



RBR Software Tools: Ruskin and RSKtools

Greg Johnson, PhD
President, RBR

December 09, 2020

Ruskin and RSKtools

Ruskin

- **Instrument simulation and configuration;**
- **Data download, visualization, annotation, and export.**

RSKtools

- **Link between RSK files and Matlab**
- **Data post-processing**



Ruskin

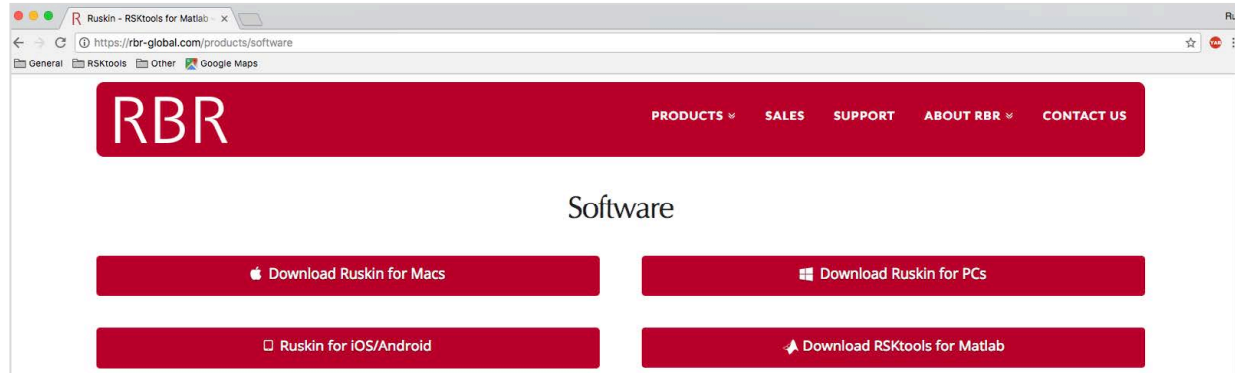
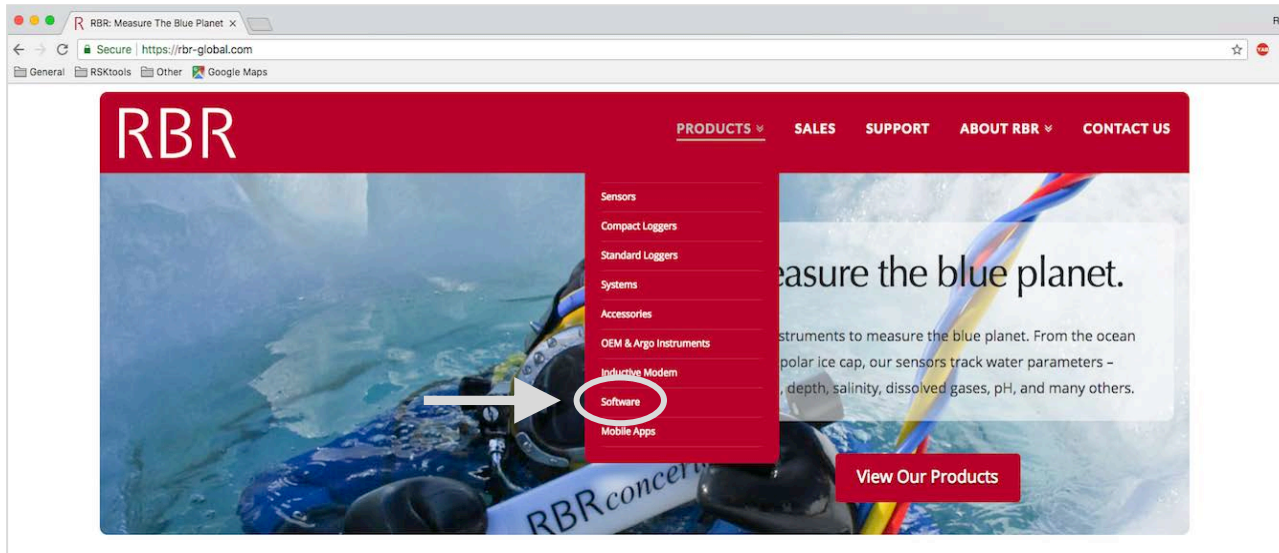
One app to rule them all

- macOS/Windows10/iOS/Android
- Every instrument we've made since 2005
- Complete deployment simulation
- Speaks your language
- Automatic update
- Contains all RUGs and RIGs
- Continuous improvement - 4x per year

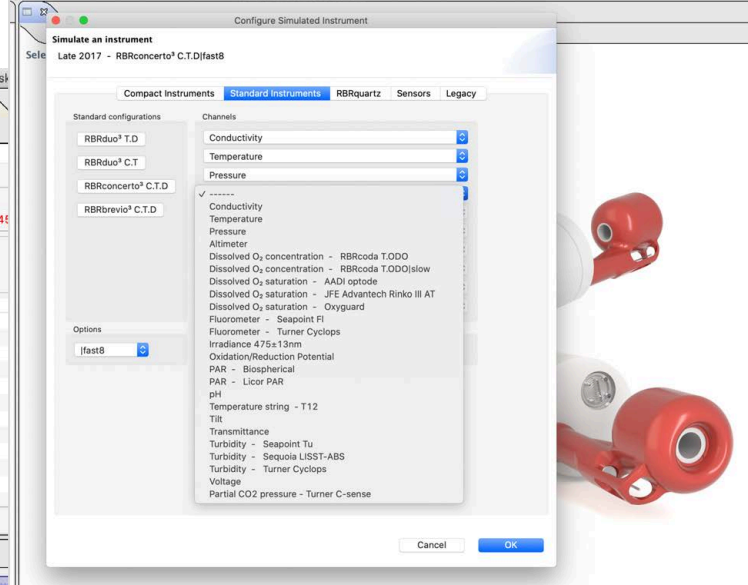
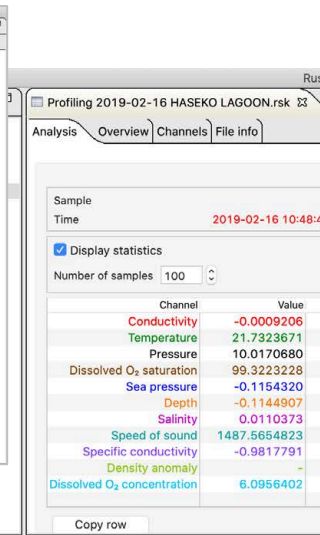
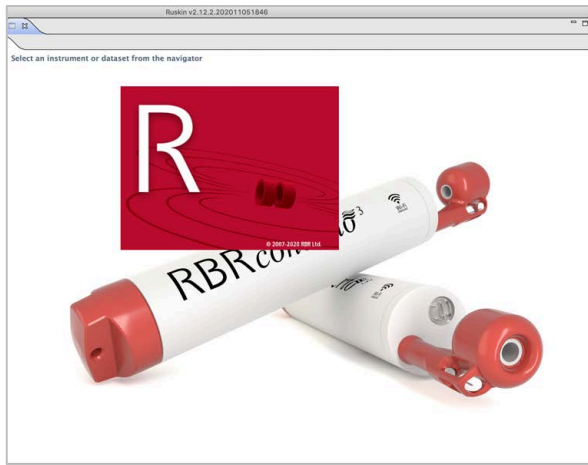


RBR

Download

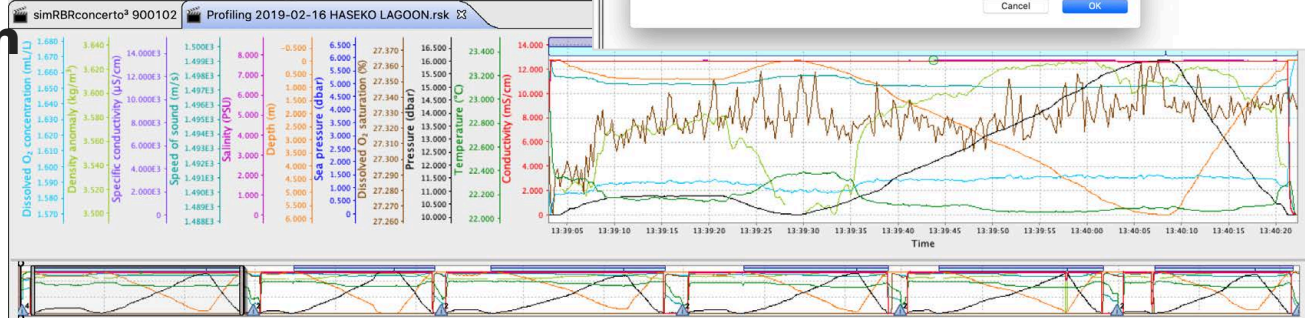


RBR



Instrument simulation and configuration.

Data download, visualization, annotation, and export.



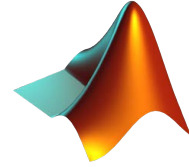
Live Demo!

RBR

RSKtools

What is RSKtools?

- Free and open source tool box written in MATLAB
- Provides access to Ruskin RSK data files
 - Ruskin RSK files are SQLite databases
 - SQLite most widely deployed database engine
- Read, post-process, and visualize RBR data
- Current version: v3.4.1
- Compatible with Matlab R2013b and later
- Does not require additional paid toolboxes



RBR

49 user-faced functions in v3.4.1

READ

Openers and Readers

RSKopen
RSKreaddata
RSKreadprofiles
RSKreadburstdata
RSKreadcalibrations
RSKfindprofiles
RSKtimeseries2profiles
CSV2RSK

PROCESS

Calculators

RSKderiveddepth
RSKderivesalinity
RSKderiveseapressure
RSKderivevelocity
RSKderiveBPR
RSKderiveC25
RSKderiveO2
RSKderivebuoyancy
RSKderivesigma
RSKderiveSA
RSKderivetheta
RSKderivesoundspeed

Post-Processors

RSKcalculateCTlag
RSKalignchannel
RSKbinaverage
RSKcorrecthold
RSKcorrectTM
RSKcorrecttau
RSKdespike
RSKremoveloops
RSKsmooth
RSKtrim
RSKgenerate2D
RSKcentrebusttimestamp

VIEW & EXPORT

Plotters

RSKplotdata
RSKplotprofiles
RSKimages
RSKplotTS
RSKplotburstdata
RSKplotdownsample

Exporters

RSK2CSV
RSK2ODV
RSK2RSK
RSK2MAT

OTHER

Miscellaneous

RSKcreate
RSKsettings
RSKaddchannel
RSKaddstationdata
RSKappendtolog
RSKremovecasts
RSKprintchannels

Help

Contents
README.md
QuickStart/Standard
QuickStart/Postprocessing

RBR

Help and Support

- QuickStart demo
 - [Getting started with RSKTOOLS \(pdf\)](#)
 - [Post-processing RBR data with RSKtools \(pdf\)](#)
- Online manuals
 - docs.rbr-global.com/rsktools
- type `help RSKtools` in the MATLAB command window
- Bugs: support@rbr-global.com



Introduction to RSKtools functions

CTD data post-processing example

Simplified CTD data post-processing pipeline:

	Step	RSKtools functions	See also
1	Open RSK file and read data	RSKopen, RSKreadprofiles, RSKreaddata, RSKprintchannels	RSKtimeseries2profiles, RSKreadburstdata
2	Visualize raw data	RSKplotprofiles	RSKplotdownsample, RSKplotdata
3	Add station metadata	RSKaddstationdata	
4	Calculate sea pressure and depth	RSKderiveseapressure, RSKderiveddepth	
5	Low pass filter C & T	RSKsmooth	
6	Align temperature to conductivity	RSKalignchannel	RSKcalculateCTlag, RSKcorrecttau
7	Remove ship heave (“loops”)	RSKderivevelocity, RSKremoveloops	RSKdespike
8	Derive variables	RSKderivesalinity, RSKderivetheta, RSKderivesigma	RSKderiveC25, RSKderivesoundspeed, RSKderiveO2, RSKderiveSA
9	Bin average by depth	RSKremovecasts, RSKbinaverage	
10	Visualize processed data	RSKplotprofiles	RSKplotdata, RSKimages, RSKplotTS
11	Export	RSK2CSV, RSK2RSK, RSK2ODV	RSK2MAT

- **Read**
- **View**
- **Process**
- **Export**

RSKopen

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

- Extract **metadata** such as channel information, profile start and end times, serial number, sampling rate, etc.

RSKreaddata

```
rsk = RSKreaddata(rsk, 't1', tstart, 't2', tend);
```

- Read data from all channels as a *time series*. Optionally specify start and end times to read select periods.

RSKreadprofiles

```
rsk = RSKreadprofiles(rsk, 'profile', 1:4, 'direction', 'down');
```

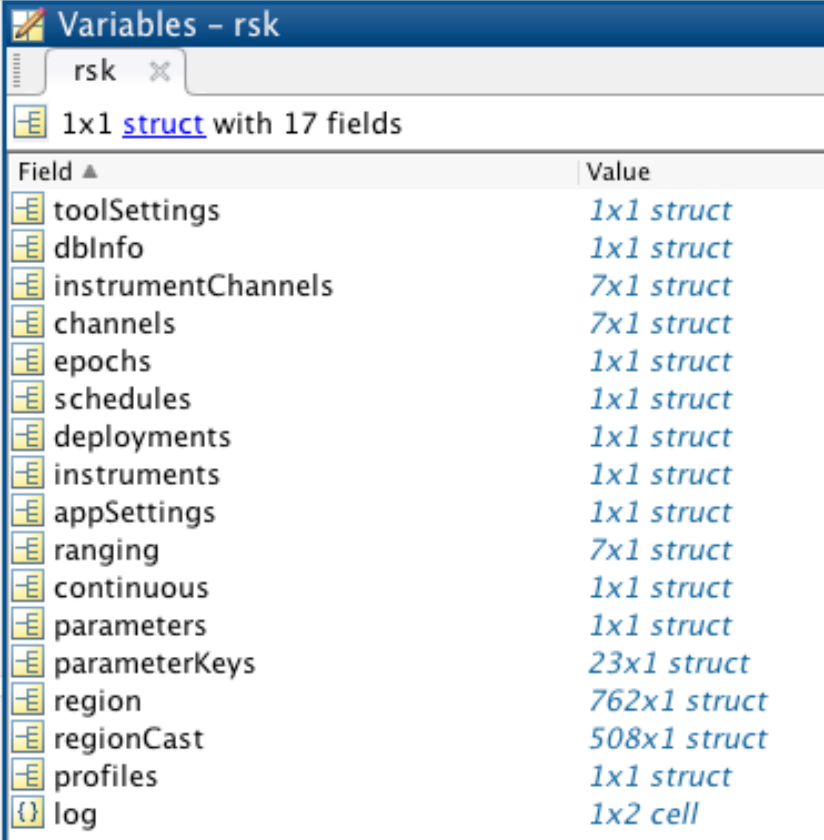
- Uses profile “events” from Ruskin to read the data and store as profiles

RBR

RSKopen

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

- *dbInfo*
 - version, format, file and path
- *channels*
 - channel name, unit
- *continuous*
 - sampling rate
- *profiles*
 - profile start and end times
- *instruments*
 - instrument name
 - instrument serial number
- *log*
 - data processing log



The screenshot shows a MATLAB Variables window titled "Variables - rsk". It displays a table of fields for the variable 'rsk', which is a 1x1 struct with 17 fields. Each field is listed with its name and its corresponding value type.

Field	Value
toolSettings	1x1 struct
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
region	762x1 struct
regionCast	508x1 struct
profiles	1x1 struct
log	1x2 cell

RSKprintchannels

```
RSKprintchannels(rsk)
```

Display important information about the instrument and dataset

```
>> 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/m ² /s'
7	'Chlorophyll'	'µg/l'

RSKreaddata

```
rsk = RSKreaddata(rsk);
```

Or

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



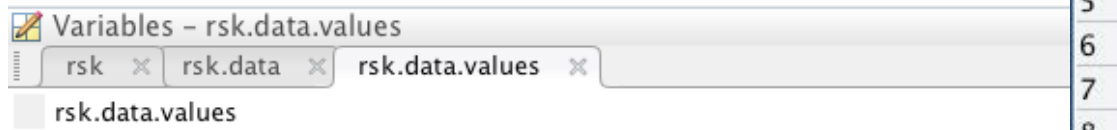
The screenshot shows the MATLAB Variables window for a workspace named 'rsk'. It contains two variables: 'rsk' and 'rsk.data'. The 'rsk.data' variable is expanded to show its structure. It has two fields: 'tstamp' and 'values'. The 'tstamp' field is a 1924x1 double array, and the 'values' field is a 1924x7 double array.

Field ▲	Value
tstamp	1924x1 double
values	1924x7 double

- Data subset specified by start time **t1** and end time **t2**.

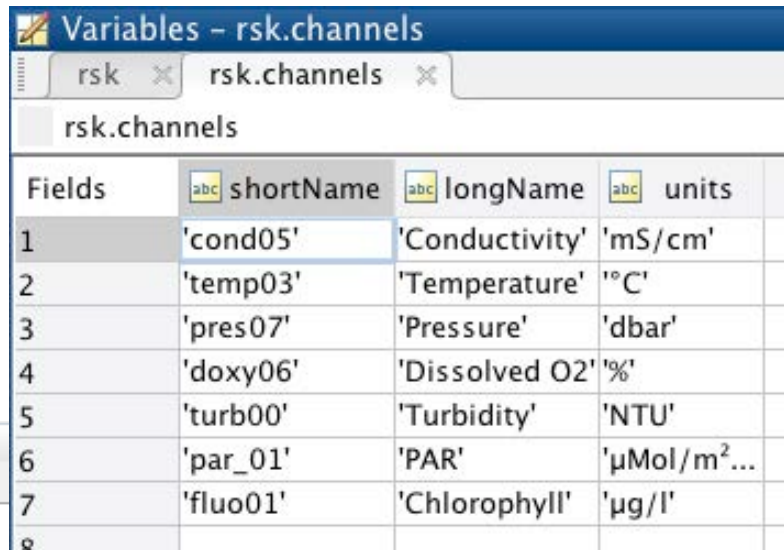
rsk.data

- Row: number of data samples
- Column: number of channels



Variables - rsk.data.values

	1	2	3	4	5	6	7
1	39.9973	16.2695	10.1034	51.1352	5.0452	4.6909	0.6533
2	39.9873	16.2648	10.1266	51.1329	4.5730	4.6886	1.9782
3	39.9887	16.2553	10.1247	51.1581	4.1794	4.6923	1.3185
4	39.9896	16.2555	10.1135	51.1587	4.1472	4.6913	1.6329
5	39.9860	16.2546	10.1078	51.1821	4.0244	4.6935	1.7083
6	39.9823	16.2579	10.1192	51.1562	3.9114	4.6904	1.5414
7	39.9732	16.2548	10.1165	51.1613	3.9796	4.6918	1.1302
8	39.9711	16.2321	10.1238	51.1861	4.3580	4.6917	0.8489
9	39.9746	16.2248	10.1254	51.1920	4.7382	4.6917	1.2072
10	39.9674	16.2322	10.1021	51.1833	4.4421	4.6943	1.7820
11	39.9550	16.2309	10.1107	51.1937	4.3361	4.6928	1.9826



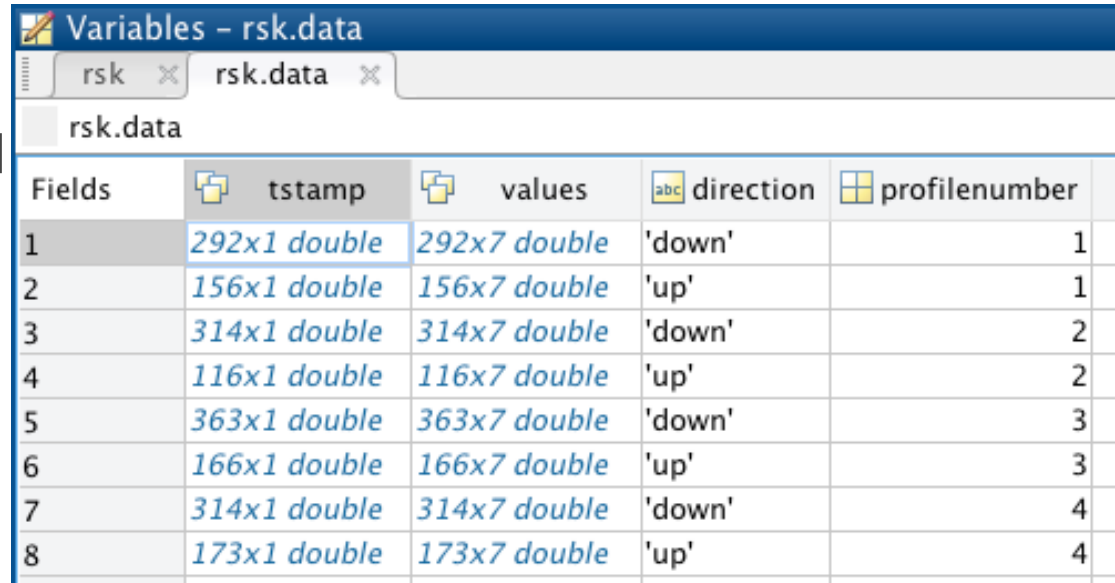
Variables - rsk.channels

Fields	shortName	longName	units
1	'cond05'	'Conductivity'	'mS/cm'
2	'temp03'	'Temperature'	'°C'
3	'pres07'	'Pressure'	'dbar'
4	'doxy06'	'Dissolved O2'	'%'
5	'turb00'	'Turbidity'	'NTU'
6	'par_01'	'PAR'	'µMol/m²...'
7	'fluo01'	'Chlorophyll'	'µg/l'
8			

RSKreadprofiles

```
rsk = RSKreadprofiles(rsk, 'profile', [1:4], 'direction', 'both');
```

- Either logger or Ruskin detects profiles and stores start and end time of each cast
- RSKreadprofiles uses that information to directly read data into profiles



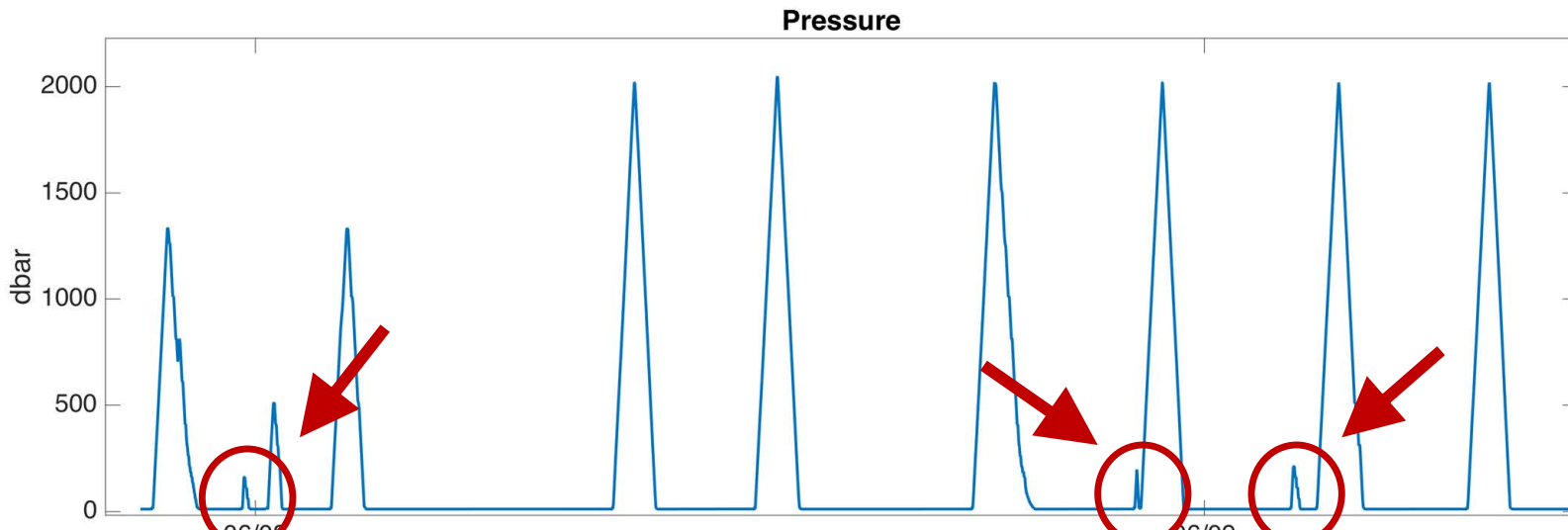
The screenshot shows the MATLAB Variables window for 'rsk.data'. It displays a table with 8 rows and 5 columns: Fields, tstamp, values, direction, and profilenumber. Each row represents a profile with its corresponding dimensions and direction.

Fields	tstamp	values	direction	profilenumber
1	292x1 double	292x7 double	'down'	1
2	156x1 double	156x7 double	'up'	1
3	314x1 double	314x7 double	'down'	2
4	116x1 double	116x7 double	'up'	2
5	363x1 double	363x7 double	'down'	3
6	166x1 double	166x7 double	'up'	3
7	314x1 double	314x7 double	'down'	4
8	173x1 double	173x7 double	'up'	4

RSKtimeseries2profiles

```
rsk = RSKtimeseries2profiles(rsk, 'pressureThreshold', 200);
```

- Reorganize time series created by RSKreaddata into profiles
- Useful when profile events were not correctly detected by logger/Ruskin
- Cast detection parameters can be tuned to for specific data sets



RSKaddstationdata

```
stations = {'S1', 'S2', 'S3', 'S4'};
rsk = RSKaddstationdata(rsk, 'cruise', {'Repeat
line'}, 'Station', stations, 'latitude', lat_vec, 'longitude', lon_ve
c);
```

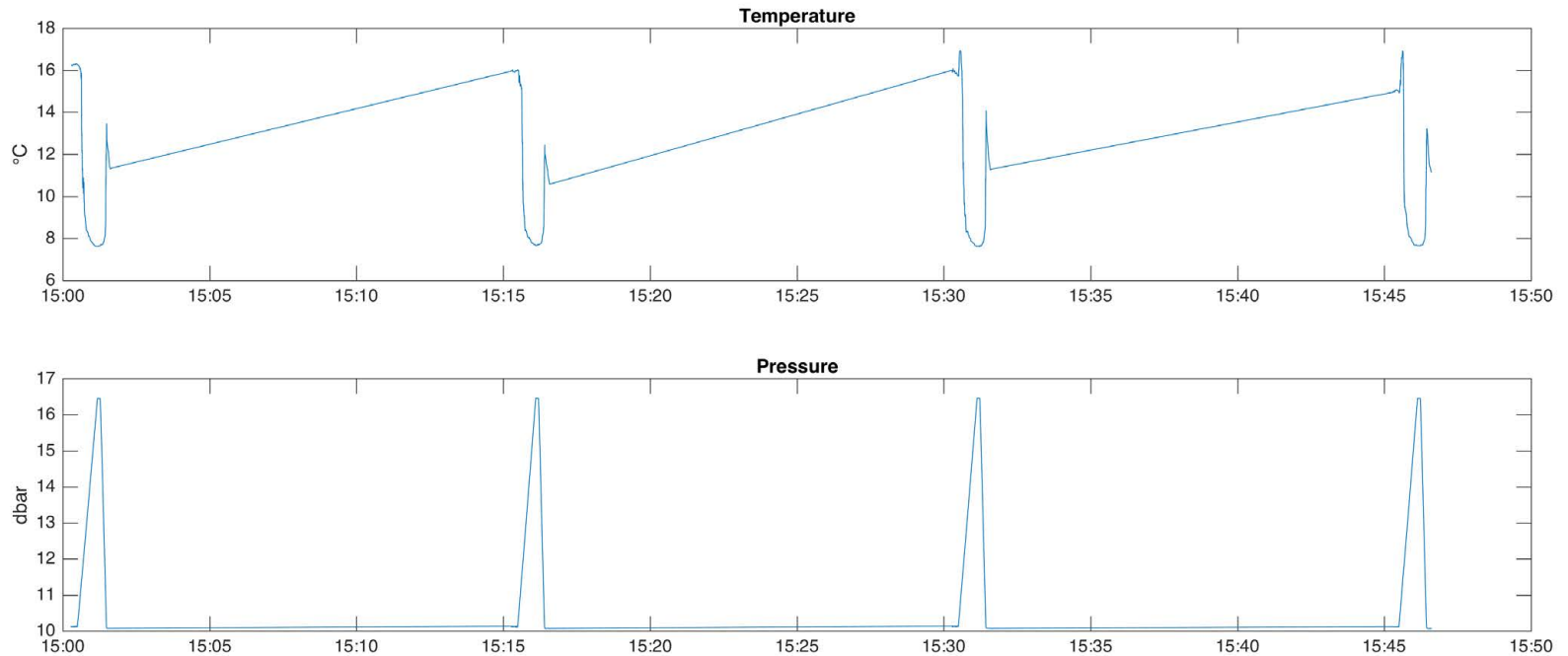
- Note this data is exported into the header by RSK2CSV.

```
>> disp(rsk.data(1))
    tstamp: [21x1 double]
    values: [21x12 double]
    direction: 'up'
    profilenumber: 1
    latitude: 45
    longitude: -25
    station: {'SK1'}
    cruise: {'Skootamatta Lake 1'}
    vessel: {'R/V RBR'}
    comment: []
    samplesinbin: [21x1 double]
```

- **Read**
- **View**
- **Process**
- **Export**

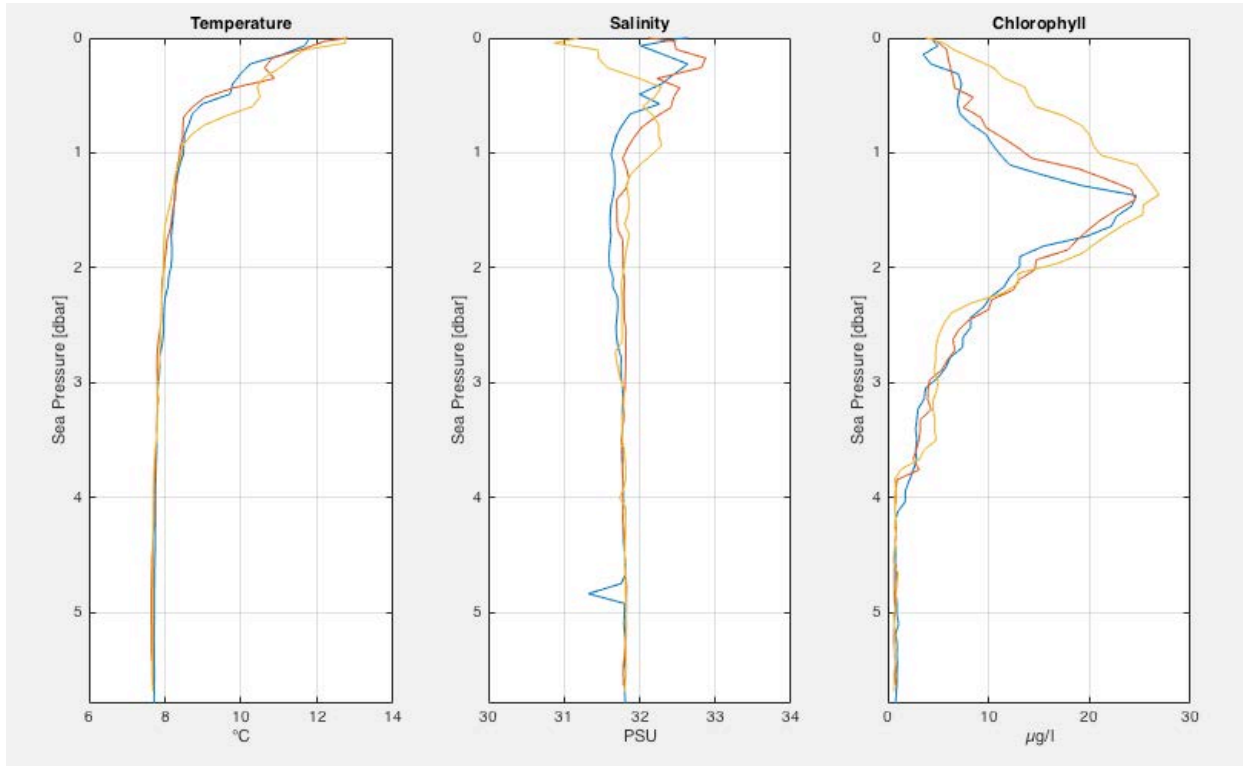
RSKplotdata

```
RSKplotdata(rsk, 'channel', {'temperature', 'pressure'});
```



RSKplotprofiles

```
RSKplotprofiles(rsk, 'channel', {'temperature', 'salinity',  
'chlorophyll'}, 'profile', [1 5 10], 'direction', 'down');
```

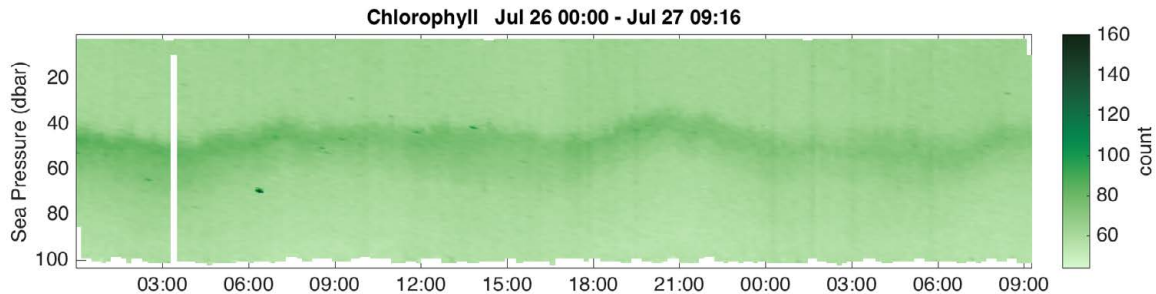
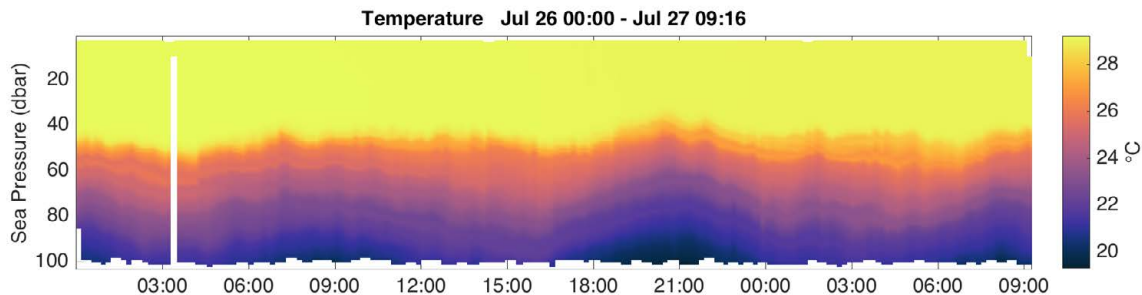
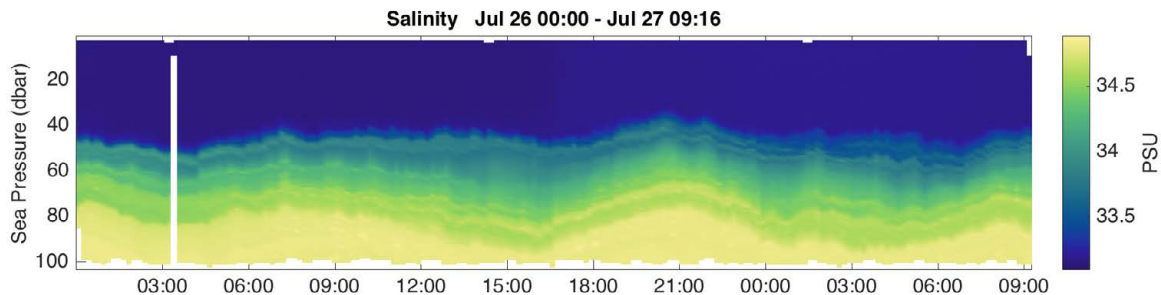


RBR

RSKimages

```
RSKimages(rsk,  
'direction', 'up')
```

- Requires bin averaging the profiles (RSKbinaverage)

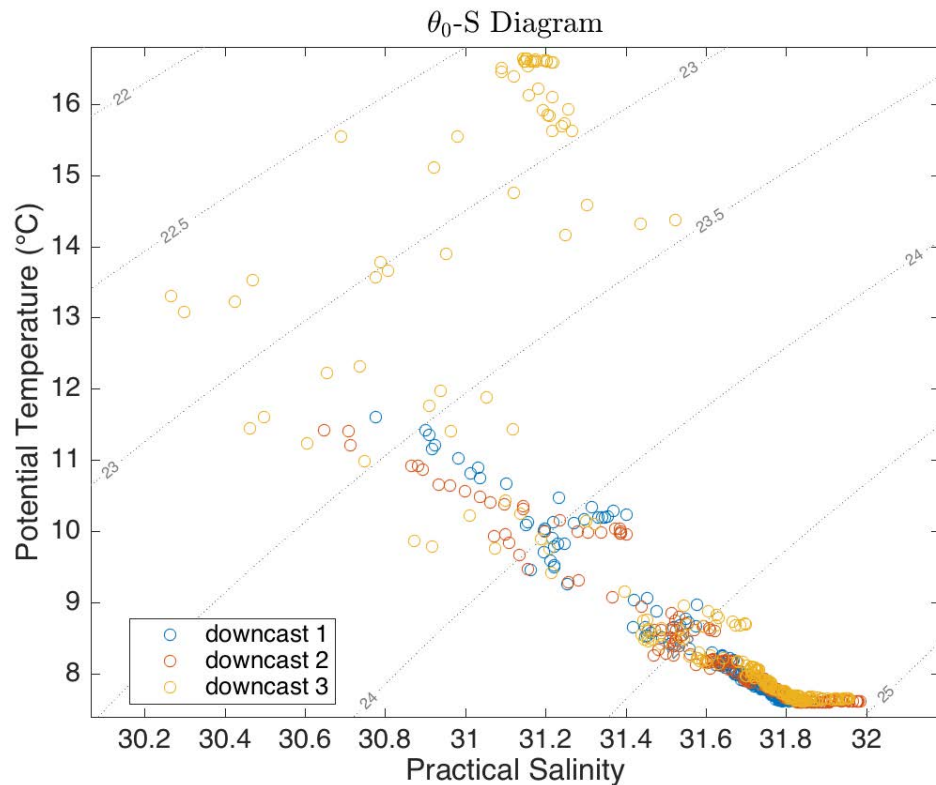


RSKplotTS

```
RSKplotTS(rsk, 'profile', 1:3, 'direction', 'down');
```

```
handles = RSKplotTS(rsk,...);
```

- outputs line handles to for customization

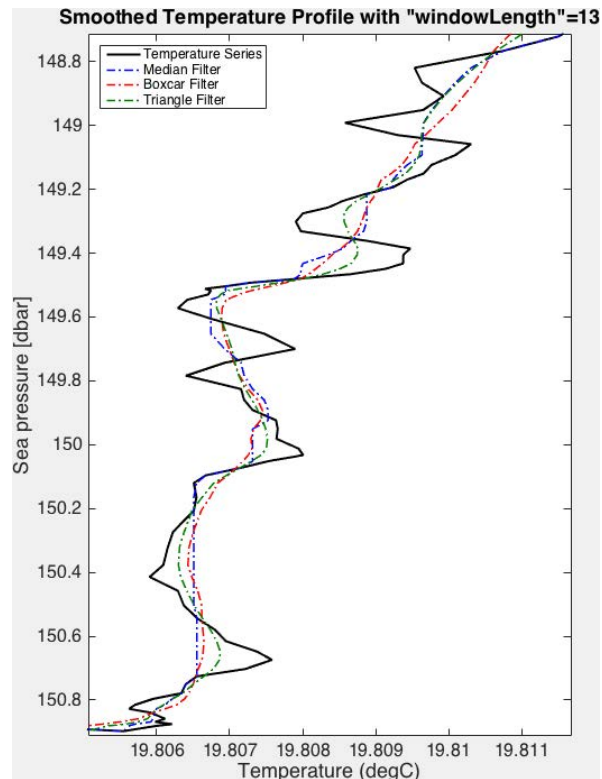


- **Read**
- **View**
- **Process**
- **Export**

RSKsmooth

```
rsk = RSKsmooth(rsk, 'temperature', 'windowLength', 5,  
'filter', 'boxcar', 'profile', 1, 'direction', 'down');
```

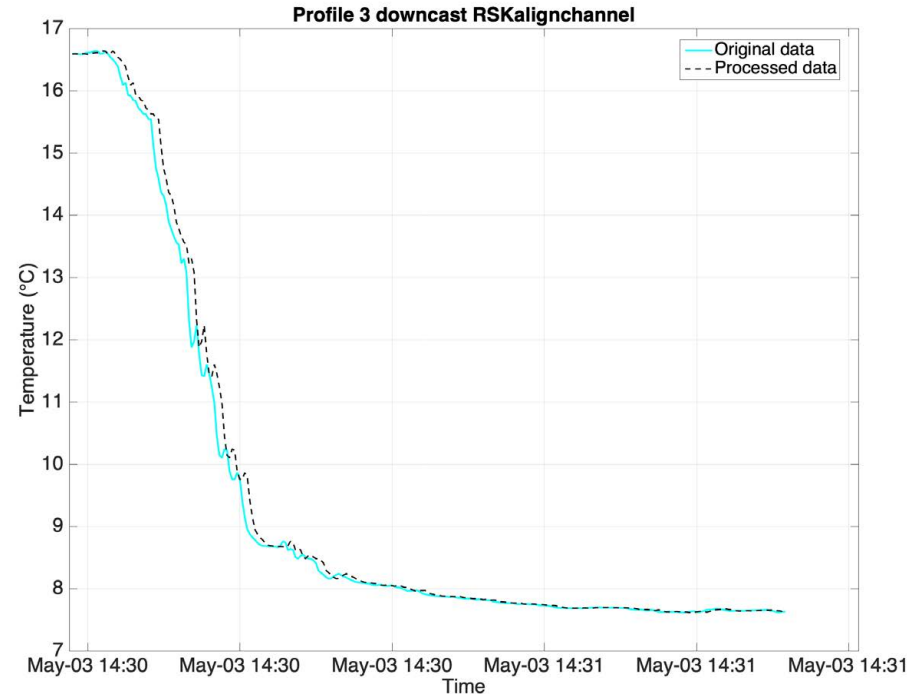
- **Low-pass filter** data with a running average or median
- Boxcar or triangular weighting windows
- User specified window length, profile number and cast direction



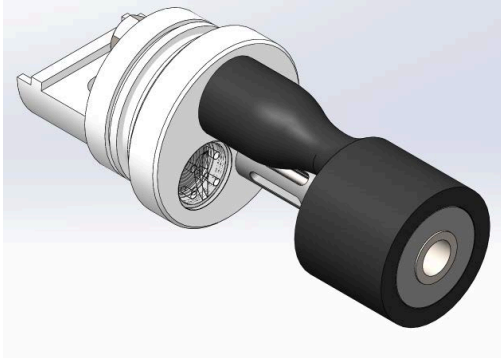
RSKalignchannel

```
rsk = RSKalignchannel(rsk, 'channel', 'temperature', 'lag', 2);
```

- Shifts a channel in time
- Can specify lag as a number of samples or a time in seconds (requires interpolation)



Time misalignment of conductivity and temperature



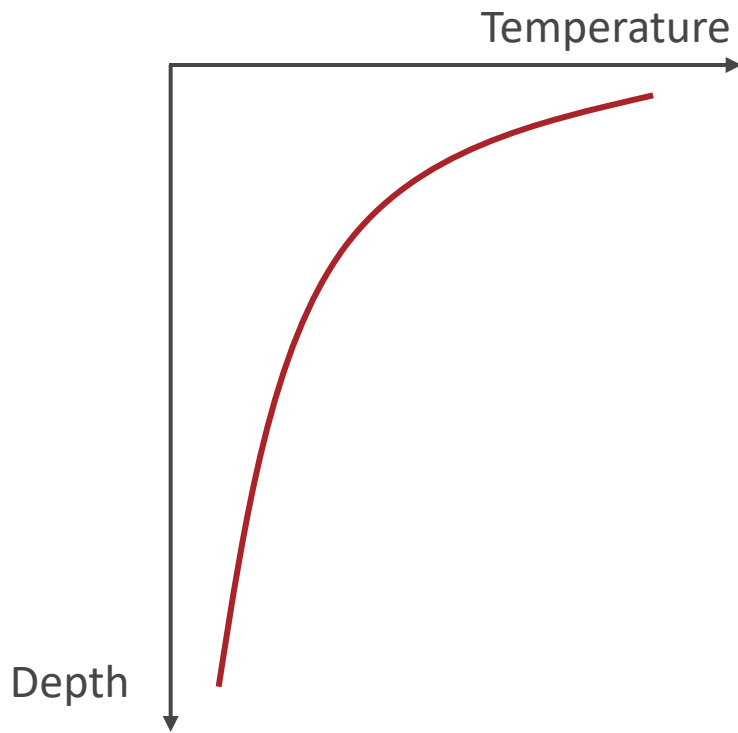
- In some instruments, conductivity and temperature are separated vertically in space
- Sensors may have different time constants
 - Both factors cause errors in derived salinity (e.g., spikes)
- For details on dynamic corrections for RBR CTDs, please see: [Webinar recording](#) and [PDF slides](#)



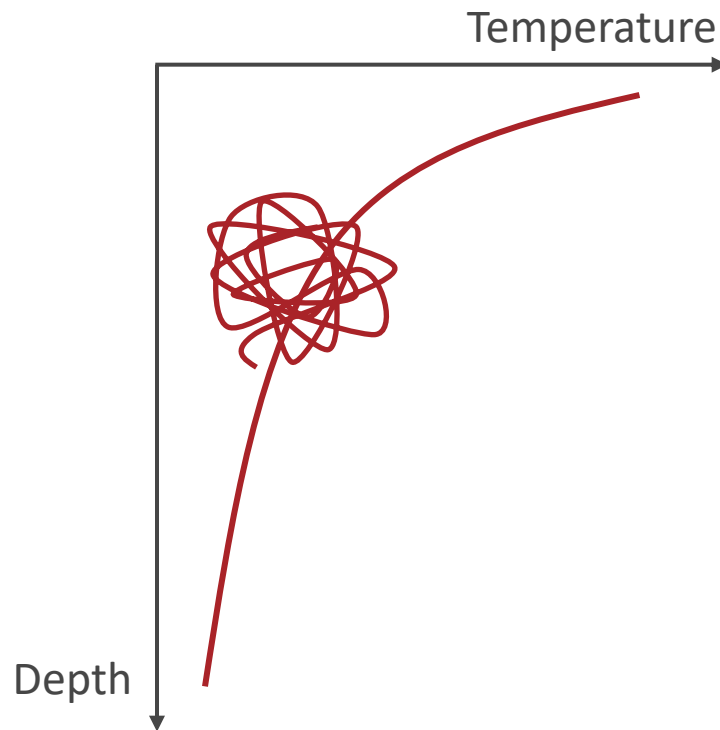
RBR

RSKremoveloops – improve data affected by ship heave

Ideal

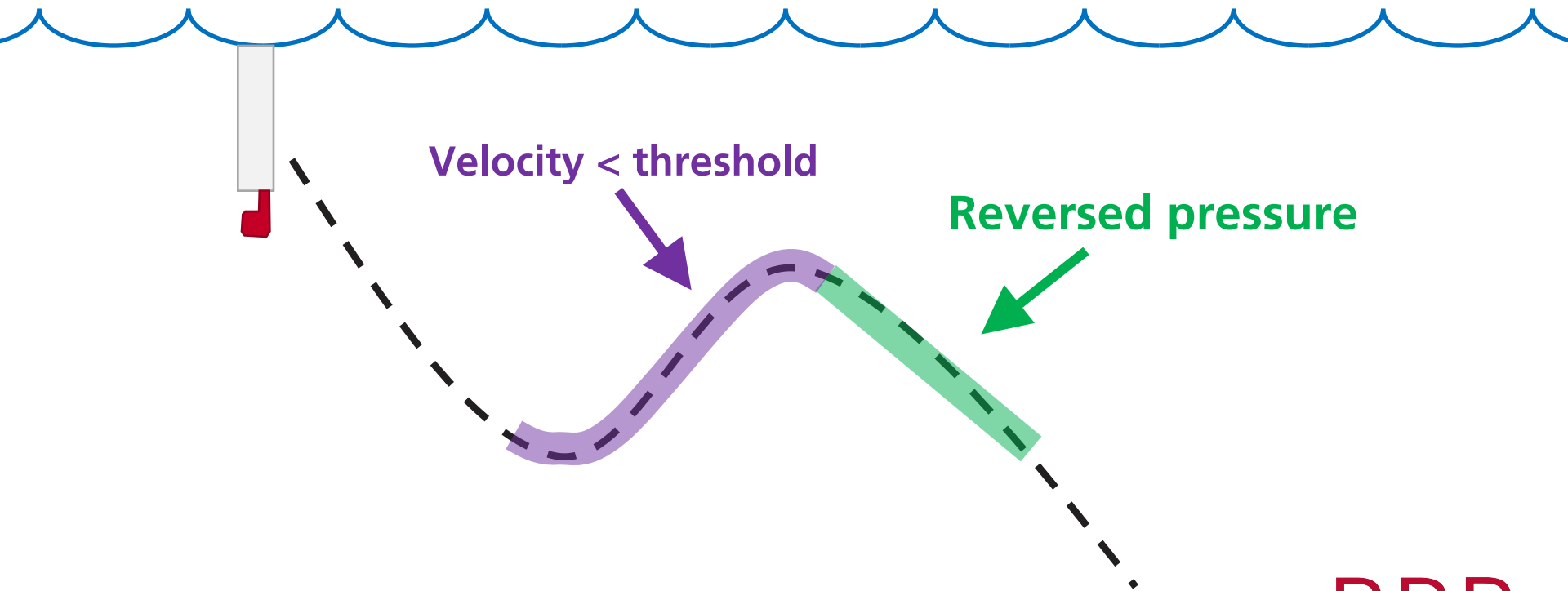


Reality



RBR

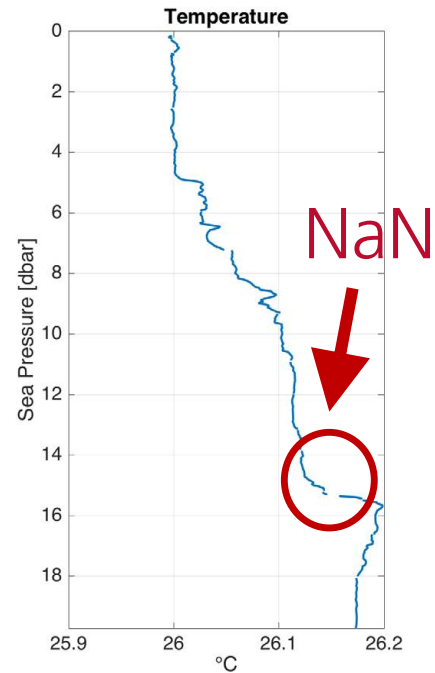
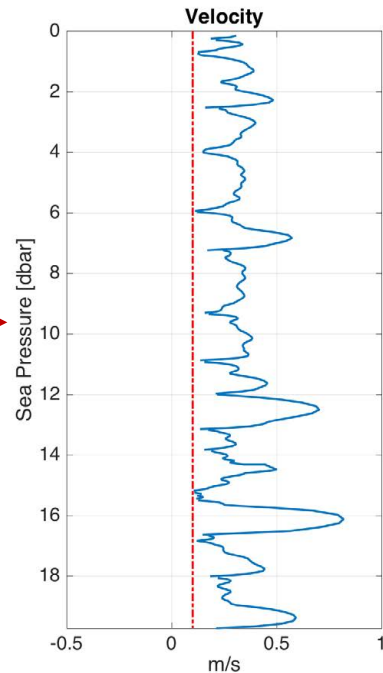
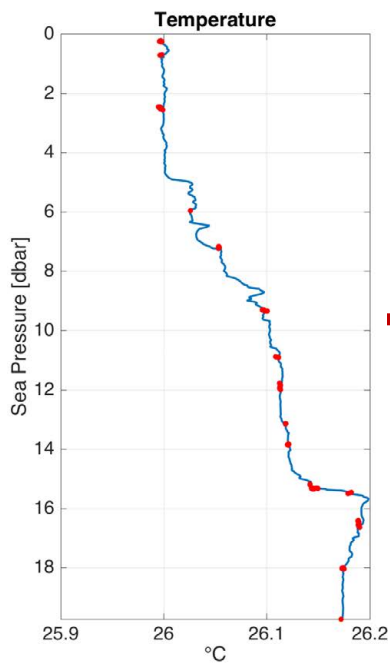
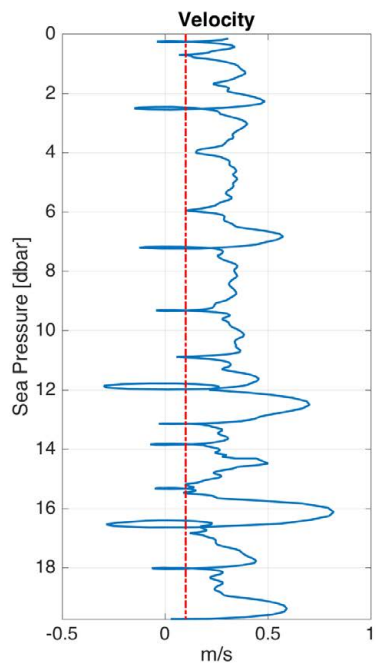
RSKremoveloops



RBR

RSKremoveloops

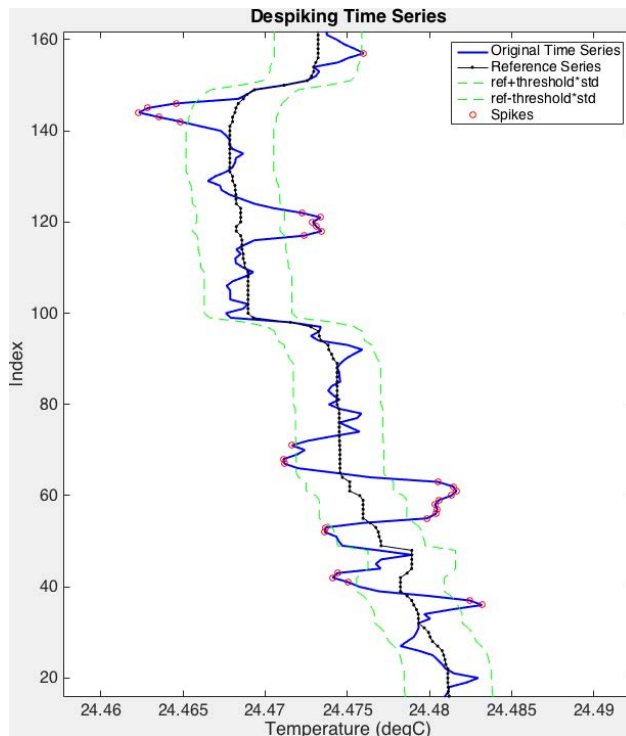
```
rsk = RSKremoveloops(rsk, 'threshold', 0.1);
```



RSKdespike

```
rsk = RSKdespike(rsk, 'temperature', 'threshold', 4,  
'windowLength', 11, 'action', 'nan');
```

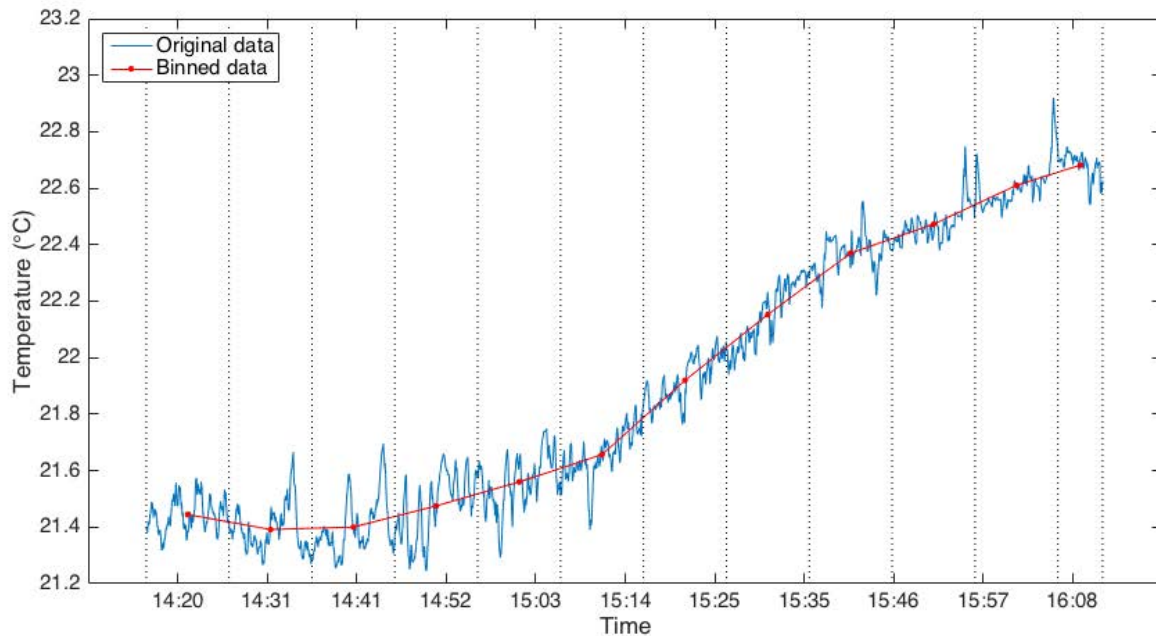
- Low-pass filter data to calculate a **reference time series** and **standard deviation** of the residuals.
- Data lying outside of 'threshold' times of standard deviations are defined as spikes.
- Three methods to deal with spikes
 - NaN
 - replace
 - interp



RSKbinaverage

```
rsk = RSKbinaverage(rsk, 'binBy', 'time', 'binSize', 600);
```

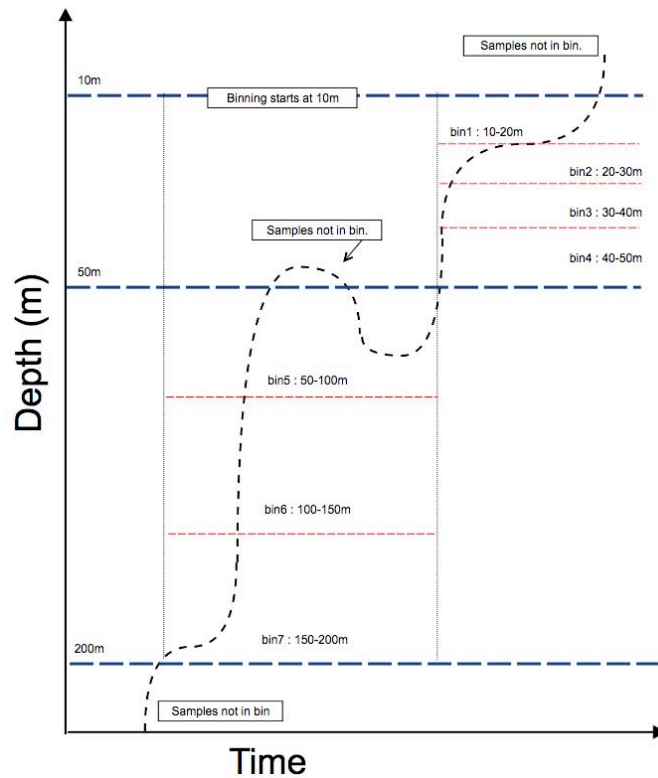
- Bin - average data within an interval
- Bin by sea pressure, depth, or time



RSKbinaverage

```
rsk = RSKbinaverage(rsk, 'direction', 'down', 'binBy', 'depth',  
'binSize', 1, 'boundary', 1);
```

- RSKbinaverage supports regime binning (variable bin widths)
- Bin average according to pressure, depth, time, or any channel

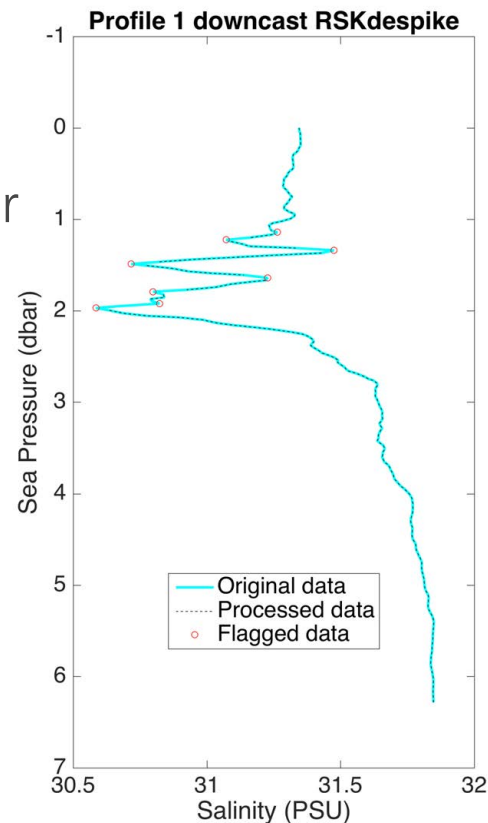


Visualization mode for post-process functions

```
rsk = RSKdespike(rsk, 'salinity',  
'threshold', 1, 'visualize', 1);
```

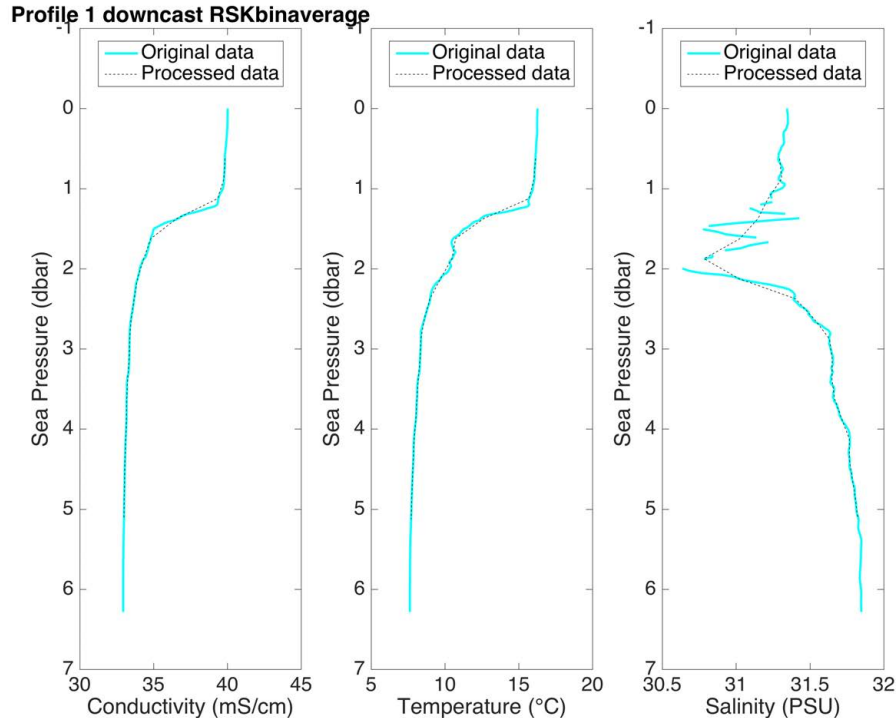
- Optional input 'visualize' paired with profile number
- Display original, processed and flagged data.

Profile number to visualize



Visualization mode for post-process functions

```
rsk = RSKbinaverage(rsk, 'binBy', 'sea pressure',  
'direction', 'down', 'visualize', 1);
```



RBR

RSKderive

- sea pressure
- profiling velocity
- Sound speed
- Specific conductivity
- O2 concentration
- O2 saturation

TEOS-10 wrappers:

- depth
- Practical and Absolute Salinity
- Potential temperature and density
- buoyancy frequency
- ...

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

Fields	shortName	longName	units
1	'cond05'	'Conductivity'	'mS/cm'
2	'temp03'	'Temperature'	'°C'
3	'pres07'	'Pressure'	'dbar'
4	'doxy06'	'Dissolved O2'	'%'
5	'turb00'	'Turbidity'	'NTU'
6	'par_01'	'PAR'	'μMol/m ² /s'
7	'fluo01'	'Chlorophyll'	'μg/l'
8	'pres08'	'Sea Pressure'	'dbar'
9	'sal_00'	'Salinity'	'PSU'

RBR

Functions: Advanced post-processing

- RSKcorrecttau
- RSKcorrectTM



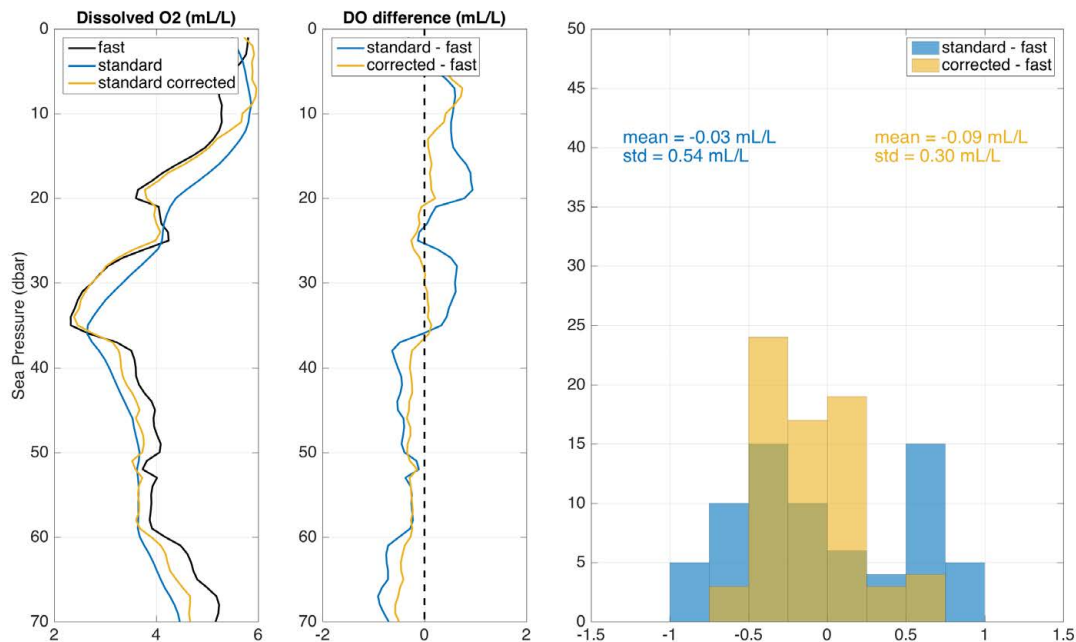
RSKcorrecttau

- Apply **Fozdar et al. (1985)** algorithm to correct the phase and amplitude response of measured signals.
- Known as signal reconstruction or “sharpening”
- Suitable for sensors with a relatively large time constant
 - Profiling with a dissolved oxygen sensor with $\tau = 8$ s
 - And/or fast profiling through **large gradients**

RSKcorrecttau

```
rsk = RSKcorrecttau(rsk, 'channel', 'Dissolved O2', 'tauResponse', 8)
```

- Application on RBRcoda T.ODO ($\tau = 8s$) in a profiling practice
- T.ODO|fast ($\tau = 1s$) serve as a reference



RSKcorrectTM

```
rsk = RSKcorrecttau(rsk, 'alpha', 0.08, 'beta', 1/8);
```

- Applies the Lueck and Picklo (1990) algorithm to correct measured conductivity for thermal inertia

$$C_T(n) = -bC_T(n-1) + \gamma a [T(n) - T(n-1)]$$

$$a = 4f_N \alpha \beta^{-1} (1 + 4f_N \beta^{-1})^{-1}$$

$$b = 1 - 2a\alpha^{-1}$$

$$C_{cor}(n) = C(n) + C_T(n)$$

Lueck, R. G. and J. J. Picklo, 1990: Thermal inertia of conductivity cells: Observations with a Sea-Bird cell. J. Atmos. Oceanic Technol., 7, pp. 756 - 768.
[https://doi.org/10.1175/1520-0426\(1990\)007<0756:TIOCCO>2.0.CO;2](https://doi.org/10.1175/1520-0426(1990)007<0756:TIOCCO>2.0.CO;2)

RBR

- **Read**
- **View**
- **Process**
- **Export**

RSK2CSV

```
RSK2CSV(rsk, 'channel', { 'conductivity', 'pressure', 'dissolved  
O2' }, 'outputdir', '/Users/folder', 'comment', 'Hey Jude');
```

```
080217_20150919_1417_profile0003.csv — Edited  
//Creator: RBR Ltd  
//Create Time: 05-Sep-2018 14:11:33  
//Instrument model firmware and serialID: RBRmaestro 12.03 80217  
//Sample period: 0.167 second  
//Processing history:  
///Users/RZhang/code/rsk_files/080217_20150919_1417.rsk opened using RSKtools v2.3.1.  
//Sea pressure calculated using an atmospheric pressure of 10.1325 dbar.  
//Cruise:  
//Station:  
//Latitude:  
//Longitude:  
//Depth:  
//Date:  
//Weather:  
//Crew:  
//Comment: Hey Jude  
  
//Time(yyyy-mm-dd HH:MM:ss.FFF), Conductivity(mS/cm), Pressure(dbar), Dissolved_O2(%), Cast_direction  
2015-09-19 08:59:05.000, 34.2349, 79.0907, 472.6810, d  
2015-09-19 08:59:05.167, 34.2363, 78.8998, 472.5748, d  
2015-09-19 08:59:05.333, 34.2414, 78.7738, 472.5124, d  
2015-09-19 08:59:05.500, 34.2504, 78.6109, 472.4069, d  
2015-09-19 08:59:05.667, 34.2559, 78.4469, 472.3188, d  
2015-09-19 08:59:05.833, 34.2618, 78.2888, 472.1892, d  
2015-09-19 08:59:06.000, 34.2679, 78.1382, 472.1196, d
```

RBR

RSK2ODV

RSK2ODV (rsk)

```
065679_20160517_1657_profile0002.txt — Edited
//<CreateTime>09-Sep-2018 13:04:12</CreateTime>|
//<Software>RSKtools</Software>
//<Source></Source>
//<SourceLast-Modified></SourceLast-Modified>
//<Version>ODV Spreadsheet V4.0</Version>
//<DataField>Ocean</DataField>
//<DataType>Profile</DataType>
//<DataVariable>label="Cast_direction" value_type="TEXT" is_primary_variable="F" comment="d-downcast u-upcast"</DataVariable>
//<MissingDataValues>NaN</MissingDataValues>
// Model=RBRconcerto
// Firmware=12.02
// Serial=65679
//Processing history:
///Users/RZhang/code/rsk_files/065679_20160517_1657_rsk opened using RSKtools v2.3.1.
//Sea pressure calculated using an atmospheric pressure of 10.1325 dbar.
```

Cruise	Station	Type	yyyy-mm-ddTHH:MM:ss.FFF	Longitude[degrees_east]	Latitude [degrees_north]	Bot. Depth [m]	Sea_Pressure[dbar]	Conductivity[mS/cm]	Temperature[°C]	Pressure[dbar]	Dissolved_O2[%]	Chlorophyll_a[µg/l]	Cast_direction
C1	S1	C	2016-05-17T07:51:21.000	0.00 0.00 0.0	357.6316 33.9095	9.1915	367.7641	97.9795	0.7727	u			
C1	S1	C	2016-05-17T07:51:21.167	0.00 0.00 0.0	357.4582 33.9090	9.1916	367.5907	98.0505	0.7727	u			
C1	S1	C	2016-05-17T07:51:21.333	0.00 0.00 0.0	357.2824 33.9089	9.1917	367.4149	97.9677	0.7725	u			
C1	S1	C	2016-05-17T07:51:21.500	0.00 0.00 0.0	357.0989 33.9091	9.1919	367.2314	97.9490	0.7726	u			
C1	S1	C	2016-05-17T07:51:21.667	0.00 0.00 0.0	356.9295 33.9088	9.1917	367.0620	98.0126	0.7733	u			
C1	S1	C	2016-05-17T07:51:21.833	0.00 0.00 0.0	356.7532 33.9088	9.1914	366.8857	97.9854	0.7754	u			
C1	S1	C	2016-05-17T07:51:22.000	0.00 0.00 0.0	356.5741 33.9086	9.1915	366.7066	97.9857	0.7771	u			
C1	S1	C	2016-05-17T07:51:22.167	0.00 0.00 0.0	356.3943 33.9071	9.1915	366.5268	97.9761	0.7785	u			
C1	S1	C	2016-05-17T07:51:22.333	0.00 0.00 0.0	356.2234 33.9075	9.1913	366.3559	97.9596	0.7810	u			
C1	S1	C	2016-05-17T07:51:22.500	0.00 0.00 0.0	356.0466 33.9076	9.1913	366.1791	97.9900	0.7784	u			
C1	S1	C	2016-05-17T07:51:22.667	0.00 0.00 0.0	355.8685 33.9079	9.1914	366.0010	97.9771	0.7751	u			
C1	S1	C	2016-05-17T07:51:22.833	0.00 0.00 0.0	355.6923 33.9075	9.1915	365.8248	97.9838	0.7728	u			
C1	S1	C	2016-05-17T07:51:23.000	0.00 0.00 0.0	355.5148 33.9076	9.1912	365.6473	98.0044	0.7718	u			

RBR

RSK2RSK

```
newfile = RSK2RSK(rsk)
```

- Write rsk file using simplest schema
- Write post-processed RBR data in an RSK file
- Readable with Ruskin
- Independent from original RSK file

Name	
▼	Tables (15)
▶	channels
▶	data
▶	dbInfo
▶	deployments
▶	downloads
▶	epochs
▶	errors
▶	events
▶	instruments
▶	region
▶	regionCast
▶	regionComment
▶	regionGeoData
▶	regionProfile
▶	schedules

RBR

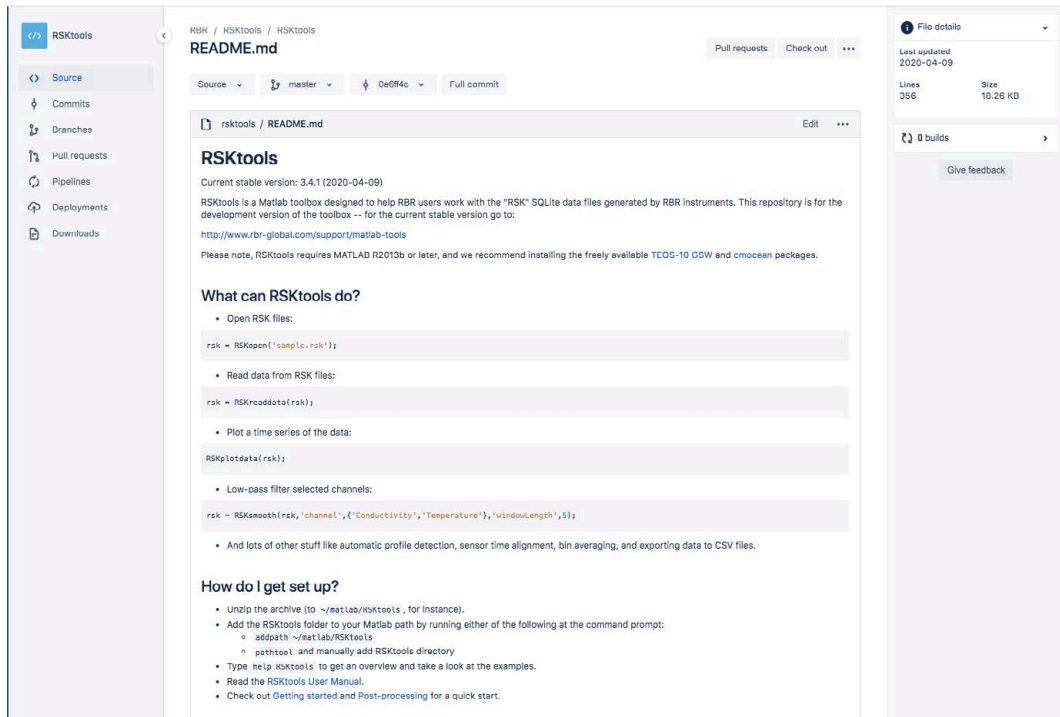
RSKcreate – convert ANY data into the rsk structure

- Any data source: other CTDs, float, gilder, etc
- Legacy RBR instruments
- Apply RSKtools function to any dataset!

```
tstamp = [735722.625196759;  
          735722.625198692;  
          735722.625200613];  
values = [39.9973,    16.2695,    10.1034;  
          39.9873,    16.2648,    10.1266;  
          39.9887,    16.2553,    10.1247];  
channel = {'Conductivity', 'Temperature', 'Pressure'};  
unit = {'mS/cm', '°C', 'dbar'};  
rsk = RSKcreate('tstamp', tstamp, 'values', values,  
               'channel', channel, 'unit', unit);
```


Bitbucket repository

- Stay on top of the most recent developments
- Anyone can contribute



The screenshot displays the Bitbucket web interface for the RSKtools repository. The left sidebar shows navigation options: Source, Commits, Branches, Pull requests, Pipelines, Deployments, and Downloads. The main content area shows the README.md file for the 'master' branch. The README includes the following information:

- RSKtools**
Current stable version: 3.4.1 (2020-04-09)
RSKtools is a Matlab toolbox designed to help RBR users work with the "RSK" SQLite data files generated by RBR instruments. This repository is for the development version of the toolbox -- for the current stable version go to:
<http://www.rbr-global.com/support/matlab-tools>
Please note, RSKtools requires MATLAB R2013b or later, and we recommend installing the freely available TCOS-10 GDW and cmoocean packages.
- What can RSKtools do?**
 - Open RSK files:

```
rsk = RSKopen('sample.rsk');
```
 - Read data from RSK files:

```
rsk = RSKreaddata(rsk);
```
 - Plot a time series of the data:

```
RSKplotdata(rsk);
```
 - Low-pass filter selected channels:

```
rsk = RSKsmooth(rsk, 'channel', {'Conductivity', 'Temperature'}, 'windowLength', 5);
```
 - And lots of other stuff like automatic profile detection, sensor time alignment, bin averaging, and exporting data to CSV files.
- How do I get set up?**
 - Unzip the archive (to ~/matlab/RSKtools, for instance).
 - Add the RSKtools folder to your Matlab path by running either of the following at the command prompt:
 - `addpath ~/matlab/RSKtools`
 - `pathtool` and manually add RSKtools directory
 - Type `help rsktools` to get an overview and take a look at the examples.
 - Read the RSKtools User Manual.
 - Check out [Getting started and Post-processing](#) for a quick start.

<https://bitbucket.org/rbr/rsktools/src/master/>

RBR

Online manuals - <https://docs.rbr-global.com/rsktools>

The screenshot shows a web browser window with the following details:

- Browser Tab:** RSKtools - RSKtools
- Address Bar:** Secure | <https://docs.rbr-global.com/rsktools>
- Navigation:** Back, Forward, Refresh buttons.
- Bookmarks:** General, RSKtools, Other, Google Maps.
- Page Content:**
 - Left Sidebar:**
 - RBR
 - RSKtools
 - > Introduction
 - > Standard Functions
 - > RSKopen.m
 - RSKreaddata.m
 - RSKreadprofiles.m
 - RSKfindprofiles.m
 - RSKreadevents.m
 - RSKreadburstdata.m
 - RSKtrim.m
 - > Post-processing
 - RSKderiveseapressure.m
 - RSKderiveddepth.m
 - RSKderivesalinity.m
 - RSKderivevelocity.m
 - RSKderiveBPR.m
 - RSKderiveCGF.m
 - Main Content Area:**
 - RSKtools
 - RSKtools
 - Introduction
 - RSKtools is a collection of Matlab m-files designed to open, read, plot, and process data from RBR loggers.
 - RSKtools organises logger data in a Matlab structure, referred to as the RSK structure, which contains the logger and deployment metadata. The RSK structure supports two data formats: a continuous time series, or a collection of profiles. After loading the data into Matlab, it is easy to plot and process the data using default input arguments. All optional input arguments are name-value pair arguments which make the functions customisable and flexible.
 - Installing
 - RSKtools requires MATLAB R2013b or later.
 - We suggest downloading the [Gibbs-SeaWater Oceanographic Matlab Toolbox](#) and the oceanographic colourmap "[cmocean](#)" toolboxes.
 - What is in this document?
 - Introduction
 - Document Formatting
 - Release Notes
 - Standard Functions
 - RSKopen.m
 - RSKgetprofiles.m
 - RSKreaddownsample.m
 - RSKreadgeodata.m
 - RSKreaddata.m
 - RSKreadprofiles.m
 - RSKfindprofiles.m

Help from the Matlab command window

help rsktools

```
Command Window
>> help rsktools
rsktools
Version 3.0.0 2018-11-14

1. This toolbox depends on the presence of a functional mksqlite
library. We have included a couple of versions here for Windows (32 bit/
64 bit), Linux (64 bit) and Mac (64 bit). If you might need to compile
another version, the source code can be downloaded from
https://sourceforge.net/projects/mksqlite/files/. RSKtools currently uses
mksqlite Version 2.5.

2. Opening an RSK file. Use RSKopen with a filename as the argument:

RSK = RSKopen('sample.rsk');

This generates an RSK structure with all the metadata from the database,
and a downsampled version of the data. The downsampled version is useful
for generating figures of very large data sets.

3. Use RSKreaddata to read data from the RSK file:

RSK = RSKreaddata(RSK, 't1', <starttime>, 't2', <endtime>);

This reads a portion of the 'data' table into the RSK structure
(replacing any previous data that was read this way). The <starttime>
and <endtime> values are the range of data to be read. Depending on the
amount of data in your dataset, and the amount of memory in your
computer, you can read bigger or smaller chunks before Matlab will
run out of memory. The times are specified using the Matlab 'datetime'
format. You will find the start and end times of the deployment useful
reference points - these are contained in the RSK structure as the
RSK.epochs.starttime and RSK.epochs.endtime fields.

4. Plot the data!

RSKplotdata(RSK)
```

help RSKdespike

```
Command Window
>> help RSKdespike
RSKdespike - Despike a time series.

Syntax: [RSK, spike] = RSKdespike(RSK, channel, [OPTIONS])

Identifies and treats spikes using a median filtering algorithm. A
reference time series is created by filtering the input channel with
a median filter of length 'windowLength'. A residual ("high-pass")
series is formed by subtracting the reference series from the
original signal. Data in the reference series lying outside of
'threshold' standard deviations are defined as spikes. Spikes are
then treated by one of three methods (see below).

Inputs:
[Required] - RSK - Structure containing logger data.

channel - Longname of channel to despike (e.g.,
temperature, salinity, etc)

[Optional] - profile - Profile number. Default is all available
profiles.

direction - 'up' for upcast, 'down' for downcast, or
'both' for all. Default is all directions available.

threshold - Amount of standard deviations to use for the
spike criterion. Default value is 2.

windowLength - Total size of the filter window. Must be
odd. Default is 3.

action - Action to perform on a spike. The default is
'nan', whereby spikes are replaced with NaN. Other
```



Thank You!

Contact Us

rbr-global.com

info@rbr-global.com

+1 613 599 8900

RBR