Monte Carlo Simulation Tools for REDD+ Uncertainty Estimates

2024-12-19

Table of Contents

## Objective

When preparing for Monte Carlo simulations, it is best practice to start by examining descriptive statistics to characterize the empirical distributions of input variables. This preliminary analysis typically includes statistical tests for normality and visualizations of univariate distributions, such as histograms, kernel density plots, and Q-Q plots. Together, these tools provide critical insights into the shape, spread, symmetry, skewness, and presence of potential outliers in the data. Although this preliminary step may seem minor, it substantially influences uncertainty estimates, which can directly translate into increased financial returns, particularly within forest project landscapes exhibiting non-normal data distributions.

Accurately characterizing data distributions also helps in identifying and addressing biases, thereby ensuring high data quality and increasing confidence in subsequent estimations of biomass and carbon emissions. Selecting appropriate statistical distributions, informed by exploratory analyses, significantly enhances the reliability and precision of Monte Carlo simulations. Consequently, such careful statistical characterizations reduce overall uncertainty in forest biomass and emissions estimates. In turn, this strengthens the credibility of jurisdictional claims made under REDD+ programs and maximizes potential financial returns for Guyana from carbon financing initiatives.

Univariate distribution visualizations additionally provide auditors with useful diagnostic resources, enabling rapid identification and characterization of biases commonly encountered in biomass data. These diagrams help auditors efficiently assess the technical rigor and statistical approaches implemented by the project to monitor and manage uncertainty (ART, 2021: 8). Winrock strongly recommends incorporating distribution analyses early in a project’s quantitative planning and throughout its technical standard operating procedures (SOPs). Such early integration represents a low hanging fruit with cost-effective strategy and significant potential in reducing audit findings, lowering uncertainty, and enhancing financial outcomes for Guyana’s REDD+ activities. Specifically, early attention to data distributions directly informs appropriate simulation selection from the available options in SimVoi.

To effectively guide practitioners and stakeholders in selecting appropriate statistical distributions for Monte Carlo methods within forestry and REDD+ contexts, the following two tables present findings from a rapid review of relevant literature. The review identified and summarized statistical distributions frequently encountered in forestry, biomass estimation, and emissions analysis, which are disaggregated below between discrete and continuous types and according to their inherent statistical characteristics.

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

| Distribution | Statistical Criteria & Use Cases | PDF |
| --- | --- | --- |
| Normal (Gaussian) | Symmetric, bell-shaped distribution used for modeling continuous variables: biomass/ha |  |
| Lognormal | Right-skewed distribution suitable for variables constrained to positive values (e.g., emission rates). |  |
| Exponential | Models waiting times between independent events, such as forest fire occurrences or logging events. |  |
| Continuous Uniform | Assumes all values in an interval [a, b] are equally likely; useful for random spatial sampling in forests. |  |
| Chi-Square | Often used in goodness-of-fit tests to evaluate model accuracy in biomass estimation. |  |
| t-Distribution | Suitable for small sample sizes with unknown population stdev (e.g., limited forest carbon data). |  |
| Gamma | Models positively skewed data, such as biomass growth rates or carbon accumulation over time. |  |
| Weibull | Flexible distribution used in reliability analysis, e.g., modeling tree mortality. |  |

###### *Table 2: Discrete data distributions, and example use cases designed with Monte Carlo simulations.*

| Distribution | Statistical Criteria & Use Cases | PMF |
| --- | --- | --- |
| Bernoulli | Binary outcome probability, e.g., presence/absence of deforestation in an area. |  |
| Binomial | Probability of fixed #no. of successes over Bernoulli trials, e.g., no. of heads in 10 coin flips. |  |
| Poisson | Models counts of independent events within an interval, e.g., number of wildfire incidents per year. |  |
| Geometric | Models #no. of trials until the first success, e.g., number of inspections until detecting deforestation. |  |
| Negative Binomial | Counts #no. failures until successes occur, treats overdispersed or repeated deforestation detections. |  |
| Discrete Uniform | Assumes outcome in a finite set is equally likely, e.g., random sampling of inventory across a forest. |  |

Discrete distributions describe forestry monitoring scenarios where data outcomes are countable and finite. Common examples include the number of deforestation events, occurrences of wildfires, or counts of logged trees within a defined monitoring interval. Accurate representation of discrete events using appropriate distributions such as Binomial, Poisson, or Negative Binomial significantly enhances the accuracy of model predictions and uncertainty assessments. For instance, employing a Poisson distribution to model occurrences of illegal logging events can improve the precision of estimated deforestation emissions and reduce uncertainty around compliance risks.

In contrast, continuous distributions capture variables capable of taking any value within a specified range and are particularly relevant in forestry when modeling measurements such as tree heights, carbon stock densities, or biomass values. Continuous distributions like the Normal (Gaussian), Lognormal, Weibull, and Gamma distributions frequently arise in ecological modeling and biomass estimations due to their ability to realistically represent ecological variability and complex environmental factors. For example, using a Lognormal distribution for tree biomass data often provides more reliable estimates, particularly when the dataset is right-skewed due to natural variability in tree growth and forest conditions.

Central to these distributions are two mathematical concepts: Probability Mass Functions (PMFs) for discrete data and Probability Density Functions (PDFs) for continuous data. PMFs allocate specific probabilities to discrete outcomes, essential for accurately simulating events such as species occurrences or forest disturbances. PDFs describe the relative likelihood of continuous data points, enabling the robust estimation of variables like forest carbon content or annual biomass increment.

In Monte Carlo simulations, precise definition and utilization of PMFs and PDFs are crucial. These functions underpin random sampling processes that directly influence the reliability, precision, and credibility of uncertainty estimates. Given that forestry data is known to exhibit non-normal distributions due to inherent ecological heterogeneity that, informed selection and rigorous application of these functions are vital. Accurate modeling of the underlying data distribution enhances biomass and emissions estimates, significantly reduces uncertainty, and bolsters the financial and ecological credibility of REDD+ reporting initiatives (Morgan & Henrion, 1990; IPCC, 2019; ART, 2021).

Practitioners are encouraged to conduct exploratory data analysis early in their project planning stages, integrating statistical tests of normality and visual assessments (histograms, kernel density plots, Q-Q plots). Such preliminary analyses assist in diagnosing data distributions accurately, improving model selection, reducing potential auditor findings, and ultimately enhancing the financial and environmental outcomes of national REDD+ monitoring programs.

## Method

***Import***

# 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") |>  
 janitor::clean\_names() |>  
 mutate(across(where(is.numeric), ~round(.x, 1)))  
CarbonStocks\_MC = readxl::read\_excel(workbook, "CarbonStocks (MC)") |>  
 janitor::clean\_names() |>  
 mutate(across(where(is.numeric), ~round(.x, 1)))  
  
DeforestationEF = readxl::read\_excel(workbook, "Deforestation EFs") |>  
 janitor::clean\_names() |>  
 mutate(across(where(is.numeric), ~round(.x, 1)))  
DeforestationEF\_MC = readxl::read\_excel(workbook, "Deforestation EFs (MC)") |>  
 janitor::clean\_names() |>  
 mutate(across(where(is.numeric), ~round(.x, 1)))  
  
DegradationEF = readxl::read\_excel(workbook, "Degradation EFs") |>  
 janitor::clean\_names() |>  
 mutate(across(where(is.numeric), ~round(.x, 1)))  
DegradationEF\_MC = readxl::read\_excel(workbook, "Degradation EFs (MC)") |>  
 janitor::clean\_names() |>  
 mutate(across(where(is.numeric), ~round(.x, 1)))  
  
ActivityData = readxl::read\_excel(workbook, "Activity Data") |>  
 janitor::clean\_names() |>  
 mutate(across(where(is.numeric), ~round(.x, 1)))  
ActivityData\_MC = readxl::read\_excel(workbook, "Activity Data (MC)") |>  
 janitor::clean\_names() |>  
 mutate(across(where(is.numeric), ~round(.x, 1)))  
  
Emissions = readxl::read\_excel(workbook, "Emissions") |>  
 janitor::clean\_names() |>  
 mutate(across(where(is.numeric), ~round(.x, 1)))  
Emissions\_MC = readxl::read\_excel(workbook, "Emissions (MC)") |>  
 janitor::clean\_names() |>  
 mutate(across(where(is.numeric), ~round(.x, 1)))  
  
Crediting = readxl::read\_excel(workbook, "ART Crediting Period") |>  
 janitor::clean\_names() |>  
 mutate(across(where(is.numeric), ~round(.x, 1)))  
Crediting\_MC = readxl::read\_excel(workbook, "ART Crediting Period") |>  
 janitor::clean\_names() |>  
 mutate(across(where(is.numeric), ~round(.x, 1)))  
  
EmissionsReductions = readxl::read\_excel(workbook, "Emission Reductions") |>  
 janitor::clean\_names() |>  
 mutate(across(where(is.numeric), ~round(.x, 1)))  
EmissionsReductions\_MC = readxl::read\_excel(workbook, "Emission Reductions (MC)") |>  
 janitor::clean\_names() |>  
 mutate(across(where(is.numeric), ~round(.x, 1)))  
  
# Vislualize  
flextable(head(CarbonStocks\_MC[, 1:8])) |>  
 fontsize(size = 8, part = "all")

| x1 | ag\_tree\_t\_c\_ha | bg\_tree\_t\_c\_ha | saplings\_t\_c\_ha | standing\_dead\_wood\_t\_c\_ha | lying\_dead\_wood\_t\_c\_ha | sum\_carbon\_pools\_w\_o\_litter\_t\_c\_ha | litter\_t\_c\_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 |

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

| x1 | ag\_tree\_t\_c\_ha | bg\_tree\_t\_c\_ha | saplings\_t\_c\_ha | standing\_dead\_wood\_t\_c\_ha | lying\_dead\_wood\_t\_c\_ha | sum\_carbon\_pools\_w\_o\_litter\_t\_c\_ha | litter\_t\_c\_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 |  |

flextable(head(DeforestationEF\_MC[, 1:8])) |>  
 fontsize(size = 8, part = "all")

| stratum | drivers | emission\_factors | x4 | x5 | x6 | x7 | x8 |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | tC/ha | t CO2/ha |  |  |  |  |
| Combined - all forest | Forestry infrastructure | 292.629943386449 | 1072.97645908365 |  |  |  |  |
|  | Agriculture | 309.268363331635 | 1133.98399888266 |  |  |  |  |
|  | Mining (medium and large scale) | 292.629943386449 | 1072.97645908365 |  |  |  |  |
|  | Mining infrastructure | 292.629943386449 | 1072.97645908365 |  |  |  |  |
|  | Infrastructure | 292.629943386449 | 1072.97645908365 |  |  |  |  |

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

| stratum | drivers | emission\_factors | x4 | x5 | x6 | x7 | x8 |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | tC/ha | t CO2/ha | uncertainty (IPCC approach 1) | uncertainty (IPCC approach 1) |  |  |
| Combined - all forest | Forestry infrastructure | 286.725980831211 | 1051.32859638111 | 0.0482813396659251 | 35.3685737111713 |  |  |
|  | Agriculture | 302.835379349672 | 1110.3963909488 | 0.0482813396659251 | 35.3685737111713 |  |  |
|  | Mining (medium and large scale) | 286.725980831211 | 1051.32859638111 | 0.0482813396659251 | 35.3685737111713 |  |  |
|  | Mining infrastructure | 286.725980831211 | 1051.32859638111 | 0.0482813396659251 | 35.3685737111713 |  |  |
|  | Infrastructure | 286.725980831211 | 1051.32859638111 | 0.0482813396659251 | 35.3685737111713 |  |  |

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

| x1 | x2 | x3 | change\_data\_from\_wall\_to\_wall\_mapping\_by\_gfc | x5 | x6 | x7 | x8 |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | Drivers | units | 2011 | 2,012 | 2,013 | 2014 | 2,015 |
|  | Deforestation |  |  |  |  |  |  |
| Deforestation | Forestry infrastructure | ha | 186 | 240 | 330 | 204 | 313 |
|  | Agriculture |  | 41 | 440 | 424 | 817 | 379 |
|  | Mining (medium and large scale) |  | 7340 | 13,664 | 11,518 | 10434 | 6,782 |
|  | Mining infrastructure |  |  |  |  |  |  |

flextable(head(ActivityData\_MC[, 1:8])) |>  
 fontsize(size = 8, part = "all")

| x1 | x2 | x3 | change | x5 | x6 | x7 | x8 |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | Drivers | units | 2,011.0 | 2,012.0 | 2,013.0 | 2014 | 2,015.0 |
| Deforestation | Forestry infrastructure | ha | 225.2 | 194.4 | 269.9 | 229.322762617862 | 325.1 |
|  | Agriculture |  | 36.2 | 384.1 | 462.7 | 881.606014127345 | 436.2 |
|  | Mining (medium and large scale) |  | 8,835.0 | 13,156.9 | 7,686.7 | 12583.0709342038 | 7,673.2 |
|  | Mining infrastructure |  | 0.0 | 0.0 | 0.0 | 0 | 0.0 |
|  | Infrastructure |  | 116.8 | 121.1 | 330.8 | 142.547242332744 | 182.1 |

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

| drivers | units | x2011 | x2012 | x2013 | x2014 | x2015 | x2016 |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Forestry infrastructure | tCO2 | 195,547.1 | 252,318.9 | 346,938.4 | 214,471.0 | 329065.850667287 | 329,065.9 |
| Agriculture |  | 45,526.3 | 488,574.4 | 470,808.1 | 907,193.9 | 420840.232169594 | 420,840.2 |
| Mining (medium and large scale) |  | 7,716,751.9 | 14,365,353.9 | 12,109,202.8 | 10,969,562.6 | 7130110.54065667 | 7,130,110.5 |
| Infrastructure |  | 124,477.3 | 133,518.7 | 359,554.4 | 148,237.3 | 228138.3054147 | 228,138.3 |
| Settlements |  | 0.0 | 0.0 | 24,180.6 | 74,644.3 | 8410.62877104886 | 8,410.6 |
| Fire-Biomass burning |  | 48,436.9 | 193,747.7 | 101,085.8 | 272,721.0 | 1588942.11225164 | 1,588,942.1 |

flextable(head(Emissions\_MC[, 1:8])) |>  
 fontsize(size = 8, part = "all")

| drivers | units | x2011 | x2012 | x2013 | x2014 | x2015 | x2016 |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Forestry infrastructure | tCO2 | 241,594.0 | 208,610.9 | 289,633.3 | 246,057.9 | 348,853.5 | 308,654.5 |
| Agriculture |  | 41,036.5 | 435,510.7 | 524,713.8 | 999,727.1 | 494,631.4 | 454,779.0 |
| Mining (medium and large scale) |  | 9,479,796.2 | 14,117,045.3 | 8,247,621.5 | 13,501,338.9 | 8,233,167.9 | 7,599,187.4 |
| Infrastructure |  | 125,287.5 | 129,973.5 | 354,898.9 | 152,949.8 | 195,363.9 | 253,644.8 |
| Settlements |  | 0.0 | 0.0 | 26,317.3 | 78,166.4 | 8,033.6 | 7,630.6 |
| Fire-Biomass burning |  | 55,974.3 | 199,540.4 | 94,090.4 | 319,448.8 | 1,946,314.3 | 1,649,411.3 |

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

| logging\_emission\_factors | x2 | x3 | x4 | x5 | x6 | x7 | x8 |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Component | Unit | Factor (tC) | Std Dev (tC) | 90% CI (tC) | t CO2 | Std Dev (tCO2) | 90% CI (tCO2) |
| LDF | per m3 | 1.05 | 0.68 | 0.08 | 3.85 | 2.49333333333333 | 0.293333333333333 |
| Wood Density of timber harvested | per m3 | 0.4 | 0.03 | 0.003 | 1.46666666666667 | 0.11 | 0.011 |
| LIF (Skid Trails) | per km | 46.8650956266483 | 8.08 | 1.6 | 171.838683964377 | 29.6266666666667 | 5.86666666666667 |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

# flextable(head(DegradationEF\_MC[, 1:8])) |> fontsize(size = 8, part = 'all')  
# dplyr::glimpse(CarbonStocks)

### Tidy

Data cleaning tasks often needed for dataframes imported with readxl::read\_exel() function, as variables, labels and dataframes are corrupted in the process. This especially likely with summary statistics in non-standard formats, such as in Guyana’s workbook data. Re-installing and applying the function janitor::clean\_names() may sometimes solve this, but more often not. For future debugging, I added notes in this Tidy section on the steps identified to complete data cleaning.

We begin by identifying the relevant rows and columns for each pool, specifically those containing mean, standard deviation, minimum, maximum, and confidence interval values. Assuming rows in the “CarbonStocks\_MC” tab maintain the same order, these cleaning operations can hopefully be repeated quickly. A common approach involves reshaping the data so that each row represents a “Statistic,” such as mean or standard deviation, and each column corresponds to a carbon pool, like “AG Tree” or “BG Tree.”

In the chunk below we select columns pertinent to carbon pools, including “AG Tree (tC/ha)”, “BG Tree (tC/ha),” and rename them to match the “SimVoi” workbook. Subsequently, we extract the rows containing the summary statistics, and reshape the data to our preferred layout. To effectively transpose the data and transition between wide and long formats, utilize the tidyr package’s pivot\_longer() and pivot\_wider() functions, which essentially flip rows and columns. Finally, you must pivot back from long to wide layout to ensure that “Statistic” becomes a distinct column and the carbon pools, such as “AG\_Tree” and “BG\_Tree,” are represented as separate variable columns. Happy to walk you through this again if you need.

CarbonStocks = CarbonStocks |>  
 dplyr::rename(Statistic = x1) |>  
 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)  
  
# Convert wide to long, use 'Statistic' to define row  
CarbonStocks\_long <- CarbonStocks |>  
 tidyr::pivot\_longer(cols = -Statistic, names\_to = "Pool", values\_to = "Value") |>  
 mutate(Value = as.numeric(Value))  
  
# Convert from long back to wide format:  
CarbonStocks\_wide <- CarbonStocks\_long %>%  
 pivot\_wider(names\_from = Statistic, values\_from = Value)  
  
# 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)

### Distribution Analysis

# Descriptive statistics  
psych::describe(CarbonStocks)

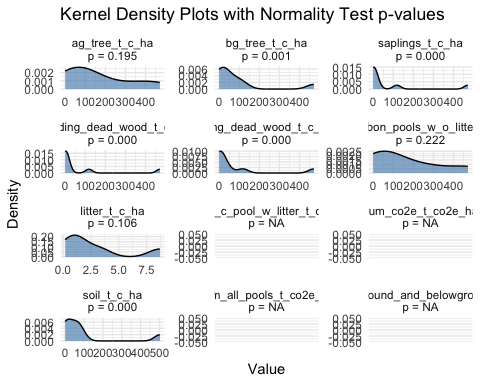
|  | vars | n | mean | sd | median | trimmed | mad | min | max | range | skew | kurtosis | se |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| x1\* | 1 | 8 | 4.5000 | 2.449490 | 4.50 | 4.5000 | 2.96520 | 1.0 | 8.0 | 7.0 | 0.0000000 | -1.6510417 | 0.8660254 |
| ag\_tree\_t\_c\_ha | 2 | 8 | 163.8375 | 169.683267 | 104.80 | 163.8375 | 145.73958 | 0.0 | 472.0 | 472.0 | 0.6781874 | -1.2269063 | 59.9920945 |
| bg\_tree\_t\_c\_ha | 3 | 8 | 94.8875 | 157.922300 | 34.75 | 94.8875 | 49.88949 | 0.0 | 472.0 | 472.0 | 1.6175633 | 1.1111536 | 55.8339644 |
| saplings\_t\_c\_ha | 4 | 8 | 76.9250 | 164.638060 | 2.85 | 76.9250 | 3.92889 | 0.1 | 472.0 | 471.9 | 1.6631166 | 1.1751098 | 58.2083442 |
| standing\_dead\_wood\_t\_c\_ha | 5 | 8 | 76.3875 | 164.854242 | 3.30 | 76.3875 | 4.74432 | 0.0 | 472.0 | 472.0 | 1.6639326 | 1.1754532 | 58.2847764 |
| lying\_dead\_wood\_t\_c\_ha | 6 | 8 | 81.2875 | 162.909725 | 8.35 | 81.2875 | 12.30558 | 0.0 | 472.0 | 472.0 | 1.6558398 | 1.1692588 | 57.5972855 |
| sum\_carbon\_pools\_w\_o\_litter\_t\_c\_ha | 7 | 6 | 157.6167 | 182.097890 | 96.60 | 157.6167 | 134.69421 | 0.0 | 472.0 | 472.0 | 0.6844433 | -1.3259625 | 74.3411524 |
| litter\_t\_c\_ha | 8 | 5 | 2.9400 | 3.410718 | 1.30 | 2.9400 | 1.63086 | 0.2 | 8.7 | 8.5 | 0.8110507 | -1.2656277 | 1.5253196 |
| sum\_c\_pool\_w\_litter\_t\_c\_ha | 9 | 1 | 272.3000 | NA | 272.30 | 272.3000 | 0.00000 | 272.3 | 272.3 | 0.0 | NA | NA | NA |
| sum\_co2e\_t\_co2e\_ha | 10 | 1 | 998.5000 | NA | 998.50 | 998.5000 | 0.00000 | 998.5 | 998.5 | 0.0 | NA | NA | NA |
| x11 | 11 | 0 | NaN | NA | NA | NaN | NA | Inf | -Inf | -Inf | NA | NA | NA |
| soil\_t\_c\_ha | 12 | 7 | 104.4143 | 178.470048 | 58.70 | 104.4143 | 70.72002 | 0.2 | 502.4 | 502.2 | 1.5096618 | 0.5723095 | 67.4553377 |
| x13 | 13 | 0 | NaN | NA | NA | NaN | NA | Inf | -Inf | -Inf | NA | NA | NA |
| sum\_all\_pools\_t\_co2e\_ha | 14 | 1 | 1213.7000 | NA | 1213.70 | 1213.7000 | 0.00000 | 1213.7 | 1213.7 | 0.0 | NA | NA | NA |
| sum\_aboveground\_and\_belowground\_live\_tree | 15 | 1 | 931.9000 | NA | 931.90 | 931.9000 | 0.00000 | 931.9 | 931.9 | 0.0 | NA | NA | NA |

psych::describe(Emissions)  
psych::describe(DeforestationEF)  
psych::describe(DegradationEF)  
psych::describe(ActivityData)  
psych::describe(EmissionsReductions)  
psych::describe(Crediting)  
  
CarbonStocks$  
MASS::truehist(CarbonStocks$ag\_tree\_t\_c\_ha,nbins=30,xlab= "ag\_tree\_t\_c\_ha",main=paste("Distribution of", "ag\_tree\_t\_c\_ha"),col = "gray")  
MASS::truehist(CarbonStocks$bg\_tree\_t\_c\_ha,nbins=30,xlab= "bg\_tree\_t\_c\_ha",main=paste("Distribution of", "bg\_tree\_t\_c\_ha"),col = "gray")  
MASS::truehist(CarbonStocks$bg\_tree\_t\_c\_ha,nbins=30,xlab= "bg\_tree\_t\_c\_ha",main=paste("Distribution of", "bg\_tree\_t\_c\_ha"),col = "gray")  
MASS::truehist(CarbonStocks$bg\_tree\_t\_c\_ha,nbins=30,xlab= "bg\_tree\_t\_c\_ha",main=paste("Distribution of", "bg\_tree\_t\_c\_ha"),col = "gray")  
MASS::truehist(CarbonStocks$bg\_tree\_t\_c\_ha,nbins=30,xlab= "bg\_tree\_t\_c\_ha",main=paste("Distribution of", "bg\_tree\_t\_c\_ha"),col = "gray")  
MASS::truehist(CarbonStocks$bg\_tree\_t\_c\_ha,nbins=30,xlab= "bg\_tree\_t\_c\_ha",main=paste("Distribution of", "bg\_tree\_t\_c\_ha"),col = "gray")  
MASS::truehist(CarbonStocks$bg\_tree\_t\_c\_ha,nbins=30,xlab= "bg\_tree\_t\_c\_ha",main=paste("Distribution of", "bg\_tree\_t\_c\_ha"),col = "gray")  
MASS::truehist(CarbonStocks$bg\_tree\_t\_c\_ha,nbins=30,xlab= "bg\_tree\_t\_c\_ha",main=paste("Distribution of", "bg\_tree\_t\_c\_ha"),col = "gray")  
  
truehist(CarbonStocks, nbins = nbins, xlab = var\_name, main = paste("Distribution of", var\_name), col="gray")  
truehist(CarbonStocks, nbins = nbins, xlab = var\_name, main = paste("Distribution of", var\_name), col="gray")  
truehist(CarbonStocks, nbins = nbins, xlab = var\_name, main = paste("Distribution of", var\_name), col="gray")  
truehist(CarbonStocks, nbins = nbins, xlab = var\_name, main = paste("Distribution of", var\_name), col="gray")  
truehist(CarbonStocks, nbins = nbins, xlab = var\_name, main = paste("Distribution of", var\_name), col="gray")  
truehist(CarbonStocks, nbins = nbins, xlab = var\_name, main = paste("Distribution of", var\_name), col="gray")  
truehist(CarbonStocks, nbins = nbins, xlab = var\_name, main = paste("Distribution of", var\_name), col="gray")

# Shapiro–Wilk normality test  
normalityTests <- function(data) {  
 numericData <- data[sapply(data, is.numeric)]  
 results <- sapply(numericData, function(x) {  
 x\_clean <- na.omit(x)  
 if (length(x\_clean) >= 3 && length(x\_clean) <= 5000) {  
 test <- shapiro.test(x\_clean)  
 c(W = test$statistic, p.value = test$p.value)  
 } else {  
 c(W = NA, p.value = NA)  
 }  
 })  
 results\_df <- as.data.frame(t(results))  
 return(results\_df)  
}  
  
# Function to plot kernel density plots with p-values annotated in facet labels  
plotKernelDensitiesWithNormality <- function(data) {  
 numericData <- data[sapply(data, is.numeric)]  
 meltedData <- melt(numericData, variable.name = "Variable", value.name = "Value")  
 norm\_results <- normalityTests(data)  
 norm\_results$Variable <- rownames(norm\_results)  
 norm\_results$p.value.formatted <- sprintf("p = %.3f", norm\_results$p.value)  
 facet\_labels <- setNames(paste0(norm\_results$Variable, "\n", norm\_results$p.value.formatted),  
 norm\_results$Variable)  
 ggplot(meltedData, aes(x = Value)) + geom\_density(fill = "steelblue", alpha = 0.6) +  
 facet\_wrap(~Variable, scales = "free", ncol = 3, labeller = as\_labeller(facet\_labels)) +  
 theme\_minimal() + labs(title = "Kernel Density Plots with Normality Test p-values",  
 x = "Value", y = "Density")  
}  
  
# Deploy:  
norm\_results <- normalityTests(CarbonStocks)  
print(norm\_results)

W.W p.value  
ag\_tree\_t\_c\_ha 0.8815235 1.947395e-01  
bg\_tree\_t\_c\_ha 0.6458968 5.427110e-04  
saplings\_t\_c\_ha 0.5603122 5.424425e-05  
standing\_dead\_wood\_t\_c\_ha 0.5559837 4.820435e-05  
lying\_dead\_wood\_t\_c\_ha 0.5865634 1.106360e-04  
sum\_carbon\_pools\_w\_o\_litter\_t\_c\_ha 0.8688451 2.216401e-01  
litter\_t\_c\_ha 0.8144912 1.058007e-01  
sum\_c\_pool\_w\_litter\_t\_c\_ha NA NA  
sum\_co2e\_t\_co2e\_ha NA NA  
soil\_t\_c\_ha 0.6168788 4.287387e-04  
sum\_all\_pools\_t\_co2e\_ha NA NA  
sum\_aboveground\_and\_belowground\_live\_tree NA NA

plotKernelDensitiesWithNormality(CarbonStocks)



# Function to plot distribution comparisons for a single numeric vector  
plotDistributionComparison <- function(CarbonStocks, var\_name = "Variable", bw\_method = "nrd0",  
 candidate\_dists = c("normal", "gamma", "lognormal", "weibull"), nbins = 30) {  
  
 # Remove missing values  
 x <- na.omit(x)  
 if (length(x) < 3) {  
 warning(paste("Not enough data in", var\_name, "to perform analysis."))  
 return(NULL)  
 }  
  
 # Set up a plot using MASS-truehist  
 truehist(CarbonStocks, nbins = nbins, xlab = var\_name, main = paste("Distribution of",  
 var\_name), col = "gray")  
  
 # Calculate and overlay a kernel density estimate with the specified  
 # bandwidth method  
 kd <- density(x, bw = bw\_method)  
 lines(kd, col = "blue", lwd = 2)  
  
 # Prepare a sequence for plotting fitted densities  
 x\_seq <- seq(min(x), max(x), length.out = 200)  
  
 # Initialize vectors for building the legend  
 legend\_labels <- c("Kernel Density")  
 legend\_colors <- c("blue")  
 legend\_lty <- c(1)  
  
 # Fit and plot a Normal distribution if requested  
 if ("normal" %in% candidate\_dists) {  
 fit\_norm <- try(fitdistr(x, "normal"), silent = TRUE)  
 if (!inherits(fit\_norm, "try-error")) {  
 dens\_norm <- dnorm(x\_seq, mean = fit\_norm$estimate["mean"], sd = fit\_norm$estimate["sd"])  
 lines(x\_seq, dens\_norm, col = "red", lwd = 2, lty = 2)  
 legend\_labels <- c(legend\_labels, "Normal Fit")  
 legend\_colors <- c(legend\_colors, "red")  
 legend\_lty <- c(legend\_lty, 2)  
 }  
 }  
  
 # Fit and plot a Gamma distribution (only if all values > 0)  
 if ("gamma" %in% candidate\_dists && all(x > 0)) {  
 fit\_gamma <- try(fitdistr(x, "gamma"), silent = TRUE)  
 if (!inherits(fit\_gamma, "try-error")) {  
 dens\_gamma <- dgamma(x\_seq, shape = fit\_gamma$estimate["shape"], rate = fit\_gamma$estimate["rate"])  
 lines(x\_seq, dens\_gamma, col = "green", lwd = 2, lty = 3)  
 legend\_labels <- c(legend\_labels, "Gamma Fit")  
 legend\_colors <- c(legend\_colors, "green")  
 legend\_lty <- c(legend\_lty, 3)  
 }  
 }  
  
 # Fit and plot a Lognormal distribution (only if all values > 0)  
 if ("lognormal" %in% candidate\_dists && all(x > 0)) {  
 fit\_lnorm <- try(fitdistr(x, "lognormal"), silent = TRUE)  
 if (!inherits(fit\_lnorm, "try-error")) {  
 dens\_lnorm <- dlnorm(x\_seq, meanlog = fit\_lnorm$estimate["meanlog"],  
 sdlog = fit\_lnorm$estimate["sdlog"])  
 lines(x\_seq, dens\_lnorm, col = "purple", lwd = 2, lty = 4)  
 legend\_labels <- c(legend\_labels, "Lognormal Fit")  
 legend\_colors <- c(legend\_colors, "purple")  
 legend\_lty <- c(legend\_lty, 4)  
 }  
 }  
  
 # Fit and plot a Weibull distribution (only if all values > 0)  
 if ("weibull" %in% candidate\_dists && all(x > 0)) {  
 fit\_weibull <- try(fitdistr(x, "weibull"), silent = TRUE)  
 if (!inherits(fit\_weibull, "try-error")) {  
 dens\_weibull <- dweibull(x\_seq, shape = fit\_weibull$estimate["shape"],  
 scale = fit\_weibull$estimate["scale"])  
 lines(x\_seq, dens\_weibull, col = "orange", lwd = 2, lty = 5)  
 legend\_labels <- c(legend\_labels, "Weibull Fit")  
 legend\_colors <- c(legend\_colors, "orange")  
 legend\_lty <- c(legend\_lty, 5)  
 }  
 }  
  
 # Add legend to the plot  
 legend("topright", legend = legend\_labels, col = legend\_colors, lwd = 2, lty = legend\_lty,  
 bty = "n")  
}  
  
# Function to loop through all numeric variables in a data frame  
exploratoryMASSAnalysis <- function(data, bw\_method = "nrd0", candidate\_dists = c("normal",  
 "gamma", "lognormal", "weibull"), nbins = 30) {  
 # Identify numeric columns  
 num\_vars <- names(data)[sapply(data, is.numeric)]  
  
 # Set up a multi-panel plotting layout (adjust rows/columns as needed)  
 n <- length(num\_vars)  
 ncol <- 2  
 nrow <- ceiling(n/ncol)  
 op <- par(mfrow = c(nrow, ncol))  
  
 # Loop over each numeric variable and generate plots  
 for (var in num\_vars) {  
 plotDistributionComparison(data[[var]], var\_name = var, bw\_method = bw\_method,  
 candidate\_dists = candidate\_dists, nbins = nbins)  
 }  
  
 # Reset plotting layout  
 par(op)  
}  
  
# Example usage:  
exploratoryMASSAnalysis(CarbonStocks, bw\_method = "nrd0", candidate\_dists = c("normal",  
 "gamma", "lognormal", "weibull"), nbins = 30)

The Coefficient of Variation CV is a standardized, unit-less measure of dispersion defined as the ratio of the standard deviation to the mean, typically expressed as a percentage. This standardization enables comparison of variability across datasets or scales, regardless of the underlying units, offering helpful tool for assessing novel data from periodic field inventories or mapping updates.

For these carbon stocks, a higher CV indicates greater relative variability or “scatter” in the data. While the CV is a useful indicator of dispersion and can signal potential non-normality, it does not provide any information on the direction of skew in the distribution.

In the following, the CV variable was computed from within the larger helper function calc\_derived\_stats. This helper function was designed as an aggregated relational estimate, which calculates CV while also comparing the reported 90% confidence interval with the standard deviation, which, under assumed normality, should approximate to ±1.645 × SD. This iterative scoring helps assess the internal consistency of the reported descriptive statistics.

# Helper function of derived descriptive statistics:  
calc\_derived\_stats <- function(df) {  
 df %>%  
 mutate(CV\_percent = 100 \* (`std. dev`/`mean of all plots (calculated)`),  
 sd\_implied\_by\_90CI = `90% CI`/1.645, 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)

## Replicating SimVoi

We utilize the replicate function to repeat a simulation following 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.

# Custom 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)  
 {draws <- truncnorm::rtruncnorm(  
 n = n\_draws,  
 a = min\_val,  
 b = max\_val,  
 mean = mean\_val,  
 sd = sd\_val)  
 return(draws)  
}  
  
simulate\_truncnorm\_from\_summary <- function(  
 mean\_val, sd\_val, min\_val=0, max\_val=Inf,   
 n\_draws=10000) {  
 draws <- truncnorm::rtruncnorm(  
 n = n\_draws,  
 a = min\_val,  
 b = max\_val,  
 mean = mean\_val,  
 sd = sd\_val  
 )  
 # Return vector of draws  
 return(draws)  
}  
  
# Repeat 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 may vote 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))  
  
  
# 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(CarbonStocks\_st\_stats, n\_draws=10000)  
  
ggplot(data.frame(AG\_draws), aes(x = AG\_draws)) +  
 geom\_histogram(aes(y = ..density..), bins = 50, fill = "skyblue", alpha = 0.7) +  
 geom\_density(col = "red") +  
 labs(title = "Monte Carlo Simulation of AG Tree Carbon Pool",  
 x = "Carbon Stock (tC/ha)", y = "Density")

## Compare simulations

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')  
  
# 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)

## 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.

## Annex II: Rapid literature review or Monte Carlo methods in REDD+

###### Table A.2: Search parameters, resource scope, and objectives informing search

|  |  |  |  |
| --- | --- | --- | --- |
| **REDD+**[[1]](#footnote-32) | **MC Application** | **Region** | **Key Findings** |
| ADD | Uncertainty of SAAB estimate | Rondônia, Brazil | Estimated ± 20% measurement error in SAAB using Monte Carlo simulations; emphasized large trees’ role in biomass. |
| ADD | AGB Uncertainty | Kenya, Mozambique | Assessed mixed-effects models in estimating mangrove biomass. |
| ADD | Blanket uncertainty propagation | Ghana | AGB prediction error >20%; addressed error propagation from trees to pixels in remote sensing. |
| ADD | Plot-based uncertainty | New Zealand | Cross-plot variance greatest magnitude of uncertainty |
| JNR | Multi-scale AGB uncertainty modeling | Minnesota, USA | Cross-scale tests showing effects of spatial resolution on AGB uncertainty. |
| N/A | Allometric uncertainty modeling | Panama | Allometric models identified as largest source of biomass estimation error. |
| ADD | Sampling and allometric uncertainty | Tapajos Nat Forest, Brazil | Significance of allometric models on uncertainty of root biomass, 95% CI, 21 plots. |
| ADD | Uncertainty of volume estimates | Santa Catarina, Brazil | Negligible effects of residual uncertainty on large-area estimates |
| N/A | Uncertainty metrics in model selection | Oregon, USA | Uncertainty estimates call for local validation or new local model development |
| ADD | AGB model uncertainty | French Guiana | AGB sub-model errors dominate uncertainty; height and wood-specific gravity errors are minor but can cause bias. |
| IFM | Emission factor uncertainty | Central Africa | Model selection is the largest error source (40%); weighting models reduces uncertainty in emission factors. |
| NA | Uncertainty in ecosystem nutrient estimate | New Hampshire, USA | Identified 8% uncertainty in nitrogen budgets, mainly from plot variability (6%) and allometric errors (5%). |

## References

1. ART, S. *The REDD+ Environmental Excellence Standard*; 2021. <https://www.artredd.org/wp-content/uploads/2021/12/TREES-2.0-August-2021-Clean.pdf>.
2. Bolker, B. (2008). *Ecological Models and Data in R.* Princeton University Press.
3. Brown, I. F.; Foster Brown, I.; Martinelli, L. A.; Wayt Thomas, W.; Moreira, M. Z.; Cid Ferreira, C. A.; Victoria, R. A. Uncertainty in the Biomass of Amazonian Forests: An Example from Rondônia, Brazil. *Forest Ecology and Management* 1995, *75* (1–3), 175–189. <https://doi.org/10.1016/0378-1127(94)03512-u>.
4. Cohen, R.; Kaino, J.; Okello, J. A.; Bosire, J. O.; Kairo, J. G.; Huxham, M.; Mencuccini, M. Uncertainty to Estimates of Above-Ground Biomass for Kenyan Mangroves: A Scaling Procedure from Tree to Landscape Level. In *Forest ecology and management*; 2013; Vol. 310, pp 968–982. <https://doi.org/10.1016/j.foreco.2013.09.047>.
5. Chen, Q.; Laurin, G. V.; Valentini, R. Uncertainty of Remotely Sensed Aboveground Biomass over an African Tropical Forest: Propagating Errors from Trees to Plots to Pixels. *Remote Sensing of Environment* 2015, *160*, 134–143. [https://doi.org/10.1016/j.rse.2015.01.009](#X6589fc6ab0dc82cf12099d1c2d40ab994e8410c).
6. Holdaway, R. J.; McNeill, S. J.; Mason, N. W. H.; Carswell, F. E. Propagating Uncertainty in Plot-Based Estimates of Forest Carbon Stock and Carbon Stock Change. *Ecosystems* 2014, *17*, 627–640. [https://doi.org/10.1007/s10021-014-9749-5](#X6589fc6ab0dc82cf12099d1c2d40ab994e8410c).
7. Chen, Q.; McRoberts, R. E.; Wang, C.; Radtke, P. J. Forest Aboveground Biomass Mapping and Estimation Across Multiple Spatial Scales Using Model-Based Inference. *Remote Sensing of Environment* 2016, *184*, 350–360. <https://doi.org/10.1016/j.rse.2016.07.023>.
8. Chave, J.; Condit, R.; Aguilar, S.; Hernandez, A.; Lao, S.; Perez, R. Error Propagation and Scaling for Tropical Forest Biomass Estimates. *Philosophical Transactions of the Royal Society of London. Series B: Biological Sciences* 2004, *359* (1443), 409–420.
9. Keller, M.; Palace, M.; Hurtt, G. Biomass Estimation in the Tapajos National Forest, Brazil. *Forest Ecology and Management* 2001, *154*, 371–382.
10. McRoberts, R. E.; Moser, P.; Oliveira, L. Z.; Vibrans, A. C. A General Method for Assessing the Effects of Uncertainty in Individual-Tree Volume Model Predictions on Large-Area Volume Estimates 222 with a Subtropical Forest Illustration. *Canadian Journal of Forest Research* 2015, *45*.
11. Melson, S. L.; Harmon, M. E.; Fried, J. S.; Domingo, J. B. Estimates of Live-Tree Carbon Stores in the Pacific Northwest Are Sensitive to Model Selection. *Carbon Balance and Management* 2011, *6*, 2.
12. Molto, Q.; Rossi, V.; Blanc, L. Error Propagation in Biomass Estimation in Tropical Forests. *Methods in Ecology and Evolution* 2013, *4*, 175–183. <https://doi.org/10.1111/j.2041-210x.2012.00266.x>.
13. Picard, N.; Bosela, F. B.; Rossi, V. Reducing the Error in Biomass Estimates Strongly Depends on Model Selection. *Annals of Forest Science* 2015, *72* (6), 811–823. <https://doi.org/10.1007/s13595-014-0434-9>.
14. Yanai, R. D.; Battles, J. J.; Richardson, A. D.; Blodgett, C. A.; Wood, D. M.; Rastetter, E. B. Estimating Uncertainty in Ecosystem Budget Calculations. *Ecosystems* 2010, *13*, 239–248. <https://doi.org/10.1007/s10021-010-9315-8>.

Limpert, E., Stahel, W. A., & Abbt, M. (2001). “Log-normal distributions across the sciences: Keys and clues.” *BioScience*, 51(5), 341–352.

Morgan, M. G., & Henrion, M. (1990). *Uncertainty: A Guide to Dealing with Uncertainty in Quantitative Risk and Policy Analysis.* Cambridge University Press.

Ross, S. M. (2019). *Introduction to Probability Models* (12th ed.). Academic Press.

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 2025-03-17  
 pandoc 3.6.1 @ /usr/local/bin/ (via rmarkdown)  
  
─ Packages ───────────────────────────────────────────────────────────────────  
 package \* version date (UTC) lib source  
 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)  
 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)  
 farver 2.1.2 2024-05-13 [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)  
 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)  
 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)  
 labeling 0.4.3 2023-08-29 [1] CRAN (R 4.3.0)  
 later 1.4.1 2024-11-27 [1] CRAN (R 4.3.3)  
 latex2exp \* 0.9.6 2022-11-28 [1] CRAN (R 4.3.0)  
 latexpdf \* 0.1.8 2023-12-19 [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)  
 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)  
 officer 0.6.7 2024-10-09 [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)  
 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)  
 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()

1. 1. ADD: Avoided deforestation degradation, IFM: Improved forest management, JNR: Jurisdictional nested REDD+ [↑](#footnote-ref-32)