Post-processing RBR data with RSKtools

Table of Contents

Introduction 1
Open the data, print the channel names, and plot a few profiles 1
Add station data 3
Derive sea pressure from total pressure 3
Data post-processing 3
Correct for A2D zero-order hold 4
Low-pass filtering 4
Alignment of conductivity and temperature 5
Remove loops 6
Derived variables 7
Bin average all channels by sea pressure 8
Compare the raw and processed data 9
Visualize data with a 2D plot 9
Export logger data to CSV files 10
Display a summary of all the processing steps 10
Other Resources 10
About this document 11

RSKtools v3.6.0; RBR Ltd. Ottawa ON, Canada; support@rbr-global.com; 2024-01-10

Introduction

`RSKtools` is RBR's open source Matlab toolbox for reading, visualizing, and post-processing RBR logger data. It provides high-speed access to large RSK data files. Users may plot data as a time series or as depth profiles using tailored plotting utilities. Time-depth heat maps can be plotted easily to visualize transects or moored profiler data. A full suite of data post-processing functions, such as functions to match sensor time constants and bin average, are available to enhance data quality. RBR is continually expanding RSKtools, and we value feedback from users so that we can make it better.

Review RSKtools Getting Started for an introduction on how to load RBR data into Matlab from RSK files, make plots, and access the data.

RSKtools includes a series of functions to post-process RBR logger data. Below we show how to implement some common processing steps to obtain the highest quality data possible.

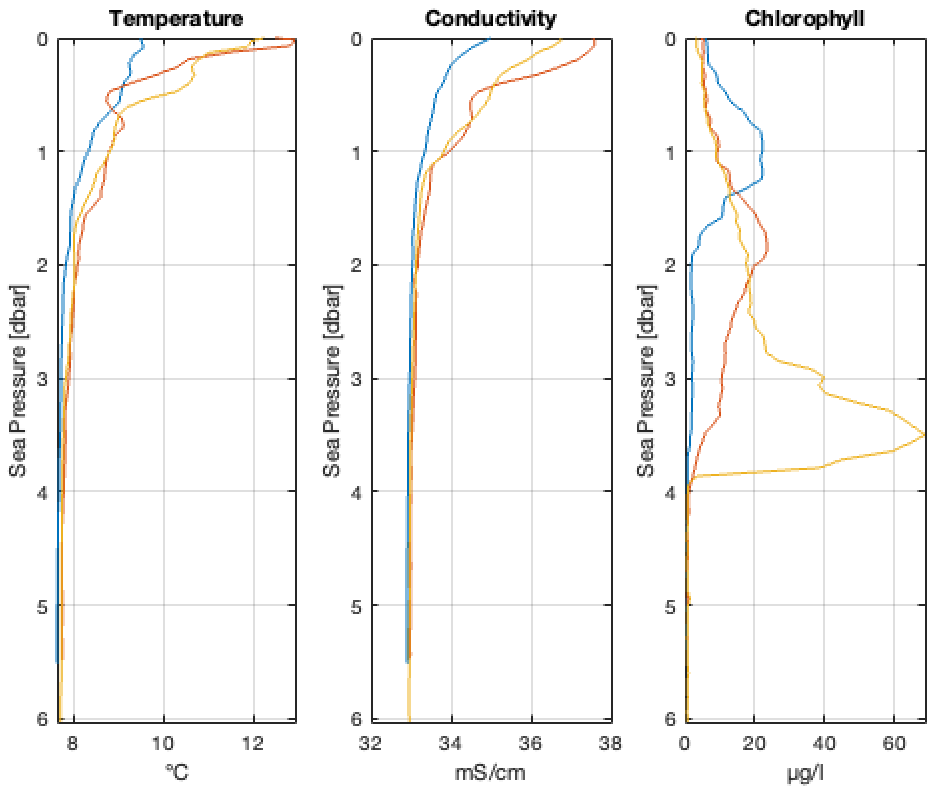
All post-processing functions are customizable with name-value pair input arguments. Documentation for each function can be accessed using the Matlab commands `doc` and `help`. For example, to open the help page for `RSKsmooth`, type: `doc RSKsmooth` at the Matlab command prompt.

Open the data, print the channel names, and plot a few profiles

First, while in the RSKtools/QuickStart directory, open a connection to the RSK logger database file:

```
rsk = RSKopen('../sample.rsk');

% print a list of all the channels in the rsk file
RSKprintchannels(rsk)

% read the upcast from profiles 1 - 20
rsk = RSKreadprofiles(rsk, 'profile', 1:20, 'direction', 'up');

% plot a few profiles of temperature, conductivity, and chlorophyll
RSKplotprofiles(rsk,'profile',[1 10 20],...
    'channel',{'temperature','conductivity','chlorophyll'});
```

```
Model: RBRconcerto
Serial ID: 80231
Sampling period: 0.167 second
    index         channel              unit

    _____    ________________    _____________

      1      {'Conductivity'}    {'mS/cm'    }
      2      {'Temperature' }    {'°C'       }
      3      {'Pressure'    }    {'dbar'     }
      4      {'Dissolved O2'}    {'%'        }
      5      {'Turbidity'   }    {'NTU'      }
      6      {'PAR'         }    {'µMol/m²/s'}
      7      {'Chlorophyll' }    {'µg/l'     }
```



Add station data

`RSKaddstationdata` appends station data to the profile data structure. The allowable fields are: latitude, longitude, station, cruise, vessel, depth, date, weather, crew, comment and description. The function is vectorized, which allows multiple station data inputs for multiple profiles. When there is only one station data input specified for multiple profiles, all profiles will be assigned with the same value.

Station data is written into both the CSV and ODV file headers when the data is exported with `RSK2CSV` and `RSK2ODV`.

```
% apply the same station data to each profile
rsk = RSKaddstationdata(rsk,'latitude',45,'longitude',-25,...
                        'station',{'SK1'},'vessel',{'R/V RBR'},...
                        'cruise',{'Skootamatta Lake 1'});

% or use vectorized inputs
rsk = RSKaddstationdata(rsk,'profile',4:6,'station',{'S1','S2','S3'},...
                        'latitude',[45,44,46],...
                        'longitude',[-25,-24,-23],...
                        'comment','RSKtools demo');
```

To view the station data in the 4th profile:

```
disp(rsk.data(4))

           tstamp: [137×1 double]
           values: [137×7 double]
        direction: 'up'
    profilenumber: 4
         latitude: 45
        longitude: -25
          station: {'S1'}
           cruise: {'Skootamatta Lake 1'}
           vessel: {'R/V RBR'}
          comment: {'RSKtools demo'}
```

Derive sea pressure from total pressure

We suggest deriving sea pressure first, especially when an atmospheric pressure other than the nominal value of 10.1325 dbar is desired. In this example we'll take atmospheric pressure to be 10 dbar.

```
rsk = RSKderiveseapressure(rsk,'patm',10);
```

Data post-processing

What follows is a generic recipe for post-processing RBR CTD data. It is a guideline - RBR CTDs are used in many environments, and with many sensor packages, and are profiled from a variety of vessels (or no vessel at all). While the basic approach here is relevant for many cases, the processing *parameters* may not apply widely.

First, keep a copy of the raw data to compare with the processed data.

```
raw = rsk;
```

Correct for A2D zero-order hold

The analog-to-digital (A2D) converter on RBR instruments must recalibrate periodically. In the time it takes for the calibration to finish, one or more samples are missed. The instrument firmware fills the missed sample with the same data measured during the previous sample, a technique called a zero-order hold.

`RSKcorrecthold` identifies zero-hold points by finding where consecutive differences of each channel are equal to zero, and then replaces these samples with a NaN or an interpolated value. See the RSKtools on-line user manual for further information.

```
rsk = RSKcorrecthold(rsk,'action','interp');
```

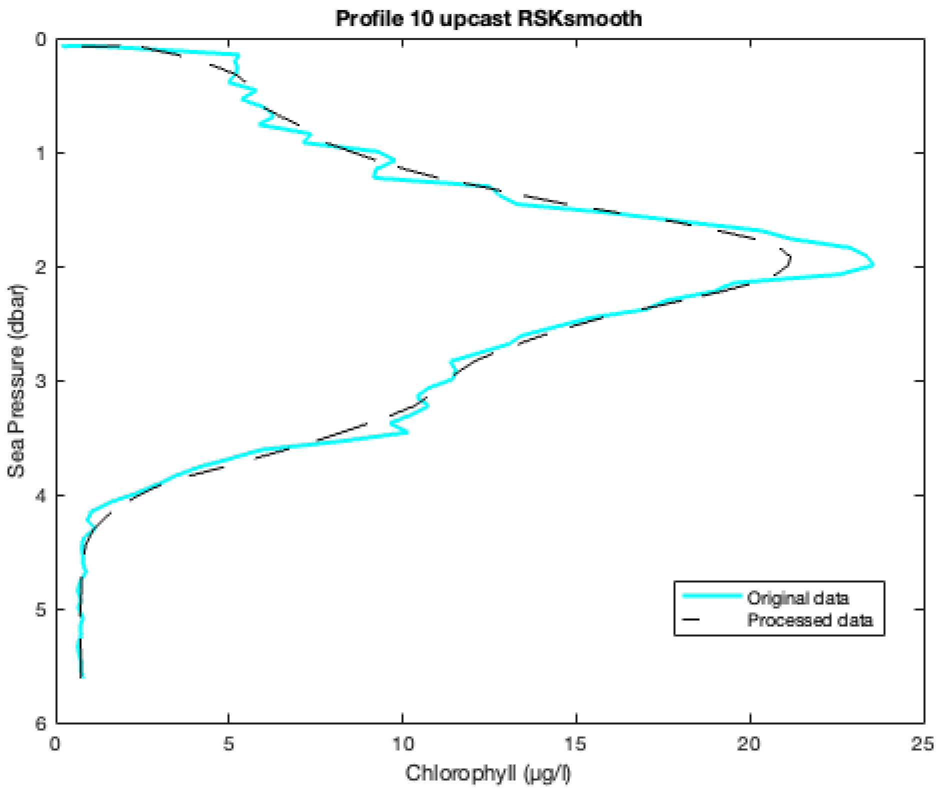
Low-pass filtering

Low-pass filtering is commonly used to reduce noise and to match sensor time constants, typically for temperature and conductivity. Users may also wish to filter other channels to simply reduce noise (e.g., optical channels such as chlorophyll-a or turbidity).

Most RBR instruments designed for profiling are equipped with thermistors that have a time constant of 100 ms, which is "slower" than the conductivity cell. When the time constants are different, salinity will contain spikes at strong gradients. One solution is to "slow down" the conductivity sensor to match the thermistor. In this example dataset, the logger sampled at 6 Hz (found using `readsamplingperiod(rsk)`), so a 5 sample running average provides more than sufficient smoothing to match the time response of the conductivity sensor to the thermistor.

Note: All functions that alter existing channels have an optional input called `'visualize'` that, when activated, plots data before and after applying the function. Users specify which profile(s) to visualize.

```
rsk = RSKsmooth(rsk,'channel',{'temperature','conductivity'},...
                'windowLength', 5);
rsk = RSKsmooth(rsk,'channel','chlorophyll','windowLength',9,...
                'visualize', 10);
```



Alignment of conductivity and temperature

Conductivity and temperature often need to be aligned in time to account for the fact these sensors are not always co-located on the logger. The implication is that, under dynamic conditions (e.g., profiling), the sensors are measuring a slightly different parcel of water at any instant.

Furthermore, sensors with long time constants introduce a time lag to the data. For example, dissolved oxygen sensors often have a long time constant, and this delays the measurement relative to the true value. This can be fixed to some degree by advancing the sensor data in time.

When temperature and conductivity are misaligned, salinity will contain spikes at sharp interfaces and a bias in continuously stratified environments. Properly aligning the sensors, together with matching the time response, will minimize salinity spiking and bias.

A common approach to determine the optimal lag is to compute and plot salinity for a range of lags, and choose the lag (often by eye) with the smallest salinity spikes at sharp temperature interfaces.

As an alternative approach, RSKtools includes a function called `RSKcalculateCTlag` that estimates the optimal lag between conductivity and temperature by minimizing salinity spiking. We currently suggest using both approaches to check for consistency. See the `RSKcalculateCTlag` help page for more information.

As a rough guide, temperature from a CTD equipped with the red combined CT cell and a fast thermistor typically requires only a very small time advance (perhaps tens of milliseconds). Temperature from a CTD equipped with a cylindrical black conductivity cell (with the thermistor on the sensor endcap) typically requires a temperature lag correction of about 0.1 s to 0.3 s (1 or 2 samples at 6 Hz).

```
% required shift of C relative to T for each profile
lag = RSKcalculateCTlag(rsk,'seapressurerange',[1 3]);
lag = -lag; % to advance temperature
lag = median(lag); % select best lag for consistency among profiles
rsk = RSKalignchannel(rsk,'channel','temperature','lag',lag);
```

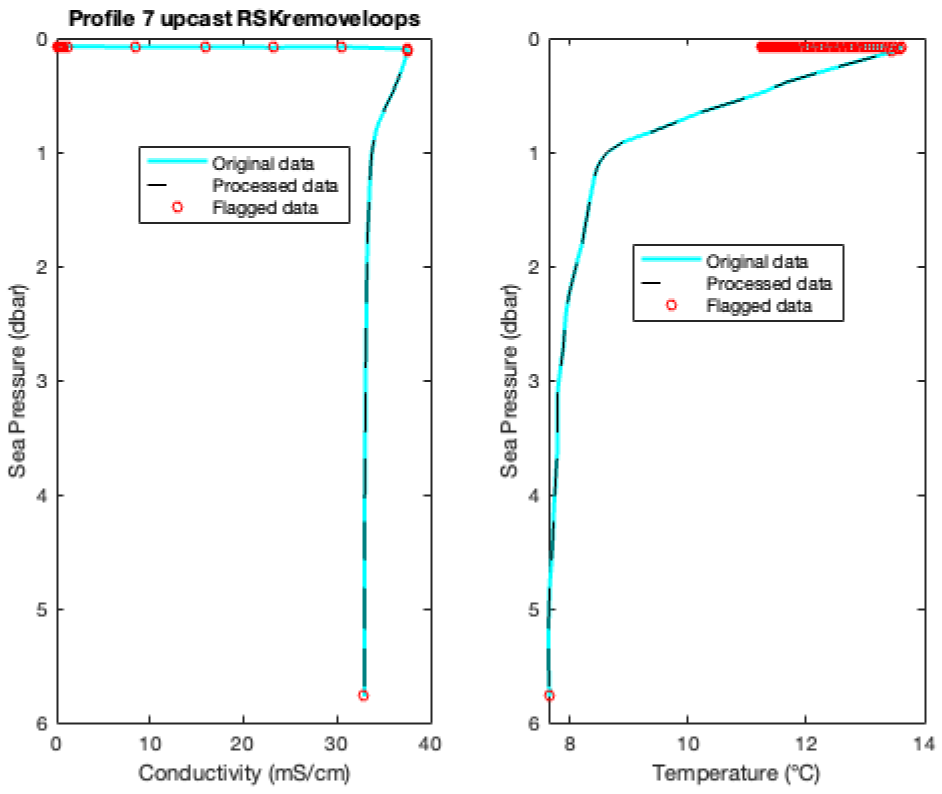
Users wishing to learn more about dynamic sensor corrections and RBR CTDs are encouraged to watch a special RBR webinar on dynamic errors from May 2020 (Youtube and PDF).

Remove loops

Working in rough seas can cause the CTD profiling rate to vary, and even change sign (i.e., the CTD momentarily changes direction). When this happens, the CTD effectively samples its own wake, degrading the quality of the profile in regions of strong gradients. The measurements taken when the instrument is profiling too slowly or during a pressure reversal should not be used for further analysis. We recommend using `RSKremoveloops` to flag and treat the data when the instrument 1) falls below a threshold speed and 2) when the pressure reverses (the CTD "loops"). Before using `RSKremoveloops`, use `RSKderivedepth` to calculate depth from sea pressure, and then use `RSKderivevelocity` to calculate profiling rate.

In the example dataset, good data is collected on the *upcast*. `RSKremoveloops`, when applied to this data, removes data when the instrument profiled slowly near the surface.

```
rsk = RSKderivedepth(rsk);
rsk = RSKderivevelocity(rsk);

% Apply the algorithm
rsk = RSKremoveloops(rsk,'threshold',0.3,'visualize',7);
```



Derived variables

RSKtools includes a few convenience functions to derive common oceanographic variables. For example, `RSKderivesalinity` computes Practical Salinity using the TEOS-10 GSW function `gsw_SP_from_C`. The TEOS-10 GSW Matlab toolbox is freely available from http://www.teos-10.org/software.htm. The result is stored in the data table along with other measured and derived channels.

```
rsk = RSKderivesalinity(rsk);
rsk = RSKderivesigma(rsk);

raw = RSKderivesalinity(raw);
raw = RSKderivesigma(raw);

% print a list of channels in the rsk file
RSKprintchannels(rsk)
```

```
Model: RBRconcerto
Serial ID: 80231
Sampling period: 0.167 second
    index          channel              unit
    _____    ___________________    _____________

      1      {'Conductivity'   }    {'mS/cm'    }
      2      {'Temperature'    }    {'°C'       }
      3      {'Pressure'       }    {'dbar'     }
      4      {'Dissolved O2'   }    {'%'        }
```

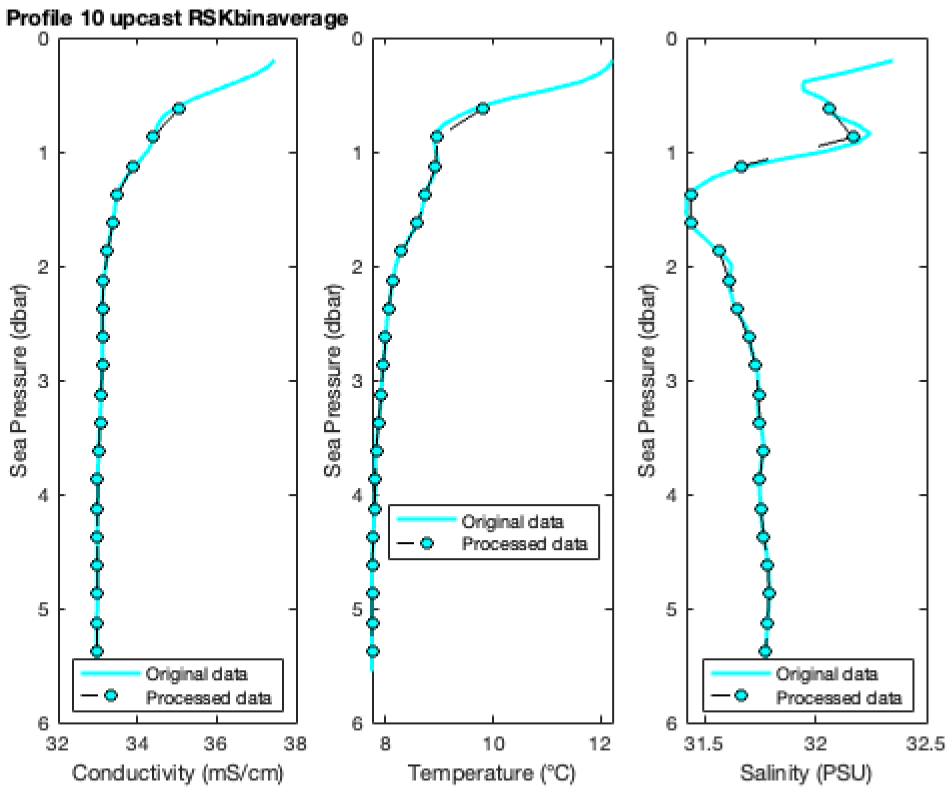
```
     5      {'Turbidity'      }    {'NTU'      }
     6      {'PAR'            }    {'µMol/m²/s'}
     7      {'Chlorophyll'    }    {'µg/l'     }
     8      {'Sea Pressure'   }    {'dbar'     }
     9      {'Depth'          }    {'m'        }
    10      {'Velocity'       }    {'m/s'      }
    11      {'Salinity'       }    {'PSU'      }
    12      {'Density Anomaly'}    {'kg/m³'    }
```

Note that users also have the choice to use the SeaWater toolbox, as well. Please read the `RSKsettings` page of the RSKtools on-line user manual for more information.

Bin average all channels by sea pressure

Bin averaging reduces sensor noise and ensures that each profile is referenced to a common grid. The latter is often an advantage for plotting data as "heatmaps." `RSKbinaverage` allows users to bin channels according to any reference, but the most common choices are time, depth, and sea pressure. It also can handle grids with a variable bin size. In the following, the data are averaged into 0.25 dbar bins.

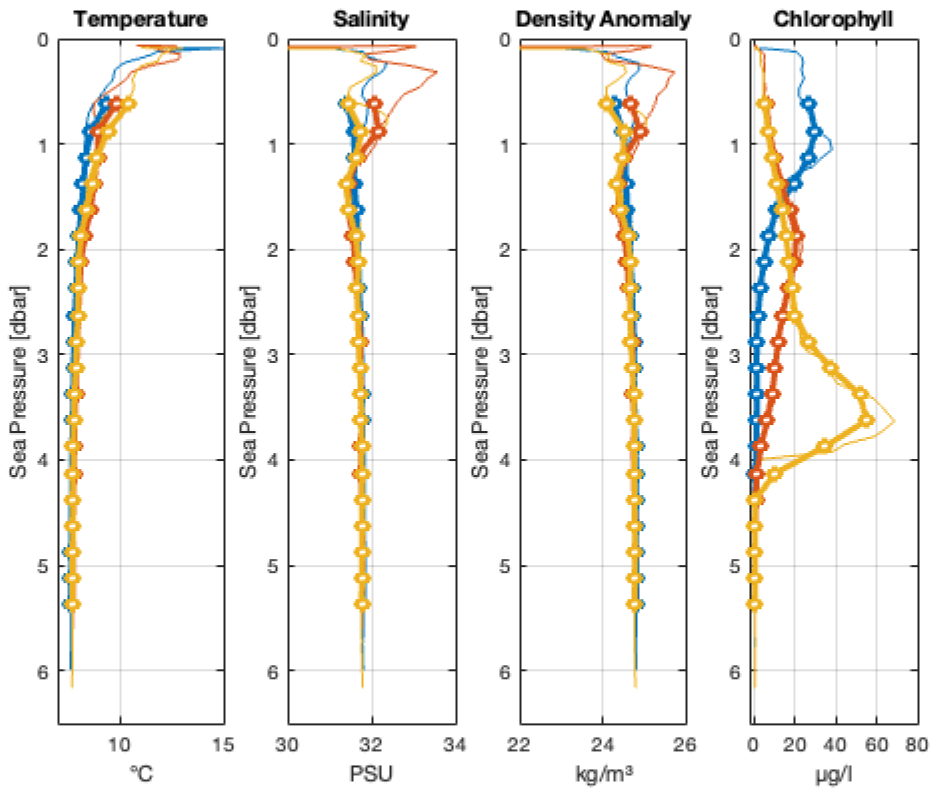
```
rsk = RSKbinaverage(rsk, 'binBy', 'Sea Pressure', 'binSize', 0.25, ...
                    'boundary', [0.5 5.5],'direction', 'up',...
                    'visualize', 10);
h = findobj(gcf,'type','line');
set(h(1:2:end),'marker','o','markerfacecolor','c')
```



Compare the raw and processed data

Use `RSKplotprofiles` to compare the binned data to the raw data for a few example profiles. Processed data are represented with thicker lines.

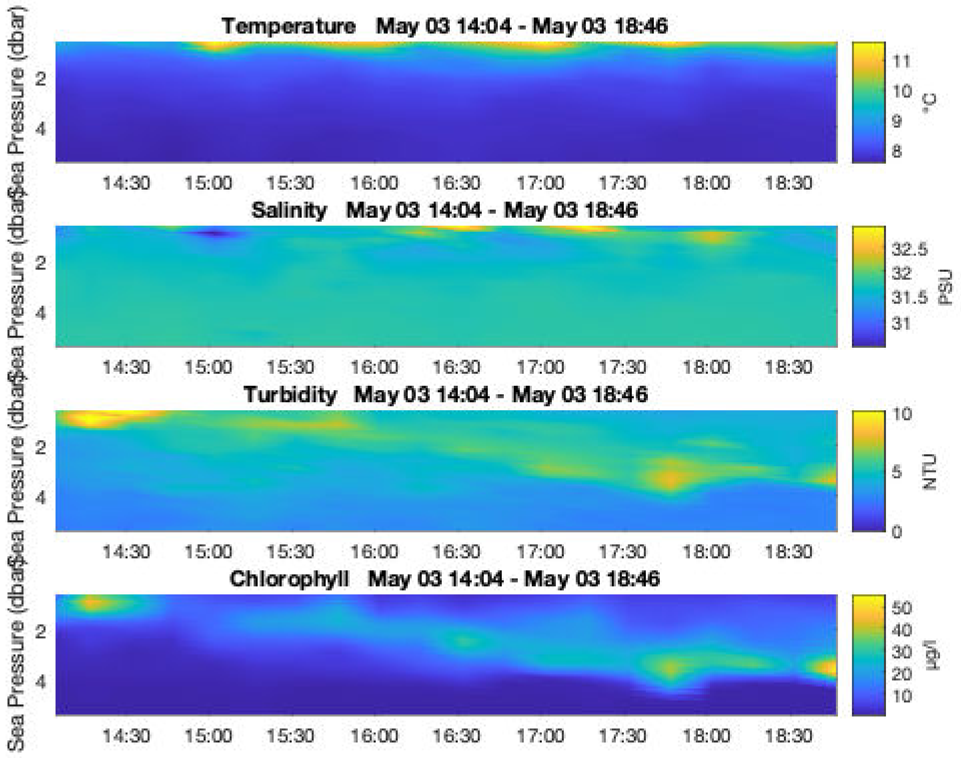
```
figure
channel = {'temperature','salinity','density anomaly','chlorophyll'};
profile  = [3 10 20];
[h1,ax] = RSKplotprofiles(raw,'profile',profile,'channel',channel);
h2 = RSKplotprofiles(rsk,'profile',profile,'channel',channel);
set(h2,'linewidth',3,'marker','o','markerfacecolor','w')
set(ax(1),'xlim',[7 15])
set(ax(2),'xlim',[30 34])
set(ax(3),'xlim',[22 26])
set(ax(4),'xlim',[-2 80])
set(ax,'ylim',[0 6.5])
```



Visualize data with a 2D plot

`RSKimages` generates a time/depth heat-map of a channel. The x-axis is time; the y-axis is a reference channel (default is sea pressure). All of the profiles must be evaluated on the same reference channel grid, which is accomplished with `RSKbinaverage`. The function supports customizable rendering to determine the length of gap shown on the plot. For more details, see [RSKtools on-line user manual]

```
figure
[im_hdl,ax_hdl] = RSKimages(rsk,'channel',
{'temperature','salinity','turbidity','chlorophyll'},'direction','up');
caxis(ax_hdl(3),[0 10])
```



Export logger data to CSV files

`RSK2CSV` writes logger data and metadata to one or more CSV files. The CSV files contain a header with important logger metadata, station metadata (if it exists), a record of the processing steps made by RSKtools, and a row of variable names and units above each column of channel data. If the data has been parsed into profiles, then one file will be written for each profile and an extra column called 'cast_direction' will be included. The column will contain 'd' or 'u' to indicate whether the sample is part of the downcast or upcast, respectively. Users can select which channels and profiles to write, the output directory, and also specify additional comments to be placed after the metadata in the file header.

```
RSK2CSV(rsk,'channel',{'Depth','Temperature','Salinity','Dissolved O2'}, 'pro-
file', 1:3 ,'comment','Hey Jude');
```

RSKtools also has export functions to write Ocean Data View files, MAT files, and Ruskin RSK files.

Display a summary of all the processing steps

Type `rsk.log{:,2}` at the command prompt.

Other Resources

We recommend reading:

- the RSKtools on-line user manual for detailed RSKtools function documentation.
- the RSKtools Getting Started for an introduction on how to load RBR data into Matlab from RSK files, make plots, and access the data.

About this document

This document was created using Matlab™ Markup Publishing. To publish it as an HTML page, run the command:

```
publish('PostProcessing.m');
```

See `help publish` for more document export options.

Published with MATLAB® R2023b