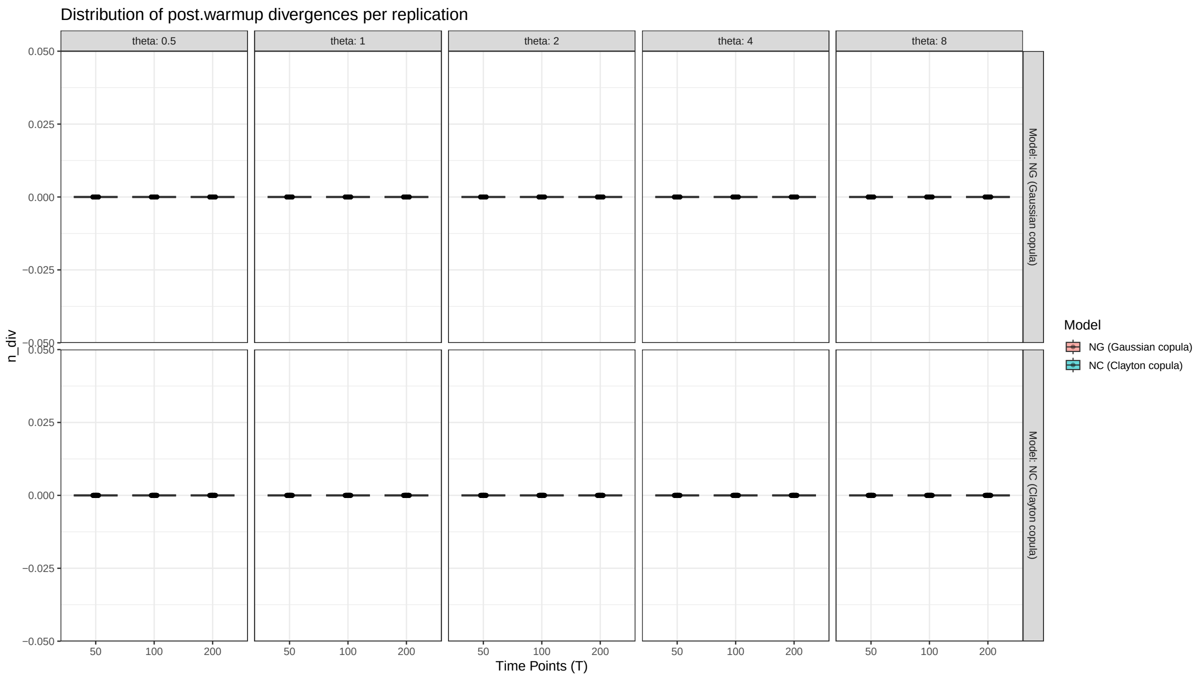
```
cond   <- prep_data(cond_raw) |>
  mutate(RMSE = sqrt(mean_bias^2 + coalesce(emp_sd^2, 0)))
rep_df <- prep_data(rep_raw)

core_params <- c("mu[1]","mu[2]","phi11","phi12","phi21","phi22")
```

3. MCMC Classification & Overview

```
RHAT_THRESHOLD <- 1.01

rep_df <- rep_df |>
  mutate(
    n_div_clean = ifelse(is.na(n_div), 0, n_div),
    mcmc_status = case_when(
      is.na(max_rhat) | status != "ok" ~ "Failed/Error",
      max_rhat > RHAT_THRESHOLD | n_div_clean > 0 ~ "Problematic",
      TRUE ~ "Clean"
    ),
    mcmc_status = factor(mcmc_status, levels = c("Clean", "Problematic", "Failed/Error"))
  )

mcmc_summary <- rep_df |>
  distinct(condition_id, rep_id, Model, mcmc_status, T, theta) |>
  group_by(Model, theta, T, mcmc_status) |>
  summarise(Count = n(), .groups = "drop")

ggplot(mcmc_summary, aes(x = T, y = Count, fill = mcmc_status)) +
  geom_bar(stat = "identity", position = "stack") +
  facet_grid(Model ~ theta, labeller = label_both) +
  labs(x = "Time Series Length (T)", y = "Number of Replications", fill = "MCMC Status",
       title = "MCMC status counts by model, theta, and T") +
  theme_bw(base_size = 13) +
  scale_fill_manual(values = c("Clean" = "#4daf4a", "Problematic" = "#ff7f00", "Failed/Error"
```



```r
# Use a parameter present in both models to get one row per replication
div_dist_data <- rep_df |>
  filter(param == "phi11") |>
  distinct(condition_id, rep_id, Model, T, theta, n_div_clean, mcmc_status) |>
  filter(mcmc_status != "Failed/Error")

ggplot(div_dist_data, aes(x = T, y = n_div_clean, fill = Model)) +
  geom_boxplot(outlier.shape = NA, alpha = 0.6, position = position_dodge(width = 0.8)) +
  geom_point(size = 1.5, alpha = 0.4, position = position_jitterdodge(jitter.width = 0.2, doc
  facet_grid(Model ~ theta, labeller = label_both) +
  theme_bw(base_size = 13) +
  labs(title = "Distribution of post-warmup divergences per replication",
       y = "n_div", x = "Time Points (T)")
```



4. Helper Utilities

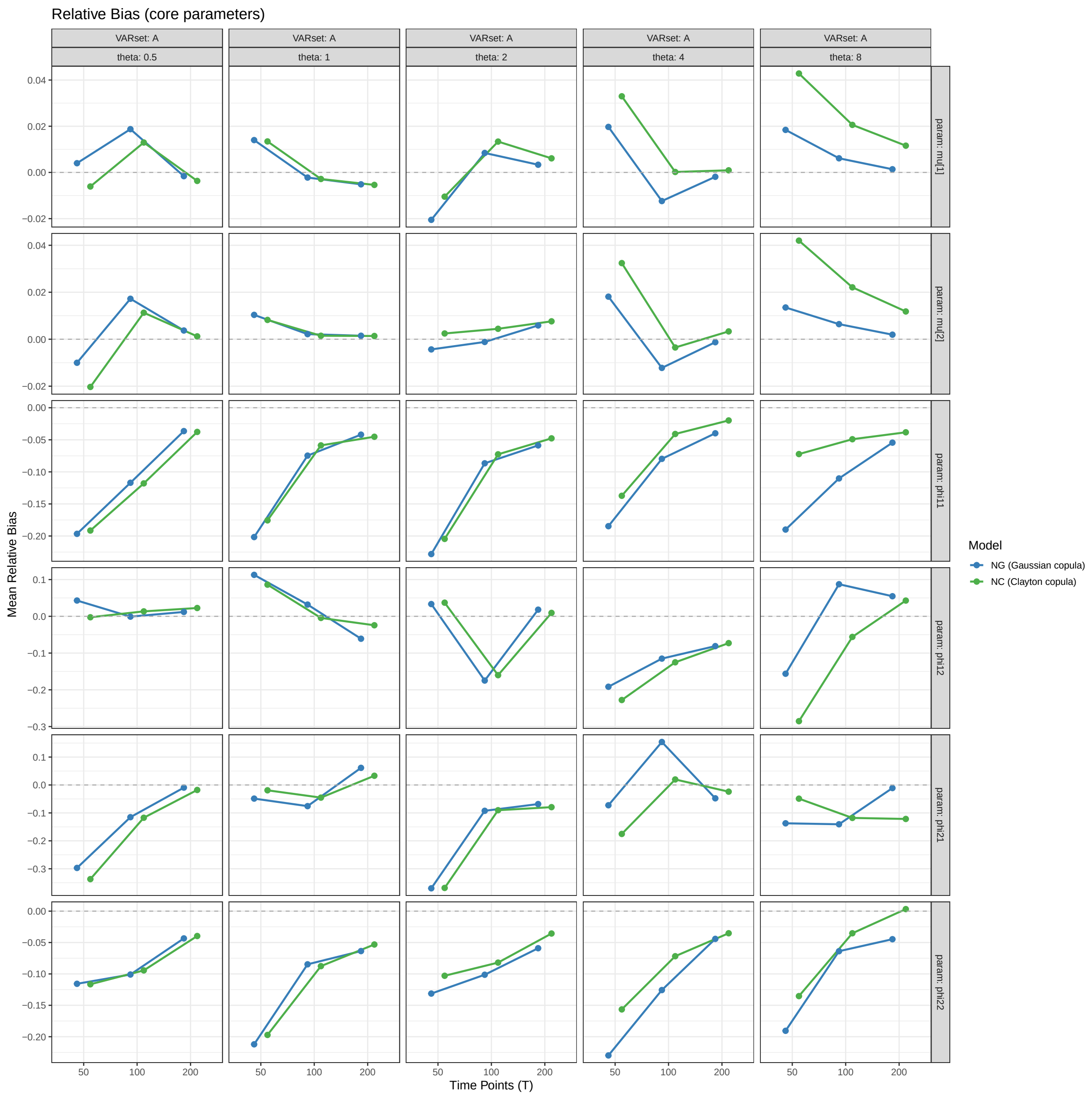
```
theme_standard <- theme_bw(base_size = 13)
dodge_width    <- 0.5
model_colors   <- c("NG (Gaussian copula)" = "#377eb8",
                    "NC (Clayton copula)"  = "#4daf4a")

plot_metric <- function(data, metric_col, ylab, title, use_free_y = FALSE, ylims = NULL) {
  data_filtered <- data |>
    filter(
      !is.na(.data[[metric_col]]),
      !is.na(T),
      !is.na(param),
      !is.na(VARset),
      !is.na(theta)
    ) |>
    droplevels()
  if (nrow(data_filtered) == 0) return(NULL)

  p <- ggplot(data_filtered, aes(x = T, y = .data[[metric_col]], color = Model, group = Mode
```

```r
    geom_line(position = position_dodge(dodge_width), linewidth = 1) +
    geom_point(position = position_dodge(dodge_width), size = 2.5) +
    facet_grid(param ~ VARset + theta, labeller = label_both,
               scales = ifelse(use_free_y, "free_y", "fixed")) +
    theme_standard +
    scale_color_manual(values = model_colors) +
    labs(title = title, y = ylab, x = "Time Points (T)")

  if (metric_col %in% c("mean_rel_bias", "sd_bias")) {
    p <- p + geom_hline(yintercept = 0, linetype = "dashed", color = "darkgrey")
  } else if (metric_col == "coverage_95") {
    p <- p + geom_hline(yintercept = 0.95, linetype = "dashed", color = "darkgrey")
  }
  if (!is.null(ylims)) p <- p + coord_cartesian(ylim = ylims)
  p
}

generate_core_plots <- function() {
  data_subset <- cond |> filter(param %in% core_params)
  cov_ylims   <- c(0.8, 1.0)
  list(
    bias     = plot_metric(data_subset, "mean_rel_bias", "Mean Relative Bias",
                           "Relative Bias (core parameters)", use_free_y = TRUE),
    coverage = plot_metric(data_subset, "coverage_95", "Empirical Coverage",
                           "95% Coverage (core parameters)", ylims = cov_ylims),
    rmse     = plot_metric(data_subset, "RMSE", "RMSE",
                           "RMSE (core parameters)", use_free_y = TRUE),
    post_sd  = plot_metric(data_subset, "mean_post_sd", "Mean Posterior SD",
                           "Posterior SD (core parameters)", use_free_y = TRUE),
    sdbias   = plot_metric(data_subset, "sd_bias", "SD-Bias",
                           "SD-Bias (core parameters)", use_free_y = TRUE)
  )
}
```

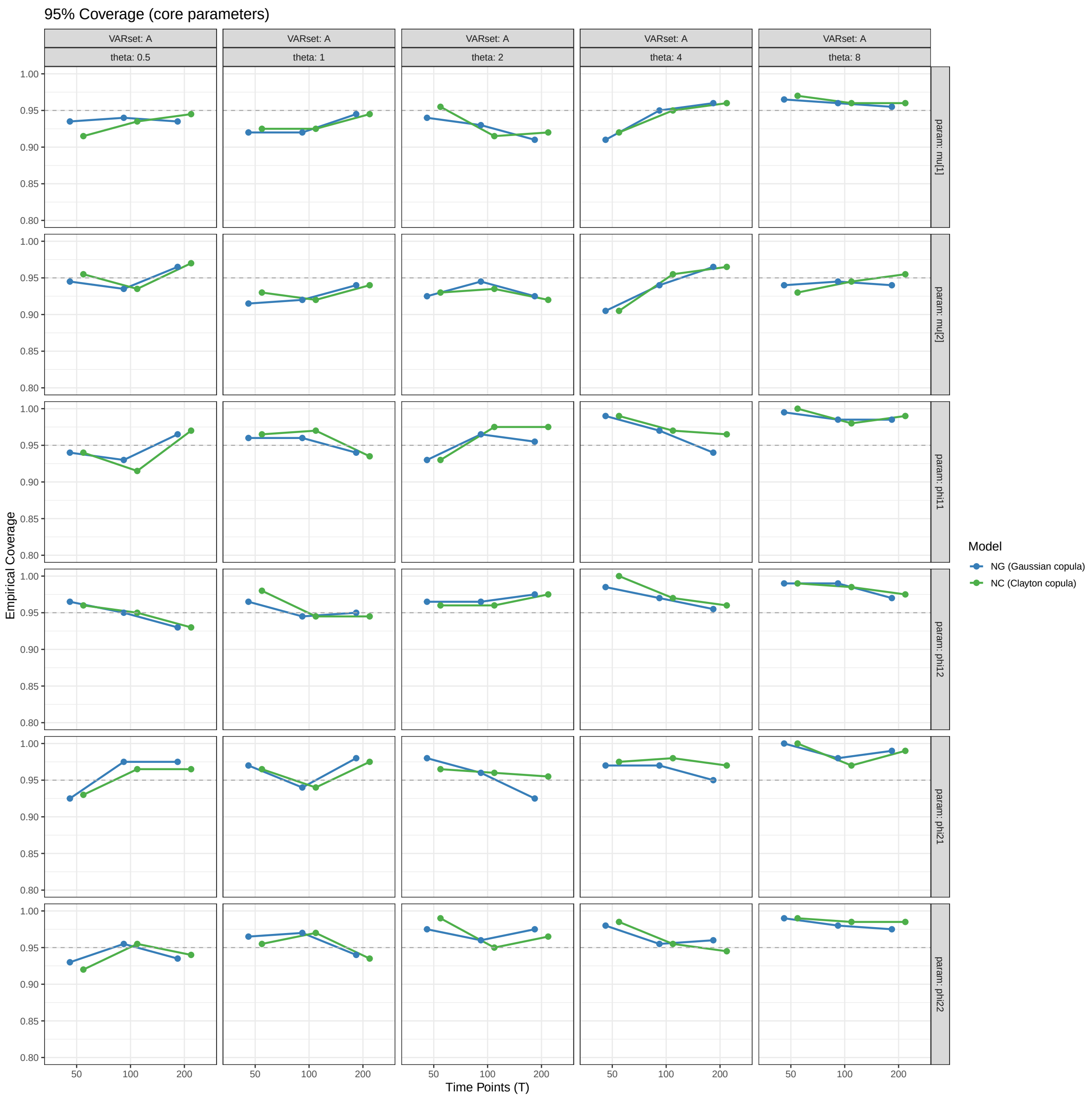
5. Core Parameters (mu, Phi)

```r
plots_core <- generate_core_plots()
```

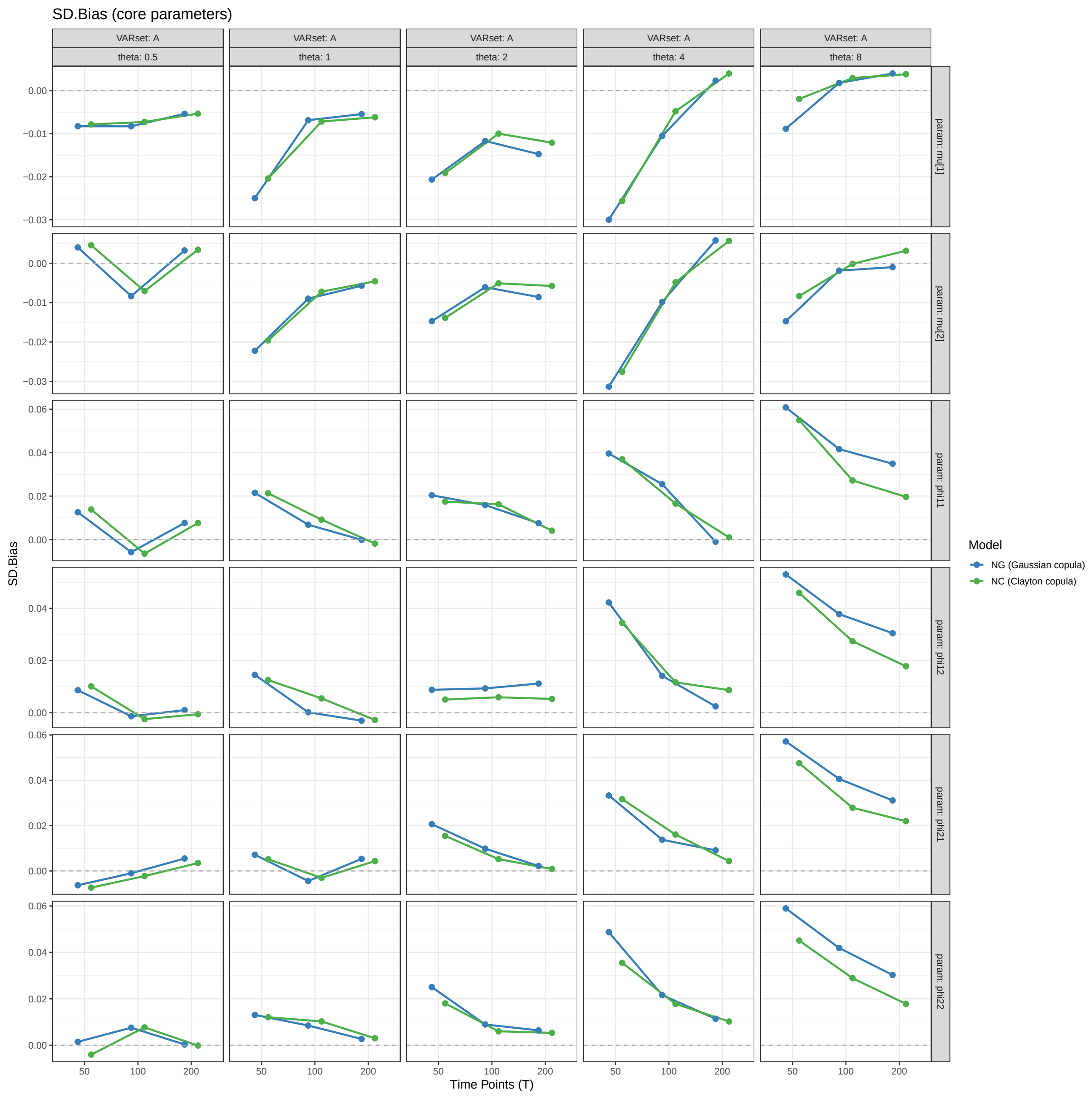
```r
print(plots_core$bias)
```



```
print(plots_core$coverage)
```



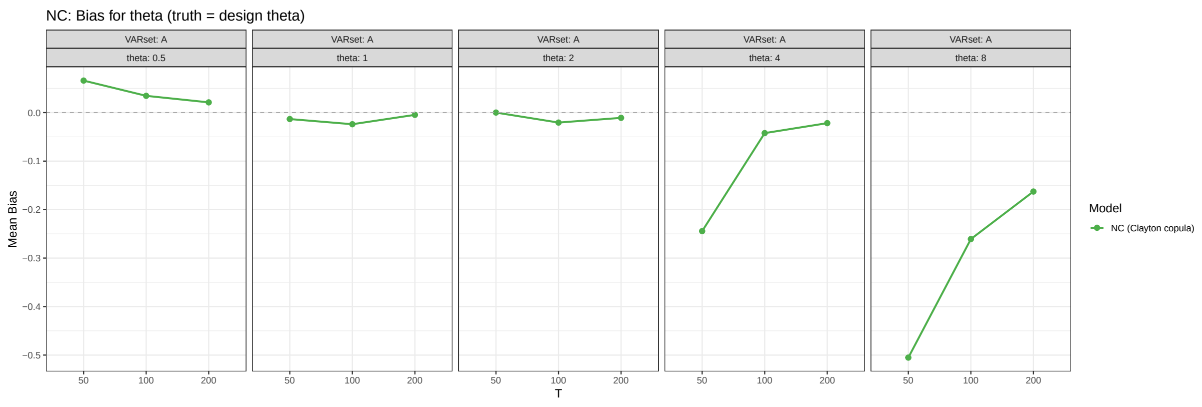
```
print(plots_core$sdbias)
```



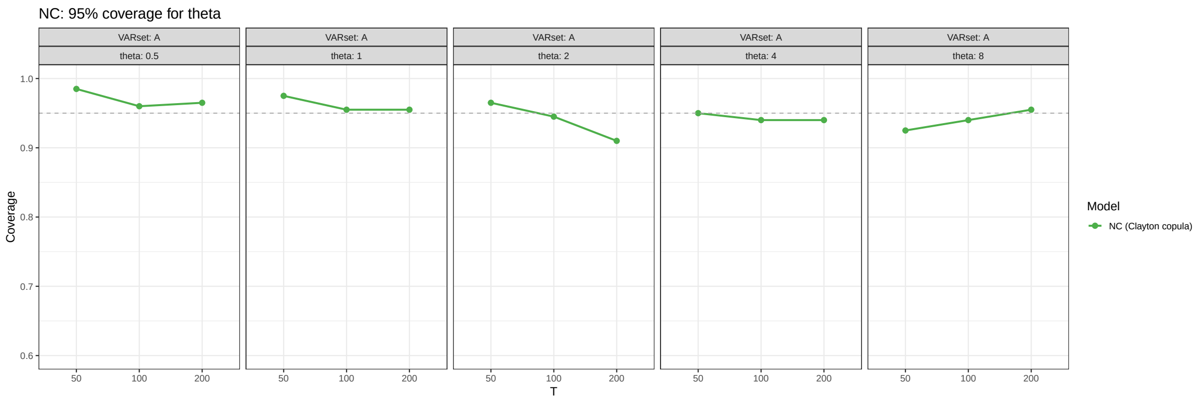
6. Copula Parameters

6.1 NC (Clayton copula): theta recovery (truth = design theta)

```
theta_data <- cond |>
  filter(Model == "NC (Clayton copula)", param == "theta")

if (nrow(theta_data) > 0) {
  ggplot(theta_data, aes(x = T, y = mean_bias, color = Model, group = Model)) +
    geom_line(position = position_dodge(0.3), linewidth = 1) +
    geom_point(position = position_dodge(0.3), size = 2.5) +
    geom_hline(yintercept = 0, linetype = "dashed", color = "darkgrey") +
    facet_grid(. ~ VARset + theta, labeller = label_both) +
    theme_standard +
    scale_color_manual(values = model_colors) +
    labs(title = "NC: Bias for theta (truth = design theta)", y = "Mean Bias", x = "T")
}
```

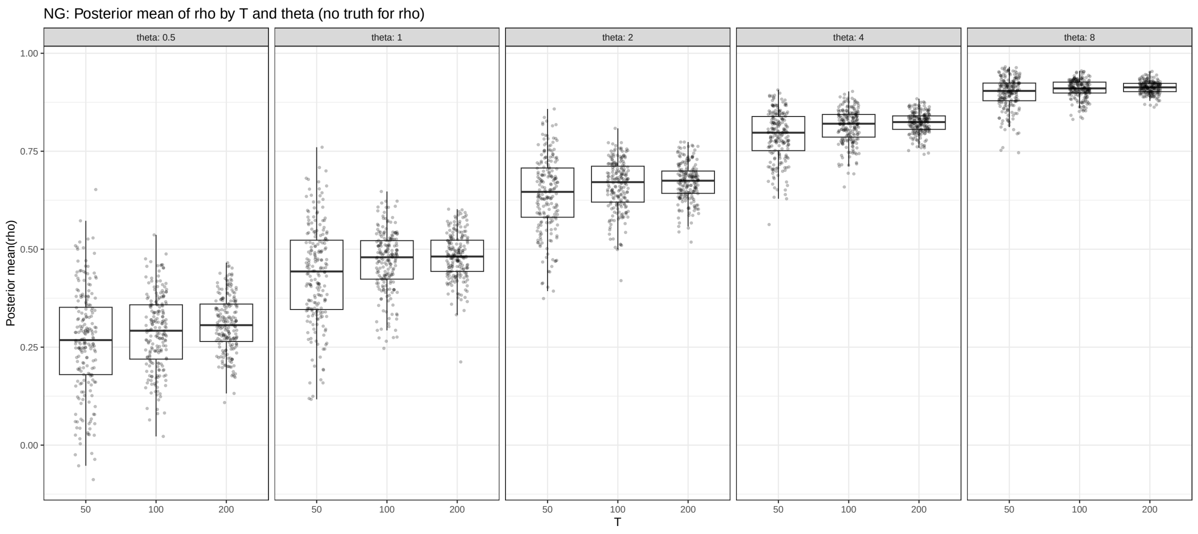


```
if (nrow(theta_data) > 0) {
  ggplot(theta_data, aes(x = T, y = coverage_95, color = Model, group = Model)) +
    geom_line(position = position_dodge(0.3), linewidth = 1) +
    geom_point(position = position_dodge(0.3), size = 2.5) +
    geom_hline(yintercept = 0.95, linetype = "dashed", color = "darkgrey") +
    facet_grid(. ~ VARset + theta, labeller = label_both) +
    theme_standard +
    scale_color_manual(values = model_colors) +
    labs(title = "NC: 95% coverage for theta", y = "Coverage", x = "T") +
    coord_cartesian(ylim = c(0.6, 1.0))
}
```



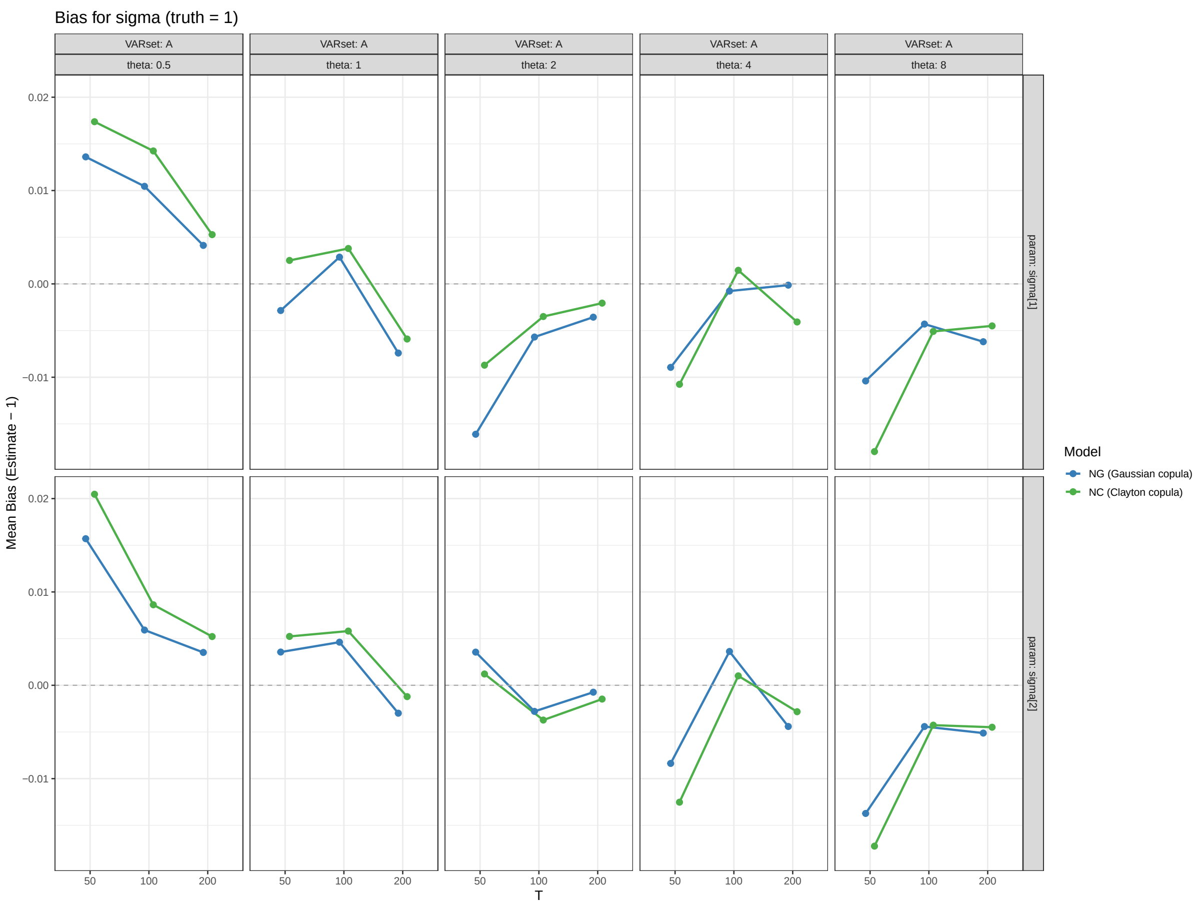
6.2 NG (Gaussian copula): rho posterior means (no ground truth under Clayton DGP)

```
rho_rep <- rep_df |>
  filter(Model == "NG (Gaussian copula)", param == "rho")

if (nrow(rho_rep) > 0) {
  ggplot(rho_rep, aes(x = T, y = post_mean)) +
    geom_boxplot(outlier.shape = NA) +
    geom_jitter(width = 0.15, alpha = 0.25, size = 1) +
    facet_grid(. ~ theta, labeller = label_both) +
    theme_standard +
    labs(title = "NG: Posterior mean of rho by T and theta (no truth for rho)", y = "Posteri
}
```



7. Marginal Variances (sigma)

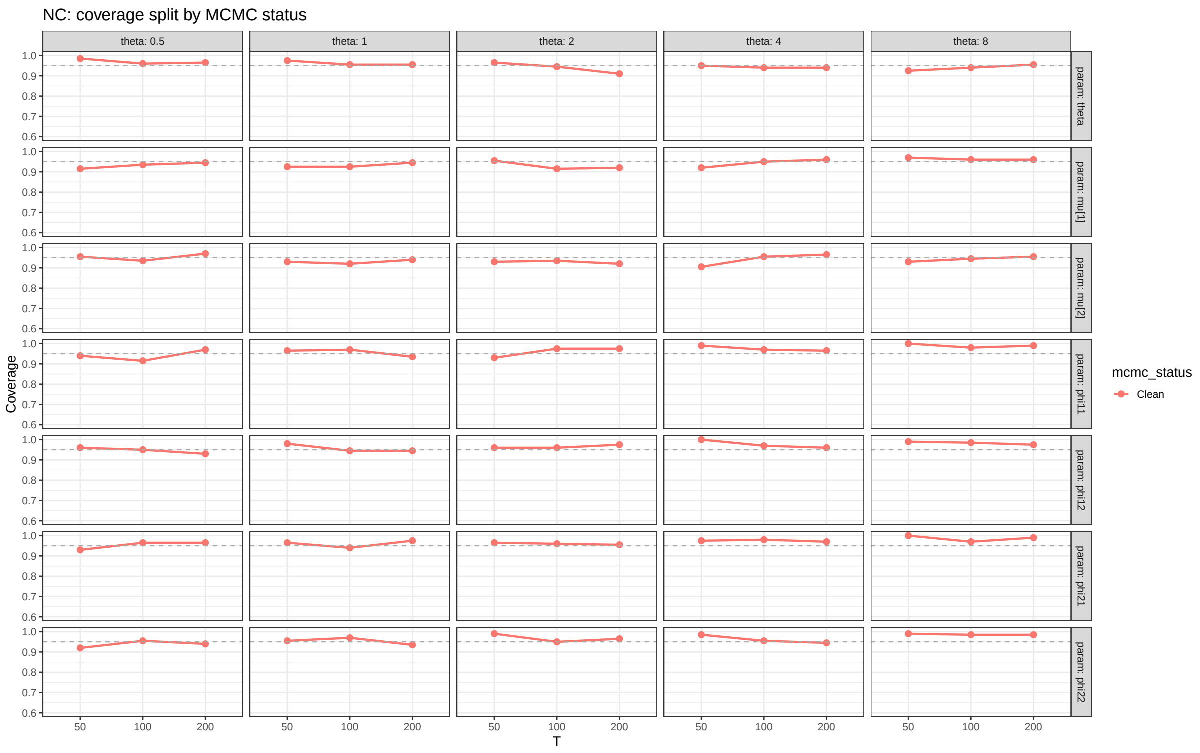
```r
sigma_cond <- cond |>
  filter(param %in% c("sigma[1]", "sigma[2]"))

if (nrow(sigma_cond) > 0) {

  # Guard against edge cases where facetting variables are missing/NA
  sigma_plot_df <- sigma_cond |>
    filter(!is.na(T), !is.na(VARset), !is.na(theta)) |>
    droplevels()

  if (nrow(sigma_plot_df) > 0) {

    base_p <- ggplot(sigma_plot_df, aes(x = T, y = mean_bias, color = Model, group = Model))
      geom_line(position = position_dodge(0.3), linewidth = 1) +
      geom_point(position = position_dodge(0.3), size = 2.5) +
      geom_hline(yintercept = 0, linetype = "dashed", color = "darkgrey") +
      theme_standard +
      scale_color_manual(values = model_colors) +
      labs(title = "Bias for sigma (truth = 1)", y = "Mean Bias (Estimate - 1)", x = "T")

    p_sigma <- tryCatch(
      base_p + facet_grid(param ~ VARset + theta, labeller = label_both),
      error = function(e) {
        tryCatch(
          base_p + facet_wrap(~ param, ncol = 1),
          error = function(e2) base_p
        )
      }
    )

    p_sigma
  }
}
```



8. MCMC Diagnostics: Status Split

```
aggregate_by_status <- function(df) {
  df |>
    filter(mcmc_status != "Failed/Error") |>
    group_by(condition_id, Model, param, mcmc_status, T, theta, VARset) |>
    summarise(
      N_valid      = n(),
      mean_rel_bias= mean(rel_bias, na.rm = TRUE),
      coverage_95  = mean(cover95, na.rm = TRUE),
      mean_post_sd = mean(post_sd, na.rm = TRUE),
      emp_sd       = sd(post_mean, na.rm = TRUE),
      mean_bias    = mean(bias, na.rm = TRUE),
      .groups = "drop"
    ) |>
    mutate(
      emp_sd = ifelse(is.na(emp_sd), 0, emp_sd),
      sd_bias = mean_post_sd - emp_sd,
      RMSE    = sqrt(mean_bias^2 + emp_sd^2)
    )
}
cond_status <- aggregate_by_status(rep_df)
```

```
status_nc <- cond_status |>
  filter(Model == "NC (Clayton copula)", param %in% c(core_params, "theta"))

if (nrow(status_nc) > 0) {
  ggplot(status_nc, aes(x = T, y = coverage_95, color = mcmc_status, group = mcmc_status)) +
    geom_line(position = position_dodge(0.3), linewidth = 1) +
    geom_point(position = position_dodge(0.3), size = 2.5) +
    geom_hline(yintercept = 0.95, linetype = "dashed", color = "darkgrey") +
    facet_grid(param ~ theta, labeller = label_both) +
    theme_standard +
    labs(title = "NC: coverage split by MCMC status", y = "Coverage", x = "T") +
    coord_cartesian(ylim = c(0.6, 1.0))
}
```



9. Export Tables

```
# Aggregated condition-level summary
export_cond <- cond |>
  dplyr::select(condition_id, Model, param, T, theta, VARset,
                N_valid, N_truth_avail,
                mean_rel_bias, coverage_95, RMSE,
                mean_post_sd, emp_sd, sd_bias,
                mean_n_div, prop_div, mean_rhat)

write_csv(export_cond, file.path(EXPORT_DIR, "analysis_summary_aggregated.csv"))

# Status-split summary
export_status <- cond_status |>
  dplyr::select(condition_id, Model, param, mcmc_status, T, theta, VARset,
                N_valid, mean_rel_bias, coverage_95, RMSE,
                mean_post_sd, emp_sd, sd_bias)

write_csv(export_status, file.path(EXPORT_DIR, "analysis_summary_status_split.csv"))
```

```r
# MCMC health counts
mcmc_health_export <- rep_df |>
  distinct(condition_id, rep_id, Model, mcmc_status, T, theta) |>
  count(Model, theta, T, mcmc_status, name = "Count") |>
  tidyr::complete(Model, theta, T, mcmc_status, fill = list(Count = 0)) |>
  arrange(Model, theta, T)

write_csv(mcmc_health_export, file.path(EXPORT_DIR, "analysis_mcmc_health_counts.csv"))
```

‘‘‘