An Australian Government Initiative
Funded through the Murray–Darling Water and Environment Research Program

Australia's National Science Agency

MD-WERP Deliverable T.8b.3

# Scripts and tools developed for trend and cluster analysis, GDE analysis, salinity mapping and time series analysis

Project RQ8b: Groundwater as an adaptation option to current water resources management

26 May 2023

Citation

Scripts and tools developed for trend and cluster analysis, GDE analysis, salinity mapping and time series analysis. MD-WERP Deliverable T2.8b2, CSIRO, Australia.

# Contents

# Figures

# Tables

# Executive summary

This report outlines the progress made in developing scripts and tools for analysing extraction bores per SDL and preparing data for various analyses, including time series analysis, groundwater level trend analysis, cluster analysis, analysis of groundwater-dependent ecosystems (GDEs) classes from the BOM GDE Atlas, and salinity mapping. These tools were created as part of the second year deliverables for the MD-WERP Project RQ8b, which aims to explore groundwater as an adaptation option for current water resources management in the Murray-Darling Basin (MDB).

The scripts have been documented and implemented in open-source environments such as Python and R to ensure maximum reproducibility and traceability, and they are thoroughly commented on to aid interpretation, use, and adaptation to other analyses. Please note that these tools are provided "as is" and should only be used for their intended purposes.

# 1 Introduction

There are three one-year activities in the MD-WERP Project RQ8b: Groundwater as an adaptation to current water resources management. The overarching objective is to identify where the opportunities for groundwater to help improve water management in the MDB are located, what these opportunities might look like and how to implement them.

In Year 1, Activity 8b.1 improved our understanding of groundwater level trends, groundwater use patterns and resilience, stress and sustainability aspects of the main alluvial aquifers of the Murray-Darling Basin (MDB). Activity 8b.2 in Year 2 identified and assessed potential opportunities, such as managed aquifer recharge and evaporation savings, for enhancing water supply across priority aquifers.

As part of Year 2 activities, the project team developed a series of scripts and analysis tools, which are reported here as part of the deliverable for MD-WERP RQ8b agreed for the second year. This report summarises the tools developed to perform different analyses, such as data preparation for time series analysis, GDE data extraction and analysis from the BOM GDE Atlas, hierarchical and unsupervised clustering analysis of groundwater levels and salinity mapping in the main alluvial aquifers of the MDB.

## 1.1 Scope of RQ8b: Groundwater as an adaptation option to current water resources management

Groundwater accounts for about 13% of total water use in the MDB from 2012-2019. Eight alluvial aquifer systems (equivalent to 19 groundwater Sustainable Diversion Limit (SDL) resource units) account for 75% of the total groundwater use in the Basin (Figure 1). Almost 75% of the groundwater use in these alluvial systems occurs in New South Wales. This report documents the tools used to analyse groundwater SDLs utilised to manage the main alluvial aquifers of the MDB studied in RQ8b.

During the second year, a series of activities were implemented as part of RQ8b Research Implementation Plan: a) analysis of groundwater-dependent ecosystems (GDEs) to calculate GDE diversity metrics; b) hierarchical and unsupervised clustering (through Self-Organising Maps, SOMs), to disentangle the spatial and temporal patterns obtained from the groundwater level trend analysis (Year 1); and c) salinity mapping in the main alluvial aquifers to account for salinity concentrations in the calculation of the groundwater footprint.

As part of the deliverables for year 2, we are reporting the scripts/tools developed to carry out the activities mentioned above. For maximum reproducibility and traceability, we include in this report the metadata used to implement the scripts described in this report.

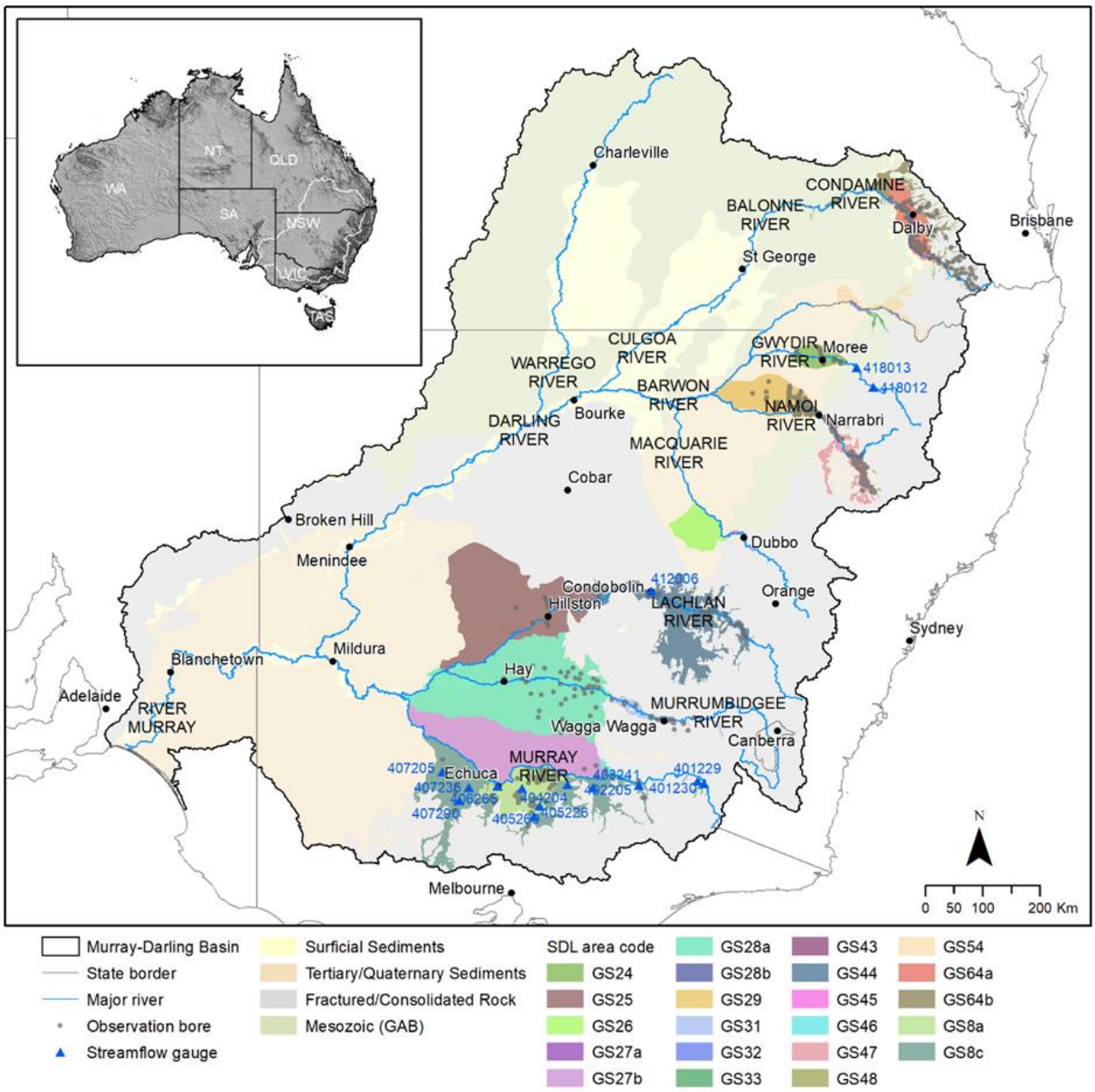


**Figure 1 Groundwater Sustainable Diversion Limits (SDL) resource units used to manage the main alluvial aquifers of the MDB**

# 2 Metadata and scripts

## 2.1 Metadata

Table 1 shows the metadata utilised in the activities implemented in Year 2 of RQ8b and used as data sources for the analyses described in the following sections of this report. Intermediate datasets (e.g., data frames) resulting from the applications of each individual script are reported in the scripted text.

**Table 1 Metadata details for the data sources used in the analysis of GDEs, standing water levels, groundwater salinity and bore locations and drilled depths/dates**

| Data | Description | Source/Access |
|---|---|---|
| • Production bore locations [lat,lon] and drilled dates to obtain a cumulative number of production bores through time<br>• Standing water level (SWL) [m] in 910 observation bores<br>• 95$^{th}$ percentile groundwater salinity [mg/L] value used for spatial interpolation | • Datasets describing the location of production and observation bores and the standing water levels (SWL) between 1971-2021.<br>• For SWLs, bores with at least two records per year for at least 40 years were selected. The selection criteria resulted in a dataset of 910 observation bores.<br>• Groundwater salinity data was selected for 12,513 observation bores with available information. For data QA upper and lower two percent were removed to filter out extreme values (outliers). The 95th percentile salinity was calculated and used for spatial interpolation for each observation bore. | National Groundwater Information System (NGIS) Version 1.7.0.<br>Last updated in July 2021. Last accessed in May 2023.<br>http://www.bom.gov.au/water/groundwater/ngis/ |
| • Regolith thickness [m] | • Bores within the selected groundwater SDL areas were extracted. Regolith thickness estimates, i.e., the thickness of unconsolidated material above bedrock used as a proxy for the thickness of alluvial aquifers, were added from CSIRO Soil and Landscape Grid.<br>• This is Version 2 of the Depth of Regolith product of Australia's Soil and Landscape Grid. The attribute maps are in raster format at a resolution of 3 arc seconds (~90 x 90 m pixels). The regolith is the in situ and transported material overlying unweathered bedrock. Period (temporal coverage; approximately): 1900-2013. | Groundwater SDL Resource units:<br>https://www.mdba.gov.au/publications/mdba-reports/murray-darling-basin-plan-groundwater-methods-report-report-cards<br><br>https://data.gov.au/dataset/ds-dga-66e3efa7-fb5c-4bd7-9478-74adb6277955/details<br>Last updated in August 2021. Last accessed in May 2023.<br><br>CSIRO Soil and Landscape Grid:<br>https://www.clw.csiro.au/aclep/soilandlandscapegrid/ProductDetails-SoilAttributes.html<br><br>Soil and Landscape Grid National Soil Attribute Maps - Depth of Regolith (3" resolution) - Release 2<br>https://doi.org/10.4225/08/55C9472F05295<br>Last updated in March 2018. Last accessed in May 2023. |
| • Areal extents for GDE ecotype classes, eco-hydrogeological zones | • Aquatic and terrestrial GDEs. Aquatic GDEs are categorised by the "potential of an ecosystem to be a GDE" into unknown, low, medium or high.<br>• Indices were based on adaptations of Simpson's and Shannon's diversity indices using areas of GDE types within each SDL area. | GDE indices were based on GDE Atlas version 2 (2019)<br>http://www.bom.gov.au/water/groundwater/gde/<br>http://www.bom.gov.au/water/groundwater/gde/metadata.shtml<br>Last updated in July 2019. Last accessed in May 2023. |

| | | |
|---|---|---|
| | • In corresponding Python scripts, data filtering, selection and processing steps are described in comment blocks. | |

## 2.2 Data preparation for time series analysis

Table 2 shows the script used to extract observation bores with information on standing water levels from the National Groundwater Information System v1.7.0. This script also filters out bores where the reported drilled depth is greater than the depth of regolith reported by Wilford et al. (2018) in the main groundwater SDLs analysed in RQ8b. Bores are assigned to groundwater SDL areas (see Figure 1) based on their coordinates and drilled depth attributes.

**Table 2 Python script to extract and prepare observation data (standing water level) from the NGIS geodatabase for time series trend analysis**

```
### script created and last updated by csiro for use in md werp project
### to prepare data for groundwater level time series analyses

# import required packages and modules
import geopandas
import pandas as pd
from shutil import copyfile
from shapely.geometry import Point

#%% read bore coords extracted from ngis v1.7.0 geodatabases for nsw, qld, vic
### http://www.bom.gov.au/water/groundwater/ngis/
x1 = pd.read_table('nsw_xys.txt',
                   sep=',', usecols=[1,8,9,18,19])
x2 = pd.read_table('qld_xys.txt',
                   sep=',', usecols=[1,8,9,18,19])
x3 = pd.read_table('vic_xys.txt',
                   sep=',', usecols=[1,8,9,18,19])
# append, filter on drilled depth
xy = x1.append([x2,x3])
xy['drill_dept'] = xy[['BoreDepth','DrilledDepth']].max(axis=1)
xy = xy.filter(['HydroID', 'Latitude', 'Longitude', 'drill_dept'])
xy.columns = ['hydroid','lat','long','drill_dept']
# filter where drilled depth is null and bores drilled >1000 m
xy = xy[~xy['drill_dept'].isnull()]
xy = xy[xy.drill < 1000]
xy = xy[xy.drill > 0]

#%% write bore xy shp for geoprocessing
xy['geometry'] = xy.apply(lambda x: Point((float(x.long), float(x.lat))), axis=1)
xy = geopandas.GeoDataFrame(xy, geometry='geometry')
xy.to_file('ngis170_nsw_qld_vic_xys.shp')
copyfile('gda94.prj','ngis170_nsw_qld_vic_xys.prj')
### CLIP TO SDLs EXTRACT DEM & REGOLITH VALUES TO PTS IN ARCGIS
### geoscience aust 1s dem product https://ecat.ga.gov.au/geonetwork/srv/eng/catalog.search#/metadata/72759
### csiro soil and landscape grid https://www.clw.csiro.au/aclep/soilandlandscapegrid/ProductDetails-SoilAttributes.html

#%% read processed pts shp back in
xy = geopandas.read_file('ngis170_mdb_alluv_dem_reg95_sdl.shp')
xy = xy.filter(['hydroid', 'lat', 'long', 'drill_dept', 'dem', 'reg95','SDL_NAME','geometry'])
# filter where drilled depth > regolith thickness
xy = xy[xy.drill_dept <= xy.reg95]
xy = xy.set_index(['hydroid'])

#%% read in levels
nsw_z = pd.read_csv('level_NSW.csv', usecols=[0,3,4,5]).set_index('hydroid')
vic_z = pd.read_csv('level_VIC.csv', usecols=[0,3,4,5]).set_index('hydroid')
qld_z = pd.read_csv('level_QLD.csv', usecols=[0,3,4,5]).set_index('hydroid')
z = nsw_z.append([vic_z, qld_z])
z.columns = ['date','meas_typ','swl']
# keep swl records
z = z[z.meas_typ =='SWL']
z = z.drop(['meas_typ'],axis=1)
# filter out zeros
z = z[z.swl >0]
# extract datetime fields
z['datetime'] = pd.to_datetime(z['date'])
z = z.drop(['date'],axis=1)
z['year'] = pd.DatetimeIndex(z['datetime']).year
z['month'] = pd.DatetimeIndex(z['datetime']).month
z['year_month'] = pd.to_datetime(z['datetime']).dt.to_period('M')
# join levels with coords on indexes (hydroid)
xyz = xy.join(z).dropna()
# drop duplicates keeping first obs per bore id & date
xyz = xyz.drop_duplicates(subset=['hydroid','datetime'], keep='first')
xyz = xyz.sort_values(by=['hydroid','datetime'])
# keep swl <= regolith thickness
xyz = xyz[xyz.swl <= xyz.reg95]
```

```
#%% seperate sdls, reassign overlapping shallow/deep sdls on drilled depth
### based on formation thicknesses in sdl report cards
goul = xyz.loc[xyz['SDL_NAME'] == 'Goulburn-Murray: Sedimentary Plain (GS8c)']
goul = goul[goul.drill_dept > 25]
shep = xyz.loc[xyz['SDL_NAME'] == 'Goulburn-Murray: Shepparton Irrigation Region (GS8a)']
shep = shep[shep.drill_dept <= 25]
goul = goul.append(shep)
gwy = xyz.loc[(xyz['SDL_NAME'] == 'Upper Gwydir Alluvium (GS43)')| (xyz['SDL_NAME'] == 'Lower Gwydir Alluvium (GS24)')]
nam = xyz.loc[(xyz['SDL_NAME'] == 'Upper Namoi Alluvium (GS47)')| (xyz['SDL_NAME'] == 'Upper Namoi Tributary Alluvium (GS48)')|
(xyz['SDL_NAME'] == 'Lower Namoi Alluvium (GS29)')]
macq = xyz.loc[(xyz['SDL_NAME'] == 'Upper Macquarie Alluvium (GS45)')| (xyz['SDL_NAME'] == 'Lower Macquarie Alluvium (GS26)')]
lach = xyz.loc[(xyz['SDL_NAME'] == 'Lower Lachlan Alluvium (GS25)')| (xyz['SDL_NAME'] == 'Upper Lachlan Alluvium (GS44)')]
murrum_ls = xyz.loc[(xyz['SDL_NAME'] == 'Lower Murrumbidgee Shallow Alluvium (GS28a)')]
murrum_ls = murrum_ls[murrum_ls.drill_dept <= 40]
murrum_m =  xyz.loc[(xyz['SDL_NAME'] =='Midâ□□Murrumbidgee Alluvium (GS31)')]
murrum_ld = xyz.loc[(xyz['SDL_NAME'] == 'Lower Murrumbidgee Shallow Alluvium (GS28a)')]
murrum_ld = murrum_ld[murrum_ld.drill_dept > 40]
murrum_ld['SDL_NAME'] = 'Lower Murrumbidgee Deep Alluvium (GS28b)'
murrum = murrum_ld.append([murrum_m,murrum_ls])
murray_u = xyz.loc[(xyz['SDL_NAME'] == 'Upper Murray Alluvium (GS46)')]
murray_ls = xyz.loc[(xyz['SDL_NAME'] == 'Lower Murray Shallow Alluvium (GS27a)')]
murray_ls = murray_ls[murray_ls.drill_dept <= 20]
murray_ld = xyz.loc[(xyz['SDL_NAME'] == 'Lower Murray Shallow Alluvium (GS27a)')]
murray_ld = murray_ld[murray_ld.drill_dept > 20]
murray_ld['SDL_NAME'] = 'Lower Murray Deep Alluvium (GS27b)'
murray = murray_u.append([murray_ls,murray_ld])
con = xyz.loc[(xyz['SDL_NAME'] == 'Upper Condamine Alluvium (Central Condamine Alluvium) (GS64a)')| (xyz['SDL_NAME'] == 'Upper
Condamine Alluvium (Tributaries) (GS64b)')]

#%% write out csv files of swl time series for each main sdl group
### these data are further processed in R to conduct time series analyses
goul.to_csv('goul_alluv_sdl_regolith95_swl_v04.csv')
gwy.to_csv('gwy_alluv_sdl_regolith95_swl_v04.csv')
nam.to_csv('nam_alluv_sdl_regolith95_swl_v04.csv')
macq.to_csv('macq_alluv_sdl_regolith95_swl_v04.csv')
lach.to_csv('lach_alluv_sdl_regolith95_swl_v04.csv')
murrum.to_csv('murrum_alluv_sdl_regolith95_swl_v04.csv')
murray.to_csv('murray_alluv_sdl_regolith95_swl_v04.csv')
con.to_csv('condamine_alluv_sdl_regolith95_swl_v04.csv')
```

## 2.3 Trend analysis

Table 3 shows the R code to detect trend magnitudes and statistical significance using three different methods: Mann-Kendall/Beta-slope, linear regression, and two-period comparison/innovative trend analysis (ITA). These algorithms and formulas can be found in Fu et al. (2022). Data sets are obtained from scripts described in section 2.2 and further analysed as an R data frame.

**Table 3 Script to perform trend analysis on the 910 observation bores with at least 2 records per year for the period 1971-2021**

```
Method 1: Mann-Kendall with two self-defined functions, kendall and betaval
#mann-kendall testing function

kendall<- function(x) {

NN<-length(x)
SVal<-0

for (ii in 1:(NN-1)) {
for (jj in (ii+1):NN) {
if (x[jj]>x[ii]) SVal<-(SVal+1)
if (x[jj]<x[ii]) SVal<-(SVal-1)
}
}

ties<-rle(x)$lengths
ties_sum<-sum(ties*(ties-1)*(2*ties+5))
VarS<-(NN*(NN-1)*(2*NN+5)-ties_sum)/18

if (SVal==0) {ZVal<-0} else {
if (SVal<0) {ZVal<-(SVal+1)/sqrt(VarS)} else {ZVal<-(SVal-1)/sqrt(VarS)}
}

return(ZVal)
} #end of function

#beta value estimation

betaval<- function(x,t) {

NN<-length(x)
beta.val<-rep(NA, NN*(NN-1)/2)
```

```
acc<-0
for (ii in 1:(NN-1)) {
for (jj in (ii+1):NN) {
ind<-acc+(jj-ii)
beta.val[ind]<-(x[jj]-x[ii])/(t[jj]-t[ii])
}
acc<-acc+(NN-ii)
}

return(median(beta.val))

} #end of function

To use it with a time series x = x1, x2 … xn and time/year (for example y7121 <- 1971:2021)
z.value <- kendall(x[!is.na(x)])
beta.magnitue <- betaval(x[!is.na(x)],y7121[!is.na(x)])
The corresponding p-value can be obtained by:
p.value <- pnorm(z.value)

Method 2: simple linear regression with lm function
p.value <- summary(lm(x ~ y7121))$coefficients[2,1]
trend.magnitude <- summary(lm(x ~ y7121))$coefficients[2,4]


Method 3: Two-Period Comparison and Innovative Trend Analysis (ITA). It simply compares mean values between two periods of time
that do not necessarily need to have equal lengths. However, ITA does require equal lengths to explore the trends at different
quantiles. Below code is an example of total of 50 years of data, which can be easily modified with different lengths.

trend.magnitude <- 2*(mean(x[26:50],na.rm=T)-mean(x[1:25],na.rm=T))/50
p.value <- (t.test(x[26:50],y=x[1:25]))$p.value
```

## 2.4 Cluster analysis

Table 4 and Table 5 show the main scripts used for the cluster analysis of the groundwater level trends using the self-organising map (SOM) algorithm. The objective of this analysis is to disentangle the spatial patterns in groundwater level trends to identify the dominant behaviours observed in hydrographs for the groundwater level trends in the main alluvial aquifers of the MDB.

SOM is an unsupervised artificial neural network used for clustering, dimension reduction and visualisation. The algorithm finds similarities in observations from high-dimensional data sets and forms ordered clusters of the data. The SOM is resilient to high levels of missing data and is therefore popular in the environmental sciences (Clark, 2022).

**Table 4 Main script for unsupervised cluster analysis of bore level trends in the MDB alluvial aquifers**

```
# Self-organising map for clustering MDB groundwater bores

#-------------------------------------------
# Required packages

library(tidyverse)
library(kohonen)
library(viridis)
library(zeallot)

# Required function
source("SOMfunc.R")

#-------------------------------------------
# Data


# Depth to groundwater level data:
# matrix with 910 bores * 51 years
load("AnnGWData.RData");
summary(ann.data); head(ann.data); dim(ann.data)

# Lat/long, SDL and drill depth data:
# matrix with 910 bores * 5 descriptors
load("BoreInfo.RData");
summary(bore.info); head(bore.info); dim(bore.info)

# Check for missing data
ann.data.df<-data.frame(ann.data)
colnames(ann.data.df)<-gsub("X","Year", colnames(ann.data.df))
naniar::vis_miss(ann.data.df)
MissingDataPerBore<-rowSums(is.na(ann.data)); hist(MissingDataPerBore);

#-------------------------------------------
# Pre-processing: scale data
# Transform to range 0-1 by column (min=>0, max=>1): (x-min(x)) / (max(x)-min(x))

# Find min and max of each row (ie. min and max measurements for each bore)
```

```
mins_sc <- apply(ann.data, 1, min, na.rm=TRUE);mins_sc; # 1 rows, 2 columns
maxs_sc <- apply(ann.data, 1, max, na.rm=TRUE);maxs_sc;
# Scaling operates on columns - data is transposed, scaled, and then transposed back to scale by bore
ann.data.sc <- t(data.frame(scale(t(ann.data), center = mins_sc, scale = maxs_sc - mins_sc)))
# Check scaling (mins and maxes are now all 0 and 1 respectively)
apply(ann.data.sc, 1, min, na.rm=TRUE);
apply(ann.data.sc, 1, max, na.rm=TRUE);


#-------------------------------------------
# Run the SOM

# Select subset of data if needed
som_data <- ann.data.sc # all data

# Set dimensions of SOM grid
width <- 3;
height <- 2;
nodes <- width*height

# Call the SOM function
c(MDB_ts.som, MDB_codes.som, MDB_codes_long, MDB_data_long, MDB_all, tspernode, node_assigns) %<-% SOMfunc(width, height,
som_data)

#-------------------------------------------
# Post-processing

# Reset node numbering to match hierarchical clustering order
MDB_all_reordered <- MDB_all %>%
  mutate(node_hier=
           if_else(node=="1", 6,
            if_else(node=="2", 4,
             if_else(node=="3", 3,
              if_else(node=="4", 2,
               if_else(node=="5", 5, 1))))); MDB_all_reordered

MDB_codes_long_reordered <- MDB_codes_long %>%
  mutate(node_hier=
           if_else(node=="1", 6,
            if_else(node=="2", 4,
             if_else(node=="3", 3,
              if_else(node=="4", 2,
               if_else(node=="5", 5, 1))))); MDB_codes_long_reordered

# Number of time series allocated to each node
node_hier=c(6,4,3,2,5,1)
tspernode_hier <-  data.frame(tspernode, node_hier) %>% arrange(node_hier)
tspernode_hier

#-------------------------------------------
# Plots with reordered nodes

# Plot prototypes (representative vector of each cluster)
MDB_codes_long_reordered %>%
  mutate(node = as_factor(node_hier)) %>%
  mutate(Year = year + 1970) %>%

ggplot(aes(x=Year, y=WLinv, fill=node))+
  theme_minimal()+
  geom_point(colour = "black", size = 0.5)+
  geom_line(colour="black", linewidth = 0.05) +
  xlab("Year")+
  ylab("Water level (scaled)")+
  ggtitle("Representative groundwater level time series by cluster")+
  scale_color_viridis(discrete = TRUE, option = "D")+
  facet_wrap(~node_hier, ncol = width)+
  geom_text( # add number of time series in each cluster
    data    = tspernode_hier,
    mapping = aes(x = -Inf, y = -Inf, label = ts_count),
    hjust   = -1.5, vjust   = -3,
    size = 3.5, colour = 'grey45')+
  theme(legend.position = "none",
        strip.background = element_blank(),
        strip.text.x = element_blank())


# Plot all observations, by cluster
# Coloured smoothers are added to show trends
MDB_all_reordered %>%
  mutate(node = as_factor(node_hier)) %>% # change to factor for colouring
  mutate(Year = year + 1970) %>%

ggplot(aes(x=Year, y=WLinv, fill=node))+
  theme_minimal()+
  geom_point(color='grey', size=0.5)+
  geom_smooth(aes(color=node),se=FALSE)+
  facet_wrap(~node_hier, ncol = width)+
  geom_text( # add number of time series in each cluster
    data    = tspernode_hier,
    mapping = aes(x = -Inf, y = -Inf, label =ts_count),
    hjust   = - 1.5, vjust   = -3,
    size = 3.5, colour = 'grey45')+
  scale_color_viridis(discrete = TRUE, option = "D")+
  xlab("Year")+
  ylab("Water level (scaled)")+
  ggtitle("Scaled observations with smoothers")+
  theme(legend.position = "none",
        strip.background = element_blank(),
        strip.text.x = element_blank())


#-------------------------------------------
# Errors between bore time series and allocated cluster time series
# (MSE and RMSE by bore)
Errors<-data.frame(matrix(ncol = 3, nrow = dim(som_data)[1]))
colnames(Errors) <- c('node','MSE', 'RMSE'); head(Errors)
```

```
tspernode_hier$node<-as.numeric(tspernode_hier$node)

for (bn in 1:dim(som_data)[1]){ # for each bore used in SOM analysis

  nn<-node_assigns$node[bn]; nn # node this bore is assigned to
  nodedata<-data.frame(t(MDB_codes.som[nn,])); # node time series
  boredata<-data.frame(MDB_ts.som$data[[1]][bn,]); # bore time series

  Errors$node[bn]<-nn;
  Errors$MSE[bn]<-hydroGOF::mse(nodedata, boredata, na.rm=TRUE);
  Errors$RMSE<-sqrt(Errors$MSE)

}; head(Errors, 10)

# Convert to renumbered clusters (to match hierarchical clustering numbers)
Error_SOM <- Errors %>% rowid_to_column("Bore") %>%
             left_join(., tspernode_hier[,c("node","node_hier")], by="node") %>%
             select(c("Bore","node_hier","MSE","RMSE")); head(Error_SOM)

table(Error_SOM$node_hier)

#-------------------------------------------
```

**Table 5 R function MDB_SOM.R for running the self-organising map (SOM) analysis of groundwater level trend patterns in the alluvial aquifers of the MDB**

```
# Function for running self-organising map

SOMfunc <- function(width, height, som_data) {

  set.seed(1010)

  # Build grid
  MDB_ts.grid = somgrid(xdim= width,
                        ydim= height,
                        topo="hexagonal")

  # Run SOM
  MDB_ts.som = som(
    som_data,
    MDB_ts.grid,
    rlen=400, # number of iterations
    alpha=c(0.05,0.01), # learning rate
    maxNA.fraction = 0.7) # maximum allowable missing data per bore

  # Summary
  glimpse(MDB_ts.som)
  print(summary(MDB_ts.som))

  #-------------------------------------------
  # Record prototype time series for each node (code vectors)
  # This is the output of the SOM algorithm

  # Prototypes (patterns at each node)
  MDB_codes.som <- data.frame(MDB_ts.som$codes); # row for each node, column for each year
  head(MDB_codes.som); dim(MDB_codes.som);

  # Number of time series allocated to each node
  tspernode <- data.frame(table(MDB_ts.som$unit.classif)) %>%
               dplyr::rename("node"="Var1",
               ts_count="Freq"); tspernode

  # Reformat to enable plotting of results
  # Add year index, and transpose (row for each year, column for each node)
  MDB_codes_df <- data.frame(t(rbind(year=seq(1:dim(MDB_ts.som$codes[[1]])[2]),
                  MDB_codes.som))) %>%
                  set_names(~(.) %>%
                  str_replace_all("V", "node"));
  head(MDB_codes_df); dim(MDB_codes_df); # 51 years, nodes + index

  # Long format for plotting
  MDB_codes_long <- MDB_codes_df %>%
                    pivot_longer(cols = -c("year"),
                    names_to = "node",
                    values_to = "WL") ; MDB_codes_long

  # Remove word 'node' for plotting
  MDB_codes_long$node<-gsub("node","",as.character(MDB_codes_long$node)) %>%
                       as.numeric(); str(MDB_codes_long);

  # Add column to identify prototypes
  MDB_codes_long <- MDB_codes_long %>% mutate(bore="proto")

  str(MDB_codes_long)

  #-------------------------------------------
  # Record the input data time series mapping to each node

  # Add node assignments to input data
  node_assigns<-data.frame(node=MDB_ts.som$unit.classif); head(node_assigns)
  MDB_data<-data.frame(node=MDB_ts.som$unit.classif,MDB_ts.som$data[[1]]); head(MDB_data)

  # List of bores and their assigned nodes
  data_node_list <-MDB_data %>%
                   rownames_to_column('bore') %>%
                   select(c('bore','node'));head(data_node_list)

  # Add year index, and transpose (51 years, bores + index)
  MDB_data_df <- data.frame(t(rbind(
                                  year=seq(1:dim(MDB_ts.som$data[[1]])[2]),
                                  MDB_ts.som$data[[1]]))
```

```
                );MDB_data_df; dim(MDB_data_df); # 51, 911 (910 bores + year index)

  # Long format for plotting
  MDB_data_long <- MDB_data_df %>%
                      pivot_longer(cols = -c("year"),
                                   names_to = "bore",
                                   values_to = "WL") %>%
                    left_join(data_node_list, by='bore'); MDB_data_long; dim(MDB_data_long)

  # Remove X before bore number
  MDB_data_long$bore <- gsub("X","",as.character(MDB_data_long$bore)) %>% as.character()

  #---------------------------------------------

  # Find cluster membership of a specific bore
  # MDB_clusters %>% filter(bore=="3")

  # List bores in a specific cluster
  # MDB_data %>% filter(node==1) %>% labels

  #---------------------------------------------
  # Invert water levels from 'depth below surface'
  MDB_data_long <- MDB_data_long %>% mutate(WLinv = (1-WL));
  MDB_codes_long <- MDB_codes_long %>% mutate(WLinv = (1-WL));

  #---------------------------------------------
  # Combine input data and node vectors
  MDB_all <- rbind(MDB_data_long, MDB_codes_long); head(MDB_all); dim(MDB_all)
  summary(MDB_all); table(MDB_all$bore, MDB_all$node)


  return(list(MDB_ts.som, MDB_codes.som, MDB_codes_long, MDB_data_long, MDB_all, tspernode, node_assigns))
}
```

Table 6 shows the main script for the hierarchical clustering analysis for the trends in depth-to-water table (DTW) data obtained for the 910 observation bores fulfilling the data selection criteria (at least two records per year for 40 years in the period 1971-2021). The first step consists in normalising the data to subsequently apply statistical clustering techniques (hclust).

**Table 6 Script for hierarchical clustering analysis for comparison against the SOM algorithm**

```
#The first step is to normalize the data given different attributions have different units and magnitudes
#normalized to [0,1]
x.sc <- t(apply(x,1,function(x) (x-min(x,na.rm=T))/(max(x,na.rm=T)-min(x,na.rm=T))))

#The second step is cluster analysis with dist and hcluts functions

#cluster and plotting
hc <- hclust(dist(ann.data.sc))
hc2<-as.dendrogram(hc)
plot(hc2, xlab="",ylab="Dist",leaflab="none",main="Dendrogram of DTW")

#The third step is to output cluster results (cutree function) and plotted by number of cluster k
Category <- cutree(hc,k=6)
x.order <- hc$order
output.k6 <- cbind(Category,x.order,Category[x.order])

source("http://addictedtor.free.fr/packages/A2R/lastVersion/R/code.R")
mycol<-c("#FF6B6B","#4ECDC4", "#556270","#66CC00","#FF00FF","#0066CC","#FF8000","#FFFF00","#CC0066","#00FFFF","#4C9900","#404040")
A2Rplot(hc, k = 6, boxes = FALSE, col.down = mycol[2:11],show.labels= F, main = "DTW Dendrogram (6 groups)")
```

## 2.5 GDE analysis

Table 7 shows the script employed to analyse GDE areal extents for different GDE categories: terrestrial and aquatic, and classify them in different ecotype, sub-ecotype and ecohydrological zone areas. Areal extents are further filtered by GDE potential (unknown, low, intermediate, and high) of connection with groundwater. Areal extents are used to calculate GDE diversity metrics based on the Shannon and Simpson diversity indices (Gorelick, 2006; Spellerberg & Fedor, 2003).

**Table 7 Main script for GDE analysis and calculation of Shannon and Simpson diversity indices**

```
### gde indices for use in aquifer prioritisation study
```

```
# import required packages and modules
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt

#%% read in summary table of gde ecotype, sub-ecotype and ecohydrological zone areas
### by sdl areas generated in arcgis
### based on gde atlas version 2.1 2019 http://www.bom.gov.au/water/groundwater/gde/

# terrestrial gdes
ter = pd.read_excel('merge_sum_terr_s_etype_ehz_gwdep_sdl_v02.xls')
ter.columns = ['oid','sdl','ecotyp','ehz','gwdep','frq','km2','pct']
n_ter_sdl = ter.groupby(['sdl']).nunique().filter(['ecotyp']).reset_index() #count unique ecotyp per sdl

#aquatic gdes
aqu = pd.read_excel('merge_sum_aqu_etype_ehz_gwdep_sdl_v02.xls')
aqu.columns = ['oid','sdl','ecotype','ehz','gwdep','frq','km2','pct']
aqu['ecotyp'] = aqu['ecotype'] + " " + aqu['ehz']
aqu = aqu.drop(['ecotype'],axis=1)
n_aqu_sdl = aqu.groupby(['sdl']).nunique().filter(['ecotyp']).reset_index() #count unique ecotyp per sdl

#stats
aqu_s1 = aqu.groupby(['sdl']).sum().reset_index()
aqu_s1 = aqu_s1.filter(['sdl', 'km2','pct'])
n_aqu_sdl['pct'] = aqu_s1['pct']
n_aqu_sdl.to_csv('n_aqu_sdl_v02.csv')
ter_s1 = ter.groupby(['sdl']).sum().reset_index()
ter_s1 = ter_s1.filter(['sdl', 'km2','pct'])
n_ter_sdl['pct'] = ter_s1['pct']
n_ter_sdl.to_csv('n_ter_sdl_v02.csv')
aqu_s2 = aqu.groupby(['ecotyp']).sum().reset_index()
aqu_s2 = aqu_s2.filter(['ecotyp', 'km2','pct'])
ter_s2 = ter.groupby(['ecotyp']).sum().reset_index()
ter_s2 = ter_s2.filter(['ecotyp', 'km2','pct'])

#merge aqu and ter gdes and filter
x = ter.append(aqu,sort=True)
x['sdln'] = x['sdl'].apply(lambda st: st[st.find("GS"):st.find(")")]) #pull out sdl code
x['sdln'].loc[(x['sdl'] == "Upper Condamine Alluvium (Central Condamine Alluvium) (GS64a)")] = "GS64a"
x['sdln'].loc[(x['sdl'] == "Upper Condamine Alluvium (Tributaries) (GS64b)")] = "GS64b"
x['gwd'] = x['gwdep'].str[:2] #relabel Hi Lo Mo Kn Un
xf = x.loc[(x['gwd'] == 'Hi')| (x['gwd'] == 'Mo')| (x['gwd'] == 'Kn')] #filter out Lo and Unkn
x1 = x.groupby(['sdln','sdl','ecotyp']).sum().reset_index()
x1f = xf.groupby(['sdln','sdl','ecotyp']).sum().reset_index()
n_eco_sdl = x1.groupby(['sdl']).nunique().filter(['ecotyp']) #count unique ecotyp per sdl
x_pct = x.groupby(['sdl']).sum()
n_eco_sdl['pct'] = x_pct['pct']
n_eco_sdl.to_csv('n_eco_sdl_v02.csv')

#%% Simpsons Diveristy Index (D) D = 1-(sum[n(n-1)]/N(N-1))
#   BY ECOTYPE
x1['m2'] = x1['km2'] * 1000000
x1['n-1'] = x1['m2'] - 1
x1['n(n-1)'] = x1['m2'] * x1['n-1']
sim = x1.groupby(['sdl']).sum()
sim['N-1'] = sim['m2'] -1
sim['N(N-1)'] = sim['m2'] * sim['N-1']
sim['D'] = 1 - sim['n(n-1)'] / sim['N(N-1)']
sim['Dn'] = sim['D'] / np.max(sim['D'])
sim['DA'] = sim['D'] * (sim['pct'] * 0.01)
sim['DAn'] = sim['DA'] / np.max(sim['DA'])
sim = sim.sort_values(by=['D'], ascending=False).reset_index().filter(['sdl','km2','pct','D','Dn','DA','DAn',])

#%% plt pct area
D = sim[['sdl','pct']].sort_values(['pct'])
plt.rcParams['font.size'] = '7'
plt.figure(figsize=(13/2.54 , 9/2.54),dpi=200)
plt.barh(D['sdl'], D['pct'], align='center', color='grey')
plt.title("Percent SDL area", ha='right')
plt.xlabel("Percent area")
plt.tight_layout(pad=1)
plt.savefig('gde_pct_area_v02.png')
plt.show()
plt.close()

#%% plt D
D = sim[['sdl','D']].sort_values(['D'])
plt.rcParams['font.size'] = '7'
plt.figure(figsize=(13/2.54 , 9/2.54),dpi=200)
plt.barh(D['sdl'], D['D'], align='center', color='grey')
plt.title("Simpson's diversity index", ha='right')
plt.xlabel("D")
plt.tight_layout(pad=1)
plt.savefig('simpsons_x1_v02.png')
plt.show()
plt.close()

#plt Da
D = sim[['sdl','DA']].sort_values(['DA'])
plt.rcParams['font.size'] = '7'
plt.figure(figsize=(13/2.54 , 9/2.54),dpi=200)
plt.barh(D['sdl'], D['DA'], align='center', color='grey')
plt.title("Simpson's diversity index area weighted", ha='right')
plt.xlabel("D")
plt.tight_layout(pad=1)
plt.savefig('simpsons_x1a_v02.png')
plt.show()
plt.close()

#%% Simpsons Diveristy Index (D) D = 1-(sum[n(n-1)]/N(N-1))
#   BY ECOTYPE AND FILTERED BY GWDEP
x1f['m2'] = x1f['km2'] * 1000000
x1f['n-1'] = x1f['m2'] - 1
x1f['n(n-1)'] = x1f['m2'] * x1f['n-1']
simf = x1f.groupby(['sdl']).sum()
simf['N-1'] = simf['m2'] -1
simf['N(N-1)'] = simf['m2'] * simf['N-1']
```

```
simf['Df'] = 1 - simf['n(n-1)'] / simf['N(N-1)']
simf['Dfn'] = simf['Df'] / np.max(simf['Df'])
simf['DAf'] = simf['Df'] * (simf['pct'] * 0.01)
simf['DAfn'] = simf['DAf'] / np.max(simf['DAf'])
simf = simf.sort_values(by=['Df'], ascending=False).reset_index().filter(['sdl','km2','pct','Df','Dfn','DAf','DAfn'])
simf.columns = ['sdl','km2_f','pct_f','Df','Dfn','DAf','DAfn']

# plt pct area
D = simf[['sdl','pct_f']].sort_values(['pct_f'])
plt.rcParams['font.size'] = '7'
plt.figure(figsize=(13/2.54 , 9/2.54),dpi=200)
plt.barh(D['sdl'], D['pct_f'], align='center', color='grey')
plt.title("Percent SDL area excl. low, unknown GDE potential",ha='right')
plt.xlabel("Percent area")
plt.tight_layout(pad=1)
plt.savefig('gde_pct_area_f_v02.png')
plt.show()
plt.close()

# plt Df
D = simf[['sdl','Df']].sort_values(['Df'])
plt.rcParams['font.size'] = '7'
plt.figure(figsize=(13/2.54 , 9/2.54),dpi=200)
plt.barh(D['sdl'], D['Df'], align='center', color='grey')
plt.title("Simpson's diversity index excl. low, unknown GDE potential",ha='right')
plt.xlabel("D")
plt.tight_layout(pad=1)
plt.savefig('simpsons_x1f_v02.png')
plt.show()
plt.close()

# plt DAf
D = simf[['sdl','DAf']].sort_values(['DAf'])
plt.rcParams['font.size'] = '7'
plt.figure(figsize=(13/2.54 , 9/2.54),dpi=200)
plt.barh(D['sdl'], D['DAf'], align='center', color='grey')
plt.title("Simpson's diversity index area weighted\nexcl. low, unknown GDE potential",ha='right')
plt.xlabel("D")
plt.tight_layout(pad=1)
plt.savefig('simpsons_x1af_v02.png')
plt.show()
plt.close()

#%% Shannons (H) H = -∑[(pi) × ln(pi)], pi = n/N
#   BY ECOTYPE
N = x1.groupby(['sdl']).sum().reset_index()
N = N.filter(['sdl','m2']).set_index('sdl')

n = x1.set_index('sdl').join(N, on='sdl', rsuffix='_tot')
n['pi'] = n['m2'] / n['m2_tot']
n['logpi'] = np.log(n['pi']) * -1
n['pi_logpi'] = n['pi'] * n['logpi']

shan = n.groupby(['sdl']).sum().filter(['pct','pi_logpi']).reset_index()
shan.columns = ['sdl','pct','H']
shan['Hn'] = shan['H'] / np.max(shan['H'])

shan['HA'] = shan['H'] * (shan['pct'] * 0.01)
shan['HAn'] = shan['HA'] / np.max(shan['HA'])
shan.sort_values(by=['H'], ascending=False)
shan = shan.filter(['sdl','H','Hn','HA','HAn'])

# plt H
H = shan[['sdl','H']].sort_values(['H'])
plt.rcParams['font.size'] = '7'
plt.figure(figsize=(13/2.54 , 9/2.54),dpi=200)
plt.barh(H['sdl'], H['H'], align='center', color='grey')
plt.xlabel("H")
plt.title("Shannon's diversity index",ha='right')
plt.tight_layout(pad=1)
plt.savefig('shannons_x1_v02.png')
plt.show()
plt.close()

# plt HA
H = shan[['sdl','HA']].sort_values(['HA'])
plt.rcParams['font.size'] = '7'
plt.figure(figsize=(13/2.54 , 9/2.54),dpi=200)
plt.barh(H['sdl'], H['HA'], align='center', color='grey')
plt.xlabel("H")
plt.title("Shannon's diversity index area weighted",ha='right')
plt.tight_layout(pad=1)
plt.savefig('shannons_x1a_v02.png')
plt.show()
plt.close()

#%% Shannons (H) H = -∑[(pi) × ln(pi)], pi = n/N
#   BY ECOTYPE AND FILTERED BY GWD
N = x1f.groupby(['sdl']).sum().reset_index()
N = N.filter(['sdl','m2']).set_index('sdl')

n = x1f.set_index('sdl').join(N, on='sdl', rsuffix='_tot')
n['pi'] = n['m2'] / n['m2_tot']
n['logpi'] = np.log(n['pi']) * -1
n['pi_logpi'] = n['pi'] * n['logpi']

shanf = n.groupby(['sdl']).sum().filter(['pct','pi_logpi']).reset_index()
shanf.columns = ['sdl','pct','Hf']
shanf['Hfn'] = shanf['Hf'] / np.max(shanf['Hf'])

shanf['HAf'] = shanf['Hf'] * (shanf['pct'] * 0.01)
shanf['HAfn'] = shanf['HAf'] / np.max(shanf['HAf'])
shanf.sort_values(by=['Hf'], ascending=False)
shanf = shanf.filter(['sdl','Hf','Hfn','HAf','HAfn'])

# plt Hf
H = shanf[['sdl','Hf']].sort_values(['Hf'])
```

```
plt.rcParams['font.size'] = '7'
plt.figure(figsize=(13/2.54 , 9/2.54),dpi=200)
plt.barh(H['sdl'], H['Hf'], align='center', color='grey')
plt.xlabel("H")
plt.title("Shannon's diversity index excl. low, unknown GDE potential",ha='right')
plt.tight_layout(pad=1)
plt.savefig('shannons_x1f_v02.png')
plt.show()
plt.close()

# plt HAf
H = shanf[['sdl','HAf']].sort_values(['HAf'])
plt.rcParams['font.size'] = '7'
plt.figure(figsize=(13/2.54 , 9/2.54),dpi=200)
plt.barh(H['sdl'], H['HAf'], align='center', color='grey')
plt.xlabel("H")
plt.title("Shannon's diversity index area weighted\nexcl. low, unknown GDE potential",ha='right')
plt.tight_layout(pad=1)
plt.savefig('shannons_x1af_v02.png')
plt.show()
plt.close()

#%% join indices into single table
dfm = sim.merge(shan,on='sdl').merge(simf,on='sdl').merge(shanf,on='sdl')
dfm.to_csv('gde_sdl_diversity_indices_v02.csv')
```

## 2.6 Groundwater salinity mapping

Table 8 shows the script to analyse the groundwater salinity values obtained from the National Groundwater Information System v1.7.0. Groundwater salinity values are allocated to specific SDLs and upper and lower 2nd percentile values are filtered out to remove extreme values (outliers). Subsequently, a series of quantiles are calculated for each bore with salinity measurements, which are then further utilised for spatial interpolation.

**Table 8 Python script for processing groundwater salinity data and obtained 95th percentile used for salinity spatial interpolation**

```
### script to prepare data for groundwater salinity interpolation using rram classes
### for use in groundwater footprint aquifer prioritisation study

# import required packages and modules
import geopandas
import pandas as pd
from shutil import copyfile
from shapely.geometry import Point
import numpy as np
import matplotlib.pyplot as plt

#%% bore coords extracted from ngis v1.7.0 geodatabases for nsw, qld, vic
### http://www.bom.gov.au/water/groundwater/ngis/
### CLIP TO SDLs EXTRACT DEM & REGOLITH VALUES TO PTS IN ARCGIS
### geoscience aust 1s dem product https://ecat.ga.gov.au/geonetwork/srv/eng/catalog.search#/metadata/72759
### csiro soil and landscape grid https://www.clw.csiro.au/aclep/soilandlandscapegrid/ProductDetails-SoilAttributes.html

#%% read in processed bore xys
xy = geopandas.read_file('ngis170_mdb_alluv_dem_reg95_sdl.shp')
xy = xy.filter(['hydroid', 'lat', 'long', 'drill_dept', 'dem', 'reg95','SDL_NAME','geometry'])
# filter where drilled depth > regolith thickness
xy = xy[xy.drill_dept <= xy.reg95]
xy = xy.set_index(['hydroid'])

#%% seperate sdls, reassign overlapping shallow/deep sdls on drilled depth
### based on formation thicknesses in sdl report cards
## vic
shep = x.loc[x['SDL_NAME'] == 'Goulburn-Murray: Shepparton Irrigation Region (GS8a)']
shep = shep[shep.drill_dept <= 25]
goul = x.loc[x['SDL_NAME'] == 'Goulburn-Murray: Sedimentary Plain (GS8c)']
goul = goul[goul.drill_dept > 25]
goul = goul.append(shep) # n=32442 no dup
## nsw
murrum_l = x.loc[(x['SDL_NAME'] == 'Lower Murrumbidgee Shallow Alluvium (GS28a)')]
murrum_ls = murrum_l[murrum_l.drill_dept <= 40]
murrum_m =  x.loc[(x['SDL_NAME'] =='Midâ□□Murrumbidgee Alluvium (GS31)')]
murrum_m['SDL_NAME'] = "Mid-Murrumbidgee Alluvium (GS31)"
murrum_ld = murrum_l[murrum_l.drill_dept > 40]
murrum_ld['SDL_NAME'] = 'Lower Murrumbidgee Deep Alluvium (GS28b)'
murrum = murrum_ls.append([murrum_m,murrum_ld])
murray_u = x.loc[(x['SDL_NAME'] == 'Upper Murray Alluvium (GS46)')]
murray_l = x.loc[(x['SDL_NAME'] == 'Lower Murray Shallow Alluvium (GS27a)')]
murray_ls = murray_l[murray_l.drill_dept <= 20]
murray_ld = murray_l[murray_l.drill_dept > 20]
murray_ld['SDL_NAME'] = 'Lower Murray Deep Alluvium (GS27b)'
murray = murray_u.append([murray_ls,murray_ld])
gwy = x.loc[(x['SDL_NAME'] == 'Upper Gwydir Alluvium (GS43)')| (x['SDL_NAME'] == 'Lower Gwydir Alluvium (GS24)')]
nam = x.loc[(x['SDL_NAME'] == 'Upper Namoi Alluvium (GS47)')| (x['SDL_NAME'] == 'Upper Namoi Tributary Alluvium (GS48)')|
(x['SDL_NAME'] == 'Lower Namoi Alluvium (GS29)')]
```

```
macq = x.loc[(x['SDL_NAME'] == 'Upper Macquarie Alluvium (GS45)')| (x['SDL_NAME'] == 'Lower Macquarie Alluvium (GS26)')]
lach = x.loc[(x['SDL_NAME'] == 'Lower Lachlan Alluvium (GS25)')| (x['SDL_NAME'] == 'Upper Lachlan Alluvium (GS44)')]
bord = x.loc[(x['SDL_NAME'] == 'NSW Border Rivers Alluvium (GS32)')| (x['SDL_NAME'] == 'NSW Border Rivers Tributary Alluvium
(GS33)')| (x['SDL_NAME'] == 'Queensland Border Rivers Alluvium (GS54)')]
## qld
con = x.loc[(x['SDL_NAME'] == 'Upper Condamine Alluvium (Tributaries) (GS64b)')| (x['SDL_NAME'] == 'Upper Condamine Alluvium
(Central Condamine Alluvium) (GS64a)')]
# append all check dup
xy = murrum.append([murray,gwy,nam,macq,lach,bord,goul,con]).set_index('hydroid')
#dup=xy.groupby(['hydroid']).size()
## cleanup var list
del [goul,con,shep,murray,murray_l,murray_ld,murray_ls,murray_u,
     murrum,murrum_l,murrum_ld,murrum_ls,murrum_m,gwy,nam,macq,lach,bord]

#%% read salinity tables
s1 = pd.read_csv('salinity_NSW.csv').set_index('hydroid')
s2 = pd.read_csv('salinity_QLD.csv').set_index('hydroid')
s3 = pd.read_csv('salinity_VIC.csv').set_index('hydroid')
sal = s1.append([s2,s3])
del s1,s2,s3
## datetime field, keep ec readings, filter and rename cols
sal['datetime'] = pd.to_datetime(sal['bore_date'])
sal = sal[sal.uom == 'uS/cm']
sal = sal[sal.quality_flag != '0']
sal['year'] = pd.DatetimeIndex(sal['datetime']).year
sal['month'] = pd.DatetimeIndex(sal['datetime']).month
sal['year_month'] = pd.to_datetime(sal['datetime']).dt.to_period('M')
print(list(sal.columns))
sal = sal.filter(['result','datetime','year','month','year_month'])
sal.columns = ['uscm','datetime','year','month','year_month']

#%% join salinity obs to xys
sal_xy = sal.join(xy).dropna()
## examine histogram, check min/max vals
sal_xy['uscm'].hist(bins=50)
print(sal_xy['uscm'].min(), sal_xy['uscm'].max())
## trim upper and lower 2%ile to remove extreme outliers
sal_xy = sal_xy.query('uscm < uscm.quantile(.98) & uscm > uscm.quantile(.02)')
## examine histogram and min/max vals
sal_xy['uscm'].hist(bins=50)
print(sal_xy['uscm'].min(), sal_xy['uscm'].max())
sal_xy = sal_xy.reset_index()

#%% sdl boxplots
sal_xy.boxplot(column=['uscm'], by='CODE', figsize=(17/2.54 ,13/2.54), rot=45, grid=False, fontsize=7)
plt.savefig('sdl_salinity_boxplot_v04.png')

#%% calc bore salinity quantiles
uscm = sal_xy.filter(['hydroid','uscm'])
uscm.columns = ['hydroid','min']
uscm_p = uscm.groupby('hydroid').min()
uscm_p['p05'] = uscm.groupby('hydroid').quantile(0.05)
uscm_p['p10'] = uscm.groupby('hydroid').quantile(0.10)
uscm_p['p25'] = uscm.groupby('hydroid').quantile(0.25)
uscm_p['p50'] = uscm.groupby('hydroid').quantile(0.5)
uscm_p['p75'] = uscm.groupby('hydroid').quantile(0.75)
uscm_p['p90'] = uscm.groupby('hydroid').quantile(0.90)
uscm_p['p95'] = uscm.groupby('hydroid').quantile(0.95)
uscm_p['log_p95'] = np.log(uscm_p['p95'])
uscm_p['n'] = uscm.groupby('hydroid').size()
uscm_p = uscm_p[uscm_p.n > 1]
uscm_xy = uscm_p.join(xy).reset_index()
print(uscm_xy.groupby(['SDL_NAME']).size())

#%% write bore salinity quantiles to csv and shp
## pts shp file used for salinity interpolation based on p95 vals
uscm_xy.to_csv('ngis170_salinity_uscm_mdb_alluv_sdl_regolith_v04.csv')
uscm_xy['geometry'] = uscm_xy.apply(lambda x: Point((float(x.long), float(x.lat))), axis=1)
uscm_xy = geopandas.GeoDataFrame(uscm_xy, geometry='geometry')
uscm_xy.to_file('ngis170_salinity_uscm_mdb_alluv_sdl_regolith_v04.shp')
copyfile('gda94.prj','ngis170_salinity_uscm_mdb_alluv_sdl_regolith_v04.prj')

#%% sdl salinity pctiles
uscm = sal_xy.filter(['SDL_NAME','CODE','uscm'])
sdl_p = uscm.groupby(['SDL_NAME','CODE']).min()
sdl_p.columns = ['min']
sdl_p['max'] = uscm.groupby(['SDL_NAME','CODE']).max()
sdl_p['mean'] = uscm.groupby(['SDL_NAME','CODE']).mean()
sdl_p['p50'] = uscm.groupby(['SDL_NAME','CODE']).quantile(0.5)
sdl_p['skew'] = uscm.groupby(['SDL_NAME','CODE']).skew()
sdl_p['n'] = uscm.groupby(['SDL_NAME','CODE']).size()
sdl_p['p05'] = uscm.groupby(['SDL_NAME','CODE']).quantile(0.05)
sdl_p['p10'] = uscm.groupby(['SDL_NAME','CODE']).quantile(0.10)
sdl_p['p25'] = uscm.groupby(['SDL_NAME','CODE']).quantile(0.25)
sdl_p['p75'] = uscm.groupby(['SDL_NAME','CODE']).quantile(0.75)
sdl_p['p90'] = uscm.groupby(['SDL_NAME','CODE']).quantile(0.90)
sdl_p['p95'] = uscm.groupby(['SDL_NAME','CODE']).quantile(0.95)
sdl_p = sdl_p.reset_index()
sdl_p.to_csv('ngis170_salinity_uscm_mdb_alluv_sdl_pctiles_v04.csv')

#%% plt sdl pctiles
p = sdl_p[['SDL_NAME','CODE','p05','p50','p95']].sort_values(['p50'])
plt.rcParams['font.size'] = '7'
plt.figure(figsize=(17/2.54 , 17/2.54),dpi=300)
width = 0.35
labels=p['SDL_NAME']
plt.bar(labels, p['p95'], width, label='p95')
plt.bar(labels, p['p50'], width, label='p50')
plt.bar(labels, p['p05'], width, label='p05')
plt.ylabel('uScm')
plt.xticks(rotation=90)

# indicate rram salinity classes (tds ~ ec/0.64)
plt.axhline(y=2344, lw=0.5, c="blue")
plt.text(0.02,1344, "Fresh <1500 mgL TDS", c="blue")
plt.axhline(y=4688, lw=0.5, c="yellow")
plt.text(0.02,3688, "Brackish 1500-3000 mgL TDS", c="yellow")
```

```
plt.axhline(y=21875, lw=0.5, c="orange")
plt.text(0.02,20875, "Saline 3000-14000 mgL TDS", c="orange")
plt.text(0.02,22875, "Highly saline >14000 mgL TDS", c="red")
plt.legend()
plt.tight_layout()
plt.savefig('ngis170_salinity_uscm_mdb_alluv_sdl_pctiles_v04.png')
```

# References


Clark, S. R. (2022). Unravelling groundwater time series patterns: Visual analytics-aided deep learning in the Namoi region of Australia. *Environmental Modelling and Software*, *149*(September 2021), 105295. https://doi.org/10.1016/j.envsoft.2022.105295

Fu, G., Rojas, R., & Gonzalez, D. (2022). Trends in Groundwater Levels in Alluvial Aquifers of the Murray–Darling Basin and Their Attributions. *Water*, *14*(11), 1808. https://doi.org/10.3390/w14111808

Gorelick, R. (2006). Combining richness and abundance into a single diversity index using matrix analogues of Shannon's and Simpson's indices. *Ecography*, *29*, 525–530.

Spellerberg, I. F., & Fedor, P. J. (2003). A tribute to Claude Shannon (1916-2001) and a plea for more rigorous use of species richness, species diversity and the "Shannon-Wiener" Index. *Global Ecology & Biogeography*, *12*, 177–179. http://www.blackwellpublishing.com/journals/geb

Wilford, J., Ross, S., Thomas, M., & Grundy, M. (2018). *Soil and Landscape Grid National Soil Attribute Maps - Depth of Regolith (3" resolution) - Release 2*. V6. CSIRO. Data Collection. https://doi.org/10.4225/08/55C9472F05295

**As Australia's national science agency and innovation catalyst, CSIRO is solving the greatest challenges through innovative science and technology.**

CSIRO. Unlocking a better future for everyone.


**Contact us**

1300 363 400
+61 3 9545 2176
csiroenquiries@csiro.au
www.csiro.au

**For further information**

Land and Water
Dr Rodrigo Rojas
+61 **07 3833 5600**
Rodrigo.rojas@csiro.au
**csiro.au/en/about/people/business-units/land-and-water**