ARGOs Exploratory Data Analysis

Axel Amzallag, Josh Hug, Tim Kalnins

18 Feb 2021

```
library(rmarkdown)
library(knitr)
```

Domain Restriction

As we begin to tackle the ARGOs project one of the most daunting initial challenges faced is dealing with the geospatial breadth of the data. With over 4000 floats distributed across the entire surface of the World Ocean, an area spanning over 361.9 million square kilometers, it would be prudent to restrict our domain to a more reasonable size at the onset. As such, we have come to the agreement that we will initially restrict our analyses to the geospatial box defined by the following four points (latitude degree north, longitude degree east): (60, 120), (60, 160),(50, 160), (50, 120). This box bounds the waters surrounding the Japanese archipelago, and was chosen for the following characteristics:

- The existence of a wide range of climatic types: the top of the box resides in a subarctic climate while the bottom has a humid subtropical climate.
- A smaller number of observations, making our initial correlation length scales computations, mean-field constructions, and thermocline estimations easier.
- The potential for interesting applications/interactions:
  - linking our findings with seismic/tsunami data
  - the impact of climatic changes on the area's fishing industry
  - geopolitical relevance: thermocline estimation is applicative to submarine warfare as the changing water densities surrounding the thermocline can act as a reflect barrier to sonar signals.
- The high population density of the surrounding coasts and heavily trafficked shipping lanes makes this particular area globally significant.

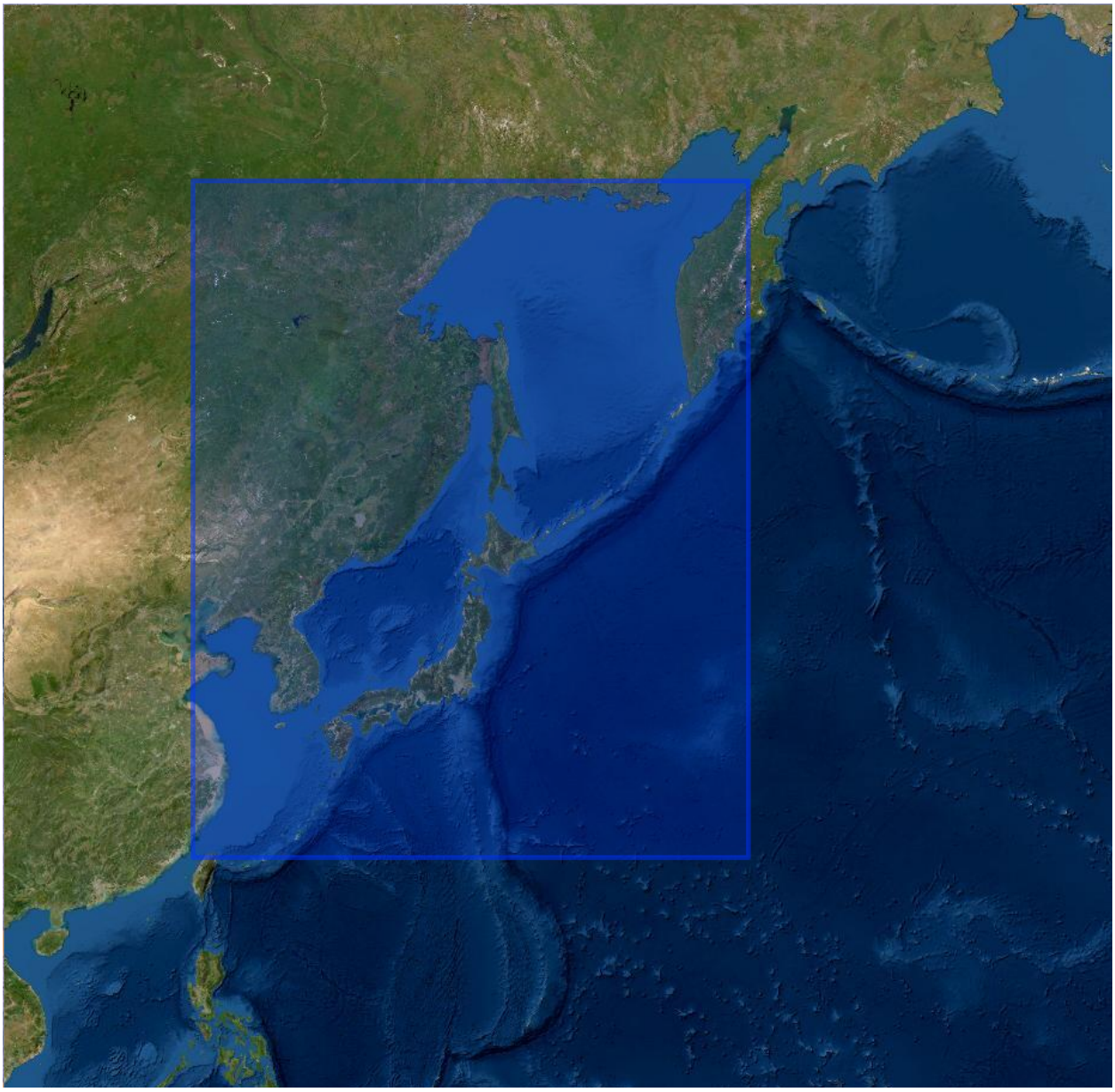


Figure 1: Visualization of our initial geospatial bounds

Data Scraping

The ARGOs project stores the data reported by their floats as binary `NetCDF` files; making them available to the public via multiple `FTP` servers. To automate the data acquisition process, a shell script (featuring the `wget` utility) was written to access all `NetCDF` files from a given year and month range. As an example the following one-liner would scrape and store all `NetCDF` files reported between the time frame of January 2017 and February 2020 in the directory `./data`.

```
./download.sh 2017 2020 01 02
```

```
#!/usr/bin/env bash

## short script to download NetCDF binary files from ARGOs ftp servers
## feed a start year , end year, start month end month
## to select all files within {start year,..,end year} X {start month,..,end month}
## depends on wget
##
##   >  chmod 755 download.sh
##   >  ./download.sh

if [ $# -eq 0 ];then
    echo call with help flag [-h] for usage information
    echo "./download.sh [-h] start_year end_year start_month end_month"
    exit 1
fi

if [[ "$1" == "-h" ]]; then
    echo Usage information
    echo -e Run \"./download.sh start_year end_year start_month end_year\" to extract all NetCDF files
    echo from all months from start_month to end_month during the timespan of start_year end_year
    echo
    echo
    echo -e Run \"./download.sh -y year start_month end_year\" to extract all NetCDF files
    echo from all months from start_month to end_month in the given year
    exit 0

fi

if [[ "$1" == "-y" ]]; then
    year=$2
    strt_mnth=$3
    end_mnth=$4
    url=$(echo ftp://ftp.ifremer.fr/ifremer/argo/geo/pacific_ocean/${year}/{$strt_mnth..$end_mnth}/*)
    eval wget ${url}  -q -P ./data
    exit 0
fi

strt_year=$1
strt_mnth=$3
end_year=$2
end_mnth=$4

url=$(echo ftp://ftp.ifremer.fr/ifremer/argo/geo/pacific_ocean/{$strt_year..$end_year}/{$strt_mnth..$en
eval wget ${url}  -q -P ./data
```

Initial Data Wrangling & Formatting Choices

The data is in a atypical format that is not easy to manipulate. The method that we use here is to essentially create two data frames, the first that contains all the necessary variables for which there is only 1 measurement per buoy. These are variables like the longitude, latitude and Julian date of the buoy as it sends data to the satellite. The second data frame contains all of the measurements the buoys take on their ascent such as salinity, temperature and pressure. In addition we reproduce the column of Julian date and buoy number in order to uniquely identify the rows. This acts as a kind of database with two tables. The key is the combination of the Julian date and the buoy ID number. The only function that will actually be called is the final function `get_multiple_dfs` which takes in a vector of file names as well as the requested necessary variables and measurements and produces a list of two dataframes as described above with all the observations of all the files combined restricted to the latitude and longitude we described earlier.

```
library(tidyverse)
library(ncdf4)


# Helper function that pivots a data frame. The rows become observations
# for a single covariate, which is the only column.

cov_piv_long <- function(single_covariate, nc) {
  pre_pivot <- data.frame(ncvar_get(nc, single_covariate))
  pivoted_df <- pivot_longer(pre_pivot, cols  = colnames(pre_pivot)) %>%
                select(value)
  colnames(pivoted_df) <- single_covariate
  return(pivoted_df)
}


# Takes in a single file name, key variables, and a vector of covariates
# and returns a list of data frames.
#
# The outputted list contains the following:
#     Element 1: Data frame with only the key variables. This data frame is
#                relatively small, as it contains only 1 row per buoy.
#
#     Element 2: Data frame with the observed data. All of the key variables from
#                the first data frame are included with each observation. This
#                data frame is much larger than the first one, as each buoy has
#                many rows.
#

get_dfs <- function(fileName, key_vars = c("LONGITUDE", "LATITUDE", "JULD"), covs = NULL) {
  # Make the first data frame: key variables filtered to desired region
  nc <- nc_open(fileName)
  key_vars_df <-
    data.frame(purrr::map(.x = key_vars, ~ ncvar_get(nc, .x)))
  colnames(key_vars_df) <- key_vars

  # Filter to the correct latitude and longitude
  key_vars_df <- key_vars_df %>%
    filter((LONGITUDE <= 160  ) & (LONGITUDE >= 120) &
           (LATITUDE <= 60 ) & (LATITUDE >=25 ))

  # Make the second data frame: covariates and keys together in the
```

```
  # same data frame
  if (!is.null(covs)) {
    keys_missing <-
      purrr::map_dfc(.x = covs, .f = ~ cov_piv_long(.x, nc))
    num_obs <- dim(keys_missing)[1]
    num_buoys <- length(ncvar_get(nc, "PLATFORM_NUMBER"))
    keys <-
      data.frame(
        PLATFORM_NUMBER = rep(ncvar_get(nc, "PLATFORM_NUMBER"),num_obs / num_buoys),
        JULD = rep(ncvar_get(nc, "JULD"),num_obs / num_buoys))

    # Joining data frames for second output
    covs_df <- cbind(keys, keys_missing)
    covs_df <- dplyr::inner_join(key_vars_df, covs_df,
                                 by=c("PLATFORM_NUMBER","JULD"))
    return(list(key_vars_df, covs_df))
  } else{
    return(list(key_vars_df))
  }
}

# Applies the funtion get_dfs to all files in vec_files and row binds them
# together. Output is a list similar in structure to get_dfs.

get_multiple_dfs <- function(vec_files, key_vars, covs = NULL) {
  combined_keys <-
    purrr::map_dfr(.x = vec_files, ~ get_dfs(
      fileName = .x,
      key_vars = key_vars,
      covs = covs
    )[[1]])

  if (!is.null(covs)) {
    combined_covs <-
      purrr::map_dfr(.x = vec_files, ~ get_dfs(
        fileName = .x,
        key_vars = key_vars,
        covs = covs
      )[[2]])
    return(list(combined_keys, combined_covs))
  } else {
    return(list(combined_keys))
  }
}
```

Plots and Early Examinations

From our inspection of a very small slice of the overall data one characteristic becomes very apparent: there is a great deal of sparsity. As shown by the following printouts every float has at least some observations with absent feature values. In many cases, every observation for a given float is missing a feature value. Determining how to best handle this sparsity will clearly be a priority for us as we go forward with the project.

Following these table printouts, are some simple plots of float temperature and pressure readings. Clearly there is a strong relationship between the two, and the shapes provided capture how drastically different the data can be between floats.

```
library(lubridate)
```

```
##
## Attaching package: 'lubridate'
```

```
## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```
## AN EXAMPLE


# Key Values and Attributes: 1 observation per buoy
buoy <- "PLATFORM_NUMBER"
lon <- "LONGITUDE"
lat <- "LATITUDE"
tempqc <- "TEMP_ADJUSTED_QC"
presqc <-"PRES_ADJUSTED_QC"
salqc <- "PSAL_QC"
juld <- "JULD"
juldqc <- "JULD_QC"
refdt <- "REFERENCE_DATE_TIME"


# Covariates: Thousands of observations per buoy
temp <- "TEMP_ADJUSTED"
pres <- "PRES_ADJUSTED"
sal <- "PSAL_ADJUSTED"


file2 <- "../Data_Pipeline/data/20210202_prof.nc"
file1 <- "../Data_Pipeline/data/20210101_prof.nc"
key_vars <-c(buoy, lon, lat, tempqc, presqc, salqc, juld, juldqc, refdt)
covs <- c(temp, pres, sal)

bothDataFrames <- get_multiple_dfs(c(file1,file2),key_vars = key_vars, covs = covs)
covariatesDF <- bothDataFrames[[2]]


# This converts the universal dates to a more standard date format based on the
# origin date given by REFERENCE_DATE_TIME
covariatesDF$DATE_TIME <- lubridate::as_date(covariatesDF$JULD,
                   origin=lubridate::as_datetime(covariatesDF$REFERENCE_DATE_TIME))

glimpse(covariatesDF)
```

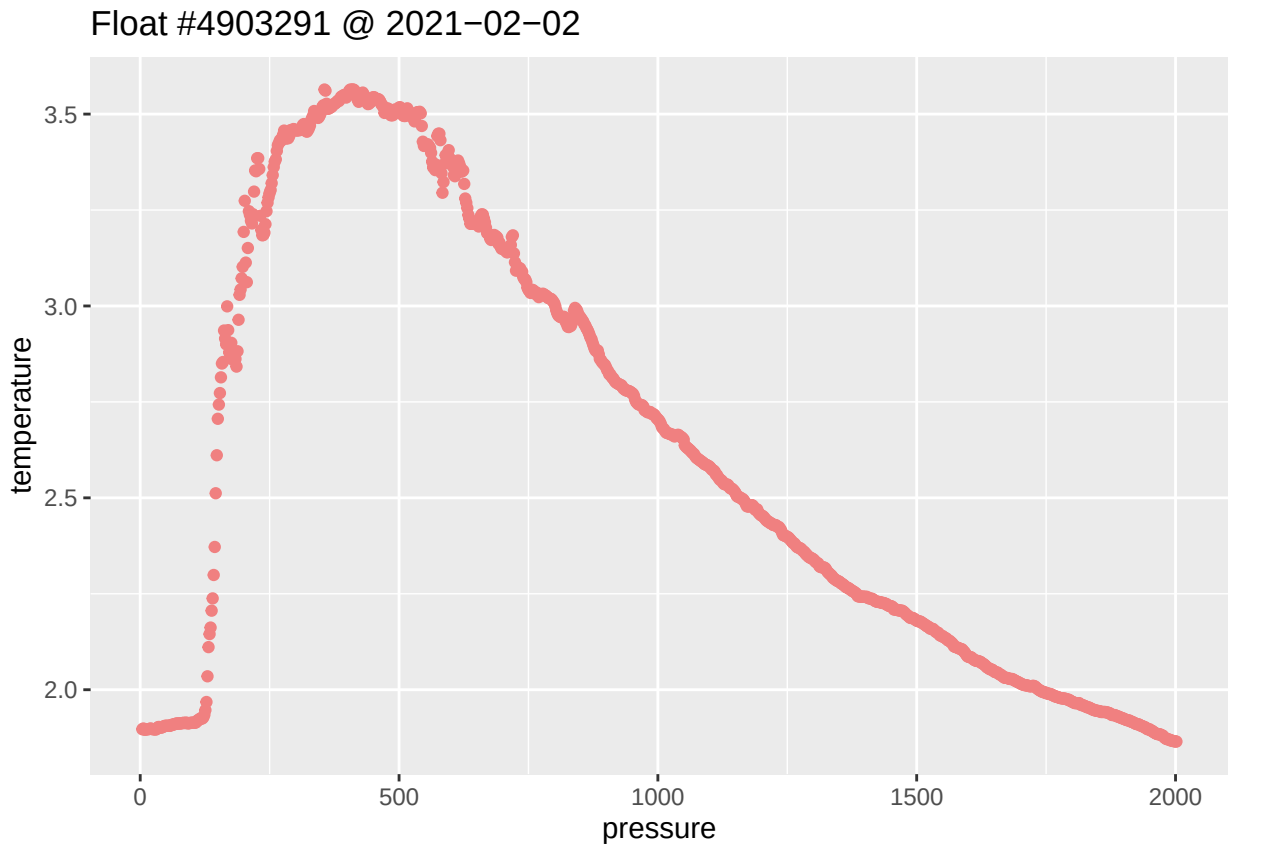
```
## Rows: 27,341
## Columns: 13
## $ PLATFORM_NUMBER     <chr> "2903632 ", "2903632 ", "2903632 ", "2903632 ",...
## $ LONGITUDE           <dbl> 151.457, 151.457, 151.457, 151.457, 151.457, 15...
## $ LATITUDE            <dbl> 34.301, 34.301, 34.301, 34.301, 34.301, 34.301,...
## $ TEMP_ADJUSTED_QC    <chr> "1111111111111111111111111111111111111111111111...
## $ PRES_ADJUSTED_QC    <chr> "1111111111111111111111111111111111111111111111...
## $ PSAL_QC             <chr> "1111111111111111111111111111111111111111111111...
## $ JULD                <dbl> 25933.5, 25933.5, 25933.5, 25933.5, 25933.5, 25...
## $ JULD_QC             <chr> "1111111111111111111111111111111111111111111111...
## $ REFERENCE_DATE_TIME <chr> "19500101000000", "19500101000000", "1950010100...
## $ TEMP_ADJUSTED       <dbl> 19.1616, 19.1625, 19.1668, 19.1687, 19.1697, 19...
## $ PRES_ADJUSTED       <dbl> 4.45, 6.15, 8.15, 10.15, 12.15, 14.15, 16.15, 1...
## $ PSAL_ADJUSTED       <dbl> 34.5817, 34.5825, 34.5824, 34.5822, 34.5824, 34...
## $ DATE_TIME           <date> 2021-01-01, 2021-01-01, 2021-01-01, 2021-01-01...
```

```r
# proportion of entries that have missing data
grouped <- covariatesDF %>% group_by(PLATFORM_NUMBER) %>%
    summarise(sum(is.na(PRES_ADJUSTED)) /  length(PRES_ADJUSTED))

print(grouped, n=24)
```

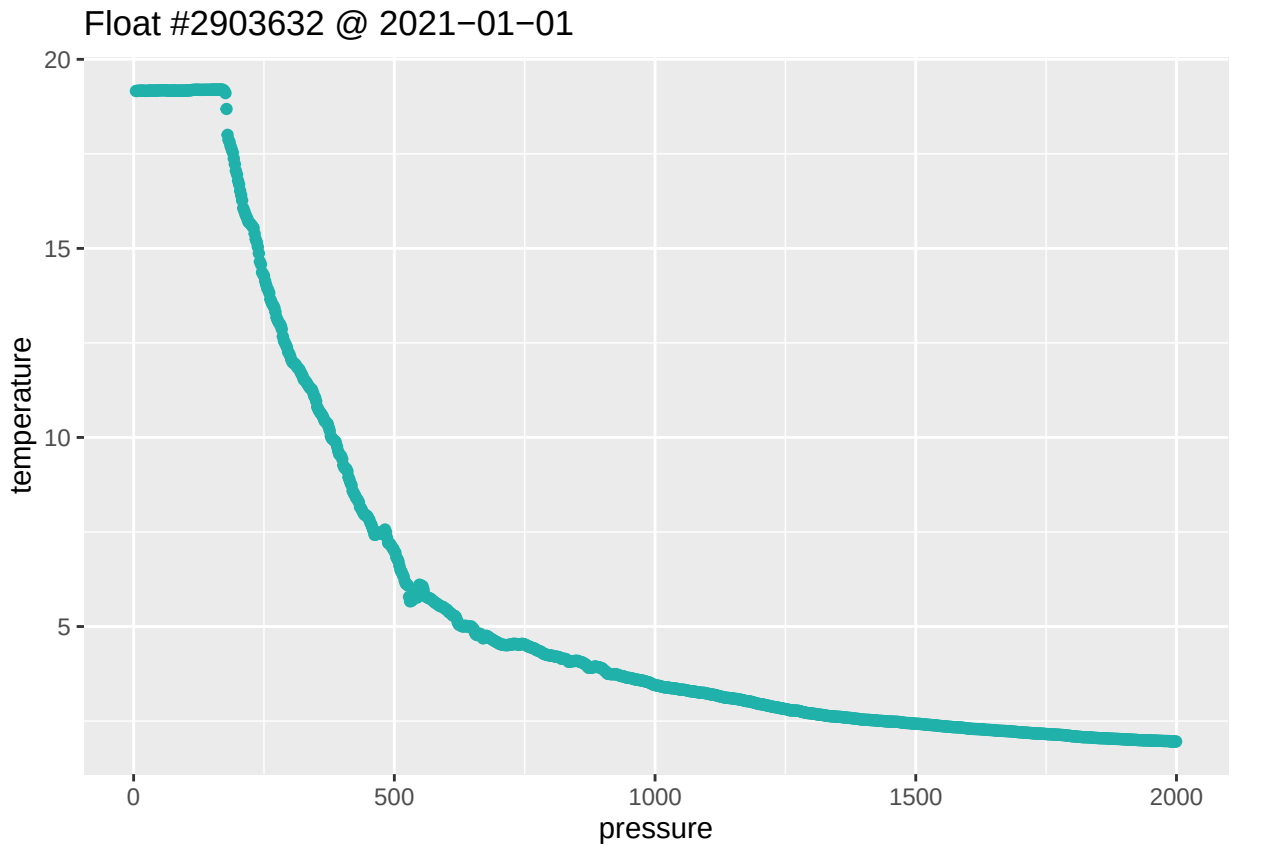
```
## # A tibble: 24 x 2
##    PLATFORM_NUMBER `sum(is.na(PRES_ADJUSTED))/length(PRES_ADJUSTED)`
##  * <chr>                                                       <dbl>
##  1 "2901799 "                                                  1
##  2 "2901800 "                                                  1
##  3 "2901801 "                                                  1
##  4 "2901802 "                                                  1
##  5 "2901803 "                                                  1
##  6 "2901804 "                                                  1
##  7 "2902754 "                                                  1
##  8 "2903211 "                                                  1
##  9 "2903342 "                                                  1
## 10 "2903361 "                                                  1
## 11 "2903387 "                                                  0.925
## 12 "2903621 "                                                  1
## 13 "2903625 "                                                  1
## 14 "2903628 "                                                  1
## 15 "2903631 "                                                  0.0114
## 16 "2903632 "                                                  0.0123
## 17 "2903634 "                                                  1
## 18 "2903637 "                                                  1
## 19 "2903640 "                                                  1
## 20 "2903644 "                                                  1
## 21 "2903646 "                                                  1
## 22 "2903647 "                                                  1
## 23 "2903673 "                                                  1
## 24 "4903291 "                                                  0.0128
```

```r
dat <- covariatesDF

# plot of buoys pressure vs temp
# generate 6 individual plot objects
p1 <- ggplot(dat %>% filter(PLATFORM_NUMBER == "4903291 "), aes(x=PRES_ADJUSTED, y=TEMP_ADJUSTED)) +
    geom_point(color="lightcoral") +
    labs(x= "pressure", y= "temperature", title = "Float #4903291 @ 2021-02-02")

# dat %>% filter(PLATFORM_NUMBER == "2903632 ") %>% select(DATE_TIME) %>% unique()

p2 <- ggplot(dat %>% filter(PLATFORM_NUMBER == "2903632 ") %>%
             filter(str_detect(DATE_TIME, "2021-01-01")),
             aes(x=PRES_ADJUSTED, y=TEMP_ADJUSTED)) +
    geom_point(color="lightseagreen") +
    labs(x= "pressure", y= "temperature", title = "Float #2903632 @ 2021-01-01")

p3 <- ggplot(dat %>% filter(PLATFORM_NUMBER == "2903632 ") %>%
             filter(str_detect(DATE_TIME, "2021-02-02")),
             aes(x=PRES_ADJUSTED, y=TEMP_ADJUSTED)) +
    geom_point(color="lightskyblue") +
    labs(x= "pressure", y= "temperature", title = "Float #2903632 @ 2021-01-02")

p4 <- ggplot(dat %>% filter(PLATFORM_NUMBER == "2903631 ") %>%
             filter(str_detect(DATE_TIME, "2021-01-01")),
             aes(x=PRES_ADJUSTED, y=TEMP_ADJUSTED)) +
    geom_point(color="lightskyblue4") +
    labs(x= "pressure", y= "temperature", title = "Float #2903631 @ 2021-01-01")


p5 <- ggplot(dat %>% filter(PLATFORM_NUMBER == "2903631 ") %>%
             filter(str_detect(DATE_TIME, "2021-02-02")),
             aes(x=PRES_ADJUSTED, y=TEMP_ADJUSTED)) +
    geom_point(color="thistle4") +
    labs(x= "pressure", y= "temperature", title = "Float #2903631 @ 2021-02-02")


p6 <- ggplot(dat %>% filter(PLATFORM_NUMBER == "2903387 ") %>%
             filter(str_detect(DATE_TIME, "2021-02-02")),
             aes(x=PRES_ADJUSTED, y=TEMP_ADJUSTED)) +
    geom_point(color="mediumseagreen") +
    labs(x= "pressure", y= "temperature", title = "Float #2903387 @ 2021-02-02")
```

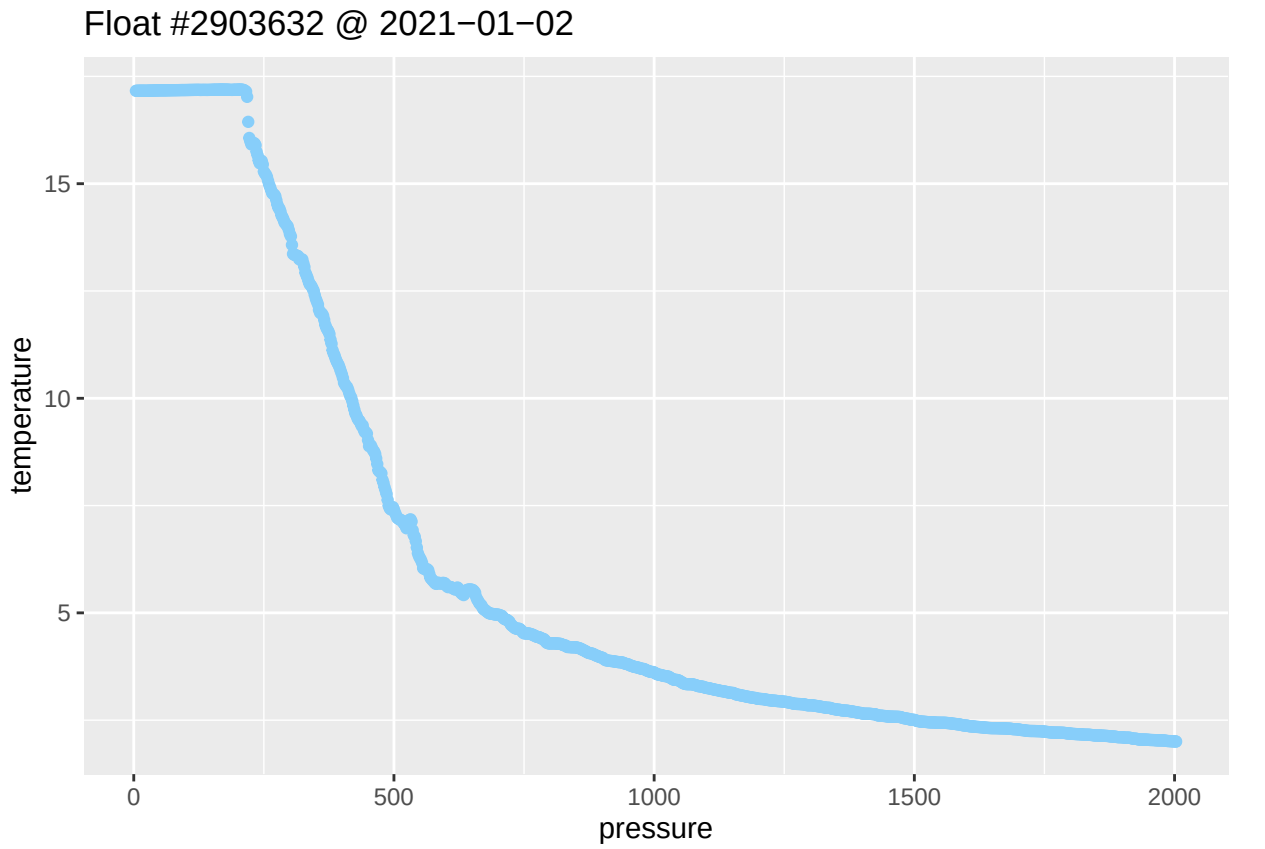
```
print(p1)
```


Float #4903291 @ 2021−02−02
temperature
3.5
3.0
2.5
2.0
0
500
1000
1500
2000
pressure

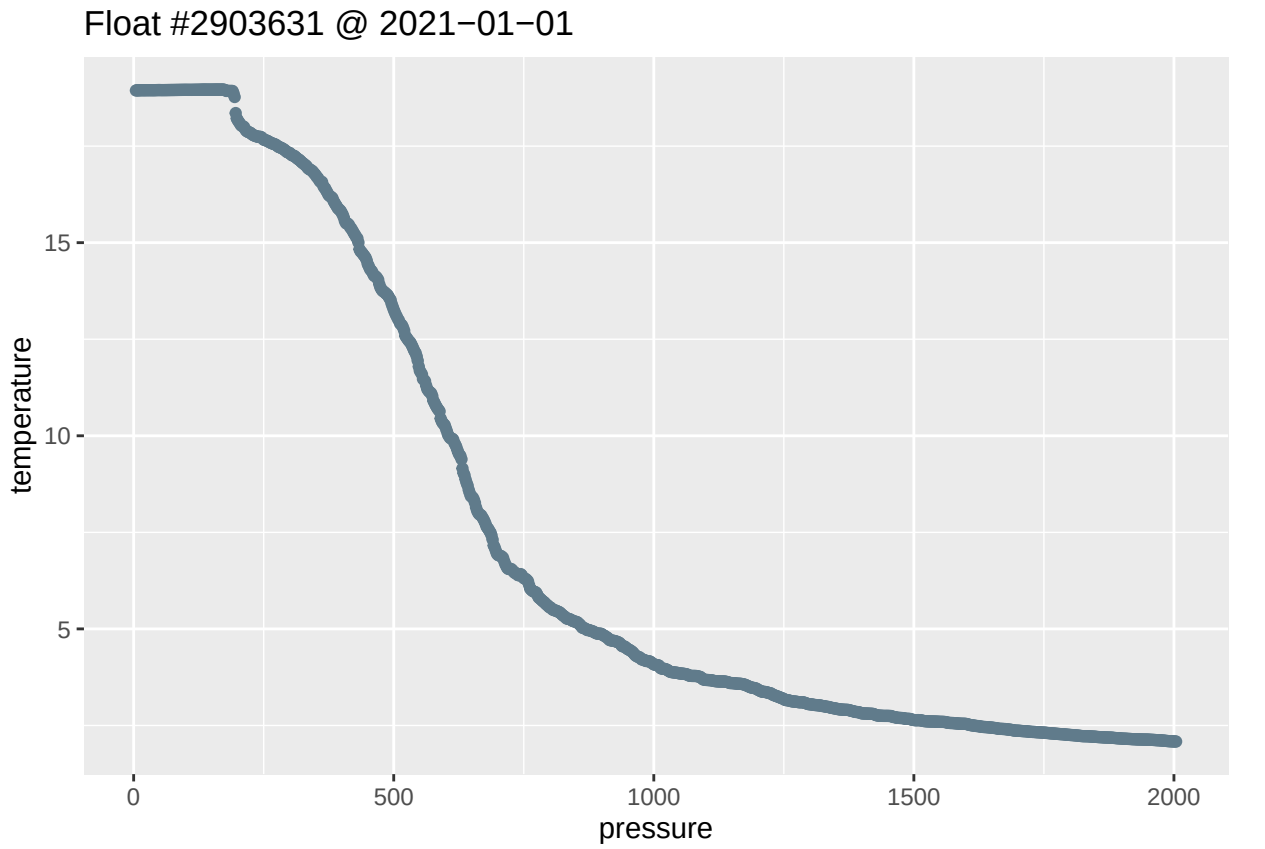
```
print(p2)
```


Float #2903632 @ 2021−01−01
temperature
pressure
20
15
10
5
0
500
1000
1500
2000

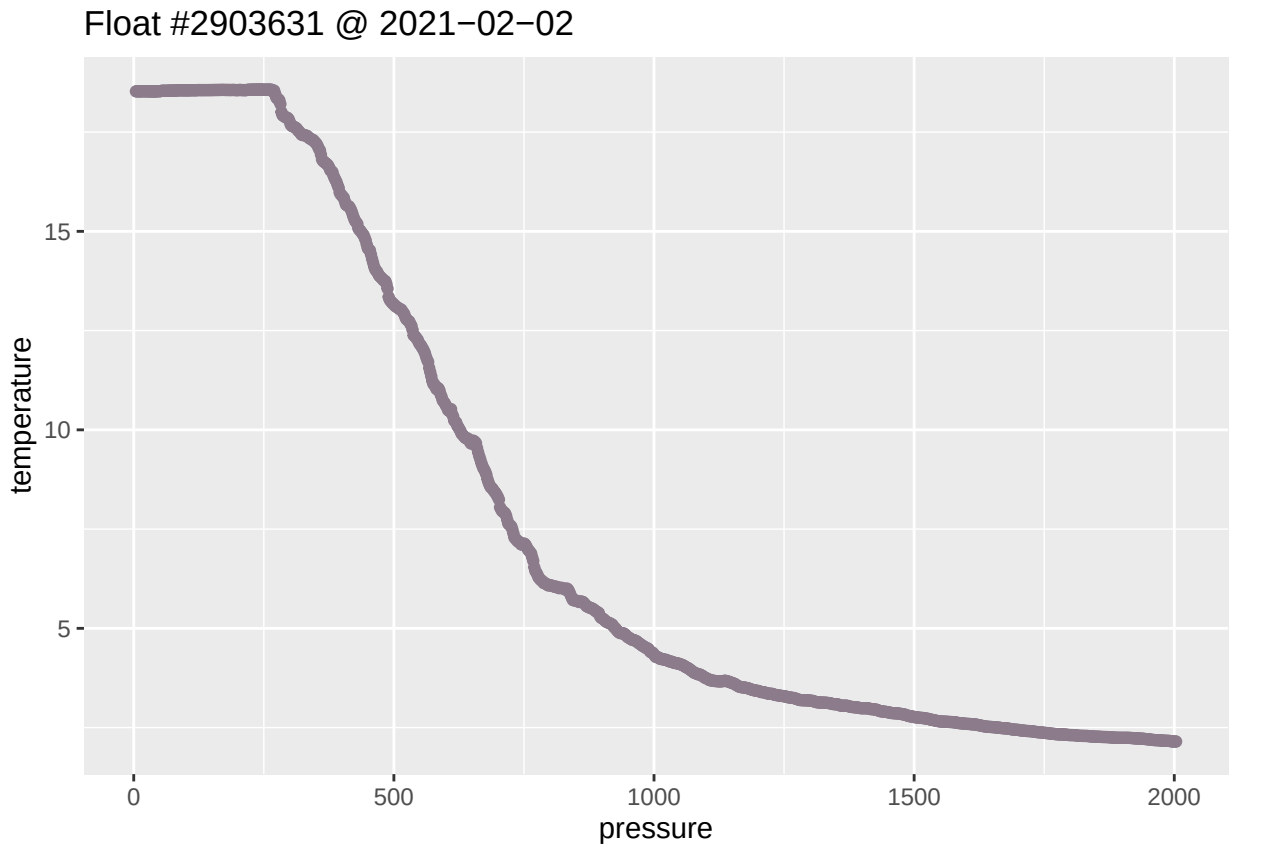
```
print(p3)
```


Float #2903632 @ 2021−01−02
temperature
15
10
5
0
500
1000
1500
2000
pressure

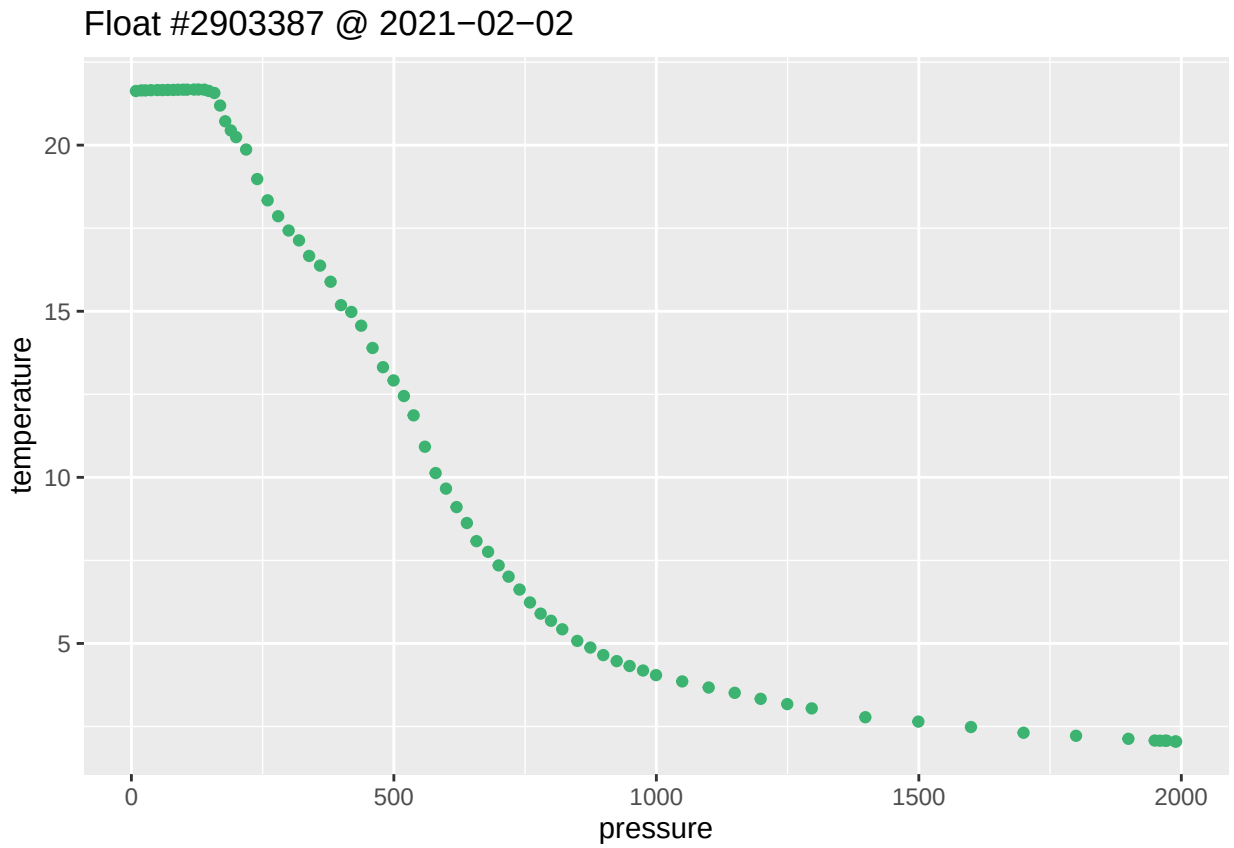
```
print(p4)
```


Float #2903631 @ 2021−01−01
temperature
15
10
5
0
500
1000
1500
2000
pressure

```
print(p5)
```


Float #2903631 @ 2021−02−02
temperature
15
10
5
0
500
1000
1500
2000
pressure

```
print(p6)
```


Float #2903387 @ 2021−02−02
temperature
20
15
10
5
0
500
1000
1500
2000
pressure