Texas Power Generation Capacity Expansion Solutions Mapping App Documentation

Table of Contents

[Abstract 1](#_Toc482401725)

[1. Introduction 2](#_Toc482401726)

[2. Data cleaning 2](#_Toc482401727)

[2a. Reading from the data file 2](#_Toc482401728)

[2b. Reading the solutions file 3](#_Toc482401729)

[2c. Connecting to tech & county index 3](#_Toc482401730)

[2d. Removing redundant columns 4](#_Toc482401731)

[2e. Aggregating values 4](#_Toc482401732)

[3. Joining attribute data to map file 4](#_Toc482401733)

[4. Leaflet functions 4](#_Toc482401734)

[5. Shiny Interface 4](#_Toc482401735)

# Abstract

The purpose of this document is to explain how an app was built to present solutions of the electricity capacity expansion problem for Texas. This app uses the programming language, R and Shiny, a web framework for making R apps. The solutions are plotted on a map using the package, Leaflet.

The electricity capacity expansion problem was formulated as a bi-objective linear program and solved using CPLEX. The solutions are written to a file which are read by the app and presented in the form of an interactive interface.

The interface allows a user to select the kind of plot to display from:

1. Technology – which technology should be used to generate electricity in a county
2. Capacity – total power to be generated in a county across technologies
3. Cost – Amount to be spent by a county i.e. budget allocation
4. Water Utilization – The ratio of recommended water usage to water overhead in a county

The user can see the aforementioned plots as recommended by the following solutions:

1. Cost Optimal – The solution that optimizes the overall cost of capacity expansion
2. Robust – The robust solution i.e. one that is least affected by implementation uncertainty
3. Water Optimal – The solution that optimizes the overall water usage because of the capacity expansion

# Introduction

Jornada, Leon (2015) formulate the requirements of electric capacity expansion for the state of Texas considering energy requirements in the year 2040 as a multi-objective linear program with 2 objectives viz. cost and water utilization. This is then solved using the ∊-constraint method. With every iteration of this method a new optimal solution is generated. The solutions generated are then checked for robustness against changes that may occur in the implementation phase.

While the mentioned paper considers only 33 locations in Texas, the problem used to write this document has been solved for all 254 counties in Texas. Each solution answers the following questions

1. Which counties should a power plant be built in?
2. What technology should the power plant use to generate electricity?
3. What should be the capacity of the power plant?

This is described by the variable which is the designed capacity (MW) of a power plant at location i using technology j.

# Data cleaning

## 2a. Reading from the data file

The data file used to solve the linear program is a useful starting point for the analysis since it has the coefficients of the objective function and constraints. These will be used later in calculating costs and water utilization rates in each county. This document will use a file named “SPP\_Model\_v2.DAT” as an example.

DataFile <- scan("SPP\_Model\_v2.DAT")

The data file is structured such that the first three lines specify the number of locations (counties - 254), number of technologies (9) and number of dispatchable technologies (6) respectively.

locationsCount <- DataFile[1]

techCount <- DataFile[2]

dispTechCount <- DataFile[3]

Following this are 2 tables of equal dimensions - locationsCount X techCount i.e. 254 x 9. The first table has the normalized cost per location (row) and technology (column) times capacity factor per location and technology. The second table has the water usage per technology times capacity factor per location (row) and technology (column). These two tables can be merged into a single dataframe for future analysis using a single instance of a nested for loop.

The “nextTable” variable ensures that 2 data points can be read in one iteration which are at a distance of 254 x 9 points. i.e. one entire table away from each other. The first point is stored as the cost and the second as the water utilization rate. This way one single row of a dataframe is built and bound to the existing dataframe. Let the dataframe be named “CostWater”.

CostWater <- data.frame()

nextTable <- locationsCount\*techCount

for(ID in 1:locationsCount){

for(j in 1:techCount){

pos <- techCount\*(ID-1) + j + 3

newrow <- c(ID,j-1,DataFile[pos], DataFile[nextTable + pos])

CostWater <- rbind(CostWater, newrow)

}

}

names(CostWater) <- c("ID", "j\_index", "CostRate", "WaterUtilRate")

The columns "ID", "j\_index", "CostRate", "WaterUtilRate" indicate the county, technology, the normalized cost per location per technology and the normalized water usage per technology per location times capacity factor respectively.

The rest of the data file is not used in further analysis and is hence ignored.

## 2b. Reading the solutions file

The solutions created by CPLEX are written to a file. These are in the form of a table with 5 columns. This can be checked as follows. This document uses the file “VarVals.sol” as an example.

df <- as.data.frame(read.table("VarVals.sol", header = T))

head(df)

Output:

Sol\_index X\_vals i\_index j\_index robust

1 0 347.9161 54 0 1

2 0 514.3200 135 0 1

3 0 71.0865 138 0 1

4 0 1600.2957 188 0 1

5 0 954.0028 194 0 1

6 0 1047.6607 239 0 1

As the name suggests, the Sol\_index is the number of the solution created by CPLEX. In this file there are 139 solutions. The x\_vals, i\_index and j\_index are the parts of the variable defined earlier which is the designed capacity (MW) of a power plant at location i using technology j. Thus, x\_vals is the designed capacity of a power plant in location denoted by i\_index using technology denoted by j\_index.

## 2c. Connecting to technology & county names

The dataframe “df” created in the previous step is not very easy to understand since all the information is presented in the form of just numbers. These numbers denote some factual information i.e. the technology names and county names that need to be connected to the dataframe in order to make sense of the data.

The mapping of “j\_index” to names of technology is done in the CSV file “techData.csv”. Hence, this file is read and merged with the dataframe “df” using the common column “j\_index”.

techData <- read.csv("techData.csv", header = TRUE)

df <- merge(df, techData, by = "j\_index")

The mapping of “i\_index” to names of counties is done in the CSV file “County\_Overheads.csv”. This is read into the “counties” dataframe.

counties <- read.csv("County\_Overheads.csv", header = TRUE)

head(counties)

Output:

ID County.Name Overhead FIPS

1 1 Anderson 89249.462 48001

2 2 Andrews 4666.090 48003

3 3 Angelina 134167.294 48005

4 4 Aransas 0.000 48007

5 5 Archer 11974.662 48009

6 6 Armstrong 8120.483 48011

However, the indices in this file associated with county names are 1 greater than i\_index and are in the column named “ID”. This is handled simply by creating a column named “ID” in “df” that is i\_index + 1.

df$ID <- df$i\_index+1

df <- merge(df, counties, by = "ID", all.y = TRUE)

Some counties are not present in any solution. i.e. no power plant is to be built in these. However, while merging, the above code will add rows from the “counties” dataframe with <NA> values in columns for counties which are not present in the solutions.

Finally, the data file read into the dataframe “CostWater” can be merged to the dataframe “df” now.

df <- merge(df, CostWater, by = c("ID","j\_index"), all.x = TRUE)

This will add data from the “CostWater” data frame to rows with unique combinations of “ID” and “j\_index” in “df” matching those in the “CostWater” dataframe.

## 2d. Removing redundant columns

While reading the solution and data file, the columns, ID, i\_index, j\_index were useful. However, ID, i\_index and FIPS are essentially the same column represented differently. Since FIPS has significance in the real-world as it is a standard used by the US government, it will be kept while the other 2 columns will be removed. Similarly, j\_index is redundant since the names of technologies are already present in the dataframe. Also, the “Overhead” column inherited from the “counties” dataframe is not needed right now.

df$Overhead <- NULL

df$ID <- NULL

df$i\_index <- NULL

df$j\_index <- NULL

While merging dataframes, some string columns get stored as factors. These need to be identified and converted to strings i.e. the data type “character” in R.

i <- sapply(df,is.factor)

df[i] <- lapply(df[i], as.character)

## 2e. Calculating cost and water columns

Using the data read from data file and solution file, the cost and water utilization for each combination of county technology and solution.

df$Cost <- df$CostRate\*df$X\_vals

df$WaterUtil <- df$WaterUtilRate\*df$X\_vals

## 2f. Aggregating values

In order to plot the solutions at a county level, the data read from solutions and calculated using the data file must be aggregated at a county level.

For example, a solution can recommend a county have a power plant each using solar, wind and coal. This must be represented uniquely on the map differently from a county that only has a single power plant using wind. This can be achieved using the aggregate function.

df<-df[order(df$tech, df$Sol\_index),]

techlist <- aggregate(tech ~ Sol\_index+FIPS+County.Name+robust, df, FUN=paste, collapse = ', ')

Before the technology names are aggregated, the dataframe is sorted by “tech”. This is done to avoid aggregating different permutations of same technologies as different values. Eg. if Solar PV appears before Biomass for a county but in the other order for a different county, these would be plotted as different technologies. The sorting avoids this since the technologies can only appear in alphabetical order and only one combination will be created. In the example of Biomass and Solar, the combination will always be “Biomass, Solar PV” and never the other way round.

head(techlist)

Output:

Sol\_index FIPS County.Name robust tech

1 135 48001 Anderson 0 Solar PV

2 135 48003 Andrews 0 Solar PV

3 135 48005 Angelina 0 Biomass, Solar PV

4 135 48009 Archer 0 Solar PV

5 135 48011 Armstrong 0 Solar PV

6 135 48013 Atascosa 0 Solar PV

This table shows that the solution 135 recommends Biomass and Solar PV power plants in the county Angelica while just Solar PV in Anderson, Andrews, Archer.

Similarly, the columns “Cost”, “X\_vals” (capacity) and “WaterUtil” can be aggregated. The values in the “tech” column are strings. Hence it was aggregated using the paste function. However, the other columns are numeric and hence will be aggregated using the sum function.

caplist <- aggregate(X\_vals ~ Sol\_index+FIPS+County.Name+robust, df, FUN=sum)

costlist <- aggregate(Cost ~ Sol\_index+FIPS+County.Name+robust, df, FUN=sum)

waterlist <- aggregate(WaterUtil ~ Sol\_index+FIPS+County.Name+robust, df, FUN=sum)

Finally, what remains, is to join all these columns in one single data frame known as “newdf”

newdf <- cbind(techlist, caplist$X\_vals, costlist$Cost, waterlist$WaterUtil)

names(newdf)[names(newdf) == "caplist$X\_vals"] <- "Capacity"

names(newdf)[names(newdf) == "costlist$Cost"] <- "Cost"

names(newdf)[names(newdf) == "waterlist$WaterUtil"] <- "WaterUtilization"

head(newdf)

Output:

FIPS Sol\_index County.Name robust tech Capacity Cost WaterUtilization

1 48001 135 Anderson 0 Solar PV 48 338.6091 72

2 48001 118 Anderson 1 Solar PV 48 338.6091 72

3 48001 119 Anderson 1 Solar PV 48 338.6091 72

4 48001 120 Anderson 1 Solar PV 48 338.6091 72

5 48001 121 Anderson 1 Solar PV 48 338.6091 72

6 48001 122 Anderson 1 Solar PV 48 338.6091 72

# 

The water utilization is not a good measure across counties. In some counties, the water utilization might be high but be only a fraction of the overhead water available. In order to normalize this column, the “Overheads” column from “counties” dataframe is re-introduced to the dataframe “newdf” and a new column “WaterUtilized” is calculated as the ratio of water utilization and overhead.

newdf <- merge(newdf, counties[,c("FIPS","Overhead")], all.x = TRUE, by = "FIPS")

newdf$WaterUtilized <- newdf$WaterUtilization/newdf$Overhead

# Joining attribute data to map file

Now that the data is ready in a format that has only a single entry against a county within a solution, this can be shown in a map. The Texas map is downloaded as a SpatialPolygonsDataFrame called “Texas.Rda”. The library sp is needed to read this.

library(sp)

load("Texas.Rda")

In this map file, information such as latitude, longitude, county name etc. is stored in a format that R can understand. The FIPS codes in this file are stored as characters. Hence, the column “FIPS” in newdf also needs to be modified to make the merging of the data and map seamless.

newdf$FIPS <- as.character(newdf$FIPS)

To plot a map for a specific solution, that solution’s data needs to be added to the map data. The user-defined function makeSolMap does that.

makeSolMap <- function(solNumber){

texasSolMap <- merge(Texas, newdf[newdf$Sol\_index==solNumber,], by = "FIPS", all.x = TRUE)

return(texasSolMap)

}

# Leaflet functions

As discussed earlier, the user can request for 4 kinds of plots viz. Technology, Capacity, Cost and Water Utilization. What this means is that selecting one these is a call to one of the following user-defined functions respectively.

## 4a. drawSolTechMap

## 4b. drawSolCapMap

## 4c. drawSolCostMap

## 4d. drawSolWaterMap

# 

# Shiny Interface

Since the user can plot three solutions viz. Cost Optimal, Robust and Water Optimal, the solution indices of these can be saved in a vector.

The cost optimal solution is simply the first solution or one with Sol\_index = 0

costOptimalSol <- 0

Till now, the column, “robust” has not been discussed. The robust solution is one which has 0 in the “robust” column. The index for this can be found simple by finding all rows in “newdf” that have the value 0 in the “robust” column and checking the solution index for these rows. The added advantage of this is that if there are more than one robust solutions, all of them are captured.

robustSols <- unique(newdf[newdf$robust==0 & !is.na(newdf$robust),]$Sol\_index)

The water optimal solution is simply the last solution or the one with the largest value of Sol\_index.

waterOptimalSol <- max(newdf$Sol\_index, na.rm = TRUE)

Saving these in a vector with clearly defined names.

solsToPlot <- c(costOptimalSol, robustSols, waterOptimalSol)

names(solsToPlot) <- c("Cost Optimal", "Robust", "Water Optimal")