Project Walk-through (step-by-step) for the Central Washington Initiative (CWI) "demo" project cwi_demo

Prepared by: Cat Ducat and Jeremy Fried, 16 December 2024

Purpose: A step-by-step demonstration of the process used for creating the cwi_demo project to promote learning the BioSum workflow

This document: cwi_demo_step-by-step.pdf

Companion documents and data: cwi_demo_overview.pdf, cwi_demo_project.7zip

This project walk-through documentation of the cwi_demo project supports new BioSum users in learning the workflow of a BioSum analysis. It does not replace the BioSum User Guide, which more fully describes the workflow and options available in BioSum, or the CWI_demo project overview, which documents the specific assumptions for this project. Other than a suggested exercise of adding a third active management prescription, this project is already complete, with results tables included in the optimizer scenario results databases. Learning may be enhanced by following these steps to build your own version of the project in a different file space, relying on the provided CWI_demo project at any step in the workflow as a comparative reference to identify potential missteps.

Project Setup

Supplied Files:

1. BioSum Keywords (.kcp [keyword control program] file)
    a. Specifies FVS outputs (tables) to produce, parameters influencing the calculations that go into those tables, and the timing of the FVS projection "cycles"
2. EC_FOFEM_MortCalc (.kcp file)
    a. Defines groups of tree species in the East Cascades FVS variant for purpose of applying mortality rates pre-calculated by the First Order Fire Effects Model. These rates were previously calculated, by diameter class and species group, in FOFEM. We used region wide FIA average values as inputs to FOFEM (e.g., of diameter, height, crown ratio) for the FIA sampled trees in that species group and diameter class. This FVS "addfile" (a .kcp file coding an optional FVS workflow that is executed in the FVS event monitor) instructs FVS to sum-up the wood volume in trees predicted to die (based on these rates and the TPA represented by each sample tree) in the event of a fire with a 7-foot flame length (more accurately, this is calculated as the mean of that predicted mortality for a 6- and an 8-foot flame length).

3. Regen_SpeciesMethod_EC_minplots (.kcp file)
    a. Inserts species-specific natural tree regeneration into FVS simulations based on stand composition, tree size and canopy cover. See Busby et al. (draft ms) for description of the REGIMPUTE approach implemented here.
4. 873_cwidemo (.kcp file) - Prescription 873 parameters (includes thinning style, residual stand density targets, surface fuel treatment and associated thresholds for implementing them, etc.) - see project overview and comments within the kcp for treatment descriptions
5. 874_cwidemo (.kcp file) - Prescription 874
6. 763_cwidemo (.kcp file) - Additional Prescription for Practice Opportunity (note that this prescription was NOT specified in the provided BioSum project nor simulated in FVS—those are practice opportunities for the BioSum learner)
7. PlotCNlist (.txt file containing one Plot CN [control number] per row and no heading): Plot CNs were obtained via GIS overlay of a point layer generated from the public plot coordinates in the Washington State SQLite FIADB.PLOT table on the CWI landscape boundary polygon layer. Plots were selected for overlay if the belonged to EVALID 532101 (WASHINGTON 2021: 2012-2021: CURRENT AREA, CURRENT VOLUME).
8. Optimizer Weights (.txt file) – weights that can be imported into Optimizer and used to calculate weighted mean values of FVS predicted stand attributes from selected simulation years

Workflow used to build CWI_demo

I. DATABASE MODULE

1. Create new project → cwi_demo_YYYYMMDD (replacing the YYYY with the current4 digit year and the MMDD with month and day can be helpful for maintaining version control)
2. Load the plots in the PlotCNlist into the project just created.
3. Check project data sources. All file statuses should read "found." All sources needed can be found in the files included in this demo project, except the WA FIADB, which can be downloaded from the FIA DataMart. If you do not already have the WA FIADB database, make sure to download the Washington state level SQLite database SQLite_FIADB_WA.db available on the DataMart as a .zip file. The file paths displayed in this dialog provide some useful clues for looking "under the hood" at where BioSum stores a project's components.

II. FVS MODULE

For instructions on conducting FVS simulations suitable for BioSum, please refer to the FVS User Guide and BioSum User Guide; note that FVS simulations must start from the BioSum-created FVSIn.DB – output from FVS simulations not initiated from that BioSum-created database can NOT be loaded into BioSum.

1. Assign plot Variants: Run audit, if plots appear in the table in the upper half of your screen, choose the "check all boxes" option and then the "assign variants" function, then SAVE. Do re-run the audit to ensure all plots have been successfully assigned to an FVS variants (all will be assigned to EC for this set of plots because it is the only variant in the CWI landscape)
2. Add Rx Prescriptions
    a. For this project we defined two active management prescriptions, 873 and 874, that build on management concepts used by silviculturists in this region. There is also a grow-only 'prescription' (999), that can be used as a base or reference outcome when evaluating treatment effectiveness. See the project overview document to learn about these prescriptions.
        i. Pro tip for quality assurance: Add the path and file name to both the prescription .kcp (as a comment) and within the description of the prescription (entered into the dialog box when creating the treatment, which can also be edited at a later time). Consistently maintaining all .kcp files in the [project_name]\fvs\data\[variant_code]\ folder to which they apply makes them easier to find!
    b. Add harvest methods
        i. We selected:
            1. Low Slope Harvest Method: Ground-based Mech WT
                a. Justification: As a fuel treatment, whole tree harvest leaves less residues, potentially avoiding the need for follow-on prescribed fire or pile and burn; mechanical harvesting will work on these slopes and with the tree size classes to be harvested and is typically the safest and least-cost way to implement these treatments.
            2. Steep Slope Harvest Method: Cable Manual WT
                a. Justification: Can't get a harvester on steep slopes (unless using tethered logging, which is also a valid option, where such equipment is available), so a cable system is needed, accompanied by manual felling. Trees are mostly small enough that they can be yarded as whole trees (not bucked and limbed in the woods) increasing the residues that can

be recovered to the landing and potentially utilized. If addressing residues via post-treatment fire (piled or broadcast burn), and if recovery of residues for energy generation is not a priority, or not feasible owing to lack of markets, tethered systems are a good alternative here.

c. Additional CPA
    i. Add any flags for additional cost per acre (CPA) components incurred conditionally in the .kcp file (e.g., contingent on amount of harvest residues) or for a particular prescription, in all cases, or processor scenario.
        1. For this project, we have the activity (which incurs additional costs) flags set for our fuel treatments, pILE for a pile/burn and bROAD for a broadcast burn.
        2. In 873, pile/burn and broadcast burns are both coded into the kcp file to occur ONLY when there is thinning activity and a post-thinning fuel load threshold (12 tons/acre in this case) is exceeded.
        3. In 874, the flags do NOT have to be added at this time. Additional cost components for 874 are assigned in the processor module, because fuel treatments in 874 are not conditional and occur whenever trees are cut.

3. Create Rx packages
    a. Create three Rx packages – 873, 874, and 999 to hold the defined prescriptions. Except for the grow-only (GO) package 999, the same treatment is applied at all four cycles; however, different Rxs can be used at different time points (BioSum cycles) in the package, if desired.
    b. *Pro tip*: Carry over Rx kcp description and kcp file path information to description box on the New FVS Treatment Package Window when a package involves the repeat of the same prescription at each BioSum cycle
        i. FVSOUT_EC_P873-873-873-873-873
        ii. FVSOUT_EC_P874-874-874-874-874
        iii. FVSOUT_EC_P999-999-999-999-999 (grow only/baseline)

4. Tree Species
    a. After running the audit, one variant-tree species combination may be returned in the results box— EC 378. To proceed, assuming you are using the provided set of plots, you'll need to provide the common name of the tree, which is:

northwestern paper birch (the species code will be mapped to the 'other hardwoods' species group).

i. The northwestern paper birch (FIA species code 378) is not represented in any of the FVS regional variants; thus, it does not automatically populate with a common name, genus, or species. In this scenario, the analyst must use the REF_SPECIES table, from the FIADB database, to identify the correct common name from the species code (spcd) displayed. *Only the common name needs to be entered for the audit to pass.*

b. To update the tree species table as described above, check the box next to the unrecognized tree species (378), and click the "Add Checked Items to Tree Species Table" button. Then, scroll to the bottom of the Tree Species Table in the bottom half of the window. The last observation will only have the first two columns populated with "EC" and "378" - click on that row and select "edit." Add the common name of the tree, then make sure to SAVE. Finally, re-run the audit to ensure your changes were properly saved. At this point, the audit should run successfully.

5. FVS Input

    a. Navigate to the same SQLite_FIADB_WA.db database used to load plots in BioSum's database module.

III. PROCESSOR MODULE

1. Tree Species and Diameter Groups
    a. Tree Species Groups
        i. *NOTE:* Make sure the "Show only species found in the FVS tree tables" box is checked before assigning species to groups to reduce the decision burden of assignments species to groups (there's nothing to be gained by making assignments for species that are not found in your project).
2. Harvest methods (aka harvest systems) were specified at the rx level; however, those could be overridden here with other choices. For other settings, such as for move-in costs, this project relies on the default values provided in BioSum.
3. Escalators: Default escalators (0.65, Cycle 2; 0.44, Cycle 3; 0.30, Cycle 4) correspond to a discount rate (alternative use of funds assumption) of 4%.
4. Additional CPA: Assign pILE and bROAD costs ($150 and $300, respectively, and in one assignment at a time) in the Default $/ac box, then in the drop-down menu, select "assign default value to all component NULL values"

i. *NOTE*: After running processor, only prescriptions with additional cost components (e.g., fuel treatments), will appear in the "additional_kcp_cpa" table in the scenario_results.db

Ad-hoc workflow for generating pre & post_fvs_custom tables (can also be found at biosum.info/downloads)

Especially when some stand attributes require post-processing for their calculation (for example, when the need for them was not anticipated when writing the kcp code), the creative analyst can perform these calculations via database queries, storing the calculated values in two tables: PRE_FVS_CUSTOM and POST_FVS_CUSTOM. For this project, we used this workflow to calculate the SURVRATE variable, defined as 100 – MORTRATE (MORTRATE was calculated in the FVS event monitor based on code in the FOFEM .kcp). Collecting all of the stand attributes to be used in an analysis into these custom tables is a good use for this workflow, though we did not implement it here.

We followed these steps to add one new variable, SURVRATE, to new FVS_CUSTOM tables to make it available for optimizer analysis:

1. Create a new Access database and link to the desired tables, both PRE and POST, from PREPOST_FVSOUT.DB (project > fvs > db) – note that in this example, we're using only information (MORTRATE) from the COMPUTE table and we are using the PRE/POST_FVS_SUMMARY tables as the scaffolding to build the FVS_CUSTOM tables; the POTFIRE and other tables might be useful if the user wishes for all FVS metrics to be copied, via analyst-written queries, into the CUSTOM tables. In this case, the tables needed include:
    a. FVS_OUT_COMPUTE
    b. FVS_OUT_POTFIRE
    c. FVS_OUT_SUMMARY
    d. Any other tables needed, depending on the variables of interest
2. In a separate window, open the FVSOut.db in SQLite Studio and select a table as a "template" for the PRE/POST_FVS_CUSTOM tables to be created. PRE/POST_FVS_SUMMARY is often the best choice as it is all but guaranteed to contain output for every stand for every PRE and POST cycle case.
    a. Choose the PRE_FVS_SUMMARY table and select "Create similar table" from the dropdown menu. Name the new table "PRE_FVS_CUSTOM" and remove all columns EXCEPT the first 9 (biosum_cond_id, rxpackage, rx, rxcycle, fvs_variant, CASEID, STANDID, YEAR, RUNTITLE), committing structure changes at each step.

b. Then use the "add column" button and title it (in this case, SURVRATE, which should be defined with the data type Double) to create an empty column in the FVS_CUSTOM table that can be populated via queries drawing on attributes in other tables for the same cases in Access, R, etc.
    i. Step b should be repeated to create the POST_FVS_CUSTOM table; however, instead of using the POST_FVS_SUMMARY table as the scaffolding, you can create a new table from the PRE_FVS_CUSTOM table you have already created and name it POST_FVS_CUSTOM. Doing it this way drops the step of adding SURVRATE again to the post table (next step).

   At this point, our PREPOST_FVSOUT.db will contain two empty PRE and POST_FVS_CUSTOM tables.

3. To populate the columns, queries can be used in Access, R, Python, etc. In this project, we used three queries in Access that manipulate the tables just created in SQLite Studio:
    a. First, we linked to the FVS PRE/POST tables database, which now includes the empty PRE/POST_FVS_CUSTOM tables created in the SQLite DB.
    b. We then appended the first 8 columns of PRE/POST_FVS_SUMMARY into the empty PRE/POST_FVS_CUSTOM tables, respectively (PRE – PRE, POST – POST) using the following query:

```
INSERT INTO PRE_FVS_CUSTOM ( fvs_variant, biosum_cond_id,
CASEID, STANDID, rxpackage, rx, rxcycle, [YEAR] )
SELECT PRE_FVS_SUMMARY.fvs_variant,
PRE_FVS_SUMMARY.biosum_cond_id, PRE_FVS_SUMMARY.CASEID,
PRE_FVS_SUMMARY.STANDID, PRE_FVS_SUMMARY.rxpackage,
PRE_FVS_SUMMARY.rx, PRE_FVS_SUMMARY.rxcycle,
PRE_FVS_SUMMARY.Year
FROM PRE_FVS_SUMMARY;
```

    Don't forget to make a copy of this query, edit every case of PRE to read POST, and run that query also (to fill in the first 8 columns of the POST_FVS_CUSTOM table).

    c. Next, we used an update query to populate the empty SURVRATE columns, using an expression that calculates SURVRATE as [1-MORTRATE]:

```
UPDATE PRE_FVS_COMPUTE INNER JOIN PRE_FVS_CUSTOM ON
(PRE_FVS_COMPUTE.fvs_variant = PRE_FVS_CUSTOM.fvs_variant) AND
(PRE_FVS_COMPUTE.rxcycle = PRE_FVS_CUSTOM.rxcycle) AND
(PRE_FVS_COMPUTE.rx = PRE_FVS_CUSTOM.rx) AND
(PRE_FVS_COMPUTE.rxpackage = PRE_FVS_CUSTOM.rxpackage) AND
(PRE_FVS_COMPUTE.biosum_cond_id = PRE_FVS_CUSTOM.biosum_cond_id)
SET PRE_FVS_CUSTOM.SURVRATE = 1-[MORTRATE];
```

Don't forget to make a copy of this query, edit every case of PRE to read POST, and run that query.

d. Finally, we used a rounding query to limit the number of decimal places to four:

```
UPDATE POST_FVS_CUSTOM SET POST_FVS_CUSTOM.SURVRATE =
Round([SURVRATE],4);
```

Don't forget to make a copy of this query, edit every case of PRE to read POST, and run that query.

IV. OPTIMIZER MODULE

1. Define Calculated Variables:
   In this step, we are calculating the weighted values of each of the selected variables below. Values are weighted across the length of the simulation. While as many attributes as desired can be added in this part of optimizer, only up to four variables can be defined for each scenario, and effectiveness can be defined for all four if desired. In this project, we have only defined effectiveness for the first variable (SURVRATE for scenario 1, CBH for scenario 2). The other variables listed are 'carry-over' variables that can be used for quality assessment or to monitor trends in these or other stand attributes.
   a. Once you have selected your stand and economic attributes of interest, click "Recalculate all" to obtain the weighted variables.
   b. Stand attributes (FVS variables) used as effectiveness metrics in this project's optimizer scenarios:
      i. **Scenario 1: SURVRATE_1** (pre/post_FVS_CUSTOM_WEIGHTED table) – used to assign effectiveness as well as in the optimization as an attribute to be maximized
      ii. **Scenario 2: CBH_1** (pre/post_FVS_POTFIRE_WEIGHTED table) -- used to assign effectiveness as well as an in the optimization as an attribute to be maximized
   c. Economic attributes used in both example optimizer scenarios for this project:
      i. **Net_revenue_1** (the total present net value of 4 decades of management: sales of wood less costs of operations) – used to break ties when multiple RxPkgs are equally effective.
   d. Additional weighted variables are defined (not yet implemented in this project for effectiveness determination or optimization, calculating them at this stage of the workflow makes them available should they be needed for this purpose in a new scenario and makes them available in the outputs of existing scenarios as

they may be of interest to track in the all_cycles_effective table in the optimizer_results.db):

i. **WEIGHTED.ALLVOL (ALLVOL_1):** Potentially useful for tracking value, potential habitat, potential carbon storage, and site occupancy as represented by merchantable wood volume
ii. **WEIGHTED.QMD (QMD_1):** Useful for evaluating relative progress towards a forest structure comprising larger trees and higher quality wildlife habitat
iii. **WEIGHTED.MORT_VOL_SEV (MORT_VOL_SEV_1):** Useful as a resistance metric that incorporates FVS-modeled effects of changes made to surface fuels (e.g., prescription burns) and implications for future mortality (e.g., fewer small, fire vulnerable trees after prescribed fire consumes some of them, reducing the percent mortality and increasing volume survival rate)

e. Weights: Import provided optimizerweights.txt text file

i. Cycle 1 PRE: 0.0 (the record for these years, 1 for FFE tables and 2 otherwise, is used only to provide pre-treatment descriptors for stand attributes before management simulation begins)
ii. Cycle I POST – Cycle 4 PRE: 0.125 (each represents 5 out of the 40 years of simulated growth and management)
iii. Cycle 4 POST: 0.25 (with the timing in this project, this final record is the best representation of the last quarter of the 4-decade simulation, accounting for, in the SUMMARY table for example, years 32-41)

Accounting for missing values and reweighting variables ad-hoc workflow

Some stand attributes cannot be calculated at some points in an FVS simulation. For example, canopy base height (CBH) is undefined when there are no trees in the tree list (e.g., after a clearcut, fire or other major disturbance). The same is true for torching and crowning index, when there are very few trees on a plot, owing to the process by which those are calculated. Even SURVRATE (percent of the volume that can be expected to survive fire) cannot be calculated at times when there are no trees since it is derived from MORTRATE – a ratio of fire-killed tree volume to total tree volume. Even if the numerator of that ratio is zero, a zero in the denominator makes it undefined and uninterpretable.

For FVS Fire and fuels extension attributes like CBH, FVS will record a –1 as a missing value code and BioSum will convert that value to null when calculating the weighted mean attribute values. For SURVRATE, we also have null values in the source tables (in the PREPOST database). In all cases of null values in the PREPOST tables, BioSum's calculations convert these

null values to zeros when computing the weighted mean, resulting in a weighted mean value that is biased low.

A workflow has been devised to correct for that bias by adjusting the weights to make it as if only non-null cases were used to calculate the weighted mean. See biosum.info Downloads/UsersGuide section for an explainer of this workflow.

2. Ownership
    a. All ownership categories are selected for this project.
3. Cost and Revenue
    a. Round Trip Truck and Driver Haul Cost per GT Hour: $10.00
        i. These may vary if an analyst uses a different GIS dataset
    b. Processor Scenario: This project only features one processor scenario, which should be checked in this tab. Should an analyst wish to add additional processor scenarios, they would appear here.
4. Wood processing sites
    a. Click the "Load GIS Data" button to bring in the travel times database provided along with this project (gis > db > gis_travel_times.db)
        i. The gis_travel_times.db contains the list of all processing sites (whether they still exist or not) and travel times required to haul to those sites
    b. Once in the Wood Processing Sites tab in the Optimization Scenario Rule Definition tab, the full list of processing sites will appear in an interactive table; you can filter based on whether the mills still exist, whether they process merchantable wood or chips, or by state or county.
        i. When optimizer is run, it selects the closest processing sites, regardless of whether they still exist, to which to send harvested wood. For the purposes of this project, we checked the boxes of the processing sites selected most often by the BioSum optimizer (from another project using the same plot data) and filtered out any that no longer exist, for a total of six processing sites (note: Merch = Merchantable Wood):
            - 1205 – Yakama Forest Products (Merch; Yakima, WA)
            - 1241 – Hampton Lumber Mills (Merch; Lewis, WA)
            - 1359 – Willis Enterprises Cle Elum (Merch; Kittitas, WA)
            - 1360 – Willis Enterprises Cle Elum (Chips; Kittitas, WA)
            - 1383 – Clean Burn Pellets (Chips; Pierce, WA)
            - 1409 – Cumberland Logs to Lumber (Merch; King, WA)
5. Filter plot and condition records

a. In this project, we have not added any filters on plots; however, analysts can filter our plots based on characteristics
b. In this project, we have only excluded non-forested conditions up to this point in the analysis; however, we must also remove reserved conditions (no treatment permitted), since no trees can be cut from these stands. On the "Filter Conditions Record", an SQL query can be used to define the condition attribute filter; here, we have added "`WHERE @@CondTable@@.reservcd = 0`" to select ONLY unreserved (reservcd=0) conditions to go through optimizer.

6. FVS variables (Scenario 1): We did not define effectiveness for any variables except variable 1 in this project; however, any of the weighted variables can be used to define variable-specific effectiveness or construct the overall effectiveness statements:
    a. **Variable 1: SURVRATE_1**
        i. Variable 1 Effectiveness Definition: Any improvement in survival rate and not null (*variable1_change > 0 AND post_variable1_value is not NULL*)
    b. Variable 2: ALLVOL_1
    c. Variable 3: QMD_1
    d. Variable 4: MORTALITY_VOL_SEV_1
        i. Overall Effectiveness Definition: *variable1_effective* = *Y*

7. FVS variables (Scenario 2)
    a. **Variable 1: CBH_1 (Canopy Base Height)**
        i. Variable 1 Effectiveness Definition: CBH improves by at least 2 ft. and the post-treatment CBH value exceeds 15 ft, and not null. (*variable1_change >= 2 AND post_variable1_value > 15 AND post_variable1_value is not NULL)*
    b. Variables 2-4 are the same as in Scenario 1
        i. Overall Effectiveness Definition: *variable1_effective* = *Y*

Analyzing Outputs: A Quick Guide to Answering a few Policy-Relevant Questions

BioSum's output is in the form of a queryable database containing a wealth of information pertinent to a broad range of questions. We provide four examples here of how to extract such policy-relevant information using the results of Optimizer Scenario 1. The following queries were constructed in an MS Access database containing links to the following tables in these three databases (1 Access mdb and 2 SQLite db):

Cwi_demo_20241205\db\master.mdb for

- Cond

Cwi_demo_20241205\fvs\db\PREPOST_FVSOUT.db for

- PRE_FVS_CUSTOM

Cwi_demo_20241205\optimizer\scenario1\db\optimizer_results.db for

- All_cycles_best_rx_summary
- All_cycles_effective
- Econ_by_rx_cycle
- Econ_by_rx_utilized_sum

1. Thinking about treatment effectiveness, feasibility and what is actually accomplished, on average, one might ask: On how many acres could each treatment be effective, on how many of these would revenues from sales of wood cover treatment and haul costs, what would resistance look like (in terms of percent of live volume that survives fire) if the treatment were implemented and how would that change relative to no treatment (grow-only)? We used survival volume percent (calculated for this query as SURVRATE*100) as the effectiveness metric, and rated feasibility as self-paying or not.
   a. *Query:*

```
SELECT all_cycles_effective.rxpackage, Int(Sum([cond].[acres])) AS
AreaAc, IIf([max_nr_dpa]>0,"self-pay","subsidy required") AS
Feasibility, Int(100*Avg([post_variable1_value])) AS
SurvPct_40yr_mean, Int(100*Avg([variable1_change])) AS
SurvPct_40yr_mean_change

FROM (all_cycles_effective INNER JOIN cond ON
all_cycles_effective.biosum_cond_id = cond.biosum_cond_id) INNER
JOIN econ_by_rx_utilized_sum ON (all_cycles_effective.rxpackage =
econ_by_rx_utilized_sum.rxpackage) AND (cond.biosum_cond_id =
econ_by_rx_utilized_sum.biosum_cond_id)

WHERE (((all_cycles_effective.variable1_effective_yn)="Y"))

GROUP BY all_cycles_effective.rxpackage, IIf([max_nr_dpa]>0,"self-
pay","subsidy required");
```

i. Query Result:

Query Result				
rxpackage	AreaAc	Feasibility	SurvPct_40yr_mean	SurvPct_40yr_mean_change
873	619974	self-pay	68	12
873	475028	subsidy required	51	10
874	300036	self-pay	57	3

Query Result				
rxpackage	AreaAc	Feasibility	SurvPct_40yr_mean	SurvPct_40yr_mean_change
874	124853	subsidy required	40	2

Shows us that 873 (thin from below) is more effective at leaving a more fire resistant forest (effective on more acres, leads to a higher resistance level and greater increase in resistance relative to doing nothing), than 874 (thin across diameters), as one would expect. Moreover, 873 is able to self-pay on more acres. For both Rx Packages, treatment on acres that self-pay generated greater resistance than on acres that don't.

2. How much wood (both merchantable wood and dirty chips), per acre, is harvested in the first half versus the second half of the 40 year simulation?

   This requires a couple of queries, with the 2nd feeding off of the first.

   a. *Query 1: AcreTot12*

```
SELECT cond.owngrpcd, Sum(econ_by_rx_cycle.acres) AS Cyc1_2_ac
FROM econ_by_rx_cycle INNER JOIN cond ON
econ_by_rx_cycle.biosum_cond_id = cond.biosum_cond_id
WHERE (((econ_by_rx_cycle.rxcycle)="1" Or
econ_by_rx_cycle.rxcycle)="2"))
GROUP BY cond.owngrpcd;
```

      i. Query Result: Calculates the total acres treated in the first twenty years (Rx cycles 1 and 2) within each ownership class. This query is needed to run query 2 below, which uses the outputs of this query as the denominator for calculating wood output (volume, cf) per acre.

Owner Group Code (owngrpcd)	Total Treated Acres in Rx Cycles 1 and 2
10	1,157,013.13
20	3,720.98
30	260,638.56
40	230,954.35

   b. *Query 2: ChipsandMerch2*

```
SELECT cond.owngrpcd, Sum(cond.acres) AS AllCyc1_2_TreatedAcres,
Sum(IIf([UseBiomass_YN]='Y',[econ_by_rx_cycle].[acres],0)) AS
AcresWhereChipsUtilized, Avg(cond.vol_ac_grs_ft3) AS
   MeanCFperAcYr1,
Avg(cond.balive) AS MeanBAft2perAcYr1,
   Sum(IIf([usebiomass_yn]='Y'
```

```
[chip_wt_gt],0)*[econ_by_rx_cycle].[acres])/Sum([cond].[acres])
   AS
ChipGTperAc, Sum([merch_vol_cf]*[econ_by_rx_cycle]
[acres]/[Cyc1_2_ac]) AS MerchCFperAc
FROM (econ_by_rx_cycle INNER JOIN cond ON
econ_by_rx_cycle.biosum_cond_id = cond.biosum_cond_id) INNER JOIN
AcreTot12 ON cond.owngrpcd = AcreTot12.owngrpcd
WHERE (((econ_by_rx_cycle.rxcycle)="1" Or
econ_by_rx_cycle.rxcycle)="2"))
GROUP BY cond.owngrpcd;
```

i. Query result: Calculates the mean wood volume (cf) ouptuts of both merchantable wood and (utilized) chips per acre across treated acres within each ownership class in Rx cycles 1 and 2, combined;

Owner Group Code (owngrpcd)	All Treated Acres in Rx Cycles 1 & 2 (AllCyc1_2_ TreatedAcres)	All acres where chips utilized (AcresWhereChipsUtilized)	Mean cubic feet/acre (MeanCFperAcYr1)	Mean basal area (ft2)/acre (MeanBAft2perAcYr1)	Chip wt in green tons/acre (ChipGTperAc)	Merch Wood Volume (cf)/acre (MerchCFperAc)
10	1,157,013.13	776,781.55	3,798.21	144.39	17.63	1.937.62
20	3,720.98	3,720.98	2,176.14	99.44	11.55	928.12
30	260,638.56	238,876.59	2,441.26	106.94	13.14	828.28
40	230,954.35	194,170.87	2,404.00	103.2	16.54	1,226.8

ii. The queries above can be easily modified to calculate the mean wood volume outputs in years 22-42 by substituting Rx cycles "3" and "4" in to

```
WHERE (((econ_by_rx_cycle.rxcycle)="1" Or
econ_by_rx_cycle.rxcycle)="2"))
```

for both queries.

3. What is the mean improvement in SURVRATE on plots where the initial survival rate was greater than 50% versus those where initial survival rate was less than or equal to 50%?
   a. *Query:*

```
SELECT Avg(all_cycles_effective.variable1_change) AS
AvgOfvariable1_change
```

```
FROM PRE_FVS_CUSTOM INNER JOIN all_cycles_effective ON
(PRE_FVS_CUSTOM.rxpackage = all_cycles_effective.rxpackage) AND
(PRE_FVS_CUSTOM.biosum_cond_id =
all_cycles_effective.biosum_cond_id)
WHERE (((PRE_FVS_CUSTOM.rxcycle)="1") AND
((PRE_FVS_CUSTOM.SURVRATE)<=0.5) AND
((all_cycles_effective.variable1_effective_yn)="Y"))
GROUP BY PRE_FVS_CUSTOM.rx;
```

i. Query Result: Calculates the mean improvement in SURVRATE (variable1_change) for stands where the initial survival rate (pre-treatment) was less than or equal to 50% (SURVRATE <= 0.5) for each Rx package:

Rx Package	Mean Improvement in SURVRATE (AvgOfvariable1_change)
873	0.13
874	3.62

ii. To calculate the mean improvement in SURVRATE for stands where initial SURVRATE exceeded 50%, simply sub ">0.5" in `((PRE_FVS_CUSTOM.SURVRATE <= 0.5))`.

4. Discussion about the extent to which haul costs make treatment infeasible is constant, so some information on what share of total costs is represented in the on-site treatment costs (felling, yarding, chipping, loading, surface fuels reduction) as opposed to haul costs for cases where treatment self-pays or does not. Asking the BioSum output database questions such as: "What percentage of the total treatment cost is attributable to harvest costs?", and setting some assumptions—we'll look at the optimal treatment on each acre, split out acres that self-pay from those that don't, split acres where chips can be utilized (owing to a facility for which haul cost is less than the chip value), counting chip haul cost only when chips can be utilized, and using the econ_by_rx_cycle as the data source (because the chip utilization is at the cycle level, and not in the sum table, and weighting outcomes by acres represented) we use this query:
   a. *Query 4*:
      SELECT econ_by_rx_cycle.usebiomass_yn AS BiomassUtilized,
      IIf(([max_nr_dpa]>=0),"self-pay","subsidy required") AS FeasibilityClass,
      Int(Sum([chip_haul_cost_dpa]*[econ_by_rx_cycle].[acres])/[AcreTot]) AS
      CHaulCstPerAc, Int(Sum([chip_val_dpa]*[econ_by_rx_cycle].[acres])/[AcreTot])

```
AS CValPerAc,
Int(Sum([merch_haul_cost_dpa]*[econ_by_rx_cycle].[acres])/[AcreTot]) AS
MHaulCstPerAc,
Int(Sum([merch_val_dpa]*[econ_by_rx_cycle].[acres])/[AcreTot]) AS MValPerAc,
Int(Sum([harvest_onsite_cost_dpa]*[econ_by_rx_cycle].[acres])/[AcreTot]) AS
HarvCstPerAc,
Int(100*([HarvCstPerAc]/([CHaulCstPerAc]+[MHaulCstPerAc]+[HarvCstPerAc]))
) AS HarvCstPct, Sum(Int([econ_by_rx_cycle].[acres])) AS AcreTot

FROM all_cycles_best_rx_summary INNER JOIN econ_by_rx_cycle ON
(all_cycles_best_rx_summary.biosum_cond_id =
econ_by_rx_cycle.biosum_cond_id) AND
(all_cycles_best_rx_summary.rxpackage = econ_by_rx_cycle.rxpackage)

GROUP BY econ_by_rx_cycle.usebiomass_yn, IIf(([max_nr_dpa]>=0),"self-
pay","subsidy required")

ORDER BY econ_by_rx_cycle.usebiomass_yn DESC , IIf(([max_nr_dpa]>=0),"self-
pay","subsidy required");
```

i. Query Result:

Query4								
Biomass Utilized	**Feasibility Class**	**CHaulCst PerAc**	**CVal PerAc**	**MHaulCst PerAc**	**MVal PerAc**	**HarvCst PerAc**	**HarvCstPct**	**AcreTot**
Y	self-pay	274	504	434	4812	1411	66	567200
Y	subsidy required	214	325	117	879	1769	84	320604
N	self-pay	545	389	935	3854	887	37	133886
N	subsidy required	254	177	103	312	882	71	278004

1. On most acres, treatment paid for itself
2. On most acres, dirty chips were utilized
3. Most of the cost is harvest (not haul) cost, except where biomass is not utilized and treatment can still self-pay (because merch value is high, even if unit haul costs are more than twice as high as where biomass is utilized)

4. Note that chip volume in the biomass not utilized cases is not actually uitlized—included here for informational value only

These examples are intended to give an idea of the range of questions that can be addressed using the outputs of a BioSum project. In the examples here, we build queries in MS Access referencing tables linked to Access from the SQLite output database produced by BioSum. Python, R, and the OK Tabling Program (a Python script that operates on SQLite FIADB tables and tables that can link to them, such as those in the BioSum outputs) are other ways to summarize and analyze BioSum outputs to generate insights that support decision-making.