Study 2: Analysis (Exponential DGP)

Table of contents


```
# load necessary libraries
suppressPackageStartupMessages({
  library(dplyr)
  library(tidyr)
  library(readr)
  library(ggplot2)
  library(stringr)
```

```
  library(knitr)
  library(RColorBrewer)
  library(ggh4x) # required for nested facets if needed
  # Load patchwork for arranging plots
  if (!requireNamespace("patchwork", quietly = TRUE)) {
      message("Package 'patchwork' is recommended for optimal plot layout.")
  } else {
      library(patchwork)
  }
})

# define paths
# BASE_DIR <- getwd() # needed for interactive running
DATA_DIR <- file.path("data")
RES_DIR <- file.path("results")
EXPORT_DIR <- file.path(RES_DIR, "exported_tables_s2")
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"),
  # Study 2 uses sim_conditions.rds (as defined in the Study 2 run_pipeline.R)
  design = file.path(DATA_DIR, "sim_conditions.rds")
)

if (!all(file.exists(unlist(files)))) {
    message("Missing required input files. Please ensure the Study 2 pipeline and analysis_s
    # Stop knitting gracefully if data is missing
    if (knitr::is_html_output() || knitr::is_latex_output()) {
        knitr::knit_exit()
    }
}
```

0. tl;dr (Anticipated Results)

0.1 Computational vs. Inferential Performance

- **NG (Normal-Gaussian)** and **SG (Skew-Normal-Gaussian)** are expected to be computationally robust (mostly "Clean" runs), but they are statistically misspecified for the Exponential DGP.

- **EG (Exponential-Gaussian)** is correctly specified but computationally demanding. We anticipate a high rate of "Problematic" MCMC runs (divergences) due to the boundary constraints inherent in the Exponential distribution, especially at lower T.
- **Divergences in EG:** A key goal is to assess if these computational issues invalidate statistical inference, or if the model still provides reliable estimates despite sampling difficulties.

0.2 Model Performance under Exponential DGP

- **DGP Characteristics:** The Exponential distribution is highly skewed (Skewness=2) and heavy-tailed (Excess Kurtosis=6).
- **Dynamics (Φ): EG** is expected to provide unbiased estimates. **NG** and **SG** will likely show attenuation bias (bias towards zero) due to their inability to capture the true shape of the innovations.
- **Dependence (ρ): EG** should recover ρ accurately. **NG** and **SG** are expected to severely underestimate ρ. This is due to the failure of the Probability Integral Transform (PIT) when the marginals are misspecified. The true skewness (2) significantly exceeds the maximum skewness the SG model can handle (≈ 0.995).
- **Intercepts (μ): EG** and **NG** are expected to be relatively unbiased for μ. **SG** might exhibit bias in μ as it attempts to compensate for the shape mismatch (similar to the `extremeCHI` findings in Study 1).

0.3 Mechanism

- The primary driver of failure for NG and SG is **PIT distortion due to marginal misfit**.
- The primary challenge for EG is **computational**. The Exponential distribution has a sharp boundary; sampling near this boundary is difficult for HMC, leading to divergences.

1. Introduction

This simulation study (Study 2) compares the performance of three Bayesian Vector Autoregressive (VAR(1)) models when the Data Generating Process (DGP) exhibits characteristics of Exponential tails. The models evaluated are the Normal-Gaussian (NG), the Skew-Normal-Gaussian (SG), and the correctly specified Exponential-Gaussian (EG) model.

1.1. Data Generating Process (DGP)

The DGP is a bivariate VAR(1) model:

$$Y_t = \mu + \Phi Y_{t-1} + \epsilon_t$$

Where μ is set to 0. The innovations ϵ_t are standardized (Mean=0, Variance=1).

The joint distribution of ϵ_t is modeled using a **Gaussian Copula** parameterized by ρ. The marginal distributions $f_i(\epsilon_{i,t})$ are **Standardized Exponential**.

> **Note**
>
> Standardized Exponential innovations are generated from an Exponential distribution (Rate=1) and then transformed using the Z-score formula (Mean=1, SD=1). The direction of the skewness (left or right) is controlled by mirroring the standardized values.

1.2. Simulation Design

The study employs a full factorial design crossing four factors, resulting in 36 unique conditions.

Table 1: Summary of the Simulation Design Factors (Study 2).

Factor	Levels
DGP Level	Standardized Exponential
Time Series Length (T)	50, 100, 200
Copula Correlation (ρ)	0.30, 0.50
VAR Parameters (Φ)	**Set A** (Symmetric): $\begin{pmatrix} 0.40 & 0.10 \\ 0.10 & 0.40 \end{pmatrix}$
	Set B (Asymmetric): $\begin{pmatrix} 0.55 & 0.10 \\ 0.10 & 0.25 \end{pmatrix}$
Skewness Direction	`++` (Both Right Skew), `--` (Both Left Skew), `+-` (Mixed)

2. Data Loading and Preparation

```r
# load the design grid
design_raw <- readRDS(files$design)

# Harmonize column names: Study 1 used skew_level, Study 2 uses dgp_level.
# We standardize on 'skew_level' internally for consistency with Study 1 plotting logic/nami
if ("dgp_level" %in% names(design_raw) && !"skew_level" %in% names(design_raw)) {
    names(design_raw)[names(design_raw) == "dgp_level"] <- "skew_level"
}

design <- design_raw |>
  select(condition_id, skew_level, direction, T, rho, VARset)


# load condition-level summary (aggregated metrics)
cond_raw <- read_csv(files$cond, show_col_types = FALSE) |>
  left_join(design, by = "condition_id")

# load replication-level summary (individual runs)
rep_raw <- read_csv(files$rep, show_col_types = FALSE) |>
    filter(!is.na(param)) |> # Filter out completely failed fits
    left_join(design, by = "condition_id")


# define parameter order and groups (Updated for Study 2)
param_levels <- c(
                  # EG specific
                  "sigma_exp[1]", "sigma_exp[2]",
                  # SG specific
                  "omega[1]","omega[2]","alpha[1]","alpha[2]",
                  # NG specific
                  "sigma[1]","sigma[2]",
                  # Core parameters
                  "mu[1]","mu[2]",
                  "phi11","phi12","phi21","phi22","rho")

# apply factor levels and clearer labels
prep_data <- function(df) {
  # Filter parameters that actually exist in the data to avoid empty levels
  existing_params <- intersect(param_levels, unique(df$param))

  df |>
    mutate(param = factor(param, levels = existing_params),
```

```
            T = factor(T),
            # Study 2 only has one level, but we keep it as a factor
            skew_level = factor(skew_level),
            rho_val = rho, # keep numeric rho
            VARset_val = VARset, # keep character VARset
            rho = factor(rho, labels = sort(unique(df$rho))),
            VARset = factor(VARset, labels = sort(unique(df$VARset))),
            # Update Model labels for Study 2
            Model = case_when(
                model == "SG" ~ "Skew-Normal (SG)",
                model == "NG" ~ "Normal (NG)",
                model == "EG" ~ "Exponential (EG)"
            ),
            Model = factor(Model, levels = c("Normal (NG)", "Skew-Normal (SG)", "Exponential
}

cond <- prep_data(cond_raw)
rep_df <- prep_data(rep_raw)


cond <- cond |>
  mutate(
    # use coalesce to handle NA emp_sd if N_valid < 2
    RMSE = sqrt(mean_bias^2 + coalesce(emp_sd^2, 0))
  )


# separate core parameters (VAR dynamics, intercepts, correlation)
core_params <- c("mu[1]","mu[2]", "phi11","phi12","phi21","phi22","rho")
```

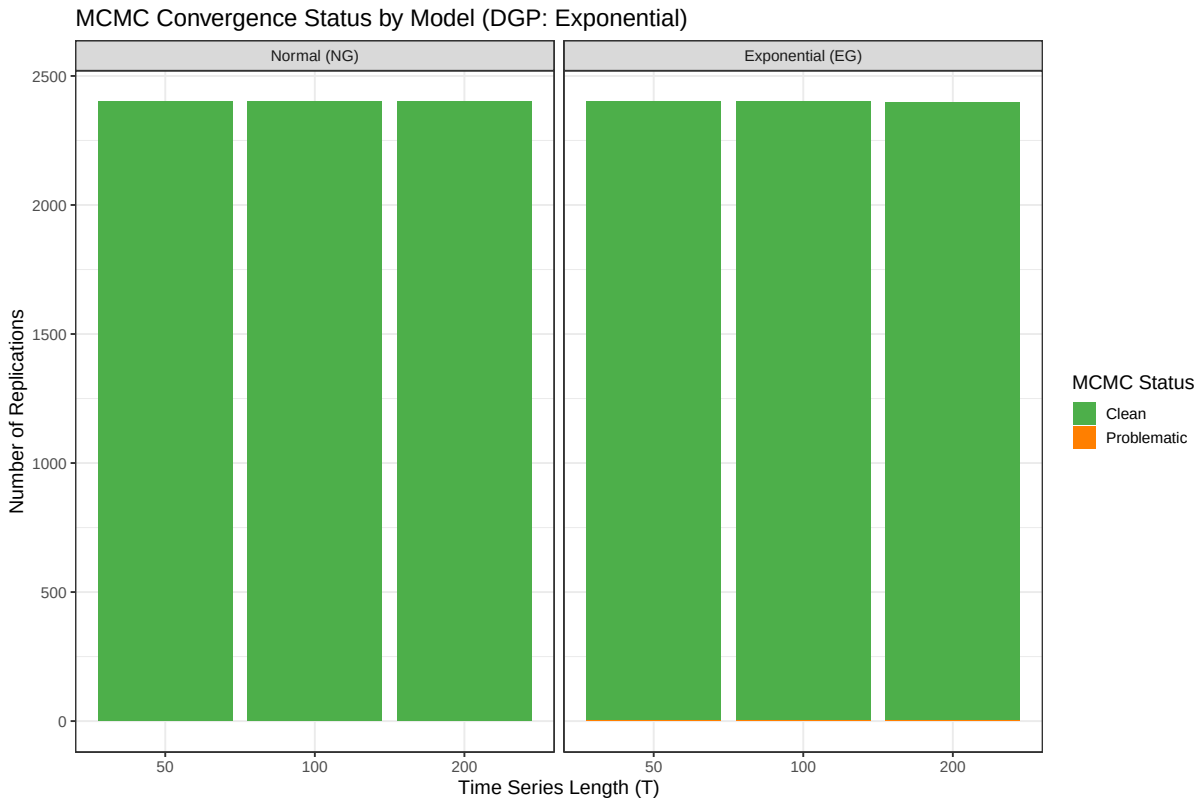
2.1. MCMC Classification and Overview

We classify runs based on MCMC diagnostics (R-hat and divergent transitions `n_div`) and summarize the computational performance.

```
RHAT_THRESHOLD <- 1.01

rep_df <- rep_df |>
  mutate(
    # Ensure n_div is treated as 0 if NA (e.g., for failed runs) - use clean version
    n_div_clean = ifelse(is.na(n_div), 0, n_div),
```

```
    mcmc_status = case_when(
      is.na(max_rhat) | status != "ok" ~ "Failed/Error",
      # classification based on Rhat and divergent transitions (n_div)
      max_rhat > RHAT_THRESHOLD | n_div_clean > 0 ~ "Problematic",
      TRUE ~ "Clean"
    ),
    mcmc_status = factor(mcmc_status, levels = c("Clean", "Problematic", "Failed/Error"))
  )

# MCMC Overview Plot (Status Counts)
# Since skew_level is constant ("Exponential"), we facet by Model and T.
mcmc_summary <- rep_df |>
  distinct(condition_id, rep_id, Model, mcmc_status, T, skew_level) |>
  group_by(Model, T, mcmc_status) |>
  summarise(Count = n(), .groups = "drop")
```

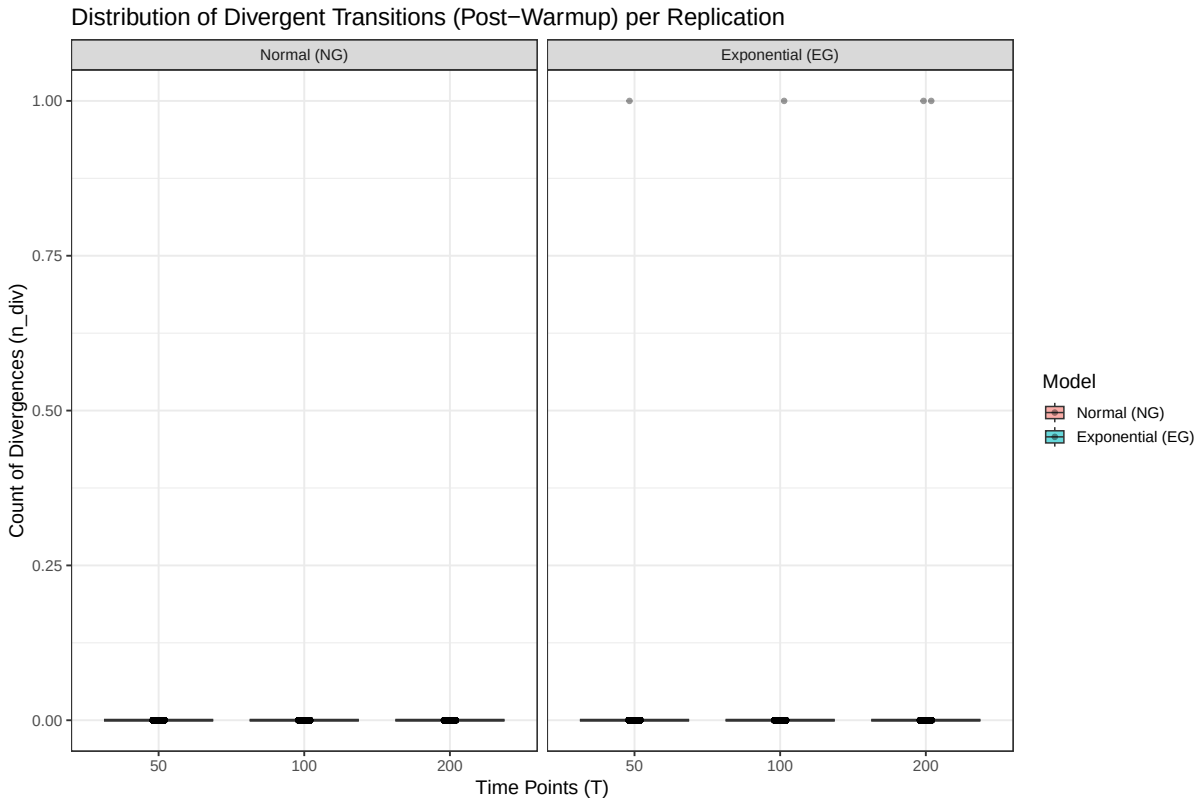
```
ggplot(mcmc_summary, aes(x = T, y = Count, fill = mcmc_status)) +
  geom_bar(stat = "identity", position = "stack") +
  facet_wrap(~ Model) +
  labs(x = "Time Series Length (T)", y = "Number of Replications", fill = "MCMC Status",
       title = "MCMC Convergence Status by Model (DGP: Exponential)") +
  theme_bw(base_size = 14) +
  scale_fill_manual(values = c("Clean" = "#4daf4a", "Problematic" = "#ff7f00", "Failed/Error"
```



Interpretation: MCMC Status Counts (Anticipated) We expect the NG model to be computationally stable ("Clean"). The EG model is anticipated to show a significant number of "Problematic" runs, particularly at T=50, due to the difficulty HMC encounters when sampling near the sharp boundaries imposed by the Exponential distribution.

```
# MCMC Overview Plot (Distribution of Divergent Transitions)
div_dist_data <- rep_df |>
    # ensure we only look at the diagnostics once per replication run
    filter(param == "rho") |>
    distinct(condition_id, rep_id, Model, T, n_div_clean, mcmc_status) |>
    # exclude runs that failed entirely before sampling (if any)
    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)) +
  # Jitter points to show individual runs
  geom_point(size = 1.5, alpha = 0.4, position = position_jitterdodge(jitter.width = 0.2, do
  facet_wrap(~ Model) +
  theme_bw(base_size = 14) +
  # Consider log1p scale if divergences are highly skewed
```

```
  # scale_y_continuous(trans = 'log1p', breaks = c(0, 10, 100, 500)) +
  labs(title = "Distribution of Divergent Transitions (Post-Warmup) per Replication",
       y = "Count of Divergences (n_div)",
       x = "Time Points (T)")
```



Interpretation: Divergent Transitions (Anticipated) The boxplots should highlight the computational challenges of the EG model, showing a wider distribution of divergences compared to NG and SG.

3. Helpers

To systematically analyze the results, we define helper functions for the standard plots.

```
# standardized visualization settings
theme_standard <- theme_bw(base_size = 14)
# Increase dodge width to accommodate three models
dodge_width <- 0.5

# Define a color palette for the three models
```

```r
model_colors <- c("Normal (NG)" = "#377eb8", "Skew-Normal (SG)" = "#ff7f00", "Exponential (E


# helper function for plotting metrics across conditions
plot_metric <- function(data, metric_col, ylab, title, use_free_y = FALSE, ylims = NULL) {

  # filter out potential NAs (e.g. if N_truth_avail = 0)
  data_filtered <- data |> filter(!is.na(.data[[metric_col]]))

  if (nrow(data_filtered) == 0) {
      message("Skipping plot '", title, "' due to missing data.")
      return(NULL)
  }

  p <- ggplot(data_filtered, aes(x = T, y = .data[[metric_col]], color = Model, group = Model
    geom_line(position = position_dodge(dodge_width), linewidth = 1) +
    geom_point(position = position_dodge(dodge_width), size = 2.5) +
    # Faceting structure remains the same: param ~ direction + VARset + rho
    facet_grid(param ~ direction + VARset + rho, labeller = label_both, scales = ifelse(use_
    theme_standard +
    # Apply the defined color palette
    scale_color_manual(values = model_colors) +
    labs(title = title, y = ylab, x = "Time Points (T)")

  # add reference lines based on the metric
  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")
  }

  # apply custom Y-axis limits if provided
  if (!is.null(ylims)) {
    p <- p + coord_cartesian(ylim = ylims)
  }

  return(p)
}

# Wrapper to generate the standard suite of plots
# Since there is only one DGP level, we don't need to iterate over skew_level
generate_plots <- function() {
```

```
  data_subset <- cond |>
    filter(param %in% core_params)

  # Set Y-axis limits for coverage (anticipating poor performance for NG/SG)
  cov_ylims <- c(0.5, 1.0)

  list(
    # Metric: Relative Bias
    bias = plot_metric(data_subset, "mean_rel_bias", "Mean Relative Bias",
                       "Relative Bias (DGP: Exponential)", use_free_y = TRUE),
    # Metric: 95% CI Coverage
    coverage = plot_metric(data_subset, "coverage_95", "Empirical Coverage",
                           "95% Coverage (DGP: Exponential)", ylims = cov_ylims),
    # Metric: RMSE (Overall Accuracy)
    rmse = plot_metric(data_subset, "RMSE", "Root Mean Squared Error",
                       "RMSE (DGP: Exponential)", use_free_y = TRUE),
    # Metric: Posterior SD (Uncertainty Estimate)
    post_sd = plot_metric(data_subset, "mean_post_sd", "Mean Posterior SD",
                          "Mean Posterior SD (DGP: Exponential)", use_free_y = TRUE),
    # Metric: SD-Bias (Calibration of Uncertainty)
    sdbias = plot_metric(data_subset, "sd_bias", "SD-Bias",
                         "SD-Bias (DGP: Exponential)", use_free_y = TRUE)
  )
}
```

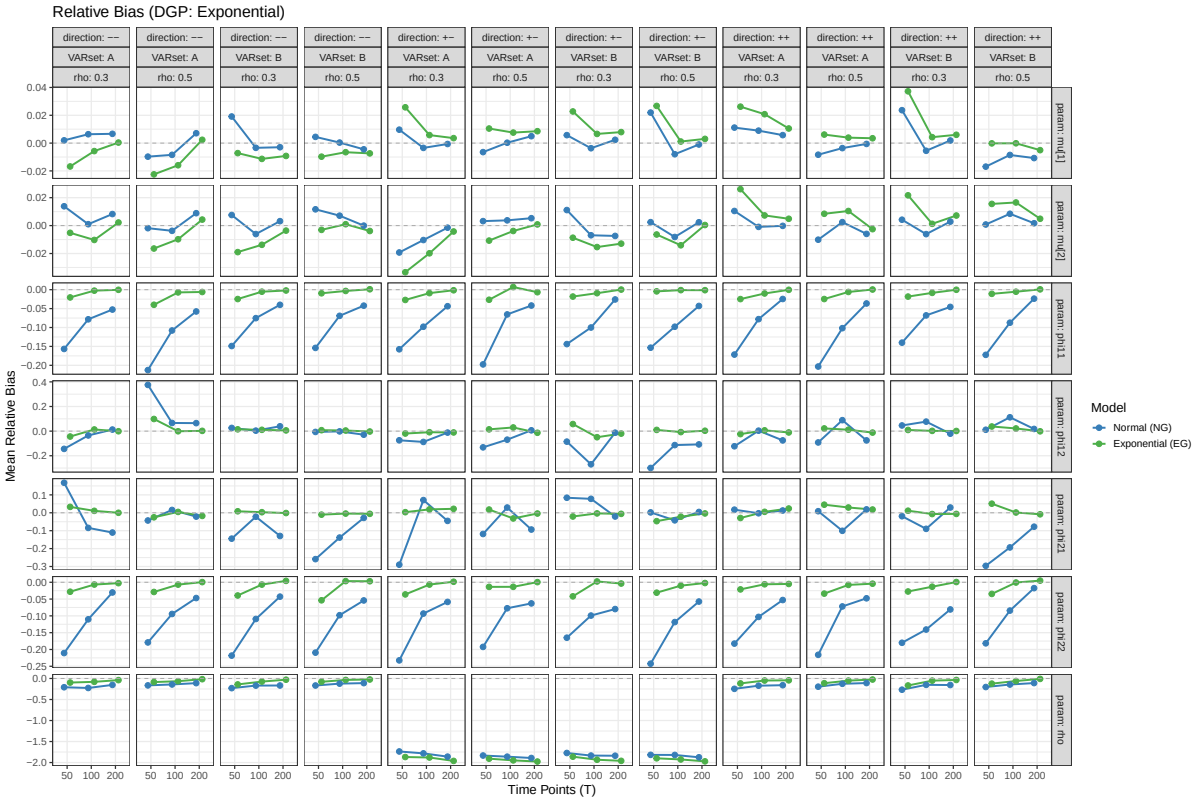
4. Analysis: Exponential DGP

We analyze the performance of the NG (misspecified), SG (misspecified), and EG (correctly specified) models under the standardized Exponential DGP.

```
plots_exp <- generate_plots()
```

4.1. Relative Bias

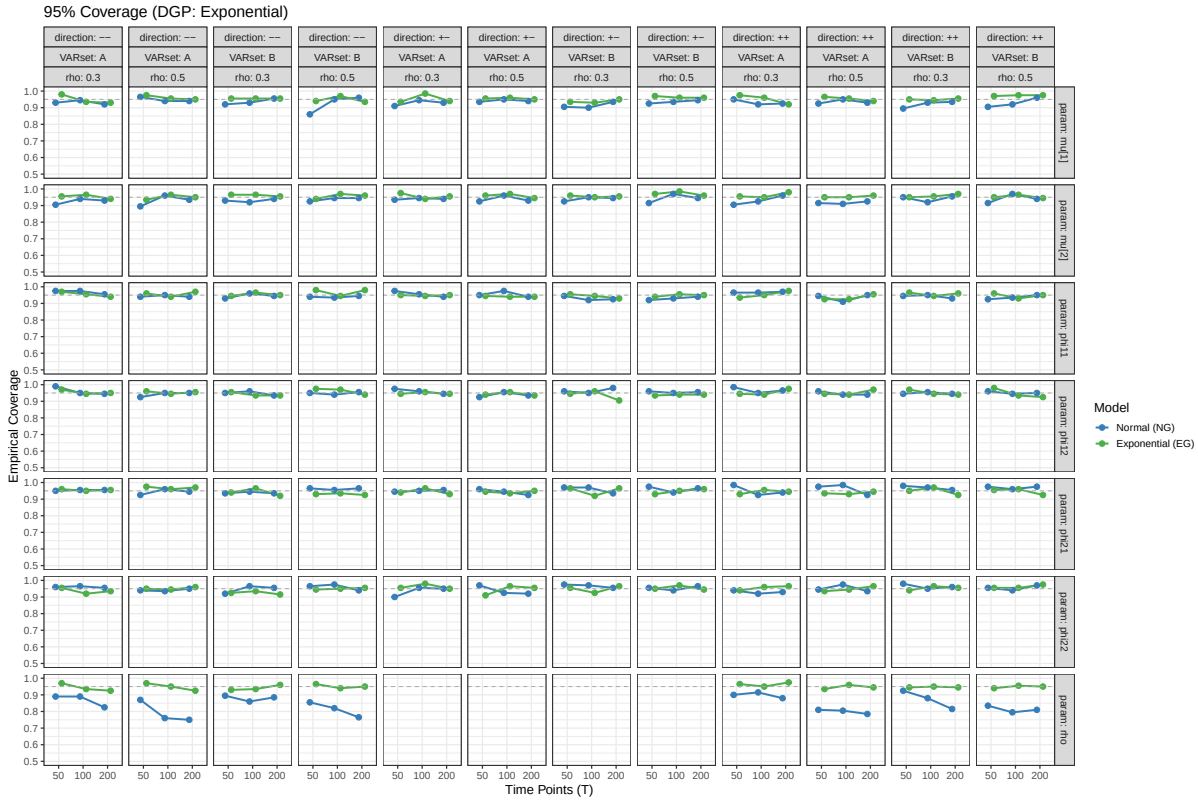
```
print(plots_exp$bias)
```



Interpretation: Relative Bias (Anticipated) - EG Model: Expected to be centered around zero bias for all parameters (Φ, ρ, μ). - **NG and SG Models:** Expected to show severe attenuation bias for the dependence parameter (ρ). The Exponential DGP (Skewness=2) is highly asymmetric, exceeding the capacity of SG (max skewness ≈ 0.995). The resulting PIT distortion leads to attenuated ρ. Biases in Φ are also expected. - **Intercepts (μ):** Similar to Study 1 (extremeCHI), the SG model might show bias in μ due to the Centered Parameterization compensating for the shape mismatch.

4.2. 95% Coverage

```
print(plots_exp$coverage)
```

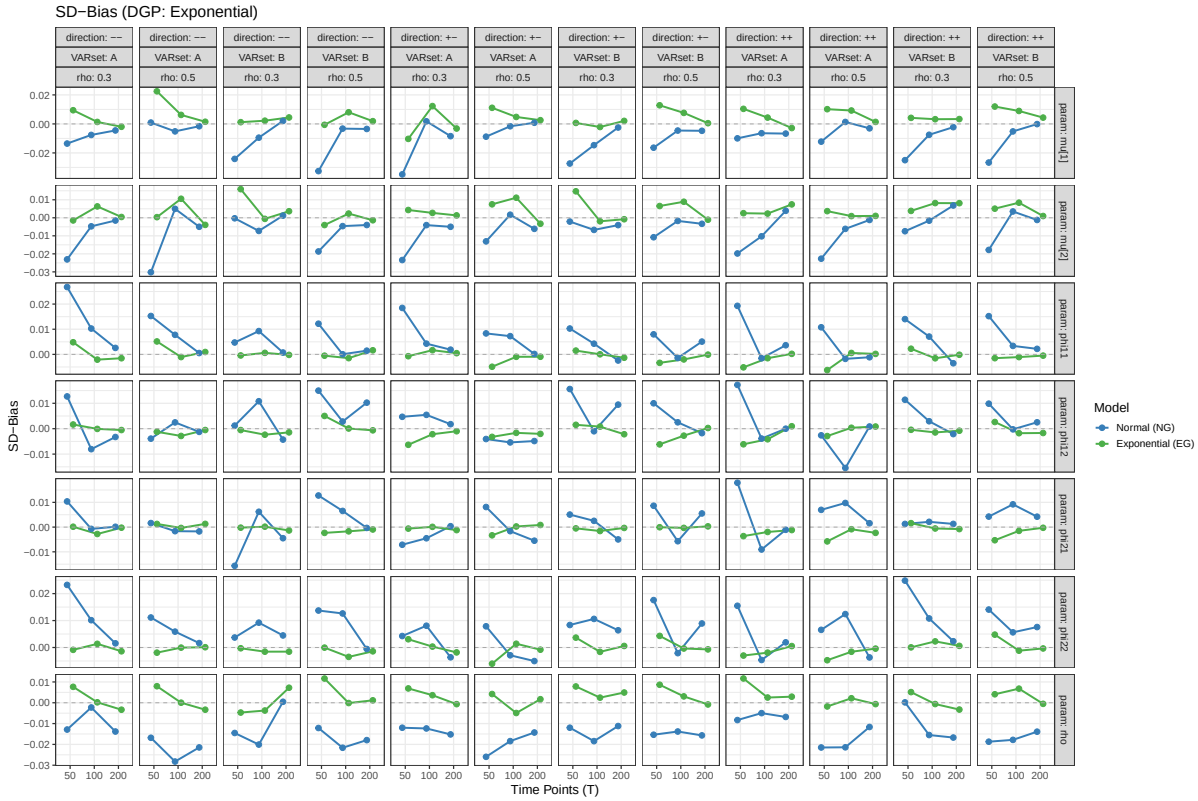


Interpretation: Coverage (Anticipated) - EG Model: Expected to achieve near-nominal (95%) coverage. - **NG and SG Models:** Expected to show severe under-coverage, especially for ρ, due to the significant bias.

4.3. SD-Bias

SD-Bias (Mean Posterior SD - Empirical SD) indicates the calibration of the model's uncertainty. Negative SD-Bias means the model is overconfident.

```
# Note: RMSE and Mean Posterior SD plots are omitted for brevity.
print(plots_exp$sdbias)
```



Interpretation: SD-Bias (Anticipated) - EG Model: Expected to be well-calibrated (SD-Bias near zero). - **NG and SG Models:** Expected to show negative SD-Bias (overconfidence).

5. Synthesis: Marginal Parameters

We examine the parameters governing the marginal distributions to understand how each model adapts to the Exponential DGP. The data is standardized (Variance=1).

5.1. EG Model: Scale Recovery (σ_{exp})

The correctly specified EG model should recover the true scale parameter $\sigma_{\text{exp}} = 1$.

```
sigma_exp_data <- cond |>
  filter(param %in% c("sigma_exp[1]", "sigma_exp[2]"), Model == "Exponential (EG)")

# We plot the absolute bias (mean_bias) here as the truth is 1.
if (nrow(sigma_exp_data) > 0) {
```

```
    ggplot(sigma_exp_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(param ~ direction + VARset + rho, labeller = label_both) +
      theme_standard +
      scale_color_manual(values = model_colors) +
      labs(title = "EG Model: Bias for Sigma_Exp (Truth=1)",
           y = "Mean Bias (Estimate - 1)",
           x = "Time Points (T)")
} else {
    message("No data available for EG marginal parameters.")
}
```

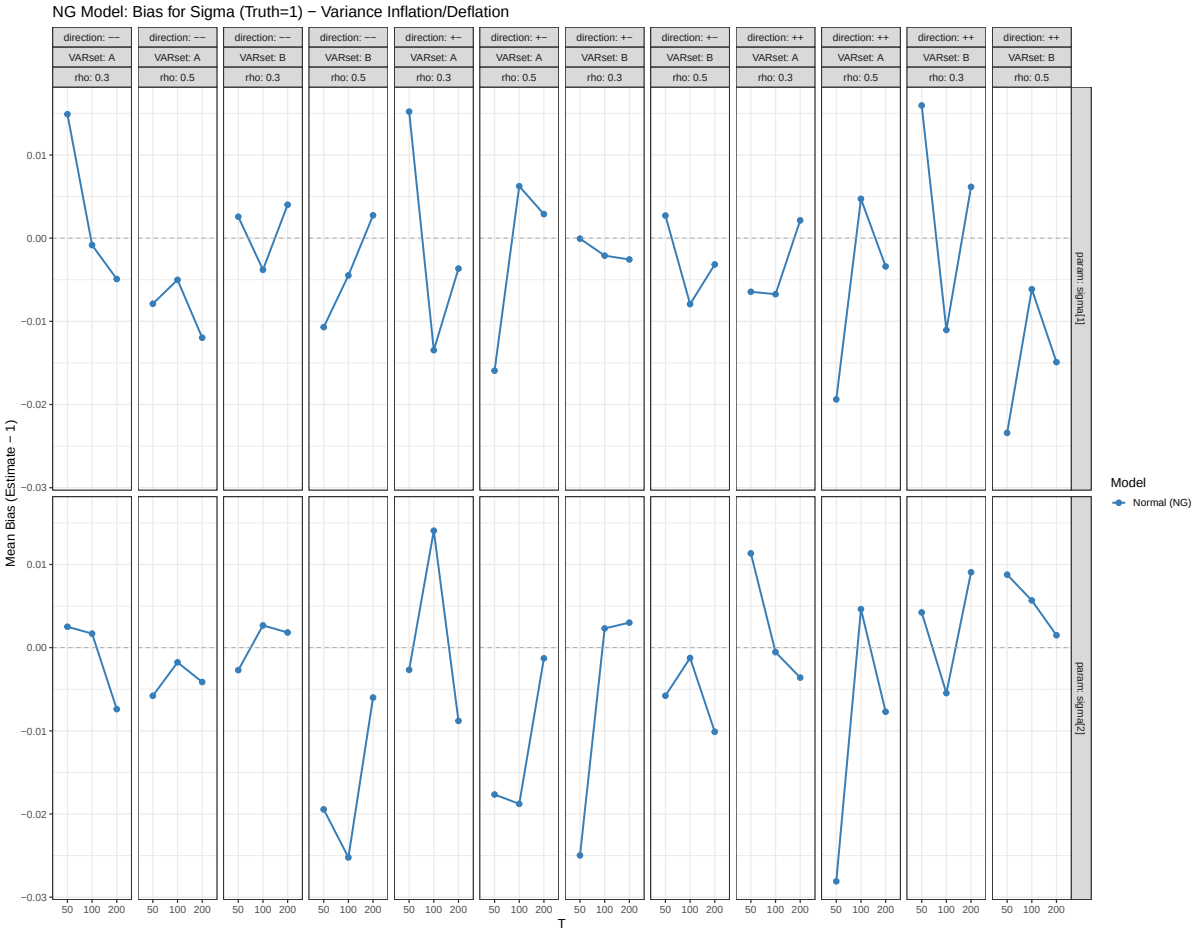
Interpretation (Anticipated): The EG model is expected to accurately recover $\sigma_{\text{exp}} = 1$ with minimal bias.

5.2. Misspecification Handling: NG (σ) and SG (α)

We investigate how the misspecified models attempt to accommodate the Exponential data.

```
# 1. Analyze NG Sigma (Variance Inflation/Deflation)
# The true standardized variance is 1. We plot the bias.
sigma_data <- cond |>
  filter(param %in% c("sigma[1]", "sigma[2]"), Model == "Normal (NG)")

# 2. Analyze SG Alpha (Shape parameter estimates)
# The true DGP is Exponential, so there is no true 'alpha'. We must calculate the average pos
alpha_estimates <- rep_df |>
    filter(param %in% c("alpha[1]", "alpha[2]"), Model == "Skew-Normal (SG)") |>
    # Aggregate across replications
    group_by(T, param, direction, VARset, rho, Model) |>
    summarise(AvgPostMean = mean(post_mean, na.rm = TRUE), .groups = 'drop')


# Plot for NG Sigma Bias
p_sigma <- NULL
if (nrow(sigma_data) > 0) {
    p_sigma <- ggplot(sigma_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(param ~ direction + VARset + rho, labeller = label_both) +
      theme_standard +
      scale_color_manual(values = model_colors) +
      labs(title = "NG Model: Bias for Sigma (Truth=1) - Variance Inflation/Deflation",
           y = "Mean Bias (Estimate - 1)", x = "T")
}

# Plot for SG Alpha Estimates
p_alpha <- NULL
if (nrow(alpha_estimates) > 0) {
    p_alpha <- ggplot(alpha_estimates, aes(x = T, y = AvgPostMean, color = Model, group = Mo
        geom_line(position = position_dodge(0.3), linewidth = 1) +
        geom_point(position = position_dodge(0.3), size = 2.5) +
        # Use nested facets for clarity
        ggh4x::facet_nested(param ~ direction + VARset + rho, labeller = label_both, scales =
        theme_standard +
        scale_color_manual(values = model_colors) +
        labs(title = "SG Model: Estimated Alpha (Shape Parameter)",
             y = "Average Posterior Mean", x = "T")
}


# Arrange plots using patchwork if loaded
if (!is.null(p_sigma) && !is.null(p_alpha) && requireNamespace("patchwork", quietly = TRUE))
    p_sigma / p_alpha
} else {
    if (!is.null(p_sigma)) print(p_sigma)
    if (!is.null(p_alpha)) print(p_alpha)
}
```



Interpretation (Anticipated): - **NG Sigma:** The NG model is expected to show bias in σ as it misinterprets the Exponential shape as variance. - **SG Alpha:** The SG model is expected to push the α parameters towards their limits in the direction corresponding to the true skewness, attempting to utilize its maximum flexibility, even though it cannot fully match the Exponential shape.

6. Impact of MCMC Diagnostics

We investigate if the statistical performance differs between "Clean" and "Problematic" runs, particularly for the EG model, which is prone to computational issues.

```
# Helper function to re-aggregate metrics, filtering by MCMC status
# It is crucial to recalculate Empirical SD, SD-Bias, and RMSE within the subgroups.
aggregate_by_status <- function(df) {
  df |>
    # Filter out total failures
```

```
    filter(mcmc_status != "Failed/Error") |>
    # Grouping must include all identifiers and design factors
    group_by(condition_id, Model, param, mcmc_status, T, skew_level, direction, VARset, rho,
    summarise(
      N_valid = n(),
      mean_rel_bias = mean(rel_bias, na.rm = TRUE),
      coverage_95 = mean(cover95, na.rm = TRUE),
      # Recalculate components for SD-Bias and RMSE within the status group
      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(
        # Handle cases where N_valid=1, leading to NA emp_sd
        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)
```

6.1. Coverage Split by MCMC Status (EG Model)

We visualize the coverage for the EG model, comparing Clean vs. Problematic runs across the detailed conditions.

```
status_comparison_data <- cond_status |>
  filter(Model == "Exponential (EG)",
         param %in% core_params)

if (nrow(status_comparison_data) > 0) {
    # Check if there is data for both Clean and Problematic to compare
    if (length(unique(status_comparison_data$mcmc_status)) > 0) {
        ggplot(status_comparison_data,
               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 by parameter and condition factors
```

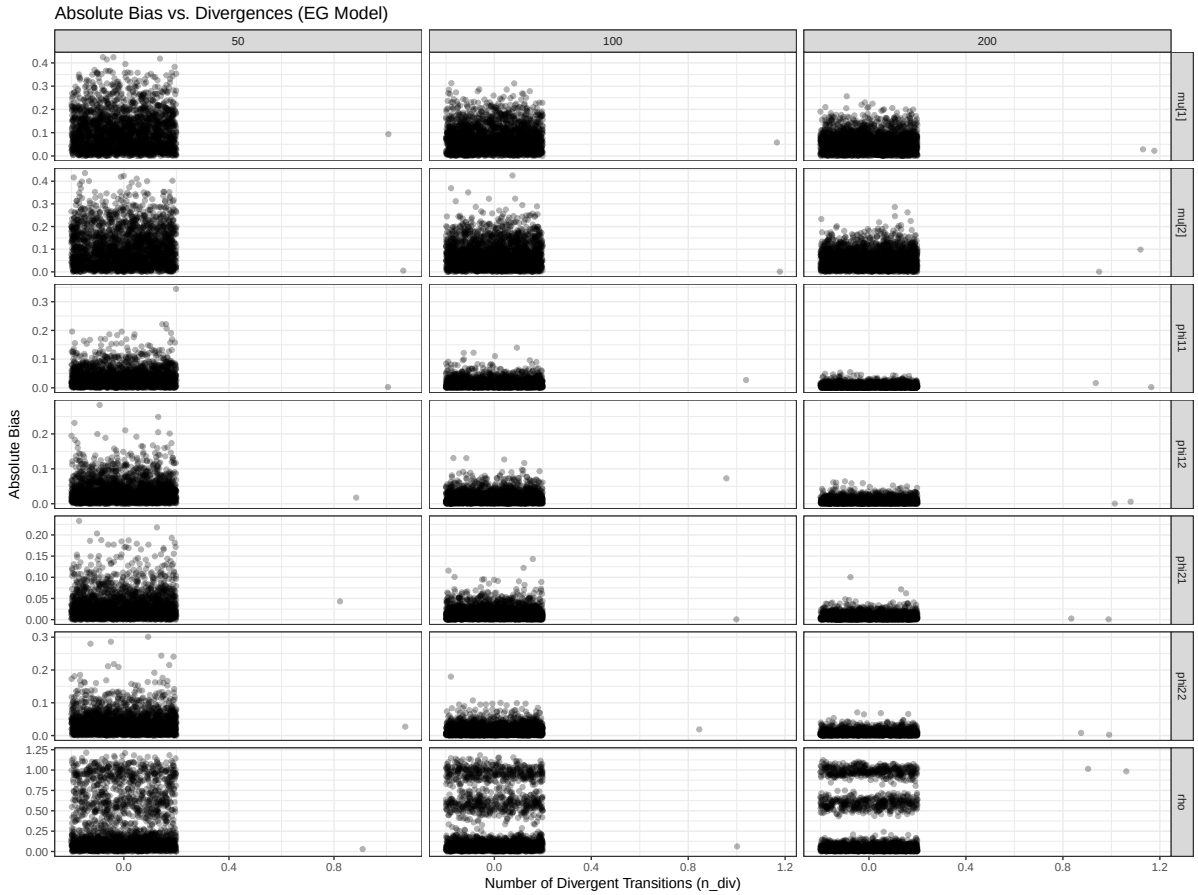
```
          facet_grid(param ~ direction + VARset + rho, labeller = label_both) +
          theme_standard +
          labs(title = "EG Model Coverage: Clean vs. Problematic Runs",
               y = "Empirical Coverage",
               x = "Time Points (T)",
               color = "MCMC Status") +
          coord_cartesian(ylim = c(0.8, 1.0))
    } else {
        message("Not enough MCMC status diversity for EG model comparison (e.g., all runs wer
    }
} else {
    message("No data available for EG model status comparison.")
}
```

Interpretation: Impact of MCMC Status (Anticipated) If the coverage remains similar between "Clean" and "Problematic" runs and stays near the nominal 95% rate, it suggests that the statistical inference is robust to the computational challenges (divergences) encountered by the EG model.

6.2. Relationship between Bias and Divergences (EG Model)

We examine if runs with more divergences exhibit higher bias at the replication level for the EG model.

```
div_bias_data <- rep_df |>
  filter(Model == "Exponential (EG)",
         param %in% core_params,
         mcmc_status != "Failed/Error")

if (nrow(div_bias_data) > 0) {
    # Use absolute bias for comparison
    ggplot(div_bias_data, aes(x = n_div_clean, y = abs(bias))) +
      geom_point(alpha = 0.3, position = position_jitter(width = 0.2), size = 2) +
      # Use a generalized additive model (gam) for smoothing
      geom_smooth(method = "gam", color = "red", linewidth = 1.5) +
      facet_grid(param ~ T, scales = "free") +
      theme_standard +
      labs(title = "Absolute Bias vs. Divergences (EG Model)",
           x = "Number of Divergent Transitions (n_div)",
           y = "Absolute Bias")
} else {
    message("No data available for EG model bias vs. divergences analysis.")
}
```



Interpretation: Bias vs. Divergences (Anticipated) We anticipate a weak or non-existent correlation, suggesting that MCMC diagnostics are indicators of computational difficulty but not necessarily predictors of statistical accuracy in this context.

7. Exporting Key Tables

We save the processed and aggregated dataframes as CSV files for external analysis.

```
# 1. Main Condition-Level Summary (Aggregated across all successful runs)
export_cond <- cond |>
  # Select relevant columns and restore original factor values. Use dgp_level for export cla
  select(condition_id, Model, param, dgp_level = skew_level, direction, T,
         rho = rho_val, VARset = VARset_val,
         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)
```

```r
write_csv(export_cond, file.path(EXPORT_DIR, "analysis_summary_aggregated_S2.csv"))


# 2. Status-Split Summary (Aggregated within Clean/Problematic groups)
export_status <- cond_status |>
    # Select relevant columns
  select(condition_id, Model, param, mcmc_status,
         dgp_level = skew_level, direction, T,
         rho = rho_val, VARset = VARset_val,
         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_S2.csv"))

# 3. MCMC Health Summary (Counts)
# Ensure all status levels are included even if count is 0
mcmc_health_export <- mcmc_summary |>
  # Use complete to fill missing combinations
  tidyr::complete(Model, T, mcmc_status, fill = list(Count = 0)) |>
  pivot_wider(names_from = mcmc_status, values_from = Count) |>
  arrange(Model, T)

write_csv(mcmc_health_export, file.path(EXPORT_DIR, "analysis_mcmc_health_counts_S2.csv"))

message("Tables exported to: ", EXPORT_DIR)
```