The Journal of Open Source Software

argodata: An R interface to oceanographic data from the International Argo Program

Dewey Dunnington[*1], **Jaimie Harbin**[1], **Dan E. Kelley**[2], **and Clark Richards**[1]

1 Fisheries and Oceans Canada, Bedford Institute of Oceanography, Dartmouth, NS, Canada **2** Department of Oceanography, Dalhousie University, Halifax, NS, Canada

DOI: 10.21105/joss.03305

Software

- Review
- Repository
- Archive

Editor: Pending Editor

Submitted: 21 May 2021
Published: 21 May 2021

Summary

This paper describes `argodata`, an R package that makes it easier to work with data acquired in the International Argo Program, which provides over two decades of oceanographic measurements from around the world. Although Argo data are publicly available in NetCDF format and several software packages are available to assist in locating and downloading relevant Argo data, the multidimensional arrays used can be difficult to understand for non-oceanographers. Given the increasing use of Argo data in other disciplines, we built a minimal interface to the data set that uses the data frame as the primary data structure. This approach allows users to leverage the rich ecosystem of R packages that manipulate data frames (e.g., the `tidyverse`) and associated instructional resources.

Introduction

The ocean is highly variable in both space and time and mapping this variability at appropriate scales is a key factor in many scientific studies. Oceanographic data have direct applications that range from the analysis of near-bottom ecosystems to air-sea interactions. More broadly, ocean measurements are needed to constrain the models that scientists use to understand the evolving state of the ocean and to make predictions about its future, particularly as a component of the global climate system.

The International Argo Program deploys and collects data from several thousand devices that are programmed to drift with and move vertically through the ocean. Sensors measure electrical conductivity, temperature, pressure, and sometimes other quantities along this vertical path yielding "profiles" that are uploaded via satellite to globally distributed data assembly centres (Roemmich et al., 2001, 2009). Since 1997, the International Argo Program has collected over 2.4 million profiles from around the globe.

Although the NetCDF data files provided by Argo data servers contain metadata that describe their contents, we identified a number of barriers to data access. These included (1) reading and decoding the index files to locate files of interest, (2) downloading and potentially caching large numbers of small NetCDF files, (3) reading the NetCDF files into a form where the data contained within can be visualized and analyzed, and (4) dealing efficiently with potentially large Argo data sets. Whereas a variety of applications have been created to address some of these barriers, the `argodata` package is our attempt to overcome these barriers for the novice to average-level programmer who may not be familiar with oceanographic conventions for storing data.

*Corresponding author.

The Journal of Open Source Software

Statement of need

In the R language, several tools are available to access data from the International Argo Program. The `oce` package provides facilities to read and analyze "profile" and "trajectory" Argo NetCDF files (Kelley, 2018; Kelley & Richards, 2021); the `argoFloats` package provides additional tools to locate, download, cache, and visualize Argo NetCDF files (Kelley et al., 2021); and `rnoaa` provides limited access to a subset of Argo data from the North Atlantic (Chamberlain, 2021). Outside of R, the `argopy` package for Python provides access to the Argo data set with some facilities for analysis and visualization (Maze & Balem, 2020), and several web applications provide visual tools to locate relevant Argo profiles based on user-defined search criteria (*OceanOPS*, 2021; Tucker et al., 2020).

Several barriers we identified are not specific to the Argo data set and can be overcome with well-established R tools. To download and potentially cache Argo NetCDF files, at least one Argo mirror provides an `rsync` target for profile and index files. The `bowerbird` package provides similar facilities for downloading and caching large numbers of files from a remote source (Raymond & Sumner, 2021). To analyze and visualize potentially large data sets, `dplyr` and `ggplot2` within the wider `tidyverse` family of packages are well-established ("Welcome to the Tidyverse," 2019; Wickham, 2016; Wickham et al., 2021). To read NetCDF files in a form that can be analyzed and plotted using `dplyr` and `ggplot2`, respectively, the `tidync` and `ncmeta` packages introduce the concept of "grids" to identify groups of variables that can be loaded into a single data frame (Sumner, 2020a, 2020b).

The `argodata` package was designed to work with a range of tools that manipulate R data frames. In particular, the `tidyverse` family of packages has a large user base and has widely and freely available educational material in several languages (Wickham & Grolemund, 2017). Whereas previous packages for R and Python propagate the multidimensional array format of Argo NetCDF files when read, the ability to leverage the `tidyverse` depends on the representation of Argo data as data frames in "tidy" (one observation per row, one variable per column) format (Wickham, 2014), around which packages in the `tidyverse` are designed.

Using `argodata`

The `argodata` package is available as an R source package from GitHub (https://github.com/ArgoCanada/argodata), installable using the `remotes` package:

```
# install.packages("remotes")
remotes::install_github("ArgoCanada/argodata")
```

For our example usage, we also load the `tidyverse`:

```
library(tidyverse)
library(argodata)
```

To locate files of interest on the Argo mirror, index files for profile, trajectory, meta, and technical parameter files are provided in compressed CSV format. `argodata` uses the `vroom` package to efficiently load these files as they can be time-consuming to repeatedly read otherwise. The most commonly-used index is for profile files:

```
(prof <- argo_global_prof())
```

```
## Loading argo_global_prof()
```

```
## Downloading 1 file from 'https://data-argo.ifremer.fr'
```

```
## # A tibble: 2,455,058 x 8
##    file     date                latitude longitude ocean profiler_type
##    <chr>    <dttm>                 <dbl>     <dbl> <chr>          <dbl>
##  1 aoml/13~ 1997-07-29 20:03:00    0.267     -16.0 A                845
##  2 aoml/13~ 1997-08-09 19:21:12    0.072     -17.7 A                845
##  3 aoml/13~ 1997-08-20 18:45:45    0.543     -19.6 A                845
##  4 aoml/13~ 1997-08-31 19:39:05    1.26      -20.5 A                845
##  5 aoml/13~ 1997-09-11 18:58:08    0.72      -20.8 A                845
##  6 aoml/13~ 1997-09-22 19:57:02    1.76      -21.6 A                845
##  7 aoml/13~ 1997-10-03 19:15:49    2.60      -21.6 A                845
##  8 aoml/13~ 1997-10-14 18:39:35    1.76      -21.6 A                845
##  9 aoml/13~ 1997-10-25 19:32:34    1.80      -21.8 A                845
## 10 aoml/13~ 1997-11-05 18:51:42    1.64      -21.4 A                845
## # ... with 2,455,048 more rows, and 2 more variables:
## #   institution <chr>, date_update <dttm>
```

A typical analysis will focus on a subset of profiles. Users can subset this index using existing knowledge of data frames in R; however, some common subsets are verbose using existing tools or difficult to compute without knowing Argo-specific filename conventions. To match the syntax of `dplyr::filter()`, `argodata` provides several `argo_filter_*()` functions to subset index data frames:

```
prof_gulf_stream_2020 <- prof %>%
  argo_filter_radius(latitude = 26, longitude = -84, radius = 500) %>%
  argo_filter_date("2020-01-01", "2020-12-31") %>%
  argo_filter_data_mode("delayed")
```

The next step is to download the selected files. This is done automatically by the load functions described below; however, one can use `argo_download()` to download (if necessary) and cache files in an index. To facilitate use of alternative cache solutions like `rsync` or `bowerbird`, we use the same file structure as the mirror itself and provide `argo_set_cache_dir()` to allow this directory to be used for all calls to `argo_download()`.

To load data from NetCDF files into meaningful data frames we draw from the concept of "grids" introduced by the `tidync` and `ncmeta` packages (Sumner, 2020a, 2020b). For example, temperature values stored in an Argo profile NetCDF file are identified by values of `N_PROF` (an integer identifying a profile within an Argo NetCDF file) and `N_LEVEL` (an integer identifying a sampling level within a profile). Temperature values can be represented by a matrix with one row per `N_LEVELS` and one column per `N_PROF` or by a data frame with variables `N_PROF`, `N_LEVELS`, and `TEMP`. Any other variables that share the dimensions of the temperature variable can be added as additional columns in the data frame. After looping through each file in a complete copy of the Argo data set, we identified 19 grids among the four file types. The most commonly-used grid is the levels grid for Argo profile files:

```
(levels <- prof_gulf_stream_2020 %>%
  argo_prof_levels())
```

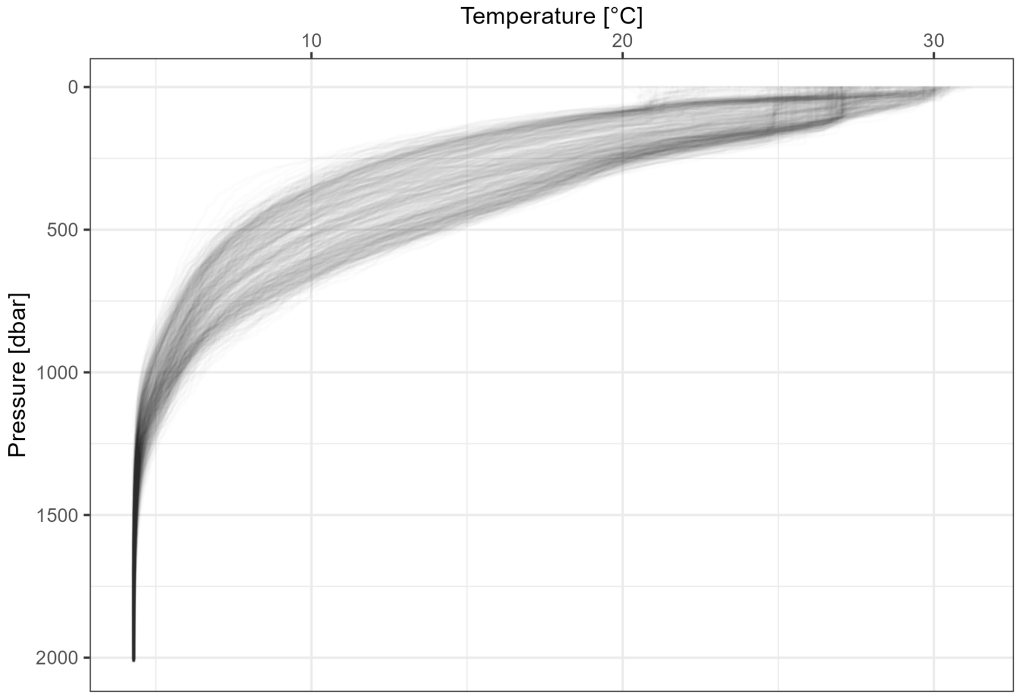
```
## Downloading 700 files from 'https://data-argo.ifremer.fr'
```

```
## Extracting from 700 files
```

```
## # A tibble: 1,360,320 x 18
##    file   n_levels n_prof  pres pres_qc pres_adjusted pres_adjusted_qc
##    <chr>     <int>  <int> <dbl> <chr>           <dbl> <chr>
##  1 aoml/~        1      1  1.12 1                1.12 1
##  2 aoml/~        2      1  2    1                2    1
##  3 aoml/~        3      1  3    1                3    1
##  4 aoml/~        4      1  4    1                4    1
##  5 aoml/~        5      1  4.96 1                4.96 1
##  6 aoml/~        6      1  6    1                6    1
##  7 aoml/~        7      1  7    1                7    1
##  8 aoml/~        8      1  7.92 1                7.92 1
##  9 aoml/~        9      1  9    1                9    1
## 10 aoml/~       10      1 10    1               10    1
## # ... with 1,360,310 more rows, and 11 more variables:
## #   pres_adjusted_error <dbl>, temp <dbl>, temp_qc <chr>,
## #   temp_adjusted <dbl>, temp_adjusted_qc <chr>,
## #   temp_adjusted_error <dbl>, psal <dbl>, psal_qc <chr>,
## #   psal_adjusted <dbl>, psal_adjusted_qc <chr>,
## #   psal_adjusted_error <dbl>
```

Like `argo_prof_levels()`, other extraction functions use the pattern `argo_{file type}_{grid}()` and use a split-apply-combine strategy that row-binds the results obtained by reading each file individually (Wickham, 2011). To facilitate users who prefer to manage their own collection of Argo files, corresponding `argo_read_{file type}_{grid}()` functions that read a single file are also exported. Extraction functions are designed to return useful inputs to `dplyr` and `ggplot2`. For example, a common way to visualize profile data is to plot a dependent variable (e.g., temperature) against pressure (as a proxy for depth), with pressure oriented vertically to simulate its orientation in space.

```
ggplot(levels, aes(x = temp, y = pres)) +
  geom_line(aes(group = file), alpha = 0.01, orientation = "y") +
  scale_y_reverse() +
  scale_x_continuous(position = "top") +
  theme_bw() +
  labs(
    x = "Temperature [°C]",
    y = "Pressure [dbar]"
  )
```

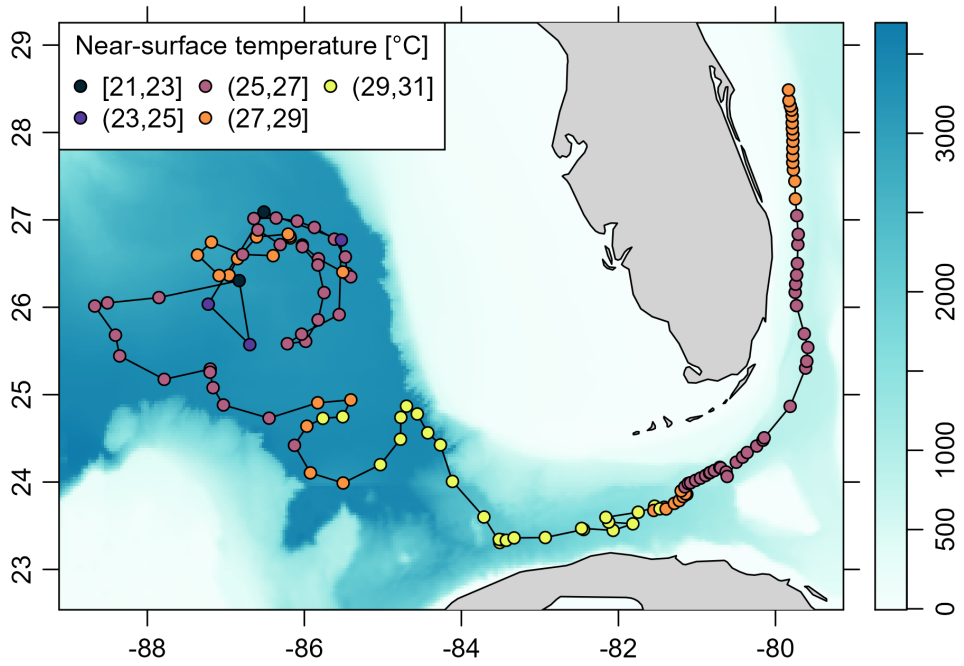


Interoperability

The `argodata` package was designed to interoperate with the `argoFloats` and `oce` packages for users who prefer to do part of their analyses using the facilities provided by these packages. In particular, these packages provide specialized functions for mapping and oceanographic analysis that are outside the scope of `argodata`. For example, one can combine the trajectory plotting capability of `argoFloats` with a `dplyr` `group_by()` and `summarise()` enabled by `argodata` and visualized using colour palettes from `cmocean` (Thyng et al., 2016).

```
library(argoFloats)

# use argoFloats to locate profiles
index <- getIndex() %>% subset(ID = 4903252)

# calculate mean surface temperature using argodata
temp_calc <- index %>%
  argo_prof_levels() %>%
  filter(pres < 10) %>%
  group_by(file) %>%
  summarise(
    near_surface_temp = mean(temp, na.rm = TRUE)
  ) %>%
  mutate(
    near_surface_temp_bin = cut_width(near_surface_temp, width = 2)
  ) %>%
  left_join(argo_global_prof(), by = "file")

# use plot method for argoFloats index and add temperatures
par(mar = c(3, 3, 1, 2))
plot(index, which = "map", type = "l")
```

```
# plot temperatures
palette(cmocean::cmocean("thermal")(5))
points(
  temp_calc$longitude, temp_calc$latitude,
  bg = temp_calc$near_surface_temp_bin, pch = 21, cex = 1
)

legend(
  "topleft",
  levels(temp_calc$near_surface_temp_bin), pt.bg = palette(), pch = 21,
  title = "Near-surface temperature [°C]", ncol = 3
)
```



Conclusion

The `argodata` package helps scientists analyze data from the International Argo Program using a minimal table-based interface. We hope that `argodata` will expand the audience of Argo data to users already familiar with data frame manipulation tools such as those provided by the `tidyverse` family of packages.

Acknowledgements

We acknowledge useful discussions with Chris Gordon, especially regarding the extraction of quality control information from Argo data files. Support for this work came from the Natural Sciences and Engineering Research Council of Canada and G7 Charlevoix Blueprint for Healthy Oceans, Seas and Resilient Coastal Communities. The data used in this paper were collected and made freely available by the International Argo Program and the national programs that

contribute to it (https://argo.ucsd.edu, https://www.ocean-ops.org). The Argo Program is part of the Global Ocean Observing System.

References

Chamberlain, S. (2021). *Rnoaa: 'NOAA' weather data from r*. https://CRAN.R-project.org/package=rnoaa

Kelley, D. E. (2018). *Oceanographic Analysis with R*. Springer-Verlag. ISBN: 978-1-4939-8842-6

Kelley, D. E., Harbin, J., & Richards, C. (2021). argoFloats: An R package for analyzing Argo data. *Frontiers in Marine Science*, *8*, 636922. https://doi.org/10.3389/fmars.2021.635922

Kelley, D. E., & Richards, C. (2021). *Oce: Analysis of oceanographic data*. https://CRAN.R-project.org/package=oce

Maze, G., & Balem, K. (2020). Argopy: A Python library for Argo ocean data analysis. *Journal of Open Source Software*, *5*(53), 2425. https://doi.org/10.21105/joss.02425

OceanOPS. (2021). https://www.ocean-ops.org/board?t=argo

Raymond, B., & Sumner, M. (2021). *Bowerbird: Keep a collection of sparkly data resources*. https://docs.ropensci.org/bowerbird

Roemmich, D., Boebel, O., Desaubies, Y., Freeland, H., Kim, K., King, B., Le Traon, P.-Y., Molinari, R., Owens, B. W., Riser, S., Send, U., Takeuchi, K., & Wijffels, S. (2001). *Argo: The Global Array of Profiling Floats*. C.J. Koblinsky; N.R. Smith. https://archimer.ifremer.fr/doc/00090/20097/

Roemmich, D., Johnson, G. C., Riser, S., Davis, R., Gilson, J., Owens, W. B., Garzoli, S. L., Schmid, C., & Ignaszewski, M. (2009). The Argo Program: Observing the global ocean with profiling floats. *Oceanography*, *22*(2), 34–43. https://doi.org/10.5670/oceanog.2009.36

Sumner, M. (2020a). *Ncmeta: Straightforward 'NetCDF' metadata*. https://CRAN.R-project.org/package=ncmeta

Sumner, M. (2020b). *Tidync: A tidy approach to 'NetCDF' data exploration and extraction*. https://CRAN.R-project.org/package=tidync

Thyng, K. M., Greene, C. A., Hetland, R. D., Zimmerle, H. M., & DiMarco, S. F. (2016). True colors of oceanography: Guidelines for effective and accurate colormap selection. *Oceanography*, *29*(3). https://doi.org/10.5670/oceanog.2016.66

Tucker, T., Giglio, D., Scanderbeg, M., & Shen, S. S. P. (2020). Argovis: A Web Application for Fast Delivery, Visualization, and Analysis of Argo Data. *Journal of Atmospheric and Oceanic Technology*, *37*(3), 401–416. https://doi.org/10.1175/JTECH-D-19-0041.1

Welcome to the tidyverse. (2019). *Journal of Open Source Software*, *4*(43), 1686. https://doi.org/10.21105/joss.01686

Welcome to the tidyverse. (2019). *Journal of Open Source Software*, *4*(43), 1686. https://doi.org/10.21105/joss.01686

Wickham, H. (2011). The Split-Apply-Combine Strategy for Data Analysis. *Journal of Statistical Software*, *40*(1, 1), 1–29. https://doi.org/10.18637/jss.v040.i01

Wickham, H. (2014). Tidy Data. *Journal of Statistical Software*, *59*(1), 1–23. https://doi.org/10.18637/jss.v059.i10

Wickham, H. (2016). *ggplot2: Elegant graphics for data analysis*. Springer-Verlag New York. ISBN: 978-3-319-24277-4

Wickham, H., François, R., Henry, L., & Müller, K. (2021). *Dplyr: A grammar of data manipulation*. https://CRAN.R-project.org/package=dplyr

Wickham, H., & Grolemund, G. (2017). *R for Data Science: Import, Tidy, Transform, Visualize, and Model Data*. O'Reilly Media. https://r4ds.had.co.nz