water

MDPI

Supplementary Material

A GIS-Based Fit for the Purpose Assessment of Brackish Groundwater Formations as an Alternative to Freshwater Aquifers

Abdullah Karim, Marangely Gonzalez Cruz, Elma A. Hernandez and Venkatesh Uddameri *

Department of Civil, Environmental and Construction Engineering, Texas Tech University, Lubbock, TX 79409-1023, USA

* Correspondence: venki.uddameri@ttu.edu

Table S1. Water Quality Standards for Municipal and Hydraulic Fracturing Use (Data from [61] and [50]; Secondary standards for Municipal Use (SMCLs) are presented in parenthesis; All units are mg/L unless noted otherwise).

	Municipal	**Oil and Gas (Hydraulic Fracturing)**	
Constituents	**MCL (SMCL)**	**Gel-Based HF**	**Slickwater HF**
pH	(6.5, 8.5)	(6.0, 8.0)	
Total dissolved Solids (TDS)	(500)	50000	150000
TH (Total Hardness) mg CaCO3/L	75	1000	
Chloride (Cl)	(250)		
Fluoride (F)	4 (2)		
Bicarbonate (HCO3)	NA	600	
Carbonate (CO3)	NA	200	30000
Nitrate-Nitrogen (NO3)	10		
Sulfate (SO4)	(250)	200	30000
Aluminum (Al)	(0.05)		
Antimony (Sb)	0.006		
Arsenic (As)	0.01		
Boron (B)	NA	10	
Barium (Ba)	2	10	30000
Beryllium (Be)	0.004		
Calcium (Ca)	NA	200	
Cadmium (Cd)	0.005		
Chromium (Cr)	0.1		
Copper (Cu)	1.3 (1)		
Iron (Fe)	(0.3)	10	130
Lead (Pb)	0.015		
Manganese (Mn)	(0.05)		
Sodium (Na)	NA	36000	
Selenium (Se)	0.05		
Dissolved Silica (Si)	NA	35	

Strontium (Sr)	NA	10	30000
Silver (Ag)	(0.1)		
Thallium (Tl)	0.002		
Uranium (U)	0.03		
Zinc (Zn)	(5)		
Alpha particles (Alpha)	15 [pCi/L]		
Beta particles (Beta)	50 [mrem/year]		
Radium Ra = Ra-226 and Ra-228 (combined)	5 [pCi/L]		
Langelier Saturation index (LSI)	(0.1) [Dim]		

Table S2. Water Quality Criteria for Agricultural FFP Assessment [66,67].

Constituents	Crop Specific Standard Limits			
	Corn	Sorghum	Cotton	Winter Wheat
Boron (B)	2.00	3.00	3.00	3.00
Chloride (Cl)	533	710	710	
Electrical Conductivity (EC) [uS/cm]	1,100	1,700	5,100	4,000
Sodium (Na)	533	710	710	
Sodium Absorption Ratio (SAR)	10	10	10	13
Total Dissolved Solids (TDS)	704	1,088	3,264	2200

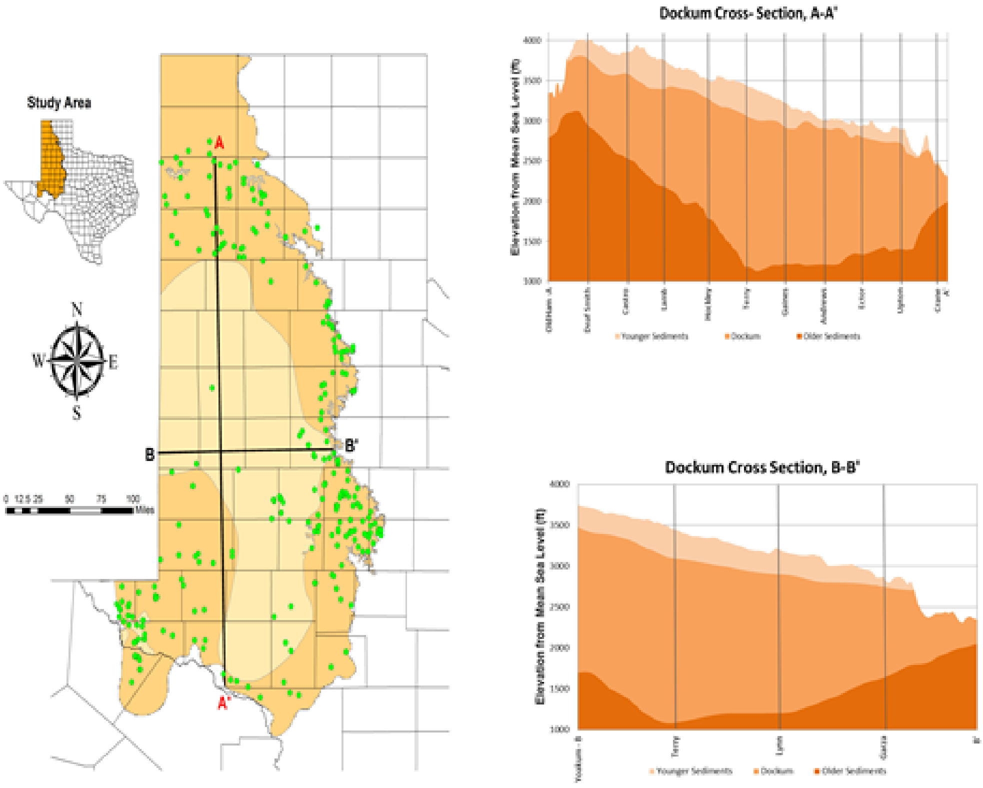

Figure S1: Cross-Sectional Profiles of Dockum HSG

R Code to Perform Natural Neighbor Interpolation

```
R Code to Perform Natural Neighbor Interpolation
# Written by Venki Uddameri and Abdullah Karim

# Cleanup the memory of any variables
rm(list=ls(all=TRUE))

# Function to perform natural neighbor interpolation
# X is a vector of spatial coordinate where interpolation is necessary (sent in as a vector)
# Y is data-frame of spatial coordinates of monitoring wells and the water quality parameters
natneighbor <- function(X,Y)
{
# Parse the datasets
xnew <- X[1]   # x-coordinate
ynew <- X[2]   # y-coordinate
Xa <- cbind(xnew,ynew)   #bind coordinates
coords <- Y[,1:2]   # coordinates where parameter has been measured
param <- Y[,3]   # values of the parameter

# Calculate the base voronoi polygon
zz <- SpatialPoints(coords,proj4string=CRS("+proj=aea +lat_1=20 +lat_2=60 +lat_0=40
+lon_0=-96 +x_0=0 +y_0=0 +ellps=GRS80 +datum=NAD83 +units=m +no_defs"),
bbox = NULL)   #create a spatial points object
vor.base <- voronoi(zz)   # calculate Voronoi polygon

# Bind the data and create a new voronoi polygon
coords.new <- rbind(coords,Xa)   # bind the data point of interest
zz.new <- SpatialPoints(coords.new,proj4string=CRS("+proj=aea +lat_1=20 +lat_2=60
+lat_0=40 +lon_0=-96 +x_0=0 +y_0=0 +ellps=GRS80 +datum=NAD83 +units=m
+no_defs"),
bbox = NULL)   # create a new spatial point object
vor.new <- voronoi(zz.new) # compute the new Voronoi polygon

vor.newx <- vor.new[nrow(coords.new),]
vor.newx <- gBuffer(vor.newx, byid=TRUE, width=0.001)   # create a small buffer to avoid
# intersection of any zero areas

# Intersect the polygon of the new point and extract the corresponding obs. points
vor.int <- gIntersection(vor.newx,vor.base,byid = TRUE)
```

```
# Calculate the area of the intersected polygon
vor.area <- gArea(vor.int,byid=T)

# Setup the dataframe and compute the weighted mean
id <- cover(vor.newx,vor.base)$id
id <- id[-1]   # remove the first point
area <- as.vector(vor.area)
paramx <- param[id]
pred <- sum(paramx*area)/sum(area)

results <- cbind(Xa,pred)   #bind the coordinates and the interpolated value
return(results)
}

# Step 2: Set Working Directory and Read Data
setwd('.\\R')   #need to change as necessary
fnpts <- read.csv('Fish_1mile.csv')   #read datapoints where interpolation is needed
pts <- fnpts[,2:3] #extract the X and Y coordinates
#Read file containing monitoring well coordinates and water quality values of a parameter
data <- read.csv('alk_tot.csv') #total alkalinity is an example
#Call the snowfall library for multicore processing
library(snowfall)

# Extract values of coordinates and parameters
##################################################
for(i in seq(4,ncol(data),1))   # skip first 3 columns which contain wellID and coordinates
{
a<-data[!is.na(data[,i]),]   # remove any NA values
xcoord <- a[,2]   # extract 2nd column whjch has X coordinate
ycoord <- a[,3]   # extract 3rd column which has Y coordinate
param <- a[,i]   #Extract water quality parameters one at a time from 4 – last column
obs <- cbind(xcoord,ycoord,param)   # bind coordinates and WQ parameter

no_cores <- 6     #Initialize cores on a multicore windows machine
cl <- makeCluster(no_cores)   # Make clusters
clusterEvalQ(cl, library(dismo))   # load dismo library into each cluster
clusterEvalQ(cl, library(sp))   # load sp library into each cluster
clusterEvalQ(cl, library(rgeos)) # load rgeos library into each cluster
clusterEvalQ(cl, library(rgdal)) # load rgdal library into each cluster


pred.bicarb <- parRapply(cl = cl, pts, natneighbor,obs)   # call natneighbor function
```

```
fin <- data.frame(X=pred.bicarb,number=rep(1:3, length(pred.bicarb)/3)) # compile data
                                                                         #from all cores
result<-unstack(fin, X~number)   # Unstack the data

stopCluster(cl)   # stop the cluster

#Write the output to the csv file
write.csv(result, paste0("interpolated_",colnames(data)[i], ".csv"), row.names=FALSE)
}
# Plot the vornoi tessellation.
plot(vor.new)
```