**COMPEAT.R EXPLANATION**

Compiled by Philip Lamb and Kate Collingridge.

Line numbers may not be entirely accurate where things have been added, but should be roughly right.

**INSTALL AND LOAD R PACKAGES**

**Line 1 – 9:** Reading in external packages

**Line 11 – 13:** Defining input and output paths (as strings)

**Line 16:** Defining assessment period using string: either 2006 – 2014 or 2015-2020

**Line 20:** Creating folders called “input” and “output” in the working directory

**DOWNLOAD AND UNPACK FILES NEEDED FOR THE ASSEMENT**

**Line 24-32:** A function to download and unzip files from specified URL

**Line 34-48:** Specifying URL depending on selected assessment period

**Line 50:** Downloading and unzipping a .zip, a .gz, and 3 .csv files from specified URL – storing them in the “input” folder **(line 20).** Stored as a named list “files” in the R environment

**Line 52-56:** Creating named strings with file paths pointed to the downloaded files in the “input” folder **(line 50)**

**ASSESSMENT UNITS AND GRID UNITS**

**Line 61- 62:** Defining units (to dataframe “units”) from the “AssessmentUnits.csv” file and retrieving coordinate reference system (EPSG:4326)

**Line 65:** Dropping dimensions and converting units dataframe to a data table

**Line 68- 69:** Reordering datatable by “ID”. Renaming “ID” as “Code”, “LongName” as “Description”, and “geometry” as “GEOM”. Saves in a “sf” (sf\_object is a table like object for storing spatial information) called “units”.

**Line 72:** Creating an ID column called “UnitID” in sf\_object “units” (line 69 – 69) with a unique value for each row.

**Line 75:** Double checking that geometries in sf\_object “units” are valid. Just displays true or false (manual check), doesn’t remove or change anything if units are invalid.

**Line 86:** Change sf\_object “units” projection to ETRS\_1989\_LAEA

**Line 88:** Calculates an area in m2 for each row in sf\_object “units”, saves values in a new column

**Line 92:** Double checking that geometries in sf\_object “units” are valid. Just displays true or false (manual check), doesn’t remove or change anything if units are invalid. One of the rows is invalid.

**Line 95:** Fixes invalid row using st\_buffer function. “Buffer or nothing” trick (no idea what this is or how it works – but it does make everything in “units” valid).

**Line 100:** Reorders sf\_object “unit” by “UnitID” (row 72) and defines this column as a “key” column (a “key” column is a super-charged row name – and is useful for identifying rows meeting certain conditions easily). In this instance I don’t think it changes much – but might be useful further down the line?

**Line 103-123:** Creating a function to create “grid units”. First converts sf\_object “units” to projection to ETRS\_1989\_LAEA (should already by that projection). Then calculates a bounding (max, min) for feature set. Extracts min and max for x and y (lat and long?). Then calculates length of x and y (e.g. x\_max – x\_min). Uses these values to create a square grid covering the geometry of sf\_object. Finally calculates distance between areas. Details of the grid and distances are saved.

**Line 128-130:** Makes grids for grid size 10000, 30000, and 60000 and sf\_object “units” – saved to gridunits10, gridunits30, and gridunits60 respectively.

**Line 132:** reads datatable “unitGridSize.csv” and sets column “UnitID” as a key column.

**Line 134 - 138:** Merges unitGridSize.csv and “units” sf\_object using correct gridsize and unitID. Saves as sf\_object.

**READ STATION SAMPLES**

**Line 148:** Reads StationSamples.txt.gz file into R environment saved as stationSamples.

**Line 161:** Turns stationSamples (line 148) into a sf\_object using EPSG code: 4326 projection

**Line 164:** Transfroms sf\_object “stationSamples” to ETRS\_1989\_LAEA projection.

**Line 174:** Adds gridunits (grid information) to sf\_object “stationSamples”

**Line 177:** Removes spatial column from sf\_object “stationSamples”

**READ INDICATOR CONFIGURATION FILES**

**Line 180:** Imports “Indicators.csv” and sets “Indicator ID” as key column

**Line 181:** Imports “IndicatorsUNits.csv” and sets “Indicator ID” and “UnitID” as key column

**Line 183 & 184:** Creates two blank list objects (probably going to be used in a loop later): wk1list and wk2 list

**LOOP INDICATORS**

**Phil comment:** In this section there is quite a lot of conditional formatting. Many of the variables (e.g GTC\_HM, GTC\_ML, STC\_HM, STC\_ML, GSC\_HM, HSC\_ML, SSC\_HM, SSC\_ML) are read in from “Indicators.csv” (rather than generated *de novo* here). Obviously the freshly computed variables are only as accurate as these are, but (as someone with little experience of OSPAR) I have no idea what many of these abbreviations are.

**Line 187 – 314:**

* For every row in indicators.csv
* Gather indicatorid, criterId, name, year.min, yar.max, month.min, month.xmax, depth.min, depth.max, metric, and response
* Load stationSamples (line 177) to object called “wk”
  + Create a column called period with correct year information
* Create a column ES grabbing correct nutrient value based on name column
  + Oxygen is divided by 0.7 (converting ml / l -> mg / l)
* Add grid size (matched based on “UnitID”
* Selects and renames certain columns of “wk” object
* Filters data to be within the min/max months and min/max depths from the indicators file. For everything except oxygen this means it is only using data at the surface (down to 10m)
* Generate salinity – normalisation curves using linear regression (between nutrient and salinity) for dissolved inorganic nitrogen and phosphorus
* Normalise salinity for situations in which significant (p > 0.05) relationships were obtained between salinity and nutrients saved to column “ESS”
  + Nothing is done with this normalised data at present, maybe to be used in the future?
* Calculates mean or min and stores result ink wk1 or wk2
* Where metric = mean, which is for everything except oxygen:
  + Calculate station mean. This is where any profiles would be averaged. In the original data profiles are binned to ICES standard depths I think (0,5,10,20,30,50,75,100,150,200,300 etc).
  + Calculate annual mean – this is calculated from the station means, and is a mean across each assessment area for each year (grouped by UnitID and Period)
* Where metric = minimum, which is for oxygen:
  + Calculate station minimum. So this will select the bin of the profile with the lowest value. Probably the deepest sample but not always?

**Line 317 - 318:** Binds the output of the loop into data.frame: wk1 and wk2

**Line 321:** Combines wk2 with indicators and indicator units; saved to wk3.

**Line 324:** Calculates standard error as new column “SE” in wk3.

**Line 327:** Calculates 95% CI as new column “CI” in wk 3.

**Line 330:** Calculates “BEST” as new column (wk3) – if “response” is 1, then BEST = ET / (1 + ACDEV / 100), otherwise BEST = ET / (1 - ACDEV / 100)

**Line 333:** Calculates “EQR” as new column (wk3) – calculation used is dependent on if BEST > ES

**Line 336:** Calculates “EQR\_GM” as new column (wk3) – calculation used is dependent on if “response” == 1

**Line 337:** Calculates “EQR\_HG” as new column (wk3): 0.475+ 0.5\*“EQR\_GM”

**Line 338:** Calculates “EQR\_PB” as new column (wk3): 2\*“EQR\_GM” – 0.95

**Line 339:** Calculates “EQR\_MP” as new column (wk3): 0.5\*“EQR\_GM” + 0.5\*“EQR\_PB”

**Line 342-346:** Calculates Ecological quality ratio scales as new column “EQRS” (wk3). Equation depends on how EQR relates to EQR\_PB / EQR\_MP / EQR\_GM / EQR\_HG.

**Line 348:** Creates column “EQRS\_Class” (wk3), classified as “high” if EQRS >= 0.8, Good if EQRS is between 0.6-0.8, Moderate if EQRS is between 0.4-0.6, “poor” when EQRS is between 0.2-0.4, and bad if EQRS is less than 0.2

**Line 354:** Calculates general temporal confidence – saved as a new column “GTC” (wk3). If N > GTC\_HM it receivers “100”, if N < GTC\_ML it receives 0, if N equal to or greater than GTC\_ML but smaller than GTC\_HM it receives “50”. GTC is the confidence in number of observations per year. This can be 100, 50 or 0 depending on how many data points related to two predefined thresholds.

**Line 357:** Calculates number of months potential – saved as a new column “NMP”. Accounts for months over multiple years (e.g. December 2019 – January 2020). This is the number of months in the assessment period for that indicator.

**Line 360:** Calculates specific temporal confidence - saved as a new column “STC” (wk3). This uses the number of months with data vs the potential number of months. So if all (or all but one for chlorophyll) the potential months are sampled it gets 100, if half (roughly) it gets 50, otherwise 0

**Line 364:** Calculate general spatial confidence as new column “GSC” (wk3), “NG” is sum of grid areas, divided by mean grid size squared. GSC is 100 if N/NG > GSC\_HM, 0 if N/NG < GSC\_ML, otherwise 50.

**Line 368-374:** Calculates specific spatial confidence as new column “SSC”. If grid areas / unit areas \* 100 > “SSC\_HM”: “SSC” is 100. If grid areas / unit areas \* 100 < SSC\_ML: “SSC” is 0. Otherwise, “SSC” is 50. This is looking at the percentage of the area (in terms of grid cells sampled) that is sampled and assigning a confidence based on that. Again the thresholds are taken from the indicators file, if 70/80% sampled then it gets 100, if 50/60 it gets 50, otherwise 0.

**Line 377:** Creates a new dataframe “wk4”, with columns calculated from wk3 (so all based on annual data for each assessment area:

* “Period”: minimum(period)\*10000 + maximum (period). This assigns it yearminyearmax e.g. 20162020.
* “ES”: mean(ES) mean of the annual values
* “SD”:standard deviation of ES SD of the annual values
* “N” = N
* “N\_OBS” = total N in wk3
* “GTC” = mean(GTC)
* “STC” = mean(STC)
* “SSC” = mean (SSC)

**Line 380:** Adds a column “NTC100” to wk4 with the number of values for which STC = 100

**Line 383:** Changes “STC” to 100 in situations where over half the number of years have an STC of 100.

**Line 388:** Combines wk4 with the indicators and indicator units table, binds to “wk5”.

**CONFIDENCE ASSESSMENTS (all done in wk5)**

**Line 393:** Creates a new column “TC” which is the average of general and specific temporal confidences

**Line 395:** Creates a new column “TC\_Class” which gives a grade based on values on TC: if TC > 75 classed as “high”, between 50 and 75 “moderate”, below 50 “low”

**Line 398:** Creates a new column “SC” which is the average of general and specific spaital confidences

**Line 400:** Creates a new column “SC\_Class” which gives a grade based on values on SC: if SC > 75 classed as “high”, between 50 and 75 “moderate”, below 50 “low”.

**Line 412:** Creates a new column “AC\_NPA” (accuracy confidence of non problem area). If Response is equal to 1, generates normal distribution around ES, using quantiles ET, and AC\_SE. Otherwise it generates distribution around ET, using quantiles ES, and AC\_SE

**Line 415:** Creates a new column “AC\_PA” (accuracy confidence of problem area) by subtracting AC\_NPA from 1.

**Line 421:** Creates a new column “AC” (accuracy confidence). Uses the larger value out of AC\_NPA and AC\_PA

**Line 424:** Creates a new column “ACC” (accuracy confidence class). If AC is greater than 0.9 it receives 100, if AC is bigger than 0.7 it receives 50, otherwise 0.

**Line 426:** Creates a new column “ACC\_Class” (accuracy confidence class class (?)). If ACC is greater than 75 it is “High”, if it is greater than or equal to “50” is “Moderate”, otherwise it is “Low”.

**Line 429:** Creates overall confidence in column “C” which is TC+SC+ACC/3

**Line 431:** Creates confidence class in column “C\_Class”: if C is greater than 75 it is “High”, if it is greater than or equal to 50 it is “Moderate”, otherwise it is “Low”.

**Line 436 – 457:** Calculates BEST, EQR, EQR\_GM, EQR\_HG,EQR\_PB, EQR\_MP, EQRS, EQRS in wk5. Exactly the same as in lines 330-348.

**CATEGORY**

**Line 462:** For rows with have EQRS scores, creates weighted mean of EQR, EQRS, and C using weights “IW” in columns “EQR”, “EQRS”, and “C” (replacing original values. Saves in dataframe “wk6”.

**Line 464:** Makes “wk6” wide (pivoting rows-> columns) using variables “N”, “EQR”, “EQRS”, and “C”. Saves in dataframe “wk7”.

**ASSESSMENT**

**Line 469:** For rows where “CategoryID” is 2 or 3: calculate minimum EQR and EQRS and set UnitID as key column. “N” is renamed to “NE”. Saved to dataframe wk81.

**Line 470:** Renames “N” to “NC”, calculates mean “C” to “C” sets “UnitID” as key column. Saves to wk82.

**Line 471:** Binds wk81 and wk82 to wk8.

**Line 473:** Binds wk7 and wk8 using “UnitID”, assigns 0 if no match is possible saved to wk9.

**Line 475-491:** Assigning classes to EQRS, EQRS\_11, EQRS\_12, EQRS\_2, and EQRS\_3. Receives “High” if value is greater than or equal to 0.8, “Good” if between 0.6-0.8, “Moderate” between 0.4-0.6, “Poor” between 0.2-0.4, and “Bad” below 0.2. Saved in wk9.

**Line 496-505:** Assigning classes to C, C\_11, C\_12, C\_2, C\_3. If C greater than or equal to 75 recieves “High”, if C is between 50 and 75 receives “Moderate”,, if below 50 receives “Low”. Saved in wk9.

**Line 508-510:** Saves wk3, wk5,wk9 to “annual\_indicator.csv”, “assessment\_indicator.csv”, and “assessment.csv” respectively.

**Line 513-519:** Setting value for plots (colours, limits, labels)

**Line 522:** Merging units and wk9 saved to wk.

**Line 524-718:** Creating figures (lots and lots of figures)