Monte Carlo Simulation Tools for REDD+ Uncertainty Estimates

2024-12-19

Contents

1. Introduction

When preparing for Monte Carlo simulations, it is good practice first to examine descriptive statistics of the data to characterize the empirical distributions of input variables. This preliminary analysis should include statistical tests of normality, along with visualizations of univariate distributions. Recommended visualizations include histograms, kernel density plots, and Q-Q plots. Together, these tools provide insights into the data's shape, spread, symmetry, skewness, and potential outliers.

In particular, these visualizations help auditors to confirm levels of bias in the dataset. They also allow for quicker evaluations of how the proponent has addressed these biases in their subsequent calculations.

Incorporating these distribution assessments early in the process will likely reduce overall findings from VVB's and registries. This is because distribution analysis is critical to selecting the appropriate functions in SimVoi, ensuring more accurate Monte Carlo estimates. In effect, bias corrections are incorporated, reducing uncertainty in the final results and improving confidence in the jurisdiction's claims of nationwide emissions reductions.

The following guide summarizes the most commonly used discrete and continuous distributions, their statistical properties, and appropriate usage contexts.

Table 1: Continuous data distributions, and example use cases for Monte Carlo simulations.

Table 1: Continuous data distributions, and example use cases for Monte Carlo simulations.

Continuous Distributions	Description
Bernoulli	Binary outcome targets (success/failure). E.g., a single coin flip.
Binomial	# of successes in Bernoulli trials. E.g., heads in 10 flips.
Poisson	# of events in a fixed interval. E.g., arrivals per hour.
Geometric	# of failures until first success. E.g., calls until a sale.
Neg. Binomial	# of failures until r successes (overdispersed Poisson).
Discrete Uniform	All finite outcomes equally likely. E.g., rolling a fair die.
Normal (Gaussian)	Symmetrical "bell curve." E.g., human heights.
Lognormal	Right-skewed; log(variable) ~ Normal. E.g., incomes.
Exponential	Time between Poisson events. E.g., arrival times.

Continuous Uniform	All values in [a,b] equally likely. E.g., random number gen.
Chi-Square	Used in hypothesis tests (e.g., goodness-of-fit).
t-Distribution	Small samples, unknown population SD.
Weibull	Reliability or lifespans.
Gamma	Models skewed data, e.g., wait times.

Table 2: Discrete data distributions, and example use cases for Monte Carlo simulations.

Discrete Distributions	**Descriptions**
Bernoulli	Binary outcome (success/failure). E.g., a single coin flip.
Binomial	# of successes in n Bernoulli trials. E.g., heads in 10 flips.
Poisson	# of events in a fixed interval. E.g., arrivals per hour.
Geometric	# of failures until first success. E.g., calls until a sale.

2. Method

Import data

```r
# Point this to the correct path where your file is located:
workbook = "./data/art/GuyanaARTWorkbookMC-thru2022-April2024_values.xlsx"
CarbonStocks = readxl::read_excel(workbook, "CarbonStocks") |>
    mutate(across(where(is.numeric), ~round(.x, 1)))
CarbonStocks_MC = readxl::read_excel(workbook, "CarbonStocks (MC)") |>
    mutate(across(where(is.numeric), ~round(.x, 1)))

DeforestationEF = readxl::read_excel(workbook, "Deforestation EFs") |>
    mutate(across(where(is.numeric), ~round(.x, 1)))
DeforestationEF_MC = readxl::read_excel(workbook, "Deforestation EFs (MC)") |>
    mutate(across(where(is.numeric), ~round(.x, 1)))

DegradationEF = readxl::read_excel(workbook, "Degradation EFs") |>
    mutate(across(where(is.numeric), ~round(.x, 1)))
DegradationEF_MC = readxl::read_excel(workbook, "Degradation EFs (MC)") |>
    mutate(across(where(is.numeric), ~round(.x, 1)))

ActivityData = readxl::read_excel(workbook, "Activity Data") |>
    mutate(across(where(is.numeric), ~round(.x, 1)))
ActivityData_MC = readxl::read_excel(workbook, "Activity Data (MC)") |>
    mutate(across(where(is.numeric), ~round(.x, 1)))

Emissions = readxl::read_excel(workbook, "CarbonStocks") |>
    mutate(across(where(is.numeric), ~round(.x, 1)))
Emissions_MC = readxl::read_excel(workbook, "CarbonStocks (MC)") |>
    mutate(across(where(is.numeric), ~round(.x, 1)))

# Vislualize
flextable(head(CarbonStocks[, 1:8])) |>
    fontsize(size = 8, part = "all")
```

...1	AG Tree (tC/ha)	BG Tree (tC/ha)	Saplings (tC/ha)	Standing Dead Wood (tC/ha)	Lying Dead Wood (tC/ha)	Sum Carbon pools w/o litter (t C/ha)	Litter (tC/ha)
mean of all plots (calculated)	205.8	48.3	3.7	2.6	8.6	269.0	3.3
std. dev	60.4	14.3	2.0	4.0	8.1	75.2	1.3
minimum	91.6	21.2	0.5	0.0	0.0		1.2
maximum	353.7	83.1	18.8	13.7	42.3		8.7
90% CI	9.2	2.2	0.3	0.6	1.2	11.5	0.2
CI as % of mean	0.0	0.0	0.1	0.2	0.1	0.0	

```r
flextable(head(CarbonStocks_MC[, 1:8])) |>
    fontsize(size = 8, part = "all")
```

...1	AG Tree (tC/ha)	BG Tree (tC/ha)	Saplings (tC/ha)	Standing Dead Wood (tC/ha)	Lying Dead Wood (tC/ha)	Sum Carbon pools w/o litter (t C/ha)	Litter (tC/ha)
tC/ha	181.1	65.0	3.5	7.3	17.1		3.7
tCO2/ha	664.2	238.2	12.8	26.9	62.6		13.7

```r
# flextable(head(DeforestationEF[, 1:8])) |> fontsize(size = 8, part = 'all')
# flextable(head(DeforestationEF_MC[, 1:8])) |> fontsize(size = 8, part =
# 'all') flextable(head(ActivityData[, 1:8])) |> fontsize(size = 8, part =
# 'all') flextable(head(ActivityData_MC[, 1:8])) |> fontsize(size = 8, part =
# 'all') flextable(head(Emissions[, 1:8])) |> fontsize(size = 8, part = 'all')
# flextable(head(Emissions_MC[, 1:8])) |> fontsize(size = 8, part = 'all')
# flextable(head(DegradationEF[, 1:8])) |> fontsize(size = 8, part = 'all')
# flextable(head(DegradationEF_MC[, 1:8])) |> fontsize(size = 8, part = 'all')
# dplyr::glimpse(CarbonStocks)
```

Tidy data

Identify the relevant rows & columns for each pool (mean, sd, min, max, 90% CI) There is always some adjustments needed to table format, layout, labelling or object classes. The most common and robust pacakge for this is `dplyr`, which we apply below.

We'll assume that rows exist in same order in the CarbonStocks_MC sheet, and we will simply run these operations by just changing the dataframe name. One approach is to reshape the dataframe so each row becomes "Statistic" and the columns become "AG Tree", "BG Tree", etc. For example, the code below tries to rename columns before setting row 2 as "mean", #3 as "std. dev", etc. The exact indexing may vary depending on how read_excel() parsed your data. We advize inspecting outputs with the commands `view(CarbonStocks)` or `glimpse(CarbonStocks)` to confirm row/column positions.

Select only the columns that contain your carbon pools (e.g. columns named "AG Tree (tC/ha)", "BG Tree (tC/ha)". We can then rename these columns to match the SimVoi workbook. From here, w derive a small table that transposes the data so each row includes data values (mean, sd, min) and each column represents a variable or the carbon pool. Note, there is a handy function for this using `tidyr::pivot_longer()`[1] or `pivot_wider`. Similarly, you can also use `tidyr::pivot_wider()` to reshape your data into a wider format.

Watch out that some column names are missing or named incorrectly, which was caused by `readxl::read_excel()` on import into R and and needed renaming below. Finally, we pivot back again from long so that "Statistic" becomes a column, and "AG_Tree, BG_Tree…" are columns.

```
CarbonStocks <- CarbonStocks %>%
    select(Statistic = 1, AG_Tree = 2, BG_Tree = 3, Saplings = 4, StandingDeadWood = 5,
        LyingDeadWood = 6, SumCarbonNoLitter = 7, Litter = 8, SumCpoolWLitter = 9,
        SumCO2e = 10, Soil_tC_ha = 11, SumALL_POOLS_CO2eha = 12, SumABGBLiveTree = 13) %>%
    slice(1:9)
# Rename missing column names
CarbonStocks <- CarbonStocks |>
    rename(Statistic = ...1)

# Transpose to long dataframe: flipping rows w/ columns
CarbonStocks_long <- CarbonStocks |>
    tidyr::pivot_longer(cols = -Statistic, names_to = "Pool", values_to = "Value") |>
    mutate(Value = as.numeric(Value))

# Pivot back to wide dataframe & "Statistic" becomes a row:
CarbonStocks_wide <- CarbonStocks_long %>%
    pivot_wider(names_from = Statistic, values_from = Value)
# Inpspect
CarbonStocks_wide
```

Descriptives

The Coefficient of Variation `CV` is a standardized measure of dispersion of a probability distribution or frequency distribution. It's computed as the ratio of the standard deviation to the mean, typically expressed as a percentage, as shown below. Unlike the standard deviation, the `CV` metric is unitless and allows you to compare variability across different datasets or scales.

$$\mathsf{CV} = \frac{\sigma}{\mu} \times 100\%$$

Effectively, in the context of these carbon stocks, a higher CV value indicates greater variability relative to the mean, which should influence how we choose between normal or log-normal distributions for simulation, for example. The CV also informs whether a distribution is appropriate; if the CV is very high, it is likely suggesting the data is more skewed than expected. In the helper function coded below, CV is calculated according to the above formula using

`CV_percent` = 100 * (`std. dev` / `mean of all plots (calculated)`)

```
# Helper function to compute multiple stats at once:
calc_derived_stats <- function(df) {
  df %>%
    mutate(
      CV_percent = 100 * (`std. dev` / `mean of all plots (calculated)`),
      # Approx check:  if 90% CI is ~ ±1.645 * sd, check consistency with workbook sd
      sd_implied_by_90CI = `90% CI` / 1.645,
```

[1] In R, you chave the option of writing the ellipsis symbol `::` between the owner package and its operating function. In effect, this `tidyr::pivot_longer()` code is calling the `pivot_longer()` function from the `tidyr` package. Alternatively, you can also write `pivot_longer()` directly without its parent package preceding and it should work fine too. Rather, coders tend to use it to avoid headaches of package conflicts, or just to help others quickly see what packages are used and working. format.

```r
      # Simplistic check: How many SDs from mean to min, max?
      SDs_below_mean = (`mean of all plots (calculated)` - minimum) / `std. dev`,
      SDs_above_mean = (maximum - `mean of all plots (calculated)`) / `std. dev`
    )
  }

CarbonStocks_stats <- calc_derived_stats(CarbonStocks_wide)
# Inspect new columns (e.g. mean_of_all_plots_calculated)
CarbonStocks_stats

# Helper function to simulate from each row assuming truncnormal
simulate_truncnorm_from_summary <- function(
  mean_val, sd_val, min_val=0, max_val=Inf,
  n_draws=10000) {
  # We use rtruncnorm from the 'truncnorm' package (`truncnorm::rtruncnorm`)
  draws <- truncnorm::rtruncnorm(
    n     = n_draws,
    a     = min_val,
    b     = max_val,
    mean  = mean_val,
    sd    = sd_val
  )
  # Return vector of draws
  return(draws)
}

# Let's do it for AG_Tree
ag_tree_stats <- CarbonStocks_stats %>% filter(Pool == "AG_Tree")

AG_mean <- ag_tree_stats$`mean of all plots (calculated)`
AG_sd   <- ag_tree_stats$`std. dev`
AG_min  <- ag_tree_stats$minimum
AG_max  <- ag_tree_stats$maximum

# We might prefer to do a = 0 if we never allow negative carbon:
AG_draws <- simulate_truncnorm_from_summary(
  mean_val = AG_mean,
  sd_val   = AG_sd,
  min_val  = 0,      # or AG_min if you prefer
  max_val  = Inf,
  n_draws  = 10000
)

# Compare results:
mean(AG_draws)
sd(AG_draws)
min(AG_draws)
max(AG_draws)
quantile(AG_draws, probs = c(0.05, 0.95))

############################################################
# 6. Summaries & plotting
############################################################
```

```r
# Quick histogram of the draws
hist(AG_draws, breaks=40, col="skyblue",
     main="Truncated Normal draws for AG Tree",
     xlab="AG Tree (tC/ha)")

# If you want to do this for each carbon pool in a loop,
# you can add a small function:

simulate_all_pools <- function(df, n_draws=10000) {
  # df is your cs_stats data frame
  # Return a named list of random draws
  out <- list()
  for (i in seq_len(nrow(df))) {
    rowi <- df[i, ]
    pool_name <- rowi$Pool
    mean_val  <- rowi$`mean of all plots (calculated)`
    sd_val    <- rowi$`std. dev`
    # Use zero for min bound; or rowi$minimum if you want to
    # replicate the workbook min
    draws <- rtruncnorm(
      n=n_draws,
      a=0,
      b=Inf,
      mean=mean_val,
      sd=sd_val
    )
    out[[pool_name]] <- draws
  }
  return(out)
}

all_draws <- simulate_all_pools(cs_stats, n_draws=10000)

# 'all_draws' is now a list with an entry for each pool:
names(all_draws)  # e.g. "AG_Tree", "BG_Tree", etc.

# You can then do further calculations, e.g. sum AG_Tree + BG_Tree draws, etc.

############################################################
# 7. Wrap up
############################################################

# This script accomplishes:
# (1) Reading the "CarbonStocks" sheet
# (2) Extracting summary statistics
# (3) Computing descriptive stats and potential checks
# (4) Demonstrating how to do truncated-normal draws
# (5) Summarizing or plotting the results.

# For an official distribution analysis, you would highlight:
# - Coefficients of variation (CV)
# - Approx symmetry (or skew) from min, max, mean, sd
# - The decision to use truncated normal at 0
# - How well the sample stats match the workbook's.
```

Replicating SimVoi

We utilize the replicate function to repeat a simulationfollowing a randomized normally truncated multiple times with `replicate(n=10000`, while determining the size of the sampled subset with `rnorm(n=100`. The first model explores sample size parameters only, replication parameters are tested below this in comparisons.

```
MEAN = CarbonStocks$`AG Tree (tC/ha)`[1]
SD = CarbonStocks$`AG Tree (tC/ha)`[2]

randtruncnormal_sim_10000 <- rnorm(n = 10000, mean = MEAN, sd = SD)
hist(randtruncnormal_sim_10000, freq = F)
AG_Tree_tC_ha = mean(randtruncnormal_sim_10000)
AG_Tree_tCO2_ha = AG_Tree_tC_ha * (44/12)
AG_Tree_tC_ha
AG_Tree_tCO2_ha
# curve(dnorm(x, mean=MEAN, sd=SD), from=0, to=450, add=T, col='red')
```

Compare simulations

```
# 10,000 simulations sampling 10 observations
randtruncnormal_sim_10000_10 = replicate(n = 10000, rnorm(n = 10, mean = MEAN, sd = SD))
hist(apply(X = randtruncnormal_sim_10000_10, MARGIN = 2, FUN = mean))
sd(apply(X = randtruncnormal_sim_10000_10, MARGIN = 2, FUN = mean))
mean(apply(X = randtruncnormal_sim_10000_10, MARGIN = 2, FUN = mean))
(mean(apply(X = randtruncnormal_sim_10000_10, MARGIN = 2, FUN = mean))) * (44/12)

# 10,000 simulations sampling 100 observations
randtruncnormal_sim_10000_100 = replicate(n = 10000, rnorm(n = 100, mean = MEAN,
    sd = SD))
hist(apply(X = randtruncnormal_sim_10000_100, MARGIN = 2, FUN = mean))
sd(apply(X = randtruncnormal_sim_10000_100, MARGIN = 2, FUN = mean))
mean(apply(X = randtruncnormal_sim_10000_100, MARGIN = 2, FUN = mean))
(mean(apply(X = randtruncnormal_sim_10000_100, MARGIN = 2, FUN = mean))) * (44/12)

# 10,000 simulations sampling 1,000 observations
randtruncnormal_sim_10000_1000 = replicate(n = 10000, rnorm(n = 1000, mean = MEAN,
    sd = SD))
hist(apply(X = randtruncnormal_sim_10000_1000, MARGIN = 2, FUN = mean))
sd(apply(X = randtruncnormal_sim_10000_1000, MARGIN = 2, FUN = mean))
mean(apply(X = randtruncnormal_sim_10000_1000, MARGIN = 2, FUN = mean))
(mean(apply(X = randtruncnormal_sim_10000_1000, MARGIN = 2, FUN = mean))) * (44/12)

# 10,000 simulations sampling 10,000 observations
randtruncnormal_sim_10000_10000 = replicate(n = 10000, rnorm(n = 10000, mean = MEAN,
    sd = SD))
hist(apply(X = randtruncnormal_sim_10000_10000, MARGIN = 2, FUN = mean))
sd(apply(X = randtruncnormal_sim_10000_10000, MARGIN = 2, FUN = mean))
mean(apply(X = randtruncnormal_sim_10000_10000, MARGIN = 2, FUN = mean))
(mean(apply(X = randtruncnormal_sim_10000_10000, MARGIN = 2, FUN = mean))) * (44/12)
```

```
devtools::session_info()
```

```
- Session info ---------------------------------------------------------------
 setting  value
 version  R version 4.3.0 (2023-04-21)
 os       macOS 15.3.2
```

```
 system   aarch64, darwin20
 ui       X11
 language (EN)
 collate  en_US.UTF-8
 ctype    en_US.UTF-8
 tz       America/Vancouver
 date     2025-03-16
 pandoc   3.6.1 @ /usr/local/bin/ (via rmarkdown)

- Packages -------------------------------------------------------------------
 package           * version    date (UTC) lib source
 abind               1.4-8      2024-09-12 [1] CRAN (R 4.3.3)
 animation         * 2.7        2021-10-07 [1] CRAN (R 4.3.3)
 askpass             1.2.1      2024-10-04 [1] CRAN (R 4.3.3)
 assertthat          0.2.1      2019-03-21 [1] CRAN (R 4.3.0)
 backports           1.5.0      2024-05-23 [1] CRAN (R 4.3.3)
 BIOMASS           * 2.2.3      2025-02-24 [1] CRAN (R 4.3.3)
 boot                1.3-31     2024-08-28 [1] CRAN (R 4.3.3)
 broom             * 1.0.7      2024-09-26 [1] CRAN (R 4.3.3)
 c2z               * 0.2.0      2023-08-10 [1] CRAN (R 4.3.0)
 cachem              1.1.0      2024-05-16 [1] CRAN (R 4.3.3)
 car                 3.1-3      2024-09-27 [1] CRAN (R 4.3.3)
 carData             3.0-5      2022-01-06 [1] CRAN (R 4.3.0)
 caret             * 7.0-1      2024-12-10 [1] CRAN (R 4.3.3)
 cellranger          1.1.0      2016-07-27 [1] CRAN (R 4.3.0)
 chromote            0.4.0      2025-01-25 [1] CRAN (R 4.3.3)
 class               7.3-23     2025-01-01 [1] CRAN (R 4.3.3)
 classInt            0.4-11     2025-01-08 [1] CRAN (R 4.3.3)
 cli                 3.6.3      2024-06-21 [1] CRAN (R 4.3.3)
 codetools           0.2-20     2024-03-31 [1] CRAN (R 4.3.1)
 colorspace          2.1-1      2024-07-26 [1] CRAN (R 4.3.3)
 data.table          1.16.4     2024-12-06 [1] CRAN (R 4.3.3)
 dataMaid          * 1.4.1      2021-10-08 [1] CRAN (R 4.3.0)
 DBI                 1.2.3      2024-06-02 [1] CRAN (R 4.3.3)
 DEoptimR            1.1-3-1    2024-11-23 [1] CRAN (R 4.3.3)
 DescTools         * 0.99.59    2025-01-26 [1] CRAN (R 4.3.3)
 devtools            2.4.5      2022-10-11 [1] CRAN (R 4.3.0)
 dials             * 1.3.0      2024-07-30 [1] CRAN (R 4.3.3)
 DiceDesign          1.10       2023-12-07 [1] CRAN (R 4.3.1)
 digest              0.6.37     2024-08-19 [1] CRAN (R 4.3.3)
 dplyr             * 1.1.4      2023-11-17 [1] CRAN (R 4.3.1)
 e1071               1.7-16     2024-09-16 [1] CRAN (R 4.3.3)
 easypackages        0.1.0      2016-12-05 [1] CRAN (R 4.3.0)
 ellipsis            0.3.2      2021-04-29 [1] CRAN (R 4.3.0)
 evaluate            1.0.3      2025-01-10 [1] CRAN (R 4.3.3)
 Exact               3.3        2024-07-21 [1] CRAN (R 4.3.3)
 expm                1.0-0      2024-08-19 [1] CRAN (R 4.3.3)
 extrafont         * 0.19       2023-01-18 [1] CRAN (R 4.3.3)
 extrafontdb         1.0        2012-06-11 [1] CRAN (R 4.3.3)
 fastmap             1.2.0      2024-05-15 [1] CRAN (R 4.3.3)
 flextable         * 0.9.7      2024-10-27 [1] CRAN (R 4.3.3)
 fontBitstreamVera   0.1.1      2017-02-01 [1] CRAN (R 4.3.3)
 fontLiberation      0.1.0      2016-10-15 [1] CRAN (R 4.3.3)
 fontquiver          0.2.1      2017-02-01 [1] CRAN (R 4.3.3)
```

```
forcats           * 1.0.0      2023-01-29 [1] CRAN (R 4.3.0)
foreach             1.5.2      2022-02-02 [1] CRAN (R 4.3.0)
formatR           * 1.14       2023-01-17 [1] CRAN (R 4.3.3)
Formula             1.2-5      2023-02-24 [1] CRAN (R 4.3.0)
fs                  1.6.5      2024-10-30 [1] CRAN (R 4.3.3)
furrr               0.3.1      2022-08-15 [1] CRAN (R 4.3.0)
future              1.34.0     2024-07-29 [1] CRAN (R 4.3.3)
future.apply        1.11.3     2024-10-27 [1] CRAN (R 4.3.3)
gdtools             0.4.1      2024-11-04 [1] CRAN (R 4.3.3)
generics            0.1.3      2022-07-05 [1] CRAN (R 4.3.0)
ggplot2           * 3.5.1      2024-04-23 [1] CRAN (R 4.3.1)
gld                 2.6.7      2025-01-17 [1] CRAN (R 4.3.3)
globals             0.16.3     2024-03-08 [1] CRAN (R 4.3.1)
glue                1.8.0      2024-09-30 [1] CRAN (R 4.3.3)
goftest             1.2-3      2021-10-07 [1] CRAN (R 4.3.3)
gower               1.0.2      2024-12-17 [1] CRAN (R 4.3.3)
GPfit               1.0-8      2019-02-08 [1] CRAN (R 4.3.0)
gridExtra           2.3        2017-09-09 [1] CRAN (R 4.3.0)
gtable              0.3.6      2024-10-25 [1] CRAN (R 4.3.3)
hardhat             1.4.0      2024-06-02 [1] CRAN (R 4.3.3)
haven               2.5.4      2023-11-30 [1] CRAN (R 4.3.1)
hms                 1.1.3      2023-03-21 [1] CRAN (R 4.3.0)
htmltools         * 0.5.8.1    2024-04-04 [1] CRAN (R 4.3.1)
htmlwidgets         1.6.4      2023-12-06 [1] CRAN (R 4.3.1)
httpuv              1.6.15     2024-03-26 [1] CRAN (R 4.3.1)
httr                1.4.7      2023-08-15 [1] CRAN (R 4.3.0)
infer             * 1.0.7      2024-03-25 [1] CRAN (R 4.3.1)
ipred               0.9-15     2024-07-18 [1] CRAN (R 4.3.3)
iterators           1.0.14     2022-02-05 [1] CRAN (R 4.3.0)
janitor           * 2.2.1      2024-12-22 [1] CRAN (R 4.3.3)
jsonlite          * 1.8.9      2024-09-20 [1] CRAN (R 4.3.3)
kableExtra        * 1.4.0      2024-01-24 [1] CRAN (R 4.3.1)
kernlab           * 0.9-33     2024-08-13 [1] CRAN (R 4.3.3)
KernSmooth          2.23-26    2025-01-01 [1] CRAN (R 4.3.3)
knitr             * 1.49       2024-11-08 [1] CRAN (R 4.3.3)
later               1.4.1      2024-11-27 [1] CRAN (R 4.3.3)
lattice           * 0.22-6     2024-03-20 [1] CRAN (R 4.3.1)
lava                1.8.1      2025-01-12 [1] CRAN (R 4.3.3)
lazyeval            0.2.2      2019-03-15 [1] CRAN (R 4.3.0)
lhs                 1.2.0      2024-06-30 [1] CRAN (R 4.3.3)
lifecycle           1.0.4      2023-11-07 [1] CRAN (R 4.3.1)
listenv             0.9.1      2024-01-29 [1] CRAN (R 4.3.1)
lmom                3.2        2024-09-30 [1] CRAN (R 4.3.3)
lubridate         * 1.9.4      2024-12-08 [1] CRAN (R 4.3.3)
magrittr            2.0.3      2022-03-30 [1] CRAN (R 4.3.0)
MASS                7.3-58.4   2023-03-07 [2] CRAN (R 4.3.0)
Matrix              1.6-5      2024-01-11 [1] CRAN (R 4.3.1)
memoise             2.0.1      2021-11-26 [1] CRAN (R 4.3.0)
mime                0.12       2021-09-28 [1] CRAN (R 4.3.0)
miniUI              0.1.1.1    2018-05-18 [1] CRAN (R 4.3.0)
minpack.lm          1.2-4      2023-09-11 [1] CRAN (R 4.3.3)
mnormt              2.1.1      2022-09-26 [1] CRAN (R 4.3.0)
modeldata         * 1.4.0      2024-06-19 [1] CRAN (R 4.3.3)
ModelMetrics        1.2.2.2    2020-03-17 [1] CRAN (R 4.3.0)
```

```
munsell             0.5.1      2024-04-01 [1] CRAN (R 4.3.1)
mvtnorm             1.3-3      2025-01-10 [1] CRAN (R 4.3.3)
nlme                3.1-166    2024-08-14 [1] CRAN (R 4.3.3)
nnet                7.3-20     2025-01-01 [1] CRAN (R 4.3.3)
nortest             1.0-4      2015-07-30 [1] CRAN (R 4.3.3)
officer             0.6.7      2024-10-09 [1] CRAN (R 4.3.3)
olsrr             * 0.6.1      2024-11-06 [1] CRAN (R 4.3.3)
openssl             2.3.1      2025-01-09 [1] CRAN (R 4.3.3)
pander              0.6.6      2025-03-01 [1] CRAN (R 4.3.3)
parallelly          1.41.0     2024-12-18 [1] CRAN (R 4.3.3)
parsnip           * 1.2.1      2024-03-22 [1] CRAN (R 4.3.1)
pillar              1.10.1     2025-01-07 [1] CRAN (R 4.3.3)
pkgbuild            1.4.6      2025-01-16 [1] CRAN (R 4.3.3)
pkgconfig           2.0.3      2019-09-22 [1] CRAN (R 4.3.0)
pkgload             1.4.0      2024-06-28 [1] CRAN (R 4.3.3)
plotly            * 4.10.4     2024-01-13 [1] CRAN (R 4.3.1)
plyr                1.8.9      2023-10-02 [1] CRAN (R 4.3.1)
pROC                1.18.5     2023-11-01 [1] CRAN (R 4.3.1)
processx            3.8.5      2025-01-08 [1] CRAN (R 4.3.3)
prodlim             2024.06.25 2024-06-24 [1] CRAN (R 4.3.3)
profvis             0.4.0      2024-09-20 [1] CRAN (R 4.3.3)
promises            1.3.2      2024-11-28 [1] CRAN (R 4.3.3)
proxy               0.4-27     2022-06-09 [1] CRAN (R 4.3.0)
ps                  1.8.1      2024-10-28 [1] CRAN (R 4.3.3)
psych             * 2.4.12     2024-12-23 [1] CRAN (R 4.3.3)
purrr             * 1.0.2      2023-08-10 [1] CRAN (R 4.3.0)
R6                  2.5.1      2021-08-19 [1] CRAN (R 4.3.0)
ragg                1.3.3      2024-09-11 [1] CRAN (R 4.3.3)
rappdirs            0.3.3      2021-01-31 [1] CRAN (R 4.3.0)
RColorBrewer      * 1.1-3      2022-04-03 [1] CRAN (R 4.3.0)
Rcpp                1.0.14     2025-01-12 [1] CRAN (R 4.3.3)
readr             * 2.1.5      2024-01-10 [1] CRAN (R 4.3.1)
readxl            * 1.4.3      2023-07-06 [1] CRAN (R 4.3.0)
recipes           * 1.1.0      2024-07-04 [1] CRAN (R 4.3.3)
remotes             2.5.0      2024-03-17 [1] CRAN (R 4.3.1)
reshape2            1.4.4      2020-04-09 [1] CRAN (R 4.3.0)
rlang               1.1.4      2024-06-04 [1] CRAN (R 4.3.3)
rmarkdown         * 2.29       2024-11-04 [1] CRAN (R 4.3.3)
robustbase          0.99-4-1   2024-09-27 [1] CRAN (R 4.3.3)
rootSolve           1.8.2.4    2023-09-21 [1] CRAN (R 4.3.3)
rpart               4.1.24     2025-01-07 [1] CRAN (R 4.3.3)
rsample           * 1.2.1      2024-03-25 [1] CRAN (R 4.3.1)
rstudioapi          0.17.1     2024-10-22 [1] CRAN (R 4.3.3)
Rttf2pt1            1.3.12     2023-01-22 [1] CRAN (R 4.3.3)
rvest               1.0.4      2024-02-12 [1] CRAN (R 4.3.1)
scales            * 1.3.0      2023-11-28 [1] CRAN (R 4.3.1)
sessioninfo         1.2.2      2021-12-06 [1] CRAN (R 4.3.0)
sf                  1.0-19     2024-11-05 [1] CRAN (R 4.3.3)
shiny               1.10.0     2024-12-14 [1] CRAN (R 4.3.3)
snakecase           0.11.1     2023-08-27 [1] CRAN (R 4.3.0)
stringi             1.8.4      2024-05-06 [1] CRAN (R 4.3.1)
stringr           * 1.5.1      2023-11-14 [1] CRAN (R 4.3.1)
survival            3.8-3      2024-12-17 [1] CRAN (R 4.3.3)
svglite             2.1.3      2023-12-08 [1] CRAN (R 4.3.1)
```

```
systemfonts         1.1.0      2024-05-15 [1] CRAN (R 4.3.3)
terra               1.8-29     2025-02-26 [1] CRAN (R 4.3.3)
textshaping         0.4.1      2024-12-06 [1] CRAN (R 4.3.3)
tibble            * 3.2.1      2023-03-20 [1] CRAN (R 4.3.0)
tidymodels        * 1.2.0      2024-03-25 [1] CRAN (R 4.3.1)
tidyr             * 1.3.1      2024-01-24 [1] CRAN (R 4.3.1)
tidyselect          1.2.1      2024-03-11 [1] CRAN (R 4.3.1)
tidyverse         * 2.0.0      2023-02-22 [1] CRAN (R 4.3.0)
timechange          0.3.0      2024-01-18 [1] CRAN (R 4.3.1)
timeDate            4041.110   2024-09-22 [1] CRAN (R 4.3.3)
tinytex           * 0.54       2024-11-01 [1] CRAN (R 4.3.3)
truncnorm         * 1.0-9      2023-03-20 [1] CRAN (R 4.3.3)
tune              * 1.2.1      2024-04-18 [1] CRAN (R 4.3.1)
tzdb                0.4.0      2023-05-12 [1] CRAN (R 4.3.0)
units               0.8-5      2023-11-28 [1] CRAN (R 4.3.1)
urlchecker          1.0.1      2021-11-30 [1] CRAN (R 4.3.0)
useful            * 1.2.6.1    2023-10-24 [1] CRAN (R 4.3.1)
usethis             3.1.0      2024-11-26 [1] CRAN (R 4.3.3)
uuid                1.2-1      2024-07-29 [1] CRAN (R 4.3.3)
vctrs               0.6.5      2023-12-01 [1] CRAN (R 4.3.1)
viridisLite         0.4.2      2023-05-02 [1] CRAN (R 4.3.0)
webshot           * 0.5.5      2023-06-26 [1] CRAN (R 4.3.0)
webshot2          * 0.1.1      2023-08-11 [1] CRAN (R 4.3.0)
websocket           1.4.2      2024-07-22 [1] CRAN (R 4.3.3)
withr               3.0.2      2024-10-28 [1] CRAN (R 4.3.3)
workflows         * 1.1.4      2024-02-19 [1] CRAN (R 4.3.1)
workflowsets      * 1.1.0      2024-03-21 [1] CRAN (R 4.3.1)
xfun                0.50       2025-01-07 [1] CRAN (R 4.3.3)
xml2                1.3.6      2023-12-04 [1] CRAN (R 4.3.1)
xtable              1.8-4      2019-04-21 [1] CRAN (R 4.3.0)
yaml                2.3.10     2024-07-26 [1] CRAN (R 4.3.3)
yardstick         * 1.3.1      2024-03-21 [1] CRAN (R 4.3.1)
zip                 2.3.1      2024-01-27 [1] CRAN (R 4.3.1)

 [1] /Library/Frameworks/R.framework/Versions/4.1-arm64/Resources/library
 [2] /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library

──────────────────────────────────────────────────────────────────────────────

# Sys.getenv() .libPaths()
```

References

Annex I: SimVoi Functions & Syntax

SimVoi adds seventeen random number generator functions defined with the following syntax:

- `RandBeta(alpha,beta,,[MinValue],[MaxValue])`
- `RandBinomial(trials,probability_s)`
- `RandBiVarNormal(mean1,stdev1,mean2,stdev2,correl12)`
- `RandCumulative(value_cumulative_table)`
- `RandDiscrete(value_discrete_table)`
- `RandExponential(lambda)`
- `RandInteger(bottom,top)`
- `RandLogNormal(Mean,StDev)`

- `RandNormal(mean,standard_dev)`
- `RandPoisson(mean)`
- `RandSample(population)`
- `RandTriangular(minimum,most_likely,maximum)`
- `RandTriBeta(minimum,most_likely,maximum,[shape])`
- `RandTruncBiVarNormal(mean1,stdev1,mean2,stdev2,correl12, [min1],[max1],[min2],[max2])`
- `RandTruncLogNormal(Mean,StDev,[MinValue],[MaxValue])`
- `RandTruncNormal(Mean,StDev,[MinValue],[MaxValue])`
- `RandUniform(minimum,maximum)`

In the following, we attempt to match the SimVoi Excel formula of

```
=[1]!randtruncnormal(CarbonStocks.B2,CarbonStocks.B3,0)
```

function, as closely as random seeding allows. According to package documentation, the `RandTruncNormal()` function "*Returns a random value from a truncated normal probability density function. This function can model an uncertain quantity with a bell-shaped density function where extreme values in the tails of the distribution are not desired.*"

In terms of simulation parameters, *"RandTruncNormal(Mean,StDev,MinValue,MaxValue)) uses values of RandNormal until a value is found between MinValue and MaxValue or until it has made 10,000 attempts."* The above formula provides a minimum value of `0`, passing to the default number of simulations of 10,000.