

---

# RSKtools for Matlab access to RBR data

## Table of Contents

Introduction .....	1
Installing .....	1
Examples of use .....	1
Working with profiles .....	3
Customizing plots .....	3
See RSKtools_vignette2 .....	4
Future plans .....	4
About this document .....	5

RSKtools v2.0.0; RBR Ltd. Ottawa ON, Canada; [support@rbr-global.com](mailto:support@rbr-global.com); 2017-07-06

## Introduction

RSKtools provides some convenience functions for common data extraction (e.g. extracting profiles from a continuous dataset) and visualisation (plotting individual profiles). From this version on, we are expanding on our post-processing functions. For plans for future additions, see the Future plans section.

## Installing

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

- Unzip the archive (to ~/matlab/RSKtools, for instance)
- Add the folder to your path inside matlab (addpath ~/matlab/RSKtools or some nifty GUI thing)
- type help RSKtools to get an overview and take a look at the examples.

## Examples of use

A connection to the database must be made to work with an RSK file using RSKtools. This connection is made using the RSKopen( ) function. Note that RSKopen does not actually read the data, but reads a /thumbnail/ of the data, which is up to 4000 samples long. The structure returned after opening an RSK looks something like:

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

rsk =
    dbInfo: [1x1 struct]
instrumentChannels: [7x1 struct]
    channels: [7x1 struct]
    epochs: [1x1 struct]
```

```
        schedules: [1x1 struct]
        deployments: [1x1 struct]
        instruments: [1x1 struct]
        appSettings: [1x1 struct]
        ranging: [7x1 struct]
        continuous: [1x1 struct]
        parameters: [1x1 struct]
        parameterKeys: [23x1 struct]
        thumbnailData: [1x1 struct]
        region: [762x1 struct]
        regionCast: [508x1 struct]
        profiles: [1x1 struct]
        log: {[7.3688e+05] 'sample.rsk opened using
RSKtools v1...']}
```

To read the actual data, we use the `RSKreaddata()` function. If given with one input argument (the variable name of the RSK structure) it will read the entire data set. Because RSK files can store a large amount of data, it may be preferable to read a subset of the data, specified using a start and end time (in Matlab datenum format, which is defined as the number of days since January 0, 0000).

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

Note that the data structure can be found in the structure at:

```
rsk.data

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

In this example, because the instrument is a "CTD"-type instrument, we can derive a new channel called Salinity. `RSKderivesalinity` adds the data to all the appropriate places (using the Practical Salinity Scale). The salinity calculation is performed by the [TEOS-10](http://teos-10.org/software.htm) package, which is available from <http://teos-10.org/software.htm>.

```
rsk = RSKderivesalinity(rsk);
rsk.channels.longName

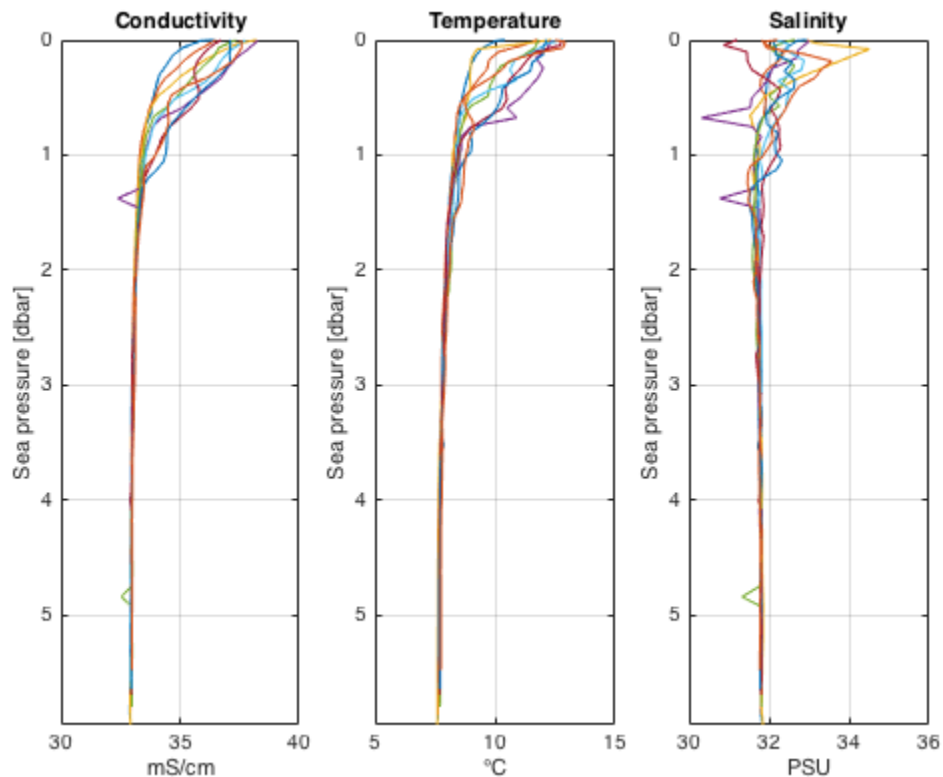
ans =
Conductivity
ans =
Temperature
ans =
Pressure
ans =
Dissolved O2
ans =
Turbidity
ans =
PAR
ans =
Chlorophyll
ans =
Salinity
```

## Working with profiles

Profiling loggers with recent versions of firmware contain the ability to detect and log profile "events" automatically; these are denoted as "downcasts" and "upcasts". The function `RSKreadprofiles()` extracts individual profiles from the raw data, based on the previously identified profiling events. Then, quick plots of the profiles can be made using the `RSKplotprofiles()` function.

If profiles have not been detected by the logger or Ruskin. The function `RSKfindprofiles()` 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 for short or fresh water profiles.

```
% load the second to tenth profiles in both directions (upcast and  
downcast)  
rsk = RSKreadprofiles(rsk, 'profile', 2:10, 'direction', 'both');  
rsk = RSKderivesalinity(rsk);  
  
% plot the upcasts of Conductivity, Temperature and Salinity  
handles = RSKplotprofiles(rsk, 'channel',  
    {'conductivity', 'temperature', 'salinity'}, 'direction', 'up');
```



## Customizing plots

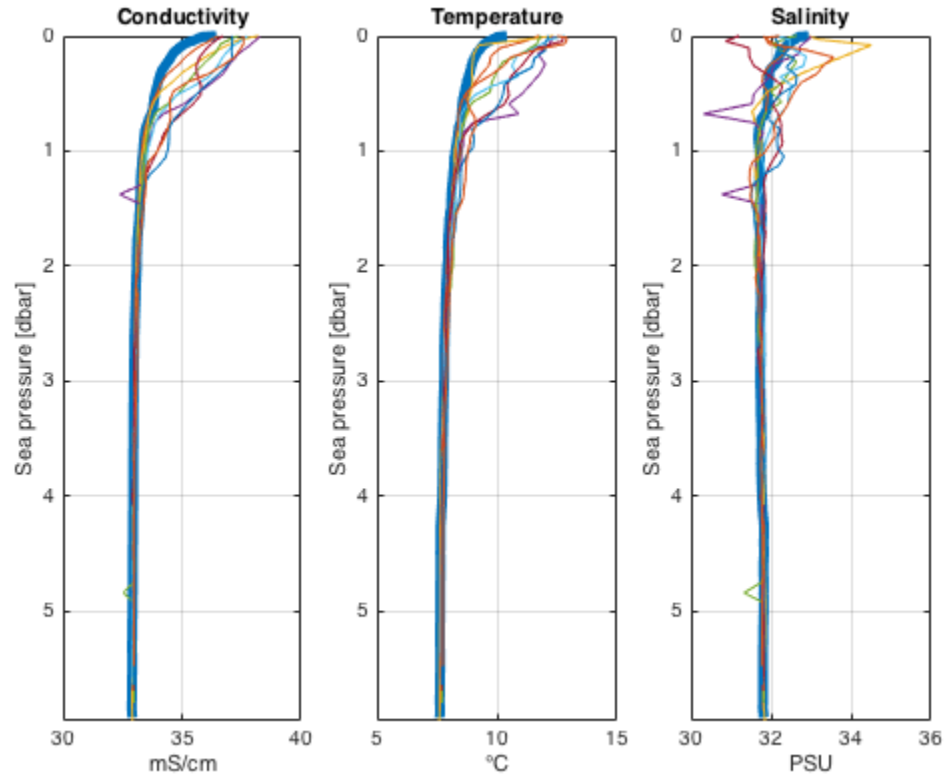
All plotting functions return a handle which enables access to the lines in the plot. The output is a matrix containing a column for each channel subplot and a row for each profile.

handles

```
% To increase the linewidth of the first profile in all subplots  
(set(handles(1,:),{'linewidth'},{5}));
```

```
handles =  
9x3 Line array:
```

Line	Line	Line
Line	Line	Line
Line	Line	Line
Line	Line	Line
Line	Line	Line
Line	Line	Line
Line	Line	Line
Line	Line	Line
Line	Line	Line



## See RSKtools\_vignette2

A second vignette is available for information on getting started with post-processing functions.

## Future plans

- Function to write metadata, log and data to a file.

- Wave processing functions.
- A TS plotting function.

## About this document

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

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

See `help publish` for more document export options.

*Published with MATLAB® R2015b*