Table of Contents

..... 1
Flags script 2
Get SSS-satellite data within a given distance from each Argo float 3
Loop each year Colocate ref-2-Satellite 4
Save Colocation output 5
[1] Read SDN dataset 5
Do colocations through time 6
Colocate REF-2-satellite (Loop through each month) 7
Pre-lcoate vars_out (ref-2-SAT colocation yearly files) - snippet - 7
[1] Load DATA 8
[1.1] Load and tidy up the reference dataset 8
[1] load Satellite SSS: 10
[1] Baltic+ Nominal, [2] Baltic Nodal Sampling, [3] Global BEC, [4] CCI+SSS 10
[1.2.1] Baltic+ [BEC-SSS] Nominal v1.0 20110101 - 20131227 10
[1.2.2] Baltic+ Nodal Sampling [version v1.0: 20110101 - 20180102] 10
load Baltic+ grid -to convert all products to the same grid 11
[1.2.3] Baltic-Global product (v001) 11
[1.2.4] CCI+SSS data (v01.07) 12
[1.2.5] Load Model in Baltic 13
[1.3] Grid SDN data to the smos-grid 14
[2] Select Satellite-SSS data at Float location (within r distance in km) 14
Satellite data at float location 15
[2.3] Argo Vertical profile interpolation 15
[2.3.1] Interpolate Z-direction (narrow depth levels) 16
[3] dSSS = SDN - SATELLITE 16
Remove empty floats (*optimize space) 17
save fn_out (with Argo-2-satellite colocations) 17


```
function [vars_out] =
 Baltic_SDN_BEC_colocations_v2r0(iyear,imonth,iregion,idata_type)

%
%
% Syntax (function):
% [vars_out] =
 Baltic_SDN_BEC_colocations(iyear,imonth,iregion,idata_type)
%
% Description
% Make SDN-2-Satellite colocations whithin a searching
% distance and within
% a time window (i.e. ±7-days).
% [1] The output from this script is to feed into
 Baltic_argo_BEC_analyses.m
% [2] This script save colocations
 [Baltic_argo_BEC_colocations_VARS2SAVE.m]
%
% Use Seadatanet files prepared by BEC for validation SSS (Baltic+)
%
% Input
% iyear, imonth
```

```
% iregion: There are four Baltic+ study regions (see Baltic+ DUM, p.
 28):
% [1] Arkona Basin           [ArB] (55°31'30.22"N,  16°16'15.06"E)
% [2] Bothian Sea            [BOS] (61°55'50.19"N,  19°14'46.24"E)
% [3] Gulf of Finland        [GOF] (59°35'55.07"N,  23°07'27.96"E)
% [4] Northern Baltic Proper [NBP] (57°36'10.05"N,  19°48'25.78"E)
% [5] ALL-Baltic             [ALL]
%
% * for more informaiton about Baltic regions type "help
 Baltic_studyregion.m"
%
%
%
% data_type:   [1] Nominal, [3] Nodal Sampling,
%              [6] Global BEC (v001);
%              [7] CCI+SSS (v01.07);
%              [8] Model (NEMO) data in Baltic
%
%
% Output
% vars_out is structure and Matlab (.mat) and NetCDF (.nc) files
% with colocated profiles.
%
%
% current version: v2r0 (2020/03/18) -
%         [1] converted script into function;
%         [2] split up colocations and stats in
%             two different scripts. This script does colocation
 (only)
%             to input in stats.
%
% History
% v1r2 (2020/01/20) - original script to do colocations and stats
%
%
 ============================================================================
% Author: rcatany
%
%
 ============================================================================

% clc;
close all

% switch off warning (there is something about interpolation)
warning('off','all');
```

Flags script

```
run_test = 0;    % flag to run script (not to save figures);
save_output = 1; % flag to save output [1], or not [0];

if run_test == 1
```

```
    clc
    warning ('Baltic_seadatanet_BEC_colocation.m is in TEST MODE')

    disp(['RUNNING TEST MODE. Figures and files might NOT BE SAVED'])
    prompt_msm = 'Do you want to save script output? [1] yes, or [0]
 not ... ';
    save_output = input(prompt_msm);
end

% set the data_type
if idata_type == 1
    version_str = 'plusv3r0';
elseif idata_type == 3
    version_str = 'v2r0';
elseif idata_type == 4
    version_str = 'v1r0_NOM';
elseif idata_type == 5
    version_str = 'v1r0_NS';
elseif idata_type == 6
    version_str = 'v001';
elseif idata_type == 7
    version_str = 'CCI+SSSv1.7';
elseif idata_type == 8
    version_str = 'REANALYSIS_PHY_003_011';

end
```

```
Not enough input arguments.

Error in Baltic_SDN_BEC_colocations_v2r0 (line 74)
if idata_type == 1
```

Get SSS-satellite data within a given distance from each Argo float

```
r = 25; % Radius distance platform (in Km)
r_str = num2str(r);

% Areas at high-lats are about 25% bigger than at equator
lat_factor = 1.25;

% multiply lat_factor to get a more accurate sampling radius
r2 = r*lat_factor;

depth_ref = 10; % Reference depth (m), by gral. assumtion 10 m

ibasin = 9; % Basin number: 9 (7 Arctic, 9 Baltic)
[xmin,xmax,ymin,ymax, basin_str] = map_lim_raf(ibasin);

fg_save = 1; % flag save figures [1]; or not [0];
fg_format = 'png';
```

```
path_root = ('/Volumes/Rogue/Data/');
folder_data = ([path_root ...
    'SSS/Baltic/BEC/Validation/indata/SDN/']);


% ==========================
%
% iyear = 2011:2013;
% imonth = 1:12;


a = length(iyear);
b = length(imonth);

% filename Argo BEC structure: argo_20110101_20110110.nc
ndays = 9; % number of days contained in each BEC Argo file




% Output variables to save
ref_vars_out = {...
    'SALT_irange','TEMP_irange','PRES_irange',...
    'lon_irange','lat_irange','time_irange'...
    'region','basin_str'};

sat_vars_out = {...
    'sss_irange',...
    'sss_error_irange',...
    'lon_sss_irange','lat_sss_irange','version_str'};

colocated_vars_out = {...
    'sss_irange_mn',...
    'sss_irange_md',...
    'sss_error_irange_mn'};

save_vars_out = [ref_vars_out sat_vars_out colocated_vars_out];
```

Loop each year Colocate ref-2-Satellite

pre-locate vars

```
% pre-lcoate Stats dSSS = SATELLITE minus ARGO
nTOT    = 500;  % Total number of sat-to-argo colocations
ndepth  = 1;   % number of depth levels

grid_size = 1/4; % SMOS-grid size 1/4 (~0.25 km)

n = 9; % add extra elements to the matrix

% maximum number grid-points in irange + n extra elements
nele = (((r2/100)*4)/grid_size)+1+n;
```

```
ncount = 0; % set counter to zero

% Colocatiion fn_out for each Regional study in the Baltic
if strcmpi(iregion,'ARB') || iregion == 1% Arkona Basin
    region = 'ARB';

elseif strcmpi(iregion,'BOS')|| iregion == 2 % Bothian Sea
    region = 'BOS';

elseif strcmpi(iregion,'GOF')|| iregion == 3 % Gulf of Finland
    region = 'GOF';

elseif strcmpi(iregion,'NBP')|| iregion == 4 % Gulf of Finland
    region = 'NBP';

elseif strcmpi(iregion,'ALL') || iregion == 5 % all-Baltic region
 stats
    region = 'ALL';

end
```

Save Colocation output

```
folder_out = [path_root...
    'SSS/' basin_str...
    '/BEC/Validation/indata/SDN/Colocations/monthly/' version_str '/'
 region '/'];
foldercheck_raf(folder_out);


% Make a log_file to record status of each ARGO-BEC file [2020/01/21]
folder_log = '/Volumes/Rogue/Data/SSS/Baltic/BEC/Validation/indata/';
fn_log = [folder_log 'SDN_MISSING_20200121.txt'];

folder_figs = '/Volumes/Rogue/scratch/Validation/';


folder_figs = [folder_figs basin_str '/SDN/' region '/'];

if fg_save == 1
    foldercheck_raf(folder_figs); %! make folder_figs
end
```

[1] Read SDN dataset

```
% Future releases might chnage filenanme of SDB dataset
fn_in = [folder_data...
    'data_from_SDN_2015-09_TS_BalticSea_QC_done_v2_filtered.nc'];


% function to read SDN dataset
[TT] = rd_SDN (fn_in);
```

```
lon = TT.lon;
lat = TT.lat;

time_number = TT.time_number;
time_str1 = datestr(time_number(1),'yyyymmdd');
time_str2 = datestr(time_number(end),'yyyymmdd');

SALT = TT.SSS_SDN;
TEMP = TT.SST_SDN;
PRES = TT.depth_SDN;

platform_type = TT.metavar3; % [B]: Bottle; or [C]: CTD
platform_ID = TT.metavar5;
vars_measured = TT.metavar12;

z_grid = 0.5; % vertical interp at 0.5 m

% function to make vertical intp (works with any prof., not only Argo)
[t_intp,s_intp,p_intp] = zinterp1_ARGO(TEMP,SALT,PRES,z_grid);

SALT = s_intp;
TEMP = t_intp;
PRES = p_intp;

ind = PRES > 100;

SALT(ind) = NaN;
TEMP(ind) = NaN;
PRES(ind) = NaN;

clear *intp


% ========================
```

Do colocations through time

```
for yy = 1:length(iyear)
    iYEAR = iyear(yy);

    disp(['Processing SDN-2-Satellite colocation '...
        basin_str ' ' num2str(iYEAR)]);

    for mm = 1:length(imonth)

        iMONTH = imonth(mm);

        % ============================
        % fn_out (.mat)
        % %         fn_out = [folder_out 'SDN_'...
        % %             sprintf('%02.0f',depth_ref)
 'm_COLOCATIONS_'...
        % %             sprintf('%02.0f',iYEAR)...
```

```
        % %             '.mat'];

        fn_out = [folder_out basin_str '_' region '_SDN_R'
r_str '_'...
            sprintf('%02.0f',depth_ref) 'm_COLOCATIONS_'...
            sprintf('%02.0f',iYEAR) sprintf('%02.0f',iMONTH) '_'...
            version_str '.mat'];


        % ================================
```

Colocate REF-2-satellite (Loop through each month)

```
        fn_out_exist = exist(fn_out,'file');

        if fn_out_exist ~= 2 || run_test == 1
```

Pre-lcoate vars_out (ref-2-SAT colocation year-ly files) - snippet -

```
            run Baltic_SDN_BEC_colocations_VARS2SAVE.m % -  snippet -

            % ==========================

            disp(['year: ' num2str(iYEAR)...
                ' month: ' sprintf('%02.0f',iMONTH)])

            % folder name with BEC-ARGO data
            folder_in = ([folder_data...
                num2str(iYEAR) '/' sprintf('%02.0f',iMONTH) '/']);

            % number of days in month
            idays = calendar(iYEAR,iMONTH);
            idays (idays == 0) = [];
            idays = sort(idays);

            for dd = 1: length(idays)

                iDAY = idays(dd);
                itime_start = datenum(iYEAR,iMONTH,iDAY);
                itime_end = itime_start + ndays;

                iTIME = [itime_start itime_end]; % keep time in vector

                % >>>>>>>>>>>>>>>>>>>>
                % load SDN-matlab file [fn]
                %                 fn = (['argo_' ...
                %                     datestr(itime_start,'yyyymmdd')
'_'...
                %                     datestr(itime_end,'yyyymmdd')]);
```

```
%
%                 fn_in = ([folder_in fn '.mat']);
% >>>>>>>>>>>>>>>>>>>>

% ===============
```

[1] Load DATA

[1.1] Load and tidy up the reference dataset

```
if ismember(itime_start,time_number) == 1

    ind = time_number == itime_start;

    lon_alfa = lon(ind);
    lat_alfa = lat(ind);

    platform_type_alfa = platform_type(ind);
    platform_ID_alfa = platform_ID(ind);

    time_number_alfa = time_number(ind);

    SALT_alfa = SALT(:,ind);
    TEMP_alfa = TEMP(:,ind);
    PRES_alfa = PRES(:,ind);

    % station close to land are empty <??>
    ind2 = any(SALT_alfa);

    lon_alfa(~ind2) = [];
    lat_alfa(~ind2) = [];
    time_number_alfa(~ind2) = [];

    SALT_alfa(:,~ind2) = [];
    TEMP_alfa(:,~ind2) = [];
    PRES_alfa(:,~ind2) = [];

    clear ind*

    % Regional study in the Baltic
    if strcmpi(region,'ARB') % North Atlantic

        ind_reg = lon_alfa >=15 & lon_alfa <= 18  &
lat_alfa > 53 & lat_alfa <= 58;

    elseif strcmpi(region,'BOS')

        ind_reg = lon_alfa >= 18 & lon_alfa <= 23 &
lat_alfa > 60 & lat_alfa <= 64;

    elseif strcmpi(region,'GOF')
```

```
                        ind_reg = lon_alfa >= 23 & lon_alfa <= 26 &
lat_alfa > 58 & lat_alfa <= 62;

                    elseif strcmpi(region,'NBP')

                        ind_reg = lon_alfa >= 18 & lon_alfa <= 23 &
lat_alfa > 56 & lat_alfa <= 59;

                    elseif strcmpi(region,'ALL') % all-Arctic region
stats
                        ind_reg = ones(size(lon_alfa));

                    end

                    lon_alfa(~ind_reg) = [];
                    lat_alfa(~ind_reg) = [];
                    time_number_alfa(~ind_reg) = [];
                    SALT_alfa(:,~ind_reg) = [];
                    TEMP_alfa(:,~ind_reg) = [];
                    PRES_alfa(:,~ind_reg) = [];
                    clear ind_reg


                    if ~isempty(lon_alfa) && ~isempty(lat_alfa) &&
sum(any(SALT_alfa))~=0

                        % Cases with more than one float (profile) in
one file
                        nprof1 = length(lon_alfa(:,1));
                        nprof2 = length(lat_alfa(:,1));

                        % Argo lon-lat must be same size
                        if size(lon_alfa) == size(lat_alfa)
                            nprof = nprof1;
                        else
                            error([...
                                'Number of ref longitudes and
latitudes'...
                                ' must be equal'])
                        end

                        % Make PRES to be same size as SALT (and TEMP)
                        [a1,b1] = size(SALT_alfa);
                        [a2,b2] = size(PRES_alfa);

                        if b1 ~= b2
                            PRES_alfa = repmat(PRES_alfa,1,b1);
                        end
```

[1] load Satellite SSS:

[1] Baltic+ Nominal, [2] Baltic Nodal Sampling, [3] Global BEC, [4] CCI+SSS

[1.2.1] Baltic+ [BEC-SSS] Nominal v1.0 20110101 - 20131227

```
                        if idata_type == 4 && itime_start >=
datenum(2011,02,01) && itime_start <= datenum(2013,12,27)
                            data_type = idata_type; % Baltic+ (SSS-BEC
NM v1.0)

                            [TT] =
rd_smos_L4_BEC_v1r3(itime_start,ibasin,data_type);

                            lon_sss1 = TT.lon;
                            lat_sss1 = TT.lat;

                            sss = TT.sss;
                            sss_error = TT.sss_error;

                            lon_sss = lon_sss1;
                            lat_sss = lat_sss1;

                        elseif idata_type == 4 && (itime_start <
datenum(2011,02,01) || itime_start > datenum(2019,08,31))
                            sss = nan;
                            lon_sss = NaN;
                            lat_sss = NaN;
                        end
```

[1.2.2] Baltic+ Nodal Sampling [version v1.0: 20110101 - 20180102]

```
                        if idata_type == 5 && itime_start >=
datenum(2011,02,01) && itime_start <= datenum(2013,12,27)
                            data_type = idata_type; % Baltic+ (SSS-BEC
NS v1.0)

                            [TT] =
rd_smos_L4_BEC_v1r3(itime_start,ibasin,data_type);

                            lon_sss1 = TT.lon;
                            lat_sss1 = TT.lat;

                            sss = TT.sss;
```

```
                sss_error = TT.sss_error;

                lon_sss = lon_sss1;
                lat_sss = lat_sss1;

            elseif idata_type == 5 && (itime_start <
 datenum(2011,02,01) || itime_start > datenum(2019,08,31))
                sss = nan;
                lon_sss = NaN;
                lat_sss = NaN;
            end
```

load Baltic+ grid -to convert all products to the same grid

```
            if idata_type == 6 || idata_type == 7 ||
 idata_type == 8

                data_type = 5;
                % store Baltic+ grid (lon/lat)
                folder_grid = ([path_root ...
                    'SSS/' basin_str '/BEC/
Baltic_grid/']);

                fn_grid =
 [folder_grid 'Baltic_plusv1.0_grid.mat'];

                if exist(fn_grid,'file') ~= 2
                    [TT] =
 rd_smos_L4_BEC_v1r3(itime_start,ibasin,data_type);

                    lon_sss = TT.lon;
                    lat_sss = TT.lat;

                    % save Arctic grid
                    folder_grid = (['/Volumes/Rogue/Data/
SSS/Baltic/BEC/' basin_str '_grid/']);
                    foldercheck_raf(folder_grid);

                    save (fn_grid,'lon_sss','lat_sss');

                else
                    load (fn_grid)
                end
            end
```

[1.2.3] Baltic-Global product (v001)

```
            if idata_type == 6 && iYEAR < 2010
                data_type = 6;
                [TT] =
 rd_smos_L4_BEC_v1r3(itime_start,ibasin,data_type);
```

```
                            sss_beta = TT.sss;

                            lon_beta = TT.lon;
                            lat_beta = TT.lat;

                            % homogenize grid (global product to
Baltic+)
                            [sss_beta2] =
griddata(lon_beta,lat_beta,sss_beta,lon_sss,lat_sss);

                            ind = lon_sss >= xmin & lon_sss <= xmax
|...
                                lat_sss >= ymin & lat_sss <= ymax ;

                            sss_beta2 (ind == 0) = NaN;

                            sss = sss_beta2;
                            sss_error = nan(size(sss));

                            clear *_beta* ind TT; % clear work space

                        elseif idata_type == 6 && iYEAR >=2010
                            sss = nan;
                            lon_sss = NaN;
                            lat_sss = NaN;


                        end
```

[1.2.4] CCI+SSS data (v01.07)

CCI+SSS data: 20100106 - 20181101

```
                        if idata_type == 7 && (itime_start >=
datenum(2011,01,01) && itime_start <= datenum(2018,11,01))

                            plot_cci_ex = 0;
                            cci_product = 1; % [1] '7-days', or [2]
'30-days' product
                            [TT] =
rd_sss_cci(itime_start,ibasin,cci_product,plot_cci_ex);

                            sss_beta = TT.sss;

                            sss_error_beta = TT.sss_bias_std;

                            % get lat/lon from Baltic+
                            lon_sss1 = lon_sss;
                            lat_sss1 = lat_sss;

                            lon_beta = TT.lon;
                            lat_beta = TT.lat;
```

```
                            % Grid Baltic product to the same grid
(all in
                            % Baltic+ grid)
                            sss =
griddata(lon_beta,lat_beta,sss_beta,lon_sss1,lat_sss1);
                            sss_error =
griddata(lon_beta,lat_beta,sss_error_beta,lon_sss1,lat_sss1);

                            lon_sss = lon_sss1;
                            lat_sss = lat_sss1;

                            clear lon_sss1 lat_sss1 *_beta*

                        elseif idata_type == 7 && (itime_start <
datenum(2011,01,01) || itime_start > datenum(2018,11,01))
                            sss = nan;
                            lon_sss = NaN;
                            lat_sss = NaN;

                        end
```

[1.2.5] Load Model in Baltic

```
                        if idata_type == 8 && itime_start >=
datenum(2011,01,01) && itime_start <= datenum(2013,12,16)

                            grid2baltic = 0; % grid model output to
Baltic grid
                            [YY,MM,DD] = datevec(itime_start);

                            disp(['model at: ' datestr(itime_start)])

                            data_type = idata_type; % Baltic+ MODEL

                            [TT] = nemo_rd_BEC_Baltic_FUN
(YY,MM,DD,ibasin,grid2baltic);

                            sss_beta = TT.SSS_model;

                            % get lat/lon from Baltic+
                            lon_sss1 = lon_sss;
                            lat_sss1 = lat_sss;

                            lon_beta = TT.lon;
                            lat_beta = TT.lat;

                            % Grid Baltic product to the same grid
(all in
                            % Baltic+ grid)
                            sss =
griddata(lon_beta,lat_beta,sss_beta,lon_sss1,lat_sss1);
```

```
                            sss_error = nan(size(sss));

                            lon_sss = lon_sss1;
                            lat_sss = lat_sss1;

                            clear lon_sss1 lat_sss1 *_beta*

                        elseif idata_type == 8 && (itime_start <
datenum(2011,01,01) || itime_start > datenum(2013,12,27))
                            sss = nan;
                            lon_sss = NaN;
                            lat_sss = NaN;
                        end
                        % =================
```

[1.3] Grid SDN data to the smos-grid

```
                        [lon_grid,lat_grid] = ...

griddata_raf(lon_alfa,lat_alfa,lon_sss,lat_sss);

                        % plot example of gridded and non-gridded data
                        plot_example = 0;

                        if plot_example == 1
                            nn =1;

run Baltic_seadatanet_BEC_colocations_PLOT.m
                        end
```

[2] Select Satellite-SSS data at Float location (within r distance in km)

```
                        for nn = 1: nprof

                            % keep Argo lon/lat gridded to smos-grid
                            if nn == 1
                                lon_alfa = lon_grid;
                                lat_alfa = lat_grid;

clear lon_grid lat_grid lon_model lat_model
                            end

                            DIST =
Distance(lon_alfa(nn),lat_alfa(nn),lon_sss,lat_sss);
                            irange = find(DIST <= r2);

                            Ln = length(irange);

                            if Ln >= 1

                                ncount = ncount+1;
```

```
                                % to reset "repetition profile check"
                                rep_prof = -1;
```

Satellite data at float location

```
                                sss_irange (1:Ln,ncount) =
sss(irange);

                                sss_error_irange (1:Ln,ncount) =
sss_error(irange);

                                sss_irange_mn (ncount) =
nanmean(sss(irange));

                                sss_irange_md (ncount) =
nanmedian(sss(irange));

                                sss_error_irange_mn (ncount) =
nanmean(sss_error(irange));

                                sss_error_irange_md (ncount) =
nanmedian(sss_error(irange));

                                lon_sss_irange (1:Ln,ncount) =
lon_sss(irange);
                                lat_sss_irange (1:Ln,ncount) =
lat_sss(irange);
```

[2.3] Argo Vertical profile interpolation

```
                                [a,b] = size(SALT_alfa);

                                xROW = 1:a;

                                % nprof_irange (1,ncount) = nprof;

                                lon_irange (1,ncount) = lon_alfa(nn);
                                lat_irange (1,ncount) = lat_alfa(nn);

                                time_irange (ncount) =
time_number_alfa(nn);

                                platform_type_irange(1,ncount) =
char(platform_type_alfa(nn));
                                platform_ID_irange(1,ncount) =
char(platform_ID_alfa(nn));

                                PRES_irange (xROW,ncount) =
PRES_alfa(:,nn);
                                SALT_irange (xROW,ncount) =
SALT_alfa(:,nn);
```

```
                                TEMP_irange (xROW,ncount) =
TEMP_alfa(:,nn);

                                clear a b
```

[2.3.1] Interpolate Z-direction (narrow depth levels)

```
% Interpolation specs: Grid dimensions
pres1 = 0;              % min pres bin
pres2 = max(PRES_alfa); % max depth (m)
z_grid = 0.5;           % interpolant dista

TEMP1 = TEMP_alfa(:,nn);
SALT1 = SALT_alfa(:,nn) ;
PRES1 = PRES_alfa(:,nn) ;

% interpolation function
[TEMP_beta_intp,SALT_beta_intp,PRES_beta_in
    zinterp1_ARGO(TEMP1,SALT1,PRES1,z_grid)

[a,b] = size(TEMP_beta_intp);

TEMP_intp(1:a,ncount) = TEMP_beta_intp;
SALT_intp(1:a,ncount) = SALT_beta_intp;
PRES_intp(1:a,ncount) = PRES_beta_intp;

clear a b *_beta
```

[3] dSSS = SDN - SATELLITE

% Done by snippet "Baltic_SDN_colocations_stats.m"

```
                        end
                    end; clear *_alfa *_beta*

                    %                   elseif isempty(lon) ||
isempty(lat)
                    %                       msg_log = ([fn_in '
EMPTY_FILE']);
                    %
make_LOGFILE(fn_log,msg_log);

                end

            elseif exist(fn_in,'file') == 0
                % write a log_file, recording the Argo (.mat)
files
                msg_log = ([fn_in ' MISSING_FILE']);
                make_LOGFILE(fn_log,msg_log);
            end

        end
```

```
%         end; clear mm dd TEMP SALT TEMP1 SALT1 lon lat
```

Remove empty floats (*optimize space)

```
[a,b] = size(lon_irange);

% index empty floats (ie. Empty salt and temp)
ind1 = (all(isnan(SALT_irange),1));
ind2 = (all(isnan(TEMP_irange),1));

ind = ind1 & ind2;

if ~isempty (ind)

    % [BUG] Issue relating to the removing nan (columns/
rows) --
    % fixed (badly) with a dirty for-loop and if-
statements

    for nn = 1:length(save_vars_out)

        eval(['var_beta = ' save_vars_out{nn} ';']);

        % To remove platform profiles (NOT depth levels)
        [a2,b2] = size(var_beta);

        if b2 == b
            eval([save_vars_out{nn} '(:,ind) = [];']);

        elseif a2 == b
            eval([save_vars_out{nn} '(ind,:) = [];']);
        end

    end

end
clear *beta*
```

save fn_out (with Argo-2-satellite colocations)

```
            disp({'New file with SDN-2-satellite colocations ';
 fn_out})

            save(fn_out,save_vars_out{:})

        elseif fn_out_exist == 2
            load(fn_out);

        else
            clc
            warning (['Missing Colocations file: ' fn_out])
```

```
            % write a log_file, recording the Argo (.mat) files
            msg_log = ([fn_in ' MISSING_FILE']);
            make_LOGFILE(fn_log,msg_log);
        end

    end
end



% Setup function output
TT = struct;

for nn = 1:length(save_vars_out)
    this_param = save_vars_out{nn};

    eval(['TT.' this_param '= ' this_param ';']);

end


vars_out = TT;
clear TT



% END-OF-SCRIPT

Exception in thread "AWT-EventQueue-0": java.lang.ClassCastException:
 javax.swing.plaf.basic.BasicComboBoxUI cannot be cast to
 com.jidesoft.plaf.ExComboBoxUI
 at com.jidesoft.grid.JideTable.removeEditor(Unknown Source)
 at com.jidesoft.grid.JideTable.editingStopped(Unknown Source)
 at
 javax.swing.AbstractCellEditor.fireEditingStopped(AbstractCellEditor.java:141)
 at
 javax.swing.AbstractCellEditor.stopCellEditing(AbstractCellEditor.java:85)
 at com.jidesoft.grid.ExComboBoxCellEditor.stopCellEditing(Unknown
 Source)
 at com.jidesoft.grid.JideTable$s_.propertyChange(Unknown Source)
 at
 java.beans.PropertyChangeSupport.fire(PropertyChangeSupport.java:335)
 at
 java.beans.PropertyChangeSupport.firePropertyChange(PropertyChangeSupport.java:32
 at
 java.beans.PropertyChangeSupport.firePropertyChange(PropertyChangeSupport.java:26
 at
 java.awt.KeyboardFocusManager.firePropertyChange(KeyboardFocusManager.java:1493)
 at
 java.awt.KeyboardFocusManager.setGlobalPermanentFocusOwner(KeyboardFocusManager.j
 at
 java.awt.DefaultKeyboardFocusManager.dispatchEvent(DefaultKeyboardFocusManager.ja
 at java.awt.Component.dispatchEventImpl(Component.java:4760)
```

```
 at java.awt.Container.dispatchEventImpl(Container.java:2297)
 at java.awt.Component.dispatchEvent(Component.java:4711)
 at java.awt.EventQueue.dispatchEventImpl(EventQueue.java:760)
 at java.awt.EventQueue.access$500(EventQueue.java:97)
 at java.awt.EventQueue$3.run(EventQueue.java:709)
 at java.awt.EventQueue$3.run(EventQueue.java:703)
 at java.security.AccessController.doPrivileged(Native Method)
 at java.security.ProtectionDomain
$JavaSecurityAccessImpl.doIntersectionPrivilege(ProtectionDomain.java:74)
 at java.security.ProtectionDomain
$JavaSecurityAccessImpl.doIntersectionPrivilege(ProtectionDomain.java:84)
 at java.awt.EventQueue$4.run(EventQueue.java:733)
 at java.awt.EventQueue$4.run(EventQueue.java:731)
 at java.security.AccessController.doPrivileged(Native Method)
 at java.security.ProtectionDomain
$JavaSecurityAccessImpl.doIntersectionPrivilege(ProtectionDomain.java:74)
 at java.awt.EventQueue.dispatchEvent(EventQueue.java:730)
 at
 java.awt.EventDispatchThread.pumpOneEventForFilters(EventDispatchThread.java:205)
 at
 java.awt.EventDispatchThread.pumpEventsForFilter(EventDispatchThread.java:116)
 at
 java.awt.EventDispatchThread.pumpEventsForHierarchy(EventDispatchThread.java:105)
 at
 java.awt.EventDispatchThread.pumpEvents(EventDispatchThread.java:101)
 at
 java.awt.EventDispatchThread.pumpEvents(EventDispatchThread.java:93)
 at java.awt.EventDispatchThread.run(EventDispatchThread.java:82)
```

Published with MATLAB® R2019a