Getting Started with RSKtools

Table of Contents

RSKtools v3.5.0; RBR Ltd. Ottawa ON, Canada; support@rbr-global.com; 2020-06-26

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.

Installing

The latest stable version of `RSKtools` can be found at http://www.rbr-global.com/support/matlab-tools.

- Download and unzip the archive (to `~/matlab/RSKtools`, for instance)
- Add the folder to your path using (`addpath ~/matlab/RSKtools` or launch the path editor gui (`pathtool`).
- type `help RSKtools` to get an overview and take a look at the examples.

Examples of use

The first step is to make a connection to the RSK file with `RSKopen`. `RSKopen` reads various metadata tables from the RSK file that contain information about the instrument channels, sampling configuration, and profile events. It also reads a downsampled version of the data if the complete dataset is large.

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

```
Setting default values for all RSKtools parameters
mksqlite Version 2.5 build: Unversioned directory, an interface from
 MATLAB to SQLite
(c) 2008-2017 by Martin Kortmann <mail@kortmann.de>
                 Andreas Martin  <andimartin@users.sourceforge.net>
```

```
based on SQLite Version 3.16.2 - http://www.sqlite.org
mksqlite utilizes:
 - DEELX perl compatible regex engine Version 1.3 (Sswater@gmail.com)
 - BLOSC/LZ4 1.3.0-rc3.dev compression algorithm (Francesc Alted /
 Yann Collett)
 - MD5 Message-Digest Algorithm (RFC 1321) implementation by Alexander
 Peslyak

Platform: MACI64, little endian
```

The structure returned after opening an RSK file will look something like:

```
disp(rsk)
```

```
          toolSettings: [1x1 struct]
                dbInfo: [1x1 struct]
           instruments: [1x1 struct]
    instrumentChannels: [10x1 struct]
              channels: [7x1 struct]
                epochs: [1x1 struct]
             schedules: [1x1 struct]
           deployments: [1x1 struct]
            continuous: [1x1 struct]
            parameters: [1x1 struct]
        parameterKeys: [23x1 struct]
          appSettings: [1x1 struct]
              ranging: [7x1 struct]
     instrumentSensors: [0x1 struct]
                region: [762x1 struct]
            regionCast: [508x1 struct]
              profiles: [1x1 struct]
                   log: {[7.3797e+05]  '../sample.rsk opened using
 RSKtools...'}
```

Nearly all of the fields in `rsk` are not directly useful to users; instead they are accessed by RSKtools functions.

Reading data from the rsk file

To read data from the instrument, use the `RSKreaddata` function. RSKreaddata will read the full dataset by default. Because RSK files can store a large amount of data, it may be preferable to read a subset of the data, specified using start and end times in Matlab `datenum` format.

```
t1 = datenum(2014, 05, 03);
t2 = datenum(2014, 05, 04);
rsk = RSKreaddata(rsk, 't1', t1, 't2', t2);
```

Note that the logger data can be found in the `data` field of the `rsk` structure:

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

```
    tstamp: [22346x1 double]
    values: [22346x7 double]
```

where `rsk.data.tstamp` contains the sample timestamps in Matlab datenum format, and `rsk.data.values` contains the channel data. Each column in `rsk.data.values` contains data from a different channel. The channel names and units for each column in `data` are contained in `rsk.channels`. To view of all channel names and units, run:

```
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'
```

To plot the data as a time series, use `RSKplotdata`.

Working with profiles

`RSKreaddata` reads the instrument data into a single time series as opposed to a series of profiles. When Ruskin downloads data from a logger with a pressure channel, it will detect, timestamp, and record profile upcast and downcast "events" automatically.

Users may wish to read the data as a series of profiles instead of a time series. The function `RSKreadprofiles` reads CTD data, and organizes it as a collection of profiles according to the profile event timestamps. For example, to read the upcast and downcast of profiles 6 to 8 from the RSK file, run:

```
rsk = RSKreadprofiles(rsk, 'profile', 6:8, 'direction', 'both');
```

After reading the profiles, they can be plotted very easily with `RSKplotprofiles`.

Note: If profiles have not been detected by the logger or Ruskin, or if the profile timestamps do not correctly parse the data into profiles, the functions `RSKfindprofiles` and `RSKtimeseries2profiles` can be used. The `pressureThreshold` argument, which determines the pressure reversal required to trigger a new profile, and the `conductivityThreshold` argument, which determines if the logger is out of the water, can be adjusted to improve profile detection when the profiles were very shallow, or if the water was very fresh.

RSKtools includes a convenient plotting option to overlay the pressure data with information about the profile events. For details please consult the `RSKplotdata` page in the RSKtools on-line user manual.

Deriving new channels from measured channels

In this particular example, Practical Salinity can be derived from conductivity, temperature, and pressure because the file comes from a CTD-type instrument. `RSKderivesalinity` is a wrapper for the

TEOS-10 GSW function `gsw_SP_from_C`, and it adds a new channel called `Salinity` as a column in `rsk.data.values`. The TEOS-10 GSW Matlab toolbox is freely available from http://teos-10.org/software.htm. Salinity is a function of sea pressure, and sea pressure must be derived from the measured total pressure before computing salinity. In the following example, the default value of atmospheric pressure at sea level, 10.1325 dbar, is used.

```
rsk = RSKderiveseapressure(rsk);
rsk = RSKderivesalinity(rsk);
```

A handful of other EOS-80 derived variables are supported, such as potential temperature and density. RSKtools also has wrapper functions for a few common TEOS-10 variables such as Absolute Salinity.

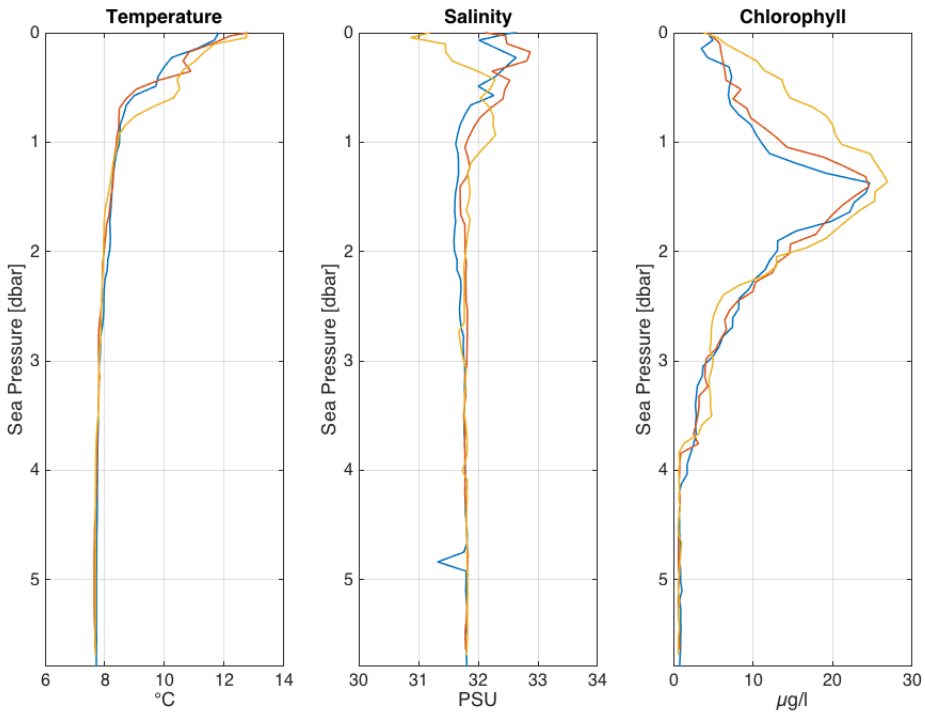
Note that users also have the choice to use the SeaWater toolbox, as well. Please read the `RSKsettings` page of the RSKtools on-line user manual for more information.

Note: Salinity, sea pressure, and other channels added by RSKtools should be derived after using `RSKreadprofiles`. `RSKreadprofiles` reads raw data from the RSK data *file*, instead of referring to the data in the Matlab RSK *structure* (see `RSKtimeseries2profiles` for organizing data in the `rsk` structure into profiles).

Plotting

RSKtools contains a number of convenient plotting utilities. If the data was organized as profiles, then it can be easily plotted with `RSKplotprofiles`. For example, to plot the upcasts of temperature, salinity, and chlorophyll, from this example, run:

```
[l_hdls,ax_hdls] = RSKplotprofiles(rsk, 'channel',
 {'temperature','salinity','chlorophyll'},'direction', 'up');
```



Customising plots

The plotting functions return graphics handles enabling access to the axes and line objects. For example, the line handles are stored in a matrix containing a column for each channel subplot and a row for each profile.

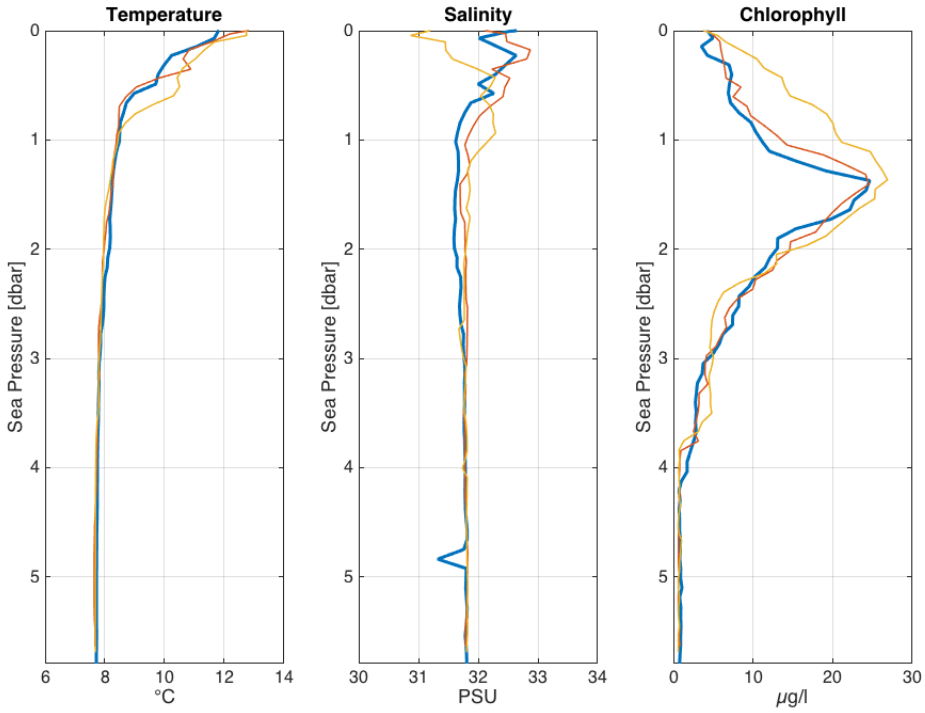
```
disp(l_hdls)
```

```
  3x3 Line array:

    Line    Line    Line
    Line    Line    Line
    Line    Line    Line
```

To increase the line width of the first profile in all subplots, run:

```
set(l_hdls(1,:),'linewidth',3);
```



Accessing individual channels and profiles

The channel data are stored in `rsk.data`. If the data was parsed into profiles, `data` is a 1xN structure array, where each element is an upcast or downcast from a single profile containing both the timestamps and a matrix of channel data. At times, it is necessary to extract data and store it in a new array. `RSKtools` has functions to access the data from particular channels and profiles. For example, to access the timestamps, sea pressure, temperature, and dissolved O2 from the upcast of the 1st profile, run:

```
profind = getdataindex(rsk,'direction','up','profile',1);
```

```
[tempcol,o2col,prescol] = getchannelindex(rsk,
{'temperature','dissolved o2','sea pressure'});

time        = rsk.data(profind).tstamp;
seapressure = rsk.data(profind).values(:,prescol);
temperature = rsk.data(profind).values(:,tempcol);
o2          = rsk.data(profind).values(:,o2col);
```

Other Resources

We recommend reading:

- The RSKtools on-line user manual for detailed RSKtools function documentation.
- The RSKtools post-processing guide for an introduction on how to process RBR profiles with RSKtools. The post-processing suite contains, among other things, functions to low-pass filter, align, de-spike, trim, and bin average the data. It also contains functions to export the data to ODV and CSV files.

About this document

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

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

See `help publish` for more document export options.

Published with MATLAB® R2015b