Workflow for Morris Sensitivity Analysis of RHESSys: Data Processing and Running the Analysis

0. **Code Notes**

   This readme provides a step-by-step set of instructions that describe how to set up, run, and analyze results for a Morris sensitivity analysis of a Regional HydroEcological Simulation System (RHESSys) model. The instructions are intended to be general, such that any project may rely on this step-by-step process. That said, the instructions are based on the steps needed for the cited Smith et al. study, and modifications to code and/or to the steps may be needed for other studies. Guidelines are provided where code edits may be required.

   Rivanna is the HPC resource at The University of Virginia that was used to run this code.

   For the example study, ~20 GB of RAM is required for step 16.xi, at least as the code is written. The code could be modified to need a max of ~6 GB.

IF YOU ONLY WANT TO GENERATE THE SMITH ET AL. PAPER FIGURES:
Skip to step 36, and see the data repository readme file to learn how to run the R script.

Full Workflow:

1. **Create a Main Directory in /scratch/<user ID>/<MainDir>** (e.g., /scratch/js4yd/MorrisSA)
   This directory will have all of the files needed to run GIS2RHESSys (model preprocessing), RHESSys, and the sensitivity analysis.

2. **Move Data and Code to the Main Directory**
   Move from the Smith et al. data repository for the exact code used.
   - Date_analysis GitHub: https://github.com/laurencelin/Date_analysis
     - Commit from Sept 12, 2019: 54448364b5c71e368f204e4ce0cd78034fcbb3f4
   - ssurgo_extraction GitHub: https://github.com/laurencelin/ssurgo_extraction
     - JDS modified code:
     - JDS modified a commit from L. Lin on Sept 4, 2019: 568acc4ca6218a6ac3d0f1e79fbd6aef3a66947f
       - Modified ssurgo_extraction.R to report 5 significant digits.
       - Modified ssurgo_soiltexture2gis.R with additional library loads: sp and rgdal. On some computers (R versions?), the sf package is default instead of sp. If you receive an error about the sf package when this script is run, uncomment the line #use_sp() in this file to force R to use the sp package.
       - R library paths were added as an input argument.
   - GIS2RHESSys GitHub: https://github.com/laurencelin/GIS2RHESSys
     - JDS modified code and data:

- JDS modified a commit from L. Lin on Sept 25, 2019: af7a26cc6f03be51fbd4aa6f75a976d04e38dd67
    - The csv files downloaded from L. Lin's GitHub do not contain all of the possible parameters for soils, land use, and vegetation. JDS modified these files to include all of the possible parameters. These files must contain the same exact parameters as the def files that will be used to run RHESSys.
    - Changed some of the vegetation, soil, and LULC csv file parameter values. For all parameters that will remain constant across all model runs, these files must have the desired value specified for those parameters. Other parameters that are changing between SA replicates will be updated by the scripts.
    - Modified zone_cluster.R with an argument to set the random seed.
    - Added R library calls (sp, rgdal, and/or XML) to several R scripts (both aggregate_lulc scripts, basin_determine, elevation_analysis, and zone_cluster).
    - Changed the g2w script to import downloaded GitHub libraries instead of accessing from the internet upon running. Also added round() and signif() statements for outputs. Also added all of the zone and hillslope def parameter options in this script.
    - Modified the workflows shell script to remove randomness in the septic tank location assignment (assigned septic tanks to all lawn patches around roofs). Also moved grass_delineation_1 and 2 file contents to the workflows script (they didn't run on Rivanna).
    - R library paths were added as an input argument.

- raw_data folder with Roads (shapefile), DEM (*.tif), LULC (*.tif), Soils (SSURGO), etc. required for your model in GIS2RHESSys and ssurgo_extraction
- workflows_Singularity_Baisman30m_Rivanna_FromScratch.sh for running GIS2RHESSys to obtain the initial basemap conditions for RHESSys. The output from this script will be used for every RHESSys run, and be modified according to the SA replicate parameter values.
- One of the RunArray.sh scripts for running SA replicates. See step 14 for script options.
- RenameOutputFiles.sh script for moving and renaming the output files, if needed.

3. **Change several variables in the workflows…sh script for your project**

Section 1.1

- GITHUBLIBRARIES – full path to the GIS2RHESSys library in step 2
- SSURGOLIBRARIES – full path to the ssurgo_extraction library in step 2
- RLIBPATH – full path to the R libraries in the user's home directory in step 4b

Section 1.2

- PROJDIR – full path to the directory made in step 1.
- EPSGCODE – the EPSG code for the map projection of the output data. Must be UTM.

- RESOLUTION – the desired grid cell resolution of output raster files (5 – 30 m good for RHESSys modeling)
- gageLat, gageLong – the WGS84 coordinates of the basin outlet location. Does not have to be a stream gauge location, but having a gauge there makes more sense for model calibration.
- expectedDrainageArea - Expected area of the watershed in m^2
- expectedThresholdModelStr – The minimum drainage area upstream of a patch that results in a patch being designated as a stream patch. Defines the major stream channels for RHESSys. Unit: m^2
- expectedThresholdStrExt – The contributing area upstream of the ModelStr set point that sets the limits of upstream extension of streams. Extension includes headwater channels for riparian delineation and surface water routing. Unit: m^2
- GRASS_drainarea_lowerbound / upperbound – set the tolerance level of the watershed delineation. Default is set as 2% tolerance to expectedDrainageArea
- RHESSysNAME – name of the folder that will contain the output files for this script. These files will be used as input files for RHESSys.
- LOCATION_NAME – the name of the folder that will contain the output GIS data

Section 1.3

- downloadedDEMfile – name of the DEM file from step 2

Section 1.5

- Near bottom of section, set the default assignment of slope and aspect when they are null (unit: degrees):

```
slope = if(isnull(slope_),0.143,slope_)
aspect = if(isnull(aspect_),abs(drain)*45,aspect_)
```

Section 1.6

- set the random seed to use for clustering zones.

Section 1.8

- downloadedSSURGOdirectoryPATH – full path to the SSURGO data from step 2
    i. You may also need to set the path to the spatial data for soils on the following line, and the MU key *.csv file location.

Section 1.9

- downloadedLULCfile – path to the land cover data from step 2
- Near the bottom of the section, set the path to the csv file defining how the land cover code values should be translated into RHESSys land cover classes:

```
"$PROJDIR"/GIS2RHESSys/lulc_1m_Chesapeake_Conservancy.csv
```

Section 1.10

- change properties of each vegetation stratum. To add more vegetation species, see instructions in the more recent versions of RHESSysEastCoast. An example is provided for a second tree species in this workflow (which will result in a file with all no data grid cells).

i. grassStratumID number corresponding to the def file stratum_default_ID. Must match the ID number in the vegCollection.csv file in step 2.
ii. grass1FFrac value for coverage fraction of the patch.
iii. grass1LAI value for the maximum LAI in a growing season.

Section 1.11

- downloadedROADfile – full path to the roads shapefile from step 2

Section 1.13

- Set properties defining storm drains and sewers, and soil compaction
    i. Buffer to use (distances=##) (patch resolution should be the minimum ## value)
    ii. Thresholds for sewer drainage (distances=##)
    iii. Thresholds for additional surface drainage
    iv. Septic system thresholds

Section 2.1

- Set climate station ID and file name
    i. Can add more climate stations, if available. One ID number and name per station.
    ii. Can uncomment crop and shrub if those exist in the area of interest
    iii. Make sure the template file has all of the necessary map data names in it

4. **Run GIS2RHESSys in the Singularity container**

The code repository contains a definition file to create the Singularity image file. The definition file is rhessys-v3.def, and the resulting image file should be named rhessys_v3.img. The following command should build the image. You may need to ask your HPC staff to place this in a designated container location.

```
singularity build rhessys-v3.img rhessys-v3.def
```

Check that in your home directory's .grass7 directory that there is no rc file. This rc file tells GRASS where the main GRASS directory is on the computer. If it exists, delete it as long as you know that's okay for your other work that relies on GRASS. A new rc file will be created when GRASS is run. If you cannot delete it, I do not know enough about GRASS to suggest an alternative method. Seek IT support, or use another computer/username. Similarly, you will not be able to run other GRASS processes until this analysis has completed.

Start an interactive Singularity session:

```
singularity shell <path to container image file>/rhessys_v3.img
```

Check the contents of the R packages in this container by running R through GRASS.

a. Make a GRASS directory: Replace the EPSG code number with your map projection requirements. Replace directory name with your desired full path directory for GRASS. Home folder or scratch will be fine, as this directory will not be used for computation.

```
grass74 -c 'EPSG:26918' -e <directory name>
```

b. Run R through GRASS to be able to check packages and install any necessary packages

```
grass74 <full path to GRASS directory>/PERMANENT --exec R
```

i. use installed.packages() to see if these packages are installed: sp, XML, rgdal, rgrass7

ii. For any packages that are missing, use install.packages() to install. Install in the order listed above. If prompted, type yes to add R packages to your home directory. Do not install with dependencies for this! If you need to, choose a CRAN mirror to download the packages. I use Oregon State University.

Run: sh workflows_FromScratch.sh to run GIS2RHESSys. This should be run in the main directory created in step 1.

Check for errors and warnings in the printed terminal output

- Some errors will happen that are non-fatal or are intended
  - i. ERROR 6: SetColorTable()
  - ii. ERROR: No existing MASK to remove
  - iii. WARNING: Vector map <tmp> already exists and will be overwritten
  - iv. WARNING: No areas selected from vector map <roads>
- Others may be fatal in that reported data are not as desired

  WARNING: Busy SQLITE db, already waiting for 10 seconds...

5. **Add RHESSys input files to the created RHESSysNAME directory**

   Move from the Smith et al. data repository for the exact code used.
   - tecfiles directory with the tecfile in it.
     - Note that the end date must be the day after the last day for which you want data!
   - clim directory with climate station base, rain, tmin, tmax, etc. files
     - Note that the dates in these timeseries must match the tecfile start and end date. These files in the data repository were made in the BES_GF_BR_SL.R script in the EnGauge GitHub.
       - EnGauge commit from Nov. 24, 2019: 78b16e37a11c4e1f5791610ca62a88b6ca6570b5
       - Cleaner (no hard-coded numbers) in commit from March 9th, 2020 2153a4114871b38da00e216659e32b382f2d02bd
   - defs directory with all of the def files needed for RHESSys to run. These should have been output from the GIS2RHESSys workflows shell script.
     - For all parameters that will remain constant, these def files must have the desired value specified for those parameters. If they were specified in the *Collection.csv files in step 2, then they are also specified in these def files.

6. **Make a file of parameters that will be evaluated in the sensitivity analysis.**

   This file only includes parameters that will change values across runs.
   Example in data repository: BaismanMorrisSamplingProblemFile_Full.csv

Rows: parameters
Columns: unique parameter name, RHESSys parameter name, lower bound, upper bound
This file uses the following notation for prefixes for unique parameter names:

h – hillslope parameters
z – zone parameters
l#– land use parameters
s# – soil parameters
v# – vegetation parameters
All #s are the ID number found in the *Collection.csv files.
Compacted and uncompacted soils of the same texture are assumed to have the format s10# and s#, respectively.

7. **In the main directory, create a directory to do Morris SA runs** (e.g. RHESSysRuns)
   In that folder place the following:
   - output – empty directory where .out files will be written
   - the file from step 6 (e.g., BaismanMorrisSamplingProblemFile_Full.csv)
   - Sum3CheckFun.py
   - MorrisSampling.py
     - Note: this script has ID numbers corresponding to the values in step 6. You may need to modify if you added or removed variables from the list. Modifications include:
       - Search for "Check sums" for ID numbers to change.
       - At top of the script, many sys.exit commands are a function of ID number for soil and vegetation. You may need to edit these or add more if you change the vegetation and soil ID numbers used.
       - Some of the regular expressions have ID numbers in them. Search for "regex" to find them.
   - MorrisSamplingLocations_BeforeProcessing.R
   - MorrisSamplingDiagnostics.R
   - MorrisSampleCreation.sh

8. **Check that packages are installed**
   R: sensitivity library must be installed for R version 3.5.3. This can be installed outside of the singularity container.
   Python: make sure these Python 2.7 packages are installed: os, pandas, sys, re, numpy, random, string, signal. On Rivanna, Python packages are installed using pip and can be found in /sfs/qumulo/qhome/<userID>/.local/lib/python2.7/site-packages/

9. **Run MorrisSampleCreation.sh**
   This will create the Morris sampling locations from the specified parameter bounds using the MorrisSamplingLocations_BeforeProcessing.R script, and post-process them to ensure that they are within the constraints of the parameters using the MorrisSampling.py script. The output is a csv file of the Morris sampling trajectories containing all of the SA replicates as rows. The

MorrisSamplingDiagnostics.R script is run to make a plot of parameter correlation values before and after processing for constraints.

10. **Make a permanent storage directory for the RHESSys Sensitivity Analysis results**
    On most systems, the directory in which code is run is not recommended as permanent storage (e.g. it is purged every 90 days). So, results should be moved to a permanent storage directory once they complete. It's not recommended to do this within the SA run, if possible, because the file transfer is a slow process when done one-at-a-time that can be completed quicker as a batch folder move after all output has been generated.

11. **Move RHESSys East Coast to a directory**
    Move from the Smith et al. data repository for the exact code used.
    RHESSysEastCoast source code GitHub: https://github.com/laurencelin/RHESSysEastCoast
        Commit from Sept 13, 2019: a0e174ec7435ad41c59d401599d7885baca2e482
        Which was edited by J.D. Smith on Nov. 5, 2019 to correct an error loading in a parameter: e45ab8828e40665b5873a27cdd8087794a6aa768

    This can be placed in any of your directories on Rivanna, and you must specify where to find this directory when you run RHESSys in the step 14 shell script. /scratch/ may be a better location to place this for speed of file transfers.

12. **Compile RHESSysEC**
    In the directory you made, use "make" to compile the executable for RHESSysEC.
    There will be warnings when this is compiled. These warnings do not affect the code. Similarly, errors are all within print statements and do not affect the code.
        For this study, the -Os CFLAGS flag was not used in the makefile, but this can significantly speed up the RHESSys runtime. However, users should check that outputs are minimally affected with and without the -Os compile flag. Laurence Lin reported that vegetation growth mode outputs did not match well in a more recent version of RHESSysEC.

13. **Move Scripts into Directories, and Modify if necessary**
    - ModifyVeg.py
    - MakeDefs_fn.py
        i. If you have changed the def file contents, or have added or removed def files, then this script may need to be edited to handle those changes, as follows:
            1. If you remove a def file, delete that file's code from this script. The specific lines are:
                a. Before the main for loop, there is code to load in the def files to variable names. Delete the file load for each removed def file.
                b. Each def file has one chunk of code in the main for loop. You will have to delete that chunk for each removed def file. Code chunks can be found by searching for the removed variable name in step a. All chunks begin with a comment containing the

def file parameter name prefix (e.g., #v3). All chunks end with a .to_csv function followed by a del.

2. If you add a def file, you will have to add code corresponding to that file in this script, as follows:
    a. Before the main for loop, there is code to load in the def files to variable names. Add a file load here for each new def file.
    b. Each def file has one chunk of code in the main for loop. You will have to add a chunk for each new def file. Chunks of code can be copied for soils and land use and vegetation, and modified using the following steps: 1) change the variable name to match the one created in step a, 2) modify the string.find() and string.split() functions to match the def file parameter prefix (e.g. ‘s8’), 3) modify the sys.stderr error print statements to describe your new def file’s contents, 4) modify the .to_csv file name, and 5) if you’re modifying vegetation code, there are several parameters that likely will need to be the same among all of the vegetation species being modeled. The grass vegetation file currently shows how to copy these variables from one def file to another.

ii. For soils def files, values of m_z and soil_depth are a function of variables. These are set in the code.

iii. For tree vegetation, epc.deadwood_flig is a function of other variables. This is set in the code.

iv. For grass vegetation, several variables were set to match tree parameters’ values (expand, litterfall, leaf on, leaf off, gs thresholds). This was implemented under the assumption that these variables would be similarly affected by climate for all vegetation species.

v. 3 other grass vegetation variables were set as a function of generated variables (PAR transmittance, K reflectance, and min percent leafg).

vi. For non vegetation parameters, K absorptance was set from other generated values.

14. **Run RHESSys with an array of replicates using the RunArray.sh script**

The number of replicates that can be run at once depends on your computer (IT can tell you). On Rivanna, the maximum array number is 9999. So, for more than 9999 replicates it is necessary to run multiple job array submissions.

There are several versions of this shell script available. Each one is similar, but serves a slightly different purpose.

- RunArray.sh: Runs the RHESSys replicate and moves the results to the specified output directory. This method is not recommended because of the long time required to move these files. The move can be completed faster in batch after all replicates are run, and it will not waste compute time if completed after the batch is complete.

- RunArray_NoFileTransfer.sh: Same as RunArray without transferring files to a final destination. Recommended to use this.
- RunArray_NoFileTransfer_Above10000.sh: Same as RunArray_NoFileTransfer, except that this implements a method for running replicates with array IDs > 9999.
- RunArray_NoFileTransfer_DelExistingFolders.sh: Same as RunArray_NoFileTransfer, but deletes existing folders. This is useful if you need to re-run a replicate that failed before completing preprocessing (i.e. the steps before running RHESSys).
- RunLoopSerial.sh: This shell script is used for running one replicate at a time. Useful for testing.
- RHESSys.sh: Used for running only RHESSys. This is useful if a replicate failed while running RHESSys (i.e., all preprocessing steps completed).
- RHESSys_Above10000.sh: Used for running only RHESSys for array IDs > 9999.

For each of these scripts, the following changes may be needed to run on your computer:

- SLURM directories (-D, -o), allocation (-A), email (--mail), etc.
  - -D should match the directory made in step 7.
  - -o should create files in the output folder created in step 7.
- SINGIMAGE – path where the Singularity image file is located
- BASEDIR – directory made in step 1
- EPSGCODE, RESOLUTION, RHESSysNAME, LOCATION_NAME, MAPSET – all of these variables' values should match the values in the workflows…fromScrach.sh script from step 3.
- RHESSysModelLoc – path to where the RHESSys executable is located.
- RHESSysRuns – if your directory in step 7 is not named RHESSysRuns, find and replace with what you called it.
- In the call to MakeDefs_fn.py, edit the name of the file from step 6 (e.g., BaismanMorrisSamplingProblemFile_Full.csv), and also the decimal precision (e.g., 10)
- If needed, edit the names of the GIS2RHESSys *Collection.csv files. (e.g., lulcCollectionEC.csv, vegCollection_modified.csv). Find and replace in script. The vegetation file appears in multiple lines.
- If needed, edit the name of the lulcFrac csv file (e.g., lulcFrac30m.csv). This is output from the workflows…FromScratch.sh script in step 3.
- Section 2.1 – make the edits made to the workflows…FromScratch file that were made in step 3. Note: the lines in this script are slightly different than the file in step 3, so this section cannot be copied and pasted from that script.
- Change the RHESSys command call according to your needs. (e.g., executable name, start and end dates, output types, etc.)

15. **Check for missing runs and errors in final set of output files**
The jobs failed randomly when run on Rivanna. The errors happened at various stages of the job runs. Some runs failed before an output file was created (termed "missing runs"). Some failed before preprocessing was complete. Others failed while RHESSys was running (timeout errors). The SummarizeTimeseries.R script is used to first look for any missing runs (missing .out files), and then read the .out files and scan for any "error" or "traceback" in them. The script will

return job array indices of the runs that have to be resubmitted as string variables that can be copied directly into the HPC terminal to submit a job array. Upon being rerun, none of these jobs failed for this work.

The following R packages must be installed, in addition to previously mentioned R packages: stringi, stringr, GISTools, vroom

The SummarizeTimeseries.R script is best run in an interactive session in the following steps:

i. Run the CheckOutput function to check all of the .out files in the output folder for missing information, errors, and tracebacks.
ii. If there are missing runs from what were expected, rerun those and repeat step i after those runs complete.
iii. If there are errors and tracebacks in .out files, the first line with an error or traceback will be reported for each .out file (Eline and Tline columns of the errs dataframe). Diagnose for each unique line number whether or not the replicates that had those errors need to be rerun. To do this, the you may want to inspect the .out files and the Run# folders for those replicates to see what data were reported.
iv. Once you've determined what to do for a unique line number, the following format can be used to extract all job array indices that have that line number:
```
L155 = errs$ind[which(errs$ind %in% Errors)][which(errs$Eline[which(errs$ind %in% Errors)] == 155)]
```
v. If those job array indices have to be rerun, the CheckOutput.R script's CheckLines function will provide a string that can be copied directly into the HPC terminal to submit a job array of these indices.
vi. Once all errors and tracebacks have been rerun, start again from step i and repeat as needed until all errors and tracebacks have been handled.
vii. The time of a RHESSys simulation is reported at the bottom of each output file. The section "Time in hours for completed results" extracts those runtimes and summarizes as plots.

16. **Compile streamflow results into files (SummarizeTimeseries.R, continued): Note: 20 GB of RAM required for step xi.**

viii. The .out files have now been checked and it's time to check the RHESSys output. The section "Go through results folders" gathers all of the folders in your directory. It is recommended to test this processing on a small subset of the total number of RHESSys runs.
ix. The following section uses the worldfile output to extract information about the area of the basin and hillslopes. A map is also created.
x. The section "Check that the input def file parameter values match the output parameter values" is a check on the RHESSys model's use of parameter information. This should pass if the code has been used properly.
xi. Finally, the section "Loop through all of the folders (SA replicates) and extract the data needed" is used to extract all of the basin and hillslope streamflow and saturation deficit values and combine into one matrix for each variable. Other variables could be extracted from the basin and hillslope files, if desired, but the code would need to be modified to do so. There is an option to plot figures for all variables

for each replicate. This will be very time consuming if completed for all replicates, but it is a good idea to complete and inspect the results for a small subset of them.

xii. After these matrices are compiled, they are saved to txt and csv files, as well as an Rdata file. Their names have the format: SAResults_BasinStreamflow_p4_CorrOrder

xiii. Plots of the streamflow and saturation deficit over time for all replicates are provided for basin and hillslopes in the section "Make plots of the streamflow and saturation deficit observed across the SA runs".

17. **(optional) Run the RenameOutputFiles.sh script**
    This script renames all of the output files from 1-based index to a 0-based index to match the Python indices for the replicates. It then moves them to the specified output directory. This allows the .out files' numbers to match the numbers in the replicate folder names.

18. **Make a directory for processing TN data**
    This directory should be within the directory made in step 1.

19. **Move data and code to this directory**
    - Streamflow .txt files for basin and hillslope from step 16
    - DateColumnNames.txt file (in repo)
    - WRTDS_modifiedFunctions.R script
    - WRTDS interpolation tables made in the BES_GF_SL_BR.R script in the EnGauge repository (S6 of the Smith et al. data repo).
    - ExtractTN.sh script
        - Modify the command line arguments according to your directories and file names
        - Note that the newest WRTDS functions allow for an additional interpolation table to be specified for the square of log flow. If you need to use that function, this code will need to be modified.
    - TNFileExtraction.R script
    - ErrCheckTN.sh script
        - Modify the command line arguments according to your directories and simulation time length
    - ErrCheckTNFileExtraction.R script
    - JoinTN_Basin.sh, JoinTN_Hill05.sh, JoinTN_HillMed.sh, JoinTN_Hill95.sh
        - Modify the command line arguments according to your directories and filenames.
    - TNFileJoining_Basin.R, TNFileJoining_Hill05.R, TNFileJoining_HillMed.R, TNFileJoining_Hill95.R

20. **Compute TN using a job array over the number of days in the simulation**

    R libraries for EGRET, survival, and pracma must be installed.

    The ExtractTN.sh script is used to run the TNExtraction.R script to compute TN from the streamflow using provided WRTDS interpolation tables.

21. **Check for missing runs and errors in the .out files**
    Run the ErrCheckTN.sh script.
    Inspect the TNExtractErrCheck.out file for any errors or missing files.

22. **Compile TN results into files**
    Separate scripts are provided to join basin and hillslope files on HPCs. It was faster to split these into these 4 scripts than to run all at once.
    - JoinTN_Basin.sh, JoinTN_Hill05.sh, JoinTN_HillMed.sh, JoinTN_Hill95.sh run the R script corresponding to their name.

    Takes about 30 mins for the hillslope join, 3 minutes for the basin join.
    About 5GB RAM needed for hillslope.

23. **(DEPRECATED) Reorder the results to match the trajectory order**
    For the initial studies, a code error caused the replicates to be ordered incorrectly (by name instead of number). The ReorderingSAresults.R script was used to reorder the files to the correct trajectory order. It should no longer be needed.

24. **Check that packages are installed to compute the likelihood of parameter sets**
    The following Python packages are needed:
    numpy, math, warnings, pandas, matplotlib, scipy, mpi4py, and pyDOE

25. **Define the likelihood model**
    The version used for Smith et al. is in the data repository.

    The outputs of interest for this study are the timeseries of streamflow (Q) estimated from RHESSys and the WRTDS-estimated total nitrogen (TN) concentration at the watershed outlet's gauge location. Using those simulated timeseries, we compute the residuals compared to the observed timeseries (post-spinup period). The likelihood model describes the distribution of the streamflow residuals for this study, but can be used for the TN data as well. Files for TN are provided in the GitHub repository. The data repository only contains Q code, which GitHub repository also contains TN code.

    Likelihood Function: likelihood.py
    The selected likelihood error model is a skewed exponential power model (generalized normal distribution), which has 6 parameters each for the Q and TN error models. The likelihood.py script was modified from the spotpy GitHub repository's likelihoods.py script's generalizedLikelihoodFunction function corresponding to the commit from March 7th, 2018: 55e696ce965eb6ac35a8884ccb2db060123fc6d5

26. **Evaluate Residual Error Transformations**
    The likelihood parameters are estimated via constrained maximum likelihood estimation (sequential least squares in Python scipy [SLSQP algorithm]). A multi-start approach is used, in which starting locations are sampled using a Latin hypercube sample from specified parameter

bounds. The Flow_MLEfits.py and TN_MLEfits.py scripts perform the MLE approach. These scripts were modified iteratively in an interactive session for several parameter sets (one parameter set at a time) to test the effects of the following algorithm parameters and residual transformations on the resulting likelihood estimation. It is useful to uncomment the lines containing Qdf_success (TNdf_success) for this evaluation. Note that the for loop that fills elements of these data frames was run for only 1 parameter set at a time for these evaluations, and the MPI components of the script were not used.

- Evaluate number of starting locations needed so that at least one of the starting locations converges successfully. The numsamps variable determines how many starting locations are used. 200 was sufficient for this study.

- Evaluate with and without Box Cox transformation. The observed data are transformed, and the same transformation is applied to the simulated data. Search for TrueQ_BC or TrueTN_BC. Comment out the ss.boxcox transformation lines of code. Replace TrueQ_BC (TrueTN_BC) with TrueQ (TrueTN).

- Evaluate with and without de-seasonalization of residuals by month. This is completed for months in these scripts. Search for month (currently only in Flow_MLEfits.py), and uncomment all lines with month in them. Comment out the ObjFunc line that does not have month in it.

27. Set the Likelihood Parameter Bounds

For the desired transformation above, determine the upper and lower bounds of the likelihood parameters. For this study, default values from spotpy were initially used, and the results were plotted to see if any of the parameters were consistently found at the bounds, or significantly away from the bounds. Several parameters were adjusted for subsequent searches: beta, xi, bias (mu).

The bounds must be set by the user in the scripts.

- Search for “bounds=” in the Flow_MLEfits.py and TN_MLEfits.py scripts. The order of bounds in this list is beta, xi, sigma0, sigma1, rho, mu.
- Also adjust the bounds of the uniform random variables. The code chunk that adjusts these starts with “paramsInit[:,0] =” and takes the format (parameter value)*(upper – lower bound) + lower bound

28. Compute the likelihood for flow and TN

Setup and run the Flow_MLEfits.sh and TN_MLEfits.sh scripts.
Several items must be set in the corresponding .py scripts:

- Initial random seed: e.g., the 518 in np.random.seed(seed=i+518). Initial seed should be different for flow and TN.
- number of LHS samples: numsamps
- Full path to streamflow observations .txt file for the basin outlet (step 16): TrueQ

- Full path to simulated streamflow .txt file for the basin outlet (step 16): SimQ
- Output file name: Qdf.to_csv
- parameter bounds (step 27)

Several items must be set in the shell scripts:

- Number of MPI processors (number of nodes, number of tasks per node). It may be desirable to use more processors for Q because it has more data than for TN.
- Other SLURM commands: Runtime, directories, email, etc.

29. **Collect the parameters into single files**
    Run the collectParams.py script.

    Several items must be set in this script:

    - Number of MPI processors (nproc) used in the previous step for Q and TN.
    - The output file names used in the previous step for Q and TN data
    - The output file name to use for the compiled Q and TN data

30. **Plot the results**
    Run the plot_logL_v_SSE.py script.

    Several items must be set in this script:

    - The output file names used in the previous step for the compiled Q and TN data (Qdf and TNdf)

31. **Set up input data for computing Morris elementary effects on Rivanna**
    The RHESSysSensitvityAnalysisSetup.R script is used to create RData files that will be used as input to a job array run on Rivanna (step 34). The line containing "save.image" at the end of the script creates a set of RData files that are used in scripts run on an HPC. The EEs_All_Setup_paper.RData file in the data repository can be loaded to run this script, if desired. Otherwise, the following files are needed to run this script:

    - InputParams: MorrisSamples_AfterProcessing.csv from step 9
    - OrigParams: MorrisSamples_BeforeProcessing.csv from step 9
    - ParamRanges: BaismanMorrisSamplingProblemFile_Full.csv from step 6
    - BasinSF, BasinTN HillSF, HillTN, etc.: Basin and hillslope streamflow and TN data compiled in steps 16 and 22. There are two sets in this script because of an error with 5 SA runs (labeled as Add5).
    - obs, obsTN: Observed streamflow and TN data for basin outlet: BaismanStreamflow_Cal.txt, TN_Cal.txt
    - Likes200: The likelihoods for each model run, output from step 29. For the _pre dataset, the likelihoods were computed after running the line in this script that writes the table needed to compute the likelihoods.

Additionally, you will need to know the date from which the elementary effect computation should start. This is used to remove the spin-up dates from the timeseries. These are '2004-10-01' and '2010-09-30' in this study.

The main directory and directory of the ColorFunctions.R script must be specified at the top.

32. **Make a directory for computing Morris elementary effects**
    This directory should be within the directory made in step 1.

33. **Move data and code to this directory**
    - .RData file from step 31
    - ExtractSA_AllMetrics_[b or h1 or h2].sh
        - Modify the command line arguments according to your directories
    - ExtractSAtrajectories_AllMetrics_[b or h1 or h2].R
    - JoinSA_AllMetrics.sh
        - Modify the command line arguments according to your directories
    - JoinSAtrajectories_AllMetrics.R
    - MorrisSamples_BeforeProcessing.csv from step 9

34. **Compute Morris elementary effects with a job array over the number of trajectories**
    Run the ExtractSA….sh script. Takes about 40 mins.
    Check the .out files for errors.

35. **Compile TN results into files**
    Run the JoinSA_AllMetrics.sh script.
    Check the .out file for errors

36. **Analyze sensitivity results**
    The resulting EEs_All file contains the combined EEs for all variables for basin and hillslope for all SA metrics. The EEs_All_Setup_paper.RData file in the data repository can be loaded to obtain this data and all other data needed to run the RHESSysSensitivityAnalysis.R script. That script is used to compute Morris metrics for these runs. This script does the following.

    Aggregate EEs for Parameters with Dependencies
    Note that some parameters in RHESSys depend on the values of other parameters because of constraints. The elementary effects for these variables must be aggregated. This is completed in this script using the mean(abs(EE)) to aggregate.

    Bootstrap EEs to estimate confidence intervals
    Bootstrapping parameters may be specified by the user. The bootstrapping is completed on the original data, not the aggregated data. This allows for more variation in the aggregated results.

    Make figures for paper
    Placed into directories, as explained in the Smith et al. data repository.