Harvested Wood Products vR Documentation

Jeremiah Groom

2022-04-12

Contents

1 Summary **5**

1.1 Acknowledgements 5

1.2 Abbreviations 6

2 Introduction **7**

2.1 General description of the HWP model 7

2.2 Model history 7

2.3 Model utility 10

2.4 HWP model assumptions and shortcomings 10

2.5 Future work 11

3 HWP Model Operations **13**

3.1 HWP model function 13

3.2 Monte Carlo simulation 25

4 Operating and changing the model **41**

4.1 User interaction with the HWP model 41

4.2 Basic web interactivity – Shiny app 41

4.3 Operating the model from your own machine 44

5 Providing Your Own Data **53**

5.1 Overview 53

5.2 Creation of the input file 53

5.3 The quality assurance step 60

5.4 Uploading your data and running the model from the Shiny app 64
5.5 Using the Stand-alone to run the HWP model and Monte Carlo simulation on your data 65
5.6 Generated Tables 65
5.7 What to know about running the Monte Carlo simulation 68

6 Blocks **69**
6.1 Equations 69
6.2 Theorems and proofs 69
6.3 Callout blocks 69

7 Sharing your book **71**
7.1 Publishing 71
7.2 404 pages 71
7.3 Metadata for sharing 71

Chapter 1

Summary

The Harvested Wood Products (HWP) model is a deterministic model which relies on optional and required user-supplied information. The model is described in Stockmann et al. [2012]. There have been several builds of the model, on platforms including Microsoft Excel, C++, and now R [R Core Team, 2021]. Another effort is currently underway to construct an interactive version of the model in Python. This article describes a build of the model based on data from the U.S. west coast states of California, Oregon, and Washington. This R-based platform allows for user interaction with California and Oregon data via a web application (https://groomanalyticsllc.shinyapps.io/HWP_Shared_Test/). Users may also upload their own data onto the app, run their data through the HWP model and associated Monte Carlo simulation, interact with their data, and construct and save figures. Alternatively, users may access the full Shiny app code as well as a stand-alone version of the HWP model implementation. The stand-alone version allows users the opportunity to step through the code more easily, save model arrays for hand-calculation of outcomes, and alter the code to meet their own data needs. Both versions allow users to download or generate the tables necessary to recreate figures to their own specifications. This document describes the model and structure of the code, explains how to generate input files, and provides instruction on accessing model code.

1.1 Acknowledgements

Many thanks to Nadia Tase and Andrew Yost for supporting the development of this work through feedback, discussion, and their own efforts to dive into the code and understand the model at its core. Jimmy Kagan and Amanda Reynolds provided administrative support. We received feedback and support for the earlier iteration of the R-version of the model from Todd Morgan, Dan

Loffler, Keith Stockmann, [OTHERS].
This project was funded by the California Department of Forestry and Fire Protection Purchase Order 3540-0000277546. Earlier work on the project was funded by the Oregon Department of Forestry and supported by the Institute for Natural Resources, Oregon State University.

1.2 Abbreviations

The following description uses many acronyms for brevity. Here is a list of the most common:

BF: Board Feet
BBF: Billion Board Feet
BLM: Bureau of Land Management
BWEC: Burned with Energy Capture
EWEC: Emitted with Energy Capture
EWOEC: Emitted without Energy Capture
EUR: End-use Product Ratios
HWP: Harvested Wood Products
CCF: Hundred Cubic Feet
IPCC: Intergovernmental Panel on Climate Change
MBF: Thousand Board Feet), MT: Metric Tons
MMT C: Million Metric Tons of Carbon
PPR: Primary Product Ratios
PIU; Products in Use
SWDS: Solid Waste Disposal Sites
Tg C: Teragrams of Carbon: Tg C
Tg CO2e: Teragrams of Carbon Dioxide Equivalent
TPO: Timber Product Output
TPR: Timber Product Ratios
USFS: United States Forest Service

A note about units for carbon:
Publications from the United States have frequently described carbon storage and emissions using the metric million metric tons carbon, or MMT C. We are using the equivalent metric, teragrams of carbon (Tg C) in the HWP model and this manual to align with IPCC reporting requirements.

Chapter 2

Introduction

2.1 General description of the HWP model

An overview schematic of the HWP model is provided in Figure 2.1. For every year of consideration, timber harvest results in board-feet of timber arriving at mills. These harvest values may be subdivided by the land ownership that produced the timber. Timber values are multiplied by the timber product classes, then the primary product ratios for each timber products class, and finally the end-use ratios for each primary products class. These resulting values are transformed to Tg C. Some end-use ratios involve material that is burned for energy capture. Other end-use products become products in use, although the process of creating those results in a percentage of the material ending up as discarded products. Material entering discarded products is proportionally allocated to dumps, landfill, compost, or are recovered (recycled) or burned without energy capture.

In the subsequent year more timber enters the cycle. However, the previous year's carbon in products in use, dumps, recovered products, and the decaying portion of landfills are subjected to a half-life decay. The outputs either end up in the discard category or are emitted to the atmosphere as CO_2.

2.2 Model history

The HWP model began as an Excel macro for the USFS, as described by Stockmann et al. [2012]. A newer version of the tool (v1) was made available in 2012 [Loeffler et al., 2014]. The states of Oregon, California and Washington are signatories to a Memorandum of Understanding for Pacific Coast Temperate Forests (MOU). The MOU partners have a common requirement to generate reliable estimates of the stock and flux of carbon in harvested wood products.

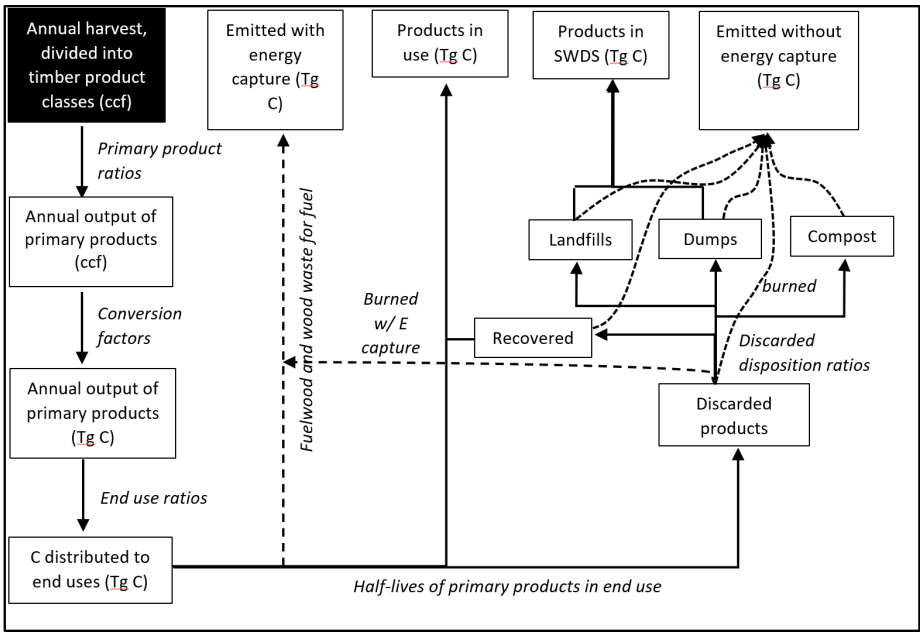


Figure 2.1: General schematic for calculations to quantify HWP storage and emissions, based on @stockmann2012 Figure 7. Note that in this version Compost is not connected to SWDS, an arrow to SWDS has been added for emissions from Recovered, and that the code calculates end-use C distribution in one step after organizing TPO, PPR, EUR, and conversion factors.

A west-coast specific state-level variant of the model was developed in C++; however, it did not fully account for the carbon mass entering the cycle from timber harvest, nor did it generate reliable error estimates using a Monte Carlo simulation approach. Perhaps most importantly, the software was not in a form that could be readily viewed, understood, or changed by others. Representatives from each west-coast state and the USFS agreed that the software needed to be re-written in order to correct these errors.

Groom Analytics LLC undertook programming the Stockmann et al. [2012] model in R for Oregon, and subsequently deployed the R version to a web application (https://groomanalyticsllc.shinyapps.io/HWP_Shared_Test/). This document and the available code are publicly released to enable model transparency and assist in further model development.

The USFS is currently developing a Python-based version of the model to be released soon. A schematic of this history is presented in Figure 2.2.

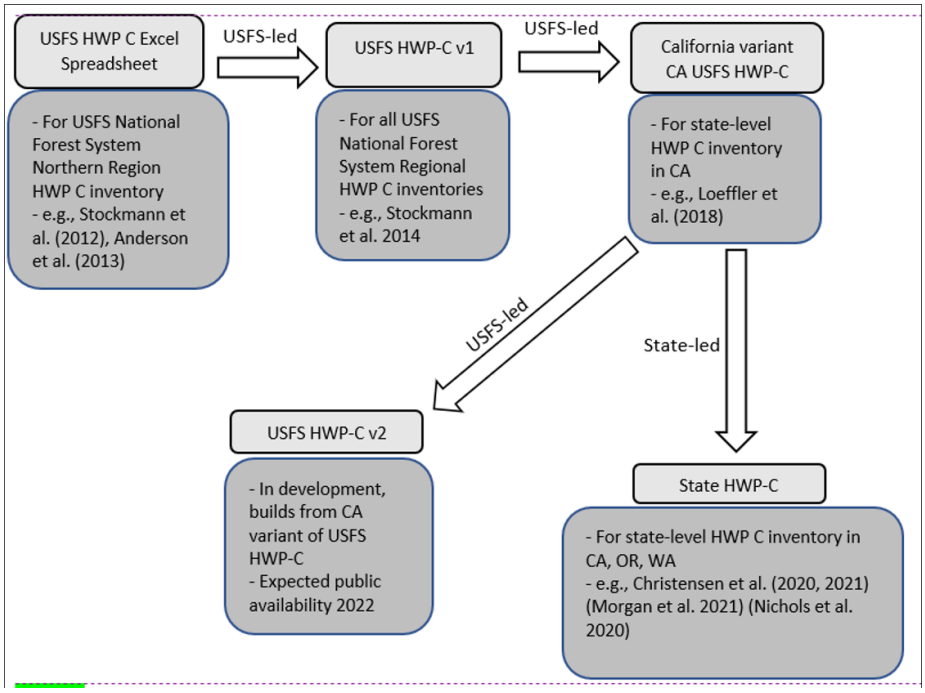


Figure 2.2: Schematic of the HWP model development pathway. The version described here is the State HWP-C variant.

2.3 Model utility

The HWP model (and associated visualizations) is a long-term accounting tool for carbon emissions and storage from harvested wood products. Users can easily understand how harvest levels have changed over time, by ownership and in total. Given changes in disposal methods and the assumed lifetime use of wood products, users can examine what proportion of harvested wood products remain in use, in disposal sites, or as CO_2 emitted to the atmosphere. The model may be useful for planners and decision-makers to understand the emissions consequences of promoting wood products with different use lifespans.

2.4 HWP model assumptions and shortcomings

The HWP model is a closed model, and tracks only carbon harvested within a specified area. Therefore, at a state level, emissions from HWP may be greater than expected from timber harvested within the state if the state imports timber or timber products. Similarly, exported timber or timber products may experience different utilization or decay processes.

The model assumes that harvested timber volume and ratios such as TPR, PPR, and EUR are valid. The model can be expected to perform poorly if the ratios or other values are not based on solid evidence. The Monte Carlo attempts to explore the range of possible outcomes given different levels of certainty in the data. The model does, ultimately, rely on many different data types. Some of these, such as landfill decay rates, may be difficult to enumerate and thus we are unsure of their true values.

The model does not (currently) differentiate between carbon in landfills that is emitted as methane vs. CO_2. Methane is a much more potent greenhouse gas than CO_2, albeit with a shorter residence time.

The Monte Carlo takes a specific approach towards varying the ratios, constraining all ratios to sum to one. Given the selected range of possible values for each variable, the resulting randomly selected values may not be independent from one another and may be unintentionally biased. That is, within a ratio type that must sum to one and for a particular model iteration, if one value is randomly selected to equal 1.0, the remaining values for that ratio set must equal 0.

Finally, we are not aware of any method for overall model validation. Estimates for individual pieces of the model, such as product or decay half-lives, can be improved. But, like all models, the HWP model simplifies reality greatly and hopefully provides users with useful insight by tracking and summarizing complicated calculation outputs in a reasonable way.

2.5 Future work

Users can improve the model's utility by researching and altering model input tables with their own regional information. There is interest in incorporating a means for tracking the production of CO_2 separately from methane as methane is a much more powerful greenhouse gas that results from anoxic decomposition such as occurs in landfills. In the short term we plan to update the model so that users can control the discard ratios for paper and wood products. Finally, work is currently being done at the University of Massachusetts to explore the consequences of errors in estimating the capability of landfills to retain carbon without decay.

Chapter 3

HWP Model Operations

3.1 HWP model function

3.1.1 Model inputs

The HWP model is a deterministic model which relies on optional and required user-supplied information. The user must supply annual timber harvest volumes in board feet (BF) by ownership, annual timber product ratios (TPR) that allocate harvest to different timber product classes, annual primary product ratios (PPR) that allocate timber products to different primary products and residue uses, and annual end-use product ratios (EUR) which are subcategories of PPR that represent the final products produced.

The user may, at a minimum, provide timber harvest volumes and then rely on default California and Oregon values. These include a board foot to cubic foot conversion factors for sets of years and conversion factors between hundred cubic feet (CCF) and metric tons of carbon (MT C) or teragrams of carbon (Tg C) for the primary products. The model uses end-use half-life values to determine how long carbon resides in Products in Use (PIU) before becoming discarded.

The user may also provide ratios for discard fates. These ratios determine, by year, what fraction of discarded paper and wood are channeled into one of six discard categories: burned (Emitted without Energy Capture, EWOEC), burned with energy capture (BWEC), composted, sent to a dump, enter the non-decaying portion of a landfill, or become landfill subject to decay. Users may also adjust half-life values for paper and wood discards in landfills and dumps and the fraction of material entering landfills that is subject to decay or remains permanently.

3.1.2 Summary of model operations

In the next section I provide a detailed schematic of how the HWP model code operates. A more general schematic of the process, very similar to Figure 7 in Stockmann et al. [2012], is presented in the Introduction (Figure 1). Here I provide a written description of the code's actions. We start out with data sets and variable information that are arranged by different combinations of year, ownership, and ratio types. First the model combines (i.e., performs a database join function) harvest data (units = BBF) with board feet to cubic feet conversion factors. The model combines Primary Product Ratios (PPR) with Timber Product Ratios (TPR) and then joins those with the timber harvest volume in cubic feet by year. These are then joined with conversion values for cubic feet to metric tons of carbon, and the End Use Ratios (EUR). The result is a data set called eu_ratios, and values for EUR, TPR, PPR, the harvest amount, and the cubic foot to metric tons of carbon conversion factor are multiplied together to produce a variable called MTC for Metric Tons of Carbon. This variable serves as the carbon input into the model and has the correct units of carbon input per year for each EUR category.

For each year, some of the carbon, fuel wood, is immediately burned and is Emitted with Energy Capture (EEC). A certain percentage (8% for California and Oregon data) of the carbon allocated to each of the remaining EUR categories is discarded and enters the Discarded Products process and the remaining 92% become the annual carbon addition to the Products in Use (PIU) process. The model currently assumes that 100% of pulp wood becomes paper with no loss.

The Products in Use process calculates, for every year, the amount of carbon remaining from the year prior and the amount of carbon entering the process as new PIU. A decay function considers the previous year's carbon amounts in each EUR/ownership category, applies a decay calculation, adds the result to the amount of incoming PIU carbon, and saves the value as the amount of PIU carbon for that year. It also records the amount of carbon that was discarded from the process and sends that carbon to the Discarded Products process.

The Discarded Products process uses proportions to divide incoming discarded products among six fates: burn, burn with energy capture, compost, dump, landfill, and recover (recycle). Burned and composted proportions are treated as EWOEC. The proportion that is burned with energy capture will be treated as EEC (for California and Oregon all carbon is currently modeled as BWOEC). Discarded Products that are recovered or sent to dumps are processed with the same decay function used for Products in Use, but with different half-life values. The outputs that do not decay in a given year are treated either as Products in Use (recovered) or Solid Waste Disposal Site (SWDS) material (dumps). Discarded Products that are sent to Landfill are treated like Dumps, except that a proportion of the landfill is not subject to decay. That amount will become one of the carbon pools that comprise SWDS. The remainder of carbon

in the landfill decays and is treated as EWOEC or remains in the landfill and is counted as another SWDS carbon pool: landfill material subject to decay.

The HWP code's outputs are EEC and combined versions of EWOEC, PIU, and SWDS. The final EEC values are the original EEC values for fuel wood plus the fraction of the discarded materials that are burned with energy capture. The final EWOEC amounts are combinations of emissions from dumps, recovered materials, landfill decay, burnt refuse, and compost materials. PIU is the combination of the original PIU values plus recovered carbon that has not decayed. SWDS is the non-decaying and decaying carbon in landfills plus remaining carbon in dumps.

3.1.3 Model schematic description

Below I provide a schematic of how the HWP model operates. The schematic color codes items and flows from the top to bottom of each page. Inputs, such as worksheets from the Excel data file, are yellow. Calculated matrices or arrays are blue. Final output, which may be manipulated to produce tables and figures, are purple. Calculations or data transformations are peach colored.

The file and object names listed in the schematic may be found in the provided model code. My intent is to provide a written description (above) that assists readers with understanding the schematic, which in turn can be used to interpret how the R code (HWP_Model_Function.R) works. The R code itself is annotated. Readers may alter the code to incorporate their own files or improve upon the model (see Operating and Changing the Model).

Transformations include database functions such as pivots and joins. A pivot in this case usually involves lengthening a matrix by taking several columns of data and stacking them into one column, with a second column identifying which of the original columns they came from. This format is useful for combining data with different levels of information. For instance, we might have data with rows = different ownerships and columns = years. We can pivot those data such that there is a column of ownership, a column for the years (years will be repeated for each block of ownerships), and a column for the values. Say we have another pivoted data set that has data for ownership, years, and Timber Product Ratios. We can join the two data sets by matching years and ownership values and duplicating the data values from the first data set across all of the Timber Product Ratios for each year/ownership combination. A somewhat more complicated version of this example is described in the Model Start page of the schematic.

Ultimately, the code may produce a final pivot length of well over 100,000 rows. This is equal to the number of End Use Ratio values * number of ownerships (including a Total column) * number of years. For instance, the eu_ratio matrix has this many rows and several useful variables that are used to calculate metric tons of carbon from harvest as distributed across EURs, years, and ownerships.

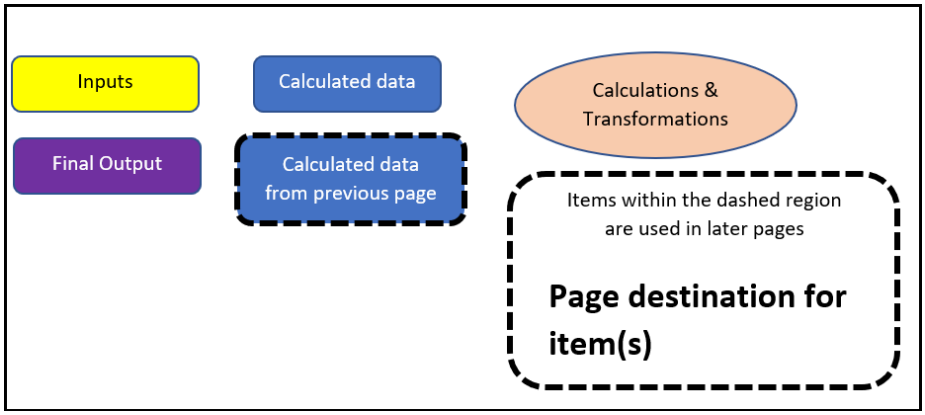
This value is then placed in a three-dimensional array with row number = number of EUR rows, depth = ownership number, and column number = number of years of data. The HWP code frequently uses arrays of these dimensions for storing values from many types of calculated data.

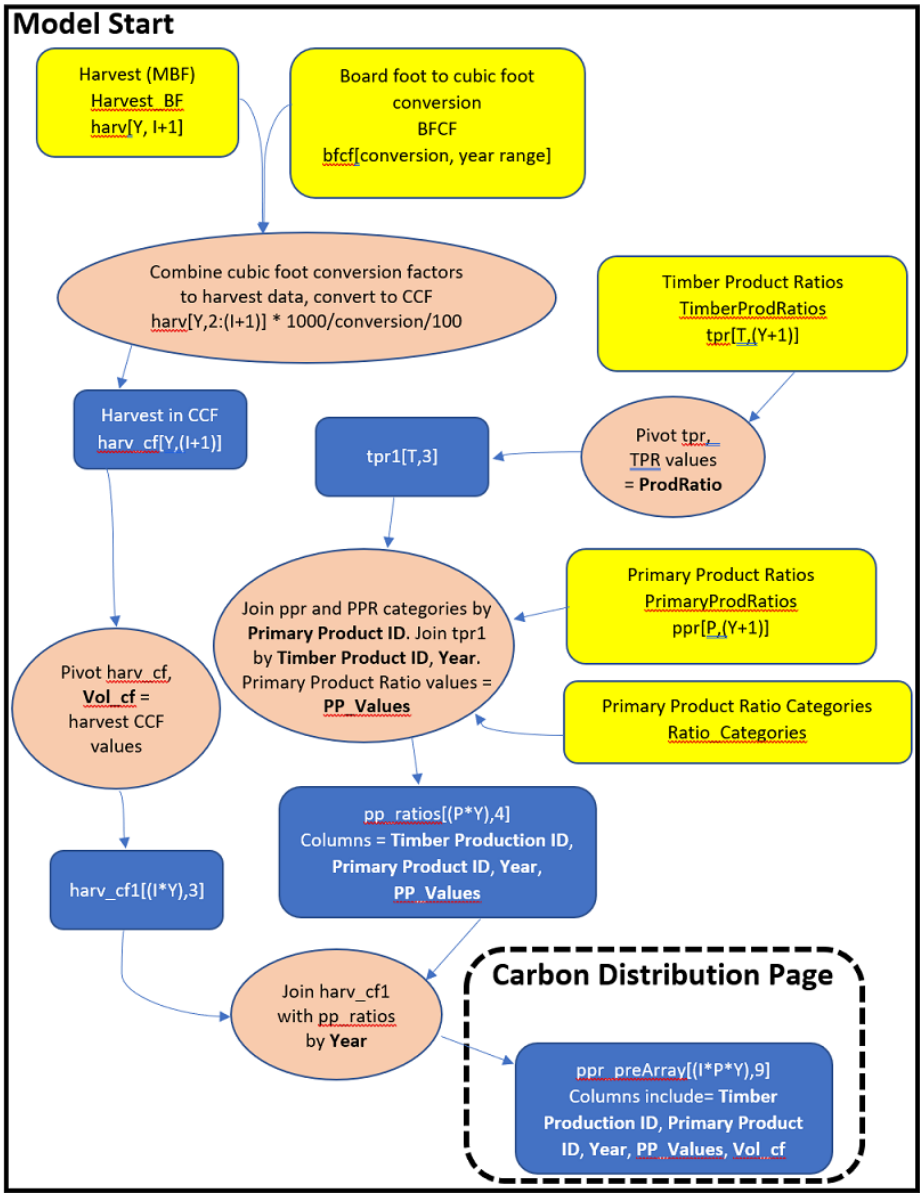
Each page of the following schematic has a title. These titles may be referred to within the schematic, as certain program products are used in later pages. If a calculated data set's boarders have thick black dashes, the data set is from a previous page. If a calculated data set lies within some white space with a thick black dashed boarder, the white space includes information on which page that product is destined for.

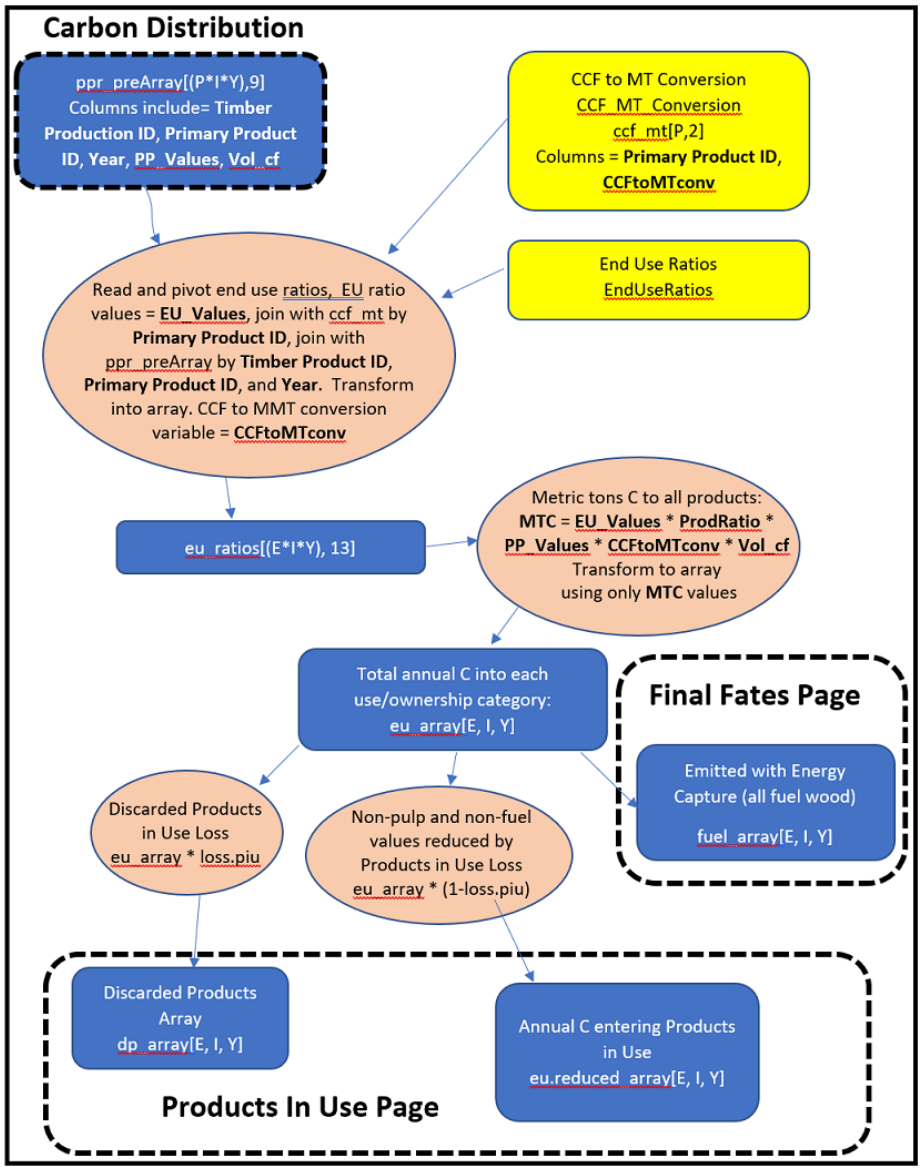
The HWP program has a decay function that is used repeatedly. The same function operates on Products in Use and the decaying portion of landfill, dumps, and recovered (recycled) products. As an example, for PIU, the function iterates through years for a given ownership/EUR combination. It calculates the amount of PIU carbon carried over from a previous year by subjecting that carbon to its appropriate half-life value. Then it adds that carbon to the incoming PIU carbon for that year. Then it moves forward one year and repeats.

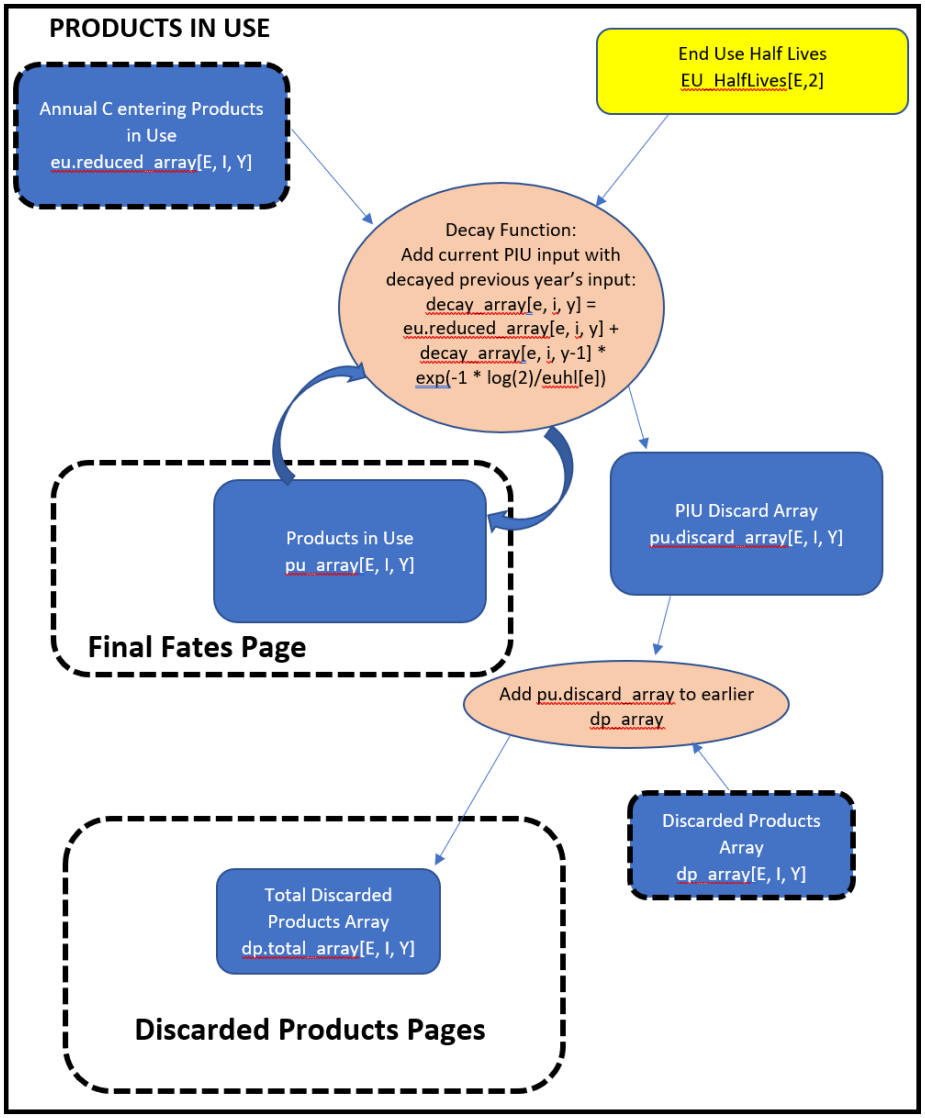
3.1.4 Model Schematic

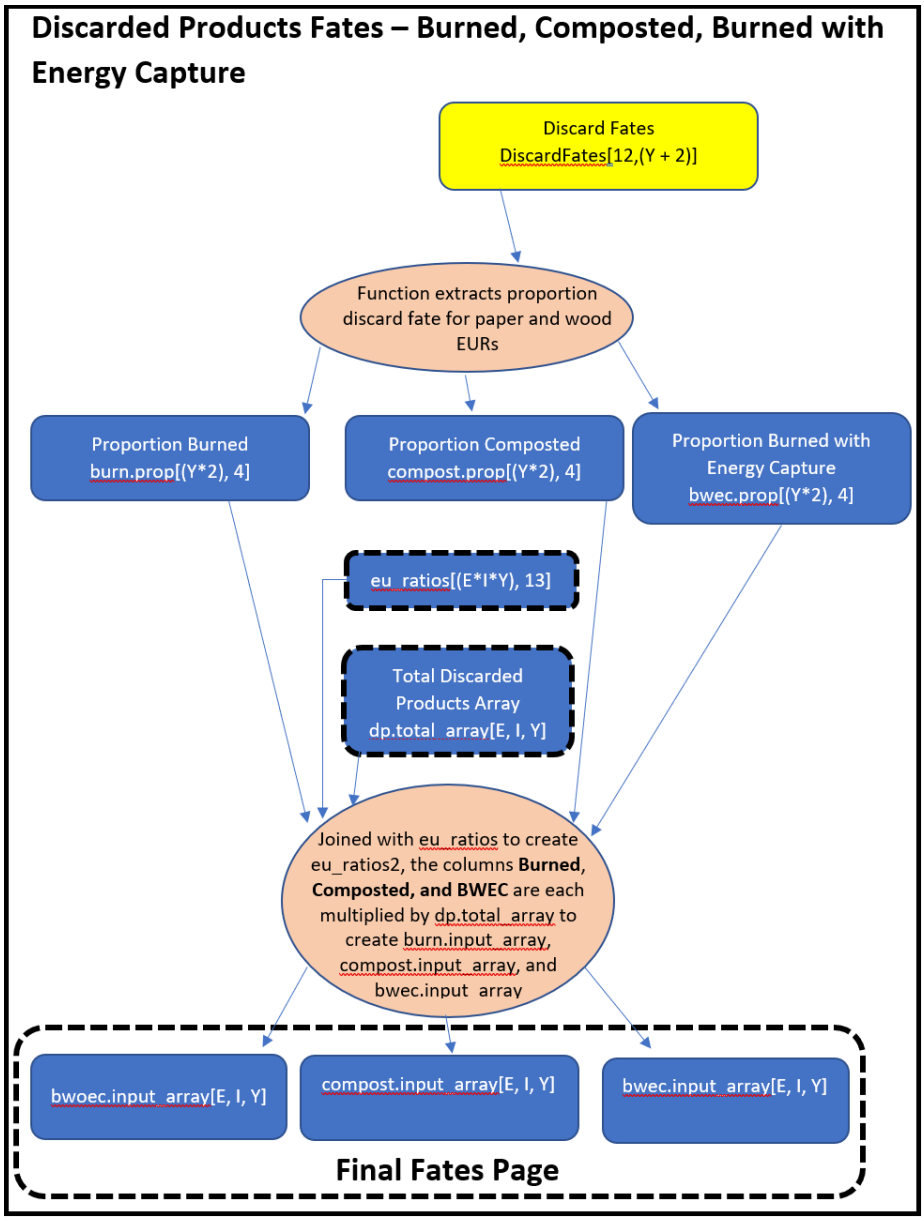
Model Dimensions Key:
Y = number of years (California = 68, from 1952 to 2019)
y = particular year
I = ownerships including Total (California = 6)
i = particular ownership
T = Timber product ratio number (California = 40)
P = Primary product ratio number (California = 64)
E = End use ratio number (California = 224)
e = particular EUR
loss.piu = Loss percentage for products entering PIU (California = 8%)

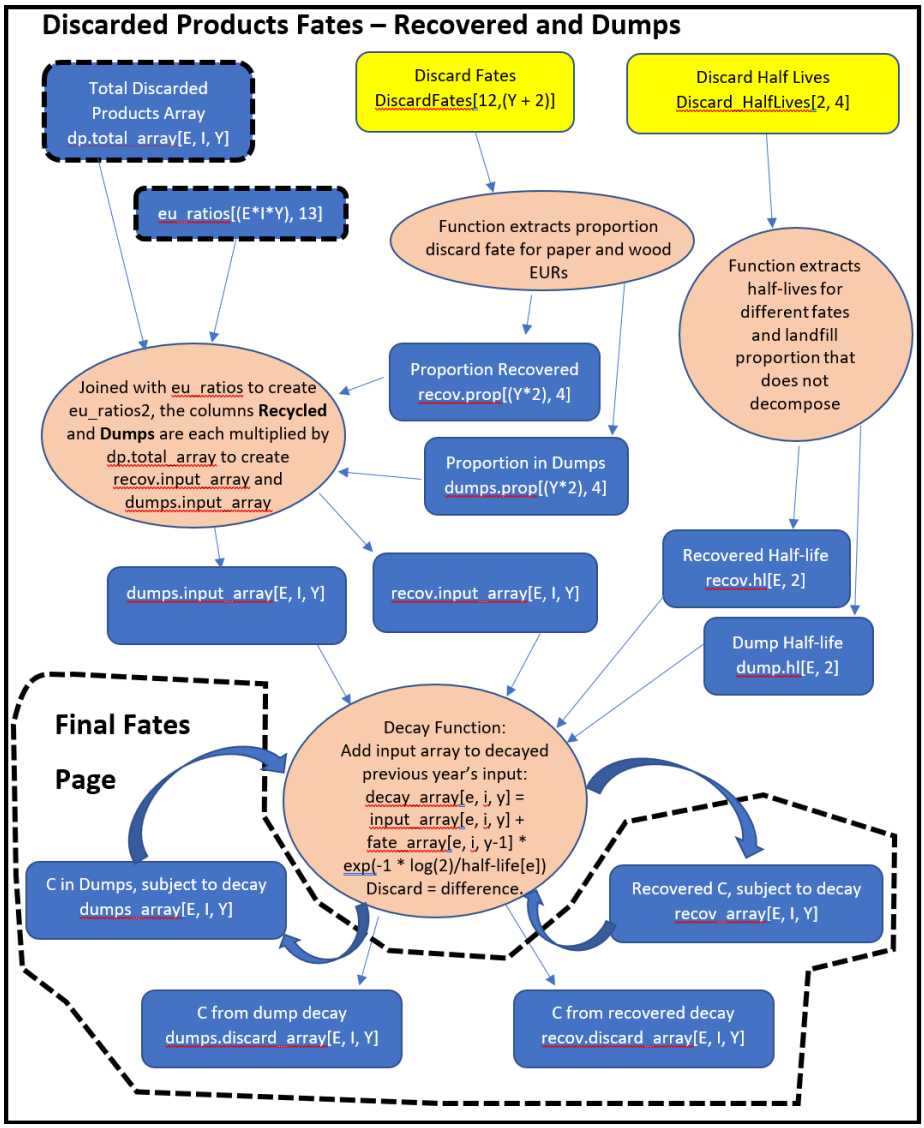


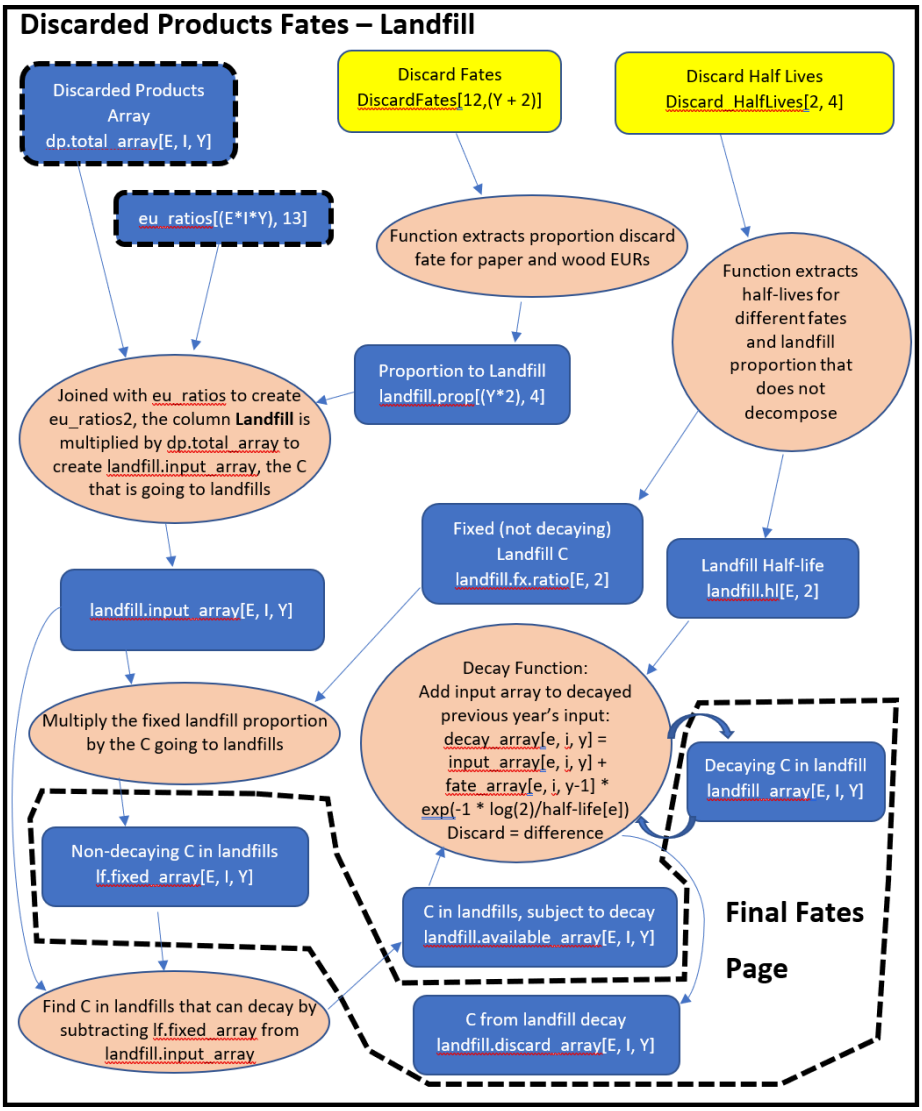

Model Start
Harvest (MBF)
Harvest_BF
harv[Y, I+1]
Board foot to cubic foot conversion
BFCF
bfcf[conversion, year range]
Combine cubic foot conversion factors to harvest data, convert to CCF
harv[Y,2:(I+1)] * 1000/conversion/100
Timber Product Ratios
TimberProdRatios
tpr[T,(Y+1)]
Harvest in CCF
harv_cf[Y,(I+1)]
tpr1[T,3]
Pivot tpr, TPR values = ProdRatio
Join ppr and PPR categories by Primary Product ID. Join tpr1 by Timber Product ID, Year. Primary Product Ratio values = PP_Values
Primary Product Ratios
PrimaryProdRatios
ppr[P,(Y+1)]
Pivot harv_cf, Vol_cf = harvest CCF values
Primary Product Ratio Categories
Ratio_Categories
pp_ratios[(P*Y),4]
Columns = Timber Production ID, Primary Product ID, Year, PP_Values
harv_cf1[(I*Y),3]
Join harv_cf1 with pp_ratios by Year
Carbon Distribution Page
ppr_preArray[(I*P*Y),9]
Columns include= Timber Production ID, Primary Product ID, Year, PP_Values, Vol_cf

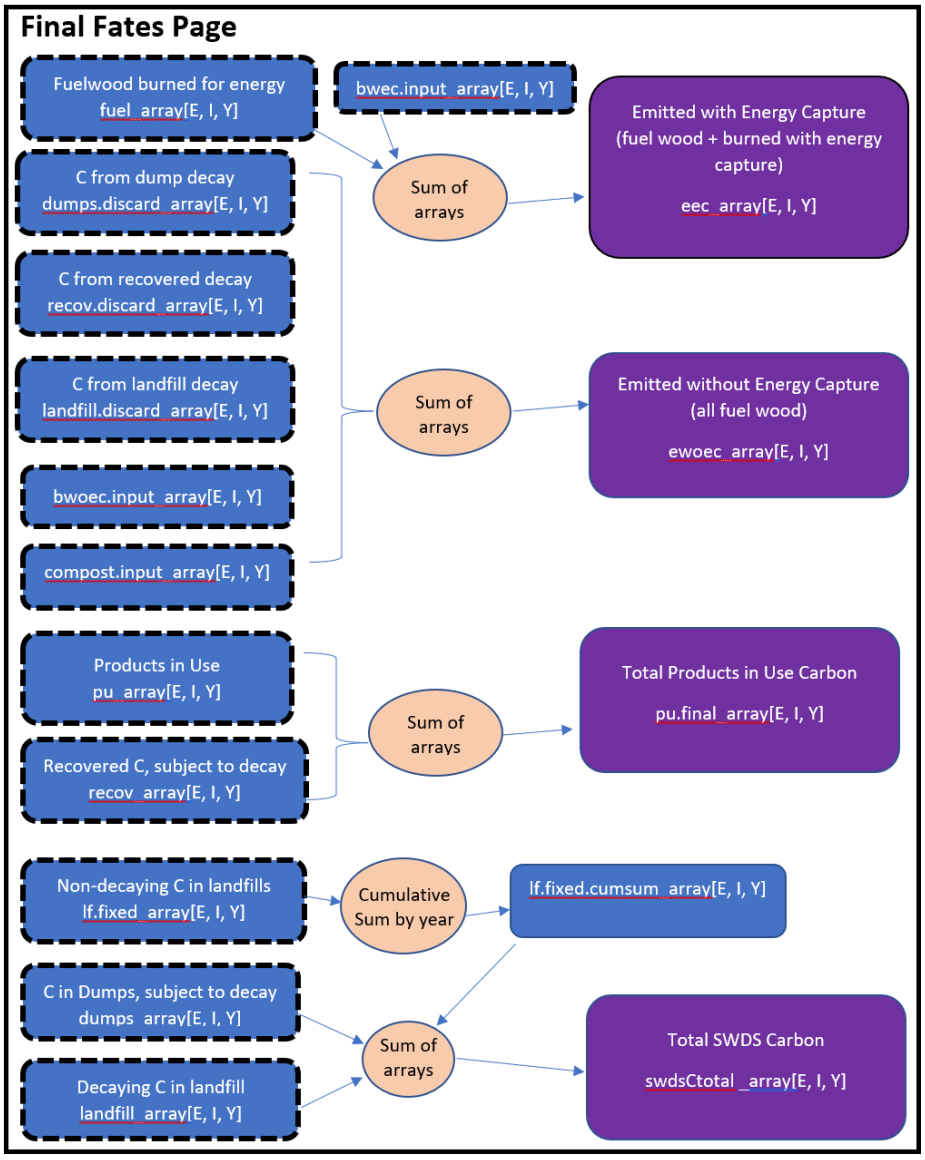

Carbon Distribution
ppr_preArray[(P*I*Y),9]
Columns include= Timber Production ID, Primary Product ID, Year, PP_Values, Vol_cf
CCF to MT Conversion
CCF_MT_Conversion
ccf_mt[P,2]
Columns = Primary Product ID, CCFtoMTconv
End Use Ratios
EndUseRatios
Read and pivot end use ratios, EU ratio values = EU_Values, join with ccf_mt by Primary Product ID, join with ppr_preArray by Timber Product ID, Primary Product ID, and Year. Transform into array. CCF to MMT conversion variable = CCFtoMTconv
eu_ratios[(E*I*Y), 13]
Metric tons C to all products:
MTC = EU_Values * ProdRatio * PP_Values * CCFtoMTconv * Vol_cf
Transform to array using only MTC values
Total annual C into each use/ownership category:
eu_array[E, I, Y]
Final Fates Page
Emitted with Energy Capture (all fuel wood)
fuel_array[E, I, Y]
Discarded Products in Use Loss
eu_array * loss.piu
Non-pulp and non-fuel values reduced by Products in Use Loss
eu_array * (1-loss.piu)
Discarded Products Array
dp_array[E, I, Y]
Annual C entering Products in Use
eu.reduced_array[E, I, Y]
Products In Use Page


PRODUCTS IN USE
Annual C entering Products in Use
eu.reduced_array[E, I, Y]
End Use Half Lives
EU_HalfLives[E,2]
Decay Function:
Add current PIU input with decayed previous year's input:
decay_array[e, i, y] =
eu.reduced_array[e, i, y] +
decay_array[e, i, y-1] *
exp(-1 * log(2)/euhl[e])
Products in Use
pu_array[E, I, Y]
Final Fates Page
PIU Discard Array
pu.discard_array[E, I, Y]
Add pu.discard_array to earlier dp_array
Discarded Products Array
dp_array[E, I, Y]
Total Discarded Products Array
dp.total_array[E, I, Y]
Discarded Products Pages


Discarded Products Fates – Burned, Composted, Burned with Energy Capture
Discard Fates
DiscardFates[12,(Y + 2)]
Function extracts proportion discard fate for paper and wood EURs
Proportion Burned
burn.prop[(Y*2), 4]
Proportion Composted
compost.prop[(Y*2), 4]
Proportion Burned with Energy Capture
bwec.prop[(Y*2), 4]
eu_ratios[(E*I*Y), 13]
Total Discarded Products Array
dp.total_array[E, I, Y]
Joined with eu_ratios to create eu_ratios2, the columns Burned, Composted, and BWEC are each multiplied by dp.total_array to create burn.input_array, compost.input_array, and bwec.input_array
bwoec.input_array[E, I, Y]
compost.input_array[E, I, Y]
bwec.input_array[E, I, Y]
Final Fates Page


Discarded Products Fates – Recovered and Dumps
Total Discarded Products Array dp.total_array[E, I, Y]
Discard Fates DiscardFates[12,(Y + 2)]
Discard Half Lives Discard_HalfLives[2, 4]
eu_ratios[(E*I*Y), 13]
Function extracts proportion discard fate for paper and wood EURs
Function extracts half-lives for different fates and landfill proportion that does not decompose
Proportion Recovered recov.prop[(Y*2), 4]
Joined with eu_ratios to create eu_ratios2, the columns Recycled and Dumps are each multiplied by dp.total_array to create recov.input_array and dumps.input_array
Proportion in Dumps dumps.prop[(Y*2), 4]
Recovered Half-life recov.hl[E, 2]
dumps.input_array[E, I, Y]
recov.input_array[E, I, Y]
Dump Half-life dump.hl[E, 2]
Final Fates Page
Decay Function: Add input array to decayed previous year's input: decay_array[e, i, y] = input_array[e, i, y] + fate_array[e, i, y-1] * exp(-1 * log(2)/half-life[e]) Discard = difference.
C in Dumps, subject to decay dumps_array[E, I, Y]
Recovered C, subject to decay recov_array[E, I, Y]
C from dump decay dumps.discard_array[E, I, Y]
C from recovered decay recov.discard_array[E, I, Y]


Discarded Products Fates – Landfill
Discarded Products Array dp.total_array[E, I, Y]
Discard Fates DiscardFates[12,(Y + 2)]
Discard Half Lives Discard_HalfLives[2, 4]
eu_ratios[(E*I*Y), 13]
Function extracts proportion discard fate for paper and wood EURs
Function extracts half-lives for different fates and landfill proportion that does not decompose
Proportion to Landfill landfill.prop[(Y*2), 4]
Joined with eu_ratios to create eu_ratios2, the column Landfill is multiplied by dp.total_array to create landfill.input_array, the C that is going to landfills
Fixed (not decaying) Landfill C landfill.fx.ratio[E, 2]
Landfill Half-life landfill.hl[E, 2]
landfill.input_array[E, I, Y]
Decay Function: Add input array to decayed previous year's input: decay_array[e, i, y] = input_array[e, i, y] + fate_array[e, i, y-1] * exp(-1 * log(2)/half-life[e]) Discard = difference
Multiply the fixed landfill proportion by the C going to landfills
Decaying C in landfill landfill_array[E, I, Y]
Non-decaying C in landfills lf.fixed_array[E, I, Y]
C in landfills, subject to decay landfill.available_array[E, I, Y]
Final Fates Page
Find C in landfills that can decay by subtracting lf.fixed_array from landfill.input_array
C from landfill decay landfill.discard_array[E, I, Y]



3.1.5 Model Outputs

The model described above treats timber harvest, product manufacture, and disposal as processes that occur within the same year. To match other reports, outputs are altered (SHIFTYEAR = TRUE; see Providing Your Own Data / Creation of the input file). All trees harvested in a given year are treated as processed within that year and the carbon is distributed according to the end use product ratios of that same year. Most of the carbon is immediately burned

or turned into Products in Use. A fraction (the default 8%) is immediately discarded and distributed to the different discard fates using the discarded disposition ratios for the same year. The carbon fate for a given year is reported for the following year. Carbon that enters the model in a given year does not undergo a half-life decay until the following year. For example, the model uses the Timber Production Output for 1906 and all product and discard ratios for 1906 to distribute the carbon in Products in Use, Burned with Energy Capture, and the discard fates. The sum of those fates for 1906 is recorded in tables as occurring by January 1, 1907. Any carbon from 1906 that was distributed to Products in Use, Landfill, or Dumps was subject to a half-life decay or disposal in 1907.

When users upload their data to the Shiny application or run the stand-alone model, they have the opportunity to download or generate summary tables of the model output. The provided tables should allow, with effort, the user to reproduce all Shiny application figures except for those associated with the Sankey diagram.

The stand-alone code can optionally save the arrays used to generate the tables in the folder "Arrays". Unlike the table outputs for stocks and emissions the arrays are not year-shifted. The arrays are provided for results-checking and data exploration. The arrays are three dimensional but are saved as Excel workbooks. Excel worksheet rows represent the end use ratios, columns represent years, and each worksheet tab is an ownership with the final tab representing the "Total" carbon value. Some arrays are combinations of other arrays, described below. File "eu_array.xslx" is the distribution of carbon (metric tons) to end uses. Values are used to convey how harvested wood in each ownership category translates to Tg C for each of the end use products. This array is effectively the starting point for all C fates.

Users are encouraged to generate the arrays (OUTPUT_ARRAYS = TRUE) in the stand-alone model and verify calculations. The above flow diagram provides array names and a general description of how they are used. The code file HWP_Model_Function.r provides more detail about how they are used.

3.2 Monte Carlo simulation

3.2.1 Simulation specifications

The Monte Carlo (MC) simulation alters the values of at least 16 different parameters, listed in Table 3.1. The columns Min Value, Peak Value, and Max Value describe the desired 90% confidence interval for the range of random proportions against which the parameters of interest will be adjusted. In Table 3.1 harvest, timber product ratios and primary product ratios each have two parameters with confidence intervals. These parameters represent different year

Table 3.1: Monte Carlo simulation target parameters and ranges for provided California data.

Parameter ID	Parameter Name	First Year	Last Year	Min Value	Peak Value	Max Value	Sum to One
1	CCF to MgC conversion factors	n/a	n/a	0.95	1	1.05	No
2	End use product half lives	n/a	n/a	0.85	1	1.15	No
3	End use product ratios	n/a	n/a	0.85	1	1.15	Yes
4	Discarded disposition ratios (paper)	n/a	n/a	0.85	1	1.15	Yes
5	Discarded disposition ratios (wood)	n/a	n/a	0.85	1	1.15	Yes
6	Landfill decay limits (paper)	n/a	n/a	0.85	1	1.15	No
7	Landfill decay limits (wood)	n/a	n/a	0.85	1	1.15	No
8	Landfill half lives (paper)	n/a	n/a	0.85	1	1.15	No
9	Landfill half lives (wood)	n/a	n/a	0.85	1	1.15	No
10	Dump half lives (paper)	n/a	n/a	0.85	1	1.15	No
11	Dump half lives (wood)	n/a	n/a	0.85	1	1.15	No
12	Recovered half lives (paper)	n/a	n/a	0.85	1	1.15	No
13	Recovered half lives (wood)	n/a	n/a	0.85	1	1.15	No
14	Harvest	1952	1979	0.80	1	1.20	No
15	Harvest	1980	2100	0.85	1	1.15	No
16	Timber product ratios	1952	1979	0.80	1	1.20	Yes
17	Timber product ratios	1980	2100	0.85	1	1.15	Yes
18	Primary product ratios	1952	1979	0.80	1	1.20	Yes
19	Primary product ratios	1980	2100	0.85	1	1.15	Yes

sets. The Oregon data have three year sets for harvest, timber production, and primary product ratios. Each parameter requires a separate MC random variable. Finally, parameters for timber product ratios and end use product ratios have a sum-to-one characteristic. For instance, timber product ratios (TPRs) represent the proportion of total harvest for a given year that is allocated to each of the timber product ratio values. While the proportions change during the time series the full set of TPR proportions will always sum to one.

3.2.2 Simulation Sampling

As in other HWP publications such as Stockmann et al. [2012] and Anderson et al. [2013], the random variables in the Monte Carlo simulations are drawn from triangular distributions. The distributions all have a peak value of 1.0 (Table 3.1) and symmetrically taper to given 90% confidence interval bounds. The values from the triangular distribution are used as proportions for adjusting parameter values.

Drawing random variables from triangular distributions is a multi-step process. The first step involves drawing random variables from a standard uniform (0, 1) distribution using a Latin Hypercube Sampling (LHS) process. The standard uniform distribution is an excellent starting place for randomly selecting points along other distributions. The random uniform points can be translated as locations along some other distribution's Cumulative Distribution Function (CDF). However, with unorganized, truly random draws of random uniform variables for more than one distribution (we are using at least 16 separate distributions, with one to three distributions per parameter) there is a strong probability that samples will by chance generally fall in the same local region for several of the distributions. In other words, a purely random sample may be inefficient. Many

Table 3.2: Translation of 90% Confidence Intervals to endpoints for triangular distributions.

90% Confidence Interval Range	Endpoints Used
0.7, 1.3	0.57, 1.43
0.8, 1.2	0.71, 1.29
0.85, 1.15	0.78, 1.22
0.95, 1.05	0.93, 1.07

iterations are needed to obtain a full suite of the possible MC values across distributions. LHS offers a method for partitioning the uniform distributions and forcing the random selections across distributions to avoid clumping. As a result, many fewer iterations are needed to achieve stable estimates from the MC simulation.

For this analysis I used the function "randomLHS" in the package "lhs" [Carnell, 2021] to conduct a random LHS sample. When we obtain random draws for a given distribution, the package divides the uniform distribution into as many partitions as there are iterations and then randomly selects a value from within that range. The partitions themselves are randomly ordered.

Once we have a matrix from the LHS (rows = number of iterations, columns = desired number of distributions) we can transform the cell values into draws from triangular distributions Wikipedia: Triangular Distribution.

$$\text{Triangular Random Variable} = \begin{cases} if 0 < U < F(c) & a + \sqrt{U(b-a)(c-a)} \\ else & b - \sqrt{(1-U)(b-a)(c-a)} \end{cases}$$

In this equation U is the random uniform variable (the LHS draw), $F(c) = (c-a)/(b-a), a$ is the minimum value for the triangle, b is the maximum value, and c is the midpoint. Note that the specifications require 90% confidence intervals and do not state what the endpoints for the triangular distributions should be. I derived the endpoints that corresponded with the desired 90% confidence intervals (Table 3.2) using an optimization algorithm.

Each random draw from a triangular distribution was used to adjust an MC parameter (Table 3.1) for all years in a single iteration. The next iteration would use the subsequent random draw. The only exception occurs when the 90% confidence intervals changed within the year set for an iteration. In those instances (i.e., for Harvest, Timber Production Ratios, and Primary Production Ratios) we simulate random draws for three random variables if there were three sets of years identified for each parameter. For each iteration we use the appropriate random triangular variable draw for the given year set (e.g., 1906 to 1945).

There is a final complication: we would like some random variables to be correlated with one another (e.g., Stockmann et al. [2012]). For instance, Harvest has diminishing confidence intervals for annual timber production over time. We have a separate random variable for each of those three sets of years. In a given iteration we can imagine that, had a practitioner developed those values, the intervals would likely have been constructed in similar ways and therefore would have been correlated (biased). Therefore, we force a correlation among the three Harvest variables. Timber Product Ratios and End Use Product Ratios can each have sets of correlated variables and I created pairs of correlated random triangular variables for all paper/wood discard disposition ratios.

I used a Pearson's correlation of 0.5 for all correlations. This was also done by Anderson et al. [2013] and Stockmann et al. [2012]. I used the Pearson's correlation to create a Pearson's correlation matrix, with the number of columns and rows corresponding to the number of random variables to correlate. For a three-variable correlation I would use the following matrix and create a Choleski decomposition from it:

$$\text{Pearson's Correlation Matrix} = \begin{bmatrix} 1 & 0.5 & 0.5 \\ 0.5 & 1 & 0.5 \\ 0.5 & 0.5 & 1 \end{bmatrix}$$

$$\text{Choleski Decomposition} = \begin{bmatrix} 1 & 0.5 & 0.5 \\ 0 & 0.866 & 0.289 \\ 0 & 0 & 0.816 \end{bmatrix}$$

Linear multiplication of the three variables against the Choleski decomposition alters the second and third variables such that they are correlated to the first. I create the correlated triangular random variables by first transforming an iteration's three LHS random uniform variables into values from a standard normal distribution by treating the random uniform variables as probability quantiles from a standard normal distribution. The three standard normal values are then altered and become correlated by linearly multiplying them by the Choleski decomposition. The values are transformed back to a standard uniform distribution by finding the normal cumulative distribution function values of the three normally distributed and correlated random variables. These random variables are then ready to be processed as described above into draws from a triangular distribution.

3.2.3 Sampling and the HWP program

The MC code operates by first running the original HWP program. This creates many values and arrays that the MC simulation relies upon and which we do not want to duplicate with each simulation iteration. I simplified the MC simulation code by moving several functions outside of the code and removing specific

outputs and their precursors that were no longer needed. The goal of the MC simulation was to produce estimates and confidence intervals for overall values for SWDS, PIU, EWOEC, EEC, and their combinations.

Before entering the MC loop, the program performs all LHS draws, forces correlations among specific variables, and transforms the random draws into draws from triangular distributions. Many of the parameter values are then manipulated so that they may be combined into the MC loop's creation of the eu_ratio matrix. Several of the parameters adjusted for each MC iteration affect calculations of the values for the matrix eu_ratios. Specifically, the eu_ratios value for MTC is calculated as follows:

MTC = Harvest * CCF to MTC conversion * TPR * PPR * EUR

This matrix, in the original version of the HWP program, is used to create the eu_array, which distributes all HWP inputs for a given year into PIU, discarded products (EWOEC, SWDS), and EEC. The eu_array object has a dimension for the ownership categories. Since the MC loop only focuses on total values across ownerships (the final category), the eu_array item becomes a two-dimensional matrix with EUR categories for rows and years for the columns.

Below I describe how each parameter is altered.

CCF to MgC Conversion Factors
This is a simple alteration. Within a single iteration, one random triangular variable value is multiplied across the conversion factor CCFtoMTconv values for all years.

Harvest
For Harvest we use one or more correlated random variables for every MC iteration. Each random variable is used to adjust its corresponding range of years. Subsequent iterations use a different draw of correlated random variables. The matrix of random triangular variables, with years = rows, iterations = columns, is multiplied by a vector of the harvest volume per year. For each iteration in the MC loop a different vector of these altered values is used.

Timber Product Ratios
This parameter is more complicated than Harvest because the values within a year must sum to one. It relies on three correlated random variables which are processed as described above. California and Oregon have 40 TPR categories. For each iteration, the code examines all 40 proportion values and determines which one, for a given year, is the maximum value. It finds the product of the random triangular variable value and the maximum proportion. If the resulting value is 1 then the maximum proportion is set to 1.0 and all of the other 39 values are set to 0.0. If it is < 1 then the code calculates the ratio of (1 – new maximum proportion) / (1 – old maximum proportion) and multiplies this proportion against all of the remaining proportions. All 40 altered values should sum to 1.0.

The result is that for each year there are 40 altered TPR values. When the

MTC variable is calculated for each iteration, the altered values are effectively propagated across all PPR and EUR categories per year. This MC sum-to-one approach does have some substantial drawbacks. For instance, Timber Product Ratio 2 is Softwood Sawtimber. It accounts for 77% to 97% of harvest in a given year. Multiplying the triangular distribution value by Timber Product Ratio 2 frequently results in values that are 1.0. Therefore, the alteration of this variable frequently results in the same value, and the distribution of Timber Product Ratio 2 is triangular until the distribution crosses the 1.0 boundary, at which point the remainder of the distribution is stacked at 1.0.

Primary Product Ratios
Three correlated random variables were used to alter PPR for each MC iteration, as was done for TPR. California and Oregon have 64 Primary Product Ratios (PPRs). The 64 PPRs differ from the TPR values in that they sum by year to 40, not 1. Sets of PPR ratios sum to 1; most sets include only one ratio. The PPRs were altered in the same fashion as TPR values with one difference. The code found, for each PPR set with more than one value, the maximum proportion and then adjusted that proportion with the random triangular parameter value. It adjusted the remainder of the values in the PPR set as described for TPR. If there was only one value in the set, it was always equal to 1.0 and the code never adjusted it. These 1.0 values allowed the TPR value to be carried through PPR. Similarly, if the maximum ratio in a set was already 1.0, no alteration occurred.

These values are propagated across all EUR categories per year so that they may be multiplied against other values in the eu_ratio matrix to create the altered MTC values.

End Use Product Ratios
End Use Product ratios were altered as described for PPRs. One difference is that the EUR alterations only used one random triangular variable value for all years in an iteration (i.e., not three correlated variables) because the triangular distribution confidence intervals were held constant across years (Table 3.1). California and Oregon EUR values for a given year summed to 64. The code therefore examined sets of EUR values for each PPR category. The values were combined with other values in the eu_ratio matrix to assist in altering the MTC variable.

Product Half Lives
This was a straightforward alteration. California and Oregon have 224 half-life values, one per EUR. The same random triangular random variable value was multiplied by all 224 values per iteration. Each iteration of the MC loop used a different vector of those 224 values in the decay function that is used to determine PIU over time.

Decay and Half-Life Limits for Landfills, Dumps, and Recovered Pools
Description of these parameters is combined because they relied on the same process. Each one has two correlated random triangular variables, for paper and wood. A function inputs the original 224 ratios for each, where most of the

224 ratios are for wood and a few are for paper. The function also obtains the two vectors of correlated random variables. Each iteration has a single random variable value multiplied by the paper ratio and another for the wood ratio. The altered vectors are then combined into a matrix. When the MC loop runs, each iteration obtains a differently altered vector of 224 values to enter into the decay function.

Discarded Disposition Ratios for Wood and Paper
These ratios control what fraction of waste ends up as burned, burned with energy capture, recycled, composted, placed in a landfill, or sent to a dump (the six fates). The values can differ by year. I wrote a function that operated on the ratios for wood and paper separately. The function performed the sum-to-one approach on the six ratios for either wood or paper. The original HWP analysis process relied on a function for each of these six fates that was time consuming. This function was removed for the MC simulation. Prior to the MC loop I created a matrix that contained altered values for wood and paper across all years and for each iteration for the six fates. The MC loop calls, for each iteration, a different subset of the matrix to use for calculations involving the six fates.

3.2.4 Monte Carlo results and development

It appears that the full version of the MC simulation reaches convergence with 2000 iterations (Figure 3.1). Convergence was assessed visually by examining, over the iterations, the estimated values for the mean, standard error (SE), and the lower and upper empirical 90% confidence interval limits. All four values appear to have reached their stable values by 1500 iterations, as verified with iterations 1501 - 2000. (Note: the calculated SE values are not used beyond this figure.)

The Monte Carlo outputs are summarized in Figures 3.2 and 3.3. In Figure 3.2 we see the mean Tg C value for emissions (Emitted with Energy Capture, Emitted Without Energy Capture) and pools (Products in Use, SWDS) along with the empirical 90% confidence intervals. Figure 3.3 combines Products in Use and SWDS into a single carbon pool. The four carbon fates in Figure 3.2 were obtained by calculating, for each iteration, the cumulative sum of carbon for each fate across all years. Each fate has 2000 MC iterations for each year. In Figure 3.3, it is difficult to tell, but the confidence interval range between 1906 and 1920 are between -37% and + 42% of the mean. In 2017 the range is between -17% and +17% of the mean.

The correlation procedure described above appeared to work. Table 3.3 contains the estimated correlation coefficients for the three triangular distribution random variables used for Primary Products Ratios. The target correlation coefficient was 0.5.

Histograms of the three vectors of Primary Products Ratios triangular random

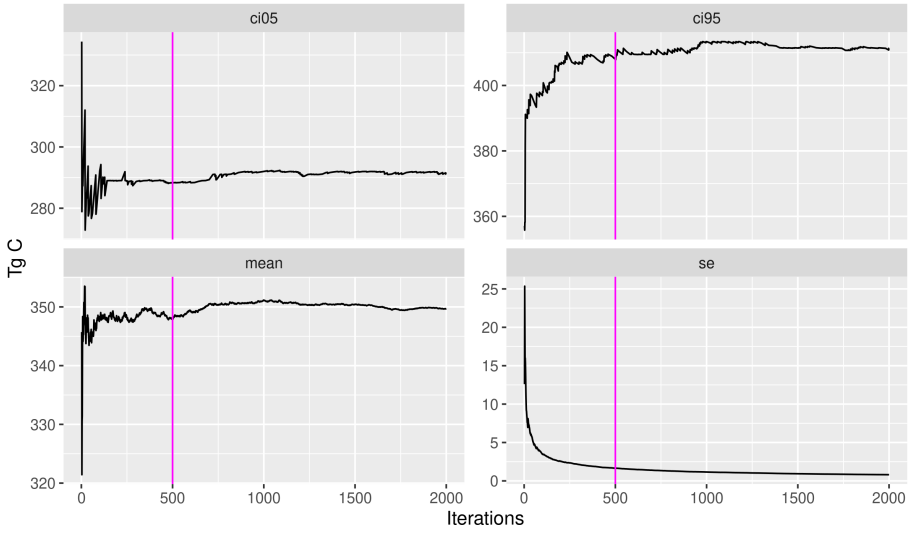


Figure 3.1: Monte Carlo convergence assessment for the upper and lower bounds of the empirical 90% confidence interval, the mean, and the standard error for the full model.

Table 3.3: Correlation matrix for Primary Products Ratios random variables, 2000 draws.

	Variable 1	Variable 2	Variable 3
Variable 1	1.000	0.505	0.508
Variable 2	0.505	1.000	0.521
Variable 3	0.508	0.521	1.000

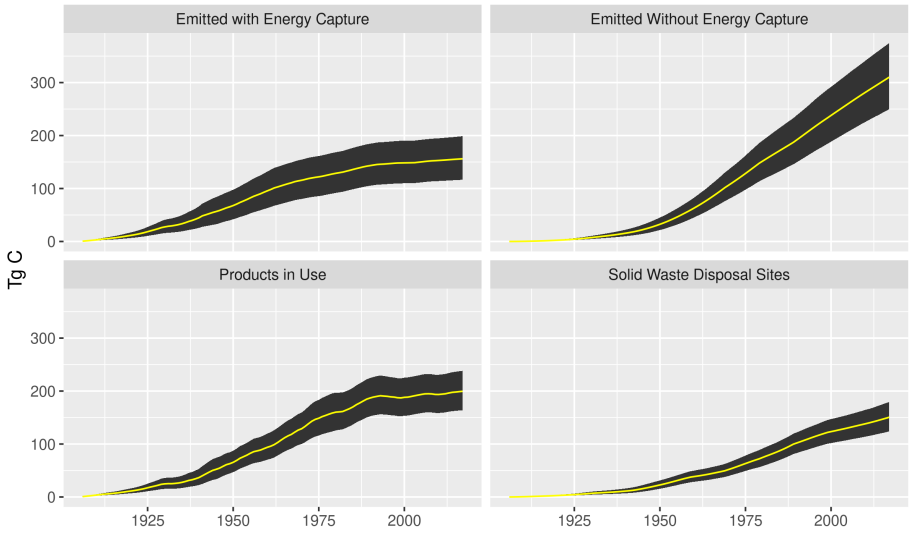


Figure 3.2: Monte Carlo simulation mean (yellow) and 90% confidence interval ranges for four HWP carbon fates across years.

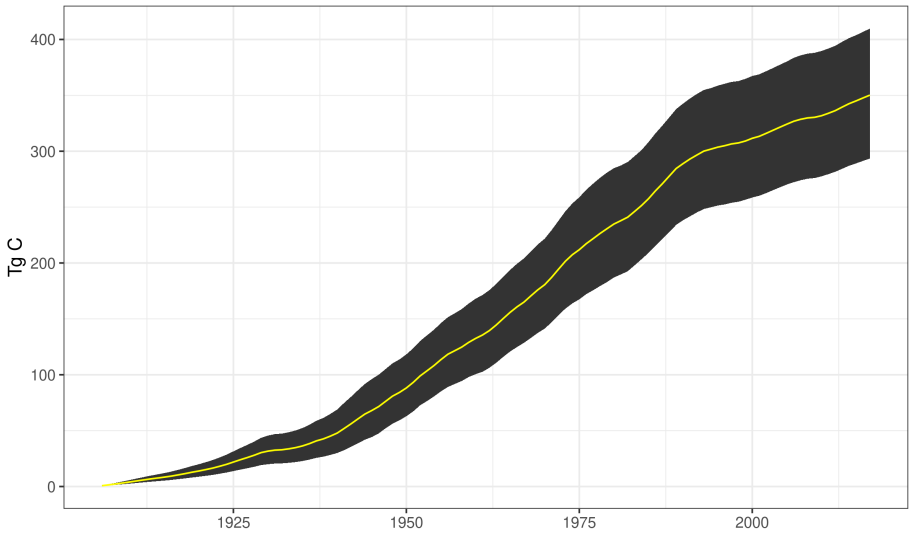


Figure 3.3: Monte Carlo simulation mean (yellow) and 90% confidence interval ranges for the total Tg C value for SWDS and Products in Use.

variables are presented in Figure 3.4. The values are correlated (Table 3.3) and the histogram distributions correspond to 90% confidence intervals and endpoints (see Table 3.2 for triangular distribution endpoints for 0.7-1.3, 0.8-1.2, and 0.85-1.15 90% confidence intervals). Although Variables 2 and 3 do not evidence as tight a triangular distribution as Variable 1, their distributions appear approximately triangular and do not appear skewed.

Figures 3.5 and 3.6 examine triangular distribution changes on sum-to-one ratio sets. In Figure 3.5, PPR.21 is the largest proportion in its sum-to-one group of proportions and was directly adjusted by the random triangular variable draws. PPR.17 was adjusted to ensure that it and the other variables in the group still summed to one. Note that although PPR.21 was the only one directly altered, PPR.17 evidences a triangular distribution as well. The percentage range of PPR.17 values was broader, however, than the percentage range of values for PPR.21.

Figure 3.6 shows the consequences of altering a sum-to-one ratio beyond 1.0, in this case using TPR.2 during the year 2002. The code truncates the value at 1.0, forcing TPR.1 to become zero. The two ratio sets appear to otherwise conform to triangular distributions.

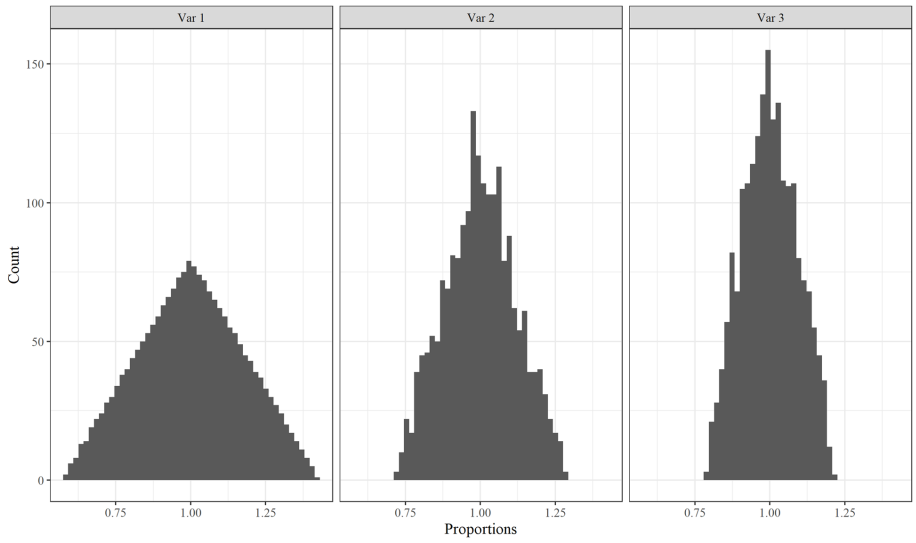


Figure 3.4: Histograms of correlated triangular distribution random variables for Primary Product Ratios.

The MC simulation was developed in pieces. I first altered the values for the non-sum-to-one variables: CCF to MgC conversion factors, annual harvest amounts, product half-lives, landfill decay limits, and the half-lives of decaying paper and wood in landfill, dumps, and recovered products. These alterations should

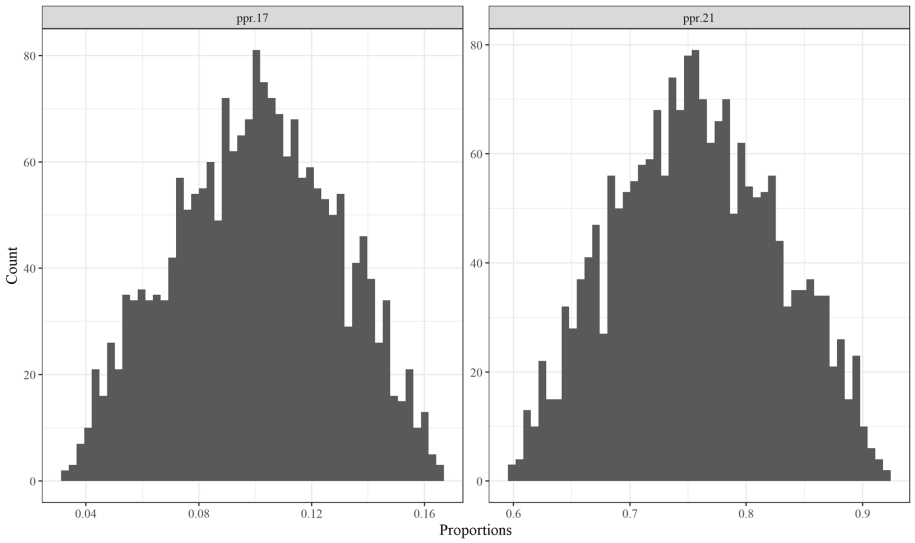


Figure 3.5: Histograms of sum-to-one Primary Product Ratios variables PPR.17 and PPR.21 that do not overlap with 0.0 or 1.0. Both variables are for the year 1994.

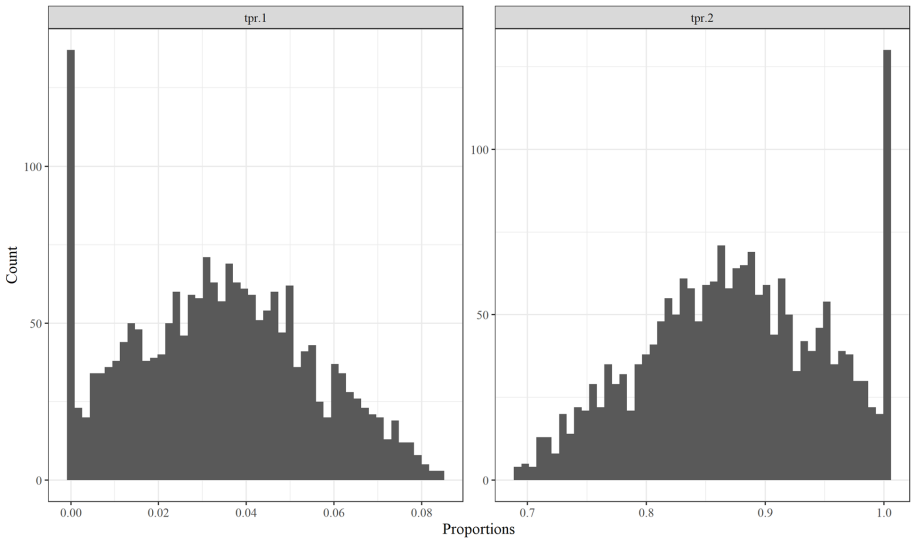


Figure 3.6: Histograms of sum-to-one Timber Product Ratios variables TPR.1 and TPR2; the values for TPR.2 were truncated at 1.0. Both variables are for the year 2002.

affect all categories, since harvest amounts and the conversion factors control all carbon input into the model, product half-lives affect PIU residence and SWDS input, and the SWDS decay limits and half lives affect residency in SWDS.

I next introduced Timber Production Ratios as the first sum-to-one MC alterations, then Primary Production Ratios, then End Use Ratios. Finally, I added the Discarded Disposition Ratios (DDR). These additions did not affect MC results for all carbon fates (Emitted with Energy Capture, Emitted without Energy Capture, Solid Waste Disposal Sites, and Products in Use) equally.

Figure 3.7 provides results for Emitted with Energy Capture. We can see that the MC 90% confidence intervals increase between the first (No Sum-to-One MC Variables) and second (TPR) versions of the model. When I introduced variation into the Timber Production Ratios and then Primary Product Ratios the ratios for fuel wood were altered. Those fuel wood ratios are not altered by changes to EUR, as EUR in this case effectively serves as a "pass-through" category for fuel wood where the ratios remain 1.0 and are not altered by the simulation.

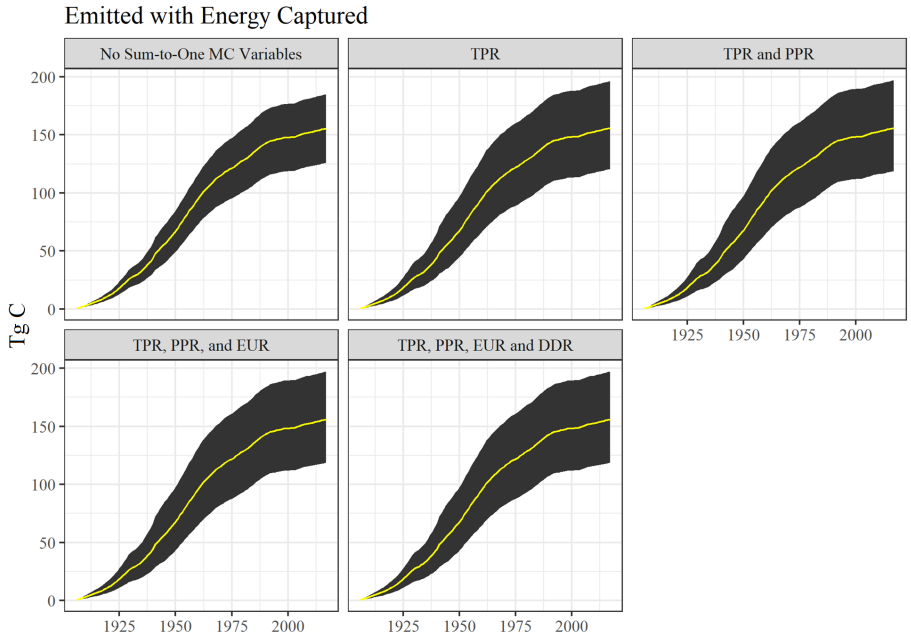


Figure 3.7: Monte Carlo simulation outputs for Emitted with Energy Captured for different Monte Carlo construction stages. See text for acronym definitions.

Figure 3.8, Emitted without Energy Capture, is interesting because graphically there does not appear to be a large change between MC construction iterations. However, the confidence interval ranges do increase across all versions. Each

version differs by between 1 and 4% of the previous version. Therefore, most of the gain in variability was achieved by altering the first set of variables. All other variable alterations did affect the amount of carbon emitted without energy capture, be it affecting the discard dispositions or the End Use Product Ratios (which would in turn be affected by EUR product half-lives and all other variables affecting the next steps of disposal and decay). The effect was not large, however.

Figure 3.9, depicting the MC results for carbon stored in Solid Waste Disposal Sites, really does not evidence an increase in the MC confidence interval until the final iteration when Discarded Disposition Ratios are changed. Changing the DDR affects where disposed carbon winds up. The initial version of the model is already altering carbon entering the system and product half-lives. These are likely the two major drivers of variation for SWDS. Altering the ratios of which products carbon goes to affects things relatively slightly.

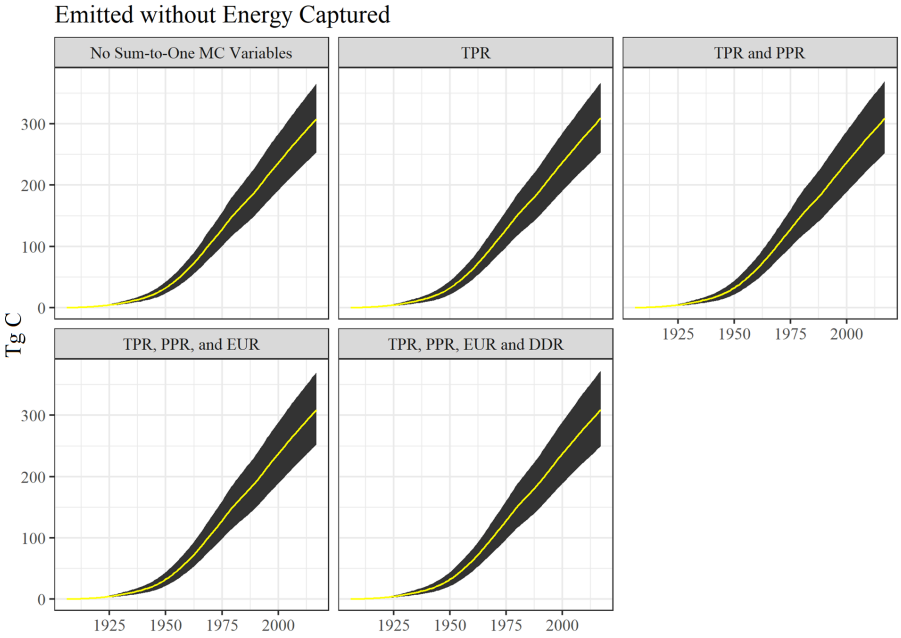


Figure 3.8: Monte Carlo simulation outputs for Emitted without Energy Captured for different Monte Carlo construction stages. See text for acronym definitions.

Products in Use (Figure 3.10) outcomes for the different MC model versions has similar results as EWOEC. Subsequent additions of sum-to-one variables do increase the confidence intervals, by about 1 to 2%. Therefore, adding the sum-to-one variables did not affect the model much. As we might expect, including

the DDR variable changes did not affect PIU confidence intervals at all since those changes only interact with discarded carbon.

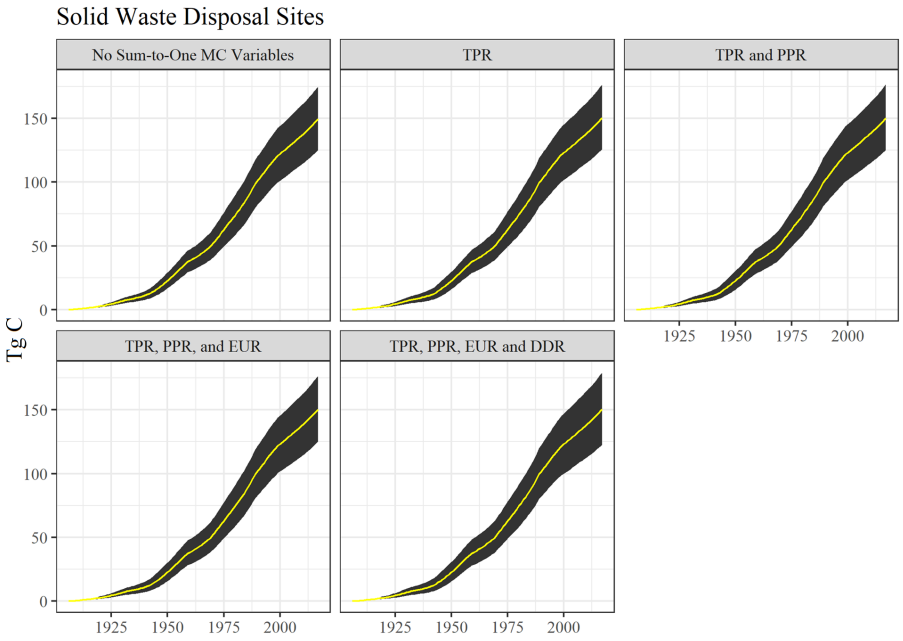


Figure 3.9: Monte Carlo simulation outputs for Solid Waste Disposal Sites for different Monte Carlo construction stages. See text for acronym definitions.

The simulation also did not use different random triangular variable values within a parameter for a given iteration. For instance, the California and Oregon have 224 product decay ratios. In a single iteration they were all multiplied by the same draw value from a triangular distribution. Conceivably, there could be a separate random draw for every product decay ratio, and every other sort of parameter subset. This could result in creating a LHS procedure with hundreds of separate random variables instead of the number used here.

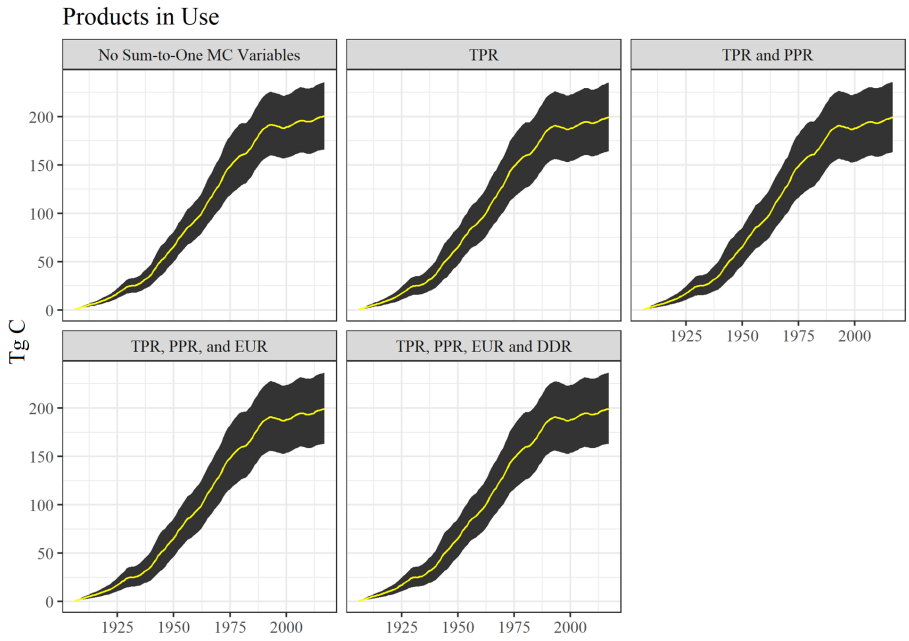


Figure 3.10: Monte Carlo simulation outputs for Products in Use for different Monte Carlo construction stages. See text for acronym definitions.

Chapter 4

Operating and changing the model

4.1 User interaction with the HWP model

There are two approaches available to users for interacting with the HWP model. The first way is to use the web application. Users can view a variety of graphical outputs from the California and Oregon data sets and manipulate the figures to a degree. Users can also upload their own data (see the next section, Providing your own data) and view the same sorts of figures, reflecting their own data. Users can also download the model to their computer. This option allows them to run the web app locally on their own machine. It also lets them operate the stand-alone version of the model, which is a code-based environment that does not produce the web app.

4.2 Basic web interactivity – Shiny app

The web site allows users to switch between two default data sets, California and Oregon state-wide timber harvest. The menu bar on the left has some drop-down plotting and data upload options.

The app was designed to aid users by providing figures that could usefully be incorporated into reports. To that end, all figures can be saved (with one exception) and the title for each figure can be customized.

4.2.1 Timber Harvest Summaries

The three figure options are Annual Timber Harvest (the starting default), Fate of Harvest Carbon, and Utilized Timber Harvest.

- The first option, Annual Timber Harvest, shows the volume of harvest by year and ownership. Users can toggle between seeing just total harvest volumes, ownership volumes, or both simultaneously. As in all figures, users can select seeing results by Tg C or Tg CO_2e. This figure additional allows users to see results by BBF. Results can be displayed by the year or cumulatively over time.

- Fate of Harvest Carbon. This tab has two figures, a Sankey diagram and a more static figure to the lower right. Users can, by mouse, manipulate the location of the vertical bars. Mouse-overs of the figure will also reveal the amount of carbon (Tg C / Tg CO_2e) in pools or transitioning to pools. The figure depicts the fate, over time, of harvested carbon from a single year. The user selects the harvest date of interest and the years post-harvest to track the fate of carbon in the system. The same data are displayed in the lower-right figure, with the figure displaying over time the residence of carbon in different pools and emissions categories. The vertical line depicts the data slice displayed in the Sankey diagram. The Sankey diagram's html code can be downloaded but cannot be saved as a pdf (the stand-alone version of the model can provide a PNG-savable version). The lower-right figure may be saved as a PNG file.

- Utilized Timber Halflives. The HWP model has different half-lives for products in use. This figure bins the annual volume of harvest sent to different end use products by their associated half-lives. The figure displays values by either the harvest volume or proportion of total harvest.

4.2.2 Carbon Storage and Emissions

These four figures track the size of pools and emissions categories over time.

- Carbon Storage by Ownership. If harvest data by ownership are available, the figure plots the amount of carbon stored in products in use and SWDS. Users can select which ownership values are displayed along with whether to display products in use data, SWDS data, or both.

- Carbon Storage and Emissions. This selection allows viewers to access three similar figures of cumulative carbon states over time. All three allow users to view any combination of products in use, SWDS, and emissions categories.

  - Summary Pool and Emission Categories: This figure displays general storage (products in use, SWDS) and emissions (emissions with energy capture, emission without energy capture).
  - Pool and emission category components: This figure allows users to visualize emissions and storage pool components.
  - Pools and emissions by halflife category: Storage and emissions categories are displayed by halflife category.

- Annual Net Change in Carbon Storage. These figures display annual change in stocks (products in use, SWDS; IPCC simple decay approach – stock change) or net balance (previous year's harvest, emissions with and without energy capture; IPCC simple decay approach - net balance). The option to add a line with "net" displays the annual change in products in use plus SWDS. For the "IPCC simple decay approach - net balance" figure, the previous year's harvest value minus the change in emissions values equal the change in products in use plus SWDS.

- Monte Carlo Estimates. Two of these three figures display mean cumulative values from the Monte Carlo simulations with 90% confidence intervals.

  - Cumulative carbon in individual storage and emission pools: There are separate results for Emitted with Energy Capture, Emitted without Energy Capture, Products in Use, and Solid Waste Disposal Sites.

  - Cumulative carbon in storage pools combined (Products in use + SWDS): This figure summed the products in use + SWDS values within each iteration of the Monte Carlo and provides the mean plus 90% confidence interval.

  - Monte Carlo convergence evaluation: This figure displays sequential iteration results for the final year of the Monte Carlo simulation. It shows values for the 5th and 95th percentiles of values (i.e., the boundaries of the 90% confidence intervals), the value for the mean and standard error (se) over time. The Monte Carlo estimation process allows for a burn-in period, for which iteration values prior to that period are discarded from the estimation (pink vertical line). If values after the burn-in period appear unstable, more iterations may be needed to arrive at reasonable values. If they appeared to stabilize long before the burn-in period, then the Monte Carlo may provide reliable results with fewer iterations.

4.2.3 Documentation and Data Upload

These items provide users with links to this documentation, which in turn describes aspects of the web application, the workings of the HWP model, and how users can upload their own data and download the source code.

- Documentation: The link to this document.
- Data templates: Regional templates that users can alter when preparing their own data for entry into the HWP model.
- Upload Data: This selection allows users to load their own Excel file of data, verify that the file is formatted completely, and run the model. Once the model is run, the upper left "Select a data set" option will have a third option; namely, the new data set. All of the Shiny figures (except for the Monte Carlo Estimates selection) will plot the new data accordingly. The user must run the Monte Carlo separately from the page to populate the Monte Carlo figures with their data. The Monte Carlo simulation is run separately because it may take a few minutes to run. For details on preparing and uploading your own data, see the section Providing your own data.

4.3 Operating the model from your own machine

The Shiny [Chang et al., 2021] application and stand-alone implementation of the HWP model are available on GitHub. You will need to have Git installed on your computer, connect Git to RStudio, and finally tell RStudio where the GitHub files are to download (pull). Once you have these files you will have the most up-to-date version of the deployed code.

4.3.1 R, RStudio, GitHub, and downloading the files

You will need to have R and RStudio installed on your computer. Both are free. R is the base program that executes the code; RStudio is an IDE, or Integrated Development Environment, that brings ease and power to using R. You will operate R through RStudio.

You will need to have a program called Git installed on your computer. It may already be installed. There is a bit of a learning curve to getting Git up and going, as well as connecting to GitHub through RStudio. However, there is a great website that can smooth the way: Happy Git and GitHub for the useR. Another excellent website is here. If you are not already using GitHub or similar, getting Git/GitHub/RProject set up may take an hour or two. I

recommend reading through the website carefully and following its instructions. Going slowly is going quickly.

Based on Chapter 16 in Happy Git and Github (specifically 16.2), and assuming your RStudio connects to GitHub, perform the following:
1. Select a location on your computer where you would like a folder containing the code to be placed.
2. Start R studio.
3. Select the following menu options: File > New Project > Version Control > Git
4. In the "repository URL" paste the URL of the HWP model's GitHub repository: https://github.com/jeremygroom/HWP_Shared_Test.git
This will automatically populate the blank below the URL with the project name (HWP_Shared_Test).
5. Click "Create Project"

RStudio will now open the HWP_Shared_Test.Rproj file for you. Within RStudio select from the menu File > Open File… . Select the file app.r and click on the Open button. The next step is to run all of the code in the app.r file. There are several ways to do this. One way is to click on the code itself, select all of the code (ctrl+A on a PC, apple+A on a Mac), and then ctrl+Enter (or apple+Enter) will run everything. Or, go to the menu item and select Code > Source. From within this project, open the R files "app.r" to access and run the Shiny app (Open the "HWP_Stand_Alone_Code.R" to run the model outside of the app.

Users can run one or more lines of code by highlighting them and pressing ctrl-Enter or apple-Enter, or from the RStudio menu selecting Code > Run Selected Line(s).

The first line of the app.r or HWP_Stand_Alone_Code.R is: `renv::restore()`. The first time you run this line the project file will (with your permission) download all versions of all libraries and the R instance used to run the app [Ushey, 2022]. It takes a little while, but you'll only need to do it once. This version control step hopefully helps prevent conflicts between your newer or older versions of previously-installed R libraries and the code.

The stand-alone relies heavily upon functions and code chunks in the Shiny app file (HWP_Shiny_app > R_code_data). Updates to model code affect both the app and stand-alone code. The guts of the HWP model are in the file HWP_Model_Function.r . The file HWP_Model_Prep.R is used in several places for preparing data for entry into the model or versions of the model (HWP, Monte Carlo HWP, Sankey HWP).

4.3.2 File structure

/HWP_Shared_Test/
This folder contains all files associated with the model, including template data,

the Shiny application, the stand-alone code and folders, R environment controls, and the RStudio project that runs it all.

HWP_Shared_Test.Rproj: Open this to operate the Shiny app or stand-alone from your computer.

app.r: This is the main code file that runs the Shiny application.

HWP_Stand_Alone_Code.R: This is the main code file for running the stand-alone HWP model.

Append_Harvest_Data.R: This code will append new harvest data to your existing data file.

README.md: This file may contain some information about the GitHub version.

renv.loc: R environment information.

/HWP_Shared_Test/HWP Data/
This folder contains Oregon, California, and Washington data that may be uploaded into the app.

/HWP_Shared_Test/HWP Data/Templates/
This folder contains sub-folders that direct users to Excel templates that they can use when building their input file.

HWP_Shiny_App/www/
This file only contains the Groom Analytics LLC logo for the application.

HWP_Shiny_App/R_code_data/
Most of the code that the Shiny application and stand-alone HWP model depend upon is in this folder:

- dl_module.R: Shiny module that allows figures to be downloaded.
- global.r: Initial code for reading in data for the Shiny application.
- HWP.CA.output.rds, HWP.OR.output.rds: prepared data files that the Shiny application automatically imports to populate its figures.
- HWP_Model_Function.r: This is the raw code for the HWP model, as is run by the stand-alone and the Shiny application. Note that simplified versions of this code are used for the Sankey diagram (located in PlotFunctions1.r) and the Monte Carlo simulation (see MC_Code.R).
- HWP_Model_Prep.R: Code that extracts values from imported data and prepares them for entry into the HWP model functions.
- HWP_Output_Prep.R: Code that simplifies variable names from HWP model function output.

- HWP_Tables_Code.R: Generates output tables (CSV files) that the user can use to recreate plots. The Shiny application has a button that will appear on the Upload Data selection that allows users to download the tables. The stand-alone code also uses this code to populate its own folder for tables (see below).
- LakePic_MC2.png, ShrubPic3.png: Pretty pictures for populating figure space when user data are not available.
- MC_Code.R: This code runs the Monte Carlo simulation using user data. It is used by the Shiny application and the stand-alone model.
- ModelRunPage.R: This module produces the Upload Data page for the Shiny application.
- Plot modules. Each of these provides the code to run individual Shiny figure tabs:
  - Plot_AnNetChCStor_Module.R: Produces the Annual Net Change in Carbon Storage tab underneath the header Carbon Storage and Emissions.
  - Plot_AnnTimHarvest_Module.R: Produces the Annual Timber Harvest tab underneath the header Timber Harvest Summaries.
  - Plot_CStorEm_Module.R: Produces the Carbon Storage and Emissions tab underneath the header Carbon Storage and Emissions.
  - Plot_CStorOwn_Module.R: Produces the Carbon Storage by Ownership tab underneath the header Carbon Storage and Emissions.
  - Plot_FateHarvC_Module.R: Produces the Fate of Harvest Carbon tab underneath the header Timber Harvest Summaries.
  - Plot_MCest_Module.R: Produces the Monte Carlo Estimates tab underneath the header Carbon Storage and Emissions.
  - Plot_UtTimbHL_Module.R: Produces the Utilized Timber Halflives tab underneath the header Timber Harvest Summaries.
- PlotFunctions1.r: This code contains many of the functions used repeatedly by figures and the HWP model variants (standard HWP, Monte Carlo, and Sankey). It also contains the Sankey HWP model function. This code is source by the Shiny application and the stand-alone model.
- QA_Code_Shiny.r: This code runs a Quality Assurance procedure on imported data. It verifies that the data are formatted correctly for entry into the HWP and Monte Carlo models. See “The QA Step” under “Providing Your Own Data”.

/HWP_Stand_Alone_Files/Arrays/
This folder serves as a destination for the arrays generated by the stand-alone code, given that the user's data file indicates that arrays should be generated. The .gitignore file prevents the source code from uploading the arrays to GitHub.

/HWP_Stand_Alone_Files/QAQC_Reports/
This folder contains the error report (Error_Report.csv) generated by the stand-alone code's call to QA_Code_Shiny.r. The .gitignore file prevents the source code from uploading any error reports in this folder to GitHub.

/HWP_Stand_Alone_Files/Standalone_R_files/
This folder contains two code files, MC_Tables.R and Sankey_Code.r. If the user indicates that the stand-alone code should generate tables of the Monte Carlo outcome, the MC_Tables.R file generates the tables. If the user indicates that the stand-alone code should generate a Sankey figure the Sankey_Code.r file will do so.

/HWP_Stand_Alone_Files/Tables/
This folder serves as a destination for tables generated by the stand-alone code, given that the user's data file indicates that arrays should be generated. The tables are the same as those generated by the Shiny application after the user has run the HWP model on their own data and then selected the button that generates the tables. The .gitignore file prevents the source code from uploading any tables in the folder to GitHub.

/renv/
This folder contains R environment variables that the Shiny application (app.r) and stand-alone model (HWP_Stand_Alone_Code.R) rely upon. The environmental variables include version information for R and the libraries used. The user can ignore this folder.

/rsconnect/
This folder contains shinyapps.io deployment information. The user can ignore this folder.

4.3.3 Running the Shiny app from your computer

As stated above, if this is the first time you are running this code, it may take a little while as renv will need to load appropriate program and package versions. Once you have run the Shiny application code, RStudio will display the application in its own window. A button at the top of the window will allow you to open the application in your own browser, but this isn't necessary. The application should function as it does online when run from the Shinyapps.io server.

The user can alter the Shiny code to produce different figures or to extract intermediary or final data frames used for figure generation. As a warning, if the

user alterations cause an error, the Shiny application can break. Troubleshooting in a Shiny application is not as straightforward as it is in standard R code (In the online book Mastering Shiny by Wickham [2021], see Chapter 5 Section 2, Debugging). I will not discuss the workings of Shiny to any extent here (if interested, see the previous link for more information), but if the user inserts the code `browser()` immediately after an item of interest, saves the change to the code, re-runs the Shiny app from their computer, and selects the figure or item of interest from the application, the code will halt at the `browser()` call and the user can inspect all elements created to that point.

4.3.4 The Stand-alone model: benefits, utility, and how to operate

The stand-alone model provides the user with little additional utility over the web-based Shiny application. It will produce arrays used by the model, and these can be useful for reconstructing analysis results and understanding how the model operates. It can also produce the Sankey diagram, but within RStudio. The RStudio Plots window has an Export option that will save the HTML code as a PNG file, an option that the Shiny application cannot provide. However, the PNG resolution is not much greater than a screen shot of the Sankey from the web application. The PNG version also does not reflect any manipulation of the figure by the user after the figure was produced (e.g., if the user slides the categories around to better separate them, the PNG will show the figure as initially produced).

The main advantage of the stand-alone model is that the user can step through it line by line to better understand what the code is doing. The code is arranged in a much more readable way than it is for the Shiny application. If the user wishes to alter the model code, be aware that changes to the HWP function or MC function may "break" the downloaded Shiny app was well (this is easily remedied via a quick pull request from the GitHub repository – select the "Git" tab in one of the RStudio windows and click the "Pull" button).

The user can step through the code piecewise or run all of the code at once after placing their data file in the HWP Data folder and entering their file name into the code as described below.

To run the stand-alone code, open the file "HWP_Shared_Test.Rproj". This will open the R Project using the program R Studio. Open the file "HWP_Stand_Alone_Code.R". Then, run the `renv::restore()` line plus the library load code. If this is the first time you are running the `renv::restore()` call, the process may take a few minutes to complete.

The next few lines select whether you wish to generate the Sankey diagram, which year you would like to generate it for, and for how many years the decay process should be run. Change `GENERATE.SANKEY = TRUE` to `GENERATE.SANKEY = FALSE` if you do not wish to generate the Sankey diagram. Change the year

in `SANKEY.HARVEST.YEAR` and `SANKEY.YEARS.OF.HARVEST` to your year of interest and the number of years of decay, respectively. It would probably be most efficient to figure out these years/decay times beforehand using the Shiny application.

The `### Folder locations` lines should mostly remain unchanged unless you would like to move files about for some reason. The `IMPORT.DATA.FILE` value should be changed to the file name that you would like to import. Be sure your file is in the HWP Data folder (unless you've redefined the `IMPORT.DATA.FOLDER` location accordingly). Run these lines.

Run the lines for loading the data, including those for the QA test (approximately lines 50 through 85). The code, given your file name, will open your file and extract worksheets. If the user data prescribes the QA test to be run, this code will source the QA_Code_Shiny.r file. Otherwise, it will skip the QA test and load the files automatically. The QA test takes little computation time, so omitting this step does not speed up the code greatly.

Run lines 90 – 111. These lines source code that prepares the data for entry into the model and extracts certain options from the user data, like whether to save tables and arrays, and the locations where they should be saved. The code also sources the PlotFunctions1.r code to obtain some functions used by the HWP model, and the HWP model function itself.

Running lines 112 – 128 causes the code to run the HWP model and extract the model function's outputs.

Running lines 130 – 148 enact a year-shift in the data if the user wishes. The year-shift causes all fates from harvest (waste from processing, placement of carbon in products in use, etc.) to occur the year after harvest.

Running lines 150 – 164 will generate table CSV files in the Tables folder if the user's data defines `OUTPUT_TABLES` to be `TRUE`. Otherwise the code is skipped.

Lines 165 – 200 will generate array Excel files in the Arrays folder if the user's data defines `OUTPUT_ARRAYS` to be `TRUE`. Otherwise the code is skipped. Note that array generation does take some noticeable time; it may be useful to set this option in the data to FALSE when running the overall code set frequently.

Lines 200 – 210 cause the Monte Carlo to run. The Monte Carlo can be very time-consuming to run. Consider setting the data option to `FALSE` if only outputs from the regular HWP model are needed.

Lines 211 – 220 run the Sankey code if the user selected for the generation of the Sankey diagram to take place (`TRUE`).

4.3.5 How to add new data years to your data set

Users may wish to generate model outputs annually or every few years with new harvest values. However, they may not want to alter all of the worksheets in

their data file by hand to account for the newly added harvest years. The file Append_Harvest_Data.R takes, as input, an Excel file with a single worksheet of harvest by board feet, formatted in the same way as their earlier Harvest_BF worksheet. The user points the code toward their original data file, the new Harvest_BF Excel file, and provides a name for the output Excel file. The code will make all necessary changes to the original file and append the new years. Importantly, where needed, the code will read values used for the year prior and duplicate them for the new data. Therefore, if the user wishes to update Timber Product Ratios or End Use Ratios, they will need to do so by hand, possibly after this procedure is run.

As an example, say we wish to add 2018 and 2019 harvest levels to Oregon's data. We open the Append_Harvest_Data.R file in the `HWP_Shared_Test.Rproj` RStudio project, run the first few lines that load the needed libraries, and then alter the following lines to read:

```
ORIG.DATA.FILE <- "Oregon_Inputs_HWP_Model.xlsx"
NEW.DATA <- "Oregon_BF_2019.xlsx"
DATA.FOLDER <- "HWP Data/"
OUT.DATA <- "Oregon_Inputs_HWP_Modelv2.xlsx"
```

The first of those lines refers to the original data, the second the name of the file with the Harvest_BF data for the new years, the third line identifies the location of the input data files, and the last line is the name the user wishes to use for the updated data set.

The new data can provide all of the original data for harvest board feet in addition to the more recent harvest data. Or, the user can provide just the new data. Once those four lines are correct the user can select and run all of the code from the file. The code will attempt to generate a new Excel file with the desired updates. The code also performs some QA checks:

- Verifies that data are numeric
- Verifies that the original and new data do not duplicate or skip years
- If ownership data are present in the original data, the code verifies that ownership data exist in the new data too, and that the Total column equals the sum of other ownership columns.

The code will print error messages if any of those tests are not passed.

If errors are encountered and the files are altered to correct those errors, be sure to close the newly generated file before running the code again. Otherwise, R will generate an error message.

Chapter 5

Providing Your Own Data

5.1 Overview

Users may import their own data into the Shiny application and have the application run the HWP model and Monte Carlo simulation. The following documentation is intended to assist with the construction and explanation of the input file. Once the user has built their own input file, the file can be uploaded to the Shiny application or read by the stand-alone model. In either scenario a QA check verifies that the data are correctly compiled. Once this is done, the application can display the user's data and the user can download tables of the output. The stand-alone model can provide arrays and tables of the output.

The data reside in a single workbook formatted as a MS Excel file with the .xlsx extension. The Excel file has many tabs, described below. Provided examples and templates may be modified by the user when they construct their own import file. The examples and templates are located in the GitHub folder "HWP Data".

There are many values that the user should consider altering based on regional available and historic mill data, dump and landfill decomposition information, etc. At a minimum users can provide yearly values of total harvested billion board feet from their region, alter the rest of the worksheets to reflect their year values, and make use of all other available values from one of the examples or templates.

5.2 Creation of the input file

This explanation may work best if the user opens example files, such as CA_Inputs_HWP_Model.xlsx or Oregon_Inputs_HWP_Model.xlsx, and examines the files while reading the description of each worksheet.

The worksheet HWP_MODEL_OPTIONS allows the user to customize the operation and output of the HWP model. The values in this worksheet affect the operation of the stand-alone model and/or the Shiny application. Below are options that the user may adjust:

- DATASET.NAME (string): This value only affects the Shiny application. This name will appear as an option in the upper-left pulldown menu in the application. Avoid using the names "California" or "Oregon".
- QA_TEST (TRUE, FALSE) – This value only affects the stand-alone model. The default is TRUE. If TRUE, the stand-alone code will examine all worksheets except HWP_MODEL_OPTIONS. See Quality Assurance Step below. If FALSE, the code will skip the quality assurance step and attempt to load the worksheets without examining them. There is little system time needed to conduct the QA test.
- OUTPUT_ARRAYS (TRUE, FALSE) – This value only affects the stand-alone model. If users would like the code to produce the analysis arrays so that they can trouble-shoot model issues or verify calculations, cell B2 should be TRUE (default). Otherwise, FALSE. There is a noticeable time savings in code execution to set to FALSE.
- OUTPUT_TABLES (TRUE, FALSE) – This value only affects the stand-alone model. This option allows users to select whether the code produces output tables for the base HWP analysis. Default = TRUE.
- SHIFTYEAR (TRUE, FALSE) – Affects both the Shiny application and stand-alone model. If the user wants tabular and figure outputs of emissions and pools reported as occurring the year after harvest, cell E2 should be TRUE. If the user wants those outputs reported as occurring during the same year as harvest, select FALSE.
- PIU.LOSS (Double, 0.00 to 1.00) – Affects both the Shiny application and stand-alone model. The current estimated loss of carbon entering PIU through manufacturing is 0.08, or 8%. Enter the default value of 0.08 unless you wish to alter the amount. Cannot exceed 1.0 or fall below 0.
- ARRAYLOC (Text) – This value only affects the stand-alone model. The default value is "HWP_Stand_Alone_Files/Arrays/". If you wish to save arrays to a different folder, change the folder location value. If you are using Windows, be sure to use the forward slash "/" between folder levels and not the back slash.
- TABLELOC (Text) – This value only affects the stand-alone model. The default value is "HWP_Stand_Alone_Files/Tables/". If you wish to save tables to a different folder, change the folder location value. If you are using Windows, be sure to use the forward slash "/" between folder levels and not the back slash.

- RUN.MC (TRUE, FALSE) – This value only affects the Shiny application. If you would like to run the Monte Carlo simulation on your data set, ensure that this value is TRUE. The main model code will run and then verify that this value is TRUE, at which time it will source the Monte Carlo R code and run it. Note: running the Monte Carlo simulation is very time consuming. Set this option to FALSE if you are running the stand-alone model and are not currently interested in the Monte Carlo output.
- R (Numeric, between 0 to 1) – Affects both the Shiny application and stand-alone model. This value is the target Pearson's correlation coefficient that will be used to correlate triangular distribution values in the Monte Carlo. For instance, if the worksheet MonteCarloDistrParameters has three accuracy values for Harvest, depending on the year of harvest, the value for R will determine how correlated the three random draws are that are used to adjust the Harvest values for each iteration. The default is 0.5, as was used in Stockmann et al. [2012].
- OPT.START.VALUE (Numeric, between 0 and 1) – Affects both the Shiny application and stand-alone model. The MonteCarloDistrParameters states minimum and maximum 90% Confidence Intervals from which the Monte Carlo random variables are to be drawn. However, when we transform random uniform values into values from a triangular distribution, we need to know the triangular distribution endpoints. In the Monte Carlo code, the function "ab.boundaries.fcn" performs a search algorithm to find the triangular distribution end points given the confidence intervals provided in the MonteCarloDistrParameters worksheet. The default start value for the algorithm is 0.5 and likely does not need to be altered for the Monte Carlo code to function adequately.
- N.ITER (Integer, 1 or greater) – Affects both the Shiny application and stand-alone model. This value determines the number of iterations the Monte Carlo performs. In the Shiny application, the Monte Carlo Estimates selection "Monte Carlo convergence estimation" produces convergence plots that track estimates for the last year examined in the model. The plot shows the values for the mean, standard error, and 95% and 5% Confidence Intervals given the iterations performed. Of the four, the Mean is most informative. If the line is stable for many iterations (we seem to obtain fairly stable results with 2000 iterations) then you likely have a sufficient number of iterations. The more iterations, the slower the code will run. Too few iterations will provide unreliable results. The default value is currently 100 runs which is likely too few but is sufficient to verify that the process works. 1000 to 2000 might be a recommended number for obtaining reliable estimates and determining what the length of the burn-in period should be.
- BURN.IN (Integer, 0 or greater) - This value only affects the Shiny application. In the same figure mentioned in the previous N.ITER description,

there is a vertical magenta line that signifies the burn-in period. Everything to the left of the line is discarded when results are plotted in the Shiny application. The user should set the BURN.IN value to the number of iterations after which the Monte Carlo appears fairly stable, e.g., the estimate for the mean approximates a horizontal line.

- MC.ARRAY.OUT (TRUE, FALSE) – This value only affects the standalone model. If TRUE, the Monte Carlo code will generate the (large) Microsoft Excel workbook MC_All.xlsx that contains all MMTC values for each year and each iteration of the Monte Carlo code. There is a separate tab for Emitted with Energy Capture (eec), Emitted Without Energy Capture (ewoec), Solid Waste Disposal Sites carbon (swdsC), and Products in Use (pu). The default is TRUE; if it is FALSE the code will run a bit faster and the large (~10 MB?) file will not be produced. The file is provided for model assessment purposes.

The worksheet Harvest_BF contains annual harvest values in billion board feet (BBF). The left column must contain the heading "Year" and have years listed in sequential order from least to greatest. Ownership columns may follow, with a final column entitled "Total" that must equal the ownership values. Users may omit all but the Year and Total columns. Users may have ownership values appear at some point after the first year of total harvest values; however, leave the ownership cells prior to the first year of ownership values blank and not filled with zeros. The QA check sums the values to determine if they equal the total value, so a series of zeros obviously will cause the QA check to fail.

The worksheet BFCF contains the BBF to hundred Cubic Feet (CCF) conversion factors based on time periods. There are only three columns, Conversion, StartYear, and EndYear. Ensure that the first Start Year begins with the minimum Harvest_BF Year and that conversion time periods do not overlap. The Conversion column values are formatted as doubles and StartYear and EndYear values are integers.

The worksheet TimberProdRatios contain the timber product ratios, or TPR. The first column must be TimberProductID. The remainder of the columns are named after the sequential years of harvest data, from the earliest year to the final year. All year-column values must sum to one. Row values may change over time.

The worksheet PrimaryProdRatios contain the primary product ratios, or PPR. The first column must be PrimaryProductID. The remainder of the columns are named after the sequential years of harvest data, from the earliest year to the final year. All year-column values must sum to the number of TPR rows. Row values may change over time.

The worksheet EndUseRatios contains the end use ratios, or EUR. The first column must be EndUseID. The remainder of the columns are named after the sequential years of harvest data, from the earliest year to the final year. All

year-column values must sum to the number of PPR rows. Row values may change over time.

The worksheet RatioCategories acts as a link between TPR, PPR, and EUR categories. The first three column labels, TimberProductID, PrimaryProductID, and EndUseID, must have the same names as the respective first columns of the previous three worksheets described. The fourth through sixth column names are TimberProduct, PrimaryProduct, and EndUseProduct, and contain text describing what the categories contain. Note that EndUseProduct is a subcategory of PrimaryProduct, and PrimaryProduct is a subcategory of TimberProduct. The R code ignores columns 4 through 6; they are there for assisting users in identifying row values.

The worksheet CCF_MT_Conversion contains carbon CCF to metric tons (MT) conversion factors for individual primary products. The first column title must be PrimaryProductID, and the second column CCFtoMTconv. The first column values must be integers and include all primary product ratio categories.

The worksheet EU_HalfLives contains half-life values (in years) for each of the end use categories. The first column must be titled EndUseID, the second EU_HalfLife. The values for EndUseID must be integers and include all end use ratio categories.

The worksheet DiscardFates contains values that apportions discarded carbon to different discard categories: BWEC = Burned with Energy Capture, BWoEC = Burned Without Energy Capture, Recovered, Composted, Landfills, and Dumps. Paper (defined by the code as primary products where the column EndUseProduct in the worksheet RatioCategories contains the word "pulp") and wood products are allowed to have different discard ratio values. The first two columns must be named DiscardType and DiscardDestination. The remainder of the column titles are the years, in order, from longest ago to most recent. The sum of each year-column should be two (all wood and all paper values within a column should sum to one).

The worksheet Discard_HalfLives provides half-life values in years for paper and wood in dumps (Dumps), recovered products (Recovered), and the decaying portion of landfills (Landfills_decay). It also, possibly confusingly, includes values for the proportion of wood and paper that end up in the non-decaying portion of landfills (Landfills_fixed). The columns must be labeled Type, Dumps, Landfills_fixed, Landfills_decay, and Recovered.

The worksheet MonteCarloValues allows users to define how much "noise" to introduce into the HWP model for the Monte Carlo simulation and to specify aspects of the Monte Carlo that they would like to control. The worksheet needs to be altered to conform to the user's data structure for the Monte Carlo to run. That is, if users have data from 1970 – 2018, they will need to change First_Year and Last_Year values for some of the rows. Users can change First_Year, Last_Year, MinCI, MaxCI, and CI values. They may add or delete rows for Harvest, TimberProdRatios, and PrimaryProdRatios. Each

row of the worksheet represents a vector of random draws, with one random draw per iteration. If our model had 100 iterations and this worksheet had 20 rows, the R script will obtain 2000 random uniform draws between zero and one, 100 random values per row. (Please use more than 100 iterations for your final runs.) The R script will change these values to triangular distributions with a peak at 1.0 (column Peak Value) and lower and upper confidence intervals as defined by the columns MinCI and MaxCI. The resulting random numbers for a given parameter (row) will have values that are above or below 1.0. In general, the parameter values associated with each of the parameters are multiplied by the random number for a single iteration. If the random number is 0.86, the parameter value will be reduced to 86% of its original size. Below we describe each of the columns and column values in this worksheet.

- Parameter_ID: This column is included for the user's information; the R script does not rely upon it. Each of the parameters to be altered is listed here. Some parameters (namely, those after Parameter ID number 13) may have more than one set of random numbers generated for it. For instance, if we wish for Harvest values to be modeled as becoming more accurate through time, reflecting improvements in reporting and data collection, we may wish to indicate three periods of years with increasing estimate precision. A series of random numbers will be drawn for each of the time periods, and then correlated (see above). Within an iteration, this example would have three correlated random numbers that would be used for the three periods of years. The parameter ID for Harvest would be the same number repeated three times (14, 14, 14).

- Parameter_Name: The column name, "Parameter Name", is not used by the code. However, the column values are names for specific parameters that are used by the code. These must remain unchanged. The names can be repeated as necessary (following on the example for Parameter ID, Parameter ID 14, 14, 14 would be associated with Parameter Name "Harvest", "Harvest", "Harvest"). Below are descriptions of which parameters and worksheets each relates to:

  - CCFtoMTC: This name translates as "hundred cubic feet converted to metric tons carbon". Each Monte Carlo iteration will alter the conversion values associated with the worksheet CCF_MT_Conversion.
  - EndUse_HalfLives: Each Monte Carlo iteration's random number for this parameter will adjust the end use half life values found in EU_halflives.
  - EndUseRatios: This variable refers to the end use ratio values found in the worksheet EndUseRatios. Each iteration of the Monte Carlo will use one random value to change the end use ratio values. See the section "Sum-To-One Constraint" to understand how the values

are altered.

- DiscardedDispositionRatios, Paper: There are two rows in the Parameter Name column with the name DiscardedDispositionsRatios. The adjacent column, "Paper", indicates whether these disposition ratios are for paper (1) or wood (0). The disposition ratios are from the worksheet DiscardFates. They determine what fraction of paper or wood enter the categories of Burned with Energy Capture, Burned Without Energy Capture, Recovered, Composted, Landfills, or Dumps. Like End Use Ratios, these ratios are subject to the sum-to-one constraint.
- LandfillDecayLimits: The wood and paper values represent the landfill decay limits, or the proportion of landfill material that is subject to decay. The values are obtained from the worksheet Discard_HalfLives.
- Landfill_HalfLives, Dump_HalfLives, Recovered_HalfLives: the six rows with these parameter names are associated with wood and paper decomposition half-lives for materials subject to decay in landfills, dumps, and in recovered products. The values are obtained from the worksheet Discard_HalfLives.
- Harvest: This parameter references the cubic-foot volume amounts harvested each year. The values are originally provided as board feet from the worksheet Harvest_BF, but they are transformed in the original (non-Monte Carlo) portion of the analysis and the transformed values are used in the Monte Carlo. There may be more than one row labeled as Harvest, reflecting different accuracies to be used for different year-ranges.

- TimberProdRatios: Like Harvest, there may be more than one entry for TimberProdRatios. The original worksheet is TimberProdRatios. Like EndUseRatios, a random value associated with a range of years will alter some, but not all, Timber Product Ratio values. See the section "Sum-To-One Constraint".
- PrimaryProdRatios. This type of parameter is altered much the same as TimberProdRatios and the original values may be found in the worksheet PrimaryProdRatios. It too is subject to the "Sum-To-One Constraint".

- Paper: As described above for DiscardedDispositionRatios and other parameters, some parameters represent paper (1) or wood (0) for their associated parameters.

- First Year, Last Year: These are values that users must change. The values in this column represent the first year of a given year period. For

instance, if our data set included harvest from 1906 to 2017 but we wanted to increase the precision of the Monte Carlo values in steps leading to the final year, then we need to indicate what years each level of precision refers to. If only one level of precision is used, provide the first year of harvest (e.g., 1906) under the First Year column and the last year of harvest (2017) to the Last Year column. If there are three Harvest rows, we want the First Year to capture all beginning years for given year periods and Last Year to capture the terminal year. For the first Harvest we might have First and Last Year values of 1906 and 1945, then in the second row the values 1946 and 1979, and finally the third row would contain a First Year of 1980 to 2017. Crucially: the very first First Year must be the first year of your harvest values, whether for Harvest, TimberProdRatios, or PrimaryProdRatios. There must not be overlap in years (e.g., if the first row had 1906 to 1945 and the next row had 1941 to 1955, the sets would overlap). There must not be gaps (e.g., 1906 to 1945, then 1947 to 1955, leaving out 1946). You do not have to have the same sets of years used among parameter types (e.g., Harvest year sets do not need to match TimberProdRatios year sets). You do not need parameter types to have the same number of sets (Harvest could have two rows, PrimaryProdRatios could have three). You may have more or fewer than three year-sets per parameter type.

- MinCI, Peak Value, MaxCI: As described above, these values are used to define the triangular distributions from which the random Monte Carlo values will be generated. For these three columns, Peak Values should all be 1.0, with the MinCI and MaxCI symmetric around this value (the Monte Carlo code assumes symmetric triangular distributions). That is, the MinCI and MaxCI must differ from the Peak Value by the same amount (e.g., MinCI = 0.85, Max CI = 1.15. Both are 0.15 distant from 1.0). The MinCI and MaxCI are the user-specified confidence intervals; e.g., the user states what they wish the confidence intervals should be around 1.0.

- CI: This value allows the user to change what sort of a confidence interval is used. The default value is a 90% confidence interval, as was used in Stockmann et al. [2012]. The user could change this value to 0.95, or a 95% confidence interval. This sort of adjustment will affect the spread of the triangular distribution, with a 90% confidence interval having a wider spread.

5.3 The quality assurance step

The Quality Assurance (QA) step is mandatory for data brought into the Shiny application and optional for the stand-alone program (see above: QA_TEST = TRUE/FALSE). The QA code, QA_Code_Shiny.r, assesses the imported

Excel file data. The QA step has over 55 checks it performs on the incoming data to help ensure that the program will run correctly. As it runs through its checks, it populates the file Error_Report.csv in the QA folder with results. It first verifies that it can load a worksheet and then conducts subsequent tests on it. The code performs tasks such as verifying that column labels are correct and that years and TPR/PPR/EUR categories match among files. If it finds an error, it will record a "1" in the Terminate column that matches the issue at hand and describe the problem in the Comments column. If no problem is found the Comments column will indicate which checks it made and found correct. A zero in the Comments column may be ignored. If an error is found, the R session will terminate (stand-alone model) or an error message will be produced preventing further analysis (Shiny application). This is on purpose! It prevents the code from attempting to proceed. If termination occurs, check the error report to find out which worksheet was triggering the error and why. Below we describe the attributes that are checked for each worksheet:

All 10 worksheets: The QA code verifies that each worksheet correctly loaded. A misspelling of the worksheet name will trigger a terminal error.
Havest_BF:

- The first and last column names must be "Year" and "Total".
- Values are numeric.
- The sum of ownership amounts must equal the sum of the total amounts. These values can deviate from one another if a user saves the CSV file from an Excel file. The decimal places may be lost with numbers rounded. This rounding may cause ownership values to no longer sum to the Total amount. Slight adjustments to the Total amount or an ownership category amount will be necessary to correct.

BFCF:

- The first three column names must be "Conversion", "StartYear", and "EndYear".
- The final year value under Years must equal the last year value of Harvest_BF.
- Values are numeric.

TimberProdRatios:

- The number of TPR years equals the number of harvest years in Harvest_BF.

- The year columns must be transformable to numeric values. E.g., a text value of "1952" can become the integer 1952. A year column name such as "X1952" cannot be transformed into an integer, at least not with the code provided.
- The first column name must be "TimberProductID".
- The year columns must sum to 1.0.
- Table values are numeric.

PrimaryProdRatios:

- The number of PPR years equals the number of harvest years in Harvest_BF.
- The year columns must be transformable to numeric values.
- The first column name must be "PrimaryProductID".
- The year columns must sum to the number of Timber Product Ratios, the number of rows in TimberProductRatios.
- Table values are numeric.

CCF_MT_Conversion:

- The two columns must be named "PrimaryProductID" and "CCFtoMTconv".
- The listed number of Primary Products matches the number in the file PrimaryProdRatios.
- Table values are numeric.

EndUseRatios:

- The number of years for which there are End Use Ratios must match the number of harvest years.
- The year columns must be transformable to numeric values.
- The first column name must be "EndUseID".

- The year columns must sum to the number of Primary Product Ratios, the number of rows in PrimaryProdRatios.
- Table values are numeric.

RatioCategories:

- The column names must be, in order, "TimberProductID", "PrimaryProductID", "EndUseID", "TimberProduct", "PrimaryProduct", "EndUseProduct".
- The word "fuel" must appear in some rows of column "EndUseProduct".
- The word "pulp" must appear in some rows of column "EndUseProduct".
- The number of Timber Product IDs must match the number in TimberProdRatios.
- The number of Primary Product IDs must match the number in PrimaryProdRatios.
- The number of End Use IDs must match the number in EndUseRatios.

EU_halflives:

- The column names must be, in order, "EndUseID" and "EU_HalfLife".
- The number of EndUseID row numbers equals the number of rows in EndUseRatios.
- Table values are numeric.

DiscardFates:

- The first two columns must be named "DiscardType" and "DiscardDestination".
- There must be the same number of years as in Harvest_BF.
- The year column labels must be transformable to numeric values.
- Year column values must be numeric and sum to 2.0.

MonteCarloDistrParameters:

- The column names must be, in order, "Parameter_ID", "Parameter_Name", "Paper", "First_Year", "Last_Year", "MinCI", "Peak_Value", "MaxCI", and "CI".

- The parameter names need to be labeled "CCFtoMTC", "EndUse_HalfLives", "EndUseRatios", "DiscardedDispositionRatios", "LandfillDecayLimits", "Landfill_HalfLives", "Dump_HalfLives", "Recovered_HalfLives", "Harvest", "TimberProdRatios", and "PrimaryProdRatios". Note that some row names can (and should) be duplicated as needed (see above).

- The values for the MinCI and MaxCI must be symmetric around 1.0.

- The parameters with multiple rows and First_Year/Last_Year values must have the years be in order and without overlap or gaps.

- The time series begins on the first year and the last year is within the final time series set.

- Table values aside from column Parameter_Name are numeric.

5.4 Uploading your data and running the model from the Shiny app

Once the user has created their data file, they can open the Shiny app, either online or locally (see Running the Shiny App From Your Computer under section Operating and Changing the Model), and navigate to the tab Upload Data, under the navigation bar item Documentation and Data Upload. The user selects their Excel file by clicking the Browse button and navigating to their file and opening it. The Shiny app will display the name chosen to represent the data set (DATASET.NAME, see above). The user then runs the QA procedure by clicking on the button "Run data file quality control". A button appears that allows the user to "Download the QA output table". The user then clicks on the button "Run HWP Model". If the QA test failed, an error message will appear to that effect. We recommend that the user download the QA output table to determine where the error occurred. If the QA test was passed, the button should instead take a moment to process the data through the model, announce in Model Status that the model run was a success, and change the selected data set under the "Select a data set" option in the sidebar to the user's uploaded data name. At this point the user can visit all figures in the Shiny application and view their data displayed in the figures. They can also download tables summarizing the model run outputs by clicking on the now-appearing button "Download HWP Tables" on the right of the window.

If the user only has Total harvest data (no ownership information) then certain figures will either not attempt to display ownership information or will provide pretty pictures instead. Other figures will be unaffected.

If the user wishes to run the Monte Carlo simulation, then once the HWP model has been run they click the "Run Monte Carlo" button. A busy icon appears, letting the user know that the simulation is in progress. The user is also informed of how long the simulation may take. The Model Status box then has a message appear, "Monte Carlo is complete!". At that point the user may download summary tables for the Monte Carlo. The Monte Carlo Estimates tab under Carbon Storage and Emissions can now display the new simulation results.

Note: If the user tried to view Monte Carlo simulation results after running the HWP model but before actually running the Monte Carlo simulation, they are presented with an image that explains that the Monte Carlo simulation data are not available. If the user then runs the Monte Carlo simulation and returns to the Monte Carlo Estimates, the no-data image will remain until they switch to a previous data set (California or Oregon) and then back to their own data set. At that time their Monte Carlo results should appear. This feature may be improved in the future.

5.5 Using the Stand-alone to run the HWP model and Monte Carlo simulation on your data

See the section "The Stand-alone model: benefits, utility, and how to operate" (Section 4.3.4).

5.6 Generated Tables

Whether using the web version of the Shiny application, the downloaded version, or the downloaded stand-alone model, the user can obtain the same sets of output tables from the HWP model and the Monte Carlo simulation. Previous sections describe how to download or save the tables. This section describes the tables and their contents.

T1.0.Annual_Harvest.csv: This table provides harvest amounts by ownership + Total if ownership data are available, and just Total if not. The data are provided as annual input amounts or as cumulative sums. Finally, the harvest volume is provided in the following metrics: BBF, TgC, and $TgCO_2e$. These data should allow users to generate the Annual Timber Harvest figure from the application.

T2.0.Harvest_Halflives.csv: This table should allow the user to create the Shiny application figure Utilized Timber Halflives. For all years it provides the amount or proportion (pct) of harvested timber destined for fates with different half-lives. Results are given in TgC, $TgCO_2e$, and as a proportion (pct) of the total. Fuel has no half-life as it is burned immediately. Short, Medium, and Long have half-lives of 1-6 years, 7 – 30 years, and 31 or more years, respectively.

T3.0.Cumulative.Ownership.Storage.Emissions.csv: If the user data does not contain ownership information then this table is empty. Otherwise, this table can be used to generate the figure Carbon Storage by Ownership from the Shiny application. The table provides cumulative values for each ownership in TgC. Columns are generated for EEC ("*eec*"), EWOEC ("*ewoec*"), PIU ("*pu*"), and SWDS ("*swds*"). The figure does not use the emissions categories, but they are provided for the user's interest.

T3.5.Cumulative.Ownership.Storage.Emissions.csv: If the user data does not contain ownership information then this table is empty. This table provides the same information as T3.0 but in $TgCO_2e$ instead of TgC.

T4.0.CumulativeStorageEmissions_summary.csv: This table provides the cumulative amount of carbon stored in PIU and SWDS pools as well as the cumulative amount of carbon emitted as EEC or EWOEC. The units are TgC and $TgCO_2e$. These data can be used to reconstruct the Shiny application figure Carbon Storage and Emissions: Summary pool and emissions categories.

T4.5.CumulativeStorageEmissions_summary.csv: This table provides a detailed version of T4.0, again with cumulative values in TgC and $TgCO_2e$. It breaks out the overall PIU into PIU and Recovered, SWDS into a non-decaying portion of the landfill (Landfill_fixed), a decaying portion of the landfill (Landfill_available), and dumps. For emissions we have burned without energy capture (Fuelwood), emissions from dumps, landfills, and recovered products (DumpsEmit, LandfillEmit, RecoveredEmit), compost, and burned waste with and without energy capture (BurnEnergyCapture, BurnNoCapture). These data can be used to reconstruct the Shiny application figure Carbon Storage and Emissions: Pool and emission category components.

T4.8.CumulativeStorageEmissions_halflives.csv: These data can be used to reconstruct the Shiny application figure Carbon Storage and Emissions: Pools and emissions by halflife category. Aside from the year column, columns in this table depict the amount of carbon (TgC, $TgCO_2e$) that resides in pools or has been cumulatively emitted from sources with different halflives. SWDS (swds), PIU (pu), and EWOEC (E) are broken into their halflife category components, short, medium, and long (st, md, lng). Since EEC (eec) represents fuel wood that was burned immediately, it has no halflife components.

T5.0.AnnualStorageEmissionsChange.csv: These data can be used to reconstruct the Shiny application figure Annual Net Change in Carbon Storage. Values are in TgC and $TgCO_2e$. We subtracted the value of one year's SWDS, PIU, EWOEC, and EEC values from the previous to determine what the annual

change was. NetStockChange is the sum of the stock, or pool, values of SWDS and PIU. The figure for "IPCC production approach – stock change" relies on SWDSchange, PIUchange, and NetStockChange values. The figure "IPCC simple decay approach = net balance" relies on the columns Harvest/Harvest_CO2, EECchange, EWOECchange, and NetStockChange. NetStockChange is the same line as appeared in the "IPCC production approach – stock change" figure. The column Harvest is in TgC and Harvest_CO2 is in TgCO$_2e. The value given for harvest is the amount of harvest the year prior to the one listed under column Year. The harvest column also equals the sum of the changes in SWDS, PIU, EWOEC, and EEC.

MC_ComponentsSummary.csv: This table provides the information necessary to produce the figure Monte Carlo Estimates: Cumulative carbon in individual storage and emissions pools. There are values for each year and pool/emission type (Column: Type.M) such as eec (Emitted with Energy Capture), ewoec (Emitted Without Energy Capture), swdsC (Solid Waste Disposal Sites) and pu (Products in Use). The figure is constructed using Means (yellow line), lci, and uci (lower and upper 90% confidence intervals). The Means are calculated by finding the mean value for an emission or pool type among iteration values for a given year. The columns lci and uci represent lower and upper 90% confidence intervals, derived directly from the distribution of iteration values for the pool or emissions type from a given years. The two columns pct_lci and pct_uci represent a percentage of the mean for the lower and upper confidence intervals. Pct_lci = lci / Means; pct_uci = uci / Means. The values in this table are cumulative MTC; the Shiny application transforms them to TgC and $TgCO_2e$.

MC_PIU_Plus_SWDS.csv: This table provides the information necessary to produce the figure Monte Carlo Estimates: Cumulative carbon in storage pools combined (Products in Use + SWDS). The values units are cumulative MTC which the Shiny application transforms to TgC and $TgCO_2e$. The column Mean represents the mean value of SWDS plus PIU values for a given year across iterations. The columns lci and uci represent lower and upper 90% confidence intervals, derived directly from the distribution of iteration values for given years.

MC_All.xlsx: This file is only produced by the stand-alone code if the user sets Run.MC = TRUE and MC.ARRAY.OUT = TRUE. The file has four worksheets: eec, ewoec, swdsC, and pu, representing EEC, EWOEC, SWDS, and PIU. Each column is an iteration while each row is a year. The data are in MTC. The figure "Monte Carlo Estimates: Monte Carlo convergence evaluation" is created using the last year (last row) of the swdsC and pu worksheets. The "mean" value is the successive means of the SWDS + PIU iteration values summed, starting with only the first column and adding columns to consideration (represented by the Iterations axis at the bottom of the figure). The se, ci05, and ci95 columns are the estimated standard error and empirical 5% and 95% confidence intervals based on the number of iterations being considered. If the user specifies 2000 iterations over 100 years of harvest data the resulting

Excel file can be tens of megabytes in size.

5.7 What to know about running the Monte Carlo simulation

As described above, there are several controls in the worksheet HWP_MODEL_OPTIONS. The user can set the number of iterations they would like the Monte Carlo simulation to run, and set the burn-in period length. The burn-in period is the number of initial iterations that are not considered when calculating means and confidence intervals. You can also set the parameters R and OPT.START if you wish.

If you would like to see the Monte Carlo array, which includes all estimates from all iterations, you will need to run the stand-alone version of the model and set MC.ARRAY.OUT to TRUE.

If you are trying to get the model to run or are playing around with data loading and execution on the Shiny App or stand-alone, I recommend setting the number of iterations to something small like 100. You may find that the Monte Carlo estimates really aren't that far off of from the standard deterministic HWP model run. A small number of iterations will prevent you from waiting for several minutes while the simulation runs. Running the simulations for 1000 iterations may demonstrate a reasonable burn-in number to enter into the input Excel file. When you are ready to obtain a "final" estimate you can increase the number of iterations to 1500, 2000, or more.

If you have ratio values that are close to boundaries (e.g., 0.03 and 0.98), you may notice that confidence intervals do not appear symmetrical around your mean estimate. Please see the Results and Development section under Model Function. The Monte Carlo simulation is encountering a "sum-to-one" boundary constraint that is frequently forcing one ratio value to its boundary (e.g., 1.0) and subsequently setting all related ratios to their opposite boundaries (e.g., 0.0). This can result in a less continuous range of outputs, affecting the distribution of results.

Chapter 6

Blocks

6.1 Equations

Here is an equation.

$$f(k) = \binom{n}{k} p^k (1-p)^{n-k} \tag{6.1}$$

You may refer to using `\@ref(eq:binom)`, like see Equation (6.1).

6.2 Theorems and proofs

Labeled theorems can be referenced in text using `\@ref(thm:tri)`, for example, check out this smart theorem 6.1.

Theorem 6.1. *For a right triangle, if c denotes the* length *of the hypotenuse and a and b denote the lengths of the* ***other*** *two sides, we have*

$$a^2 + b^2 = c^2$$

Read more here https://bookdown.org/yihui/bookdown/markdown-extensions-by-bookdown.html.

6.3 Callout blocks

The R Markdown Cookbook provides more help on how to use custom blocks to design your own callouts: https://bookdown.org/yihui/rmarkdown-cookbook/custom-blocks.html

Chapter 7

Sharing your book

7.1 Publishing

HTML books can be published online, see: https://bookdown.org/yihui/bookdown/publishing.html

7.2 404 pages

By default, users will be directed to a 404 page if they try to access a webpage that cannot be found. If you'd like to customize your 404 page instead of using the default, you may add either a `_404.Rmd` or `_404.md` file to your project root and use code and/or Markdown syntax.

7.3 Metadata for sharing

Bookdown HTML books will provide HTML metadata for social sharing on platforms like Twitter, Facebook, and LinkedIn, using information you provide in the `index.Rmd` YAML. To setup, set the `url` for your book and the path to your `cover-image` file. Your book's `title` and `description` are also used.

This `gitbook` uses the same social sharing data across all chapters in your book- all links shared will look the same.

Specify your book's source repository on GitHub using the `edit` key under the configuration options in the `_output.yml` file, which allows users to suggest an edit by linking to a chapter's source file.

Read more about the features of this output format here:

https://pkgs.rstudio.com/bookdown/reference/gitbook.html

Or use:

```
?bookdown::gitbook
```

Bibliography

Nathaniel Anderson, Jesse Young, Keith Stockmann, Kenneth Skog, Sean Healey, Daniel Loeffler, J Greg Jones, and James Morrison. Regional and forest-level estimates of carbon stored in harvested wood products from the united states forest service northern region, 1906-2010. *Gen. Tech. Rep. RMRS-GTR-311. Fort Collins, CO: US Department of Agriculture, Forest Service, Rocky Mountain Research Station. 114 p.*, 311, 2013.

Rob Carnell. *lhs: Latin Hypercube Samples*, 2021. URL https://github.com/bertcarnell/lhs. R package version 1.1.3.

Winston Chang, Joe Cheng, JJ Allaire, Carson Sievert, Barret Schloerke, Yihui Xie, Jeff Allen, Jonathan McPherson, Alan Dipert, and Barbara Borges. *shiny: Web Application Framework for R*, 2021. URL https://shiny.rstudio.com/. R package version 1.7.1.

Dan Loeffler, Nathaniel Anderson, Keith Stockmann, Ken Skog, Sean Healey, J Greg Jones, James Morrison, and Jesse Young. Estimates of carbon stored in harvested wood products from united states forest service southern region, 1911-2012. *Unpublished report. Missoula, MT: US Department of Agriculture, Forest Service, Rocky Mountain Research Station, Forestry Sciences Laboratory. 27 p.*, 2014.

R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2021. URL https://www.R-project.org/.

Keith D Stockmann, Nathaniel M Anderson, Kenneth E Skog, Sean P Healey, Dan R Loeffler, Greg Jones, and James F Morrison. Estimates of carbon stored in harvested wood products from the united states forest service northern region, 1906-2010. *Carbon Balance and Management*, 7(1):1–16, 2012.

Kevin Ushey. *renv: Project Environments*, 2022. URL https://rstudio.github.io/renv/. R package version 0.15.2.

Hadley Wickham. *Mastering shiny*. " O'Reilly Media, Inc.", 2021.