**Primer SQL-database**

(Sep 10, 2013 by Ryan Murphy)

The SoilWat R wrapper generates the same structured data format. The output data can be in comma separated files each corresponding with an output table. The size of these tables, as large as 42 GB, can prohibit any productive use of these tables without further splitting. Further splitting would reduce ones ability to quickly compare different data sets. A sql database referenced here is one that would be useful to a wide range of people without a lot of overhead in the learning process. With a few simple commands, a graphical interface, or a plug in anyone can quickly analyze data.

**Why SQLite**

SQLite is portable, requires no persistent process, no configuration, and is a transactional SQL database engine. The actual database is nothing more then a file on the hard drive. This makes the database easy to move and use on different operating systems. To learn more about SQLite database visit the website [www.sqlite.org](http://www.sqlite.org/).

**Scenario Data**

All the projects have the same structured output. Scenario data is located in one database. The scenario database, which contains all the output tables, file name is “dbTables.db”. The tables within are the aggregation tables from the runs. It has 21 unique tables, each table representing an aggregated output or grouped output. The standard deviation to these tables is also in the database with a similar name. These tables are dependent on options set before simulation runs. The tables with daily values are the majority.

Tables within dbTables.db

---Overall Aggregates--- The variable output\_aggregates is used to turn on/off the columns of this table.

1. **aggregation\_overall\_mean -** There are 51 options available, around 47 was set for these projects. They can be grouped into inputs, climate and weather, climatic dryness, climatic control, yearly water balance, daily extreme values, ecological dryness, mean monthly values, and potential regeneration.

---Daily Tables--- The variable output\_aggregate\_daily is used to turn on or off the following outputs

1. **aggregation\_seasons\_dailyvalues\_aet\_mean**
2. **aggregation\_seasons\_dailyvalues\_deepdrainage\_mean**
3. **aggregation\_seasons\_dailyvalues\_evaporationsoil\_mean**
4. **aggregation\_seasons\_dailyvalues\_evaporationsurface\_mean**
5. **aggregation\_seasons\_dailyvalues\_evaporationtotal\_mean**
6. **aggregation\_seasons\_dailyvalues\_infiltration\_mean**
7. **aggregation\_seasons\_dailyvalues\_pet\_mean**
8. **aggregation\_seasons\_dailyvalues\_rain\_mean**
9. **aggregation\_seasons\_dailyvalues\_runoff\_mean**
10. **aggregation\_seasons\_dailyvalues\_snowfall\_mean**
11. **aggregation\_seasons\_dailyvalues\_snowmelt\_mean**
12. **aggregation\_seasons\_dailyvalues\_snowpack\_mean**
13. **aggregation\_seasons\_dailyvalues\_swa\_mean \***
14. **aggregation\_seasons\_dailyvalues\_swc\_mean \***
15. **aggregation\_seasons\_dailyvalues\_swp\_mean \***
16. **aggregation\_seasons\_dailyvalues\_temperaturemax\_mean**
17. **aggregation\_seasons\_dailyvalues\_temperaturemin\_mean**
18. **aggregation\_seasons\_dailyvalues\_totalprecipitation\_mean**
19. **aggregation\_seasons\_dailyvalues\_transpiration\_mean \***
20. **aggregation\_seasons\_dailyvalues\_vwc\_mean \***

The daily tables are all formatted the same. The first columns until the Scenario column give information about the rows (e.g., site location, elevation, treatment level, etc.). The columns following correspond to a day of the year. The tables marked with an asterisk above relate to an output that describes soil layers. These tables have an additional column in the header indicating the layer.

**Ensemble Data**

The other database files which have a naming schema 'dbEnsemble\_Aggregation \*.db'. The \* represents a table name from above list (e.g. swa, overall, vwc, etc). The name of the database file corresponds to a table. This means that there are 21 different ensemble database files. The tables in that database correspond to the ensemble name and rank level used to generate that table. There are two ensemble families: SRESA2 and SRESB1. There are three ranks: 2,8,15. Three tables are generated for each ensemble family for each rank. This means there should be 18 tables in each ensemble database.

**Getting Started with SQLite**

Download and install the binaries for your OS. Visit [www.sqlite.org/download.html](http://www.sqlite.org/download.html) to download binaries.

If you are planning on using R to view the data, you can install the package RSQLite from the package database. From R the command would be, install.packages(“RSQLite”).

Python also has module to interact with a sqlite database. See <http://docs.python.org/2/library/sqlite3.html>

Ubuntu has a graphical application called sqliteman.

OSX has multiple applications, a comparative list can be found at <http://www.barefeetware.com/sqlite/compare/?mlp>

Windows multiple options as well for a graphical interface. Some are listed below.

<http://sqlitebrowser.sourceforge.net/>

<http://sqliteadmin.orbmu2k.de/>

<http://sqlitestudio.one.pl/>

**SQL basics**

Structured Query Language is an international standard for database manipulation. The following will give a brief overview of getting data out of the database.

SQL is used to Select rows and columns from the database and return them to you.

A basic syntax for selecting all the rows in a table daily.aet is

SELECT \* FROM daily.aet;

“\*” says select all the columns in the table. Using a list of column names in the form (column\_1, column\_2, …..), one could select only the columns needed from the table.

Using WHERE will let you select only the rows that you require.

SELECT \* FROM daily.aet WHERE RunID = 5;

This would select all the rows where the column RunID is equal to five. You can also use these operators, =, <>,>,<,>=,<=, BETWEEN, LIKE, IN.

Select rows between a value.

SELECT \* FROM daily.aet WHERE RunID BETWEEN 5 AND 10;

Select a ensemble family from the table.

SELECT \* FROM daily.aet WHERE Scenario LIKE '%srbsa2%';

Multiple row conditions can be applied by using AND and OR statements.

SELECT P\_id, Label FROM daily.aet WHERE RunID = 10 AND Scenario LIKE '%SRESA2';

**Terminal and SQLite**

After installing sqlite, sqlite3 command will be available in the terminal. Viewing rows on the terminal is not the best way to look at large sets of data. One can use sqlite3 to export the data to a csv file. Once content with the sql output simple enter this into the sqlite3 session.

> sqlite3 'databasename.db'

.mode csv

.header on

.out file.csv

SELECT STATMENT;

If you need the data from multiple database tables, use sqlite's ATTACH statement;

> sqlite3

sqlite> ATTACH 'dbname1.db' AS X;

sqlite> ATTACH 'dbname2.db' AS Y;

SELECT \* FROM X.daily.aet;

SELECT \* FROM Y.daily.swp;

**R and SQLite**

R has a package called RSQlite that makes dealing with SQLite database easy. To install type the following into the R session.

install.packages("RSQLite")

To use the package use the following lines

library(RSQLite)

drv <- dbDriver("SQLite")

con <- dbConnect(drv, "path/to/database/file")

List Tables in the database file

dbListTables(con)

To send a query

res <- dbSendQuery(con, "SQL STATEMENT GOES HERE")

To fetch results, n is the number of rows you want, -1 will give you all selected rows.

data <- fetch(res,-1)

Clear results

dbClearResult(res)

The last three steps can be replaced by dbGetQuery(con, “SQL STATMENT”)

This function combines those steps into one function, but this function lacks the ability to keep results and the ability to return only a certain amount of rows.

**Example:** Compare the output for 'TtoAET\_mean' of all sites in region 1 between current and ensemble scenarios

#First lets load RSQLite package and Connect to the database

library(RSQLite)

drv <- dbDriver("SQLite")

con <- dbConnect(drv, "/home/ryan/Documents/Work/1\_PC\_TempDry\_Simulations\_Prj04\_r1\_Rsoilwat/4\_Data\_SWOutputAggregated/dbTables.db")

#Let see what tables are in the database

Tables <- dbListTables(con)

Tables

[1] "Aggregation\_Overall\_Mean"

[2] "Aggregation\_Overall\_SD"

[3] "Aggregation\_Seasons\_DailyValues\_AET\_Mean"

[4] "Aggregation\_Seasons\_DailyValues\_AET\_SD"

[5] "Aggregation\_Seasons\_DailyValues\_DeepDrainage\_Mean"

[6] "Aggregation\_Seasons\_DailyValues\_DeepDrainage\_SD"

[7] "Aggregation\_Seasons\_DailyValues\_EvaporationSoil\_Mean"

[8] "Aggregation\_Seasons\_DailyValues\_EvaporationSoil\_SD"

[9] "Aggregation\_Seasons\_DailyValues\_EvaporationSurface\_Mean"

[10] "Aggregation\_Seasons\_DailyValues\_EvaporationSurface\_SD"

[11] "Aggregation\_Seasons\_DailyValues\_EvaporationTotal\_Mean"

[12] "Aggregation\_Seasons\_DailyValues\_EvaporationTotal\_SD"

[13] "Aggregation\_Seasons\_DailyValues\_Infiltration\_Mean"

[14] "Aggregation\_Seasons\_DailyValues\_Infiltration\_SD"

[15] "Aggregation\_Seasons\_DailyValues\_PET\_Mean"

[16] "Aggregation\_Seasons\_DailyValues\_PET\_SD"

[17] "Aggregation\_Seasons\_DailyValues\_Rain\_Mean"

[18] "Aggregation\_Seasons\_DailyValues\_Rain\_SD"

[19] "Aggregation\_Seasons\_DailyValues\_Runoff\_Mean"

[20] "Aggregation\_Seasons\_DailyValues\_Runoff\_SD"

[21] "Aggregation\_Seasons\_DailyValues\_SWAatSWPcrit1500kPa\_Mean"

[22] "Aggregation\_Seasons\_DailyValues\_SWAatSWPcrit1500kPa\_SD"

[23] "Aggregation\_Seasons\_DailyValues\_SWAatSWPcrit3000kPa\_Mean"

[24] "Aggregation\_Seasons\_DailyValues\_SWAatSWPcrit3000kPa\_SD"

[25] "Aggregation\_Seasons\_DailyValues\_SWAatSWPcrit3500kPa\_Mean"

[26] "Aggregation\_Seasons\_DailyValues\_SWAatSWPcrit3500kPa\_SD"

[27] "Aggregation\_Seasons\_DailyValues\_SWAatSWPcrit3900kPa\_Mean"

[28] "Aggregation\_Seasons\_DailyValues\_SWAatSWPcrit3900kPa\_SD"

[29] "Aggregation\_Seasons\_DailyValues\_SWC\_Mean"

[30] "Aggregation\_Seasons\_DailyValues\_SWC\_SD"

[31] "Aggregation\_Seasons\_DailyValues\_SWP\_Mean"

[32] "Aggregation\_Seasons\_DailyValues\_SWP\_SD"

[33] "Aggregation\_Seasons\_DailyValues\_Snowfall\_Mean"

[34] "Aggregation\_Seasons\_DailyValues\_Snowfall\_SD"

[35] "Aggregation\_Seasons\_DailyValues\_Snowmelt\_Mean"

[36] "Aggregation\_Seasons\_DailyValues\_Snowmelt\_SD"

[37] "Aggregation\_Seasons\_DailyValues\_Snowpack\_Mean"

[38] "Aggregation\_Seasons\_DailyValues\_Snowpack\_SD"

[39] "Aggregation\_Seasons\_DailyValues\_TemperatureMax\_Mean"

[40] "Aggregation\_Seasons\_DailyValues\_TemperatureMax\_SD"

[41] "Aggregation\_Seasons\_DailyValues\_TemperatureMin\_Mean"

[42] "Aggregation\_Seasons\_DailyValues\_TemperatureMin\_SD"

[43] "Aggregation\_Seasons\_DailyValues\_TotalPrecipitation\_Mean"

[44] "Aggregation\_Seasons\_DailyValues\_TotalPrecipitation\_SD"

[45] "Aggregation\_Seasons\_DailyValues\_Transpiration\_Mean"

[46] "Aggregation\_Seasons\_DailyValues\_Transpiration\_SD"

[47] "Aggregation\_Seasons\_DailyValues\_VWC\_Mean"

[48] "Aggregation\_Seasons\_DailyValues\_VWC\_SD"

#Let us find out the column names in table Aggregation\_Overall\_Mean

colNames <- dbListFields(con,name=Tables[1]) #index for table is 1 in Tables

colNames

[1] "P\_id"

[2] "RunID"

[3] "Labels"

[4] "ID"

[5] "Region"

[6] "X\_WGS84"

…............

#Let get the header columns. Header is the first columns until Scenario column

headerColumns <- colNames[1:which(colNames == "Scenario")]

#Lets add the header and the value we are going to look at

fetchColumns <- c(headerColumns, colNames[which(colNames=="TtoAET\_mean")])

#Lets build our SQL string

SQL<-paste("SELECT ",paste(fetchColumns,sep="",collapse=",")," FROM ",Tables[1]," WHERE Region=1 AND Scenario='Current';",sep="")

SQL

[1] "SELECT P\_id,RunID,Labels,ID,Region,X\_WGS84,YearStart,SimStartYear,YearEnd,Y\_WGS84,ELEV\_m,Mask\_Current,Mask\_Future,Experimental\_Label,LookupWeatherFolder,PotentialNaturalVegetation\_CompositionShrubsC3C4\_Paruelo1996,PotentialNaturalVegetation\_CompositionShrubs\_Fraction,PotentialNaturalVegetation\_CompositionC3\_Fraction,PotentialNaturalVegetation\_CompositionC4\_Fraction,PotentialNaturalVegetation\_CompositionAnnuals\_Fraction,AdjMonthlyBioMass\_Precipitation,AdjMonthlyBioMass\_Temperature,AdjRootProfile,RootProfile\_C3,RootProfile\_C4,RootProfile\_Shrubs,Vegetation\_TotalBiomass\_ScalingFactor,Vegetation\_Litter\_ScalingFactor,Scenario,TtoAET\_mean FROM Aggregation\_Overall\_Mean WHERE Region=1 AND Scenario='Current';"

#Get the data

ScenarioData <- dbGetQuery(con, SQL)

#Get Ensemble Data

#Create a new connection to the ensemble database

Econ <- dbConnect(drv, "/home/ryan/Documents/Work/1\_PC\_TempDry\_Simulations\_Prj04\_r1\_Rsoilwat/4\_Data\_SWOutputAggregated/dbEnsemble\_Aggregation\_Overall.db")

#Get tables

eTables<-dbListTables(Econ)

eTables

[1] "Ensemble\_SRESA2\_Rank02\_20130520\_PC\_TemperateArid\_Prj04r1\_v49\_Means"

[2] "Ensemble\_SRESA2\_Rank02\_20130520\_PC\_TemperateArid\_Prj04r1\_v49\_SDs"

[3] "Ensemble\_SRESA2\_Rank02\_20130520\_PC\_TemperateArid\_Prj04r1\_v49\_ScenarioRanks"

[4] "Ensemble\_SRESA2\_Rank08\_20130520\_PC\_TemperateArid\_Prj04r1\_v49\_Means"

[5] "Ensemble\_SRESA2\_Rank08\_20130520\_PC\_TemperateArid\_Prj04r1\_v49\_SDs"

[6] "Ensemble\_SRESA2\_Rank08\_20130520\_PC\_TemperateArid\_Prj04r1\_v49\_ScenarioRanks"

[7] "Ensemble\_SRESA2\_Rank15\_20130520\_PC\_TemperateArid\_Prj04r1\_v49\_Means"

[8] "Ensemble\_SRESA2\_Rank15\_20130520\_PC\_TemperateArid\_Prj04r1\_v49\_SDs"

[9] "Ensemble\_SRESA2\_Rank15\_20130520\_PC\_TemperateArid\_Prj04r1\_v49\_ScenarioRanks"

[10] "Ensemble\_SRESB1\_Rank02\_20130520\_PC\_TemperateArid\_Prj04r1\_v49\_Means"

[11] "Ensemble\_SRESB1\_Rank02\_20130520\_PC\_TemperateArid\_Prj04r1\_v49\_SDs"

[12] "Ensemble\_SRESB1\_Rank02\_20130520\_PC\_TemperateArid\_Prj04r1\_v49\_ScenarioRanks"

[13] "Ensemble\_SRESB1\_Rank08\_20130520\_PC\_TemperateArid\_Prj04r1\_v49\_Means"

[14] "Ensemble\_SRESB1\_Rank08\_20130520\_PC\_TemperateArid\_Prj04r1\_v49\_SDs"

[15] "Ensemble\_SRESB1\_Rank08\_20130520\_PC\_TemperateArid\_Prj04r1\_v49\_ScenarioRanks"

[16] "Ensemble\_SRESB1\_Rank15\_20130520\_PC\_TemperateArid\_Prj04r1\_v49\_Means"

[17] "Ensemble\_SRESB1\_Rank15\_20130520\_PC\_TemperateArid\_Prj04r1\_v49\_SDs"

[18] "Ensemble\_SRESB1\_Rank15\_20130520\_PC\_TemperateArid\_Prj04r1\_v49\_ScenarioRanks"

#Look at SRESA2 rank 08. We can reuse the column names we made but replace Scenario with EnsembleName and add Level

fetchColumns<-gsub(pattern="Scenario", replacement="EnsembleName", x=fetchColumns)

fetchColumns<-c(fetchColumns[1:which(fetchColumns=="EnsembleName")],"Level",fetchColumns[-(1:which(fetchColumns=="EnsembleName"))])

eSQL <- paste("SELECT ",paste(fetchColumns,sep="",collapse=",")," FROM ",eTables[1]," WHERE Region=1;",sep="")

#Now get our Data

EnsembleData <- dbGetQuery(Econ,eSQL)

#Now we can compare a scenario run to the ensemble data

> ScenarioData$TtoAET\_mean[1]

[1] 0.2166366

> EnsembleData$TtoAET\_mean[1:5]

[1] 0.1817833 0.1506032 0.1984648 0.1759483 0.1599423

#Write that that data out to file with header information.

write.csv(ScenarioData,file="ScenarioData\_TtoAET\_mean.csv")

write.csv(EnsembleData,file=”EnsembleData\_TtoAET\_mean.csv”)