US Army Corps of Engineers®

U.S. Army Corps of Engineers Assimilation of NOAA National Data Buoy Center Archive Data Processing Code

Candice Hall

U.S. Army Corps of Engineers (USACE) Assimilation of NOAA National Data Buoy Center (NDBC) Archive data processing codes included within this document:

1. Master run_*xxxx*.R
2. download_data_1.R
3. read_NetCDF_pre_2011.R
4. read_NetCDF_post_2011.R
5. concat_data_2.R
6. concat_ndbc.R
7. concat_ncei.R
8. verify_netcdf_3.R
9. geoClean_data_4.R
10. plot_stdmet.R
11. plots_spec.R
12. create_best_data_5.R
13. build_thredds_netcdf_6.R

US Army Corps
of Engineers®

Master run_*xxxx*.R

```
#-------------------------------------------------------------------------------------------
#-------------------------------------------------------------------------------------------

## master script

# This script initiates the function scripts within this product suite within the correct order.
# This script can run all the buoy stations in serial, or can be copied to run the function scripts
# on the HPC in parallel.

#-------------------------------------------------------------------------------------------
#-------------------------------------------------------------------------------------------
## install libraries
# install.packages("lubridate")
# install.packages("R.utils")
# install.packages("qpcR")
# install.packages("plyr")
# install.packages("dplyr")
# install.packages("ncdf4")
# install.packages("tidyverse")
# install.packages("readxl") # dependent on tidyverse
# install.packages("stringr")
# install.packages("data.table")
# install.packages("tidyr")
# install.packages("gridExtra")
# install.packages("oce")
# install.packages("naniar")
# install.packages("broom")
# install.packages("openair")  # polarplots
# install.packages("plotly")
# install.packages("magrittr")
# install.packages("grid")
# install.packages("devtools")
# install.packages("lsr")
# install.packages("RColorBrewer")
# install.packages("viridis")
# install.packages("colorRamps")
# install.packages("ggplot2")
# install.packages("ggmap")
# install.packages("maps")
# install.packages("mapdata")

## load libraries (local runs)
library(lubridate)
library(R.utils)
library(qpcR)
library(plyr)
library(dplyr)
library(tibble)
# library(ncdf4)
library(tidyverse)
```

```r
library(readxl) # dependent on tidyverse
library(stringr)
library(data.table)
library(tidyr)
library(gridExtra)
# library(sf)
# library(oce)
library(naniar)
library(broom)
library(openair) # polarplots
library(plotly)
library(magrittr)
# library(grid)
# library(devtools)
library(lsr)
library(RColorBrewer)
library(viridis)
library(colorRamps)
library(ggplot2)
library(ggmap)
library(maps)
library(mapdata)
library(modeest)

## load libraries (HPC runs)

# library(lubridate, lib="/p/home/candice/Rlibs/")
# library(R.utils, lib="/p/home/candice/Rlibs/")
# library(qpcR, lib="/p/home/candice/Rlibs/")
# library(plyr, lib="/p/home/candice/Rlibs/")
# library(dplyr, lib="/p/home/candice/Rlibs/")
# library(ncdf4, lib="/p/home/candice/Rlibs/")
# library(tidyverse, lib="/p/home/candice/Rlibs/")
# library(readxl, lib="/p/home/candice/Rlibs/") # dependent on tidyverse
# library(stringr, lib="/p/home/candice/Rlibs/")
# library(data.table, lib="/p/home/candice/Rlibs/")
# library(tidyr, lib="/p/home/candice/Rlibs/")
# library(gridExtra, lib="/p/home/candice/Rlibs/")
# library(sf, lib="/p/home/candice/Rlibs/")
# library(oce, lib="/p/home/candice/Rlibs/")
# library(naniar, lib="/p/home/candice/Rlibs/")
# library(broom, lib="/p/home/candice/Rlibs/")
# library(openair, lib="/p/home/candice/Rlibs/") # polarplots
# library(plotly, lib="/p/home/candice/Rlibs/")
# library(magrittr, lib="/p/home/candice/Rlibs/")
# library(grid, lib="/p/home/candice/Rlibs/")
# library(devtools, lib="/p/home/candice/Rlibs/")
# library(lsr, lib="/p/home/candice/Rlibs/")
# library(RColorBrewer, lib="/p/home/candice/Rlibs/")
# library(viridis, lib="/p/home/candice/Rlibs/")
```

```
# library(colorRamps, lib="/p/home/candice/Rlibs/")
# library(ggplot2, lib="/p/home/candice/Rlibs/")
# library(ggmap, lib="/p/home/candice/Rlibs/")
# library(maps, lib="/p/home/candice/Rlibs/")
# library(mapdata, lib="/p/home/candice/Rlibs/")
# library(modeest, lib="/p/home/candice/Rlibs/")


## select the following data locations and start date
start_date <- 1970
drive <- "F:/Candice/"
# drive <- "/p/work/candice/"
data_dir <- paste0(drive, "projects/WaveTrends/data/")
# data_dir <- paste0(drive, "projects/WaveTrends/annual_runs/data/")
setwd(data_dir)

## set colors for each buoy
plot_colors <- viridis(n=6)
# plot_colors <- matlab.like2(4)
color_ndbc_raw <- plot_colors[1]
color_ndbc_orig <- plot_colors[5]    #"#440154FF" # purple
color_ndbc_recalc <- plot_colors[3] # "#3B528BFF"  #"5 = #FDE725FF" # yellow
color_chl_calc <- plot_colors[4]  # "#21908CFF"   #"#21908CFF" # green
color_WIS <- "red"     # "#5DC863FF"

## set colors for wind/wave polar plots
# brewer.pal(n = 8, name = "RdBu")
colour1 <- viridis(n=4)
cols1 <- c(colour1[4], colour1[3], colour1[2], colour1[1])
colour2 <- viridis(n=5)
cols2 <- c(colour2[5], colour2[4], colour2[3], colour2[2], colour2[1])
# cols1 <- c("black", "green", "blue", "red")
# plot type, symbol, size
type = "l"
pch = "."
lwd = 0.5
cex = 0.5

## set plot parameters
## set parameters common to all plots
xlab = "Date"   # label for x axis
width = 2000+2000    # width of exported plot
height = 1500+1500   # height of exported plot
res = 300  # plot resolution
width1 = 1000
height1 = 700
par1 = c(5,5,4,4)
Delta <- '\U0394'
degree <- '\U00B0'
## my.grid function formats
```

```r
my.format <- "%m-%d-%Y" # "%m-%Y" (long datasets) or "%m-%d-%Y" (short datasets)
my.period <- "weeks" # "months" (long datasets) or "weeks (short datasets)
## my grid function for plots
my.grid <-function(dataset, my.period = "year", my.format = "%Y"){
        grid(nx=NA, ny=NULL)
        abline(v=axis.POSIXct(1, at=seq(min(dataset[1,1]), max(dataset[nrow(dataset),1]),
                                        by= my.period), format=my.format),
               col = "lightgray", lty = "dotted", lwd = par("lwd"))
}
## function to capitalize string
simpleCap <- function(x) {
        s <- strsplit(x, " ")[[1]]
        paste(toupper(substring(s, 1,1)), substring(s, 2),
              sep="", collapse=" ")
}

## The choice of significance level at which you reject H0 is arbitrary.
sig <- 0.01

## to run all of the buoy stations in serial, read in the NDBC buoy list as follows:
list_ndbc <- read.csv(paste0(data_dir,"NDBC_buoys.csv"),header = TRUE)
list_ndbc <- dplyr::filter(list_ndbc, list_ndbc$owner == "NDBC")
list_ndbc_buoy <- as.character(list_ndbc$station)
buoy_ls <- list_ndbc_buoy
rm(list_ndbc_buoy, list_ndbc)
print(buoy_ls)

# ## to run all of the buoy stations in serial, use the following code that assigns the names of this master
# ## files to the scripts to isolate a specific buoy number:
# buoy_ls <- gsub(".R","",unlist(strsplit(rstudioapi::getSourceEditorContext()$path,"__"))[2])
# print(buoy_ls)
# print(Sys.time())

#-------------------------------------------------------------------------------------------
#-------------------------------------------------------------------------------------------

## download ndbc and ncei datasets from ndbc and ncei websites
print("starting data download...")
start_time <- Sys.time()
print(Sys.time())
## Be aware that NDBC changes 'Dxx' extensions in NCEI NetCDF data! The code is currently set to test for 'D'
extensions between 1-20.
for (buoys in buoy_ls){
        print(buoys)
        source(paste0(data_dir,"download_data_1.R"))
        download_data_1(buoys, start_year = start_date, data_dir)
        rm(download_data_1, read_NetCDF_post_2011)
}
print(Sys.time())
end_time <- Sys.time()
```

```r
print("finished data download")

#-------------------------------------------------------------------------------

## look for empty folders in each data directory and delete

## ndbc unzipped folder
# Get vector of all folder names
dirlist <- list.dirs(unzip_ndbc_dir)
length(dirlist)
# Extract vector of empty folder names
empty_dir <- dirlist[sapply(dirlist, function(x) length(list.files(x))==0)]
length(empty_dir)
print(empty_dir)
# Remove empty folders
unlink(empty_dir, recursive=TRUE, force=FALSE)
# Get vector of all file names
ff <- dir(unzip_ndbc_dir, recursive=TRUE, full.names=TRUE)
length(ff)
# Extract vector of empty files' names
eff <- ff[file.info(ff)[["size"]]==0]
length(eff)
print(eff)
# Remove empty files
unlink(eff, recursive=TRUE, force=FALSE)
rm(dirlist,empty_dir,ff, eff)

## ndbc zipped folder
# Get vector of all folder names
dirlist <- list.dirs(zip_ndbc_dir)
length(dirlist)
# Extract vector of empty folder names
empty_dir <- dirlist[sapply(dirlist, function(x) length(list.files(x))==0)]
length(empty_dir)
print(empty_dir)
# Remove empty folders
unlink(empty_dir, recursive=TRUE, force=FALSE)
# Get vector of all file names
ff <- dir(zip_ndbc_dir, recursive=TRUE, full.names=TRUE)
length(ff)
# Extract vector of empty files' names
eff <- ff[file.info(ff)[["size"]]==0]
length(eff)
print(eff)
# Remove empty files
unlink(eff, recursive=TRUE, force=FALSE)
rm(dirlist,empty_dir,ff, eff)

##-------------------------------------------------------------------------------

```

```r
## ncei netCDF folder
# Get vector of all folder names
dirlist <- list.dirs(netCDF_ncei_dir)
length(dirlist)
# Extract vector of empty folder names
empty_dir <- dirlist[sapply(dirlist, function(x) length(list.files(x))==0)]
length(empty_dir)
print(empty_dir)
# Remove empty folders
unlink(empty_dir, recursive=TRUE, force=FALSE)
# Get vector of all file names
ff <- dir(netCDF_ncei_dir, recursive=TRUE, full.names=TRUE)
length(ff)
# Extract vector of empty files' names
eff <- ff[file.info(ff)[["size"]]==0]
length(eff)
print(eff)
# Remove empty files
unlink(eff, recursive=TRUE, force=FALSE)
rm(dirlist,empty_dir,ff, eff)

## ncei ASCII folder
# Get vector of all folder names
dirlist <- list.dirs(ascii_ncei_dir)
length(dirlist)
# Extract vector of empty folder names
empty_dir <- dirlist[sapply(dirlist, function(x) length(list.files(x))==0)]
length(empty_dir)
print(empty_dir)
# Remove empty folders
unlink(empty_dir, recursive=TRUE, force=FALSE)
# Get vector of all file names
ff <- dir(ascii_ncei_dir, recursive=TRUE, full.names=TRUE)
length(ff)
# Extract vector of empty files' names
eff <- ff[file.info(ff)[["size"]]==0]
length(eff)
print(eff)
# Remove empty files
unlink(eff, recursive=TRUE, force=FALSE)
rm(dirlist,empty_dir,ff, eff)

#-----------------------------------------------------------------------------------------
#-----------------------------------------------------------------------------------------

## If this is an annual run, merge the previous and updated metadata sheets for use in concat step.
print("starting merge metadata...")
start_time <- Sys.time()
print(Sys.time())
source(paste0(data_dir,"merge_metadata_1a.R"))
```

```r
merge_metadata_1a(data_dir)
rm(merge_metadata_1a)
print(Sys.time())
end_time <- Sys.time()
print("finished merge metadata")

#-------------------------------------------------------------------------------------------
#-------------------------------------------------------------------------------------------

## concatenate ndbc and ncei datasets from downloaded data
print("starting data concat...")
start_time <- Sys.time()
print(Sys.time())
for (buoys in buoy_ls){
     source(paste0(data_dir,"concat_data_2.R"))
     concat_data_2(buoys, start_year = start_date, data_dir)
     rm(concat_data_2,concat_ndbc, concat_ncei)
     rm_ls = ls(pattern = paste0("s_", buoys))
     rm(list = rm_ls)
     rm(rm_ls)
}
print(Sys.time())
end_time <- Sys.time()
print("finished data concat")

#-------------------------------------------------------------------------------------------
#-------------------------------------------------------------------------------------------

## ** update NDBC google spreadsheets downloads **
## Download latest NDBC metadata sheets (download folders)

## verify ncei netcdf metadata with NDBC Google spreadsheets
print("starting ncei metadata verification...")
start_time <- Sys.time()
print(Sys.time())
for (buoys in buoy_ls){
     source(paste0(data_dir,"verify_netcdf_3.R"))
     verify_netcdf_3(buoys, data_dir)
     rm(verify_netcdf_3)
}
print(Sys.time())
end_time <- Sys.time()
print("finished ncei metadata verification")

#-------------------------------------------------------------------------------------------
#-------------------------------------------------------------------------------------------

## geoclean the ndbc and ncei data
print("starting data geoClean...")
start_time <- Sys.time()
```

```
print(Sys.time())
for (buoys in buoy_ls){
     source(paste0(data_dir,"geoClean_data_4.R"))
     geoClean_data_4(buoys, data_dir)
     rm(geoClean_data_4, plot_stdmet, plots_spec)
}
print(Sys.time())
end_time <- Sys.time()
print("finished data geoClean")

#-------------------------------------------------------------------------------------------
#-------------------------------------------------------------------------------------------

## create best available ndbc dataset with ncei metadata
print("starting best data...")
start_time <- Sys.time()
print(Sys.time())
for (buoys in buoy_ls){
     source(paste0(data_dir,"create_best_data_5.R"))
     create_best_data_5(buoys, data_dir)
     rm(create_best_data_5)
}
print(Sys.time())
end_time <- Sys.time()
print("finished best data")

#-------------------------------------------------------------------------------------------
#-------------------------------------------------------------------------------------------

# build CHL Thredds NDBC netcdf file
start_time <- Sys.time()
print(Sys.time())
for (buoys in buoy_ls){
     source(paste0(data_dir,"build_thredds_netcdf_6.R"))
     build_thredds_netcdf_6(buoys, data_dir)
     rm(build_thredds_netcdf_6)
}
print(Sys.time())
end_time <- Sys.time()

#-------------------------------------------------------------------------------------------
#-------------------------------------------------------------------------------------------
#-------------------------------------------------------------------------------------------


```

US Army Corps
of Engineers®

download_data_1.R

```r
download_data_1 <- function(buoys = "list of buoys", start_year = "start year", data_dir = "data_dir"){

        ##-------------------------------------------------------------------------------------------
        ## script to download NDBC web files and NCEI netcdf files
        ## Hall 12/28/2019
        ##-------------------------------------------------------------------------------------------

        ## Actions:
        ## 1.   Sets data locations
        ## 2.   Lists buoy ID's for downloading
        ## 3.   Finds current end date for monthly and yearly data downloads
        ## 4.   Downloads NDBC website historical data from the NDBC website
        ## 5.   Unzips NDBC files
        ## 6.   Downloads NDBC NetCDF data from the NCEI website
        ## 7.   Calls 'read_NetCDF_post_2011.R and read_NetCDF_pre_2011.R' to extract NetCDF files to ASCII
        ## 8.   Attaches datetime, lat and lon to all datasets - including gps/latitude and gps/longitude = repeated
        to confirm concat accuracy
        ## 9.   Calls 'read_NetCDF_metadata.R' script to extract all netCDF metadata from global and variable
        attributes

        #-------------------------------------------------------------------------------------------
        #-------------------------------------------------------------------------------------------

        # load libraries (local run)
        library(lubridate)
        library(R.utils)
        # library("ncdf4")
        library(qpcR)
        library(plyr)
        library(dplyr)
        library(ncdf4)

        ## load libraries (HPC run)
        # library(lubridate, lib="/p/home/candice/Rlibs/")
        # library(R.utils, lib="/p/home/candice/Rlibs/")
        # library("ncdf4", lib="/p/home/candice/Rlibs/")
        # library(qpcR, lib="/p/home/candice/Rlibs/")
        # library(plyr, lib="/p/home/candice/Rlibs/")
        # library(dplyr, lib="/p/home/candice/Rlibs/")
        # library(ncdf4, lib="/p/home/candice/Rlibs/")

        ##-------------------------------------------------------------------------------------------
        ## set paths
        ##-------------------------------------------------------------------------------------------
        setwd(data_dir)
        # create new folders for data
        if (!file.exists(paste0(data_dir,"raw_data/"))) {dir.create((paste0(data_dir,"raw_data/")))}
        input_dir <- paste0(data_dir,"raw_data/")
        ## set new output directories for raw and zipped datasets
        # ndbc
```

```r
if (!file.exists(paste0(input_dir,"ndbc/"))) {dir.create((paste0(input_dir,"ndbc/")))}
ndbc_dir <- paste0(input_dir,"ndbc/")
if (!file.exists(paste0(ndbc_dir,"zipped/"))) {dir.create((paste0(ndbc_dir,"zipped/")))}
zip_ndbc_dir <- paste0(ndbc_dir,"zipped/")
if (!file.exists(paste0(ndbc_dir,"unzipped/"))) {dir.create((paste0(ndbc_dir,"unzipped/")))}
unzip_ndbc_dir <- paste0(ndbc_dir,"unzipped/")
# ncei
if (!file.exists(paste0(input_dir,"ncei/"))) {dir.create((paste0(input_dir,"ncei/")))}
ncei_dir <- paste0(input_dir,"ncei/")
if (!file.exists(paste0(ncei_dir,"netCDF/"))) {dir.create((paste0(ncei_dir,"netCDF/")))}
netCDF_ncei_dir <- paste0(ncei_dir,"netCDF/")
if (!file.exists(paste0(ncei_dir,"ascii/"))) {dir.create((paste0(ncei_dir,"ascii/")))}
ascii_ncei_dir <- paste0(ncei_dir,"ascii/")
if (!file.exists(paste0(ncei_dir,"metadata/"))) {dir.create((paste0(ncei_dir,"metadata/")))}
metadata_ncei_dir <- paste0(ncei_dir,"metadata/")

##---------------------------------------------------------------------------------------------
## set buoy stations for downloading (for stand-alone use)
##---------------------------------------------------------------------------------------------

# list_ndbc <- read.csv(paste0(data_dir,"NDBC_buoys.csv"),header = TRUE)
# list_ndbc <- dplyr::filter(list_ndbc, list_ndbc$owner == "NDBC")
# list_ndbc_buoy <- as.character(list_ndbc$station)
#
# buoys <- list_ndbc_buoy
# rm(list_ndbc)

#---------------------------------------------------------------------------------------------
#---------------------------------------------------------------------------------------------

# downloading NDBC website historical data

#---------------------------------------------------------------------------------------------
## sample web addresses for all data related to each buoy station

## yearly Standard meteorological data:
https://www.ndbc.noaa.gov/data/historical/stdmet/44014h1990.txt.gz
## yearly Spectral wave density data:
https://www.ndbc.noaa.gov/data/historical/swden/44014w1996.txt.gz
## yearly Spectral wave (alpha1) direction data:
https://www.ndbc.noaa.gov/data/historical/swdir/44014d1996.txt.gz
## yearly Spectral wave (alpha2) direction data:
https://www.ndbc.noaa.gov/data/historical/swdir2/44014i1998.txt.gz
## yearly Spectral wave (r1) direction data:
https://www.ndbc.noaa.gov/data/historical/swr1/44014j1998.txt.gz
## yearly Spectral wave (r2) direction data:
https://www.ndbc.noaa.gov/data/historical/swr2/44014k1998.txt.gz

## monthly Standard meteorological data:
https://www.ndbc.noaa.gov/data/stdmet/Jan/4401412019.txt.gz
```

```
## monthly Spectral wave density data:
https://www.ndbc.noaa.gov/data/swden/Jan/4401412019.txt.gz
## monthly Spectral wave (alpha1) direction data:
https://www.ndbc.noaa.gov/data/swdir/Jan/4401412019.txt.gz
## monthly Spectral wave (alpha2) direction data:
https://www.ndbc.noaa.gov/data/swdir2/Jan/4401412019.txt.gz
## monthly Spectral wave (r1) direction data:
https://www.ndbc.noaa.gov/data/swr1/Jan/4401412019.txt.gz
## monthly Spectral wave (r2) direction data:
https://www.ndbc.noaa.gov/data/swr2/Jan/4401412019.txt.gz

##-------------------------------------------------------------------------------------------
## set lists of downloading parameters
##-------------------------------------------------------------------------------------------

# end date for monthly and yearly data downloads
library(lubridate)
if(month(Sys.Date())== 1 | month(Sys.Date())== 2){
        current_year <- as.numeric(unlist(strsplit(as.character(Sys.Date()),"-"))[1])-1
}else{
        current_year <- as.numeric(unlist(strsplit(as.character(Sys.Date()),"-"))[1])
}
# yearly files
start_year <- as.numeric(start_year)
years <- seq(start_year,current_year,1)
month <- month.abb[c(1:12)]

# dataset types (Standard meteorological, Spectral wave density, Spectral wave (alpha1) direction,
#                Spectral wave (alpha2) direction, Spectral wave (r1) direction, Spectral wave (r2) direction)
dataTypes <- c("stdmet", "swden", "swdir", "swdir2", "swr1","swr2")
dataType_ab <- c("h", "w", "d", "i","j","k")

##-------------------------------------------------------------------------------------------
## download code
##-------------------------------------------------------------------------------------------

# looping through the listed buoy ID's
for (buoy in buoys){
        print(paste0("Starting NDBC download: ",buoy))

        # # start writing to an output file
        # sink(paste0(data_dir,"0_ndbc_download_buoy_",buoy,"_",Sys.Date(),".txt"))
        # print(paste0("Starting NDBC download: ",buoy))

        ## create buoy specific download folder
        if (!file.exists(paste0(zip_ndbc_dir,buoy,"/"))) {dir.create((paste0(zip_ndbc_dir,buoy,"/")))}
        ## for each dataset type, set the data type
        for (dataT in dataTypes){
                # dataT <- dataTypes[1]
                ab <- dataType_ab[match(dataT,dataTypes)]
```

```
                        ## downloading yearly files
                        ## https://www.ndbc.noaa.gov/data/historical/stdmet/44014h1990.txt.gz
                        ## https://www.ndbc.noaa.gov/data/historical/swden/44014wb1997.txt.gz
                        for (year in years){
                                # year <- years[8]
                                # URL website
                                fileUrl1 <-
                                paste0("https://www.ndbc.noaa.gov/data/historical/",dataT,"/",buoy,ab,year,".txt.gz")
                                fileUrl2 <-
                                paste0("https://www.ndbc.noaa.gov/data/historical/",dataT,"/",buoy,ab,"b",year,".txt.gz
                                ")
                                #https://www.ndbc.noaa.gov/data/historical/stdmet/44014h2001.txt.gz
                                print(fileUrl1)
                                 # set download location and file name
                                destfile = paste0(zip_ndbc_dir,buoy,"/",buoy,ab,year,"_",dataT,".txt.gz")
                                destfile2 = paste0(zip_ndbc_dir,buoy,"/",buoy,ab,"b",year,"_",dataT,".txt.gz")
                                # Download data from the website
                                if (!file.exists(destfile)) {
                                        try(download.file(fileUrl1, mode = "wb", destfile = destfile))
                                        print(destfile)
                                } else {print("Data are already downloaded: yearly")}
                                if (!file.exists(destfile2)) {
                                        try(download.file(fileUrl2, mode = "wb", destfile = destfile2))
                                        print(destfile2)
                                } else {print("Data are already downloaded: yearly")}
                        }
                        library(lubridate)
                        if(month(Sys.Date()) == 1) {
                                year1 <- current_year-1
                                ## downloading monthly files of the current year
                                ## https://www.ndbc.noaa.gov/data/stdmet/Jan/4401412019.txt.gz
                                for(m in month){
                                        # m <- month[11]
                                        num_mth <- match(m, month.abb)
                                        print(m)
                                        # URL website
                                        fileUrl2 <-
                                        paste0("https://www.ndbc.noaa.gov/data/",dataT,"/",m,"/",buoy,num_mth,year1,".t
                                        xt.gz") # normal monthly NDBC file name
                                        fileUrl3 <-
                                        paste0("https://www.ndbc.noaa.gov/data/",dataT,"/",m,"/",buoy,"a",year1,".txt.g
                                        z") # new name??
                                        fileUrl4 <-
                                        paste0("https://www.ndbc.noaa.gov/data/",dataT,"/",m,"/",buoy,".txt") #
                                        another new name???
                                        print(fileUrl2)
                                        # set download location and file name
                                        # stdmet/Jan/4401412019.txt.gz
                                        destfile2 =
                                        paste0(zip_ndbc_dir,buoy,"/",buoy,ab,year1,"_",match(m,month.abb),"_",dataT,".t
```

```
                                xt.gz")
177                             # Download data from the website
178                             if (!file.exists(destfile2)) {
179                                     try(download.file(fileUrl2, mode = "wb", destfile = destfile2))
180                                     print(destfile2)
181                             } else {print("Data are already downloaded: fileUrl2")}
182                             if (!file.exists(destfile2)) {
183                                     try(download.file(fileUrl3, mode = "wb", destfile = destfile2))
184                                     print(destfile2)
185                             } else {print("Data are already downloaded: fileUrl3")}
186                             if (!file.exists(destfile2)) {
187                                     try(download.file(fileUrl4, mode = "wb", destfile = destfile2))
188                                     print(destfile2)
189                             } else {print("Data are already downloaded: fileUrl4")}
190                     }
191             }else{  year1 <- current_year
192                     ## downloading monthly files of the current year
193                     ## https://www.ndbc.noaa.gov/data/stdmet/Jan/4401412019.txt.gz
194                     for(m in month){
195                             # m <- month[11]
196                             num_mth <- match(m, month.abb)
197                             print(m)
198                             # URL website
199                             fileUrl2 <-
                                paste0("https://www.ndbc.noaa.gov/data/",dataT,"/",m,"/",buoy,num_mth,year1,".t
                                xt.gz") # normal monthly NDBC file name
200                             fileUrl3 <-
                                paste0("https://www.ndbc.noaa.gov/data/",dataT,"/",m,"/",buoy,"a",year1,".txt.g
                                z") # new name??
201                             fileUrl4 <-
                                paste0("https://www.ndbc.noaa.gov/data/",dataT,"/",m,"/",buoy,".txt") #
                                another new name???
202                             print(fileUrl2)
203                             # set download location and file name
204                             # stdmet/Jan/4401412019.txt.gz
205                             destfile2 =
                                paste0(zip_ndbc_dir,buoy,"/",buoy,ab,year1,"_",match(m,month.abb),"_",dataT,".t
                                xt.gz")
206                             # Download data from the website
207                             if (!file.exists(destfile2)) {
208                                     try(download.file(fileUrl2, mode = "wb", destfile = destfile2))
209                                     print(destfile2)
210                             } else {print("Data are already downloaded: fileUrl2")}
211                             if (!file.exists(destfile2)) {
212                                     try(download.file(fileUrl3, mode = "wb", destfile = destfile2))
213                                     print(destfile2)
214                             } else {print("Data are already downloaded: fileUrl3")}
215                             if (!file.exists(destfile2)) {
216                                     try(download.file(fileUrl4, mode = "wb", destfile = destfile2))
217                                     print(destfile2)
```

```
                                        } else {print("Data are already downloaded: fileUrl4")}

                                }
                        }
                }
                print(paste0("Finished NDBC download: ",buoy))
                # end writing to an output file
                # sink()
                # print(paste0("Finished NDBC download: ",buoy))
        }

        ## Different NDBC formats examples...
        # https://www.ndbc.noaa.gov/data/stdmet/Oct/44014a2019.txt.gz
        # https://www.ndbc.noaa.gov/data/stdmet/Nov/44014.txt
        # https://www.ndbc.noaa.gov/data/stdmet/Oct/44014a2019.txt.gz
        # https://www.ndbc.noaa.gov/data/swr2/Oct/44014a2019.txt.gz

        ##-------------------------------------------------------------------------------------------
        ## copy and unzip datafiles code
        ##-------------------------------------------------------------------------------------------

        library(R.utils)
        # copy files to new folder and unzip
        for (buoy in buoys){
                print(paste0("Starting NDBC unzip: ",buoy))

                # copy files to new folder for unzipping
                list.zip <- list.files(path = paste0(zip_ndbc_dir,buoy,"/"), pattern = ".gz", full.names = TRUE)

                if(length(list.zip)> 0){

                        # start writing to an output file
                        # sink(paste0(data_dir,"0_ndbc_unzip_buoy_",buoy,"_",Sys.Date(),".txt"))
                        # print(paste0("Starting NDBC unzip: ",buoy))

                        ## create buoy specific unzip folder
                        if (!file.exists(paste0(unzip_ndbc_dir,buoy,"/")))
                        {dir.create((paste0(unzip_ndbc_dir,buoy,"/")))}

                        # copy files to new folder for unzipping
                        list.zip <- list.files(path = paste0(zip_ndbc_dir,buoy,"/"), pattern = ".gz", full.names =
                        TRUE)
                        list.unzipped <- list.files(path = paste0(unzip_ndbc_dir,buoy,"/"), pattern = ".txt",
                        full.names = FALSE)
                        # copy new files over
                        for (file in list.zip){
                                file_present <- gsub(".gz","",unlist(strsplit(file,paste0(zip_ndbc_dir,buoy,"/")))[2])
                                if(file_present %in% list.unzipped){
                                        print("file already exists")
                                }else{
```

```
                                        file.copy(file, paste0(unzip_ndbc_dir,buoy,"/"))
                                }
                        }
                        # unzip files in unzipped folder
                        list.unzip <- list.files(path = paste0(unzip_ndbc_dir,buoy,"/"), pattern = ".gz", full.names
                        = TRUE)
                        for(i in list.unzip){
                                print(i)
                                tryCatch({
                                        gunzip(i, remove = TRUE)
                                }, error = function(e){
                                        print("already unzipped")
                                        unlink(i)
                                })
                        }

                        print(paste0("Finished NDBC upzip: ",buoy))
                        # end writing to an output file
                        # sink()
                }else{print("no ndbc data to unzip")}

                print(paste0("Finished NDBC unzip: ",buoy))

        }

        #-------------------------------------------------------------------------------------------
        #-------------------------------------------------------------------------------------------

        # downloading NCEI NDBC NetCDF data

        #-------------------------------------------------------------------------------------------

        # ncei website

        ## sample web addresses for all data related to each buoy station

        ## ftp://ftp.nodc.noaa.gov/pub/f291/200301/44014_200301.Z
        ## https://data.nodc.noaa.gov/thredds/fileServer/ndbc/cmanwx/2011/01/NDBC_44014_201101_D1_v00.nc
        ## https://data.nodc.noaa.gov/thredds/fileServer/ndbc/cmanwx/2019/01/NDBC_44014_201901_D5_v00.nc

        ## https://www.ncei.noaa.gov/data/oceans/ndbc/cmanwx/2020/03/NDBC_41002_202003_D7_v00.nc
        ## https://www.ncei.noaa.gov/data/oceans/ndbc/cmanwx/2020/04/NDBC_41001_202004_D4_v00.nc

        ## NCEI NODC THREDDS NDBC CMANWX server
        #https://data.nodc.noaa.gov/thredds/fileServer/ndbc/cmanwx/2006/12/44014_200612.nc

        ##-------------------------------------------------------------------------------------------
        ## download code
        ##-------------------------------------------------------------------------------------------
        library(lubridate)
```

```
314
315         # looping through the listed buoy ID's
316         for (buoy in buoys){
317
318                 print(paste0("Starting NCEI download: ",buoy))
319
320                 # # start writing to an output file
321                 # sink(paste0(data_dir,"0_ncei_download_buoy_",buoy,"_",Sys.Date(),".txt"))
322                 # print(paste0("Starting NCEI download: ",buoy))
323
324                 ## create buoy specific download folder
325                 if (!file.exists(paste0(netCDF_ncei_dir,buoy,"/"))) {dir.create((paste0(netCDF_ncei_dir,buoy,"/")))}
326
327                 ## downloading netCDF year_month files
328                 ## old format for pre and post 2011 files - as of Apr 2020
329                 ## https://data.nodc.noaa.gov/thredds/fileServer/ndbc/cmanwx/2006/12/44014_200612.nc
330                 ## https://data.nodc.noaa.gov/thredds/fileServer/ndbc/cmanwx/2011/01/NDBC_44014_201101_D1_v00.nc
331
332                 ## new format for pre and post 2011 files - as of Apr 2020
333                 ## https://www.ncei.noaa.gov/data/oceans/ndbc/cmanwx/1980/01/41001_198001.nc
334                 ## https://www.ncei.noaa.gov/data/oceans/ndbc/cmanwx/2020/04/NDBC_41001_202004_D4_v00.nc
335
336                 for(year in years){
337                         # year <- 2020
338                         for(m in month){
339                                 # m <- month[7]
340                                 num_mth <- match(m, month.abb)
341                                 print(m)
342
                                    ##-------------------------------------------------------------------------------
                                    ---
343                                 # # for pre-2011 F291 file formats (no longer available but retaining code for
                                    historical value)
344                                 # if(year <2011){
345                                 #         # URL website
346                                 #         # ftp://ftp.nodc.noaa.gov/pub/f291/201001/44014_201001.Z
347                                 #         fileUrl1 <- paste0("ftp://ftp.nodc.noaa.gov/pub/f291/",year,sprintf("%02d",
                                    num_mth),"/",buoy,"_",year,sprintf("%02d", num_mth),".Z") # pre-2011
348                                 #         print(fileUrl1)
349                                 #         # set download location and file name
350                                 #         destfile1 = paste0(netCDF_ncei_dir,buoy,"/",buoy,"_",year,sprintf("%02d",
                                    num_mth),".Z")
351                                 #
352                                 #         # Download data from the website
353                                 #         if (!file.exists(destfile1)) {
354                                 #                 try(download.file(fileUrl1, mode = "wb", destfile = destfile1))
355                                 #                 print(destfile1)
356                                 #         } else {print("Data are already downloaded: fileUrl4")}
357                                 # }
358
```

```
                                ##------------------------------------------------------------------------------
                                ---
359                             # # post 2011
360                             if(year >= 2011){
361                                     for(d in 1:20){
362                                             # print(d)
363                                             # fileUrl1 <-
                                                paste0("https://data.nodc.noaa.gov/thredds/fileServer/ndbc/cmanwx/",yea
                                                r,"/",sprintf("%02d", num_mth),"/NDBC_",buoy,"_",year,sprintf("%02d",
                                                num_mth),"_D",d,"_v00.nc") # post 2011
364                                             fileUrl1 <-
                                                paste0("https://www.ncei.noaa.gov/data/oceans/ndbc/cmanwx/",year,"/",sp
                                                rintf("%02d", num_mth),"/NDBC_",buoy,"_",year,sprintf("%02d",
                                                num_mth),"_D",d,"_v00.nc") # post 2011
365                                             print(fileUrl1)
366                                             destfile1 =
                                                paste0(netCDF_ncei_dir,buoy,"/",buoy,"_",year,sprintf("%02d",
                                                num_mth),"_D",d,".nc")
367                                             print(destfile1)
368                                             # Download data from the website
369                                             if (!file.exists(destfile1)) {
370                                                     try(download.file(fileUrl1, mode = "wb", destfile = destfile1))
371                                                     print(destfile1)
372                                             } else {print("Post 2011 data are already downloaded")}
373                                     }
374                             }
375                             # pre 2011
376                             if(year < 2011){
377                                     # fileUrl2 <-
                                        paste0("https://data.nodc.noaa.gov/thredds/fileServer/ndbc/cmanwx/",year,"/",sp
                                        rintf("%02d", num_mth),"/",buoy,"_",year,sprintf("%02d", num_mth),".nc") #
                                        post 2011
378                                     fileUrl2 <-
                                        paste0("https://www.ncei.noaa.gov/data/oceans/ndbc/cmanwx/",year,"/",sprintf("%
                                        02d", num_mth),"/",buoy,"_",year,sprintf("%02d", num_mth),".nc") # post 2011
379                                     destfile2 = paste0(netCDF_ncei_dir,buoy,"/",buoy,"_",year,sprintf("%02d",
                                        num_mth),".nc")
380                                     print(destfile2)
381                                     if (!file.exists(destfile2)) {
382                                             try(download.file(fileUrl2, mode = "wb", destfile = destfile2))
383                                             print(destfile2)
384                                     } else {print("Pre 2011 data are already downloaded")}
385                             }
386                     }
387             }
388             print(paste0("Finished NCEI download: ",buoy))
389             # # end writing to an output file
390             # sink()
391             # print(paste0("Finished NCEI download: ",buoy))
392     }
```

```
393
394
395         ##-------------------------------------------------------------------------------------
396         ## call 'read_NetCDF_pre_2011.R' and 'read_NetCDF_post_2011.R' scripts and extract NetCDF files to ASCII
397         ##-------------------------------------------------------------------------------------
398
399         # copy and extract files to new folder
400         print(buoys)
401
402         for (buoy in buoys){
403                 print(paste0("Starting NCEI read_NetCDF: ",buoy))
404
405                 # # find spectral frequencies used over time
406                 list.zip <- list.files(path = paste0(netCDF_ncei_dir,buoy,"/"), full.names = TRUE)
407
408                 if(length(list.zip) > 0){
409                         # # start writing to an output file
410                         # sink(paste0(data_dir,"0_ncei_readNetCDF_buoy_",buoy,"_",Sys.Date(),".txt"))
411                         # print(paste0("Starting NCEI read_NetCDF: ",buoy))
412
413                         library("ncdf4")
414                         library(qpcR)
415                         # concat all spectral wave frequencies used over time
416                         wave_wpm_available_ALL <- data.frame(matrix(NA, nrow = 0, ncol = 48))
417                         spec_r_type_ALL <- data.frame(matrix(NA, nrow = 0, ncol = 48))
418
419                         for(file in list.zip){
420                                 # file <- list.zip[1]
421                                 tryCatch({
422                                         nc <- nc_open(file)                             # Reading the netcdf data
423                                         print(nc$filename)
424                                         # extract data
425                                         time <- data.frame(ncvar_get(nc, "time"), stringsAsFactors = FALSE)
426                                         # wave wpm check
427                                         tryCatch({
428                                                 wave_wpm <- ncvar_get(nc, "wave_wpm")
429                                                 # human readable time
430                                                 time[,1] <- as.POSIXct(time[,1],origin = "1970-01-01",tz = "uct") #
                                                    seconds since 1970-01-01 00:00:00 UTC"
431                                                 colnames(time) <- c("time")
432                                                 # save wave spectral frequency bands
433                                                 wave_wpm_available <-
                                                    t(data.frame(c(as.character(time[1,1]),signif(wave_wpm,6)),
                                                    stringsAsFactors = FALSE))
434                                                 library(plyr)
435                                                 wave_wpm_available_ALL <- qpcR:::rbind.na(wave_wpm_available_ALL,
                                                    wave_wpm_available)
436                                                 rownames(wave_wpm_available_ALL) <- NULL
437                                                 rm(wave_wpm_available)
438                                         }, error = function(e){print("no wave_wmp")}
```

```
                                        )

                                        # r data check
                                        dat_list <- NULL

                                        for (i in 1:nc$nvars){
                                                j<-nc$var[[i]]$name
                                                # print(j)
                                                dat_list <- c(dat_list,j)
                                        }
                                        # find r1 data
                                        if(any(grepl("*/r1$",dat_list))){
                                                r_num <- grep("*/r1$",dat_list)[1]
                                                r_dat <- data.frame(t(ncvar_get(nc, dat_list[r_num])),
                                                stringsAsFactors = FALSE)
                                                # human readable time
                                                time[,1] <- as.POSIXct(time[,1],origin = "1970-01-01",tz = "uct") #
                                                seconds since 1970-01-01 00:00:00 UTC"
                                                colnames(time) <- c("time")
                                                # save wave spectral frequency bands
                                                r_available <- data.frame(c(as.character(time[1,1]), r_dat[1,]),
                                                stringsAsFactors = FALSE)
                                                library(plyr)
                                                spec_r_type_ALL <- qpcR:::rbind.na(spec_r_type_ALL, r_available)
                                                rownames(spec_r_type_ALL) <- NULL
                                                rm(r_available)
                                        }else{print(paste0("has no R: ",file))}
                                        nc_close(nc)
                                        rm(nc,time,dat_list, j)
                                }, error = function(e){print(paste0("can't open: ",file))}
                                )
                        }
                        write.csv(wave_wpm_available_ALL, paste0(ascii_ncei_dir,buoy,"_spec_freq_available_ALL.csv"),
                        row.names=FALSE, na = "NaN")
                        rm(wave_wpm_available_ALL)
                        write.csv(spec_r_type_ALL, paste0(ascii_ncei_dir,buoy,"_spec_r_type_ALL.csv"),
                        row.names=FALSE, na = "NaN")
                        rm(spec_r_type_ALL)

                        ## 2011 onward NetCDF format

                        ## create buoy specific unzip folder
                        if (!file.exists(paste0(ascii_ncei_dir,buoy,"/")))
                        {dir.create((paste0(ascii_ncei_dir,buoy,"/")))}
                        print(paste0("working on post-2011: ", file))

                        # copy files to new folder for unzipping
                        list.zip <- list.files(path = paste0(netCDF_ncei_dir,buoy,"/"), pattern = "_D", full.names =
                        TRUE)
```

```
482                     if(length(list.zip)>0){
483
484                             list.extracted <- list.files(path = paste0(ascii_ncei_dir,buoy,"/"), full.names =
                                FALSE)
485
486                             # copy new files over
487                             for (file in list.zip){
488                                     # file <- list.zip[115]
489                                     file_present <-
                                        gsub(".nc","",unlist(strsplit(file,paste0(netCDF_ncei_dir,buoy,"/")))[2])
490                                     if(file_present %in% list.extracted){
491                                             print("file already exists")
492                                     }else{
493                                             file.copy(file, paste0(ascii_ncei_dir,buoy,"/"))
494                                             print(paste0("copying file over: ", file))
495                                     }
496                             }
497                             rm(file_present)
498
499                             # Use read_NetCDF_post_2011.R script and extract NetCDF files to ASCII
500                             list.zip <- list.files(path = paste0(ascii_ncei_dir,buoy,"/"), pattern = ".nc",
                                full.names = TRUE)
501                             source(paste0(data_dir,"read_NetCDF_post_2011.R"))
502                             for(file in list.zip){
503                                     # file <- list.zip[1] # watch for missing time_10second, release flags for 10
                                        second data that doesn't
504                                     file
505                                     read_NetCDF_post_2011(file, buoy, ascii_ncei_dir)
506                             }
507
508                             # remove files if extracted
509                             ext_files <- list.files(path = paste0(ascii_ncei_dir,buoy,"/"), pattern = ".nc",
                                full.names = FALSE)
510                             for (e in ext_files){
511                                     # e <- ext_files[1]
512                                     e_full <- paste0(ascii_ncei_dir,buoy,"/",e)
513                                     destfile <- paste0(ascii_ncei_dir,buoy,"/",gsub(".nc","",e))
514                                     if(file.exists(destfile)){file.remove(e_full)}
515                             }
516                            print(paste0("finished post-2011: ", file))
517
518                     }else{print("no available post-2011 files")}
519
520
521                     ## pre-2011 NetCDF format
522
523                     # reset directory
524                     setwd(data_dir)
525                     print(paste0("working on pre-2011: ", file))
526
```

```
# # copy files to new folder for unzipping
# list.zip <- list.files(path = paste0(netCDF_ncei_dir,buoy,"/"), pattern = buoy, full.names = TRUE)
list.netcdf <- list.files(path = paste0(netCDF_ncei_dir,buoy,"/"), full.names = TRUE)
list.ascii <-  list.files(path = paste0(ascii_ncei_dir,buoy,"/"), full.names = TRUE)

if (start_year <= 2010){
        dates <- c(1970:2010)
        ## find the files for this data setup in netcdf folders
        files <- list()
        for (i in 1:length(dates)){
                files[[i]] <- list.netcdf[grepl(paste0("_",dates[i]),list.netcdf)]
        }
        list.netcdf <- unlist(files)
        ## find the files for this data setup in ascii folders
        files <- list()
        for (i in 1:length(dates)){
                files[[i]] <- list.ascii[grepl(paste0("_",dates[i]),list.ascii)]
        }
        list.ascii <- unlist(files)
        # look for extracted files
        list.extracted <- NULL
        for(l in list.ascii){
                # l <- list.ascii[1]
                stripped <- unlist(strsplit(l, paste0("/",buoy,"/")))[2]
                list.extracted <- c(list.extracted,stripped)
        }
        # copy new files over
        for (file in list.netcdf){
                # file <- list.netcdf[1]
                file_present <-
                gsub(".nc","",unlist(strsplit(file,paste0(netCDF_ncei_dir,buoy,"/")))[2])
                if(file_present %in% list.extracted){
                        print("file already exists")
                }else{
                        file.copy(file, paste0(ascii_ncei_dir,buoy,"/"))
                        print(paste0("copying file over: ", file))
                }
        }

        # Use 'read_NetCDF_pre_2011.R' script and extract NetCDF files to ASCII
        source(paste0(data_dir,"read_NetCDF_pre_2011.R"))
        files <-  list.files(path = paste0(ascii_ncei_dir,buoy,"/"), pattern = ".nc",
        full.names = TRUE)

        for(file in files){
                # file <- files[202]
                file
                read_NetCDF_pre_2011(file, buoy, ncei_dir, ascii_ncei_dir)
        }
```

```
                                # remove files if extracted
                                ext_files <- list.files(path = paste0(ascii_ncei_dir,buoy,"/"), pattern = ".nc",
                                full.names = FALSE)
                                for (e in ext_files){
                                        # e <- ext_files[1]
                                        e_full <- paste0(ascii_ncei_dir,buoy,"/",e)
                                        destfile <- paste0(ascii_ncei_dir,buoy,"/", unlist(strsplit(e, ".nc"))[1])
                                        destfile2 <- paste0(ascii_ncei_dir,buoy,"/", unlist(strsplit(e, "_D"))[1])
                                        if(file.exists(destfile)){try(file.remove(e_full))}
                                        if(file.exists(destfile2)){try(file.remove(e_full))}
                                }
                                print(paste0("finished pre-2011: ", file))

                                # buoy <- buoys[1]
                                print(paste0("Finished NCEI read_NetCDF: ",buoy))
                        }
                        # end writing to an output file
                        # sink()
                        print(paste0("Finished NCEI read_NetCDF: ",buoy))
                }else{print("empty folder - no data")}
                # # end writing to an output file
                print(paste0("Finished NCEI read_NetCDF: ",buoy))
        }
        rm(r_dat)

        #-----------------------------------------------------------------------------------------
        #-----------------------------------------------------------------------------------------
        #
        # Extract all metadata
        #
        #-----------------------------------------------------------------------------------------
        # buoys with no post-2011 data:
        # buoys <- c(41006, 41018, 41021, 41022, 41023, 42007, 42038, 42041, 42042, 42053, 42054, 42080,
        #            44004, 44028, 44070, 45011, 46003, 46023, 46030, 46045, 46051, 46062, 46063, 46079,
        #            46106, 46107, 48011, 51026, 51028)

        for (buoy in buoys){
                print(paste0("Starting NCEI metadata dump: ",buoy))
                time_pause <- 18 # seconds

                # look for data
                list.zip <- list.files(path = paste0(netCDF_ncei_dir,buoy,"/"), pattern = "_D", full.names = TRUE)

                if(length(list.zip)>0){
                        # # start writing to an output file
                        # sink(paste0(data_dir,"0_ncei_metadata_buoy_",buoy,"_",Sys.Date(),".txt"))
                        # print(paste0("Starting NCEI read_NetCDF: ",buoy))

                        # create buoy folder
                        if (!file.exists(paste0(metadata_ncei_dir,buoy,"/")))
```

```
                        {dir.create((paste0(metadata_ncei_dir,buoy,"/")))}

                        source(paste0(data_dir,"read_NetCDF_metadata.R"))
                        read_NetCDF_metadata(buoy, list.zip, netCDF_ncei_dir, metadata_ncei_dir, time_pause)

                        # # # end writing to an output file
                        # sink()
                        # print(paste0("Finished metadata: ",buoy))
                }else{
                        print("no netCDF data available for download")
                        # sink()
                        print(paste0("Finished metadata: ",buoy))
                }
                # # preventative
                # sink()
        }


        #-------------------------------------------------------------------------------------------
        #-------------------------------------------------------------------------------------------
        # clean glob environ
        rm(list = ls())
        #-------------------------------------------------------------------------------------------
        #-------------------------------------------------------------------------------------------
}
```

US Army Corps
of Engineers®

read_NetCDF_pre_2011.R

```r
read_NetCDF_pre_2011 <- function(x = "list of files", buoy = "buoy number", ncei_dir = "ncei_dir", ascii_ncei_dir = "ascii_ncei_dir"){

        print(x)
        library("ncdf4")
        ## create buoy specific extract folder
        folder_name <- gsub(".nc","",unlist(strsplit(x, paste0("/",buoy,"/")))[2])
        if (!file.exists(paste0(ascii_ncei_dir,buoy,"/",folder_name)))
        {dir.create((paste0(ascii_ncei_dir,buoy,"/",folder_name)))}
        tryCatch({
                nc <- nc_open(x)                          # Reading the netcdf data
                # identify variables in .nc file
                dat_list <- NULL
                for (i in 1:nc$nvars){
                        j<-nc$var[[i]]$name
                        # print(j)
                        dat_list <- c(dat_list,j)
                }
                # extract data
                lat <- ncvar_get(nc, "lat")
                lon <- ncvar_get(nc, "lon")
                time <- data.frame(ncvar_get(nc, "time"), stringsAsFactors = FALSE)
                # human readable time
                time[,1] <- as.POSIXct(time[,1],origin = "1970-01-01",tz = "uct") # seconds since 1970-01-01 00:00:00 UTC"
                colnames(time) <- c("time")
                # export lat / lon and time
                lat_t <- merge(time,lat)
                colnames(lat_t) <- c("time", "lat")
                lon_t <- merge(time,lon)
                colnames(lon_t) <- c("time", "lon")
                # concat time, lat, lon
                library(dplyr)
                time_position <- full_join(lat_t, lon_t, by = "time")
                # consistency with post 2011 files
                lat_t$qc_flag <- NA; lat_t$release_flag <- NA
                lon_t$qc_flag <- NA; lon_t$release_flag <- NA
                # export files
                write.csv(lat_t, paste0(ascii_ncei_dir,buoy,"/",folder_name, "/",folder_name,"_lat.csv"), row.names=FALSE, na = "NaN")
                write.csv(lon_t, paste0(ascii_ncei_dir,buoy,"/",folder_name, "/",folder_name,"_lon.csv"), row.names=FALSE, na = "NaN")
                write.csv(time, paste0(ascii_ncei_dir,buoy,"/",folder_name, "/",folder_name,"_time.csv"), row.names=FALSE, na = "NaN")
                write.csv(time_position, paste0(ascii_ncei_dir,buoy,"/",folder_name, "/",folder_name,"_time_position.csv"), row.names=FALSE, na = "NaN")
                rm(lat_t, lon_t, time)
                # extract data through variable names, attached time stamp and export
                for(j in dat_list[5: length(dat_list)]){
                        print(j)
```

```
                        dat <- data.frame(ncvar_get(nc, j), stringsAsFactors = FALSE)
                        if(dim(dat)[1]==dim(time_position)[1]){
                                dat <- cbind(time_position,dat)
                                colnames(dat) <- c("time","lat",'lon',j)
                                # to match post 2011 dataset
                                dat$qc_flag <- NA; dat$release_flag <- NA

                        }else if(dim(dat)[2] == dim(time_position)[1]){                # spectral parameters
                                if(dim(dat)[1] == 2){
                                        dat <- dat
                                }else{  dat <- data.frame(t(dat), stringsAsFactors = FALSE)
                                        dat <- cbind(time_position,dat)
                                        if("wave_wpm_bnds" %in% dat_list){
                                                wave_wpm <- ncvar_get(nc, "wave_wpm")
                                                wave_col <- c("time","lat",'lon',signif(wave_wpm,4))
                                                if(dim(dat)[2]==length(wave_col)){
                                                        colnames(dat) <- wave_col
                                                }
                                        }
                                        if("depth_bnds" %in% dat_list){
                                                depth <- ncvar_get(nc, "depth")
                                                sub_col <- c("time","lat",'lon',depth)
                                                if(dim(dat)[2]==length(sub_col)){
                                                        colnames(dat) <- sub_col
                                                }
                                        }
                                 }
                        }else{
                                dat <- dat
                        }
                        write.csv(dat, paste0(ascii_ncei_dir,buoy,"/",folder_name, "/",folder_name,"_",j,".csv"),
                        row.names=FALSE, na = "NaN")
                        rm(dat)
                        if("depth_bnds" %in% dat_list){
                                tryCatch({
                                        rm(depth)
                                }, error = function(e){
                                        print("no depth")
                                })
                        }
                }
                nc_close(nc)
                rm(nc, time_position)
        }, error = function(e){print(paste0("can't open: ",file))}
        )
}

```

US Army Corps
of Engineers®

read_NetCDF_post_2011.R

```r
read_NetCDF_post_2011 <- function(x = "list of files", buoy = "buoy number", ascii_ncei_dir = "ascii_ncei_dir"){
        library("ncdf4")
        library("stringr")
        ## create buoy specific extract folder
        folder_name <- gsub(".nc","",unlist(strsplit(x, paste0("/",buoy,"/")))[2])
        dir.create((paste0(ascii_ncei_dir,buoy,"/",folder_name)))

        tryCatch({
                nc <- nc_open(x)                              # Reading the netcdf data
                print(nc$filename)
                # extract time data
                time <- data.frame(ncvar_get(nc, "time"), stringsAsFactors = FALSE)
                # test for 10 second time field
                tryCatch({time_10min = data.frame(ncvar_get(nc, "time10"), stringsAsFactors = FALSE)
                }, error = function(e) {print("no 10-second time field")
                })
                # time_10min <- data.frame(ncvar_get(nc, "time10"), stringsAsFactors = FALSE)
                # human readable time
                time[,1] <- as.POSIXct(time[,1],origin = "1970-01-01",tz = "uct") # seconds since 1970-01-01 00:00:00
                UTC"
                colnames(time) <- c("time")
                time_list <- ls(pattern = "time")
                if("time_10min" %in% time_list){
                        time_10min[,1] <- as.POSIXct(time_10min[,1],origin = "1970-01-01",tz = "uct") # seconds since
                        1970-01-01 00:00:00 UTC"
                        colnames(time_10min) <- c("time")
                }
                # check for empty NetCDF file before continuing.
                if(nc$nvars != 0){
                        # Get a list of the nc variable names.
                        dat_list <- attributes(nc$var)$names
                        # lat and lon
                        if("gps_1/latitude" %in% dat_list){
                                lat <- cbind(time,data.frame(ncvar_get(nc, "gps_1/latitude"), stringsAsFactors =
                                FALSE),
                                             data.frame(ncvar_get(nc, "gps_1/latitude_qc"), stringsAsFactors = FALSE),
                                             data.frame(ncvar_get(nc, "gps_1/latitude_release"), stringsAsFactors =
                                             FALSE))

                                lon <- cbind(time, data.frame(ncvar_get(nc, "gps_1/longitude"), stringsAsFactors =
                                FALSE),
                                             data.frame(ncvar_get(nc, "gps_1/longitude_qc"), stringsAsFactors = FALSE),
                                             data.frame(ncvar_get(nc, "gps_1/longitude_release"), stringsAsFactors =
                                             FALSE))
                                colnames(lat) <- c("time", "lat", "qc_flag", "release_flag")
                                colnames(lon) <- c("time", "lon","qc_flag", "release_flag")
                        }else{
                                lat <- time
                                lat$lat <- NA; lat$qc_flag <- NA; lat$release_flag <- NA
                                lon <- time
```

```
        lon$lon <- NA; lon$qc_flag <- NA; lon$release_flag <- NA
}
# concat time, lat, lon
library(dplyr)
time_position <- full_join(lat[,1:2], lon[,1:2], by = "time")
# export files
write.csv(lat, paste0(ascii_ncei_dir,buoy,"/",folder_name, "/",folder_name,"_lat.csv"),
row.names=FALSE, na = "NaN")
write.csv(lon, paste0(ascii_ncei_dir,buoy,"/",folder_name, "/",folder_name,"_lon.csv"),
row.names=FALSE, na = "NaN")
write.csv(time, paste0(ascii_ncei_dir,buoy,"/",folder_name, "/",folder_name,"_time.csv"),
row.names=FALSE, na = "NaN")
write.csv(time_position, paste0(ascii_ncei_dir,buoy,"/",folder_name,
"/",folder_name,"_time_position.csv"), row.names=FALSE, na = "NaN")

if("time_10min" %in% time_list){
        time_10min_position <- full_join(time_10min, lat[,1:2], by = "time")
        time_10min_position <- full_join(time_10min_position, lon[,1:2], by = "time")
        colnames(time_10min_position) <- c("time", "lat", "lon")
        # fill in missing lat/lon
        library(tidyr)
        time_10min_position <- time_10min_position %>% fill(c(lat,lon),.direction = "downup")
        write.csv(time_10min, paste0(ascii_ncei_dir,buoy,"/",folder_name,
        "/",folder_name,"_time_10min.csv"), row.names=FALSE, na = "NaN")
        write.csv(time_10min_position, paste0(ascii_ncei_dir,buoy,"/",folder_name,
        "/",folder_name,"_time_10min_position.csv"), row.names=FALSE, na = "NaN")
        rm(time_10min)
}

rm(lat,lon, time)
# remove qc and release fields from main dat list
dat_use <- Filter(function(x) !any(grepl("_qc", x)), dat_list)
dat_use <- Filter(function(x) !any(grepl("_release", x)), dat_use)

a = 1; g = 1; b = 1; d = 1; w = 1
a_ls <- NULL; g_ls <- NULL; b_ls <- NULL; d_ls <- NULL; w_ls <- NULL
# extract data through variable names, attached time and position, and export
for(j in dat_use){
        dat <- data.frame(ncvar_get(nc, j), stringsAsFactors = FALSE)
        # find qc and release code
        if(paste0(j,"_qc") %in% dat_list){
                qc <- data.frame(ncvar_get(nc, paste0(j,"_qc")), stringsAsFactors = FALSE)
        }
        if(paste0(j,"_release") %in% dat_list){
                rl <- data.frame(ncvar_get(nc, paste0(j,"_release")), stringsAsFactors = FALSE)
        }
        # define column name
        if(!is.na(unlist(strsplit(j, "/"))[2])){
                dat_name <- unlist(strsplit(j, "/"))[2]
        }else{
```

```
89                                      dat_name <- j
90                              }
91                              # build dataframe
92                              if(dim(dat)[1]==dim(time_position)[1]){
93                                      if(paste0(j,"_qc") %in% dat_list){
94                                              if(paste0(j,"_release") %in% dat_list){
95                                                      dat <- cbind(time_position,dat, qc, rl)
96                                                      colnames(dat) <- c("time","lat",'lon',dat_name,
                                                        "qc_flag","release_flag")
97                                              }
98                                      }else{
99                                              dat <- cbind(time_position,dat)
100                                             colnames(dat) <- c("time","lat",'lon',dat_name)
101
102                                     }
103                             }
104                             if(dim(dat)[2] == dim(time_position)[1]){               # spectral parameters
105                                     if(dim(dat)[1] == 2){
106                                             dat <- dat
107                                     }else{
108                                             dat <- data.frame(t(dat), stringsAsFactors = FALSE)
109                                             dat <- cbind(time_position,dat)
110                                             tryCatch({
111                                                     wave_wpm <- ncvar_get(nc, "wave_wpm")
112                                                     colnames(dat) <- c("time","lat",'lon',signif(wave_wpm,4))
113                                             }, error = function(e){print("no wave_wpm data")})
114                                     }
115                             }
116                             if("time_10min" %in% time_list){
117                                     if(dim(dat)[1]==dim(time_10min_position)[1]){
118                                             if(dim(rl)[1] == dim(time_position)[1]){
119                                                     rl <- cbind(time_position[1], rl)
120                                                     dat <- cbind(time_10min_position,dat, qc)
121                                                     time_new <- dplyr::left_join(time_10min_position[1], rl, by =
                                                        "time")
122                                                     dat <- left_join(dat,time_new, by = "time")
123                                             }else{
124                                                     dat <- cbind(time_10min_position,dat, qc, rl)
125                                             }
126                                             colnames(dat) <- c("time","lat","lon",dat_name,
                                                "qc_flag","release_flag")
127                                     }
128                                     if(dim(dat)[1] < dim(time_10min_position)[1] & dim(dat)[1] >
                                        dim(time_position)[1]){                 # spectral parameters
129                                             if(j != "time_wpm_40_bnds"){
130                                                     if(dim(qc)[1]==dim(rl)[1]){
131                                                             dat <- cbind(dat, qc, rl)
132                                                             colnames(dat) <- c(dat_name, "qc_flag","release_flag")
133                                                     }else{
134                                                             dat <- cbind(dat, qc)
```

```
                                                colnames(dat) <- c(dat_name, "qc_flag")
                                                dat$release_flag <- NA
                                        }
                                }else{
                                        dat <- dat
                                        colnames(dat) <- dat_name                       }
                        }
                }

                # define export name and export
                dat_name <- gsub("/","_",j)
                find_dat <- unlist(strsplit(dat_name,"_"))[1]
                payload <- names(unlist(nc$fqgn2Rindex))
                # print(dat_name)
                if(find_dat == "gps"){
                        if(dat_name %in% g_ls){g <- sum(str_count(g_ls, paste0(dat_name,"$")))+1}
                        g_ls <- c(g_ls,dat_name)
                        payload_num <- unlist(payload[grepl(find_dat,payload, ignore.case = TRUE)])
                        payload_num <- unlist(strsplit(payload_num[g],paste0("/",find_dat)))[1]
                }else if(find_dat == "anemometer"){
                        if(dat_name %in% a_ls){a <- sum(str_count(a_ls, paste0(dat_name,"$")))+1}
                        a_ls <- c(a_ls,dat_name)
                        payload_num <- unlist(payload[grepl(find_dat,payload, ignore.case = TRUE)])
                        payload_num <- unlist(strsplit(payload_num[a],paste0("/",find_dat)))[1]
                }else if(find_dat == "barometer"){
                        if(dat_name %in% b_ls){b <- sum(str_count(b_ls, paste0(dat_name,"$")))+1}
                        b_ls <- c(b_ls,dat_name)
                        payload_num <- unlist(payload[grepl(find_dat,payload, ignore.case = TRUE)])
                        payload_num <- unlist(strsplit(payload_num[b],paste0("/",find_dat)))[1]
                }else if(find_dat == "air"){
                        if(dat_name %in% d_ls){d <- sum(str_count(d_ls, paste0(dat_name,"$")))+1}
                        d_ls <- c(d_ls,dat_name)
                        payload_num <- unlist(payload[grepl(find_dat,payload, ignore.case = TRUE)])
                        payload_num <- unlist(strsplit(payload_num[d],paste0("/",find_dat)))[1]
                }else if(find_dat == "wave"){
                        if(dat_name %in% w_ls){w <- sum(str_count(w_ls, paste0(dat_name,"$")))+1}
                        w_ls <- c(w_ls,dat_name)
                        payload_num <- unlist(payload[grepl(find_dat,payload, ignore.case = TRUE)])
                        payload_num <- unlist(strsplit(payload_num[w],paste0("/",find_dat)))[1]
                }else{
                        payload_num <- unlist(strsplit(payload[grepl(find_dat,payload, ignore.case =
                        TRUE)],"/"))[1]
                }
                print(paste0(payload_num,"_",dat_name))
                write.csv(dat, paste0(ascii_ncei_dir,buoy,"/",folder_name,
                "/",folder_name,"_",payload_num,"_",dat_name,".csv"), row.names=FALSE, na = "NaN")
                rm(dat)
                if(exists("qc")){rm(qc)}
                if(exists("rl")){rm(rl)}
        }
```

```
                        nc_close(nc)
                        rm(nc, time_position)
                        if("time_10min" %in% time_list){
                                rm(time_10min_position)
                        }
                }else{
                        print("No data in NetCDF")
                        if(exists("nc")){nc_close(nc)}
                }
        }, error = function(e){print(paste0("can't open: ",file))}
        )
}

```

US Army Corps
of Engineers®

concat_data_2.R

```r
concat_data_2 <- function(buoys = "list of buoys", start_year = "2020", data_dir = "data_dir"){

        ##-------------------------------------------------------------------------------------
        ## concat NDBC web files and NCEI netCDF files
        ## Hall 12/23/2019
        ##-------------------------------------------------------------------------------------

        ## Actions:
        ## 1. Sets data locations

        ## To handle the downloaded NDBC website data, this step allows for the management of:
        ## Differing yearly file formats and spectral frequencies;
        ## Concatenates multiple date and time columns into one field;
        ## Removes redundant dates, as well as visibility and tide columns in stdmet data;
        ## If necessary (not needed for NDBC data but code provision is in place), allocates spectral data
       ##   into appropriate 38 frequencies (old wave sensors), and 47 frequencies (new wave sensors); and
        ## Converts NDBC r1 and r2 values to their correct values (NDBC stored data are scaled by 100 to
       ##   reduce storage requirements, so data are multiplied by 0.01).

        ## To handle the NCEI data, this step:
        ## Selects only relevant data for concatenation;
        ## Concatenates stdmet data to match NDBC website data nomenclature;
        ## Applies NDBC netCDF QC flags to the extracted data;
        ## Converts air, water and dew point temperatures from Kelvin to degree Celsius to match NDBC data;
        ## Removes zero ('0') wind gust values when no wind speed is present (data that weren't flagged in file).
        ## To handle the erroneous netCDF spectral frequency data, this part of the code steps through each
       ##   row of spectral data and attempt to match the available spectral frequency data to the appropriate
       ##   38 frequencies (old wave sensors) or 47 frequencies (new wave sensors).
        ## Removes redundant netCDF data points that are ~5-10 seconds apart.

        #-------------------------------------------------------------------------------------
        #-------------------------------------------------------------------------------------

        # load libraries (local run)
        library(lubridate)
        library(plyr)
        library(dplyr)
        library(data.table)
        library(naniar)

        ## load libraries (HPC run)
        # library(lubridate, lib="/p/home/candice/Rlibs/")
        # library(plyr, lib="/p/home/candice/Rlibs/")
        # library(dplyr, lib="/p/home/candice/Rlibs/")
        # library(data.table, lib="/p/home/candice/Rlibs/")
        # library(naniar, lib="/p/home/candice/Rlibs/")

        ##-------------------------------------------------------------------------------------
        ## set paths
        ##-------------------------------------------------------------------------------------
```

```r
# drive <- "E:/Candice/"
# drive <- "/p/work/candice/"
# data_dir <- paste0(drive, "projects/WaveTrends/annual_runs/data/")
setwd(data_dir)

raw_dir <- paste0(data_dir,"raw_data/")
setwd(data_dir)
# set input directories
unzip_ndbc_dir <- paste0(raw_dir,"ndbc/unzipped/")
ascii_ncei_dir <- paste0(raw_dir,"ncei/ascii/")

# set new out_dir
if (!file.exists(paste0(data_dir,"concat_data/"))) {dir.create((paste0(data_dir,"concat_data/")))}
out_dir <- paste0(data_dir,"concat_data/")
# ndbc
if (!file.exists(paste0(out_dir,"ndbc/"))) {dir.create((paste0(out_dir,"ndbc/")))}
ndbc_dir <- paste0(out_dir,"ndbc/")
# ncei
if (!file.exists(paste0(out_dir,"ncei/"))) {dir.create((paste0(out_dir,"ncei/")))}
ncei_dir <- paste0(out_dir,"ncei/")

# data types
dataTypes <- c("stdmet", "swden", "swdir", "swdir2", "swr1","swr2")
dataType_ab <- c("h", "w", "d", "i","j","k")
data_types_stdmet <- c("lat", "lon", "wind_direction", "wind_speed", "wind_gust",
                       "significant_wave_height", "dominant_period", "average_period", "mean_wave_direction",
                       "air_pressure", "air_temperature", "sea_surface_temperature", "dew_point_temperature")
data_types_spec <- c("c11", "c11m", "alpha1", "alpha2", "r1", "r2", "C12", "C13", "C22", "C33",
                     "Q12", "Q13", "gamma2", "gamma3", "phih", "rhq", "sensor_output") # c11 = spectral
                     energy, c11m = uncorrected spectral energy

##-------------------------------------------------------------------------------------------
## set buoy stations for stand-alone use
##-------------------------------------------------------------------------------------------

# list_ndbc <- read.csv(paste0(data_dir,"NDBC_buoys.csv"),header = TRUE)
# list_ndbc <- dplyr::filter(list_ndbc, list_ndbc$owner == "NDBC")
# list_ndbc_buoy <- as.character(list_ndbc$station)
# buoys <- buoy_ls[1]
# print(buoys)

#-------------------------------------------------------------------------------------------
#-------------------------------------------------------------------------------------------

## ndbc web data files

#-------------------------------------------------------------------------------------------

# looping through the listed buoy ID's
for (buoy in buoys){
```

```
print(paste0("Starting on ndbc... ",buoy))

#-------------------------------------------------------------------------------------

# selecting files to concatenate, then concatenating
file_list <- list.files(path = paste0(unzip_ndbc_dir,buoy), pattern = ".txt", full.names = TRUE)

if(length(file_list)>0){

        # # start writing to an output file
        # sink(paste0(data_dir,"1_ndbc_concat_",buoy,"_",Sys.Date(),".txt"))
        #
        print(paste0("Starting on ndbc... ",buoy))
        print(file_list)

        ## create buoy specific concat folder
        if (!file.exists(paste0(ndbc_dir,buoy,"/"))) {dir.create((paste0(ndbc_dir,buoy,"/")))}

        ## merging yearly and monthly datafiles of each type
        # run loops
        for (t in dataType_ab){
                print(t)
                files <- file_list[grep(pattern = paste0("/",buoy,t), file_list)]
                print(files)
                if(t == "w"){
                        spec_freq_available_ALL <- data.frame(matrix(NA, nrow = 0, ncol = 48))
                        for(spec_file in files){
                                con_1 <- file(spec_file,"r")
                                first_line <- readLines(con_1,n=3)
                                close(con_1)
                                dateString <-unlist(strsplit(first_line[[3]]," "))
                                df <- paste0(dateString[1],"/",dateString[2],"/",dateString[3])
                                df <- paste0(df, "  ", first_line[[1]])
                                df <- t(data.frame(unlist(strsplit(df, "  ")), stringsAsFactors =
                                FALSE))
                                rownames(df) <- NULL
                                library(plyr)
                                spec_freq_available_ALL <- qpcR:::rbind.na(spec_freq_available_ALL, df)
                        }
                        write.csv(spec_freq_available_ALL,
                        paste0(unzip_ndbc_dir,buoy,"_spec_freq_available_ALL.csv"), row.names=FALSE,
                        na = "NaN")
                        rm(df, spec_freq_available_ALL,con_1)
                }
                if(t == "j"){
                        spec_r_type_ALL <- data.frame(matrix(NA, nrow = 0, ncol = 48))
                        for(spec_file in files){
                                con_1 <- file(spec_file,"r")
                                first_line <- readLines(con_1, n = 2)
                                close(con_1)
```

```
147                                     first_line <- first_line[2]
148                                     dateString <-unlist(strsplit(first_line[[1]]," "))
149                                     df <- paste0(dateString[1],"/",dateString[2],"/",dateString[3])
150                                     df <- paste0(df, "  ", first_line[[1]])
151                                     df <- t(data.frame(unlist(strsplit(df, "  ")), stringsAsFactors =
                                        FALSE))
152                                     rownames(df) <- NULL
153                                     # remove empty columns
154                                     df <- data.frame(t(df[, colSums(df != "") != 0]), stringsAsFactors =
                                        FALSE)
155                                     df$X2 <- NULL
156                                     library(plyr)
157                                     spec_r_type_ALL <- qpcR:::rbind.na(spec_r_type_ALL, df)
158                             }
159                             write.csv(spec_r_type_ALL,
                                paste0(unzip_ndbc_dir,buoy,"_spec_r_type_ALL.csv"), row.names=FALSE, na =
                                "NaN")
160                             rm(df, spec_r_type_ALL,con_1)
161                     }
162
163                     # loop for each data type using concat_ndbc.R script
164                     if(t == "h"){
165                             source(paste0(data_dir,"concat_ndbc.R"))
166                             dataset <- concat_ndbc(files=files, start_year=start_year, t = "h", buoy =
                                buoy)
167
168                             # remove columns not required
169                             library(dplyr)
170                             dataset <- dplyr::select(dataset,-"VIS",-"TIDE")
171                             dataset <- dataset[rowSums(is.na(dataset)) != ncol(dataset), ]
172                             # remove files with no date
173                             dataset <- dataset[!is.na(dataset$DateTime),]
174                             # remove any duplicate rows
175                             dataset <- unique(dataset)
176                             # rename datasets
177                             if("WDIR" %in% names(dataset)){dataset <- dplyr::rename(dataset,
                                wind_direction=WDIR)}else{dataset$wind_direction <- NA;print("no WDIR
                                present, adding...")}
178                             if("WSPD" %in% names(dataset)){dataset <- dplyr::rename(dataset,
                                wind_speed=WSPD)}else{dataset$wind_speed <- NA;print("no WSPD present,
                                adding...")}
179                             if("GST"  %in% names(dataset)){dataset <- dplyr::rename(dataset,
                                wind_gust=GST)}else{dataset$wind_gust <- NA;print("no GST present, adding...")}
180                             if("WVHT" %in% names(dataset)){dataset <- dplyr::rename(dataset,
                                significant_wave_height=WVHT)}else{dataset$significant_wave_height <-
                                NA;print("no WVHT present, adding...")}
181                             if("DPD"  %in% names(dataset)){dataset <- dplyr::rename(dataset,
                                dominant_wave_period=DPD)}else{dataset$dominant_wave_period <- NA;print("no
                                DPD present, adding...")}
182                             if("APD"  %in% names(dataset)){dataset <- dplyr::rename(dataset,
```

```
average_wave_period=APD)}else{dataset$average_wave_period <- NA;print("no APD present, adding...")}
183 if("MWD"  %in% names(dataset)){dataset <- dplyr::rename(dataset, mean_wave_direction=MWD)}else{dataset$mean_wave_direction <- NA;print("no MWD present, adding...")}
184 if("PRES" %in% names(dataset)){dataset <- dplyr::rename(dataset, air_pressure_at_sea_level=PRES)}else{dataset$air_pressure_at_sea_level <- NA;print("no PRES present, adding...")}
185 if("ATMP" %in% names(dataset)){dataset <- dplyr::rename(dataset, air_temperature=ATMP)}else{dataset$air_temperature <- NA;print("no ATMP present, adding...")}
186 if("WTMP" %in% names(dataset)){dataset <- dplyr::rename(dataset, sea_surface_temperature=WTMP)}else{dataset$sea_surface_temperature <- NA;print("no WTMP present, adding...")}
187 if("DEWP" %in% names(dataset)){dataset <- dplyr::rename(dataset, dew_point_temperature=DEWP)}else{dataset$dew_point_temperature <- NA;print("no DEWP present, adding...")}
188 # formatting for missing data columns
189 if("wind_direction" %in% names(dataset)){print("data passed test")}else{dataset$wind_direction <- as.numeric(NA); print(paste0("empty wind_direction column added to NDBC stdmet for ", buoy))}
190 if("wind_speed" %in% names(dataset)){print("data passed test")}else{dataset$wind_speed <- as.numeric(NA); print(paste0("empty wind_speed column added to NDBC stdmet for ", buoy))}
191 if("wind_gust" %in% names(dataset)){print("data passed test")}else{dataset$wind_gust <- as.numeric(NA); print(paste0("empty wind_gust column added to NDBC stdmet for ", buoy))}
192 if("significant_wave_height" %in% names(dataset)){print("data passed test")}else{dataset$significant_wave_height <- as.numeric(NA);
    print(paste0("empty significant_wave_height column added to NDBC stdmet for ", buoy))}
193 if("dominant_wave_period" %in% names(dataset)){print("data passed test")}else{dataset$dominant_wave_period <- as.numeric(NA);
    print(paste0("empty dominant_period column added to NDBC stdmet for ", buoy))}
194 if("average_wave_period" %in% names(dataset)){print("data passed test")}else{dataset$average_wave_period <- as.numeric(NA);
    print(paste0("empty average_period column added to NDBC stdmet for ", buoy))}
195 if("mean_wave_direction" %in% names(dataset)){print("data passed test")}else{dataset$mean_wave_direction <- as.numeric(NA);
    print(paste0("empty mean_wave_direction column added to NDBC stdmet for ", buoy))}
196 if("air_pressure_at_sea_level" %in% names(dataset)){print("data passed test")}else{dataset$air_pressure_at_sea_level <- as.numeric(NA);
    print(paste0("empty air_pressure_at_sea_level column added to NDBC stdmet for ", buoy))}
197 if("air_temperature" %in% names(dataset)){print("data passed test")}else{dataset$air_temperature <- as.numeric(NA); print(paste0("empty air_temperature column added to NDBC stdmet for ", buoy))}
198 if("sea_surface_temperature" %in% names(dataset)){print("data passed test")}else{dataset$sea_surface_temperature <- as.numeric(NA);
```

```r
                print(paste0("empty sea_surface_temperature column added to NDBC stdmet for
                ", buoy))}
199             if("dew_point_temperature" %in% names(dataset)){print("data passed
                test")}else{dataset$dew_point_temperature <- as.numeric(NA);
                print(paste0("empty dew_point_temperature column added to NDBC stdmet for ",
                buoy))}
200
201             # reset order of df
202             dataCols <- c("DateTime", "wind_direction", "wind_speed", "wind_gust",
                "significant_wave_height", "dominant_wave_period",
203                           "average_wave_period", "mean_wave_direction",
                              "air_pressure_at_sea_level", "air_temperature",
                              "sea_surface_temperature",
204                           "dew_point_temperature" , "air_pressure_at_sea_level")
205             dataset <- dplyr::select(dataset, all_of(dataCols))
206             rm(dataCols)
207
208             # assign name and export file
209             file_name <- "_stdmet"
210             data_name <- paste0("s_",buoy,"_ndbc",file_name)
211             # print(d_list)
212             assign(data_name, dataset)
213
214             # save and export dataset
215             year_range <- paste0(year(min(dataset$DateTime,na.rm = TRUE)),"_",
                year(max(dataset$DateTime, na.rm = TRUE)))
216             write.csv(dataset, paste0(ndbc_dir, buoy, "/",data_name,"_",year_range,
                ".csv"), row.names=FALSE)
217             # saveRDS(dataset, file=paste0(ndbc_dir, buoy, "/",data_name,"_",year_range,
                ".rds"))
218             rm(dataset, year_range)
219         }else{
220             source(paste0(data_dir,"concat_ndbc.R"))
221             concat_ndbc(files=files, start_year=start_year, t = t, buoy = buoy)
222             data_avail <- ls(pattern = "dataset")
223
224             # assign name and export file
225             if(t=="w"){file_name <- "_c11"}
226             if(t=="d"){file_name <- "_alpha1"}
227             if(t=="i"){file_name <- "_alpha2"}
228             if(t=="j"){file_name <- "_r1"}
229             if(t=="k"){file_name <- "_r2"}
230
231             # saving individual datasets
232             data_list <- ls(pattern = "dataset_spec_")
233             print(data_list)
234
235             if(length(data_list)>0){
236                     for(d in data_list){
237                             print(paste0("Saving individual... ",d))
```

```
                dataset <- get(d)
                if(dim(dataset)[1]!= 0){
                        # save and export dataset
                        year_range1 <- paste0(year(min(dataset$DateTime,na.rm
                        = TRUE)),"_", year(max(dataset$DateTime, na.rm =
                        TRUE)))
                        records1 <- nrow(dataset)
                        if(unlist(strsplit(d,"_"))[3] != "new" &
                        unlist(strsplit(d,"_"))[3] != "old"){
                                count_cols <- ncol(dataset)
                                data_name <-
                                paste0("s_",buoy,"_ndbc",file_name,"_freq_",cou
                                nt_cols,"cols")

                        }else{
                                data_name <-
                                paste0("s_",buoy,"_ndbc",file_name,"_freq_",unl
                                ist(strsplit(d,"_"))[3])

                        }
                        # write.csv(dataset,
                        paste0(unlist(strsplit(concat_ind_dir,"s_"))[1],data_na
                        me,"_",year_range1,"_",records1,"_records.csv"),
                        row.names=FALSE)
                        # saveRDS(dataset, file =
                        paste0(unlist(strsplit(concat_ind_dir,"s_"))[1],data_na
                        me,"_",year_range1,".rds"))
                        rm(year_range1, records1)
                }
                rm(dataset)
        }
        rm(d, data_list)
}else{print(paste0("no ",t," data for ", buoy))}


#-------------------------------------------------------------------------------
----------
# reformat odd data

#-------------------------------------------------------------------------------
----------
data_list <- ls(pattern = "dataset_")
data_list <- data_list[data_list %in%
c("dataset_spec_new","dataset_spec_old") == FALSE]
print(data_list)

if(length(data_list)>0){
        for(d in data_list){
                df <- get(d)
                print(d)
```

```
                        if(dim(df)[1]!= 0){
                                # remove rows with no dates
                                df <- df[!is.na(df$DateTime), ]
                                # remove blank data rows
                                df <- df[rowSums(is.na(df)) != ncol(df)-1,]
                                if(dim(df)[1] > 0){
                                        # remove columns with zero or NA
                                        df <- df[, colSums(df != 0, na.rm = TRUE) > 0]
                                        # reorder columns numerically
                                        library(data.table)
                                        df <- setcolorder(df, c(1,
                                        order(as.numeric(names(df)[-1])) + 1))
                                        # ordering the df by date and selecting
                                        unique values only
                                        df <- df[order(df$DateTime),]
                                        df <- unique(df)
                                        # rename the rows to reflect unique data
                                        row.names(df) <- 1:nrow(df)
                                        # assign data
                                        assign(d,df)
                                }else{
                                        rm(list = ls()[grepl(d, ls())])
                                        print("dataset removed: all row NA or zero")
                                }
                        }
                        rm(df)
                }
        }else{print("no unusual frequency data A")}


        #-------------------------------------------------------------------------------
        ----------
        # testing data merge with old and new frequency dfs

        #-------------------------------------------------------------------------------
        ----------
        data_list <- ls(pattern = "dataset_")
        data_list <- data_list[data_list %in%
        c("dataset_spec_new","dataset_spec_old") == FALSE]
        print(data_list)

        if(length(data_list)>0){

                # dealing with multiple formats for the same frequencies
                if("dataset_spec_old" %in% data_avail){
                        print("dataset_spec_old already exists")

                }else{
                        colNames_spec_old <- c("DateTime",
                        "0.0100","0.0200","0.0300","0.0400","0.0500","0.0600","0.0700",
```

```
                "0.0800","0.0900","0.1000","0.1100","0.1200",
313
                                        "0.1300","0.1400","0.1500","0.1600","0.1
                                        700","0.1800","0.1900","0.2000","0.2100"
                                        ,"0.2200",
314
                                        "0.2300","0.2400","0.2500","0.2600","0.2
                                        700","0.2800","0.2900","0.3000","0.3100"
                                        ,"0.3200",
315
                                        "0.3300","0.3400","0.3500","0.3600","0.3
                                        700","0.3800","0.3900","0.4000")
316             dataset_spec_old <- data.frame(matrix(NA, nrow = 1, ncol =
                length(colNames_spec_old)))
317             colnames(dataset_spec_old) <- colNames_spec_old
318             dataset_spec_old$DateTime <-
                lubridate::ymd_hms(dataset_spec_old$DateTime)
319     }
320
321     if("dataset_spec_new" %in% data_avail){
322             print("dataset_spec_new already exists")
323     }else{
324             colNames_spec_new <- c("DateTime", "0.0200", "0.0325",
                "0.0375", "0.0425", "0.0475", "0.0525",
325                                     "0.0575", "0.0625", "0.0675",
                                        "0.0725", "0.0775", "0.0825",
                                        "0.0875", "0.0925", "0.1000", "0.1100",
326                                     "0.1200", "0.1300", "0.1400",
                                        "0.1500", "0.1600", "0.1700",
                                        "0.1800", "0.1900", "0.2000", "0.2100",
327                                     "0.2200", "0.2300", "0.2400",
                                        "0.2500", "0.2600", "0.2700",
                                        "0.2800", "0.2900", "0.3000", "0.3100",
328                                     "0.3200", "0.3300", "0.3400",
                                        "0.3500", "0.3650", "0.3850",
                                        "0.4050", "0.4250", "0.4450", "0.4650",
329                                     "0.4850")
330             dataset_spec_new <- data.frame(matrix(NA, nrow = 1, ncol =
                length(colNames_spec_new)))
331             colnames(dataset_spec_new) <- colNames_spec_new
332             dataset_spec_new$DateTime <-
                lubridate::ymd_hms(dataset_spec_new$DateTime)
333     }
334
335     # loop through odd frequencies to test if df can be merged with new
        and odd frequencies
336     # testing data merge
337     data_list <- ls(pattern = "dataset_")
338     data_list <- data_list[data_list %in%
        c("dataset_spec_new","dataset_spec_old") == FALSE]
```

```
        print(data_list)

        for (l in data_list){
                print(l)
                dat <- get(l)
                spec_list <- ls(pattern = "dataset_spec_")
                spec_list <- spec_list[spec_list %in%
                c("dataset_spec_new","dataset_spec_old") == TRUE]
                print(spec_list)

                for(spec in spec_list){
                        if(exists("dat")){
                                spec_dat <- get(spec)
                                ndbc_freq <- names(spec_dat)
                                dat_names <- names(dat)
                                setdiff(ndbc_freq, dat_names)
                                print(unique(dat_names %in% ndbc_freq))
                                # looping through data
                                if(any(unique(dat_names %in%
                                ndbc_freq)==FALSE)){
                                        print(paste0(spec," didn't match"))
                                }else{
                                        tryCatch({
                                                library(plyr)
                                                spec_dat <-
                                                rbind.fill(spec_dat,dat)
                                                print(paste0("concat ",spec,
                                                " and ",l," match"))
                                                assign(spec,spec_dat)
                                                rm(dat,spec_dat)
                                                rm(list = ls()[grepl(l, ls())])
                                        }, error = function(e) {
                                                print("dataset doesn't match")
                                        })
                                        if(exists("spec_dat")){rm(spec_dat)}
                                }
                                rm(spec_dat,ndbc_freq,dat_names)
                        }

                }
                if(exists("dat")){
                        assign(l,dat)
                        rm(dat)
                }
        }
}else{print("no unusual frequency data B")}


#-------------------------------------------------------------------------
----------
```

```r
# if still present, loop through odd frequencies to attempt a merge by row

#-------------------------------------------------------------------------------
----------
data_list <- ls(pattern = "dataset_")
data_list <- data_list[data_list %in%
c("dataset_spec_new","dataset_spec_old") == FALSE]
print(data_list)

if(length(data_list)>0){
        # loop through odd frequencies to attempt a merge by row
        # testing data merge
        data_list <- ls(pattern = "dataset_")
        if("dataset_spec_new" %in% data_list){df_freq_new <-
        dataset_spec_new; rm(dataset_spec_new)}
        if("dataset_spec_old" %in% data_list){df_freq_old <-
        dataset_spec_old; rm(dataset_spec_old)}
        data_list <- data_list[data_list %in%
        c("dataset_spec_new","dataset_spec_old") == FALSE]
        print(data_list)

        # try per row
        for (l in data_list){
                print(l)
                dat <- get(l)
                # loop through each row and try to match to new and old
                frequencies
                for(i in 1:nrow(dat)) {
                        print(i)
                        # Extract row and all columns
                        dat_row <- dat[i, ]
                        # remove cells with zero or NA - selects ANY NA and
                        removes column
                        library(dplyr)
                        dat_row <- dat_row %>% select_if(~ !any(is.na(.)))
                        # match data rows
                        spec_list <- ls(pattern = "df_freq_")
                        print(spec_list)

                        for(spec in spec_list){
                                if(exists("dat_row")){
                                        spec_dat <- get(spec)
                                        ndbc_freq <- names(spec_dat)
                                        dat_names <- names(dat_row)
                                        setdiff(ndbc_freq, dat_names)
                                        print(unique(dat_names %in% ndbc_freq))
                                        # looping through data
                                        if(any(unique(dat_names %in%
                                        ndbc_freq)==FALSE)){
                                                print(paste0(spec," didn't
```

```
                                                        match"))
424                                                 }else{
425                                                         tryCatch({
426                                                                 library(plyr)
427                                                                 spec_dat <-
                                                                    rbind.fill(spec_dat,dat
                                                                    _row)
428                                                                 print(paste0("concat
                                                                    ",spec, " and row
                                                                    ",i," match"))
429                                                                 assign(spec,spec_dat)
430                                                                 rm(dat_row,spec_dat)
431                                                         }, error = function(e) {
432                                                                 print("dataset
                                                                    doesn't match")
                                                                 })
433
434                                                         if(exists("spec_dat")){rm(spec_
                                                            dat)}
435                                                 }
436                                         }
437
438                                 }
439                                 if(exists("dat_row")){
440                                         count <- length(spec_list)+1
441                                         df_freq <- data.frame(matrix(NA, nrow = 0,
                                            ncol = dim(dat_row)[2]))
442                                         df_freq<-rbind(df_freq, dat_row)
443                                         new_name <- paste0("df_freq_",count)
444                                         assign(new_name, df_freq)
445                                         print(paste0("data added to NEW DF::
                                            ",new_name))
446                                         rm(dat_row, df_freq,new_name)
447                                 }
448
449                         }# end of row loop
450                         rm(dat,i,count)
451                         rm(list = ls()[grepl(l, ls())])
452                 }
453         }else{print("no usual frequency data C")}
454
455
                #-------------------------------------------------------------------------
                ----------
456             # renaming frequency datasets
457
                #-------------------------------------------------------------------------
                ----------
458             data_list <- ls(pattern = "df_freq_")
459             if("df_freq_new" %in% data_list){dataset_spec_new <- df_freq_new;
```

```r
rm(df_freq_new)}
if("df_freq_old" %in% data_list){dataset_spec_old <- df_freq_old;
rm(df_freq_old)}
data_list <- ls(pattern = "df_freq_")
if(length(data_list)>0){
        for(d in data_list){
                dat_name <- gsub("df_freq_","dataset_spec_",d)
                print(dat_name)
                assign(dat_name,get(d))
                rm(list = ls()[grepl(d, ls())])
                rm(d)
        }
}else{print("no usual frequency data D")}


#-------------------------------------------------------------------------------
----------
# trying to merge remaining odd frequencies

#-------------------------------------------------------------------------------
----------
data_list <- ls(pattern = "dataset_")
data_list <- data_list[data_list %in%
c("dataset_spec_new","dataset_spec_old") == FALSE]
print(data_list)
# n <- 0
# if there are any odd data left
if(length(data_list)>0){
        # test loop three times
        for (n in 1:3){
                print(n)
                data_list <- ls(pattern = "dataset_")
                data_list <- data_list[data_list %in%
                c("dataset_spec_new","dataset_spec_old") == FALSE]
                print(data_list)
                if(length(data_list)>1){
                        df_count <- data.frame(matrix(NA, nrow = 0, ncol = 2))
                        colnames(df_count) <- c("Name", "Count")
                        # calculate and save column dimensions
                        for(c in data_list){
                                df <- data.frame(matrix(NA, nrow = 0, ncol =
                                2))
                                colnames(df) <- c("Name", "Count")
                                df[1,] <- c(as.character(c), dim(get(c))[2])
                                df_count <- rbind(df_count, df)
                                rm(df)
                        }
                        # find the df with the most columns
                        df_longest <- df_count[match(max(df_count$Count,
                        na.rm = TRUE), df_count$Count),1]
```

```
print(df_longest)

# set main with which to merge shorter datasets
df_orig <- get(df_longest)
rm(list = ls()[grepl(df_longest, ls())])

# search for remaining odd frequency datasets
data_list <- ls(pattern = "dataset_")
data_list <- data_list[data_list %in%
c("dataset_spec_new","dataset_spec_old") == FALSE]
print(data_list)

for (l in data_list){
        print(l)
        dat <- get(l)
        if(exists("dat")){
                ndbc_freq <- names(df_orig)
                dat_names <- names(dat)
                setdiff(ndbc_freq, dat_names)
                print(unique(dat_names %in% ndbc_freq))
                # looping through data
                if(any(unique(dat_names %in%
                ndbc_freq)==FALSE)){
                        print(paste0(spec," didn't
                        match"))
                }else{
                        tryCatch({
                                library(plyr)
                                df_orig <-
                                rbind.fill(df_orig,dat)
                                print(paste0("concat
                                ",df_longest, " and
                                ",l," match"))
                                dat_name <-
                                paste0("df_freq_",
                                unlist(strsplit(df_long
                                est,"_"))[3])

                                assign(dat_name,df_orig
                                )
                                rm(dat)
                                rm(list =
                                ls()[grepl(l, ls())])
                        }, error = function(e) {
                                print("dataset
                                doesn't match")
                        })
                }
                rm(ndbc_freq,dat_names)
        }
```

```
                                        if(exists("dat")){
                                                assign(l,dat)
                                                rm(dat)
                                        }
                                }
                                rm(df_count, df_orig)

                        }else{print("only one odd frequency data left")}
                }
        }else{print("no remaining odd frequency data E")}


        #------------------------------------------------------------------------------
        ----------
        # renaming frequency datasets

        #------------------------------------------------------------------------------
        ----------
        data_list <- ls(pattern = "df_freq_")
        print(data_list)

        if(length(data_list)>0){
                for(d in data_list){
                        dat_name <- gsub("df_freq_","dataset_spec_",d)
                        print(dat_name)
                        assign(dat_name,get(d))
                        rm(list = ls()[grepl(d, ls())])
                        rm(d)
                }
        }else{print("no usual frequency data F")}


        #------------------------------------------------------------------------------
        ----------
        # formatting and exporting the final, merged data

        #------------------------------------------------------------------------------
        ----------
        data_list <- ls(pattern = "dataset_")
        print(data_list)

        if(length(data_list)>0){
                for(d in data_list){
                        # d <- data_list[1]
                        dataset <- get(d)
                        if(dim(dataset)[1]!= 0){
                                # correct for NDBC storage format - "The R1 and R2
                                values in the monthly and yearly
                                # historical data files are scaled by 100, a
                                carryover from how the data are transported
```

```
# to the archive centers. The units are hundredths, so the R1 and R2 values in those
# files should be multiplied by 0.01. ref: https://www.ndbc.noaa.gov/measdes.shtml"
if(t == "j" | t == "k"){
        dataset[,2:ncol(dataset)] <-
        dataset[,2:ncol(dataset)]*0.01
}
# remove empty rows - accounting for datasets with no lat/lon
dataset <- dataset[rowSums(is.na(dataset)) != ncol(dataset)-1,]
column_names <- names(dataset)
# remove columns with zero or NA in old freq
if("0.0100" %in% column_names){
        if(sum(dataset$`0.0100`, na.rm = TRUE)==0){dataset$`0.0100` <- NULL}
        if(sum(dataset$`0.0200`, na.rm = TRUE)==0){dataset$`0.0200` <- NULL}
        print("removing 0.0100 and 0.0200 frequency from dataset")
}else{print("no 0.0100 frequency in dataset")}
# ordering the dataset by date and selecting unique values only
dataset <- dataset[order(dataset$DateTime),]
dataset <- unique(dataset)
# rename the rows to reflect unique data
row.names(dataset) <- 1:nrow(dataset)
# save and export dataset
year_range <- paste0(year(min(dataset$DateTime,na.rm = TRUE)),"_", year(max(dataset$DateTime, na.rm = TRUE)))
records <- nrow(dataset)
if(unlist(strsplit(d,"_"))[3] != "new" & unlist(strsplit(d,"_"))[3] != "old"){
        count_cols <- ncol(dataset)-3
        data_name <-
        paste0("s_",buoy,"_ndbc",file_name,"_freq_",count_cols,"cols")
}else{
        data_name <-
        paste0("s_",buoy,"_ndbc",file_name,"_freq_",unlist(strsplit(d,"_"))[3])
}
# export
write.csv(dataset, paste0(ndbc_dir,buoy,"/",data_name,"_",year_range,"_",records,"_records.csv"), row.names=FALSE)
# saveRDS(dataset, file = paste0(ndbc_dir,buoy,"/",data_name,"_",year_range,".rds
```

```
                                                                "))
                                                                # assign(data_name, dataset, envir=globalenv())
                                                                assign(data_name, dataset)
                                                                rm(year_range, records)
                                                                rm(dataset)
                                                        }else{
                                                                rm(dataset)
                                                                rm(list = ls()[grepl(d, ls())])
                                                        }

                                                }
                                        }else{print("no data")}
                                        # housekeeping
                                        rm_ls <- ls(pattern = "dataset_")
                                        rm(list = rm_ls)
                                        rm(rm_ls,data_name,d,data_list)

                                } # end of individual dataType_ab loop

                        } # end of dataType_ab loop

                        # Save multiple objects
                        datasets <- ls(pattern = paste0(buoy,"_ndbc"))
                        print(datasets)
                        save(list = datasets, file = paste0(ndbc_dir, buoy, "/s_",buoy,"_ndbc_ALL.RData"))
                        # housekeeping
                        print(paste0("Completed data concat for ",buoy))
                        # rm(list = ls())
                        rm_ls <- ls(pattern = buoy)
                        rm(list = rm_ls)
                        rm(d, data_list,data_name, file_name, first_line, records, rm_ls, count_cols, t, datasets)

                        print(paste0("Finished ndbc... ",buoy))

                        # # Stop writing to the file
                        # sink()

                }else{print("no new ndbc data for this buoy")}

                #-------------------------------------------------------------------------------------

                print(paste0("Finished ndbc... ",buoy))
        }

        #-------------------------------------------------------------------------------------

        ## ncei ndbc netCDF data files

        #-------------------------------------------------------------------------------------
```

```r
# selecting files to concatenate, then concatenating
for(buoy in buoys){
        print(paste0("Starting on ncei... ",buoy))

        #-------------------------------------------------------------------------------------

        # selecting files to concatenate, then concatenating
        # find all files
        file_list <- list.files(path = paste0(ascii_ncei_dir,buoy), pattern = ".csv", full.names = TRUE,
        recursive = TRUE)

        # only execute if buoy data available
        if(length(file_list)>0){

                # # start writing to an output file
                # sink(paste0(data_dir,"1_ncei_concat_",buoy,"_",Sys.Date(),".txt"))

                print(paste0("Starting on ncei... ",buoy))
                print(file_list)

                ## create buoy specific concat folder
                if (!file.exists(paste0(ncei_dir,buoy,"/"))) {dir.create((paste0(ncei_dir,buoy,"/")))}

                #-------------------------------------------------------------------------------------
                # met data
                #-------------------------------------------------------------------------------------

                print(paste0("Starting stdmet data concat for ",buoy))

                source(paste0(data_dir,"concat_ncei.R"))
                print("initializing concat_ncei for stdmet...")
                concat_ncei(file_list, start_year, "stdmet", drive, buoy, file_list)
                print("finished concat_ncei for stdmet...")

                library(dplyr)
                # remove any duplicate rows
                dataset <- unique(dataset)
                # rename the rows to reflect unique data
                row.names(dataset) <- 1:nrow(dataset)

                # checking for different data nomenclature
                df_names <- names(dataset)
                if(length(grep("dominant_period",df_names, value = TRUE))>0){colnames(dataset) =
                gsub("dominant_period", "dominant_wave_period", colnames(dataset))}
                if(length(grep("average_period",df_names, value = TRUE))>0){colnames(dataset) =
                gsub("average_period", "average_wave_period", colnames(dataset))}
                if(length(grep("_pressure",df_names, value = TRUE))>0){colnames(dataset) = gsub("_pressure",
                "_pressure_at_sea_level", colnames(dataset))}
                # if("air_pressure_2" %in% names(dataset)){dataset <- dplyr::rename(dataset,
                air_pressure_at_sea_level_2 = air_pressure_2, air_pressure_at_sea_level_metadata_2 =
```

```
                air_pressure_metadata_2); print(paste0("renamed air_pressure_at_sea_level_2 column for ",
                buoy))}
702
703             # formatting for missing data columns
704             df_names <- names(dataset)
705             if(length(grep("wind_direction",df_names, value = TRUE))>0){print("wind_direction data
                present")}else{dataset$wind_direction <- as.numeric(NA); dataset$wind_direction_metadata <-
                as.numeric(NA); print(paste0("empty wind_direction column added to NCEI stdmet for ", buoy))}
706             if(length(grep("wind_speed",df_names, value = TRUE))>0){print("wind_speed data
                present")}else{dataset$wind_speed <- as.numeric(NA); dataset$wind_speed_metadata <-
                as.numeric(NA); print(paste0("empty wind_speed column added to NCEI stdmet for ", buoy))}
707             if(length(grep("wind_gust",df_names, value = TRUE))>0){print("wind_gust data
                present")}else{dataset$wind_gust <- as.numeric(NA); dataset$wind_gust_metadata <-
                as.numeric(NA); print(paste0("empty wind_gust column added to NCEI stdmet for ", buoy))}
708             if(length(grep("significant_wave_height",df_names, value =
                TRUE))>0){print("significant_wave_height data present")}else{dataset$significant_wave_height
                <- as.numeric(NA); dataset$significant_wave_height_metadata <- as.numeric(NA);
                print(paste0("empty significant_wave_height column added to NCEI stdmet for ", buoy))}
709             if(length(grep("dominant_wave_period",df_names, value = TRUE))>0){print("dominant_wave_period
                data present")}else{dataset$dominant_wave_period <- as.numeric(NA);
                dataset$dominant_wave_period_metadata <- as.numeric(NA); print(paste0("empty
                dominant_wave_period column added to NCEI stdmet for ", buoy))}
710             if(length(grep("average_wave_period",df_names, value = TRUE))>0){print("average_wave_period
                data present")}else{dataset$average_wave_period <- as.numeric(NA);
                dataset$average_wave_period_metadata <- as.numeric(NA); print(paste0("empty
                average_wave_period column added to NCEI stdmet for ", buoy))}
711             if(length(grep("mean_wave_direction",df_names, value = TRUE))>0){print("mean_wave_direction
                data present")}else{dataset$mean_wave_direction <- as.numeric(NA);
                dataset$mean_wave_direction_metadata <- as.numeric(NA); print(paste0("empty
                mean_wave_direction column added to NCEI stdmet for ", buoy))}
712             if(length(grep("air_pressure_at_sea_level",df_names, value =
                TRUE))>0){print("air_pressure_at_sea_level data
                present")}else{dataset$air_pressure_at_sea_level <- as.numeric(NA);
                dataset$air_pressure_at_sea_level_metadata <- as.numeric(NA); print(paste0("empty
                air_pressure_at_sea_level column added to NCEI stdmet for ", buoy))}
713             if(length(grep("air_temperature",df_names, value = TRUE))>0){print("air_temperature data
                present")}else{dataset$air_temperature <- as.numeric(NA); dataset$air_temperature_metadata <-
                as.numeric(NA); print(paste0("empty air_temperature column added to NCEI stdmet for ", buoy))}
714             if(length(grep("sea_surface_temperature",df_names, value =
                TRUE))>0){print("sea_surface_temperature data present")}else{dataset$sea_surface_temperature
                <- as.numeric(NA); dataset$sea_surface_temperature_metadata <- as.numeric(NA);
                print(paste0("empty sea_surface_temperature column added to NCEI stdmet for ", buoy))}
715             if(length(grep("dew_point_temperature",df_names, value =
                TRUE))>0){print("dew_point_temperature data present")}else{dataset$dew_point_temperature <-
                as.numeric(NA); dataset$dew_point_temperature_metadata <- as.numeric(NA); print(paste0("empty
                dew_point_temperature column added to NCEI stdmet for ", buoy))}
716
717             # deal with skipped GPS positions not lasting more than 4 hours
718             if("lat" %in% names(dataset)){
719                     for(c in match(c("lat","lon"),names(dataset))){
```

```
720                                 for(loops in 1:3){
721                                         find_NA <- which(is.na(dataset[,c]))
722                                         NA_count <- length(find_NA)
723                                         print(paste0("Pre count of NA: ",NA_count, ", run: ",loops))
724                                         for(p in find_NA){
725                                                 tryCatch({if(!is.na(dataset[p-1,c]) &
                                                 !is.na(dataset[p+1,c])){dataset[p,c] <- dataset[p-1,c]}},
                                                 error = function(cond){print("no lat/lon inserted")})
726                                                 tryCatch({if(!is.na(dataset[p-1,c]) &
                                                 !is.na(dataset[p+2,c])){dataset[p,c] <- dataset[p-1,c]}},
                                                 error = function(cond){print("no lat/lon inserted")})
727                                                 tryCatch({if(!is.na(dataset[p-2,c]) &
                                                 !is.na(dataset[p+2,c])){dataset[p,c] <- dataset[p-2,c]}},
                                                 error = function(cond){print("no lat/lon inserted")})
728                                                 tryCatch({if(!is.na(dataset[p-1,c]) &
                                                 !is.na(dataset[p+2,c])){dataset[p,c] <- dataset[p-2,c]}},
                                                 error = function(cond){print("no lat/lon inserted")})
729                                         }
730                                         find_NA <- which(is.na(dataset[,c]))
731                                         NA_count <- length(find_NA)
732                                         print(paste0("Post count of NA: ",NA_count, ", run: ",loops))
733                                 }
734                         }
735                 }else{
736                         dataset <- left_join(dataset, gps_positions, by = "DateTime")
737                         print(paste0("adding gps positions to ",d))
738                 }
739
740                 # reset order of df
741                 df_names <- names(dataset)
742                 dataCols <- c("DateTime","lat","lat_metadata","lon","lon_metadata")
743                 dataCols <- c(dataCols,grep("wind_direction",df_names, value = TRUE))
744                 dataCols <- c(dataCols,grep("wind_speed",df_names, value = TRUE))
745                 dataCols <- c(dataCols,grep("wind_gust",df_names, value = TRUE))
746                 dataCols <- c(dataCols,grep("significant_wave_height",df_names, value = TRUE))
747                 dataCols <- c(dataCols,grep("dominant_wave_period",df_names, value = TRUE))
748                 dataCols <- c(dataCols,grep("average_wave_period",df_names, value = TRUE))
749                 dataCols <- c(dataCols,grep("mean_wave_direction",df_names, value = TRUE))
750                 dataCols <- c(dataCols,grep("air_pressure_at_sea_level",df_names, value = TRUE))
751                 dataCols <- c(dataCols,grep("air_temperature",df_names, value = TRUE))
752                 dataCols <- c(dataCols,grep("sea_surface_temperature",df_names, value = TRUE))
753                 dataCols <- c(dataCols,grep("dew_point_temperature",df_names, value = TRUE))
754                 dataCols <- c(dataCols,grep("air_pressure_at_sea_level",df_names, value = TRUE))
755                 # reorder columns
756                 dataset <- dplyr::select(dataset, all_of(dataCols))
757                 rm(dataCols)
758
759                 # save and export
760                 year_range <- paste0(year(min(dataset$DateTime,na.rm = TRUE)),"_", year(max(dataset$DateTime,
                    na.rm = TRUE)))
```

```
761                     write.csv(dataset, paste0(ncei_dir,buoy,"/s_",buoy, "_ncei_h_stdmet_",year_range,".csv"),
                        row.names=FALSE)
762                     # saveRDS(dataset, file = paste0(ncei_dir,buoy,"/s_",buoy,"_ncei_stdmet_",year_range,".rds"))
763                     data_name <- paste0("s_",buoy,"_ncei_stdmet")
764                     assign(data_name, dataset)
765
766                     gps_positions <- dplyr::select(dataset, "DateTime", "lat", "lon")
767                     rm(dataset, year_range)
768                     print(paste0("Completed stdmet data concat for ",buoy))
769
770                     #-------------------------------------------------------------------------------------
771                     # spectral data
772                     #-------------------------------------------------------------------------------------
773                     print(paste0("Starting spectral data concat for ",buoy))
774
775                     for (t in data_types_spec){
776                             print(t)
777                             # run concat_ncei function code
778                             source(paste0(data_dir,"concat_ncei.R"))
779
780                             print("initializing concat_ncei for spec...")
781                             concat_ncei(file_list, start_year, "spec", drive, buoy, file_list, t)
782                             print("finishing concat_ncei for spec...")
783
784                             data_avail <- ls(pattern = "dataset")
785
786                             if(length(data_avail)>0){
787                                     if(t == "sensor_output"){
788                                             # ordering the dataset by date and selecting unique values only
789                                             dataset <- dataset[order(dataset$DateTime),]
790                                             dataset <- unique(dataset)
791                                             # rename the rows to reflect unique data
792                                             row.names(dataset) <- 1:nrow(dataset)
793                                             # save and export dataset
794                                             year_range <- paste0(year(min(dataset$DateTime,na.rm = TRUE)),"_",
                                                year(max(dataset$DateTime, na.rm = TRUE)))
795                                             records <- nrow(dataset)
796                                             data_name <- paste0("s_",buoy,"_ncei_",t)
797                                             print("exporting dataset")
798                                             # write.csv(dataset,
                                                paste0(ncei_dir,buoy,"/",data_name,"_",year_range,"_",records,"_records
                                                .csv"), row.names=FALSE)
799                                             # saveRDS(dataset, file =
                                                paste0(ncei_dir,buoy,"/",data_name,"_",year_range,".rds"))
800                                             assign(data_name, dataset)
801                                             rm(dataset, year_range)
802                                     }else{
803                                             # formatting the data
804                                             data_list <- ls(pattern = "dataset_")
805                                             for(d in data_list){
```

```
806                 print(paste0("Saving individual... ",d))
807                 dataset <- get(d)
808                 if(dim(dataset)[1]!= 0){
809                         # save and export dataset
810                         year_range1 <- paste0(year(min(dataset$DateTime,na.rm
                            = TRUE)),"_", year(max(dataset$DateTime, na.rm =
                            TRUE)))
811                         records1 <- nrow(dataset)
812                         if(unlist(strsplit(d,"_"))[3] != "new" &
                            unlist(strsplit(d,"_"))[3] != "old"){
813                                 count_cols <- ncol(dataset)
814                                 data_name <-
                                    paste0("s_",buoy,"_ncei_",t,"_freq_",count_cols
                                    ,"cols")
815
816                         }else{
817                                 data_name <-
                                    paste0("s_",buoy,"_ncei_",t,"_freq_",unlist(str
                                    split(d,"_"))[3])
818
819                         }
820                         # write.csv(dataset,
                            paste0(unlist(strsplit(concat_ind_dir,"s_"))[1],data_na
                            me,"_",year_range1,"_",records1,"_records.csv"),
                            row.names=FALSE)
821                         # saveRDS(dataset, file =
                            paste0(unlist(strsplit(concat_ind_dir,"s_"))[1],data_na
                            me,"_",year_range1,".rds"))
822                         rm(year_range1, records1)
823                 }
824                 rm(dataset)
825         }
826         rm(d)
827
          #-------------------------------------------------------------------
          ------------------
828         # reformat odd data
829
          #-------------------------------------------------------------------
          ------------------
830         data_list <- ls(pattern = "dataset_")
831         data_list <- data_list[data_list %in%
            c("dataset_spec_new","dataset_spec_old") == FALSE]
832         print(data_list)
833
834         if(length(data_list)>0){
835                 for(d in data_list){
836                         df <- get(d)
837                         print(d)
838                         if(dim(df)[1]!= 0){
```

```
                                        # remove rows with no dates
                                        df <- df[!is.na(df$DateTime), ]
                                        # remove blank data rows
                                        df <- df[rowSums(is.na(df)) != ncol(df)-3,]
                                        if(dim(df)[2]>3){   # to handle two column data
                                                df$count <- rowSums(df[4:ncol(df)],
                                                na.rm = TRUE)
                                                df <- dplyr::filter(df, count != 0)
                                                df$count <- NULL
                                        }
                                        if(dim(df)[1] > 0){
                                                # format data
                                                df1 <- dplyr::select(df, DateTime,
                                                lat,lon)
                                                df2 <- df
                                                df2$lat <- NULL; df2$lon <- NULL
                                                # remove columns with zero or NA
                                                df2 <- df2[, colSums(df2 != 0, na.rm
                                                = TRUE) > 0]
                                                # reorder columns numerically
                                                library(data.table)
                                                df2 <- setcolorder(df2, c(1,
                                                order(as.numeric(names(df2)[-1])) + 1))
                                                df <- full_join(df1,df2, by =
                                                "DateTime")
                                                rm(df1,df2)
                                                # ordering the df by date and
                                                selecting unique values only
                                                df <- df[order(df$DateTime),]
                                                df <- unique(df)
                                                # rename the rows to reflect unique
                                                data
                                                row.names(df) <- 1:nrow(df)
                                                # assign data
                                                assign(d,df)
                                        }else{
                                                rm(list = ls()[grepl(d, ls())])
                                                print("dataset removed: all row NA or
                                                zero")
                                        }
                                }
                                rm(df)
                        }
                }else{print("no unusual data A")}


                #-------------------------------------------------------------------
                -----------------
                # testing data merge with old and new frequency dfs
```

```
#-------------------------------------------------------------------
------------------
data_list <- ls(pattern = "dataset_")
data_list <- data_list[data_list %in%
c("dataset_spec_new","dataset_spec_old") == FALSE]
print(data_list)

if(length(data_list)>0){

        # dealing with multiple formats for the same frequencies
        if("dataset_spec_old" %in% data_avail){
                print("dataset_spec_old already exists")

        }else{
                colNames_spec_old <- c("DateTime", "lat", "lon",
                "0.0100","0.0200","0.0300","0.0400","0.0500","0.0600","
                0.0700","0.0800","0.0900","0.1000","0.1100","0.1200",

                                        "0.1300","0.1400","0.1500","0.16
                                        00","0.1700","0.1800","0.1900","
                                        0.2000","0.2100","0.2200",

                                        "0.2300","0.2400","0.2500","0.26
                                        00","0.2700","0.2800","0.2900","
                                        0.3000","0.3100","0.3200",

                                        "0.3300","0.3400","0.3500","0.36
                                        00","0.3700","0.3800","0.3900","
                                        0.4000")
                dataset_spec_old <- data.frame(matrix(NA, nrow = 1,
                ncol = length(colNames_spec_old)))
                colnames(dataset_spec_old) <- colNames_spec_old
                dataset_spec_old$DateTime <-
                lubridate::ymd_hms(dataset_spec_old$DateTime)
        }

        if("dataset_spec_new" %in% data_avail){
                print("dataset_spec_new already exists")
        }else{
                colNames_spec_new <- c("DateTime", "lat", "lon",
                "0.0200", "0.0325", "0.0375", "0.0425", "0.0475",
                "0.0525",
                                        "0.0575", "0.0625", "0.0675",
                                        "0.0725", "0.0775", "0.0825",
                                        "0.0875", "0.0925", "0.1000",
                                        "0.1100",
                                        "0.1200", "0.1300", "0.1400",
                                        "0.1500", "0.1600", "0.1700",
                                        "0.1800", "0.1900", "0.2000",
                                        "0.2100",
```

```
                                                "0.2200", "0.2300", "0.2400",
                                                "0.2500", "0.2600", "0.2700",
                                                "0.2800", "0.2900", "0.3000",
                                                "0.3100",
                                                "0.3200", "0.3300", "0.3400",
                                                "0.3500", "0.3650", "0.3850",
                                                "0.4050", "0.4250", "0.4450",
                                                "0.4650",
                                                "0.4850")
        dataset_spec_new <- data.frame(matrix(NA, nrow = 1,
        ncol = length(colNames_spec_new)))
        colnames(dataset_spec_new) <- colNames_spec_new
        dataset_spec_new$DateTime <-
        lubridate::ymd_hms(dataset_spec_new$DateTime)
}

# loop through odd frequencies to test if df can be merged
with new and odd frequencies
# testing data merge
data_list <- ls(pattern = "dataset_")
data_list <- data_list[data_list %in%
c("dataset_spec_new","dataset_spec_old") == FALSE]
print(data_list)

for (l in data_list){
        print(l)
        dat <- get(l)
        spec_list <- ls(pattern = "dataset_spec_")
        spec_list <- spec_list[spec_list %in%
        c("dataset_spec_new","dataset_spec_old") == TRUE]
        print(spec_list)

        for(spec in spec_list){
                if(exists("dat")){
                        spec_dat <- get(spec)
                        ndbc_freq <- names(spec_dat)
                        dat_names <- names(dat)
                        setdiff(ndbc_freq, dat_names)
                        print(unique(dat_names %in% ndbc_freq))
                        # looping through data
                        if(any(unique(dat_names %in%
                        ndbc_freq)==FALSE)){
                                print(paste0(spec," didn't
                                match"))
                        }else{
                                tryCatch({
                                        library(plyr)
                                        spec_dat <-
                                        rbind.fill(spec_dat,dat
                                        )
```

```
                                                print(paste0("concat
                                                ",spec, " and ",l,"
                                                match"))
                                                assign(spec,spec_dat)
                                                rm(dat,spec_dat)
                                                rm(list =
                                                ls()[grepl(l, ls())])
                                        }, error = function(e) {
                                                print("dataset
                                                doesn't match")
                                        })

                                        if(exists("spec_dat")){rm(spec_
                                        dat)}
                                }
                                rm(spec_dat,ndbc_freq,dat_names)
                        }


                }
                if(exists("dat")){
                        assign(l,dat)
                        rm(dat)
                }
        }
}else{print("no unusual frequency data B")}


#-------------------------------------------------------------------------
------------------
# if still present, loop through odd frequencies to attempt a merge
by row

#-------------------------------------------------------------------------
------------------
data_list <- ls(pattern = "dataset_")
data_list <- data_list[data_list %in%
c("dataset_spec_new","dataset_spec_old") == FALSE]
print(data_list)

if(length(data_list)>0){
        # loop through odd frequencies to attempt a merge by row
        # testing data merge
        data_list <- ls(pattern = "dataset_")
        if("dataset_spec_new" %in% data_list){df_freq_new <-
        dataset_spec_new; rm(dataset_spec_new)}
        if("dataset_spec_old" %in% data_list){df_freq_old <-
        dataset_spec_old; rm(dataset_spec_old)}
        data_list <- data_list[data_list %in%
        c("dataset_spec_new","dataset_spec_old") == FALSE]
        print(data_list)
```

```r
# try per row
for (l in data_list){
        print(l)
        dat <- get(l)
        # loop through each row and try to match to new and old frequencies
        for(i in 1:nrow(dat)) {
                print(i)
                # Extract row and all columns
                dat_row <- dat[i, ]
                # remove cells with zero or NA - selects ANY NA and removes column
                library(dplyr)
                dat_row <- dat_row %>% select_if(~ !any(is.na(.)))
                # match data rows
                spec_list <- ls(pattern = "df_freq_")
                # spec_list <- spec_list[spec_list %in% c("df_freq_new","df_freq_old") == TRUE]
                print(spec_list)

                for(spec in spec_list){
                        if(exists("dat_row")){
                                spec_dat <- get(spec)
                                ndbc_freq <- names(spec_dat)
                                dat_names <- names(dat_row)
                                setdiff(ndbc_freq, dat_names)
                                print(unique(dat_names %in% ndbc_freq))
                                # looping through data
                                if(any(unique(dat_names %in% ndbc_freq)==FALSE)){
                                        print(paste0(spec," didn't match"))
                                }else{
                                        tryCatch({
                                                library(plyr)
                                                spec_dat <- rbind.fill(spec_dat,dat_row)

                                                print(paste0("concat ",spec," and row ",i," match"))

                                                assign(spec,spec_dat)
```

```
                                                                    rm(dat_row,spec
                                                                    _dat)
1010                                                                # rm(list =
                                                                    ls()[grepl(d,
                                                                    ls())])
1011                                                            }, error =
                                                                function(e) {
1012
                                                                    print("dataset
                                                                    doesn't match")
1013                                                            })
1014
                                                                if(exists("spec_dat")){
                                                                rm(spec_dat)}
1015                                                        }
1016                                                    }
1017
1018                                            }
1019                                            if(exists("dat_row")){
1020                                                    count <- length(spec_list)+1
1021                                                    df_freq <- data.frame(matrix(NA, nrow
                                                    = 0, ncol = dim(dat_row)[2]))
1022                                                    df_freq<-rbind(df_freq, dat_row)
1023                                                    new_name <- paste0("df_freq_",count)
1024                                                    assign(new_name, df_freq)
1025                                                    print(paste0("data added to NEW DF::
                                                    ",new_name))
1026                                                    rm(dat_row, df_freq,new_name)
1027                                            }
1028
1029                                    }# end of row loop
1030                                    rm(dat,i,count)
1031                                    rm(list = ls()[grepl(l, ls())])
1032                            }
1033                    }else{print("no usual frequency data C")}
1034
1035

                    #-------------------------------------------------------------------
                    ------------------
1036                # renaming frequency datasets
1037
                    #-------------------------------------------------------------------
                    ------------------
1038                data_list <- ls(pattern = "df_freq_")
1039                if("df_freq_new" %in% data_list){dataset_spec_new <- df_freq_new;
                    rm(df_freq_new)}
1040                if("df_freq_old" %in% data_list){dataset_spec_old <- df_freq_old;
                    rm(df_freq_old)}
1041                data_list <- ls(pattern = "df_freq_")
1042                if(length(data_list)>0){
```

```
1043                    for(d in data_list){
1044                            dat_name <- gsub("df_freq_","dataset_spec_",d)
1045                            print(dat_name)
1046                            assign(dat_name,get(d))
1047                            rm(list = ls()[grepl(d, ls())])
1048                            rm(d)
1049                    }
1050            }else{print("no usual frequency data D")}
1051
1052
1053            # trying to merge remaining odd frequencies within themselves
1054
1055            data_list <- ls(pattern = "dataset_")
1056            data_list <- data_list[data_list %in%
                c("dataset_spec_new","dataset_spec_old") == FALSE]
1057            print(data_list)
1058            # if there are any odd data left
1059            if(length(data_list)>0){
1060                    # test loop three times
1061                    for (n in 1:3){
1062                            print(n)
1063                            data_list <- ls(pattern = "dataset_")
1064                            data_list <- data_list[data_list %in%
                                c("dataset_spec_new","dataset_spec_old") == FALSE]
1065                            print(data_list)
1066                            if(length(data_list)>1){
1067                                    df_count <- data.frame(matrix(NA, nrow = 0,
                                        ncol = 2))
1068                                    colnames(df_count) <- c("Name", "Count")
1069                                    # calculate and save column dimensions
1070                                    for(c in data_list){
1071                                            df <- data.frame(matrix(NA, nrow = 0,
                                                ncol = 2))
1072                                            colnames(df) <- c("Name", "Count")
1073                                            df[1,] <- c(as.character(c),
                                                dim(get(c))[2])
1074                                            df_count <- rbind(df_count, df)
1075                                            rm(df)
1076                                    }
1077                                    # find the df with the most columns
1078                                    df_longest <-
                                        df_count[match(max(df_count$Count, na.rm =
                                        TRUE), df_count$Count),1]
1079                                    print(df_longest)
1080
1081                                    # set main with which to merge shorter datasets
```

```
df_orig <- get(df_longest)
rm(list = ls()[grepl(df_longest, ls())])

# search for remaining odd frequency datasets
data_list <- ls(pattern = "dataset_")
data_list <- data_list[data_list %in%
c("dataset_spec_new","dataset_spec_old") ==
FALSE]
print(data_list)

for (l in data_list){
        print(l)
        dat <- get(l)
        if(exists("dat")){
                ndbc_freq <- names(df_orig)
                dat_names <- names(dat)
                setdiff(ndbc_freq, dat_names)
                print(unique(dat_names %in%
                ndbc_freq))
                # looping through data
                if(any(unique(dat_names %in%
                ndbc_freq)==FALSE)){
                        print(paste0(spec,"
                        didn't match"))
                }else{
                        tryCatch({
                                library(plyr)
                                df_orig <-
                                rbind.fill(df_o
                                rig,dat)

                                print(paste0("c
                                oncat
                                ",df_longest,
                                " and ",l,"
                                match"))
                                dat_name <-
                                paste0("df_freq
                                _",
                                unlist(strsplit
                                (df_longest,"_"
                                ))[3])

                                assign(dat_name
                                ,df_orig)
                                rm(dat)
                                rm(list =
                                ls()[grepl(l,
                                ls())])
                        }, error =
```

```
                                                                function(e) {
1111
                                                                        print("dataset
                                                                        doesn't match")
1112                                                            })
1113
                                                                #if(exists("spec_dat"))
                                                                {rm(spec_dat)}
1114                                                    }
1115                                                    rm(ndbc_freq,dat_names)
1116                                            }
1117                                            if(exists("dat")){
1118                                                    assign(l,dat)
1119                                                    rm(dat)
1120                                            }
1121                                    }
1122                                    rm(df_count, df_orig)
1123
1124                            }else{print("only one odd frequency data left")}
1125                    }
1126            }else{print("no remaining odd frequency data E")}
1127
1128
                    #-------------------------------------------------------------------
                    ------------------
1129                # renaming frequency datasets after trying to merge odd frequencies
                    with each other
1130
                    #-------------------------------------------------------------------
                    ------------------
1131                data_list <- ls(pattern = "df_freq_")
1132                print(data_list)
1133
1134                if(length(data_list)>0){
1135                        for(d in data_list){
1136                                dat_name <- gsub("df_freq_","dataset_spec_",d)
1137                                print(dat_name)
1138                                assign(dat_name,get(d))
1139                                rm(list = ls()[grepl(d, ls())])
1140                                rm(d)
1141                        }
1142                }else{print("no usual frequency data F")}
1143
1144
                    #-------------------------------------------------------------------
                    ------------------
1145                # formatting the merged data
1146
                    #-------------------------------------------------------------------
                    ------------------
```

```
data_list <- ls(pattern = "dataset_")
print(data_list)

if(length(data_list)>0){
        for(d in data_list){
                print(paste0("Saving concat... ",d))
                dataset <- get(d)
                # remove empty rows across df
                dataset <- dataset[rowSums(is.na(dataset)) !=
                ncol(dataset),]
                # remove rows with no datetimes
                dataset <- dataset[!is.na(dataset$DateTime),]

                if(dim(dataset)[1]!= 0){
                        # remove empty rows - accounting for datasets
                        with no lat/lon
                        column_names <- names(dataset)
                        if("lat" %in% column_names){
                                dataset <-
                                dataset[rowSums(is.na(dataset)) !=
                                ncol(dataset)-3,]
                                dataset <-
                                dataset[rowSums(is.na(dataset)) !=
                                ncol(dataset)-1,]
                        }else {dataset <-
                        dataset[rowSums(is.na(dataset)) !=
                        ncol(dataset)-1,]}
                        # remove columns with zero or NA in old freq
                        if("0.0100" %in% column_names){
                                if(sum(dataset$`0.0100`, na.rm =
                                TRUE)==0){dataset$`0.0100` <- NULL}
                                if(sum(dataset$`0.0200`, na.rm =
                                TRUE)==0){dataset$`0.0200` <- NULL}

                        }
                        if(dim(dataset)[1]!= 0){
                                # deal with skipped GPS positions not
                                lasting more than 4 hours
                                if("lat" %in% names(dataset)){
                                        for(c in
                                        match(c("lat","lon"),names(data
                                        set))){
                                                for(loops in 1:3){
                                                        # c <- 3
                                                        find_NA <-
                                                        which(is.na(dat
                                                        aset[,c]))
                                                        NA_count <-
                                                        length(find_NA)
```

```
                        print(paste0("Pre count of NA: ",NA_count, ", run: ",loops))
                        for(p in find_NA){
                            # p <- find_NA[1]
                            # print(p)
                            tryCatch({if(!is.na(dataset[p-1,c]) & !is.na(dataset[p+1,c])){dataset[p,c] <- dataset[p-1,c]}}, error = function(cond){print("no lat/lon inserted")})
                            tryCatch({if(!is.na(dataset[p-1,c]) & !is.na(dataset[p+2,c])){dataset[p,c] <- dataset[p-1,c]}}, error = function(cond){print("no lat/lon inserted")})
                            tryCatch({if(!is.na(dataset[p-2,c]) & !is.na(dataset[p+2,c])){dataset[p,c] <- dataset[p-2,c]}}, error = function(cond){print("no lat/lon inserted")})
                            tryCatch({if(!is.na(dataset[p-1,c]) & !is.na(dataset[p+2,c])){dataset[p,c] <- dataset[p-2,c]}}, error = function(cond){print("no lat/lon inserted")})
                        }
                        find_NA <- which(is.na(dataset[,c]))
                        NA_count <- length(find_NA)
                        print(paste0("Post count of NA: ",NA_count, ", run: ",loops))
                    }
                }
            }else{
                dataset <- left_join(dataset, gps_positions, by = "DateTime")
                print(paste0("adding gps positions to ",d))
            }
        }else{
            print(paste0("empty dataframe ",d))
            rm(dataset)
        }
```

```
                                                    # ordering the dataset by date and selecting
                                                    unique values only
                                                    dataset <- dataset[order(dataset$DateTime),]
                                                    dataset <- unique(dataset)
                                                    # rename the rows to reflect unique data
                                                    row.names(dataset) <- 1:nrow(dataset)
                                                    # save and export dataset
                                                    year_range <-
                                                    paste0(year(min(dataset$DateTime,na.rm =
                                                    TRUE)),"_", year(max(dataset$DateTime, na.rm
                                                    = TRUE)))
                                                    records <- nrow(dataset)
                                                    if(unlist(strsplit(d,"_"))[3] != "new" &
                                                    unlist(strsplit(d,"_"))[3] != "old"){
                                                            count_cols <- ncol(dataset)-3
                                                            data_name <-
                                                            paste0("s_",buoy,"_ncei_",t,"_freq_",co
                                                            unt_cols,"cols")
                                                    }else{
                                                            data_name <-
                                                            paste0("s_",buoy,"_ncei_",t,"_freq_",un
                                                            list(strsplit(d,"_"))[3])
                                                    }

                                                    print("exporting datasets")
                                                    write.csv(dataset,
                                                    paste0(ncei_dir,buoy,"/",data_name,"_",year_ran
                                                    ge,"_",records,"_records.csv"),
                                                    row.names=FALSE)
                                                    # saveRDS(dataset, file =
                                                    paste0(ncei_dir,buoy,"/",data_name,"_",year_ran
                                                    ge,".rds"))
                                                    # assign(data_name, dataset,
                                                    envir=parent.frame())
                                                    assign(data_name, dataset)
                                                    rm(year_range, records)
                                                    rm(dataset)
                                            }else{
                                                    rm(dataset)
                                                    rm(list = ls()[grepl(d, ls())])
                                            }                                                   }
                                    }else{print("no data to merge")}

                                    rm_ls <- ls(pattern = "dataset_")
                                    rm(list = rm_ls)
                                    rm(rm_ls,data_name,d,data_list)
                            } # end of no sensor loop

                    }else{print(paste0("no dataset:",t))} # end of individual t

```

```
                        } # end of all t loop

                        # Save multiple objects
                        datasets <- ls(pattern = paste0(buoy,"_ncei"))
                        # return(datasets)
                        print(datasets)

                        save(list = datasets, file = paste0(ncei_dir, buoy, "/s_",buoy,"_ncei_ALL.RData"))
                        # housekeeping
                        rm(data_name,datasets, file_list, t,dataset)
                        print(paste0("Completed spectral data concat for ",buoy))
                        print(paste0("Completed data concat for ",buoy))

                        rm_ls <- ls(pattern = buoy)
                        rm(list = rm_ls)
                        rm(gps_positions)

                        print(paste0("Finished ncei buoy... ",buoy))

                        # # Stop writing to the file
                        # sink()
                        #-----------------------------------------------------------------------------------------

                        print(paste0("Finished ncei buoy... ",buoy))

                }else{
                        print(paste0("No new ncei data for buoy ",buoy))
                }
        }

        #-----------------------------------------------------------------------------------------
        #-----------------------------------------------------------------------------------------
        # clean glob environ
        # rm(list = ls())
        #-----------------------------------------------------------------------------------------
        #-----------------------------------------------------------------------------------------
}
```

US Army Corps
of Engineers®

concat_ndbc.R

```r
concat_ndbc <- function(files = "list of files", start_year = "earliest dataset year", t = t, buoy = buoy){

  print(paste0("Working on...",t))

  colNames_stdmet <- c("YYYY","MM","DD","hh","mm","WDIR","WSPD","GST","WVHT","DPD","APD","MWD","PRES","ATMP","WTMP",
                       "DEWP","VIS","TIDE")

  NA_strings <- c(9.96920996838687E+36,"9.96920996838687E+36","9.96921e+36",9.96921e+36,999.00,999,
                  "-32767",-32767,"-2147483647",-2147483647,NA,
                  NaN,"NA","NaN","9969209968386869046778552952102584320.000",
                  "9.96920996838686E+36","99.0","9999.0","999","999.0","99.00","999.00")

  date_formatted <- function(df){
        # find and removed rows with names/text
        selectedRows <- df[grep("YY", df[,1]), ]
        if(nrow(selectedRows) > 0){
              df <- df[!(df[,1] %in% selectedRows[,1]),]
        }
        rm(selectedRows)
        # convert characters to numeric
        df[,] <- apply(df[,],2,function(x) as.numeric(as.character(x)))
        # adding leading 0 to month, day and hour
        df$MM <-formatC(df$MM, width = 2, format = "d", flag = "0")
        df$DD <-formatC(df$DD, width = 2, format = "d", flag = "0")
        df$hh <-formatC(df$hh, width = 2, format = "d", flag = "0")
        df$mm <-formatC(df$mm, width = 2, format = "d", flag = "0")
        if(names(df[1])== "YY"){
            if(nchar(df[1,1]) == 2){
              df$year <- as.integer(19)
              colY = c(grep("year", colnames(df)), grep("YY", colnames(df)))
              df <- cbind(YYYY = do.call(paste0, df[colY]), df)
              df <- dplyr::select(df,-"YY",-"year")
            }else{
              df <- dplyr::rename(df,"YYYY" = "YY")
            }
        }
        # Creating concatenated individual and combined date and time columns
        colst = c(grep("YYYY", colnames(df)), grep("MM", colnames(df)),
                  grep("DD", colnames(df)), grep("hh", colnames(df)),
                  grep("mm", colnames(df)))
        df <- cbind(DateTime = do.call(paste0, df[colst]), df)
        ## setting the date and time format
        library(lubridate)
        df$DateTime <- ymd_hm(df$DateTime)
        # ordering the dataset by date and selecting unique values only
        df <- df[order(df$DateTime),]
        df <- unique(df)
        # rename the rows to reflect unique data
        row.names(df) <- 1:nrow(df)
        # remove redundant columns
```

```
50          library(dplyr)
51          df <- dplyr::select(df,-"YYYY",-"MM",-"DD",-"hh",-"mm")
52          # removed rows that are all NA
53          df <- df[rowSums(is.na(df)) != ncol(df), ]
54      }
55
56      read_header <- function(file){
57          # read file
58          header <- readLines(file, n = 1)
59          if(grepl("#", header)== TRUE){header <- gsub("#", "",header)}
60          header <- gsub("\\s+", ",", gsub("^\\s+|\\s+$", "",header))
61          if(grepl("mm", header)== TRUE){
62              years <- unlist(strsplit(header, "mm"))[1]
63              freq <- unlist(strsplit(header, "mm"))[2]
64          }else{
65              years <- unlist(strsplit(header, "hh"))[1]
66              freq <- unlist(strsplit(header, "hh"))[2]
67          }
68          years <- unlist(strsplit(years, ","))
69          freq <- unlist(strsplit(freq, ","))
70          freq <- stringi::stri_remove_empty(freq, na_empty = FALSE)
71          freq <- as.numeric(freq)
72          freq <- sprintf("%1.4f", freq)
73          freq <- as.character(freq)
74          if(grepl("mm", header)== TRUE){
75              header <- c(years, "mm", freq)
76          }else{header <- c(years, "hh", freq)}
77      }
78
79      if(t != "h"){
80            colNames_spec_new <- c("DateTime", "0.0200", "0.0325", "0.0375", "0.0425", "0.0475", "0.0525",
81                                   "0.0575", "0.0625", "0.0675", "0.0725", "0.0775", "0.0825", "0.0875", "0.0925",
                                     "0.1000", "0.1100",
82                                   "0.1200", "0.1300", "0.1400", "0.1500", "0.1600", "0.1700", "0.1800", "0.1900",
                                     "0.2000", "0.2100",
83                                   "0.2200", "0.2300", "0.2400", "0.2500", "0.2600", "0.2700", "0.2800", "0.2900",
                                     "0.3000", "0.3100",
84                                   "0.3200", "0.3300", "0.3400", "0.3500", "0.3650", "0.3850", "0.4050", "0.4250",
                                     "0.4450", "0.4650",
85                                   "0.4850")
86            colNames_spec_old <- c("DateTime",
              "0.0300","0.0400","0.0500","0.0600","0.0700","0.0800","0.0900","0.1000","0.1100","0.1200",
87
                                     "0.1300","0.1400","0.1500","0.1600","0.1700","0.1800","0.1900","0.2000","0.2100",
                                     "0.2200",
88
                                     "0.2300","0.2400","0.2500","0.2600","0.2700","0.2800","0.2900","0.3000","0.3100",
                                     "0.3200",
89                                   "0.3300","0.3400","0.3500","0.3600","0.3700","0.3800","0.3900","0.4000")
90
```

```
91          dataset_spec_new <- data.frame(matrix(NA, nrow = 0, ncol = length(colNames_spec_new)))
92          colnames(dataset_spec_new) <- colNames_spec_new
93
94          dataset_spec_old <- data.frame(matrix(NA, nrow = 0, ncol = length(colNames_spec_old)))
95          colnames(dataset_spec_old) <- colNames_spec_old
96
97          count <- 0
98    }
99
100   ##-------------------------------------------------------------------------------------------
101   # Concatenate all data files to handle different data formats
102   ##-------------------------------------------------------------------------------------------
103
104   ## 2007 to present:
105   ## stdmet format:     ##YY  MM DD hh mm WDIR WSPD GST  WVHT   DPD   APD MWD   PRES  ATMP  WTMP  DEWP  VIS  TIDE
106   ##         units:     ##yr  mo dy hr mn degT m/s  m/s     m   sec   sec deg    hPa  degC  degC  degC  nmi    ft
107   ## spectral format:   #YY  MM DD hh mm   .0200  .0325  .0375  .0425  .0475  .0525  .0575  .0625  .0675  .0725
      .0775
108   ##                                      .0825  .0875  .0925  .1000  .1100  .1200  .1300  .1400  .1500  .1600
      .1700
109   ##                                      .1800  .1900  .2000  .2100  .2200  .2300  .2400  .2500  .2600  .2700
      .2800
110   ##                                      .2900  .3000  .3100  .3200  .3300  .3400  .3500  .3650  .3850  .4050
      .4250
111   ##                                      .4450  .4650  .4850
112
113   ## selecting files to concatenate, then concatenating
114   # loop through files and concat
115   if(t == "h"){
116
117         library(lubridate)
118         if(buoy == "46029"){
119               dates <- c(2006:year(Sys.Date()))
120         }else{
121               dates <- c(2007:year(Sys.Date()))
122         }
123         ## find the files for this data setup
124         file_dates <- list()
125         for (i in 1:length(dates)){
126             file_dates[[i]] <- files[grepl(paste0(dates[i],"_"),files)]
127         }
128         # remove list function from subset data list
129         file_dates <- unlist(file_dates)
130
131         # print(file_dates)
132         if(length(file_dates) > 0){
133               for(file in file_dates){
134                     # if the merged dataset doesn't exist, create it
135                     if (!exists("dataset")){
136                       dataset <- read.table(file, header=FALSE, na.strings = NA_strings,fill = T, skip = 2)
```

```
                    }
                    # if the merged dataset does exist, append to it
                    if (exists("dataset")){
                      temp_dataset <-read.table(file, header=FALSE, na.strings = NA_strings, fill = T, skip = 2)
                      dataset<-rbind(dataset, temp_dataset)
                      rm(temp_dataset)
                    }
                }
                colnames(dataset) <- colNames_stdmet
                dataset <- date_formatted(dataset)
                dataset_master <- dataset
                rm(dataset)
                # return(dataset_master)
        }
    }else{
        library(lubridate)
        dates <- c(2007:year(Sys.Date()))
        ## find the files for this data setup
        file_dates <- list()
        for (i in 1:length(dates)){
            file_dates[[i]] <- files[grepl(paste0(dates[i],"_"),files)]
        }
        # remove list function from subset data list
        file_dates <- unlist(file_dates)
        # print(file_dates)
        if(length(file_dates) > 0){
            for(file in file_dates){
                # check for empty files
                if(file.info(file)$size !=0){
                    # read header data
                    header <- read_header(file)
                    # read file
                    dataset <- read.table(file, header=FALSE, na.strings = NA_strings, fill = T, skip = 0)
                    names(dataset) <- header
                    # format date
                    dataset <- date_formatted(dataset)
                    # find all available frames
                    dataset_list <- ls(pattern = "dataset_spec")

                    for(matchable in dataset_list){
                      dat <- get(matchable)
                      if(exists("dataset")){
                          if(identical(names(dat),names(dataset))){
                            dat<-rbind(dat, dataset)
                            print(paste0("added to: ",matchable))
                            assign(matchable,dat)
                            rm(dataset)
                          }
                      }
                      rm(dat)
```

```
                    }
                    if(exists("dataset")){
                      count <- count + 1
                      dataset_spec <- data.frame(matrix(NA, nrow = 0, ncol = dim(dataset)[2]))
                      dataset_spec<-rbind(dataset_spec, dataset)
                      new_name <- paste0("dataset_spec_",count)
                      assign(new_name, dataset_spec)
                      print(paste0("data added to NEW DF:: ",new_name))
                      rm(dataset, dataset_spec)
                    }
                }else{print("no data in these files")}
            }
        }
    }

    ##-------------------------------------------------------------------------------------

    ## 2005 & 2006 - skip first line, has no # flag

    ## stdmet data format:    YYYY MM DD hh mm  WD  WSPD GST  WVHT   DPD   APD  MWD  BAR    ATMP  WTMP  DEWP  VIS  TIDE
    ##                        units: no units in files
    ## spectral data format:  YYYY MM DD hh mm  .030   .040   .050   .060   .070   .080   .090   .100   .110   .120
    ##                                          .130   .140   .150   .160   .170   .180   .190   .200   .210   .220
    ##                                          .230   .240   .250   .260   .270   .280   .290   .300   .310   .320
    ##                                          .330   .340   .350   .360   .370   .380   .390   .400

    # loop through files and concat
    if(t == "h"){
        ## selecting files to concatenate, then concatenating
        library(lubridate)
        if(buoy == "46029"){
          dates <- c(2004,2005)
        }else{
          dates <- c(2005,2006)
        }
        ## find the files for this data setup
        file_dates <- list()
        for (i in 1:length(dates)){
            file_dates[[i]] <- files[grepl(paste0(dates[i],"_"),files)]
        }
        # remove list function from subset data list
        file_dates <- unlist(file_dates)
        # print(file_dates)

        if(length(file_dates) > 0){
            for(file in file_dates){
                # if the merged dataset doesn't exist, create it
                if (!exists("dataset")){
                    dataset <- read.table(file, header=FALSE, na.strings = NA_strings,fill = T, skip = 1)
                }
```

```
                        # if the merged dataset does exist, append to it
                        if (exists("dataset")){
                              temp_dataset <-read.table(file, header=FALSE, na.strings = NA_strings, fill = T, skip =
                              1)
                              dataset<-rbind(dataset, temp_dataset)
                              rm(temp_dataset)
                        }
                }
                colnames(dataset) <- colNames_stdmet
                dataset <- date_formatted(dataset)
                ## combining datasets
                if(exists("dataset_master")){
                  dataset_master<-rbind(dataset_master, dataset)
                }else{dataset_master<-dataset}
                rm(dataset)
                # return(dataset_master)

        }
    }else{
        dates <- c(2005,2006)
        ## find the files for this data setup
        file_dates <- list()
        for (i in 1:length(dates)){
            file_dates[[i]] <- files[grepl(paste0(dates[i],"_"),files)]
        }
        # remove list function from subset data list
        file_dates <- unlist(file_dates)
        # print(file_dates)
        if(length(file_dates) > 0){
            for(file in file_dates){
              # check for empty files
              if(file.info(file)$size !=0){
                  header <- read_header(file)
                  # read file
                  dataset <- read.table(file, header=FALSE, na.strings = NA_strings, fill = T, skip = 0)
                  names(dataset) <- header
                  # format date
                  dataset <- date_formatted(dataset)
                  # find all available frames
                  dataset_list <- ls(pattern = "dataset_spec")

                  for(matchable in dataset_list){
                      dat <- get(matchable)
                      if(exists("dataset")){
                            if(identical(names(dat),names(dataset))){
                                dat<-rbind(dat, dataset)
                                print(paste0("added to: ",matchable))
                                assign(matchable,dat)
                                rm(dataset)
                            }
```

```
                    }
                    rm(dat)
                }
                if(exists("dataset")){
                    count <- count + 1
                    dataset_spec <- data.frame(matrix(NA, nrow = 0, ncol = dim(dataset)[2]))
                    dataset_spec<-rbind(dataset_spec, dataset)
                    new_name <- paste0("dataset_spec_",count)
                    assign(new_name, dataset_spec)
                    print(paste0("data added to NEW DF:: ",new_name))
                    rm(dataset, dataset_spec)
                }
            }
        }
    }
}


##---------------------------------------------------------------------------------------

## 2000 & 2004 - no minute column - don't skip lines, missing tide data in some sets

## stdmet data format:    YYYY MM DD hh WD   WSPD GST  WVHT  DPD   APD  MWD  BAR    ATMP  WTMP  DEWP  VIS  TIDE
##                        units: no units in files
## spectral data format:  YYYY MM DD hh   .030   .040   .050   .060   .070   .080   .090   .100   .110   .120
##                                       .130   .140   .150   .160   .170   .180   .190   .200   .210   .220
##                                       .230   .240   .250   .260   .270   .280   .290   .300   .310   .320
##                                       .330   .340   .350   .360   .370   .380   .390   .400

Names <- c("YYYY", "MM", "DD", "hh", "WDIR", "WSPD", "GST",  "WVHT", "DPD", "APD", "MWD",  "PRES", "ATMP", "WTMP",
"DEWP", "VIS", "TIDE")

# loop through files and concat
if(t == "h"){
    ## selecting files to concatenate, then concatenating
    library(lubridate)
    if(buoy == "46029"){
      dates <- c(2000:2003)
    }else{
      dates <- c(2000:2004)
    }
    ## find the files for this data setup
    file_dates <- list()
    for (i in 1:length(dates)){
        file_dates[[i]] <- files[grepl(paste0(dates[i],"_"),files)]
    }
    # remove list function from subset data list
    file_dates <- unlist(file_dates)
    # print(file_dates)
    if(length(file_dates) > 0){
```

```
            for(file in file_dates){
                    # if the merged dataset doesn't exist, create it
                    if (!exists("dataset")){
                      dataset <- read.table(file, header=FALSE, na.strings = NA_strings, fill = T)
                    }
                    # if the merged dataset does exist, append to it
                    if (exists("dataset")){
                      temp_dataset <-read.table(file, header=FALSE, na.strings = NA_strings, fill = T)
                      dataset<-rbind(dataset, temp_dataset)
                      rm(temp_dataset)
                    }
            }
            # rename columns
            colnames(dataset) <- Names
            ## adding in minutes column
            dataset$mm <- as.integer(0)
            dataset <- date_formatted(dataset)
            ## combining datasets
            if(exists("dataset_master")){
              dataset_master<-rbind(dataset_master, dataset)
            }else{dataset_master<-dataset}
            rm(dataset)
            # return(dataset_master)

        }
    }else{
        dates <- c(2000:2004)
        ## find the files for this data setup
        file_dates <- list()
        for (i in 1:length(dates)){
            file_dates[[i]] <- files[grepl(paste0(dates[i],"_"),files)]
        }
        # remove list function from subset data list
        file_dates <- unlist(file_dates)
        if(length(file_dates) > 0){
            for(file in file_dates){
                # check for empty files
                if(file.info(file)$size !=0){
                    header <- read_header(file)
                    # read file
                    dataset <- read.table(file, header=FALSE, na.strings = NA_strings, fill = T, skip = 0)
                    names(dataset) <- header
                    # add missing minute column
                    dataset$mm <- as.integer(0)
                    # format date
                    dataset <- date_formatted(dataset)
                    # find all available frames
                    dataset_list <- ls(pattern = "dataset_spec")

                    for(matchable in dataset_list){
```

```
                    dat <- get(matchable)
                    if(exists("dataset")){
                          if(identical(names(dat),names(dataset))){
                            dat<-rbind(dat, dataset)
                            print(paste0("added to: ",matchable))
                            assign(matchable,dat)
                            rm(dataset)
                          }
                    }
                    rm(dat)
                  }
                  if(exists("dataset")){
                      count <- count + 1
                      dataset_spec <- data.frame(matrix(NA, nrow = 0, ncol = dim(dataset)[2]))
                      dataset_spec<-rbind(dataset_spec, dataset)
                      new_name <- paste0("dataset_spec_",count)
                      assign(new_name, dataset_spec)
                      print(paste0("data added to NEW DF:: ",new_name))
                      rm(dataset, dataset_spec)
                  }
              }
          }
      }
  }

  ##-------------------------------------------------------------------------------------------

  ## 1999 - no TIDE or minute column

  ## stdmet data format:    YYYY MM DD hh WD WSPD GST  WVHT   DPD   APD  MWD   BAR    ATMP  WTMP  DEWP  VIS
  ##                        units: no units in files
  ## spectral data format:  YYYY MM DD hh   .030   .040   .050   .060   .070   .080   .090   .100   .110   .120
  ##                                       .130   .140   .150   .160   .170   .180   .190   .200   .210   .220
  ##                                       .230   .240   .250   .260   .270   .280   .290   .300   .310   .320
  ##                                       .330   .340   .350   .360   .370   .380   .390   .400

  Names <- c("YYYY", "MM", "DD", "hh", "WDIR", "WSPD", "GST",  "WVHT", "DPD", "APD", "MWD",  "PRES", "ATMP", "WTMP",
  "DEWP", "VIS")

  ## selecting files to concatenate, then concatenating
  library(lubridate)
  dates <- c(1999)
  ## find the files for this data setup
  file_dates <- list()
  for (i in 1:length(dates)){
      file_dates[[i]] <- files[grepl(paste0(dates[i],"_"),files)]
  }
  # remove list function from subset data list
  file_dates <- unlist(file_dates)
  # print(file_dates)
```

```
434
435     # loop through files and concat
436     if(t == "h"){
437           if(length(file_dates) > 0){
438                 for(file in file_dates){
439                     # if the merged dataset doesn't exist, create it
440                     if (!exists("dataset")){
441                       dataset <- read.table(file, header=FALSE, na.strings = NA_strings, fill = T, skip = 1)
442                     }
443                     # if the merged dataset does exist, append to it
444                     if (exists("dataset")){
445                       temp_dataset <-read.table(file, header=FALSE, na.strings = NA_strings, fill = T, skip = 1)
446                       dataset<-rbind(dataset, temp_dataset)
447                       rm(temp_dataset)
448                     }
449                 }
450                 ## rename columns
451                 colnames(dataset) <- Names
452                 ## adding in minutes column
453                 if(buoy == 41009){ # using minute data from NetCDF files
454                   dat1 <- filter(dataset, dataset$MM < 9)
455                   min <- data.frame(c(NA,rep(c(20,50),nrow(dat1)/2)), stringsAsFactors = FALSE)
456                   dat1$mm <- min[1:nrow(min)-1,]
457                   dat2 <- filter(dataset, dataset$MM >= 9)
458                   dat2$mm <- as.integer(0)
459                   dataset <- rbind(dat1,dat2)
460                   rm(dat1,dat2,min)
461                 }else{
462                   dataset$mm <- as.integer(0)
463                 }
464                 dataset$TIDE <- as.logical(NA)
465                 dataset <- date_formatted(dataset)
466                 ## combining datasets
467                 if(exists("dataset_master")){
468                   dataset_master<-rbind(dataset_master, dataset)
469                 }else{dataset_master<-dataset}
470                 rm(dataset)
471
472           }
473     }else{
474         if(length(file_dates) > 0){
475             for(file in file_dates){
476                 # check for empty files
477                 if(file.info(file)$size !=0){
478                     header <- read_header(file)
479                     # read file
480                     dataset <- read.table(file, header=FALSE, na.strings = NA_strings, fill = T, skip = 0)
481                     names(dataset) <- header
482                     # add missing minute column
483                     dataset$mm <- as.integer(0)
```

```
                    # format date
                    dataset <- date_formatted(dataset)
                    # find all available frames
                    dataset_list <- ls(pattern = "dataset_spec")

                    for(matchable in dataset_list){
                        dat <- get(matchable)
                        if(exists("dataset")){
                              if(identical(names(dat),names(dataset))){
                                dat<-rbind(dat, dataset)
                                print(paste0("added to: ",matchable))
                                assign(matchable,dat)
                                rm(dataset)
                              }
                        }
                        rm(dat)
                    }
                    if(exists("dataset")){
                        count <- count + 1
                        dataset_spec <- data.frame(matrix(NA, nrow = 0, ncol = dim(dataset)[2]))
                        dataset_spec<-rbind(dataset_spec, dataset)
                        new_name <- paste0("dataset_spec_",count)
                        assign(new_name, dataset_spec)
                        print(paste0("data added to NEW DF:: ",new_name))
                        rm(dataset, dataset_spec)
                    }
                }
            }
        }
    }

    ##-------------------------------------------------------------------------------------

    ##1979 & 1998 - no tide and minute columns

    ## stdmet data format:    YY MM DD hh WD   WSPD GST  WVHT  DPD   APD  MWD  BAR    ATMP  WTMP  DEWP  VIS
    ##                        units: no units in files
    ## spectral data format:  YY MM DD hh  .0200  .0325  .0375  .0425  .0475  .0525  .0575  .0625  .0675  .0725
    ##                        .0775  .0825  .0875  .0925  .1000  .1100  .1200  .1300  .1400  .1500  .1600  .1700
    ##                        .1800  .1900  .2000  .2100  .2200  .2300  .2400  .2500  .2600  .2700  .2800  .2900
    ##                        .3000  .3100  .3200  .3300  .3400  .3500  .3650  .3850  .4050  .4250  .4450  .4650  .4850

    Names <- c("YY", "MM", "DD", "hh", "WDIR", "WSPD", "GST",  "WVHT", "DPD", "APD", "MWD",  "PRES", "ATMP", "WTMP",
    "DEWP", "VIS")

    ## selecting files to concatenate, then concatenating
    library(lubridate)
    dates <- c(start_year:1998)
    ## find the files for this data setup
    file_dates <- list()
```

```
    for (i in 1:length(dates)){
        file_dates[[i]] <- files[grepl(paste0(dates[i],"_"),files)]
    }
    # remove list function from subset data list
    file_dates <- unlist(file_dates)

    # loop through files and concat
    if(t == "h"){
        if(length(file_dates) > 0){
              for(file in file_dates){
                    # if the merged dataset doesn't exist, create it
                    if (!exists("dataset")){
                      dataset <- read.table(file, header=FALSE, na.strings = NA_strings, fill = T, skip = 1)
                    }
                    # if the merged dataset does exist, append to it
                    if (exists("dataset")){
                      temp_dataset <-read.table(file, header=FALSE, na.strings = NA_strings, fill = T, skip = 1)
                      dataset<-rbind(dataset, temp_dataset)
                      rm(temp_dataset)
                    }
              }
              ## rename columns
              colnames(dataset) <- Names
              ## adding in missing column
              dataset$TIDE <- as.logical(NA)
              dataset$Year <- as.integer(19)
              ## Creating new column with '19', and combining YY column to be 4 digits.
              colst = c(grep("Year", colnames(dataset)), grep("YY", colnames(dataset)))
              dataset <- cbind(YYYY = do.call(paste0, dataset[colst]),
                               dataset)
              dataset$YYYY <- as.numeric(as.character(dataset$YYYY))
              ## delete working columns of YYY and YY values
              dataset$Year <- NULL; dataset$YY <- NULL
              ## adding in minutes column
              if(buoy == 41009){ # using minute data from NetCDF files
                    dataset$mm <- as.integer(0)
                    dataset <- date_formatted(dataset)
                    dat1 <- filter(dataset,dataset$DateTime < as.Date("1992-08-01 00:00:00"))
                    dat1$mm <- as.numeric(rep(c(00,30),nrow(dat1)/2))
                    minute(dat1$DateTime) <- as.numeric(dat1$mm)
                    dat1 <- dplyr::select(dat1,-"mm")
                    dat2 <- filter(dataset,dataset$DateTime >= as.Date("1992-08-01 00:00:00"))
                    dat2$mm <- as.numeric(rep(c(20,50),nrow(dat2)/2))
                    minute(dat2$DateTime) <- as.numeric(dat2$mm)
                    dat2 <- dplyr::select(dat2,-"mm")
                    dataset <- rbind(dat1,dat2)
                    rm(dat1,dat2)
              }else{
                    dataset$mm <- as.integer(0)
                    dataset <- date_formatted(dataset)
```

```
583             }
584 
585             ## combining datasets
586             if(exists("dataset_master")){
587               dataset_master<-rbind(dataset_master, dataset)
588             }else{dataset_master<-dataset}
589             rm(dataset)
590             # return(dataset_master)
591 
592         }
593     }else{
594         if(length(file_dates) > 0){
595             for(file in file_dates){
596                 # check for empty files
597                 if(file.info(file)$size !=0){
598                     header <- read_header(file)
599                     # read file
600                     dataset <- read.table(file, header=FALSE, na.strings = NA_strings, fill = T, skip = 0)
601                     names(dataset) <- header
602                     # add missing minute column
603                     dataset$mm <- as.integer(0)
604                     # format date
605                     dataset <- date_formatted(dataset)
606                     # find all available frames
607                     dataset_list <- ls(pattern = "dataset_spec")
608 
609                     for(matchable in dataset_list){
610                         dat <- get(matchable)
611                         if(exists("dataset")){
612                               if(identical(names(dat),names(dataset))){
613                                 dat<-rbind(dat, dataset)
614                                 print(paste0("added to: ",matchable))
615                                 assign(matchable,dat)
616                                 rm(dataset)
617                               }
618                           }
619                         rm(dat)
620                       }
621                     if(exists("dataset")){
622                         count <- count + 1
623                         dataset_spec <- data.frame(matrix(NA, nrow = 0, ncol = dim(dataset)[2]))
624                         dataset_spec<-rbind(dataset_spec, dataset)
625                         new_name <- paste0("dataset_spec_",count)
626                         assign(new_name, dataset_spec)
627                         print(paste0("data added to NEW DF:: ",new_name))
628                         rm(dataset, dataset_spec)
629                     }
630                 }
631             }
632         }
```

```
633     }
634
635     ##------------------------------------------------------------------------------------------
636     # fix structure
637     ##------------------------------------------------------------------------------------------
638     if(t == "h"){
639           dataset <- dataset_master
640           rm(dataset_master)
641           # ordering the dataset by date and selecting unique values only
642           dataset <- dataset[order(dataset$DateTime),]
643           dataset <- unique(dataset)
644           # rename the rows to reflect unique data
645           row.names(dataset) <- 1:nrow(dataset)
646           assign("dataset",dataset)
647     }else{
648           # adding leading 0 to month, day and hour
649           data_list <- ls(pattern = "dataset_spec")
650           for(d in data_list){
651               dataset <- get(d)
652               if(dim(dataset)[1] != 0){
653                     # ordering the dataset by date and selecting unique values only
654                     dataset <- dataset[order(dataset$DateTime),]
655                     dataset <- unique(dataset)
656                     # rename the rows to reflect unique data
657                     row.names(dataset) <- 1:nrow(dataset)
658                     # export from function
659                     assign(d, dataset, envir=parent.frame())
660               }
661               rm(dataset)
662           }
663     }
664   }
```

US Army Corps
of Engineers®

concat_ncei.R

```r
concat_ncei <- function(files = "list of files", start_year = "earliest dataset year", input = "stdmet or spec",
drive = "drive", buoy = "buoy", file_list = "file_list", t = "variable"){

  NA_strings <- c(9.96920996838687E+36,"9.96920996838687E+36","9.96921e+36",9.96921e+36,999.00,999,
                  "-32767",-32767,"-2147483647",-2147483647,NA,
                  NaN,"NA","NaN","9969209968386869046778552952102584320.000",
                  "9.96920996838686E+36","99.0","9999.0","999","999.0","99.00","999.00","", " ")
    # format header
    read_header <- function(df){
        years <- df[1:3]
        freq <- df[4:length(df)]
        freq <- as.numeric(freq)
        freq <- sprintf("%1.4f", freq)
        freq <- as.character(freq)
        df <- c(years, freq)
    }
    # functions to test for date formats
    library(lubridate)
    IsDate_dmy_hms <- function(mydate, date.format = "%d-%m-%Y %h:%m:%s") {
      tryCatch(!is.na(dmy_hms(mydate, date.format)),
               error = function(err) {FALSE})
    }
    IsDate_ymd_hms <- function(mydate, date.format = "%Y-%m-%d %h:%m:%s") {
      tryCatch(!is.na(ymd_hms(mydate, date.format)),
               error = function(err) {FALSE})
    }
    IsDate_mdy_hm <- function(mydate, date.format = "%m/%d/%Y %h:%m") {
      tryCatch(!is.na(mdy_hm(mydate, date.format)),
               error = function(err) {FALSE})
    }

    setwd(data_dir)

    raw_dir <- paste0(data_dir,"raw_data/")
    setwd(data_dir)
    # set input directories
    unzip_ndbc_dir <- paste0(raw_dir,"ndbc/unzipped/")
    ascii_ncei_dir <- paste0(raw_dir,"ncei/ascii/")

    # set new out_dir
    if (!file.exists(paste0(data_dir,"concat_data/"))) {dir.create((paste0(data_dir,"concat_data/")))}
    out_dir <- paste0(data_dir,"concat_data/")
    # ndbc
    if (!file.exists(paste0(out_dir,"ndbc/"))) {dir.create((paste0(out_dir,"ndbc/")))}
    ndbc_dir <- paste0(out_dir,"ndbc/")
    # ncei
    if (!file.exists(paste0(out_dir,"ncei/"))) {dir.create((paste0(out_dir,"ncei/")))}
    ncei_dir <- paste0(out_dir,"ncei/")

    # data types
```

```r
dataTypes <- c("stdmet", "swden", "swdir", "swdir2", "swr1","swr2")
dataType_ab <- c("h", "w", "d", "i","j","k")
data_types_stdmet <- c("lat", "lon", "wind_direction", "wind_speed", "wind_gust",
                       "significant_wave_height", "dominant_period", "average_period", "mean_wave_direction",
                       "air_pressure", "air_temperature", "sea_surface_temperature", "dew_point_temperature")
data_types_spec <- c("c11", "c11m", "alpha1", "alpha2", "r1", "r2", "C12", "C13", "C22", "C33",
                     "Q12", "Q13", "gamma2", "gamma3", "phih", "rhq", "sensor_output") # c11 = spectral energy,
                     c11m = uncorrected spectral energy


if(input == "stdmet"){
      #-------------------------------------------------------------------------------------------
      # met data
      #-------------------------------------------------------------------------------------------
      # # merging yearly and monthly datafiles of each type
      library(lubridate)
      library(tidyr)
      # standard met data - create NCEI matching NDBC web file nomenclature
      for (t in data_types_stdmet){
          print(paste0("Working on...",t))


          if (!file.exists(paste0(ncei_dir,buoy,"/data_availability")))
          {dir.create((paste0(ncei_dir,buoy,"/data_availability")))}
          data_avail_dir <- paste0(ncei_dir,buoy,"/data_availability/s_")

          # add sensor information
          metadata <- read.csv(paste0(raw_dir,"ncei/metadata/",buoy,"_metadata_ALL.csv"))

          # subset conditions for each variable - because NBDC loves duplication and format change for no reason
          # handle different netcdf variable names for waves
          if(grepl("dominant", t)){t1 <- "dominant_wave_period"; files2 <- file_list[grep(pattern = paste0("_",t1),
          file_list)]}
          if(grepl("average", t)){t1 <- "average_wave_period"; files2 <- file_list[grep(pattern = paste0("_",t1),
          file_list)]}
          files <- file_list[grep(pattern = paste0("_",t), file_list)]
          if(exists("files2")){files <- c(files2, files); rm(files2)}

          # available files ALL
          files <- sort(files)
          df_files <- gsub(paste0(ascii_ncei_dir, buoy,"/",buoy,"_"),"",files)
          df_files <- df_files[order(files)]
          df_files <- unique(df_files)
          df_files <- data.frame(df_files, stringsAsFactors = F)
          df_files <- separate(df_files, df_files, into = c('date', 'file'),sep = "/", remove = TRUE)
          write.csv(df_files, paste0(data_avail_dir,buoy, "_",t,"_available_files_all.csv"), row.names=FALSE)

          if(t == "lat"){
                # used:
                df_files <- gsub(paste0(ascii_ncei_dir, buoy,"/",buoy,"_"),"",files)
```

```
        df_files <- df_files[order(files)]
        df_files <- unique(df_files)
        df_files <- data.frame(df_files, stringsAsFactors = F)
        df_files <- separate(df_files, df_files, into = c('date', 'file'),sep = "/", remove = TRUE)
        write.csv(df_files, paste0(data_avail_dir,buoy, "_",t,"_available_files_USED.csv"), row.names=FALSE)
        rm(df_files)

    }else if(t == "lon"){
        # removed:
        variable2 <- files[grep(pattern = "_longwave_", files)]
        if(length(variable2) != 0){files <- files[!grepl(paste(variable2, collapse="|"), files)]}
        # used:
        df_files <- gsub(paste0(ascii_ncei_dir, buoy,"/",buoy,"_"),"",files)
        df_files <- df_files[order(files)]
        df_files <- unique(df_files)
        df_files <- data.frame(df_files, stringsAsFactors = F)
        df_files <- separate(df_files, df_files, into = c('date', 'file'),sep = "/", remove = TRUE)
        write.csv(df_files, paste0(data_avail_dir,buoy, "_",t,"_available_files_USED.csv"), row.names=FALSE)
        rm(df_files)

    }else if(t == "wind_direction"){
        # removed:
        continuous <- files[grep(pattern = paste0("_continuous_",t), files)]
        if(length(continuous) != 0){files <- files[!grepl(paste(continuous, collapse="|"), files)]}
        wind_dir_58 <- files[grep(pattern = "_58", files)]
        if(length(wind_dir_58) != 0){files <- files[!grepl(paste(wind_dir_58, collapse="|"), files)]}
        # isolate secondary files
        files_secondary <- files[grep(pattern = "_2_", files)]
        if(length(files_secondary) != 0){if(unique(files == files_secondary)==TRUE){files_secondary <- NULL}}
        if(length(files_secondary) != 0){files <- files[!grepl(paste(files_secondary, collapse="|"), files)]}
        if(length(files_secondary) == 0){rm(files_secondary)}
        # used:
        df_files <- gsub(paste0(ascii_ncei_dir, buoy,"/",buoy,"_"),"",files)
        df_files <- df_files[order(files)]
        df_files <- unique(df_files)
        df_files <- data.frame(df_files, stringsAsFactors = F)
        df_files <- separate(df_files, df_files, into = c('date', 'file'),sep = "/", remove = TRUE)
        write.csv(df_files, paste0(data_avail_dir,buoy, "_",t,"_available_files_USED.csv"), row.names=FALSE)
        rm(df_files)

        # filter metadata
        metadata <- dplyr::filter(metadata, grepl("anemometer",payload_sensor))

    }else if(t == "wind_speed"){
        # removed:
        continuous <- files[grep(pattern = paste0("_continuous_",t), files)]
        if(length(continuous) != 0){files <- files[!grepl(paste(continuous, collapse="|"), files)]}
        max_1 <- files[grep(pattern = "_max_1", files)]
        if(length(max_1) != 0){files <- files[!grepl(paste(max_1, collapse="|"), files)]}
        peak <- files[grep(pattern = "_peak_", files)]
```

```
        if(length(peak) != 0){files <- files[!grepl(paste(peak, collapse="|"), files)]}
        wind_speed_58 <- files[grep(pattern = "_58", files)]
        if(length(wind_speed_58) != 0){files <- files[!grepl(paste(wind_speed_58, collapse="|"), files)]}
        # isolate secondary files
        files_secondary <- files[grep(pattern = "_2_", files)]
        if(length(files_secondary) != 0){if(unique(files == files_secondary)==TRUE){files_secondary <- NULL}}
        if(length(files_secondary) != 0){files <- files[!grepl(paste(files_secondary, collapse="|"), files)]}
        if(length(files_secondary) == 0){rm(files_secondary)}
        # used:
        df_files <- gsub(paste0(ascii_ncei_dir, buoy,"/",buoy,"_"),"",files)
        df_files <- df_files[order(files)]
        df_files <- unique(df_files)
        df_files <- data.frame(df_files, stringsAsFactors = F)
        df_files <- separate(df_files, df_files, into = c('date', 'file'),sep = "/", remove = TRUE)
        write.csv(df_files, paste0(data_avail_dir,buoy, "_",t,"_available_files_USED.csv"), row.names=FALSE)
        rm(df_files)

        # filter metadata
        metadata <- dplyr::filter(metadata, grepl("anemometer",payload_sensor))

    }else if(t == "wind_gust"){
        # removed:
        ave <- files[grep(pattern = paste0(t, "_averaging_period"), files)]
        if(length(ave) != 0){files <- files[!grepl(paste(ave, collapse="|"), files)]}
        windgust_2 <- files[grep(pattern = "_wind_gust_2", files)]
        if(length(windgust_2) != 0){files <- files[!grepl(paste(windgust_2, collapse="|"), files)]}
        wind_gust_58 <- files[grep(pattern = "_58", files)]
        if(length(wind_gust_58) != 0){files <- files[!grepl(paste(wind_gust_58, collapse="|"), files)]}
        # isolate secondary files
        files_secondary <- files[grep(pattern = "_2_", files)]
        if(length(files_secondary) != 0){if(unique(files == files_secondary)==TRUE){files_secondary <- NULL}}
        if(length(files_secondary) != 0){files <- files[!grepl(paste(files_secondary, collapse="|"), files)]}
        if(length(files_secondary) == 0){rm(files_secondary)}
        # used:
        df_files <- gsub(paste0(ascii_ncei_dir, buoy,"/",buoy,"_"),"",files)
        df_files <- df_files[order(files)]
        df_files <- unique(df_files)
        df_files <- data.frame(df_files, stringsAsFactors = F)
        df_files <- separate(df_files, df_files, into = c('date', 'file'),sep = "/", remove = TRUE)
        write.csv(df_files, paste0(data_avail_dir,buoy, "_",t,"_available_files_USED.csv"), row.names=FALSE)
        rm(df_files)

        # filter metadata
        metadata <- dplyr::filter(metadata, grepl("anemometer",payload_sensor))

    }else if(t == "dominant_period"){
        # isolate secondary files
        files_secondary <- files[grep(pattern = "_2_", files)]
        if(length(files_secondary) != 0){if(unique(files == files_secondary)==TRUE){files_secondary <- NULL}}
        if(length(files_secondary) != 0){files <- files[!grepl(paste(files_secondary, collapse="|"), files)]}
```

```
            if(length(files_secondary) == 0){rm(files_secondary)}
            # used:
            df_files <- gsub(paste0(ascii_ncei_dir, buoy,"/",buoy,"_"),"",files)
            df_files <- df_files[order(files)]
            df_files <- unique(df_files)
            df_files <- data.frame(df_files, stringsAsFactors = F)
            df_files <- separate(df_files, df_files, into = c('date', 'file'),sep = "/", remove = TRUE)
            write.csv(df_files, paste0(data_avail_dir,buoy, "_",t,"_available_files_USED.csv"), row.names=FALSE)
            rm(df_files)

            # filter metadata
            metadata <- dplyr::filter(metadata, grepl("wave_sensor",payload_sensor))

        }else if(t == "average_period"){
            # isolate secondary files
            files_secondary <- files[grep(pattern = "_2_", files)]
            if(length(files_secondary) != 0){if(unique(files == files_secondary)==TRUE){files_secondary <- NULL}}
            if(length(files_secondary) != 0){files <- files[!grepl(paste(files_secondary, collapse="|"), files)]}
            if(length(files_secondary) == 0){rm(files_secondary)}
            # used:
            df_files <- gsub(paste0(ascii_ncei_dir, buoy,"/",buoy,"_"),"",files)
            df_files <- df_files[order(files)]
            df_files <- unique(df_files)
            df_files <- data.frame(df_files, stringsAsFactors = F)
            df_files <- separate(df_files, df_files, into = c('date', 'file'),sep = "/", remove = TRUE)
            write.csv(df_files, paste0(data_avail_dir,buoy, "_",t,"_available_files_USED.csv"), row.names=FALSE)
            rm(df_files)

            # filter metadata
            metadata <- dplyr::filter(metadata, grepl("wave_sensor",payload_sensor))

        }else if(t == "air_pressure"){
            # isolate secondary files
            files_secondary <- files[grep(pattern = "_2_", files)]
            if(length(files_secondary) != 0){if(unique(files == files_secondary)==TRUE){files_secondary <- NULL}}
            if(length(files_secondary) != 0){files <- files[!grepl(paste(files_secondary, collapse="|"), files)]}
            if(length(files_secondary) == 0){rm(files_secondary)}
            # # used:
            df_files <- gsub(paste0(ascii_ncei_dir, buoy,"/",buoy,"_"),"",files)
            df_files <- df_files[order(files)]
            df_files <- unique(df_files)
            df_files <- data.frame(df_files, stringsAsFactors = F)
            df_files <- separate(df_files, df_files, into = c('date', 'file'),sep = "/", remove = TRUE)
            write.csv(df_files, paste0(data_avail_dir,buoy, "_",t,"_available_files_USED.csv"), row.names=FALSE)
            rm(df_files)

            # filter metadata
            metadata <- dplyr::filter(metadata, grepl("barometer",payload_sensor))

        }else if(t == "air_temperature"){
```

```
                # removing dew point temperature
                dew <- files[grep(pattern = "_dew_", files)]
                if(length(dew) != 0){files <- files[!grepl(paste(dew, collapse="|"), files)]}
                # isolate secondary files
                files_secondary <- files[grep(pattern = "_2_", files)]
                if(length(files_secondary) != 0){if(unique(files == files_secondary)==TRUE){files_secondary <- NULL}}
                if(length(files_secondary) != 0){files <- files[!grepl(paste(files_secondary, collapse="|"), files)]}
                if(length(files_secondary) == 0){rm(files_secondary)}
                # used:
                df_files <- gsub(paste0(ascii_ncei_dir, buoy,"/",buoy,"_"),"",files)
                df_files <- df_files[order(files)]
                df_files <- unique(df_files)
                df_files <- data.frame(df_files, stringsAsFactors = F)
                df_files <- separate(df_files, df_files, into = c('date', 'file'),sep = "/", remove = TRUE)
                write.csv(df_files, paste0(data_avail_dir,buoy, "_",t,"_available_files_USED.csv"), row.names=FALSE)
                rm(df_files)

                # filter metadata
                metadata <- dplyr::filter(metadata, grepl("air_temperature",payload_sensor))

        }else { # significant wave height, mean wave direction, sea surface temperature, dew point temperature
                # # isolate secondary files
                files_secondary <- files[grep(pattern = "_2_", files)]
                if(length(files_secondary) != 0){if(unique(files == files_secondary)==TRUE){files_secondary <- NULL}}
                if(length(files_secondary) != 0){files <- files[!grepl(paste(files_secondary, collapse="|"), files)]}
                if(length(files_secondary) == 0){rm(files_secondary)}
                # used:
                df_files <- gsub(paste0(ascii_ncei_dir, buoy,"/",buoy,"_"),"",files)
                df_files <- df_files[order(files)]
                df_files <- unique(df_files)
                df_files <- data.frame(df_files, stringsAsFactors = F)
                df_files <- separate(df_files, df_files, into = c('date', 'file'),sep = "/", remove = TRUE)
                write.csv(df_files, paste0(data_avail_dir,buoy, "_",t,"_available_files_USED.csv"), row.names=FALSE)
                rm(df_files)

                if(t == "dew_point_temperature"){metadata <- dplyr::filter(metadata,
                grepl("air_temperature",payload_sensor))
                }else if(t == "sea_surface_temperature"){metadata <- dplyr::filter(metadata,
                grepl("ocean_temperature",payload_sensor))
                }else {metadata <- dplyr::filter(metadata, grepl("wave_sensor",payload_sensor))}
        }
        print(head(files))
        print(tail(files))

        if(length(files)>0){
            # loop for each data type
            if(exists("files_secondary")){
              data_sets <- c("files", "files_secondary")
            } else{
              data_sets <- "files"
```

```
293                 }
294                 # order metadata file
295                 if(dim(metadata)[1]>0){
296                     metadata <- metadata[order(metadata$file),]
297                     # rename the rows to reflect unique data
298                     row.names(metadata) <- 1:nrow(metadata)
299                 }
300                 for (df in data_sets){
301                     df_2 <- get(df)
302                     for (file in df_2){
303                           print(file)
304                           # if the merged dataset doesn't exist, create it
305                           if (!exists("dataset")){
306                                 dataset <- read.table(file, header=TRUE, na.strings = NA_strings, fill = T, sep =
                                    ",", stringsAsFactors = FALSE)
307                                 # set date format
308                                 if(IsDate_dmy_hms(dataset[1,1])==TRUE){dataset[,1] <- dmy_hms(dataset[,1]);
                                    print("dmy_hms dates converted")}
309                                 if(IsDate_ymd_hms(dataset[1,1])==TRUE){dataset[,1] <- ymd_hms(dataset[,1]);
                                    print("ymd_hms dates converted")}
310                                 if(IsDate_mdy_hm(dataset[1,1])==TRUE){dataset[,1] <- mdy_hm(dataset[,1]);
                                    print("mdy_hm dates converted")}
311
312                                 # reformatting the data structure
313                                 if("qc_flag" %in% names(dataset)){print("qc data available")}else{dataset$qc_flag <-
                                    NA}
314                                 if("release_flag" %in% names(dataset)){print("release_flag
                                    available")}else{dataset$release_flag <- NA}
315
316                                 if(dim(dataset)[2]>4){
317                                     if(t == "lat"){dataset <- dplyr::select(dataset, time, lat, qc_flag, release_flag)}
318                                     if(t == "lon"){dataset <- dplyr::select(dataset, time, lon, qc_flag, release_flag)}
319                                 }
320                                 # converting Kelvin to deg C if necessary
321                                 if(t != "lat" & t != "lon"){
322                                     if("temperature" %in% unlist(strsplit(t,"_"))){
323                                       dataset[,4] <- ifelse(dataset[,4] >= 100,
324                                                             dataset[,4] - 273.15,
325                                                             dataset[,4])
326                                       print("temperature converted from Kelvin to C")
327                                     }
328                                     # converting pressure to bars if necessary
329                                     if("pressure" %in% unlist(strsplit(t,"_"))){
330                                       dataset[,4] <- ifelse(dataset[,4] > 10000,
331                                                             dataset[,4] / 100,
332                                                             dataset[,4])
333                                       print("pressure convert to bars")
334                                     }
335                                     if("lat" %in% names(dataset)){dataset <- dplyr::select(dataset, -"lat");
                                        print("lat removed before merge")}
```

```
                    if("lon" %in% names(dataset)){dataset <- dplyr::select(dataset, -"lon");
                    print("lon removed before merge")}
                }
                colnames(dataset) <- c("DateTime", t, "qc_flag", "release_flag")
                # save only unique data
                dataset <- unique(dataset)
                # add sensor information
                dataset$payload_sensor <-
                gsub(".csv","",unlist(strsplit(file,"/"))[length(unlist(strsplit(file,"/")))])
            }
            # if the merged dataset does exist, append to it
            if (exists("dataset")){
                temp_dataset <-read.table(file, header=TRUE, na.strings = NA_strings, fill = T, sep =
                ",", stringsAsFactors = FALSE)

                # set date format
                if(IsDate_dmy_hms(temp_dataset[1,1])==TRUE){temp_dataset[,1] <-
                dmy_hms(temp_dataset[,1]); print("dmy_hms dates converted")}
                if(IsDate_ymd_hms(temp_dataset[1,1])==TRUE){temp_dataset[,1] <-
                ymd_hms(temp_dataset[,1]); print("ymd_hms dates converted")}
                if(IsDate_mdy_hm(temp_dataset[1,1])==TRUE){temp_dataset[,1] <-
                mdy_hm(temp_dataset[,1]); print("mdy_hm dates converted")}

                # reformatting the data structure
                if("QC_Flags" %in% names(temp_dataset)){temp_dataset <- dplyr::rename(temp_dataset,
                "qc_flag" = "QC_Flags")}
                if("qc_flag" %in% names(temp_dataset)){print("qc data
                available")}else{temp_dataset$qc_flag <- NA}
                if("release_flag" %in% names(temp_dataset)){print("release_flag
                available")}else{temp_dataset$release_flag <- NA}

                if(dim(temp_dataset)[2]>4){
                    if(t == "lat"){temp_dataset <- dplyr::select(temp_dataset, time, lat, qc_flag,
                    release_flag)}
                    if(t == "lon"){temp_dataset <- dplyr::select(temp_dataset, time, lon, qc_flag,
                    release_flag)}
                }
                # converting Kelvin to deg C if necessary
                if(t != "lat" & t != "lon"){
                    if("temperature" %in% unlist(strsplit(t,"_"))){
                      temp_dataset[,4] <- ifelse(temp_dataset[,4] > 100,
                                                 temp_dataset[,4] - 273.15,
                                                 temp_dataset[,4])
                      print("temperature converted from Kelvin to C")
                    }
                    # converting pressure to bars if necessary
                    if("pressure" %in% unlist(strsplit(t,"_"))){
                      temp_dataset[,4] <- ifelse(temp_dataset[,4] > 10000,
                                                 temp_dataset[,4] / 100,
                                                 temp_dataset[,4])
```

```
                        print("pressure convert to bars")
                    }
                    if("lat" %in% names(temp_dataset)){temp_dataset <- dplyr::select(temp_dataset,
                    -"lat"); print("lat removed before merge")}
                    if("lon" %in% names(temp_dataset)){temp_dataset <- dplyr::select(temp_dataset,
                    -"lon"); print("lon removed before merge")}
                }
                colnames(temp_dataset) <- c("DateTime", t, "qc_flag", "release_flag")
                 # save only unique data
                temp_dataset <- unique(temp_dataset)
                # add sensor information
                temp_dataset$payload_sensor <-
                gsub(".csv","",unlist(strsplit(file,"/"))[length(unlist(strsplit(file,"/")))])
                # merge datasets
                dataset<-rbind(dataset, temp_dataset)
                rm(temp_dataset)
        }
    }
    # ordering the dataset by date and selecting unique values only
    dataset <- dataset[order(dataset$DateTime),]
    dataset <- unique(dataset)
    # rename the rows to reflect unique data
    row.names(dataset) <- 1:nrow(dataset)
    # for saving
    year_range <- paste0(year(dataset$DateTime[1]),"_", year(dataset$DateTime[nrow(dataset)]))
    # sep primary and secondary data
    if(df == "files"){concat_name <- paste0(buoy, "_",t,"_",year_range,"_concat_primary")}
    if(df == "files_secondary"){concat_name <- paste0(buoy, "_",t,"_",year_range,"_concat_secondary")}
    # save individual datasets
    library(lubridate)
    # write.csv(dataset, paste0(concat_ind_dir,concat_name,".csv"), row.names=FALSE)
    # saveRDS(dataset, file = paste0(concat_ind_dir,concat_name,".rds"))

    # remove flagged data
    # Looping through all NDBC QC flags and removing flagged raw data
    dataset[,3][is.na(dataset[,3])] <- 0                    # replace NA with zero when flags not
    included in original dataset
    dataset[,4][is.na(dataset[,4])] <- 0
    flags <- c("W", "R", "V", "M", "T", "D", "U", "L", "H", "2", "3","_")   # ignore S flags
    for (i in flags){
        # i <- flags[1]
        index <- dataset$qc_flag == i
        dataset[index,2] <- NA
        # dataset <- subset(dataset, qc_flag != i)
    }
    # order for released data and remove duplicated date rows
    if(nrow(dataset[!duplicated(dataset$DateTime),]) != nrow(dataset)){
        if(length(unique(dataset$release_flag))>2){
              dataset = dataset[order(dataset['DateTime'],-dataset['release_flag']),]
              dataset = dataset[!duplicated(dataset$DateTime),]
```

```
                    print("removing duplicated data based on release flag")
                }else{
                    dataset = dataset[!duplicated(dataset$DateTime),]
                    print("removing duplicated data NOT based on release flag")
                }
            }

            # remove redundant columns
            library(dplyr)
            dataset <- dplyr::select(dataset,-"qc_flag")
            dataset <- dplyr::select(dataset,-"release_flag")

            # sep primary and secondary data
            if(df == "files"){dataset1 <- dataset}
            if(df == "files_secondary"){dataset2 <- dataset}
            rm(dataset)
        } # loop through datasets

        # loop through multiple payloads
        if(length(data_sets) > 1){
            if(dim(metadata)[1]>0){
                # prepping the metadata
                metadata1 <- metadata
                metadata1$join <- gsub("/","_",paste0(gsub(".nc","",metadata1$file),
                "_",metadata1$payload_sensor,"_",t))
                metadata1 <- metadata1[,7:ncol(metadata1)]
                # handling different air pressure nomenclature
                if(t == "air_pressure"){
                  metadata1$join <- gsub("_air_pressure_at_sea_level","_air_pressure", metadata1$join)
                  dataset1$payload_sensor <- gsub("_air_pressure_at_sea_level","_air_pressure",
                  dataset1$payload_sensor)
                  dataset2$payload_sensor <- gsub("_air_pressure_at_sea_level","_air_pressure",
                  dataset2$payload_sensor)
                }
                # add primary sensor information
                dataset1 <- dplyr::rename(dataset1, join = payload_sensor)
                dataset1 <- left_join(dataset1, metadata1, by = "join")
                # add secondary sensor information
                dataset2 <- dplyr::rename(dataset2, join = payload_sensor)
                dataset2 <- left_join(dataset2, metadata1, by = "join")
                # rename columns
                colnames(dataset1) <- c("DateTime", paste0(colnames(dataset1[2:ncol(dataset1)]), "_1"))
                colnames(dataset2) <- c("DateTime", paste0(colnames(dataset2[2:ncol(dataset2)]), "_2"))
                rm(metadata, metadata1)
            }else{
                dataset$payload_sensor <- "no information available"
                colnames(dataset) <- c("DateTime", t, paste0(t,"_payload_sensor"))
            }
            # Prep for secondary sensor data if primary sensor data is NA
            dataset <- full_join(dataset1, dataset2, by = "DateTime")
```

```
rm(dataset1,dataset2)
# rename columns
colnames(dataset) = gsub("\\.x", "_1", colnames(dataset))
colnames(dataset) = gsub("\\.y", "_2", colnames(dataset))
# check for duplicates
dataset <- dataset[order(dataset$DateTime),]
dataset <- unique(dataset)
# rename the rows to reflect unique data
row.names(dataset) <- 1:nrow(dataset)
# export data
year_range <- paste0(year(min(dataset$DateTime,na.rm = TRUE)),"_", year(max(dataset$DateTime,
na.rm = TRUE)))
# write.csv(dataset, paste0(ncei_dir,buoy,"/s_",buoy,
"_ncei_",t,"_primary_secondary_",year_range,".csv"), row.names=FALSE)
# saveRDS(dataset, file =
paste0(ncei_dir,buoy,"/s_",buoy,"_ncei_",t,"_primary_secondary_",year_range,".rds"))

# concat metadata
dataset$metadata_1 <-
paste0(dataset[,3],"__",dataset[,4],"__",dataset[,5],"__",dataset[,7],"__",dataset[,8],"__",dataset
[,9],"__",dataset[,10],"__",dataset[,11],"__",dataset[,12])
dataset$metadata_1 <- gsub("__NA","",dataset$metadata_1)
dataset$metadata_2 <-
paste0(dataset[,14],"__",dataset[,15],"__",dataset[,16],"__",dataset[,18],"__",dataset[,19],"__",da
taset[,20],"__",dataset[,21],"__",dataset[,22],"__",dataset[,23])
dataset$metadata_2 <- gsub("__NA","",dataset$metadata_2)
# subset relevant data
dataset <- dataset[,c(1,2,24,13,25)]
# rename columns
if(dim(dataset)[2]==3){
      colnames(dataset)<-c("DateTime",t,paste0(t, "_metadata"))
}else if(dim(dataset)[2]==2){
      if(t %in% names(dataset)){print("")
      }else{dataset$dat <- NA; dat_names <- names(dataset); dataset <- dplyr::select(dataset,
      DateTime, dat,dat_names[2])
            colnames(dataset)<-c("DateTime",t,paste0(t, "_metadata"))
      }
}else{
      colnames(dataset) <- c("DateTime", paste0(t,"_1"), paste0(t,"_metadata_1"),paste0(t,"_2"),
      paste0(t,"_metadata_2"))
}
# export data
year_range <- paste0(year(min(dataset$DateTime,na.rm = TRUE)),"_", year(max(dataset$DateTime,
na.rm = TRUE)))
# write.csv(dataset, paste0(ncei_dir,buoy,"/s_",buoy,
"_ncei_",t,"_concat_primary_secondary_metadata_",year_range,".csv"), row.names=FALSE)
# saveRDS(dataset, file =
paste0(ncei_dir,buoy,"/s_",buoy,"_ncei_",t,"_concat_primary_secondary_metadata_",year_range,".rds")
)
# housekeeping
```

```
505                     rm(dataset1, dataset2, metadata1, metadata2)
506                 }else{
507                     dataset <- dataset1
508                     # add sensor information
509                     if(dim(metadata)[1]>0){
510                           metadata1 <- metadata
511                           metadata1$join <- gsub("/","_",paste0(gsub(".nc","",metadata1$file),
                              "_",metadata1$payload_sensor,"_",t))
512                           metadata1 <- metadata1[,7:ncol(metadata1)]
513                           # handling different air pressure nomenclature
514                           if(t == "air_pressure"){
515                             metadata1$join <- gsub("_air_pressure_at_sea_level","_air_pressure", metadata1$join)
516                             dataset$payload_sensor <- gsub("_air_pressure_at_sea_level","_air_pressure",
                                dataset$payload_sensor)
517                           }
518                           # add primary sensor information
519                           dataset <- dplyr::rename(dataset, join = payload_sensor)
520                           dataset <- left_join(dataset, metadata1, by = "join")
521                           # export data
522                           year_range <- paste0(year(min(dataset$DateTime,na.rm = TRUE)),"_",
                              year(max(dataset$DateTime, na.rm = TRUE)))
523                           # write.csv(dataset, paste0(ncei_dir,buoy,"/s_",buoy,
                              "_ncei_",t,"_primary_",year_range,".csv"), row.names=FALSE)
524                           # saveRDS(dataset, file =
                              paste0(ncei_dir,buoy,"/s_",buoy,"_ncei_",t,"_primary_",year_range,".rds"))
525
526                           # convert for concat
527                           dataset$join <- gsub(paste0("_",t),"",dataset$join)
528                           dataset$metadata <-
                              paste0(dataset[,3],"__",dataset[,4],"__",dataset[,5],"__",dataset[,7],"__",dataset[,8],"__",d
                              ataset[,9],"__",dataset[,10],"__",dataset[,11],"__",dataset[,12])
529                           dataset$metadata <- gsub("__NA","",dataset$metadata)
530                           dataset <- dataset[,c(1,2,13)]
531                           # rename columns
532                           colnames(dataset) <- c("DateTime", t, paste0(t,"_metadata"))
533                           rm(dataset1, metadata1)
534                     }else{
535                           dataset$payload_sensor <- "no information available"
536                           colnames(dataset) <- c("DateTime", t, paste0(t,"_metadata"))
537                     }
538                     rm(dataset1)
539                 }
540                 # removing duplicated date/times added from secondary dataset (accounts for slightly different data
                    values caused by significant places)
541                 dataset <- dataset[!duplicated(dataset$DateTime),]
542                 # ordering the dataset by date and selecting unique values only
543                 dataset <- dataset[order(dataset$DateTime),]
544                 dataset <- unique(dataset)
545                 # rename the rows to reflect unique data
546                 row.names(dataset) <- 1:nrow(dataset)
```

```
                # concat files
                if (!exists("ncei_concat_stdmet")){
                    ncei_concat_stdmet <- dataset
                }else{
                    ncei_concat_stdmet<-dplyr::full_join(ncei_concat_stdmet, dataset, by = "DateTime")
                    ncei_concat_stdmet <- ncei_concat_stdmet[order(ncei_concat_stdmet$DateTime),]
                }
                rm(dataset)
            }else{print(paste("no data for ",t))}
        } # end of concat data loop

        dataset <- ncei_concat_stdmet
        # # save dataset
        # year_range <- paste0(year(dataset$DateTime[1]),"_", year(dataset$DateTime[nrow(dataset)]))
        # saveRDS(dataset, file = paste0(ncei_dir,buoy,"/s_",buoy,"_ncei_concat_stdmet_",year_range,"_v1.rds"))

        # remove '0' wind gust data when no wind speed is present (not flagged in orig files)
        name_list <- names(dataset)
        if("lat_1" %in% name_list){
            dataset$lat_2 <- NULL; dataset$lat_metadata_2 <- NULL; dataset$lon_2 <- NULL; dataset$lon_metadata_2 <-
            NULL
            dataset <- dplyr::rename(dataset, lat = lat_1, lon = lon_1,lat_metadata = lat_metadata_1, lon_metadata =
            lon_metadata_1 )
        }
        # account for erroneous data
        if("wind_gust" %in% name_list){
          library(stringr)
          if(sum(str_count(name_list,"wind_gust"))>2){ # accounts for wind_gust and wind_gust_metadata
              dataset$wind_gust_1 <- ifelse(dataset$wind_gust_1 == 0 & dataset$wind_speed_1 != 0,NA,
              dataset$wind_gust_1)
              dataset$wind_gust_2 <- ifelse(dataset$wind_gust_2 == 0 & dataset$wind_speed_2 != 0,NA,
              dataset$wind_gust_2)
          }else{dataset$wind_gust <- ifelse(dataset$wind_gust == 0 & dataset$wind_speed != 0,NA, dataset$wind_gust)}
        }
        assign("dataset", dataset, envir=parent.frame())

    }else if(input == "spec"){
        #-------------------------------------------------------------------------------------
        # spectral data -
        #-------------------------------------------------------------------------------------
        # data_types_spec <- c("c11", "c11m", "alpha1", "alpha2", "r1", "r2", "C12", "C13", "C22", "C33",
        #                      "Q12", "Q13", "gamma2", "gamma3", "phih", "rhq", "sensor_output") # c11 = spectral
        energy, c11m = uncorrected spectral energy

        # start_year <- 1979
        # create data avail dir and folder
        if (!file.exists(paste0(ncei_dir,buoy,"/data_availability")))
        {dir.create((paste0(ncei_dir,buoy,"/data_availability")))}
        data_avail_dir <- paste0(ncei_dir,buoy,"/data_availability/s_")
        print(paste0("Working on...",t))
```

```
        library(lubridate)
        library(tidyr)
        # subset conditions for each variable - because NBDC loves duplication and format change for no reason
        if(t == "c11"){   # I believe k is corrected
              # see what's available:
              files1 <- file_list[grep(pattern = "spectral_density", file_list)]
              files2 <- file_list[grep(pattern = t, file_list)]
              files3 <- file_list[grep(pattern = "C11", file_list)]
              files4 <- file_list[grep(pattern = "c11", file_list)]
              files <- c(files1, files2, files3, files4)
              c11m <- files[grep(pattern = "_c11m.csv", files)]
              if(length(c11m) != 0){files <- files[!grepl(paste(c11m, collapse="|"), files)]}
              df_files <- gsub(paste0(ascii_ncei_dir, buoy,"/",buoy,"_"),"",files)
              df_files <- df_files[order(files)]
              df_files <- unique(df_files)
              df_files <- data.frame(df_files, stringsAsFactors = F)
              df_files <- separate(df_files, df_files, into = c('date', 'file'),sep = "/", remove = TRUE)
              write.csv(df_files, paste0(data_avail_dir,buoy, "_c11_nondir_spectral_available_files_all.csv"),
              row.names=FALSE)
              # used:
              files1 <- file_list[grep(pattern = "spectral_density", file_list)]
              files2 <- file_list[grep(pattern = t, file_list)]
              files <- c(files1, files2)
              files <- sort(files)
              c11m <- files[grep(pattern = "_c11m.csv", files)]
              if(length(c11m) != 0){files <- files[!grepl(paste(c11m, collapse="|"), files)]}
              c11i <- files[grep(pattern = "_c11_i.csv", files)]
              if(length(c11i) != 0){files <- files[!grepl(paste(c11i, collapse="|"), files)] }
              # payload_2 <- files[grep(pattern = "_payload_2", files)]
              df_files <- gsub(paste0(ascii_ncei_dir, buoy,"/",buoy,"_"),"",files)
              df_files <- df_files[order(files)]
              df_files <- unique(df_files)
              df_files <- data.frame(df_files, stringsAsFactors = F)
              df_files <- separate(df_files, df_files, into = c('date', 'file'),sep = "/", remove = TRUE)
              write.csv(df_files, paste0(data_avail_dir,buoy, "_c11_nondir_spectral_available_files_USED.csv"),
              row.names=FALSE)

        } else if(t == "c11m"){  # I believe i is uncorrected
              # see what's available:
              files1 <- file_list[grep(pattern = "c11", file_list)]
              files2 <- file_list[grep(pattern = "C11", file_list)]
              files <- c(files1, files2)
              c11 <- files[grep(pattern = "_c11.csv", files)]
              if(length(c11) != 0){files <- files[!grepl(paste(c11, collapse="|"), files)]}
              df_files <- gsub(paste0(ascii_ncei_dir, buoy,"/",buoy,"_"),"",files)
              df_files <- df_files[order(files)]
              df_files <- unique(df_files)
              df_files <- data.frame(df_files, stringsAsFactors = F)
              df_files <- separate(df_files, df_files, into = c('date', 'file'),sep = "/", remove = TRUE)
              write.csv(df_files, paste0(data_avail_dir,buoy,
```

```
        "_c11m_uncorrected_nondir_spectral_available_files_all.csv"), row.names=FALSE)
        # used:
        files1 <- file_list[grep(pattern = "c11", file_list)]
        files2 <- file_list[grep(pattern = "C11", file_list)]
        files <- c(files1, files2)
        files <- sort(files)
        c11 <- files[grep(pattern = "_c11.csv", files)]
        if(length(c11) != 0){files <- files[!grepl(paste(c11, collapse="|"), files)]}
        c11k <- files[grep(pattern = "_c11_k.csv", files)]
        if(length(c11k) != 0){files <- files[!grepl(paste(c11k, collapse="|"), files)]}
        c11_i <- files[grep(pattern = "_c11_i.csv", files)]
        if(length(c11_i) != 0){files <- files[!grepl(paste(c11_i, collapse="|"), files)]}
        df_files <- gsub(paste0(ascii_ncei_dir, buoy,"/",buoy,"_"),"",files)
        df_files <- df_files[order(files)]
        df_files <- unique(df_files)
        df_files <- data.frame(df_files, stringsAsFactors = F)
        df_files <- separate(df_files, df_files, into = c('date', 'file'),sep = "/", remove = TRUE)
        write.csv(df_files, paste0(data_avail_dir,buoy,
        "_c11m_uncorrected_nondir_spectral_available_files_USED.csv"), row.names=FALSE)

    }else if(t == "alpha1"){
        # see what's available:
        files <- file_list[grep(pattern = t, file_list)]
        files <- sort(files)
        df_files <- gsub(paste0(ascii_ncei_dir, buoy,"/",buoy,"_"),"",files)
        df_files <- df_files[order(files)]
        df_files <- unique(df_files)
        df_files <- data.frame(df_files, stringsAsFactors = F)
        df_files <- separate(df_files, df_files, into = c('date', 'file'),sep = "/", remove = TRUE)
        write.csv(df_files, paste0(data_avail_dir,buoy, "_alpha1_available_files_all.csv"), row.names=FALSE)
        # used:
        files <- file_list[grep(pattern = t, file_list)]
        df_files <- gsub(paste0(ascii_ncei_dir, buoy,"/",buoy,"_"),"",files)
        df_files <- df_files[order(files)]
        df_files <- unique(df_files)
        df_files <- data.frame(df_files, stringsAsFactors = F)
        df_files <- separate(df_files, df_files, into = c('date', 'file'),sep = "/", remove = TRUE)
        write.csv(df_files, paste0(data_avail_dir,buoy, "_alpha1_available_files_USED.csv"), row.names=FALSE)

    }else if(t == "alpha2"){
        # see what's available:
        files <- file_list[grep(pattern = t, file_list)]
        files <- sort(files)
        df_files <- gsub(paste0(ascii_ncei_dir, buoy,"/",buoy,"_"),"",files)
        df_files <- df_files[order(files)]
        df_files <- unique(df_files)
        df_files <- data.frame(df_files, stringsAsFactors = F)
        df_files <- separate(df_files, df_files, into = c('date', 'file'),sep = "/", remove = TRUE)
        write.csv(df_files, paste0(data_avail_dir,buoy, "_alpha2_available_files_all.csv"), row.names=FALSE)
        # used:
```

```
            files <- file_list[grep(pattern = t, file_list)]
            df_files <- gsub(paste0(ascii_ncei_dir, buoy,"/",buoy,"_"),"",files)
            df_files <- df_files[order(files)]
            df_files <- unique(df_files)
            df_files <- data.frame(df_files, stringsAsFactors = F)
            df_files <- separate(df_files, df_files, into = c('date', 'file'),sep = "/", remove = TRUE)
            write.csv(df_files, paste0(data_avail_dir,buoy, "_alpha2_available_files_USED.csv"), row.names=FALSE)

        }else if(t == "r1"){
            # see what's available:
            files <- file_list[grep(pattern = t, file_list)]
            files <- sort(files)
            df_files <- gsub(paste0(ascii_ncei_dir, buoy,"/",buoy,"_"),"",files)
            df_files <- df_files[order(files)]
            df_files <- unique(df_files)
            df_files <- data.frame(df_files, stringsAsFactors = F)
            df_files <- separate(df_files, df_files, into = c('date', 'file'),sep = "/", remove = TRUE)
            write.csv(df_files, paste0(data_avail_dir,buoy, "_r1_available_files_all.csv"), row.names=FALSE)
            # used:
            files <- file_list[grep(pattern = t, file_list)]
            df_files <- gsub(paste0(ascii_ncei_dir, buoy,"/",buoy,"_"),"",files)
            df_files <- df_files[order(files)]
            df_files <- unique(df_files)
            df_files <- data.frame(df_files, stringsAsFactors = F)
            df_files <- separate(df_files, df_files, into = c('date', 'file'),sep = "/", remove = TRUE)
            write.csv(df_files, paste0(data_avail_dir,buoy, "_r1_available_files_USED.csv"), row.names=FALSE)

        }else if(t == "r2"){
            # see what's available:
            files <- file_list[grep(pattern = t, file_list)]
            files <- sort(files)
            df_files <- gsub(paste0(ascii_ncei_dir, buoy,"/",buoy,"_"),"",files)
            df_files <- df_files[order(files)]
            df_files <- unique(df_files)
            df_files <- data.frame(df_files, stringsAsFactors = F)
            df_files <- separate(df_files, df_files, into = c('date', 'file'),sep = "/", remove = TRUE)
            write.csv(df_files, paste0(data_avail_dir,buoy, "_r2_available_files_all.csv"), row.names=FALSE)
            # used:
            files <- file_list[grep(pattern = t, file_list)]
            df_files <- gsub(paste0(ascii_ncei_dir, buoy,"/",buoy,"_"),"",files)
            df_files <- df_files[order(files)]
            df_files <- unique(df_files)
            df_files <- data.frame(df_files, stringsAsFactors = F)
            df_files <- separate(df_files, df_files, into = c('date', 'file'),sep = "/", remove = TRUE)
            write.csv(df_files, paste0(data_avail_dir,buoy, "_r2_available_files_USED.csv"), row.names=FALSE)

        }else{ # "C12", "C13", "C22", "C33", "Q12", "Q13", "gamma2", "gamma3", "phih", "rhq", "sensor_output"
            # see what's available:
            files <- file_list[grep(pattern = t, file_list)]
            files <- sort(files)
```

```
737             df_files <- gsub(paste0(ascii_ncei_dir, buoy,"/",buoy,"_"),"",files)
738             df_files <- df_files[order(files)]
739             df_files <- unique(df_files)
740             df_files <- data.frame(df_files, stringsAsFactors = F)
741             df_files <- separate(df_files, df_files, into = c('date', 'file'),sep = "/", remove = TRUE)
742             write.csv(df_files, paste0(data_avail_dir,buoy, "_",t,"_available_files_all.csv"), row.names=FALSE)
743             # used:
744             files <- file_list[grep(pattern = t, file_list)]
745             df_files <- gsub(paste0(ascii_ncei_dir, buoy,"/",buoy,"_"),"",files)
746             df_files <- df_files[order(files)]
747             df_files <- unique(df_files)
748             df_files <- data.frame(df_files, stringsAsFactors = F)
749             df_files <- separate(df_files, df_files, into = c('date', 'file'),sep = "/", remove = TRUE)
750             write.csv(df_files, paste0(data_avail_dir,buoy, "_",t,"_available_files_USED.csv"), row.names=FALSE)
751         }
752         print(head(files))
753         print(tail(files))
754         rm(df_files)
755 
756         if(length(files)>0){
757             # loop for each data type
758             if(t == "sensor_output"){
759                   for (file in files){
760                         if (!exists("dataset")){
761                               dataset <- read.table(file, header=TRUE, na.strings = NA_strings, fill = T, sep = ",",
                                    stringsAsFactors = FALSE)
762                               # set date format
763                               if(IsDate_dmy_hms(dataset[1,1])==TRUE){dataset[,1] <- dmy_hms(dataset[,1]);
                                    print("dmy_hms dates converted")}
764                               if(IsDate_ymd_hms(dataset[1,1])==TRUE){dataset[,1] <- ymd_hms(dataset[,1]);
                                    print("ymd_hms dates converted")}
765                               if(IsDate_mdy_hm(dataset[1,1])==TRUE){dataset[,1] <- mdy_hm(dataset[,1]); print("mdy_hm
                                    dates converted")}
766                               dataset <- dplyr::select(dataset, -"lat")
767                               dataset <- dplyr::select(dataset, -"lon")
768                               dataset <- dplyr::rename(dataset, "DateTime" = "time")
769                         }
770                         # if the merged dataset does exist, append to it
771                         if (exists("dataset")){
772                               temp_dataset <-read.table(file, header=TRUE, na.strings = NA_strings, fill = T, sep =
                                    ",", stringsAsFactors = FALSE)
773                               # set date format
774                               if(IsDate_dmy_hms(temp_dataset[1,1])==TRUE){temp_dataset[,1] <-
                                    dmy_hms(temp_dataset[,1]); print("dmy_hms dates converted")}
775                               if(IsDate_ymd_hms(temp_dataset[1,1])==TRUE){temp_dataset[,1] <-
                                    ymd_hms(temp_dataset[,1]); print("ymd_hms dates converted")}
776                               if(IsDate_mdy_hm(temp_dataset[1,1])==TRUE){temp_dataset[,1] <-
                                    mdy_hm(temp_dataset[,1]); print("mdy_hm dates converted")}
777                               # temp_dataset[,1] <- ymd_hms(temp_dataset[,1])   # set date format
778                               temp_dataset <- dplyr::select(temp_dataset, -"lat")
```

```
                        temp_dataset <- dplyr::select(temp_dataset, -"lon")
                        colnames(temp_dataset) <- names(dataset)
                        # merge datasets
                        dataset<-rbind(dataset, temp_dataset)
                        rm(temp_dataset)
                    }
                }
                # ordering the dataset by date and selecting unique values only
                dataset <- dataset[order(dataset$DateTime),]
                dataset <- unique(dataset)
                # rename the rows to reflect unique data
                row.names(dataset) <- 1:nrow(dataset)
                # save individual datasets
                library(lubridate)
                year_range <- paste0(year(dataset$DateTime[1]),"_", year(dataset$DateTime[nrow(dataset)]))
                # write.csv(dataset, paste0(concat_ind_dir,buoy, "_",t,"_",year_range,"_concat.csv"), row.names=FALSE)
                # saveRDS(dataset, file = paste0(concat_ind_dir,buoy, "_",t,"_",year_range,"_concat.rds"))
                # remove redundant rows
                dataset <- unique(dataset)
                library(dplyr)
                dataset <- dplyr::select(dataset,-"qc_flag")
                dataset <- dplyr::select(dataset,-"release_flag")
                dataset <- dataset[rowSums(is.na(dataset)) != ncol(dataset)-1, ]
                assign("dataset", dataset, envir=parent.frame())

            }else{

                colNames_spec_new <- c("DateTime","lat","lon","0.0200", "0.0325", "0.0375", "0.0425", "0.0475", "0.0525",
                                       "0.0575", "0.0625", "0.0675", "0.0725", "0.0775", "0.0825", "0.0875", "0.0925",
                                       "0.1000", "0.1100",
                                       "0.1200", "0.1300", "0.1400", "0.1500", "0.1600", "0.1700", "0.1800", "0.1900",
                                       "0.2000", "0.2100",
                                       "0.2200", "0.2300", "0.2400", "0.2500", "0.2600", "0.2700", "0.2800", "0.2900",
                                       "0.3000", "0.3100",
                                       "0.3200", "0.3300", "0.3400", "0.3500", "0.3650", "0.3850", "0.4050", "0.4250",
                                       "0.4450", "0.4650",
                                       "0.4850")
                colNames_spec_old <-
                c("DateTime","lat","lon","0.0100","0.0200","0.0300","0.0400","0.0500","0.0600","0.0700","0.0800","0.0900"
                ,"0.1000","0.1100","0.1200",
                                       "0.1300","0.1400","0.1500","0.1600","0.1700","0.1800","0.1900","0.2000","0.2100","
                                       0.2200",
                                       "0.2300","0.2400","0.2500","0.2600","0.2700","0.2800","0.2900","0.3000","0.3100","
                                       0.3200",
                                       "0.3300","0.3400","0.3500","0.3600","0.3700","0.3800","0.3900","0.4000")

                dataset_spec_new <- data.frame(matrix(NA, nrow = 0, ncol = length(colNames_spec_new)))
                colnames(dataset_spec_new) <- colNames_spec_new
```

```
dataset_spec_new$DateTime <- lubridate::ymd_hms(dataset_spec_new$DateTime)

dataset_spec_old <- data.frame(matrix(NA, nrow = 0, ncol = length(colNames_spec_old)))
colnames(dataset_spec_old) <- colNames_spec_old
dataset_spec_old$DateTime <- lubridate::ymd_hms(dataset_spec_old$DateTime)

count <- 0

for (file in files){
      print(file)
      # load dataset
      dataset <- read.table(file, header=TRUE, na.strings = NA_strings, fill = T, sep = ",",
      stringsAsFactors = FALSE)
      # set date format
      if(IsDate_dmy_hms(dataset[1,1])==TRUE){dataset[,1] <- dmy_hms(dataset[,1]); print("dmy_hms dates
      converted")}
      if(IsDate_ymd_hms(dataset[1,1])==TRUE){dataset[,1] <- ymd_hms(dataset[,1]); print("ymd_hms dates
      converted")}
      if(IsDate_mdy_hm(dataset[1,1])==TRUE){dataset[,1] <- mdy_hm(dataset[,1]); print("mdy_hm dates
      converted")}
      library(lubridate)
      # remove the X from the column headers
      colnames(dataset) <-  sub("X", "", colnames(dataset))
      header <- names(dataset)
      header <- read_header(header)
      colnames(dataset) <- header
      if("time" %in% names(dataset)){dataset <- dplyr::rename(dataset,"DateTime" = "time")}
      if("lat" %in% names(dataset)){
            print("lat and lon present")
      }else{
            dataset$lat <- NA; dataset$lon <- NA
            print("adding lat and lon")
      }

      # find datasets
      dataset_list <- ls(pattern = "dataset_spec_")
      dataset_list <- dataset_list[dataset_list %in% c("dataset_spec_new","dataset_spec_old") == TRUE]
      print(dataset_list)
      # build datasets

      for(matchable in dataset_list){
          print(matchable)
          dat <- get(matchable)

          if(exists("dataset")){
                ndbc_freq <- names(dat)
                dat_names <- names(dataset)
                setdiff(ndbc_freq, dat_names)
                print(unique(dat_names %in% ndbc_freq))
                # test if columns match
```

```
                    if(any(unique(dat_names %in% ndbc_freq)==FALSE)){
                        print(paste0(matchable," didn't match"))
                    }else{
                        tryCatch({
                            library(plyr)
                            dat <- rbind.fill(dat,dataset)
                            print(paste0("added to: ",matchable))
                            assign(matchable,dat)
                            rm(dataset)
                        }, error = function(e) {
                            print("dataset doesn't match")
                        })
                    }
                }
                rm(dat)
            }
            if(exists("dataset")){
                dataset_list <- ls(pattern = "dataset_spec_")
                dataset_list <- dataset_list[dataset_list %in% c("dataset_spec_new","dataset_spec_old") ==
                FALSE]
                print(dataset_list)
                # build datasets

                for(matchable in dataset_list){
                    print(matchable)
                    dat <- get(matchable)

                    if(exists("dataset")){
                        ndbc_freq <- names(dat)
                        dat_names <- names(dataset)
                        setdiff(ndbc_freq, dat_names)
                        print(unique(dat_names %in% ndbc_freq))
                        # test if columns match
                        if(any(unique(dat_names %in% ndbc_freq)==FALSE)){
                            print(paste0(matchable," didn't match"))
                        }else{
                            tryCatch({
                                library(plyr)
                                dat <- rbind.fill(dat,dataset)
                                print(paste0("added to: ",matchable))
                                assign(matchable,dat)
                                rm(dataset)
                            },  error = function(e) {
                                print("dataset doesn't match")
                            })
                        }
                    }
                    rm(dat)
                }
            }
```

```
                    if(exists("dataset")){
                        count <- count + 1
                        dataset_spec <- data.frame(matrix(NA, nrow = 0, ncol = dim(dataset)[2]))
                        dataset_spec<-rbind(dataset_spec, dataset)
                        new_name <- paste0("dataset_spec_",count)
                        assign(new_name, dataset_spec)
                        print(paste0("data added to NEW DF:: ",new_name))
                        rm(dataset, dataset_spec)
                    }
                }

                # export from function
                dat_list <- ls(pattern = "dataset_spec")
                print(dat_list)
                for(d in dat_list){
                    df <- get(d)
                    if(dim(df)[1] > 0){
                        # remove rows with no data
                        completeFun <- function(data, desiredCols) {completeVec <- complete.cases(data[,
                        desiredCols]); return(data[completeVec, ])}
                        # remove empty rows across df
                        df <- df[rowSums(is.na(df)) != ncol(df),]
                        # remove rows with no DateTime
                        df <- df[!is.na(df$DateTime),]
                        # remove rows with multiple version of missing data
                        df <- completeFun(df,1:3) # no DateTime,lat,lon
                        for(desiredCount in 3:5){df <- df[rowSums(is.na(df)) != ncol(df)-desiredCount,]} # no data
                        in data columns
                        # ordering the dataset by date and selecting unique values only
                        if(dim(df)[1]>0){
                            df <- df[order(df$DateTime),]
                            df <- unique(df)
                            # rename the rows to reflect unique data
                            row.names(df) <- 1:nrow(df)
                        }
                    }
                    # export from function
                    if(dim(df)[1] !=0){assign(d, df, envir=parent.frame())}
                }
                rm(d)
            }
        }
    }
}
```

US Army Corps
of Engineers®

verify_netcdf_3.R

```r
verify_netcdf_3 <- function(buoys = "list of buoys", data_dir = "data_dir"){

        ##-------------------------------------------------------------------------------------------
        ## validates netCDF metadata with NDBC google spreadsheet metadata as captured by NDBC, DiNapoli,2020
        ## Hall, Candice
        ##-------------------------------------------------------------------------------------------

        ## Actions:
        ## 1.  Sets data locations
        ## 2.  Read in NDBC and NCEI data if not already loaded in global environ
        ## 3.  Loads and formats the station specific metadata spreadsheet that was concatenated in step 1a above.
        ## 4.  Selects the station specific NCEI stdmet data and removes NA and duplicate data
        ## 5.  Concatenates and verifies wave metadata with NDBC metadata spreadsheets
        ## 6.  Concatenates and verifies other dual meteorological sensor metadata with the NDBC metadata spreadsheets
        ## 7.  Performs housekeeping tasks to remove excess sensor information labels that were included within the
        metadata extraction process.
        ## 8.  Removes excess hull info and verify sensor height information
        ## 9.  Double-checks that NCEI netCDF extracted metadata fields no longer contain 'no available information'.
        ## 10. Filters for unique date/time data and reorders the datasets by date/time, before reordering the
        datasets to columns structures that match NDBC stdmet datasets.
        ## 11. The verified stdmet with newly verified metadata, as well as all of the individual spectral wave
        variable datasets, are saved in an 's_buoy#_ncei_ALL_verified.Rdata' container file within buoy station
        specific folders

        #-------------------------------------------------------------------------------------------
        #-------------------------------------------------------------------------------------------
        # library(NCmisc)
        # list.functions.in.file(rstudioapi::getSourceEditorContext()$path, alphabetic = TRUE)
        ## libraries required
        # install.packages("tidyverse", lib="/p/home/candice/Rlibs/")
        # install.packages("readxl", lib="/p/home/candice/Rlibs/") # dependent on tidyverse
        # install.packages("lubridate", lib="/p/home/candice/Rlibs/")
        # install.packages("stringr", lib="/p/home/candice/Rlibs/")
        # install.packages("data.table", lib="/p/home/candice/Rlibs/")
        # install.packages("dplyr", lib="/p/home/candice/Rlibs/")
        # install.packages("tidyr", lib="/p/home/candice/Rlibs/")

        # load libraries (HPC run)
        library(readxl, lib="/p/home/candice/Rlibs/") # dependent of tidyverse
        library(backports, lib="/p/home/candice/Rlibs/") # tidyverse
        library(withr, lib="/p/home/candice/Rlibs") # tidyverse
        library(cli, lib="/p/home/candice/Rlibs") # tidyverse
        library(tzdb, lib="/p/home/candice/Rlibs") # tidyverse
        library(readr, lib="/p/home/candice/Rlibs") # tidyverse
        library(rstudioapi, lib="/p/home/candice/Rlibs") # tidyverse
        library(tidyverse, lib="/p/home/candice/Rlibs/")
        library(readxl, lib="/p/home/candice/Rlibs/") # dependent on tidyverse
        library(lubridate, lib="/p/home/candice/Rlibs/")
        library(stringr, lib="/p/home/candice/Rlibs/")
        library(data.table, lib="/p/home/candice/Rlibs/")
```

```
library(crayon, lib="/p/home/candice/Rlibs/") # dplyr
library(pillar, lib="/p/home/candice/Rlibs/") # dplyr
library(dplyr, lib="/p/home/candice/Rlibs/")
library(tidyr, lib="/p/home/candice/Rlibs/")
# library(stats, lib="/p/home/candice/Rlibs/")

# # load libraries (local run)
# library(tidyverse)
# library(readxl) # dependent on tidyverse
# library(lubridate)
# library(stringr)
# library(data.table)
# library(dplyr)
# library(tidyr)
# library(stats)


##------------------------------------------------------------------------------------------
## set paths
##------------------------------------------------------------------------------------------
# drive <- "E:/Candice/"
# drive <- "/p/work/candice/"

# data_dir <- paste0(drive, "projects/WaveTrends/annual_runs/data/")
setwd(data_dir)

# set input directories
input_dir <- paste0(data_dir,"concat_data/ncei/")

# metadata sheets
metadata_dir <- paste0(data_dir,"NDBC_metadata_sheets/")
# if (!file.exists(metadata_dir)) {dir.create(metadata_dir)}

##------------------------------------------------------------------------------------------
## set buoy stations for downloading (for stand-alone use)
##------------------------------------------------------------------------------------------
# list_ndbc <- read.csv("NDBC_buoys.csv",header = TRUE)
# list_ndbc <- dplyr::filter(list_ndbc, list_ndbc$owner == "NDBC")
# list_ndbc_buoy <- as.character(list_ndbc$station)
# buoys <- list_ndbc_buoy
# rm(list_ndbc, list_ndbc_buoy)
# print(buoys)

##------------------------------------------------------------------------------------------
## set sensor heights
##------------------------------------------------------------------------------------------
## NDBC published sensor heights (August 3, 2016): https://www.ndbc.noaa.gov/bht.shtml
## At what heights are the sensors located on moored buoys?
## Meteorological sensors are normally located at the ten meter level for the 10-meter
## and the 12-meter discus buoys and are at a nominal height of five meters for the
```

```r
## 3-meter discus buoys and the 6-meter NOMAD buoys. However, barometers are located
## inside the hull at the water level.
## Sea surface temperature sensors are located at a depth of 1.5 meters for 10-m and 12-m
## buoys and at 1 meter for all 3-m and 6-m moored buoys. Sea surface temperature sensors
## on NDBC buoys are located near one meter below the water line but they vary by hull type.
## Current hull configurations for water temperature sensors are: for 2.4- and 3-meter hulls
## at 0.7 meters; for 6-meter hulls at 0.8 meters; for 10-meter hulls at 1.1 meters; and
## for 12-meter hulls at 0.9 meters. Historically, water temperature sensors for 3-meter
## "foam" hulls were at 0.75 meters and 1.8-meter hulls were at 0.35 meters.

# wind                  # sst           # ref
wind_12m <- 10;         sst_12m <- 0.9    # NDBC, 2016, https://www.ndbc.noaa.gov/bht.shtml
wind_10m <- 10;         sst_10m <- 1.1    # NDBC, 2016, https://www.ndbc.noaa.gov/bht.shtml
wind_6m <- 5;           sst_6m <- 0.8     # NDBC, 2016, https://www.ndbc.noaa.gov/bht.shtml
wind_5m <- 5;           sst_5m <- 0.7     # No information, Rodney Riley, NDBC Engineer (pers. comms. 03/02/2021)
wind_3m <- 5;           sst_3m <- 0.7     # NDBC, 2016, https://www.ndbc.noaa.gov/bht.shtml
wind_elb <- 5.2;        sst_elb <- 1.8    # 1985 Smith
wind_lnb <- 10;         sst_lnb <- 0.9    # Large navigational buoy: NDBC, 1992, ftp://ftp.library.noaa.gov/noaa_documents.lib/NWS/National_Data_Buoy_Center/technical_bulletin/June-1992_vol-18.pdf
wind_2_8m <- NA;        sst_2_8m <- NA    # No information, Rodney Riley, NDBC Engineer (pers. comms. 03/02/2021)
wind_2_6m <- NA;        sst_2_6m <- NA    # No information, Rodney Riley, NDBC Engineer (pers. comms. 03/02/2021)
wind_2_4m <- 3.3;       sst_2_4m <- 0.7   # NDBC 'BUOY COMPARISONS Final - with sensor heights.pdf'
wind_2_3m <- 3.2;       sst_2_3m <- 1.3   # 2017 Bouchard et al., NDBC 'BUOY COMPARISONS Final - with sensor heights.pdf'
wind_2_1m <- 3.2;       sst_2_1m <- 1.1   # NDBC, 2021, https://www.ndbc.noaa.gov/station_page.php?station=45001
wind_1_8m <- 2.1;       sst_1_8m <- 0.4   # 2008 Crout et al.

for(buoy in buoys){ #

     # start writing to an output file
     print(paste0("Starting on ",buoy))

     # # start writing to an output file
     sink(paste0(data_dir,"3_ncei_metadata_verification_",buoy,"_",Sys.Date(),".txt"))
     print(paste0("Starting on... ",buoy))
     #
     #-----------------------------------------------------------------------------------------
     ## read in data if not loaded in global environ
     #-----------------------------------------------------------------------------------------

     if(file.exists(paste0(input_dir,buoy,"/s_",buoy,"_ncei_ALL.RData"))){

             print("Loading datasets")
             load(paste0(input_dir,buoy,"/s_",buoy,"_ncei_ALL.RData"))

```

```r
# load spreadsheet metadata
# /p/work/candice/projects/WaveTrends/data/NDBC_metadata_sheets
met_dat <- list.files(metadata_dir,full.names = TRUE); met_dat <-
met_dat[grepl("-Meta-Data-",met_dat)]
met_dat <- met_dat[grepl(buoy,met_dat)]
metadata <- read_excel(met_dat, col_names = TRUE, skip = 2, na = c("?","N/A"))
if("AIO Met Sensor" %in% names(metadata)){
        metadata <- dplyr::select(metadata, `Date Start`,Dep:`AIO Met Sensor`)
}else{metadata <- dplyr::select(metadata, `Date Start`,Dep:Sensor); metadata$`AIO Met Sensor` <-
NA}
# remove column name spaces and caps, replace wave NA, merge wave metadata columns and remove
redundant columns
metadata <- dplyr::rename(metadata, Date_Start = `Date Start`,depth = Dep,mooring = Mooring,hull
= Hull,payload = Payload, waveSys = WaveSys,waveSensor = Sensor, windSensor = `AIO Met Sensor`)
for(rn in 1:nrow(metadata)){if(is.na(metadata$waveSys[rn])){metadata$waveSys[rn] <-
"unknown"};if(is.na(metadata$waveSensor[rn])){metadata$waveSensor[rn] <- "unknown"}}
metadata$waveSys_Sensor <- paste0(metadata$waveSys,"; ",metadata$waveSensor)
# reformat date
metadata$DateTime <- ymd_hms(paste0(metadata$Date_Start," 00:00:00"))
metadata$waveSys <- NULL; metadata$sensor <- NULL; metadata$Date_Start <- NULL
# fill in missing data
metadata <- fill(metadata, depth,mooring,hull,payload, .direction = "up")
# reorder - spreadsheets don't have the same column order (?)
metadata <- dplyr::select(metadata, DateTime,depth,hull,mooring,payload,waveSys_Sensor)
# handle unknowns
index_hull <- which(metadata$hull=="unknown");
if(length(index_hull)>0){metadata$hull[index_hull] <- metadata$hull[index_hull+1]}

# set as data.table for date merge
metadata <- data.table(metadata)
setkey(metadata, DateTime)

# find df's to update
stdmet_ls <- ls(pattern = "ncei_stdmet")
stdmet_ls

for(df in stdmet_ls){
     dat <- get(df)

     print(paste0("working on ",df))
     print(Sys.time())

     # removing any character NA's
     dat[, 2:ncol(dat)][dat[, 2:ncol(dat)]=="NA"] <- NA

     # remove duplicate wave data
     if("significant_wave_height_1" %in% names(dat) & "significant_wave_height_2" %in%
     names(dat)){
          print("working on redundant wave data")
          wave <- dplyr::select(dat, DateTime, contains("wave"))
```

```
183
184                         for(wn in 1:nrow(dat)){
185                                 # removing redundant values
186                                 if(round(!is.na(dat$significant_wave_height_1[wn]),4) ==
                                    round(!is.na(dat$significant_wave_height_2[wn]),4)){dat$significant_wave_height_
                                    2[wn] <- NA; dat$significant_wave_height_metadata_2[wn] <- NA}
187                                 if(round(!is.na(dat$dominant_wave_period_1[wn]),4) ==
                                    round(!is.na(dat$dominant_wave_period_2[wn]),4)){dat$dominant_wave_period_2[wn]
                                    <- NA}
188                                 if(round(!is.na(dat$average_wave_period_1[wn]),4) ==
                                    round(!is.na(dat$average_wave_period_2[wn]),4)){dat$average_wave_period_2[wn]
                                    <- NA}
189                                 if("mean_wave_direction_1" %in%
                                    names(wave)){if(!is.na(dat$mean_wave_direction_1[wn]) ==
                                    !is.na(dat$mean_wave_direction_2[wn])){dat$mean_wave_direction_2[wn] <- NA}}
190                                 # replacing na primary data with secondary data
191                                 if(is.na(dat$significant_wave_height_1[wn]) &
                                    !is.na(dat$significant_wave_height_2[wn])){
192                                         dat$significant_wave_height_1[wn] <-
                                            dat$significant_wave_height_2[wn];
                                            dat$significant_wave_height_metadata_1[wn] <-
                                            dat$significant_wave_height_metadata_2[wn]
193                                         dat$significant_wave_height_2[wn] <- NA;
                                            dat$significant_wave_height_metadata_2[wn] <- NA}
194                                 if(is.na(dat$dominant_wave_period_1[wn]) &
                                    !is.na(dat$dominant_wave_period_2[wn])){
195                                         dat$dominant_wave_period_1[wn] <- dat$dominant_wave_period_2[wn];
                                            dat$dominant_wave_period_metadata_1[wn] <-
                                            dat$dominant_wave_period_metadata_2[wn]
196                                         dat$dominant_wave_period_2[wn] <- NA;
                                            dat$dominant_wave_period_metadata_2[wn] <- NA}
197                                 if(is.na(dat$average_wave_period_1[wn]) &
                                    !is.na(dat$average_wave_period_2[wn])){
198                                         dat$average_wave_period_1[wn] <- dat$average_wave_period_2[wn];
                                            dat$average_wave_period_metadata_1[wn] <-
                                            dat$average_wave_period_metadata_2[wn]
199                                         dat$average_wave_period_2[wn] <- NA;
                                            dat$average_wave_period_metadata_2[wn] <- NA}
200                                 if("mean_wave_direction_1" %in% names(wave)){
201                                         if(is.na(dat$mean_wave_direction_1[wn]) &
                                            !is.na(dat$mean_wave_direction_2[wn])){
202                                                 dat$mean_wave_direction_1[wn] <-
                                                    dat$mean_wave_direction_2[wn];
                                                    dat$mean_wave_direction_metadata_1[wn] <-
                                                    dat$mean_wave_direction_metadata_2[wn]
203                                                 dat$mean_wave_direction_2[wn] <- NA;
                                                    dat$mean_wave_direction_metadata_2[wn] <- NA}
204                                 }
205                         }
206                         if(sum(as.numeric(dat$significant_wave_height_2), na.rm =
```

```
                TRUE)==0){dat$significant_wave_height_2 <- NULL;
                dat$significant_wave_height_metadata_2 <- NULL;
207             dat <- dplyr::rename(dat,significant_wave_height = significant_wave_height_1,
                significant_wave_height_metadata = significant_wave_height_metadata_1)
                ;print("removing redundant wave height data")}
208             if(sum(as.numeric(dat$dominant_wave_period_2), na.rm =
                TRUE)==0){dat$dominant_wave_period_2 <- NULL; dat$dominant_wave_period_metadata_2 <-
                NULL;
209             dat <- dplyr::rename(dat,dominant_wave_period = dominant_wave_period_1,
                dominant_wave_period_metadata = dominant_wave_period_metadata_1);print("removing
                redundant wave dom period data")}
210             if(sum(as.numeric(dat$average_wave_period_2), na.rm =
                TRUE)==0){dat$average_wave_period_2 <- NULL; dat$average_wave_period_metadata_2 <-
                NULL;
211             dat <- dplyr::rename(dat,average_wave_period = average_wave_period_1,
                average_wave_period_metadata = average_wave_period_metadata_1);print("removing
                redundant wave avg period data")}
212             if("mean_wave_direction_1" %in% names(wave)){
213                   if(sum(as.numeric(dat$mean_wave_direction_2), na.rm =
                      TRUE)==0){dat$mean_wave_direction_2 <- NULL; dat$mean_wave_direction_metadata_2
                      <- NULL;
214                   dat <- dplyr::rename(dat,mean_wave_direction = mean_wave_direction_1,
                      mean_wave_direction_metadata = mean_wave_direction_metadata_1);print("removing
                      redundant wave dir data")}
215             }
216             rm(wn)
217       }
218
219       print(Sys.time())
220
221       # verify wave metadata with NDBC spreadsheets
222       print("verify wave metadata with NDBC spreadsheets")
223       dat_sub <- dplyr::select(dat, DateTime, significant_wave_height_metadata)
224       dat_sub <- dat_sub[complete.cases(dat_sub),]
225       # sep column
226       setDT(dat_sub)[, paste0("wave_payload_info", 1:9) :=
          tstrsplit(significant_wave_height_metadata, "__")]
227       # remove original payload
228       dat_sub <- data.frame(dat_sub, stringsAsFactors = FALSE)
229       dat_sub <- dat_sub[, -c(2,4:ncol(dat_sub))]
230       dat_sub$wave_payload_info1 <- gsub("_significant_wave_height","",dat_sub$wave_payload_info1)
231       # set as d.table and merge date/time
232       dat_sub <- data.table(dat_sub)
233       setkey(dat_sub, DateTime)
234       dat_merge <- metadata[dat_sub, roll = "nearest"]
235       dat_merge_wave <-
          setkey(data.table(dplyr::rename(dplyr::select(dat_merge,DateTime,wave_payload_info1,waveSys_S
          ensor),wave_payload_info = wave_payload_info1)), DateTime)
236       dat_merge_wave$wave_payload_info_sensor <-
          paste0(dat_merge_wave$wave_payload_info,"__",dat_merge_wave$waveSys_Sensor)
```

```r
dat_merge_wave$wave_payload_info <- NULL; dat_merge_wave$waveSys_Sensor <- NULL
dat_merge_hull <-
setkey(data.table(dplyr::select(dat_merge,DateTime,depth,mooring,hull,payload)), DateTime)
# merge with original data
dat <- data.table(dat)
setkey(dat, DateTime)
dat <- dat_merge_hull[dat, roll = "nearest"]
dat$significant_wave_height_metadata <- NULL; dat$dominant_wave_period_metadata <- NULL;
dat$average_wave_period_metadata <- NULL; dat$mean_wave_direction_metadata <- NULL
dat <- left_join(dat,dat_merge_wave, by = "DateTime")
rm(dat_merge, dat_merge_hull, dat_merge_wave, dat_sub)
# set wave metadata
dat$significant_wave_height_metadata <- dat$wave_payload_info_sensor
dat$dominant_wave_period_metadata <- NA; dat$average_wave_period_metadata <- NA;
dat$mean_wave_direction_metadata <- NA
dat$wave_payload_info_sensor <- NULL

# concat other sensor metadata
print("concatenating other sensors metadata")
# wind data
col_ls <- names(dat)[grep("wind_speed_metadata",names(dat))]
if(length(col_ls) == 2){
     dat$wind_direction_metadata_1 <- NA; dat$wind_gust_metadata_1 <- NA
     dat$wind_direction_metadata_2 <- NA; dat$wind_gust_metadata_2 <- NA
}else{dat$wind_direction_metadata <- NA; dat$wind_gust_metadata <- NA}

# remove excess sensor info
print("removing excess sensor info")
print(Sys.time())
if(grepl("ncei",df,ignore.case = TRUE)){
     if("wind_speed_metadata_1" %in% names(dat)){
          dat$wind_speed_metadata_1 <- gsub("_wind_speed","",dat$wind_speed_metadata_1);
          dat$wind_speed_metadata_2 <- gsub("_wind_speed","",dat$wind_speed_metadata_2)
     }else{dat$wind_speed_metadata <- gsub("_wind_speed","",dat$wind_speed_metadata)}

     if("air_pressure_at_sea_level_metadata_1" %in% names(dat)){
          dat$air_pressure_at_sea_level_metadata_1 <-
          gsub("_air_pressure_at_sea_level","",dat$air_pressure_at_sea_level_metadata_1);dat
          $air_pressure_at_sea_level_metadata_2 <-
          gsub("_air_pressure_at_sea_level","",dat$air_pressure_at_sea_level_metadata_2)
     }else{dat$air_pressure_at_sea_level_metadata <-
     gsub("_air_pressure_at_sea_level","",dat$air_pressure_at_sea_level_metadata)}

     if("air_temperature_metadata_1" %in% names(dat)){
          dat$air_temperature_metadata_1 <-
          gsub("air_temperature__","__",dat$air_temperature_metadata_1);dat$air_temperature_
          metadata_2 <- gsub("air_temperature__","__",dat$air_temperature_metadata_2)
          dat$air_temperature_metadata_1 <-
          gsub("___","__",dat$air_temperature_metadata_1);dat$air_temperature_metadata_2
          <- gsub("___","__",dat$air_temperature_metadata_2)
```

```
275                         dat$air_temperature_metadata_1 <-
                            gsub("___","__",dat$air_temperature_metadata_1);dat$air_temperature_metadata_2
                            <- gsub("___","__",dat$air_temperature_metadata_2)
276                         dat$air_temperature_metadata_1 <-
                            gsub("_air_temperature$","",dat$air_temperature_metadata_1);dat$air_temperature_me
                            tadata_2 <- gsub("_air_temperature$","",dat$air_temperature_metadata_2)
277                   }else{
278                         dat$air_temperature_metadata <-
                            gsub("air_temperature__","__",dat$air_temperature_metadata)
279                         dat$air_temperature_metadata <- gsub("___","__",dat$air_temperature_metadata)
280                         dat$air_temperature_metadata <- gsub("___","__",dat$air_temperature_metadata)
281                         dat$air_temperature_metadata <-
                            gsub("_air_temperature$","",dat$air_temperature_metadata)
282                   }
283                   if("dew_point_temperature_metadata_1" %in% names(dat)){
284                         dat$dew_point_temperature_metadata_1 <-
                            gsub("_dew_point_temperature","",dat$dew_point_temperature_metadata_1);dat$dew_poi
                            nt_temperature_metadata_2 <-
                            gsub("_dew_point_temperature","",dat$dew_point_temperature_metadata_2)
285                   }else{dat$dew_point_temperature_metadata <-
                      gsub("_dew_point_temperature","",dat$dew_point_temperature_metadata)}
286
287                   if("sea_surface_temperature_metadata_1" %in% names(dat)){
288                         dat$sea_surface_temperature_metadata_1 <-
                            gsub("_sea_surface_temperature","",dat$sea_surface_temperature_metadata_1);dat$sea
                            _surface_temperature_metadata_2 <-
                            gsub("_sea_surface_temperature","",dat$sea_surface_temperature_metadata_2)
289                   }else{dat$sea_surface_temperature_metadata <-
                      gsub("_sea_surface_temperature","",dat$sea_surface_temperature_metadata)}
290
291             }
292
293             # remove excess hull info and verify sensor height
294             print("removing excess hull info and verify sensor height")
295             for(rn in 1:nrow(dat)){
296                   print(dat$DateTime[rn])
297
298                   # verify hull
299                   rm(hull_id, hull_type)
300                   hull_id <- dat$hull[rn]
301                   if(str_detect(hull_id, "D")){hull_type <- stringr::str_extract(hull_id, ".{0,3}D");
                      hull_type <- gsub("D","",hull_type)}
302                   if(str_detect(hull_id, "N")){hull_type <- stringr::str_extract(hull_id, ".{0,3}N");
                      hull_type <- gsub("N","",hull_type)}
303                   if(str_detect(hull_id, "B")){hull_type <- stringr::str_extract(hull_id, ".{0,3}B")}# ;
                      hull_type <- gsub("D","",hull_type)}
304                   if(hull_type == 12){wind_height <- wind_12m; sst_height <- sst_12m};    if(hull_type
                      == 10){wind_height <- wind_10m; sst_height <- sst_10m}
305                   if(hull_type == 6){wind_height <- wind_6m; sst_height <- sst_6m};       if(hull_type
                      == 3){wind_height <- wind_3m; sst_height <- sst_3m}
```

```r
if(hull_type == "ELB"){wind_height <- wind_elb; sst_height <- sst_elb}; if(hull_type == 2.8){wind_height <- wind_2_8m; sst_height <- sst_2_8m}
if(hull_type == "LNB"){wind_height <- wind_lnb; sst_height <- sst_lnb}; if(hull_type == 2.8){wind_height <- wind_2_8m; sst_height <- sst_2_8m}
if(hull_type == 2.6){wind_height <- wind_2_6m; sst_height <- sst_2_6m}; if(hull_type == 2.4){wind_height <- wind_2_4m; sst_height <- sst_2_4m}
if(hull_type == 2.3){wind_height <- wind_2_3m; sst_height <- sst_2_3m}; if(hull_type == 1.8){wind_height <- wind_1_8m; sst_height <- sst_1_8m}
if(hull_type == 2.1){wind_height <- wind_2_1m; sst_height <- sst_2_1m}

# air pressure
if("air_pressure_at_sea_level_metadata_1" %in% names(dat)){
     if(!is.na(dat$air_pressure_at_sea_level_metadata_1[rn])){
          rm_pattern <-
          stringr::str_extract(dat$air_pressure_at_sea_level_metadata_1[rn],
          paste0(".{0,2}",hull_id,".{0,7}"))
          if(!is.na(rm_pattern)){dat$air_pressure_at_sea_level_metadata_1[rn] <-
          gsub(rm_pattern,"",dat$air_pressure_at_sea_level_metadata_1[rn])}
          rm(rm_pattern)
     }
}
if("air_pressure_at_sea_level_metadata_2" %in% names(dat)){
     if(!is.na(dat$air_pressure_at_sea_level_metadata_2[rn])){
          rm_pattern <-
          stringr::str_extract(dat$air_pressure_at_sea_level_metadata_2[rn],
          paste0(".{0,2}",hull_id,".{0,7}"))
          if(!is.na(rm_pattern)){dat$air_pressure_at_sea_level_metadata_2[rn] <-
          gsub(rm_pattern,"",dat$air_pressure_at_sea_level_metadata_2[rn])}
          rm(rm_pattern)
     }
}
if("air_pressure_at_sea_level_metadata" %in% names(dat)){
     if(!is.na(dat$air_pressure_at_sea_level_metadata[rn])){
          rm_pattern <-
          stringr::str_extract(dat$air_pressure_at_sea_level_metadata[rn],
          paste0(".{0,2}",hull_id,".{0,7}"))
          if(!is.na(rm_pattern)){dat$air_pressure_at_sea_level_metadata[rn] <-
          gsub(rm_pattern,"",dat$air_pressure_at_sea_level_metadata[rn])}
          rm(rm_pattern)
     }
}
# air temp
if("air_temperature_metadata_1" %in% names(dat)){
     if(!is.na(dat$air_temperature_metadata_1[rn])){
          rm_pattern <- stringr::str_extract(dat$air_temperature_metadata_1[rn],
          paste0(".{0,2}",hull_id,".{0,7}"))
          if(!is.na(rm_pattern)){dat$air_temperature_metadata_1[rn] <-
          gsub(rm_pattern,"",dat$air_temperature_metadata_1[rn])}
          rm(rm_pattern)
     }
```

```
341                         }
342                         if("air_temperature_metadata_2" %in% names(dat)){
343                               if(!is.na(dat$air_temperature_metadata_2[rn])){
344                                     rm_pattern <- stringr::str_extract(dat$air_temperature_metadata_2[rn],
                                      paste0(".{0,2}",hull_id,".{0,7}"))
345                                     if(!is.na(rm_pattern)){dat$air_temperature_metadata_2[rn] <-
                                      gsub(rm_pattern,"",dat$air_temperature_metadata_2[rn])}
346                                     rm(rm_pattern)
347                               }
348                         }
349                         if("air_temperature_metadata" %in% names(dat)){
350                               if(!is.na(dat$air_temperature_metadata[rn])){
351                                     rm_pattern <- stringr::str_extract(dat$air_temperature_metadata[rn],
                                      paste0(".{0,2}",hull_id,".{0,7}"))
352                                     if(!is.na(rm_pattern)){dat$air_temperature_metadata[rn] <-
                                      gsub(rm_pattern,"",dat$air_temperature_metadata[rn])}
353                                     rm(rm_pattern)
354                               }
355                         }
356                         # sea temp
357                         if("sea_surface_temperature_metadata_1" %in% names(dat)){
358                               if(!is.na(dat$sea_surface_temperature_metadata_1[rn])){
359                                     rm_pattern <-
                                      stringr::str_extract(dat$sea_surface_temperature_metadata_1[rn],
                                      paste0(".{0,2}",hull_id,".{0,7}"))
360                                     if(!is.na(rm_pattern)){dat$sea_surface_temperature_metadata_1[rn] <-
                                      gsub(rm_pattern,"",dat$sea_surface_temperature_metadata_1[rn])}
361                                     rm(rm_pattern)
362                                     dat$sea_surface_temperature_metadata_1[rn] <- gsub("no available height
                                      data", paste0(sst_height,"m"), dat$sea_surface_temperature_metadata_1[rn])
363                               }
364                         }
365                         if("sea_surface_temperature_metadata_2" %in% names(dat)){
366                               if(!is.na(dat$sea_surface_temperature_metadata_2[rn])){
367                                     rm_pattern <-
                                      stringr::str_extract(dat$sea_surface_temperature_metadata_2[rn],
                                      paste0(".{0,2}",hull_id,".{0,7}"))
368                                     if(!is.na(rm_pattern)){dat$sea_surface_temperature_metadata_2[rn] <-
                                      gsub(rm_pattern,"",dat$sea_surface_temperature_metadata_2[rn])}
369                                     rm(rm_pattern)
370                                     dat$sea_surface_temperature_metadata_2[rn] <- gsub("no available height
                                      data", paste0(sst_height,"m"), dat$sea_surface_temperature_metadata_2[rn])
371                               }
372                         }
373                         if("sea_surface_temperature_metadata" %in% names(dat)){
374                               if(!is.na(dat$sea_surface_temperature_metadata[rn])){
375                                     rm_pattern <- stringr::str_extract(dat$sea_surface_temperature_metadata[rn],
                                      paste0(".{0,2}",hull_id,".{0,7}"))
376                                     if(!is.na(rm_pattern)){dat$sea_surface_temperature_metadata[rn] <-
                                      gsub(rm_pattern,"",dat$sea_surface_temperature_metadata[rn])}
```

```
                    rm(rm_pattern)
                    dat$sea_surface_temperature_metadata[rn] <- gsub("no available height data",
                    paste0(sst_height,"m"), dat$sea_surface_temperature_metadata[rn])
               }
          }
          # dew pt temp
          if("dew_point_temperature_metadata_1" %in% names(dat)){
               if(!is.na(dat$dew_point_temperature_metadata_1[rn])){
                    rm_pattern <- stringr::str_extract(dat$dew_point_temperature_metadata_1[rn],
                    paste0(".{0,2}",hull_id,".{0,7}"))
                    if(!is.na(rm_pattern)){dat$dew_point_temperature_metadata_1[rn] <-
                    gsub(rm_pattern,"",dat$dew_point_temperature_metadata_1[rn])}
                    rm(rm_pattern)
               }
          }
          if("dew_point_temperature_metadata_2" %in% names(dat)){
               if(!is.na(dat$dew_point_temperature_metadata_2[rn])){
                    rm_pattern <- stringr::str_extract(dat$dew_point_temperature_metadata_2[rn],
                    paste0(".{0,2}",hull_id,".{0,7}"))
                    if(!is.na(rm_pattern)){dat$dew_point_temperature_metadata_2[rn] <-
                    gsub(rm_pattern,"",dat$dew_point_temperature_metadata_2[rn])}
                    rm(rm_pattern)
               }
          }
          if("dew_point_temperature_metadata" %in% names(dat)){
               if(!is.na(dat$dew_point_temperature_metadata[rn])){
                    rm_pattern <- stringr::str_extract(dat$dew_point_temperature_metadata[rn],
                    paste0(".{0,2}",hull_id,".{0,7}"))
                    if(!is.na(rm_pattern)){dat$dew_point_temperature_metadata[rn] <-
                    gsub(rm_pattern,"",dat$dew_point_temperature_metadata[rn])}
                    rm(rm_pattern)
               }
          }
          # wind sensor metadata
          if("wind_speed_metadata_1" %in% names(dat)){
               if(!is.na(dat$wind_speed_metadata_1[rn])){
                    rm_pattern <- stringr::str_extract(dat$wind_speed_metadata_1[rn],
                    paste0(".{0,2}",hull_id,".{0,7}"))
                    if(!is.na(rm_pattern)){dat$wind_speed_metadata_1[rn] <-
                    gsub(rm_pattern,"",dat$wind_speed_metadata_1[rn])}
                    rm(rm_pattern)
               }
          }
          if("wind_speed_metadata_2" %in% names(dat)){
               if(!is.na(dat$wind_speed_metadata_2[rn])){
                    rm_pattern <- stringr::str_extract(dat$wind_speed_metadata_2[rn],
                    paste0(".{0,2}",hull_id,".{0,7}"))
                    if(!is.na(rm_pattern)){dat$wind_speed_metadata_2[rn] <-
                    gsub(rm_pattern,"",dat$wind_speed_metadata_2[rn])}
                    rm(rm_pattern)
```

```
416                         }
417                    }
418                    if("wind_speed_metadata" %in% names(dat)){
419                         if(!is.na(dat$wind_speed_metadata[rn])){
420                              rm_pattern <- stringr::str_extract(dat$wind_speed_metadata[rn],
                                 paste0(".{0,2}",hull_id,".{0,7}"))
421                              if(!is.na(rm_pattern)){dat$wind_speed_metadata[rn] <-
                                 gsub(rm_pattern,"",dat$wind_speed_metadata[rn])}
422                              rm(rm_pattern)
423                         }
424                    }
425
426                    # remove unknown from redundant sensor heights
427                    # find sensor and associated heights
428                    rm(wind_s1, wind_s1_height, wind_s2, wind_s2_height)
429                    if("wind_speed_metadata_1" %in% names(dat)){
430                         if(!is.na(dat$wind_speed_metadata_1[rn])){
431                              wind_sensor <- unlist(strsplit(dat$wind_speed_metadata_1[rn], "__"));
                                 wind_s1 <- wind_sensor[length(wind_sensor)-1];if(length(wind_s1)==0){wind_s1
                                 <- dat$wind_speed_metadata_1[rn]}
432                              wind_sensor_height <- unlist(strsplit(dat$wind_speed_metadata_1[rn], "__"));
                                 #wind_s1_height <- wind_sensor_height[length(wind_sensor_height)]
433                              if(wind_sensor_height != wind_s1){wind_s1_height <-
                                 wind_sensor_height[length(wind_sensor_height)]}else{wind_s1 <- NA;
                                 wind_s1_height <- NA}
434                              if(!is.na(wind_s1_height) & !is.na(wind_s1)){
435                                        if(wind_s1_height == "no available height data" & wind_s1 == "no
                                           available manufacturer data"){wind_s1_height <-
                                           wind_height;dat$wind_speed_metadata_1[rn] <- gsub("no available
                                           height data",paste0(wind_s1_height,"m"),
                                           dat$wind_speed_metadata_1[rn])}
436                              }
437                              rm(wind_sensor, wind_sensor_height)
438                         }else{wind_s1 <- NA; wind_s1_height <- NA}
439                    }
440                    if("wind_speed_metadata_2" %in% names(dat)){
441                         if(!is.na(dat$wind_speed_metadata_2[rn])){
442                              wind_sensor <- unlist(strsplit(dat$wind_speed_metadata_2[rn], "__"));
                                 wind_s2 <- wind_sensor[length(wind_sensor)-1];if(length(wind_s2)==0){wind_s2
                                 <- dat$wind_speed_metadata_2[rn]}
443                              wind_sensor_height <- unlist(strsplit(dat$wind_speed_metadata_2[rn], "__"))
444                              if(wind_sensor_height != wind_s2){wind_s2_height <-
                                 wind_sensor_height[length(wind_sensor_height)]}else{wind_s2 <- NA;
                                 wind_s2_height <- NA}
445                              if(!is.na(wind_s2_height) & !is.na(wind_s2)){
446                                        if(wind_s2_height == "no available height data" & wind_s2 == "no
                                           available manufacturer data"){wind_s2_height <-
                                           wind_height;dat$wind_speed_metadata_2[rn] <- gsub("no available
                                           height data",paste0(wind_s2_height,"m"),
                                           dat$wind_speed_metadata_2[rn])}
```

```
447                                   }
448                                   rm(wind_sensor, wind_sensor_height)
449                              }else{wind_s2 <- NA; wind_s2_height <- NA}
450                         }
451                         if("wind_speed_metadata" %in% names(dat)){
452                              if(!is.na(dat$wind_speed_metadata[rn])){
453                                   wind_sensor <- unlist(strsplit(dat$wind_speed_metadata[rn], "__")); wind_s1
                                     <- wind_sensor[length(wind_sensor)-1];if(length(wind_s1)==0){wind_s1 <-
                                     dat$wind_speed_metadata[rn]}
454                                   wind_sensor_height <- unlist(strsplit(dat$wind_speed_metadata[rn], "__"))
455                                   if(wind_sensor_height != wind_s1){wind_s1_height <-
                                     wind_sensor_height[length(wind_sensor_height)]}else{wind_s1 <- NA;
                                     wind_s1_height <- NA}
456                                   if(!is.na(wind_s1_height) & !is.na(wind_s1)){
457                                          if(wind_s1_height == "no available height data" & wind_s1 == "no
                                           available manufacturer data"){wind_s1_height <-
                                           wind_height;dat$wind_speed_metadata[rn] <- gsub("no available height
                                           data",paste0(wind_s1_height,"m"), dat$wind_speed_metadata[rn])}
458                                   }
459                                   rm(wind_sensor, wind_sensor_height)
460                              }else{wind_s1 <- NA; wind_s1_height <- NA}
461                         }
462
463                         # cycle through wind sensor if present
464                         if(exists("wind_s1")){ # mainly for NCEI data
465                              if(!is.na(wind_s1)){
466
467                                   # air pressure
468                                   if("air_pressure_at_sea_level_metadata_1" %in% names(dat)){
469                                        # sensor 1
470                                        if(!is.na(dat$air_pressure_at_sea_level_metadata_1[rn])){
471                                             AP_1 <-
                                               unlist(strsplit(dat$air_pressure_at_sea_level_metadata_1[rn],
                                               "__"));AP_s1 <- AP_1[length(AP_1)-1]
472                                             if(grepl(wind_s1, AP_s1, ignore.case = TRUE, fixed =
                                               FALSE)){dat$air_pressure_at_sea_level_metadata_1[rn] <- gsub("no
                                               available height
                                               data",wind_s1_height,dat$air_pressure_at_sea_level_metadata_1[rn])}
473                                        }
474                                        # sensor 2
475                                        if(!is.na(dat$air_pressure_at_sea_level_metadata_2[rn])){
476                                             AP_2 <-
                                               unlist(strsplit(dat$air_pressure_at_sea_level_metadata_2[rn],
                                               "__"));AP_s2 <- AP_2[length(AP_2)-1]
477                                             if(grepl(wind_s1, AP_s2, ignore.case = TRUE, fixed =
                                               FALSE)){dat$air_pressure_at_sea_level_metadata_2[rn] <- gsub("no
                                               available height
                                               data",wind_s1_height,dat$air_pressure_at_sea_level_metadata_2[rn])}
478                                        }
479                                   }
```

```
480                     if("air_pressure_at_sea_level_metadata" %in% names(dat)){
481                          # sensor 1
482                          if(!is.na(dat$air_pressure_at_sea_level_metadata[rn])){
483                               AP_1 <-
                               unlist(strsplit(dat$air_pressure_at_sea_level_metadata[rn],
                               "__"));AP_s1 <- AP_1[length(AP_1)-1]
484                               if(exists("wind_s1")){if(grepl(wind_s1, AP_s1, ignore.case = TRUE,
                               fixed = FALSE)){dat$air_pressure_at_sea_level_metadata[rn] <-
                               gsub("no available height
                               data",wind_s1_height,dat$air_pressure_at_sea_level_metadata[rn])}}
485                          }
486                     }
487
488                     # air temperature
489                     if("air_temperature_metadata_1" %in% names(dat)){
490                          # sensor 1
491                          if(!is.na(dat$air_temperature_metadata_1[rn])){
492                               AP_1 <- unlist(strsplit(dat$air_temperature_metadata_1[rn],
                               "__"));AP_s1 <- AP_1[length(AP_1)-1]
493                               if(grepl(wind_s1, AP_s1, ignore.case = TRUE, fixed =
                               FALSE)){dat$air_temperature_metadata_1[rn] <- gsub("no available
                               height data",wind_s1_height,dat$air_temperature_metadata_1[rn])}
494                          }
495                          # sensor 2
496                          if(!is.na(dat$air_temperature_metadata_2[rn])){
497                               AP_2 <- unlist(strsplit(dat$air_temperature_metadata_2[rn],
                               "__"));AP_s2 <- AP_2[length(AP_2)-1]
498                               if(grepl(wind_s1, AP_s2, ignore.case = TRUE, fixed =
                               FALSE)){dat$air_temperature_metadata_2[rn] <- gsub("no available
                               height data",wind_s1_height,dat$air_temperature_metadata_2[rn])}
499                          }
500                          if(buoy == 41001 & year(dat$DateTime[rn])==2019){
501                               if(grepl("no available description data__no available manufacturer
                               data__no available height
                               data",dat$air_temperature_metadata_1[rn])){
502                                    if(is.na(dat$air_temperature_metadata_2[rn]) &
                                    !is.na(dat$air_pressure_at_sea_level_2[rn])){
503                                         dat$air_temperature_metadata_1[rn] <-
                                         dat$air_pressure_at_sea_level_metadata_2[rn]
504                                         dat$air_temperature_metadata_1[rn] <-
                                         gsub("barometer","air_temperature_sensor",
                                         dat$air_temperature_metadata_1[rn])
505                                    }
506                               }
507                          }
508                     }
509                     if("air_temperature_metadata" %in% names(dat)){
510                          # sensor 1
511                          if(!is.na(dat$air_temperature_metadata[rn])){
512                               AP_1 <- unlist(strsplit(dat$air_temperature_metadata[rn],
```

```
                    "__"));AP_s1 <- AP_1[length(AP_1)-1]
513                 if(exists("wind_s1")){if(grepl(wind_s1, AP_s1, ignore.case = TRUE,
                    fixed = FALSE)){dat$air_temperature_metadata[rn] <- gsub("no
                    available height
                    data",wind_s1_height,dat$air_temperature_metadata[rn])}}
514               }
515          }
516
517          # dew point temperature
518          if("dew_point_temperature_metadata_1" %in% names(dat)){
519               # sensor 1
520               if(!is.na(dat$dew_point_temperature_metadata_1[rn])){
521                    AP_1 <- unlist(strsplit(dat$dew_point_temperature_metadata_1[rn],
                    "__"));AP_s1 <- AP_1[length(AP_1)-1]
522                    if(grepl(wind_s1, AP_s1, ignore.case = TRUE, fixed =
                    FALSE)){dat$dew_point_temperature_metadata_1[rn] <- gsub("no
                    available height
                    data",wind_s1_height,dat$dew_point_temperature_metadata_1[rn])}
523               }
524               # sensor 2
525               if(!is.na(dat$dew_point_temperature_metadata_2[rn])){
526                    AP_2 <- unlist(strsplit(dat$dew_point_temperature_metadata_2[rn],
                    "__"));AP_s2 <- AP_2[length(AP_2)-1]
527                    if(grepl(wind_s1, AP_s2, ignore.case = TRUE, fixed =
                    FALSE)){dat$dew_point_temperature_metadata_2[rn] <- gsub("no
                    available height
                    data",wind_s1_height,dat$dew_point_temperature_metadata_2[rn])}
528               }
529               if(buoy == 41001 & year(dat$DateTime[rn])==2019){
530                    if(grepl("no available description data__no available manufacturer
                    data__no available height
                    data",dat$dew_point_temperature_metadata_1[rn])){
531                         if(is.na(dat$dew_point_temperature_metadata_2[rn]) &
                         !is.na(dat$air_pressure_at_sea_level_2[rn])){
532                              dat$dew_point_temperature_metadata_1[rn] <-
                              dat$air_temperature_metadata_1[rn]
533                              dat$dew_point_temperature_metadata_1[rn] <-
                              gsub("barometer","air_temperature_sensor",
                              dat$dew_point_temperature_metadata_1[rn])
534                         }
535                    }
536               }
537          }
538          if("dew_point_temperature_metadata" %in% names(dat)){
539               # sensor 1
540               if(!is.na(dat$dew_point_temperature_metadata[rn])){
541                    AP_1 <- unlist(strsplit(dat$dew_point_temperature_metadata[rn],
                    "__"));AP_s1 <- AP_1[length(AP_1)-1]
542                    if(exists("wind_s1")){if(grepl(wind_s1, AP_s1, ignore.case = TRUE,
                    fixed = FALSE)){dat$dew_point_temperature_metadata[rn] <- gsub("no
```

```
                                        available height
                                        data",wind_s1_height,dat$dew_point_temperature_metadata[rn])}}
543                                 }
544                             }
545                         }
546                         if(exists("wind_s2")){
547                             if(!is.na(wind_s2)){
548                                 # air pressure
549                                 if("air_pressure_at_sea_level_metadata_1" %in% names(dat)){
550                                     # sensor 1
551                                     if(!is.na(dat$air_pressure_at_sea_level_metadata_1[rn])){
552                                         AP_1 <-
                                            unlist(strsplit(dat$air_pressure_at_sea_level_metadata_1[rn],
                                            "__"));AP_s1 <- AP_1[length(AP_1)-1]
553                                         if(grepl(wind_s2, AP_s1, ignore.case = TRUE, fixed =
                                            FALSE)){dat$air_pressure_at_sea_level_metadata_1[rn] <-
                                            gsub("no available height
                                            data",wind_s2_height,dat$air_pressure_at_sea_level_metadata_1[r
                                            n])}
554                                     }
555                                     # sensor 2
556                                     if(!is.na(dat$air_pressure_at_sea_level_metadata_2[rn])){
557                                         AP_2 <-
                                            unlist(strsplit(dat$air_pressure_at_sea_level_metadata_2[rn],
                                            "__"));AP_s2 <- AP_2[length(AP_2)-1]
558                                         if(grepl(wind_s2, AP_s2, ignore.case = TRUE, fixed =
                                            FALSE)){dat$air_pressure_at_sea_level_metadata_2[rn] <-
                                            gsub("no available height
                                            data",wind_s2_height,dat$air_pressure_at_sea_level_metadata_2[r
                                            n])}
559                                     }
560                                 }
561                                 if("air_pressure_at_sea_level_metadata" %in% names(dat)){
562                                     # sensor 1
563                                     if(!is.na(dat$air_pressure_at_sea_level_metadata[rn])){
564                                         AP_1 <-
                                            unlist(strsplit(dat$air_pressure_at_sea_level_metadata[rn],
                                            "__"));AP_s1 <- AP_1[length(AP_1)-1]
565                                         if(exists("wind_s2")){if(grepl(wind_s2, AP_s1, ignore.case =
                                            TRUE, fixed =
                                            FALSE)){dat$air_pressure_at_sea_level_metadata[rn] <-
                                            gsub("no available height
                                            data",wind_s2_height,dat$air_pressure_at_sea_level_metadata[rn]
                                            )}}
566                                     }
567                                 }
568
569                                 # air temperature
570                                 if("air_temperature_metadata_1" %in% names(dat)){
571                                     # sensor 1
```

```
572                         if(!is.na(dat$air_temperature_metadata_1[rn])){
573                             AP_1 <- unlist(strsplit(dat$air_temperature_metadata_1[rn],
                                "__"));AP_s1 <- AP_1[length(AP_1)-1]
574                             if(grepl(wind_s2, AP_s1, ignore.case = TRUE, fixed =
                                FALSE)){dat$air_temperature_metadata_1[rn] <- gsub("no
                                available height
                                data",wind_s2_height,dat$air_temperature_metadata_1[rn])}
575                         }
576                         # sensor 2
577                         if(!is.na(dat$air_temperature_metadata_2[rn])){
578                             AP_2 <- unlist(strsplit(dat$air_temperature_metadata_2[rn],
                                "__"));AP_s2 <- AP_2[length(AP_2)-1]
579                             if(grepl(wind_s2, AP_s2, ignore.case = TRUE, fixed =
                                FALSE)){dat$air_temperature_metadata_2[rn] <- gsub("no
                                available height
                                data",wind_s2_height,dat$air_temperature_metadata_2[rn])}
580                         }
581                     }
582                     if("air_temperature_metadata" %in% names(dat)){
583                         # sensor 1
584                         if(!is.na(dat$air_temperature_metadata[rn])){
585                             AP_1 <- unlist(strsplit(dat$air_temperature_metadata[rn],
                                "__"));AP_s1 <- AP_1[length(AP_1)-1]
586                             if(exists("wind_s2")){if(grepl(wind_s2, AP_s1, ignore.case =
                                TRUE, fixed = FALSE)){dat$air_temperature_metadata[rn] <-
                                gsub("no available height
                                data",wind_s2_height,dat$air_temperature_metadata[rn])}}
587                         }
588                     }
589
590                     # dew point temperature
591                     if("dew_point_temperature_metadata_1" %in% names(dat)){
592                         # sensor 1
593                         if(!is.na(dat$dew_point_temperature_metadata_1[rn])){
594                             AP_1 <-
                                unlist(strsplit(dat$dew_point_temperature_metadata_1[rn],
                                "__"));AP_s1 <- AP_1[length(AP_1)-1]
595                             if(grepl(wind_s2, AP_s1, ignore.case = TRUE, fixed =
                                FALSE)){dat$dew_point_temperature_metadata_1[rn] <- gsub("no
                                available height
                                data",wind_s2_height,dat$dew_point_temperature_metadata_1[rn])}
596                         }
597                         # sensor 2
598                         if(!is.na(dat$dew_point_temperature_metadata_2[rn])){
599                             AP_2 <-
                                unlist(strsplit(dat$dew_point_temperature_metadata_2[rn],
                                "__"));AP_s2 <- AP_2[length(AP_2)-1]
600                             if(grepl(wind_s2, AP_s2, ignore.case = TRUE, fixed =
                                FALSE)){dat$dew_point_temperature_metadata_2[rn] <- gsub("no
                                available height
```

```
                                        data",wind_s2_height,dat$dew_point_temperature_metadata_2[rn])}
                                    }
                                }
                                if("dew_point_temperature_metadata" %in% names(dat)){
                                    # sensor 1
                                    if(!is.na(dat$dew_point_temperature_metadata[rn])){
                                        AP_1 <-
                                        unlist(strsplit(dat$dew_point_temperature_metadata[rn],
                                        "__"));AP_s1 <- AP_1[length(AP_1)-1]
                                        if(exists("wind_s2")){if(grepl(wind_s2, AP_s1, ignore.case =
                                        TRUE, fixed = FALSE)){dat$dew_point_temperature_metadata[rn]
                                        <- gsub("no available height
                                        data",wind_s2_height,dat$dew_point_temperature_metadata[rn])}}
                                    }
                                }
                            }
                        }
                    }

                    # cycle through one more time looking for 'no available height data'
                    # wind
                    if("wind_speed_metadata_1" %in% names(dat)){
                        if(!is.na(dat$wind_speed_metadata_1[rn])){
                            if(grepl("no available height data",
                            dat$wind_speed_metadata_1[rn])==TRUE){
                                dat$wind_speed_metadata_1[rn] <- gsub("no available height
                                data", paste0(wind_height,"m"), dat$wind_speed_metadata_1[rn])
                            }
                        }
                        if(!is.na(dat$wind_speed_metadata_2[rn])){
                            if(grepl("no available height data",
                            dat$wind_speed_metadata_2[rn])==TRUE){
                                dat$wind_speed_metadata_2[rn] <- gsub("no available height
                                data", paste0(wind_height,"m"), dat$wind_speed_metadata_2[rn])
                            }
                        }
                    }
                    if("wind_speed_metadata" %in% names(dat)){
                        if(!is.na(dat$wind_speed_metadata[rn])){
                            if(grepl("no available height data",
                            dat$wind_speed_metadata[rn])==TRUE){
                                dat$wind_speed_metadata[rn] <- gsub("no available height
                                data", paste0(wind_height,"m"), dat$wind_speed_metadata[rn])
                            }
                        }
                    }
                    # air pressure
                    if("air_pressure_at_sea_level_metadata_1" %in% names(dat)){
                        if(!is.na(dat$air_pressure_at_sea_level_metadata_1[rn])){
                            if(grepl("no available height data",
```

```
                                        dat$air_pressure_at_sea_level_metadata_1[rn])==TRUE){
639                                             dat$air_pressure_at_sea_level_metadata_1[rn] <- gsub("no
                                                available height data", paste0(wind_height,"m"),
                                                dat$air_pressure_at_sea_level_metadata_1[rn])
640                                     }
641                             }
642                             if(!is.na(dat$air_pressure_at_sea_level_metadata_2[rn])){
643                                     if(grepl("no available height data",
                                        dat$air_pressure_at_sea_level_metadata_2[rn])==TRUE){
644                                             dat$air_pressure_at_sea_level_metadata_2[rn] <- gsub("no
                                                available height data", paste0(wind_height,"m"),
                                                dat$air_pressure_at_sea_level_metadata_2[rn])
645                                     }
646                             }
647                     }
648                     if("air_pressure_at_sea_level_metadata" %in% names(dat)){
649                             if(!is.na(dat$air_pressure_at_sea_level_metadata[rn])){
650                                     if(grepl("no available height data",
                                        dat$air_pressure_at_sea_level_metadata[rn])==TRUE){
651                                             dat$air_pressure_at_sea_level_metadata[rn] <- gsub("no
                                                available height data", paste0(wind_height,"m"),
                                                dat$air_pressure_at_sea_level_metadata[rn])
652                                     }
653                             }
654                     }
655                     # air temperature
656                     if("air_temperature_metadata_1" %in% names(dat)){
657                             if(!is.na(dat$air_temperature_metadata_1[rn])){
658                                     if(grepl("no available height data",
                                        dat$air_temperature_metadata_1[rn])==TRUE){
659                                             dat$air_temperature_metadata_1[rn] <- gsub("no available
                                                height data", paste0(wind_height,"m"),
                                                dat$air_temperature_metadata_1[rn])
660                                     }
661                             }
662                             if(!is.na(dat$air_temperature_metadata_2[rn])){
663                                     if(grepl("no available height data",
                                        dat$air_temperature_metadata_2[rn])==TRUE){
664                                             dat$air_temperature_metadata_2[rn] <- gsub("no available
                                                height data", paste0(wind_height,"m"),
                                                dat$air_temperature_metadata_2[rn])
665                                     }
666                             }
667                     }
668                     if("air_temperature_metadata" %in% names(dat)){
669                             if(!is.na(dat$air_temperature_metadata[rn])){
670                                     if(grepl("no available height data",
                                        dat$air_temperature_metadata[rn])==TRUE){
671                                             dat$air_temperature_metadata[rn] <- gsub("no available height
                                                data", paste0(wind_height,"m"),
```

```
                                        dat$air_temperature_metadata[rn])
672                                 }
673                         }
674                 }
675                 # dew point temperature
676                 if("dew_point_temperature_metadata_1" %in% names(dat)){
677                         if(!is.na(dat$dew_point_temperature_metadata_1[rn])){
678                                 if(grepl("no available height data",
                                    dat$dew_point_temperature_metadata_1[rn])==TRUE){
679                                         dat$dew_point_temperature_metadata_1[rn] <- gsub("no available
                                        height data", paste0(wind_height,"m"),
                                        dat$dew_point_temperature_metadata_1[rn])
680                                 }
681                         }
682                         if(!is.na(dat$dew_point_temperature_metadata_2[rn])){
683                                 if(grepl("no available height data",
                                    dat$dew_point_temperature_metadata_2[rn])==TRUE){
684                                         dat$dew_point_temperature_metadata_2[rn] <- gsub("no available
                                        height data", paste0(wind_height,"m"),
                                        dat$dew_point_temperature_metadata_2[rn])
685                                 }
686                         }
687                 }
688                 if("dew_point_temperature_metadata" %in% names(dat)){
689                         if(!is.na(dat$dew_point_temperature_metadata[rn])){
690                                 if(grepl("no available height data",
                                    dat$dew_point_temperature_metadata[rn])==TRUE){
691                                         dat$dew_point_temperature_metadata[rn] <- gsub("no available
                                        height data", paste0(wind_height,"m"),
                                        dat$dew_point_temperature_metadata[rn])
692                                 }
693                         }
694                 }
695                 # copy metadata across wave variables
696                 if(!is.na(dat$dominant_wave_period[rn])){dat$dominant_wave_period_metadata[rn] <-
                    dat$significant_wave_height_metadata[rn]}
697                 if(!is.na(dat$average_wave_period[rn])){dat$average_wave_period_metadata[rn] <-
                    dat$significant_wave_height_metadata[rn]}
698                 if(!is.na(dat$mean_wave_direction[rn])){dat$mean_wave_direction_metadata[rn] <-
                    dat$significant_wave_height_metadata[rn]}
699                 if("wind_direction_1" %in% names(dat)){
700                         if(!is.na(dat$wind_speed_metadata_1[rn])){dat$wind_direction_metadata_1[rn] <-
                        dat$wind_speed_metadata_1[rn]; dat$wind_gust_metadata_1[rn] <-
                        dat$wind_speed_metadata_1[rn]}
701                         if(!is.na(dat$wind_speed_metadata_2[rn])){dat$wind_direction_metadata_2[rn] <-
                        dat$wind_speed_metadata_2[rn]; dat$wind_gust_metadata_2[rn] <-
                        dat$wind_speed_metadata_2[rn]}
702                 }
703         } # end of row checks
704
```

```r
# ordering the df by date and selecting unique values only
dat <- dat[order(dat$DateTime),]
# rename the rows to reflect unique data
row.names(dat) <- 1:nrow(dat)

if(ncol(dat)== 45){
        print("ncei data format - dual sst")
        dat <- dplyr::select(dat, DateTime, depth, mooring, hull, payload, lat,
        lat_metadata, lon, lon_metadata,
                        wind_direction_1, wind_direction_metadata_1, wind_direction_2,
                        wind_direction_metadata_2,
                        wind_speed_1, wind_speed_metadata_1, wind_speed_2,
                        wind_speed_metadata_2,
                        wind_gust_1, wind_gust_metadata_1, wind_gust_2, wind_gust_metadata_2,
                        significant_wave_height, significant_wave_height_metadata,
                        dominant_wave_period, dominant_wave_period_metadata,
                        average_wave_period, average_wave_period_metadata,
                        mean_wave_direction, mean_wave_direction_metadata,
                        air_pressure_at_sea_level_1, air_pressure_at_sea_level_metadata_1,
                        air_pressure_at_sea_level_2, air_pressure_at_sea_level_metadata_2,
                        air_temperature_1, air_temperature_metadata_1, air_temperature_2,
                        air_temperature_metadata_2,
                        sea_surface_temperature_1,
                        sea_surface_temperature_metadata_1,sea_surface_temperature_2,
                        sea_surface_temperature_metadata_2,
                        dew_point_temperature_1, dew_point_temperature_metadata_1,
                        dew_point_temperature_2, dew_point_temperature_metadata_2)
}else if(ncol(dat)== 43){
        print("ncei data format - single sst or single dew point temperature")
        if("sea_surface_temperature" %in% names(dat)){
                dat <- dplyr::select(dat, DateTime, depth, mooring, hull, payload, lat,
                lat_metadata, lon, lon_metadata,
                                        wind_direction_1, wind_direction_metadata_1,
                                        wind_direction_2, wind_direction_metadata_2,
                                        wind_speed_1, wind_speed_metadata_1, wind_speed_2,
                                        wind_speed_metadata_2,
                                        wind_gust_1, wind_gust_metadata_1, wind_gust_2,
                                        wind_gust_metadata_2,
                                        significant_wave_height,
                                        significant_wave_height_metadata,
                                        dominant_wave_period, dominant_wave_period_metadata,
                                        average_wave_period, average_wave_period_metadata,
                                        mean_wave_direction, mean_wave_direction_metadata,
                                        air_pressure_at_sea_level_1,
                                        air_pressure_at_sea_level_metadata_1,
                                        air_pressure_at_sea_level_2,
                                        air_pressure_at_sea_level_metadata_2,
                                        air_temperature_1, air_temperature_metadata_1,
                                        air_temperature_2, air_temperature_metadata_2,
                                        sea_surface_temperature,
```

```
                                                sea_surface_temperature_metadata,
                                                dew_point_temperature_1,
                                                dew_point_temperature_metadata_1,
                                                dew_point_temperature_2,
                                                dew_point_temperature_metadata_2)
                        }else if("dew_point_temperature" %in% names(dat)){
                                dat <- dplyr::select(dat, DateTime, depth, mooring, hull, payload, lat,
                                lat_metadata, lon, lon_metadata,
                                                wind_direction_1, wind_direction_metadata_1,
                                                wind_direction_2, wind_direction_metadata_2,
                                                wind_speed_1, wind_speed_metadata_1, wind_speed_2,
                                                wind_speed_metadata_2,
                                                wind_gust_1, wind_gust_metadata_1, wind_gust_2,
                                                wind_gust_metadata_2,
                                                significant_wave_height,
                                                significant_wave_height_metadata,
                                                dominant_wave_period, dominant_wave_period_metadata,
                                                average_wave_period, average_wave_period_metadata,
                                                mean_wave_direction, mean_wave_direction_metadata,
                                                air_pressure_at_sea_level_1,
                                                air_pressure_at_sea_level_metadata_1,
                                                air_pressure_at_sea_level_2,
                                                air_pressure_at_sea_level_metadata_2,
                                                air_temperature_1, air_temperature_metadata_1,
                                                air_temperature_2, air_temperature_metadata_2,
                                                sea_surface_temperature_1,
                                                sea_surface_temperature_metadata_1,sea_surface_temperatu
                                                re_2, sea_surface_temperature_metadata_2,
                                                dew_point_temperature, dew_point_temperature_metadata)
                                }else{print("error: dim = 43 but dat names don't match")}
                        }else if(ncol(dat)== 41){
                                print("ncei data format - single dew point")
                                dat <- dplyr::select(dat, DateTime, depth, mooring, hull, payload, lat,
                                lat_metadata, lon, lon_metadata,
                                                wind_direction_1, wind_direction_metadata_1, wind_direction_2,
                                                wind_direction_metadata_2,
                                                wind_speed_1, wind_speed_metadata_1, wind_speed_2,
                                                wind_speed_metadata_2,
                                                wind_gust_1, wind_gust_metadata_1, wind_gust_2, wind_gust_metadata_2,
                                                significant_wave_height, significant_wave_height_metadata,
                                                dominant_wave_period, dominant_wave_period_metadata,
                                                average_wave_period, average_wave_period_metadata,
                                                mean_wave_direction, mean_wave_direction_metadata,
                                                air_pressure_at_sea_level_1, air_pressure_at_sea_level_metadata_1,
                                                air_pressure_at_sea_level_2, air_pressure_at_sea_level_metadata_2,
                                                air_temperature_1, air_temperature_metadata_1, air_temperature_2,
                                                air_temperature_metadata_2,
                                                sea_surface_temperature, sea_surface_temperature_metadata,
                                                dew_point_temperature, dew_point_temperature_metadata)
                        }else if(ncol(dat)== 31){
```

```
                                print("ncei data format - single sensors for all")
                                dat <- dplyr::select(dat, DateTime, depth, mooring, hull, payload, lat,
                                lat_metadata, lon, lon_metadata,
                                                wind_direction, wind_direction_metadata, wind_speed,
                                                wind_speed_metadata, wind_gust, wind_gust_metadata,
                                                significant_wave_height, significant_wave_height_metadata,
                                                dominant_wave_period, dominant_wave_period_metadata,
                                                average_wave_period, average_wave_period_metadata,
                                                mean_wave_direction, mean_wave_direction_metadata,
                                                air_pressure_at_sea_level, air_pressure_at_sea_level_metadata,
                                                air_temperature, air_temperature_metadata,
                                                sea_surface_temperature, sea_surface_temperature_metadata,
                                                dew_point_temperature, dew_point_temperature_metadata)
                           }

                           # save data
                           dfv <- paste0(df,"_verified")
                           assign(dfv,dat)
                           rm(dat,AP_1, AP_2, AP_s1, AP_s2, hull_type, index_hull,wind_s1, wind_s1_height, wind_s2,
                           wind_s2_height, metadata)
                      }

                      #-------------------------------------------------------------------------------------
                      # exporting GeoCleaned datasets
                      #-------------------------------------------------------------------------------------

                      print("Exporting new verified data")
                      ncei_list <- ls(pattern = "verified")
                      print(paste0("NCEI datasets :", ncei_list))

                      # export and save verified dataset
                      for(g in ncei_list){
                           if(grepl("_verified",g)==TRUE){
                                print(g)
                                # write.table(get(g), paste0(input_dir,buoy,"/",g,".csv"), row.names=FALSE, col.names
                                = TRUE, sep = ",")
                                # saveRDS(get(g), paste0(input_dir,buoy,"/",g,".rds"))
                           }
                      }
                      # export to RData
                      ncei_list <- ls(pattern = buoy)
                      save(list = ncei_list, file = paste0(input_dir,buoy,"/s_",buoy,"_ncei_ALL_verified.RData"))

                      # # Stop writing to the file
                      sink()
                      print("test complete")

                      print(paste0("finishing metadata verification for buoy ",buoy))
                 }else{print("no new data for this buoy")}
            }
```

US Army Corps
of Engineers®

geoClean_data_4.R

```r
geoClean_data_4 <- function(buoys = "list of buoys", data_dir = "data_dir"){

      ##-------------------------------------------------------------------------------------------
      ## Compares and geographically cleans data before verifying metadata by combining NDBC web files and NCEI
      netcdf files
      ## Hall, Candice
      ##-------------------------------------------------------------------------------------------

      ## Actions:
      ## 1.  Sets data locations
      ## 2.  Read in NDBC and NCEI data if not already loaded in global environ
      ## 3.  Compares the NDBC and NCEI sourced data by matching the datasets on 'nearest' date and time.
      ## 4.  This step geographically quality controls each dataset by removing GPS positions and
      ##     associated data that are not within a pre-selected radius of the NDBC station watch circles.
      ## 5.  This step assigns verified metadata to the NDBC stdmet datasets.
      ## 6.  If desired, the final section of code within the 'geoClean_data_4.R' script produces statistical
      ##     comparisons and plots of the NDBC and NCEI datasets. A 'switch_set' turns this functionality
      ##     on and off during stand-alone use of this script. To activate this functionality in HPC runs, the
      ##     'geoClean_data_4.R' will require and additional 'switch-set' input value to function.

      #-------------------------------------------------------------------------------------------
      #-------------------------------------------------------------------------------------------
      ## load libraries (local run)
      library(lubridate)
      library(plyr)
      library(dplyr)
      library(gridExtra)
      library(data.table)
      # library(oce)
      library(naniar)
      library(tidyverse)
      library(broom)
      library(openair) # polarplots
      library(plotly)
      library(magrittr)
      library(tidyr)
      # library(grid)
      # library(devtools)
      library(lsr)
      library(stringr)
      library(RColorBrewer)
      library(viridis)
      library(colorRamps)
      library(ggplot2)
      library(ggmap)
      library(maps)
      library(mapdata)
      library(modeest)
      library(tibble)

```

```
## load libraries (HPC run)
# library(lubridate, lib="/p/home/candice/Rlibs/")
# library(plyr, lib="/p/home/candice/Rlibs/")
# library(dplyr, lib="/p/home/candice/Rlibs/")
# library(gridExtra, lib="/p/home/candice/Rlibs/")
# library(data.table, lib="/p/home/candice/Rlibs/")
# # library(oce, lib="/p/home/candice/Rlibs/")
# library(naniar, lib="/p/home/candice/Rlibs/")
# library(tidyverse, lib="/p/home/candice/Rlibs/")
# library(broom, lib="/p/home/candice/Rlibs/")
# library(openair, lib="/p/home/candice/Rlibs/") # polarplots
# library(plotly, lib="/p/home/candice/Rlibs/")
# library(magrittr, lib="/p/home/candice/Rlibs/")
# library(tidyr, lib="/p/home/candice/Rlibs/")
# # library(grid, lib="/p/home/candice/Rlibs/")
# # library(devtools, lib="/p/home/candice/Rlibs/")
# library(lsr, lib="/p/home/candice/Rlibs/")
# library(stringr, lib="/p/home/candice/Rlibs/")
# library(RColorBrewer, lib="/p/home/candice/Rlibs/")
# library(viridis, lib="/p/home/candice/Rlibs/")
# library(colorRamps, lib="/p/home/candice/Rlibs/")
# library(ggplot2, lib="/p/home/candice/Rlibs/")
# library(ggmap, lib="/p/home/candice/Rlibs/")
# library(maps, lib="/p/home/candice/Rlibs/")
# library(mapdata, lib="/p/home/candice/Rlibs/")
# library(modeest, lib="/p/home/candice/Rlibs/")
# library(tibble, lib="/p/home/candice/Rlibs/")

#-------------------------------------------------------------------------------------------
#-------------------------------------------------------------------------------------------

# set switch for script actions
# 1 == plot/stats for geoClean dataset creation, 2 == don't plot/stats
switch_set <- 2
# GPS buffer
GPS_buffer <- 1  # allows for a 1 degree radius

##-----------------------------------------------------------------------------------------
## set paths
##-----------------------------------------------------------------------------------------
# drive <- "E:/Candice/"
# drive <- "/p/work/candice/"
# data_dir <- paste0(drive, "projects/WaveTrends/annual_runs/data/")
setwd(data_dir)

# set input directories
input_dir <- paste0(data_dir,"concat_data/")
# ndbc
ndbc_dir <- paste0(input_dir,"ndbc/")
# ncei
```

```r
ncei_dir <- paste0(input_dir,"ncei/")

# set new output directories for datasets, stats and figures
if (!file.exists(paste0(data_dir,"geoClean_data/"))) {dir.create((paste0(data_dir,"geoClean_data/")))}
clean_data_dir <- paste0(data_dir,"geoClean_data/")

# set new output directories for datasets, stats and figures
if (!file.exists(paste0(clean_data_dir,"results/"))) {dir.create((paste0(clean_data_dir,"results/")))}
out_dir <- paste0(clean_data_dir,"results/")
if (!file.exists(paste0(clean_data_dir,"data/"))) {dir.create((paste0(clean_data_dir,"data/")))}
clean_dir <- paste0(clean_data_dir,"data/")
if (!file.exists(paste0(out_dir,"stats/"))) {dir.create((paste0(out_dir,"stats/")))}
stats_dir <- paste0(out_dir,"stats/")
if (!file.exists(paste0(out_dir,"figures/"))) {dir.create((paste0(out_dir,"figures/")))}
fig_dir <- paste0(out_dir,"figures/")
if (!file.exists(paste0(out_dir,"GPS_plots/"))) {dir.create((paste0(out_dir,"GPS_plots/")))}
gps_dir <- paste0(out_dir,"GPS_plots/")

##------------------------------------------------------------------------------------------
## set set parameters common to all plots
##------------------------------------------------------------------------------------------
xlab = "Date"   # label for x axis
width = 2000+2000    # width of exported plot
height = 1500+1500   # height of exported plot
res = 300
# res2 = 500
# set plot parameters
width1 = 1000
height1 = 700
par1 = c(5,5,4,4)

# # colors for each buoy
plot_colors <- viridis(n=6)
color_ndbc_raw <- plot_colors[1]
color_ndbc_orig <- plot_colors[5]    #"#440154FF" # purple
color_ndbc_recalc <- plot_colors[3] # "#3B528BFF"  #"5 = #FDE725FF" # yellow
color_chl_calc <- plot_colors[4]  # "#21908CFF"   #"#21908CFF" # green
color_WIS <- "red"     # "#5DC863FF"

# wind/wave polar plots
colour1 <- viridis(n=4)
cols1 <- c(colour1[4], colour1[3], colour1[2], colour1[1])
colour2 <- viridis(n=5)
cols2 <- c(colour2[5], colour2[4], colour2[3], colour2[2], colour2[1])
type = "l"
pch = "."
lwd = 0.5
cex = 0.5
# set parameters
Delta <- '\U0394'
```

```r
degree <- '\U00B0'
# my.grid function formats
my.format <- "%m-%d-%Y" # "%m-%Y" (long datasets) or "%m-%d-%Y" (short datasets)
my.period <- "weeks" # "months" (long datasets) or "weeks (short datasets)
# my grid function for plots
my.grid <-function(dataset, my.period = "year", my.format = "%Y"){
  grid(nx=NA, ny=NULL)
  abline(v=axis.POSIXct(1, at=seq(min(dataset[1,1]), max(dataset[nrow(dataset),1]),
                                  by= my.period), format=my.format),
         col = "lightgray", lty = "dotted", lwd = par("lwd"))
}
# function to capitalize string
simpleCap <- function(x) {
  s <- strsplit(x, " ")[[1]]
  paste(toupper(substring(s, 1,1)), substring(s, 2),
        sep="", collapse=" ")
}

# The choice of significance level at which you reject H0 is arbitrary.
sig <- 0.01

##-------------------------------------------------------------------------------------------
## set buoy stations for downloading (for stand-alone use)
##-------------------------------------------------------------------------------------------

# list_ndbc <- read.csv(paste0(data_dir,"NDBC_buoys.csv"),header = TRUE)
# list_ndbc <- dplyr::filter(list_ndbc, list_ndbc$owner == "NDBC")
# list_ndbc_buoy <- as.character(list_ndbc$station)
#
# buoys <- list_ndbc_buoy
# rm(list_ndbc, list_ndbc_buoy)
# print(buoys)
##-------------------------------------------------------------------------------------------

for(buoy in buoys){
    # buoy <- buoys[1]

    # start writing to an output file
    print(paste0("Starting on ",buoy))

    #-------------------------------------------------------------------------------------------
    ## read in data if not loaded in global environ
    #-------------------------------------------------------------------------------------------

    if(file.exists(paste0(ndbc_dir,buoy,"/s_",buoy,"_ndbc_ALL.RData"))){

          # # start writing to an output file
          # sink(paste0(data_dir,"3_geoClean_data_",buoy,"_",Sys.Date(),".txt"))
          # print(paste0("Starting on... ",buoy))

```

```r
print("Loading datasets")

load(paste0(ndbc_dir,buoy,"/s_",buoy,"_ndbc_ALL.RData"))
load(paste0(ncei_dir,buoy,"/s_",buoy,"_ncei_ALL_verified.RData"))

#------------------------------------------------
# check to remove empty dataframes, if any
#------------------------------------------------
print("checking for empty dataframes")
dat_list <- ls(pattern = buoy)
print(dat_list)
for(d in dat_list){
    dat <- get(d)
    dat <- data.frame(dat, stringsAsFactors = FALSE)
    if(dim(dat)[1]<= 1){
        rm(list = ls()[grepl(d, ls())])
        print(paste0("removing empty df: ",d))
    }else{print(paste0(d, " is not empty"))}
    rm(dat)
}

#------------------------------------------------
# isolating common metadata from verified ncei df
#------------------------------------------------
print("Handling verified ncei metadata")
# rename non verified data
ncei_list <- ls(pattern = "_ncei_stdmet")
ncei_list <- ncei_list[!ncei_list %in% grep(paste0("_verified", collapse = "|"), ncei_list, value = T)]
print(ncei_list)
dat <- get(ncei_list)
name_net <- paste0("s_",buoy,"_ncei_sdmet_netcdf")
assign(name_net, dat)

print("isolating common metadata from verified ncei df")
metadata <- ls(pattern = "_verified")
print(metadata)
df <- data.frame(get(metadata), stringsAsFactors = FALSE)
drop_cols <- c("DateTime", "lat", "lon","depth", "mooring", "hull", "payload")
meta_df <- dplyr::select(df,all_of(drop_cols))
df <- df[, !names(df) %in% c("depth", "mooring", "hull", "payload")]
# save data
assign(ncei_list, df)
metadata <- gsub("_stdmet_verified", "_station_metadata", metadata)
assign(metadata, meta_df)
# remove original data
rm(list = ls(pattern = "_verified"))
rm(ncei_list,dat,df, meta_df, metadata, drop_cols)

#-------------------------------------------------------------------------------------
## create data count table
```

```r
#-------------------------------------------------------------------------------------------
data_count <- t(data.frame("Original_count", "Original_start", "Original_end",
                           "Comparison_count", "Comparison_start",  "Comparison_end",
                           "GeoCleaned_count", "GeoCleaned_start", "GeoCleaned_end"))

# rm(Original_count, Original_start,
Original_end,Comparison_count,Comparison_start,Comparison_end,GeoCleaned_count,GeoCleaned_start,GeoClea
ned_end)
data_count <- data.frame(data_count, stringsAsFactors = FALSE)

#-------------------------------------------------------------------------------------------
## geoclean matching data
#-------------------------------------------------------------------------------------------
print("GeoCleaning matching datasets")
ncei_list <- ls(pattern = "_ncei_")
ncei_list <- ncei_list[!ncei_list %in% grep(paste0("sensor_output", collapse = "|"), ncei_list, value
= T)]
print(ncei_list)
colNames <- c("DateTime", "lat", "lon")
ncei_positions <- data.frame(matrix(NA, nrow = 0, ncol = length(colNames)))
colnames(ncei_positions) <- colNames
ncei_positions$DateTime <- lubridate::ymd_hms(ncei_positions$DateTime)

# selection matching positions
for(n in ncei_list){
      print(n)
      dat <- get(n)
      column_names <- names(dat)
      if("lat" %in% column_names){
          dat <- dplyr::select(dat,DateTime,lat,lon)
          dat$lat <- as.numeric(dat$lat)
          dat$lon <- as.numeric(dat$lon)
          ncei_positions <- rbind(ncei_positions,dat)
      }else{print(paste0("no GPS data in ", n))}
      rm(dat)
}
# find unique
ncei_positions <- unique(ncei_positions)
# ordering the df by date and selecting unique values only
ncei_positions <- ncei_positions[order(ncei_positions$DateTime),]
ncei_positions <- unique(ncei_positions)
# rename the rows to reflect unique data
row.names(ncei_positions) <- 1:nrow(ncei_positions)
# remove bad data
ncei_positions <- ncei_positions[complete.cases(ncei_positions),]

# remove bad netcdf GPS positions and correct ncei datafiles
library(modeest)
# using a sorted table of value occurrences to find most common lat/lon
lat_tail <- tail(names(sort(table(ncei_positions$lat))),1)
```

```r
lon_tail <- tail(names(sort(table(ncei_positions$lon))),1)
print(paste0("sorted table method - lat: ", lat_tail, "; lon: ", lon_tail))
# check positions in ndbc bulk
if(abs(range(ncei_positions$lat, na.rm = TRUE)[1] - range(ncei_positions$lat, na.rm = TRUE)[2]) > GPS_buffer | abs(range(ncei_positions$lon, na.rm = TRUE)[1] - range(ncei_positions$lon, na.rm = TRUE)[2]) > GPS_buffer){
    print(paste0("range - lat: ", range(ncei_positions$lat, na.rm = TRUE)[1],"; ", range(ncei_positions$lat, na.rm = TRUE)[2]))
    print(paste0("range - lon: ", range(ncei_positions$lon, na.rm = TRUE)[1],"; ", range(ncei_positions$lon, na.rm = TRUE)[2]))
    ncei_positions <- dplyr::filter(ncei_positions, lat >= as.numeric(lat_tail)-as.numeric(GPS_buffer) & lat <= as.numeric(lat_tail)+as.numeric(GPS_buffer))
    ncei_positions <- dplyr::filter(ncei_positions, lon >= as.numeric(lon_tail)-as.numeric(GPS_buffer) & lon <= as.numeric(lon_tail)+as.numeric(GPS_buffer))
    # ordering the df by date and selecting unique values only
    ncei_positions <- ncei_positions[order(ncei_positions$DateTime),]
    ncei_positions <- unique(ncei_positions)
    # rename the rows to reflect unique data
    row.names(ncei_positions) <- 1:nrow(ncei_positions)
    # remove bad data
    ncei_positions <- ncei_positions[complete.cases(ncei_positions),]

    # apply corrected positions for ncei data
    for(n in ncei_list){
        print(n)
        dat <- get(n)
        column_names <- names(dat)
        if("lat" %in% column_names){
            dat$lat <- NULL; dat$lon <- NULL
            dat <- left_join(dat, ncei_positions, by = "DateTime")
            dat <- dplyr::select(dat, all_of(column_names))
            assign(n,dat)
        }else{print(paste0("no GPS data in ", n))}
        rm(dat)
    }
}

# export as csv and rds
print("saving original GPS data")
if (!file.exists(paste0(clean_dir,buoy,"/"))) {dir.create((paste0(clean_dir,buoy,"/")))}
# write.table(ncei_positions, paste0(clean_dir,buoy,"/s_",buoy,"_GPS_ALL.csv"), row.names=FALSE, col.names = TRUE, sep = ",")
# saveRDS(ncei_positions, file = paste0(clean_dir,buoy,"/s_",buoy,"_GPS_ALL.rds"))
# rm(ncei_positions)

# list out the NDBC and NCEI data
ndbc_list <- ls(pattern = paste0("s_",buoy,"_ndbc_"))
print(ndbc_list)
ndbc_ls <- vector()
remainder <- vector()
```

```
# load individual datasets and geoclean pairs
for(n in ndbc_list){
    name_export <- n
    print(name_export)

    # find matching datasets
    ndbc <- get(n)
    ncei_name <- gsub("ndbc_", "ncei_",n)
    tryCatch({dat <- get(ncei_name)},error=function(cond){print(paste0("no matching ncei data for ",
    n))})
    if(exists("dat")==TRUE){
          if(grepl("cols", ncei_name, fixed = TRUE)){
              if("lat" %in% names(ndbc)){
                    ncei <- get(ls(pattern = gsub("ndbc_", "ncei_",n)))
              }else{
                  ncei_count <- as.numeric(gsub("cols","",unlist(strsplit(n,"_freq_"))[2]))+2
                  ncei_name2 <- gsub("ndbc_",
                  "ncei_",paste0(unlist(strsplit(n,"_freq_"))[1],"_freq_",ncei_count,"cols"))
                  # check if ncei data exists
                  if(exists(ncei_name2)){
                      ncei <- get(ls(pattern = ncei_name2))
                  }else{
                      print("no matching ncei data")
                      ndbc_ls <- c(ndbc_ls,n)
                  }
                  rm(ncei_count, ncei_name2)
              }
          }else{
              ncei <- get(ls(pattern = gsub("ndbc_", "ncei_",n)))
          }
          rm(ncei_name)
    }else {remainder <- c(remainder,n)}

    # handling ndbc data that has no matching ncei data
    if(exists("ncei")){
          # remove blank rows
          ndbc <- ndbc[rowSums(is.na(ndbc)) != ncol(ndbc)-1,]
          ncei <- ncei[rowSums(is.na(ncei)) != ncol(ncei)-1,]
          positions <- dplyr::select(ncei, DateTime, lat,lon)
          # original dfs
          ncei_orig <- ncei #<- ncei_orig
          ndbc_orig <- ndbc #<- ndbc_orig
          ncei_orig_count <- nrow(ncei_orig); ncei_orig_start <- min(ncei_orig$DateTime, na.rm =
          TRUE); ncei_orig_end <- max(ncei_orig$DateTime, na.rm = TRUE)
          ndbc_orig_count <- nrow(ndbc_orig); ndbc_orig_start <- min(ndbc_orig$DateTime, na.rm =
          TRUE); ndbc_orig_end <- max(ndbc_orig$DateTime, na.rm = TRUE)

          # checking geographical positions for service visits and buoy adrift, i.e. not in watch
          circle
```

```r
# removing multiple significant places that skew the sorted table results
positions$lat <- as.numeric(positions$lat); positions$lon <- as.numeric(positions$lon)
positions$lat <- signif(positions$lat,3)
positions$lon <- signif(positions$lon,3)

# two methods of finding mode
library(modeest)
lat_mean <- mfv(positions$lat, na_rm = TRUE)
lon_mean <- mfv(positions$lon, na_rm = TRUE)
print(paste0("mean method - lat: ", lat_mean, "; lon: ", lon_mean))
# using a sorted table of value occurrences to find most common lat/lon
lat_tail <- tail(names(sort(table(positions$lat))),1)
lon_tail <- tail(names(sort(table(positions$lon))),1)
print(paste0("sorted table method - lat: ", lat_tail, "; lon: ", lon_tail))
# check positions in ndbc bulk
print(paste0("range - lat: ", range(positions$lat, na.rm = TRUE)[1],"; ",
range(positions$lat, na.rm = TRUE)[2]))
print(paste0("range - lon: ", range(positions$lon, na.rm = TRUE)[1],"; ",
range(positions$lon, na.rm = TRUE)[2]))
rm(positions)

# matching time setkeys to only include geoClean data times from NDBC website
# set as data.tables
library(data.table)
ncei_dt <- data.table(dplyr::select(ncei,DateTime,lat,lon))
ndbc_dt <- data.table(dplyr::select(ndbc,DateTime))
setkey(ncei_dt,DateTime)
setkey(ndbc_dt,DateTime)

# manipulate and subset to include common data, using NDBC as ref dataset
ncei_dt$date <- ncei_dt$DateTime
match_dt <- ncei_dt[ndbc_dt, roll = "nearest" ]
# removing rows with no dates
match_dt <- match_dt[!with(match_dt,is.na(DateTime)| is.na(date)),]
match_dt$DateTime <- match_dt$date
match_dt <- match_dt[,1:3]
# subset
ncei2 <- data.table(ncei)
ndbc2 <- data.table(ndbc)
setkey(ncei2,DateTime)
setkey(ndbc2,DateTime)
ncei <- data.frame(ncei2[match_dt, roll = "nearest" ], stringsAsFactors = FALSE)
ndbc <- data.frame(ndbc2[match_dt, roll = "nearest" ], stringsAsFactors = FALSE)
# remove extra lat/lon in ndbc data
ncei <- ncei[ , -which(names(ncei) %in% c("i.lat", "i.lon"))]
# remove x in column names
colnames(ndbc) <- gsub("X", "", colnames(ndbc))
colnames(ncei) <- gsub("X", "", colnames(ncei))
# remove 0.0100 column if empty
if("0.0100" %in% names(ncei)){if(sum(is.na(ncei$`0.0100`))==nrow(ncei)){ncei$`0.0100` <-
```

```
                    NULL}}
433                 # re-order datasets
434                 idcols <- names(ncei)
435                 # formatting datasets
436                 if(unlist(strsplit(n,"_"))[4]== "stdmet"){
437                       ncei_metadata <- ncei
438                       idcols <- str_subset(idcols, "_metadata", negate = TRUE)
439                       idcols <- str_subset(idcols, "_2", negate = TRUE)
440                       idcols <- gsub("_1", "", idcols)
441                 }
442                 if(grepl("_freq_",name_export)){
443                     if("0.0200" %in% names(ndbc)){ndbc <- dplyr::select(ndbc, all_of(idcols))
444                     }else{ndbc$`0.0200` <- NA;ndbc <- dplyr::select(ndbc, all_of(idcols))}
445                 }
446
447                 # save comparable datasets
448                 ncei_comp <- ncei; ndbc_comp <- ndbc
449                 ncei_comp_count <- nrow(ncei_comp); ncei_comp_start <- min(ncei_comp$DateTime, na.rm =
                    TRUE); ncei_comp_end <- max(ncei_comp$DateTime, na.rm = TRUE)
450                 ndbc_comp_count <- nrow(ndbc_comp); ndbc_comp_start <- min(ndbc_comp$DateTime, na.rm =
                    TRUE); ndbc_comp_end <- max(ndbc_comp$DateTime, na.rm = TRUE)
451                 name_comp <- paste0(name_export,"_comp")
452                 assign(name_comp,ndbc_comp)
453                 name_comp <- gsub("ndbc", "ncei", paste0(name_export,"_comp"))
454                 assign(name_comp,ncei_comp)
455                 # housekeeping
456                 rm(ncei_dt,ndbc_dt,ncei2,ndbc2, match_dt, name_comp)
457
458                 # export as csv and rds
459                 print(paste0("saving comps ",n))
460                 if (!file.exists(paste0(clean_dir,buoy,"/"))) {dir.create((paste0(clean_dir,buoy,"/")))}
461                 # write.table(ncei_comp, paste0(clean_dir,buoy,"/",gsub("ndbc_", "ncei_",n),"_comp.csv"),
                    row.names=FALSE, col.names = TRUE, sep = ",")
462                 # saveRDS(ncei_comp, file = paste0(clean_dir,buoy,"/",gsub("ndbc_", "ncei_",n),"_comp.rds"))
463                 # write.table(ndbc_comp, paste0(clean_dir,buoy,"/",n,"_comp.csv"), row.names=FALSE,
                    col.names = TRUE, sep = ",")
464                 # saveRDS(ndbc_comp, file = paste0(clean_dir,buoy,"/",n,"_comp.rds"))
465
466                 # filter for pre-matching data
467                 start_date_ncei <- min(ncei_comp$DateTime, na.rm = TRUE)
468                 ncei_pre <- dplyr::filter(ncei_orig, ncei_orig$DateTime < start_date_ncei)
469                 # quick check = should == 0
470                 start_date_ndbc <- min(ndbc_comp$DateTime, na.rm = TRUE)
471                 ndbc_pre <- dplyr::filter(ndbc_orig, ndbc_orig$DateTime < start_date_ndbc)
472                 # removing empty dfs
473                 for(i in c("ndbc_pre","ncei_pre")){
474                     dat <- get(i)
475                     if(dim(dat)[1]>0){
476                         # print(paste0("Error: pre filter data available in ",n))
477                         print(paste0("Pre filter data available in ",i))
```

```
                    if(i == "ndbc_pre"){
                        # if pre-ndbc exists, there are no geographical positions that match.
                        # export and delete
                        # export as csv and rds
                        print(paste0("saving pre NDBC data: ",name_export))
                        if (!file.exists(paste0(clean_dir,buoy,"/")))
                        {dir.create((paste0(clean_dir,buoy,"/")))}
                        if (!file.exists(paste0(clean_dir,buoy,"/no_gps/")))
                        {dir.create((paste0(clean_dir,buoy,"/no_gps/")))}
                        # write.table(ndbc_pre,
                        paste0(clean_dir,buoy,"/no_gps/",name_export,"_pre_NDBC.csv"),
                        row.names=FALSE, col.names = TRUE, sep = ",")
                        # saveRDS(ndbc_pre, file =
                        paste0(clean_dir,buoy,"/no_gps/",name_export,"_pre_NDBC.rds"))
                        rm(ndbc_pre)
                    }
                }else{
                    rm(dat)
                    rm(list = ls()[grepl(i, ls())])
                    print(paste0("removing empty df: ",i))
                }
            }
            rm(i, ndbc_orig, ncei_orig)

            # filter the data on these geographical conditions
            dat_ls <- c("ncei_comp", "ncei_pre", "ndbc_comp")
            for(i in dat_ls){
                if(exists(i)){
                    print(paste0("geoCleaning: ",i))
                    dat <- get(i)
                    if(dim(dat)[1]>2){
                        dat <- dplyr::filter(dat, lat >= as.numeric(lat_tail)-as.numeric(GPS_buffer) &
                        lat <= as.numeric(lat_tail)+as.numeric(GPS_buffer))
                        dat <- dplyr::filter(dat, lon >= as.numeric(lon_tail)-as.numeric(GPS_buffer) &
                        lon <= as.numeric(lon_tail)+as.numeric(GPS_buffer))

                        dat <- unique(dat)

                        if(unlist(strsplit(n,"_"))[4]== "stdmet"){

                            #-------------------------------------------------------------------------
                            ------------
                            ## using a sorted table of value occurrences to find outliers in stdmet data

                            #-------------------------------------------------------------------------
                            ------------
                            print("Re-ordering matching stdmet columns")
                            # preserving metadata
                            met <- names(dat)
                            if(length(grep("_metadata",met))>0){
```

```
            dat_metadata <- dplyr::select(dat,c(DateTime, contains("_metadata")))
            dat <- dplyr::select(dat,!contains("_metadata"))
        }

        # identify and remove outlier
        for(c in 4:ncol(dat)){
            # c <- 5
            print(colnames(dat[c]))
            # find outliers
            outlier_table <- sort(as.numeric(names(sort(table(dat[,c])))))
            if(length(outlier_table)>1){
                if(max(outlier_table, na.rm = TRUE) >=
                2*outlier_table[length(outlier_table)-1]){
                    index <- which(dat[,c] == max(outlier_table, na.rm = TRUE))
                    dat[index,c] <- NA
                    print(paste0("outlier of ",max(outlier_table, na.rm = TRUE), "
                    removed from ",colnames(dat[c])))
                }else{print(paste0("no outliers for ",colnames(dat[c])))}
            }else{print(paste0("no outliers for ",colnames(dat[c])))}
        }
        # correcting for directional data outliers
        dir_list <- grep("_direction",names(dat))
        for(dir_column in dir_list){
            print(names(dat)[dir_column])
            direction_ls <- which(dat[,dir_column] > 360)
            print(paste0("outlier indices: ",direction_ls))
            if(length(direction_ls)>0){for(d in direction_ls){dat[d,dir_column] <-
            NA}}
            rm(direction_ls)
        }
        rm(dir_list)

        # rejoining ncei stdmet data and metadata
        if(length(grep("_metadata",met))>0){
            dat <- full_join(dat, dat_metadata, by = "DateTime")
            dat <- dplyr::select(dat, all_of(met))
        }
    }

    # export as csv and rds
    if(i == "ncei_pre"){
        name1 <- gsub("ndbc", "ncei", name_export)
        name2 <- paste0(name1,"_preNDBC_geoClean")
    }else if(i == "ndbc_pre"){
        name2 <- paste0(name_export,"_preNDBC_geoClean")
    }else if(i == "ncei_comp"){
        name1 <- gsub("ndbc", "ncei", name_export)
        name2 <- paste0(name1,"_geoClean")
    }else{
        name2 <- paste0(name_export,"_geoClean")
```

```
                                }
                                # export as csv and rds
                                print("")
                                print(paste0("saving geoCleaned: ",name2))
                                if (!file.exists(paste0(clean_dir,buoy,"/")))
                                {dir.create((paste0(clean_dir,buoy,"/")))}
                                # write.table(dat, paste0(clean_dir,buoy,"/",name2,".csv"), row.names=FALSE,
                                col.names = TRUE, sep = ",")
                                # saveRDS(dat, file = paste0(clean_dir,buoy,"/",name2,".rds"))
                                # save for RData later
                                assign(name2,dat)
                                # housekeeping
                                rm(dat, dat_metadata)
                            }else{print("df dim less than 2 rows")}
                            rm(dat)
                        }else{print(paste0("no preNDBC data for ",i))}
                        rm(i,dat)
                    }
                    # count
                    ncei_geoClean_count <- nrow(ncei_comp); ncei_geoClean_start <- min(ncei_comp$DateTime,
                    na.rm = TRUE); ncei_geoClean_end <- max(ncei_comp$DateTime, na.rm = TRUE)
                    ndbc_geoClean_count <- nrow(ndbc_comp); ndbc_geoClean_start <- min(ndbc_comp$DateTime,
                    na.rm = TRUE); ndbc_geoClean_end <- max(ndbc_comp$DateTime, na.rm = TRUE)

                    # build table
                    ncei_count <- t(data.frame(ncei_orig_count, ncei_orig_start, ncei_orig_end,
                                               ncei_comp_count, ncei_comp_start, ncei_comp_end,
                                               ncei_geoClean_count, ncei_geoClean_start, ncei_geoClean_end))
                    colnames(ncei_count) <- gsub("ndbc", "ncei", n)
                    ncei_count <- data.frame(ncei_count, stringsAsFactors = FALSE)

                    ndbc_count <- t(data.frame(ndbc_orig_count, ndbc_orig_start, ndbc_orig_end,
                                               ndbc_comp_count, ndbc_comp_start, ndbc_comp_end,
                                               ndbc_geoClean_count, ndbc_geoClean_start, ndbc_geoClean_end))
                    colnames(ndbc_count) <- n
                    ndbc_count <- data.frame(ndbc_count, stringsAsFactors = FALSE)

                    # add to table
                    data_count <- cbind(data_count,ncei_count)
                    data_count <- cbind(data_count,ndbc_count)
                    # remove unclean data
                    rm(ncei, ndbc, ncei_comp, ndbc_comp, ncei_count, ndbc_count,
                       ncei_orig_count, ncei_comp_count, ncei_geoClean_count,ncei_orig_start, ncei_orig_end,
                       ncei_comp_start, ncei_comp_end,ncei_geoClean_start,ncei_geoClean_end,
                       ndbc_orig_count, ndbc_comp_count, ndbc_geoClean_count, ndbc_orig_start, ndbc_orig_end,
                       ndbc_comp_start, ndbc_comp_end,ndbc_geoClean_start, ndbc_geoClean_end)
                }
                if(exists("ndbc")){rm(ndbc)}; if(exists("ncei")){rm(ncei)}
            }
```

```r
#-------------------------------------------------------------------------------------------
## attach stdmet metadata to geoCleaned ndbc stdmet dataset
#-------------------------------------------------------------------------------------------
ncei_met <- get(ls(pattern = "ncei_stdmet_geoClean"))
ndbc_met <- get(ls(pattern = "ndbc_stdmet_geoClean"))

# deal with significant places that skew alignment
ndbc_col_num <- grep("wind_direction", colnames(ndbc_met))
ncei_col_num <- grep("wind_direction", colnames(ncei_met))
for(n in ndbc_col_num:ncol(ndbc_met)){if(is.numeric(ndbc_met[,n]) == TRUE){ndbc_met[,n] <-
round(ndbc_met[n],2)}}
for(n in ncei_col_num[1]:ncol(ncei_met)){if(is.numeric(ncei_met[,n]) == TRUE){ncei_met[,n] <-
round(ncei_met[n],2)}}

# select variables to run
var_list <- names(ndbc_met)
# selecting only main variable per sensor to increase computational efficiency
var_list_main <- var_list[!var_list %in% c("DateTime", "lat", "lon", "wind_direction", "wind_gust",
                                           "dominant_wave_period", "average_wave_period",
                                           "mean_wave_direction")]
print(var_list_main)

print(Sys.time())

for(i in var_list_main){
    print(paste0("matching ncei metadata to ndbc data for ",i," for ",buoy))
    print(Sys.time())

    # subset data
    library(dplyr)
    ndbc <- dplyr::select(ndbc_met, DateTime, contains(i))
    ncei <- dplyr::select(ncei_met, DateTime, contains(i))

    # if no data for this variable, skip the following...
    if(sum(is.na(ndbc[i])) != nrow(ndbc)){

        # single sensor
        if(dim(ncei)[2]==3){
              print("handling single sensor")
              ncei$two <- NA; ncei$met <- NA
              colnames(ncei) <- c("DateTime",
              paste0(i,"_1"),paste0(i,"_metadata_1"),paste0(i,"_2"),paste0(i,"_metadata_2"))
        }

        # multiple sensors
        ndbc <- left_join(ndbc,ncei,by = "DateTime")
        # remove rows with no data
        ndbc <- ndbc[!is.na(ndbc[,2]), ]
        # order and re-index rows
        ndbc <- ndbc[order(ndbc$DateTime),]
```

```r
ndbc <- unique(ndbc)
row.names(ndbc) <- 1:nrow(ndbc)
# set up metadata column and select relevant data columns
ndbc$metadata <- NA
ncei_dat_ls <- names(dplyr::select(ncei, contains(i)))

# matching ndbc && ncei primary sensor where secondary data is na
var_index <- which(ndbc[i] == ndbc[ncei_dat_ls[1]] & is.na(ndbc[ncei_dat_ls[3]]))# |
is.na(ndbc[ncei_dat_ls[3]]))
length(var_index)
if(length(var_index)>0){
    print("check for published duplicate metadata")
    for(d in var_index){
        # insert metadata, accounting for NDBC practice of inserting duplicate data
        ndbc$metadata[d] <- ndbc[d,ncei_dat_ls[2]]#}
    }
}else{print("no published duplicate metadata")}

# matching ndbc && ncei primary and secondary sensor - to account for ndbc's ncei practice of
publishing duplicate data
var_index <- which(is.na(ndbc$metadata) & ndbc[i] == ndbc[ncei_dat_ls[1]] & ndbc[i] ==
ndbc[ncei_dat_ls[3]])# | is.na(ndbc[ncei_dat_ls[3]]))
length(var_index)
if(length(var_index)>0){
    print("check for published duplicate metadata")
    for(d in var_index){
        # insert metadata, accounting for NDBC practice of inserting duplicate data
        ndbc$metadata[d] <- ndbc[d,ncei_dat_ls[2]]
    }
}else{print("no published duplicate metadata")}

# ndbc data that DOESN't match ncei primary and secondary sensor - to account for possible
build errors
var_index <- which(is.na(ndbc$metadata) & ndbc[i] != ndbc[ncei_dat_ls[1]] &
ndbc[ncei_dat_ls[1]] == ndbc[ncei_dat_ls[3]])
length(var_index)
if(length(var_index)>0){
    print("check for non-matching sensor data")
    for(d in var_index){
        # insert metadata, accounting for NetCDF build errors
        ndbc$metadata[d] <- ndbc[d,ncei_dat_ls[2]]
        ndbc[d,ncei_dat_ls[1]] <- ndbc[d,i]
    }
}else{print("no non-matching sensor data")}

# ndbc data that matches ncei secondary but DOESN't match primary sensor - logical secondary
sensor usage
var_index <- which(is.na(ndbc$metadata) & ndbc[i] == ndbc[ncei_dat_ls[3]] &
ndbc[ncei_dat_ls[1]] != ndbc[ncei_dat_ls[3]])
length(var_index)
```

```
if(length(var_index)>0){
  print("check for secondary sensor metadata with non-matching primary metadata")
  for(d in var_index){
    # insert metadata
    ndbc$metadata[d] <- ndbc[d,ncei_dat_ls[4]]
  }
}else{print("no only secondary sensor metadata")}

# ndbc data that matches ncei primary but DOESN't match secondary sensor - logical primary sensor usage
var_index <- which(is.na(ndbc$metadata) & ndbc[i] == ndbc[ncei_dat_ls[1]] & ndbc[ncei_dat_ls[1]] != ndbc[ncei_dat_ls[3]])
length(var_index)
if(length(var_index)>0){
    print("check for primary sensor metadata with non-matching secondary metadata")
    for(d in var_index){
        # insert metadata
        ndbc$metadata[d] <- ndbc[d,ncei_dat_ls[2]]
    }
}else{print("no only primary sensor data")}

# Test if exists: ndbc data that DOESN'T match ncei primary or secondary but is still present - possible NDBC data QC
var_index <- which(is.na(ndbc$metadata) & is.na(ndbc[ncei_dat_ls[1]]) & is.na(ndbc[ncei_dat_ls[3]]))
length(var_index)
if(length(var_index)>0){
  print("check for remaining primary sensor metadata")
  for(d in var_index){
    # insert metadata
    ndbc$metadata[d] <- ndbc[d,ncei_dat_ls[2]]
    ndbc[d,ncei_dat_ls[1]] <- ndbc[d,i]
  }
}else{print("no remaining primary sensor metadata")}

# Test if exists: ndbc data that DOESN'T match ncei primary but is still present - possible NDBC rounding issues or data QC
var_index <- which(is.na(ndbc$metadata) & !is.na(ndbc[i]) & !is.na(ndbc[ncei_dat_ls[1]]) & !is.na(ndbc[ncei_dat_ls[2]]) & is.na(ndbc[ncei_dat_ls[3]]) & is.na(ndbc[ncei_dat_ls[4]]))
length(var_index)
if(length(var_index)>0){
    print("check for remaining primary sensor metadata")
    for(d in var_index){
        # insert metadata
        ndbc$metadata[d] <- ndbc[d,ncei_dat_ls[2]]
    }
}else{print("no remaining primary sensor metadata")}

# Test if exists: ndbc data that DOESN'T match ncei secondary but is still present - possible NDBC rounding issues or data QC
```

```
var_index <- which(is.na(ndbc$metadata) & !is.na(ndbc[ncei_dat_ls[3]]) &
!is.na(ndbc[ncei_dat_ls[4]]) & is.na(ndbc[ncei_dat_ls[1]]) & is.na(ndbc[ncei_dat_ls[2]]))
length(var_index)
if(length(var_index)>0){
    print("check for remaining secondary sensor metadata")
    for(d in var_index){
        # insert metadata
        ndbc$metadata[d] <- ndbc[d,ncei_dat_ls[4]]
    }
}else{print("no remaining secondary sensor metadata")}

# Test if exists: remaining primary metadata
var_index <- which(is.na(ndbc$metadata) & is.na(ndbc[ncei_dat_ls[1]]) &
!is.na(ndbc[ncei_dat_ls[2]]) & is.na(ndbc[ncei_dat_ls[3]]) & is.na(ndbc[ncei_dat_ls[4]]))
length(var_index)
if(length(var_index)>0){
    print("check for remaining ndbc blank metadata")
    for(d in var_index){
        # insert metadata
        ndbc$metadata[d] <- ndbc[d,ncei_dat_ls[2]]#}
    }
}else{print("no remaining ndbc blank primary metadata")}

# Test if exists: look for remaining secondary metadata
var_index <- which(is.na(ndbc$metadata) & is.na(ndbc[ncei_dat_ls[1]]) &
is.na(ndbc[ncei_dat_ls[2]]) & is.na(ndbc[ncei_dat_ls[3]]) & !is.na(ndbc[ncei_dat_ls[4]]))
length(var_index)
if(length(var_index)>0){
    print("check for remaining ndbc blank metadata")
    for(d in var_index){
        # insert metadata
        ndbc$metadata[d] <- ndbc[d,ncei_dat_ls[4]]#}
    }
}else{print("no remaining ndbc blank secondary metadata")}

# Test if exists: remaining metadata where no data matches... possible shoreside corrections?
var_index <- which(is.na(ndbc$metadata) & !is.na(ndbc[ncei_dat_ls[1]]) &
!is.na(ndbc[ncei_dat_ls[2]]) &
                    !is.na(ndbc[ncei_dat_ls[3]]) & !is.na(ndbc[ncei_dat_ls[4]]) & ndbc[i] !=
                    ndbc[ncei_dat_ls[1]] &
                    ndbc[i] != ndbc[ncei_dat_ls[3]] & ndbc[ncei_dat_ls[1]] !=
                    ndbc[ncei_dat_ls[3]])
length(var_index)
if(length(var_index)>0){
    print("check for remaining ndbc blank metadata")
    for(d in var_index){
        # insert metadata
        ndbc$metadata[d] <- ndbc[d,ncei_dat_ls[2]]#}
    }
}else{print("no remaining blank metadata")}
```

```r

# Test if exists: another metadata checks with no data matches... possible shoreside corrections?
var_index <- which(is.na(ndbc$metadata) & !is.na(ndbc[ncei_dat_ls[1]]) & !is.na(ndbc[ncei_dat_ls[2]]) &
                     is.na(ndbc[ncei_dat_ls[3]]))
length(var_index)
if(length(var_index)>0){
  print("check for remaining ndbc blank metadata with non-matching ndbc and ncei data")
  for(d in var_index){
    # insert metadata
    ndbc$metadata[d] <- ndbc[d,ncei_dat_ls[2]]#}
  }
}else{print("no remaining blank metadata")}

print(paste0("finished adding metadata to ",i))

# correct bad ncei data
ncei <- ndbc; ncei[,2] <- NULL; ncei[ncol(ncei)] <- NULL
# remove ncei metadata if columns now empty
if(unique(!is.na(ndbc$metadata)) == TRUE){
      ndbc[,ncei_dat_ls[1]] <- NULL; ndbc[,ncei_dat_ls[2]] <- NULL
      ndbc[,ncei_dat_ls[3]] <- NULL; ndbc[,ncei_dat_ls[4]] <- NULL
}

# if the df is not reduced to 3 columns, break
if(dim(ndbc)[2] != 3){quit(save = "no", status = 999)}

if(dim(ndbc)[2]==2){ndbc$metadata <- NA}
colnames(ndbc) <- c("DateTime", i, paste0(i,"_metadata"))

df <- ndbc
# check for old values that matched and resulted in incorrect metadata assignment - only applicable post 2011 with dual sensor deployment
# fill in any NA values
for(j in 1:nrow(df)){
    if(is.na(df[j,3])){tryCatch({df[j,3] <- df[j-1,3]}, error = function(cond){print("can't fill in NA")})}
}
# find first date if not in 2011
index <- which(year(df$DateTime)>=2011)
if(year(df$DateTime[1])>=2011){index <- index[-c(1,2,3,4,5)]}  # adjusting for data that starts after 2011 (i.e. nothing previous to search on below)
# check indices
if(length(index) != 0){
    for(j in index[1]:nrow(df)-4){
        if(!is.na(df[j,3])){
            tryCatch({if(df[j,3] != df[j-1,3] & df[j,3] != df[j+1,3] & df[j-1,3] == df[j+1,3]){df[j,3] <- df[j-1,3]}},error=function(cond){print("")})
            tryCatch({if(df[j,3] != df[j-2,3] & df[j,3] != df[j+2,3] & df[j-2,3] ==
```

```
                                df[j+2,3]){df[j,3] <- df[j-2,3]}},error=function(cond){print("")})
828                             tryCatch({if(df[j,3] != df[j-3,3] & df[j,3] != df[j+3,3] & df[j-3,3] ==
                                df[j+3,3]){df[j,3] <- df[j-3,3]}},error=function(cond){print("")})
829                             tryCatch({if(df[j,3] != df[j-4,3] & df[j,3] != df[j+4,3] & df[j-4,3] ==
                                df[j+4,3]){df[j,3] <- df[j-4,3]}},error=function(cond){print("")})
830                         }#else{print("NA value, skipping...")}
831                     }
832                 }else{print("no pre-2011 data")}
833
834                 # select only metadata and join to main datasets
835                 ndbc <- dplyr::select(df, DateTime, contains("metadata"))
836                 rm(df)
837                 ndbc_met <- left_join(ndbc_met, ndbc, by = "DateTime")
838
839                 # correcting ncei data for final ndbc transferal
840                 if(i != "significant_wave_height" & i != "sea_surface_temperature"){
841                     print("repairing ncei data")
842                     df <- ncei_met
843                     drop_columns <- unique(c(colnames(df)[grep(i, colnames(df))], colnames(ncei)[grep(i,
                        colnames(ncei))]))
844                     print(drop_columns)
845                     df[drop_columns] <- NULL
846                     ncei_met <- left_join(df, ncei, by = "DateTime")
847                     rm(df)
848                 }else{print('not repairing ncei data')}
849
850             }else{
851
852               print(paste0("no data for ",i))
853               df <- dplyr::select(ndbc_met, DateTime,lat)
854               df$lat <- NA
855               colnames(df) <- c("DateTime", paste0(i,"_metadata"))
856               ndbc_met <- left_join(ndbc_met, df, by = "DateTime")
857               rm(df)
858             }
859             rm(ncei, ndbc)
860         }
861         print("end variable met loop")
862         print(Sys.time())
863
864         #-------------------------------------------------------------------------------------
865
866         # transfer ndbc metadata over to variables from single sensors
867         var_names <- names(ndbc_met)
868         if("wind_speed_metadata" %in% var_names){
869             ndbc_met$wind_direction_metadata <- ndbc_met$wind_speed_metadata
870             ndbc_met$wind_gust_metadata <- ndbc_met$wind_speed_metadata
871         }
872         if("significant_wave_height_metadata" %in% var_names){
873             ndbc_met$dominant_wave_period_metadata <- ndbc_met$significant_wave_height_metadata
```

```
874                 ndbc_met$average_wave_period_metadata <- ndbc_met$significant_wave_height_metadata
875                 ndbc_met$mean_wave_direction_metadata <- ndbc_met$significant_wave_height_metadata
876             }
877
878             # re-order ndbc columns
879             ndbc_ord <-
                c("DateTime","lat","lon","wind_direction","wind_direction_metadata","wind_speed","wind_speed_metadata",
                "wind_gust","wind_gust_metadata",
880
                            "significant_wave_height","significant_wave_height_metadata","dominant_wave_period","domi
                            nant_wave_period_metadata","average_wave_period",
881                         "average_wave_period_metadata","mean_wave_direction","mean_wave_direction_metadata"
                            ,"air_pressure_at_sea_level","air_pressure_at_sea_level_metadata",
882
                            "air_temperature","air_temperature_metadata","sea_surface_temperature","sea_surface_tempe
                            rature_metadata","dew_point_temperature","dew_point_temperature_metadata")
883             ndbc_met <- dplyr::select(ndbc_met,all_of(ndbc_ord))
884
885             #-----------------------------------------------------------------------------------------
886
887             # correct ncei
888             var_list_rem <- c("wind_direction", "wind_gust")
889             print(var_list_rem)
890             for(i in var_list_rem){
891                   print(i)
892                   if(any(grepl(i, colnames(ncei_met)))==TRUE){
893                       # subset data
894                       library(dplyr)
895                       ndbc <- dplyr::select(ndbc_met, DateTime, contains(i))
896                       ncei <- dplyr::select(ncei_met, DateTime, contains(i))
897                       # join df
898                       merged_df <- merge(ndbc, ncei, by = "DateTime")
899                       # select relevant data columns
900                       ncei_dat_ls <- names(dplyr::select(ncei, contains(i)))
901
902                       # ndbc data that DOESN't match ncei primary and secondary sensor - to account for possible
                        build errors
903                       var_index <- which(!is.na(merged_df[i]) != !is.na(merged_df[ncei_dat_ls[1]]) &
                        !is.na(merged_df[i]) != !is.na(merged_df[ncei_dat_ls[3]]))
904                       length(var_index)
905                       if(length(var_index)>0){
906                           print("check for non-matching sensor data")
907                           for(d in var_index){merged_df[d,ncei_dat_ls[1]] <- merged_df[d,i]}
908                       }else{print("no non-matching sensor data")}
909
910                       # ndbc data that DOESN't match ncei primary and secondary sensor - to account for possible
                        build errors
911                       var_index <- which(merged_df[i] != merged_df[ncei_dat_ls[1]] & merged_df[ncei_dat_ls[1]] ==
                        merged_df[ncei_dat_ls[3]])
912                       length(var_index)
```

```
913                     if(length(var_index)>0){
914                       print("check for non-matching sensor data")
915                       for(d in var_index){merged_df[d,ncei_dat_ls[1]] <- merged_df[d,i]}
916                     }else{print("no non-matching sensor data")}
917 
918                     # ndbc data that DOESN't match ncei primary and secondary sensor - to account for possible
                        build errors
919                     var_index <- which(!is.na(merged_df[i]) != merged_df[ncei_dat_ls[1]] &
                        merged_df[ncei_dat_ls[1]] == merged_df[ncei_dat_ls[3]])
920                     length(var_index)
921                     if(length(var_index)>0){
922                       print("check for non-matching sensor data")
923                       for(d in var_index){merged_df[d,ncei_dat_ls[1]] <- merged_df[d,i]}
924                     }else{print("no non-matching sensor data")}
925 
926                     # remove ndbc and ncei data
927                     merged_df[,2:3] <- NULL
928                     drop_columns <- grep(i, colnames(ncei_met))
929                     ncei_met[drop_columns] <- NULL
930                     ncei_met <- left_join(ncei_met, merged_df, by = "DateTime")
931                     rm(merged_df, ndbc, ncei, var_index, drop_columns)
932                 }
933             }
934             # re-order columns
935             if(ncol(ncei_met)== 41){
936                 print("ncei data format - dual sst")
937                 ncei_met <- dplyr::select(ncei_met, DateTime, lat, lat_metadata, lon, lon_metadata,
938                                           wind_direction_1, wind_direction_metadata_1, wind_direction_2,
                                              wind_direction_metadata_2,
939                                           wind_speed_1, wind_speed_metadata_1, wind_speed_2, wind_speed_metadata_2,
940                                           wind_gust_1, wind_gust_metadata_1, wind_gust_2, wind_gust_metadata_2,
941                                           significant_wave_height, significant_wave_height_metadata,
                                              dominant_wave_period, dominant_wave_period_metadata,
942                                           average_wave_period, average_wave_period_metadata, mean_wave_direction,
                                              mean_wave_direction_metadata,
943                                           air_pressure_at_sea_level_1, air_pressure_at_sea_level_metadata_1,
                                              air_pressure_at_sea_level_2, air_pressure_at_sea_level_metadata_2,
944                                           air_temperature_1, air_temperature_metadata_1, air_temperature_2,
                                              air_temperature_metadata_2,
945                                           sea_surface_temperature_1,
                                              sea_surface_temperature_metadata_1,sea_surface_temperature_2,
                                              sea_surface_temperature_metadata_2,
946                                           dew_point_temperature_1, dew_point_temperature_metadata_1,
                                              dew_point_temperature_2, dew_point_temperature_metadata_2)
947             }else if(ncol(ncei_met)== 39){
948                 if("sea_surface_temperature" %in% names(ncei_met)){
949                   print("ncei data format - single sst")
950                   ncei_met <- dplyr::select(ncei_met, DateTime, lat, lat_metadata, lon, lon_metadata,
951                                             wind_direction_1, wind_direction_metadata_1, wind_direction_2,
                                                wind_direction_metadata_2,
```

```
                                    wind_speed_1, wind_speed_metadata_1, wind_speed_2, wind_speed_metadata_2,
                                    wind_gust_1, wind_gust_metadata_1, wind_gust_2, wind_gust_metadata_2,
                                    significant_wave_height, significant_wave_height_metadata,
                                    dominant_wave_period, dominant_wave_period_metadata,
                                    average_wave_period, average_wave_period_metadata, mean_wave_direction,
                                    mean_wave_direction_metadata,
                                    air_pressure_at_sea_level_1, air_pressure_at_sea_level_metadata_1,
                                    air_pressure_at_sea_level_2, air_pressure_at_sea_level_metadata_2,
                                    air_temperature_1, air_temperature_metadata_1, air_temperature_2,
                                    air_temperature_metadata_2,
                                    sea_surface_temperature, sea_surface_temperature_metadata,
                                    dew_point_temperature_1, dew_point_temperature_metadata_1,
                                    dew_point_temperature_2, dew_point_temperature_metadata_2)
            }else if("dew_point_temperature" %in% names(ncei_met)){
              print("ncei data format - single dew point temperature")
              dat <- dplyr::select(dat, DateTime, lat, lat_metadata, lon, lon_metadata,
                                    wind_direction_1, wind_direction_metadata_1, wind_direction_2,
                                    wind_direction_metadata_2,
                                    wind_speed_1, wind_speed_metadata_1, wind_speed_2, wind_speed_metadata_2,
                                    wind_gust_1, wind_gust_metadata_1, wind_gust_2, wind_gust_metadata_2,
                                    significant_wave_height, significant_wave_height_metadata,
                                    dominant_wave_period, dominant_wave_period_metadata,
                                    average_wave_period, average_wave_period_metadata, mean_wave_direction,
                                    mean_wave_direction_metadata,
                                    air_pressure_at_sea_level_1, air_pressure_at_sea_level_metadata_1,
                                    air_pressure_at_sea_level_2, air_pressure_at_sea_level_metadata_2,
                                    air_temperature_1, air_temperature_metadata_1, air_temperature_2,
                                    air_temperature_metadata_2,
                                    sea_surface_temperature_1,
                                    sea_surface_temperature_metadata_1,sea_surface_temperature_2,
                                    sea_surface_temperature_metadata_2,
                                    dew_point_temperature, dew_point_temperature_metadata)
            }else{print("error: dim = 39 but dat names don't match")}
        }else if(ncol(dat)== 37){
            print("ncei data format - single dew point")
            dat <- dplyr::select(dat, DateTime, lat, lat_metadata, lon, lon_metadata,
                                  wind_direction_1, wind_direction_metadata_1, wind_direction_2,
                                  wind_direction_metadata_2,
                                  wind_speed_1, wind_speed_metadata_1, wind_speed_2, wind_speed_metadata_2,
                                  wind_gust_1, wind_gust_metadata_1, wind_gust_2, wind_gust_metadata_2,
                                  significant_wave_height, significant_wave_height_metadata,
                                  dominant_wave_period, dominant_wave_period_metadata,
                                  average_wave_period, average_wave_period_metadata, mean_wave_direction,
                                  mean_wave_direction_metadata,
                                  air_pressure_at_sea_level_1, air_pressure_at_sea_level_metadata_1,
                                  air_pressure_at_sea_level_2, air_pressure_at_sea_level_metadata_2,
                                  air_temperature_1, air_temperature_metadata_1, air_temperature_2,
                                  air_temperature_metadata_2,
                                  sea_surface_temperature, sea_surface_temperature_metadata,
                                  dew_point_temperature, dew_point_temperature_metadata)
```

```
985                 }else if(ncol(dat)== 27){
986                     print("ncei data format - single sensors for all")
987                     dat <- dplyr::select(dat, DateTime, lat, lat_metadata, lon, lon_metadata,
988                                          wind_direction, wind_direction_metadata, wind_speed, wind_speed_metadata,
                                             wind_gust, wind_gust_metadata,
989                                          significant_wave_height, significant_wave_height_metadata,
                                             dominant_wave_period, dominant_wave_period_metadata,
990                                          average_wave_period, average_wave_period_metadata, mean_wave_direction,
                                             mean_wave_direction_metadata,
991                                          air_pressure_at_sea_level, air_pressure_at_sea_level_metadata,
                                             air_temperature, air_temperature_metadata,
992                                          sea_surface_temperature, sea_surface_temperature_metadata,
993                                          dew_point_temperature, dew_point_temperature_metadata)
994                 }
995
996                 #-------------------------------------------------------------------------------------
997
998                 # find unique rows only
999                 ndbc_met <- unique(ndbc_met)
1000                ncei_met <- unique(ncei_met)
1001                # rename the rows to reflect unique data
1002                row.names(ndbc_met) <- 1:nrow(ndbc_met)
1003                row.names(ncei_met) <- 1:nrow(ncei_met)
1004
1005                # save file in glob environment
1006                ndbc_name <- ls(pattern = "ndbc_stdmet_geoClean")
1007                ncei_name <- ls(pattern = "ncei_stdmet_geoClean")
1008
1009                assign(ndbc_name, ndbc_met)
1010                assign(ncei_name, ncei_met)
1011
1012                # housekeeping
1013                rm(ncei_met, ndbc_met, ncei_metadata, ndbc_name, ncei_name)
1014
1015                #-------------------------------------------------------------------------------------
1016                ## geoclean unique data
1017                #-------------------------------------------------------------------------------------
1018                print(paste0("GeoCleaning unique datasets for ", buoy))
1019                ncei_list <- ls(pattern = "_ncei_")
1020                remove <- gsub("_ndbc_", "_ncei_", ndbc_list)
1021                ncei_list <- ncei_list[!grepl("_geoClean", ncei_list)]
1022                ncei_list <- ncei_list[!grepl("_comp", ncei_list)]
1023                ncei_list <- ncei_list[!grepl("_sdmet_netcdf", ncei_list)]
1024                ncei_list <- ncei_list[str_remove_all(ncei_list, paste(remove, collapse = "|"))!= ""]
1025                # add extra ndbc data with no matching ncei
1026                if(length(ndbc_ls)>0){ncei_list <- c(ncei_list, ndbc_ls)}
1027                if(length(remainder)>0){ncei_list <- c(ncei_list, remainder)}
1028                # remove buoy info
1029                ncei_list <- gsub(paste0("s_",buoy,"_"), "", ncei_list)
1030                print(ncei_list)
```

```
1031
1032                  if("ncei_sensor_output" %in% ncei_list){
1033                        dat_name <- paste0("s_",buoy,"_ncei_sensor_output")
1034                        dat <- get(dat_name)
1035                        dat <- left_join(dat, ncei_positions, by = "DateTime")
1036                        assign(dat_name,dat)
1037                        rm(dat)
1038                  }
1039
1040                  # load individual datasets and geoclean pairs
1041                  for(n in ncei_list){
1042                        print(n)
1043
1044                        dat_source <- unlist(strsplit(n,"_"))[1]
1045                        name_export <- gsub(paste0(dat_source,"_"),"",n)
1046
1047                        # find matching datasets
1048                        dat <- get(paste0("s_",buoy,"_",n))
1049
1050                        # remove blank rows
1051                        dat <- dat[rowSums(is.na(dat)) != ncol(dat)-1,]
1052                        dat_orig_count <- nrow(dat); dat_orig_start <- min(dat$DateTime, na.rm = TRUE); dat_orig_end <-
                        max(dat$DateTime, na.rm = TRUE)
1053
1054                        # export as csv and rds
1055                        print(paste0("saving orig ",n))
1056                        if (!file.exists(paste0(clean_dir,buoy,"/"))) {dir.create((paste0(clean_dir,buoy,"/")))}
1057                        # write.table(dat,
                        paste0(clean_dir,buoy,"/","s_",buoy,"_",dat_source,"_",name_export,"_orig.csv"),
                        row.names=FALSE, col.names = TRUE, sep = ",")
1058                        # saveRDS(dat, file =
                        paste0(clean_dir,buoy,"/","s_",buoy,"_",dat_source,"_",name_export,"_orig.rds"))
1059
1060                        # checking geographical positions for service visits and buoy adrift, i.e. not in watch circle
1061                        # using a sorted table of value occurrences to find most common lat/lon
1062                        if("lat" %in% names(dat)){
1063                              positions <- dplyr::select(dat,DateTime,lat,lon)
1064                              positions$lat <- as.numeric(positions$lat); positions$lon <- as.numeric(positions$lon)
1065                              positions$lat <- signif(positions$lat,3)
1066                              positions$lon <- signif(positions$lon,3)
1067                        }else{
1068                              print("no gps data, adding data from master positions data")
1069                              positions <- ncei_positions
1070                              ## Use round.Date to round, then format to format
1071                              positions$DateTime <- format(round(positions$DateTime, units="hours"), format="%Y-%m-%d
                              %H:%M:%S")
1072                              positions$DateTime <- lubridate::ydm_hms(positions$DateTime)
1073
1074                              # Join and standardize the data frames
1075                              positions <- left_join(dat,positions, by = "DateTime")
```

```
      positions <- dplyr::select(positions,DateTime,lat,lon)
      positions$lat <- as.numeric(positions$lat); positions$lon <- as.numeric(positions$lon)
      positions$lat <- signif(positions$lat,3); positions$lon <- signif(positions$lon,3)
      # positions <- unique(positions)
}

# two methods of finding mode
library(modeest)
lat_mean <- mfv(positions$lat, na_rm = TRUE)
lon_mean <- mfv(positions$lon, na_rm = TRUE)
print(paste0("mean method - lat: ", lat_mean, "; lon: ", lon_mean))
# using a sorted table of value occurrences to find most common lat/lon
lat_tail <- tail(names(sort(table(positions$lat))),1)
lon_tail <- tail(names(sort(table(positions$lon))),1)
print(paste0("sorted table method - lat: ", lat_tail, "; lon: ", lon_tail))
# check positions in ndbc bulk
print(paste0("range - lat: ", range(positions$lat, na.rm = TRUE)[1],"; ", range(positions$lat,
na.rm = TRUE)[2]))
print(paste0("range - lon: ", range(positions$lon, na.rm = TRUE)[1],"; ", range(positions$lon,
na.rm = TRUE)[2]))

# filter the data on these geographical conditions
if(dat_source == "ndbc"){dat <- left_join(dat, positions, by = "DateTime")}
dat <- dplyr::filter(dat, lat >= as.numeric(lat_tail)-as.numeric(GPS_buffer) & lat <=
as.numeric(lat_tail)+as.numeric(GPS_buffer))
dat <- dplyr::filter(dat, lon >= as.numeric(lon_tail)-as.numeric(GPS_buffer) & lon <=
as.numeric(lon_tail)+as.numeric(GPS_buffer))
dat_clean <- unique(dat)
# count
dat_geoClean_count <- nrow(dat_clean)
dat_geoClean_count <- nrow(dat_clean); dat_geoClean_start <- min(dat_clean$DateTime, na.rm =
TRUE); dat_geoClean_end <- max(dat_clean$DateTime, na.rm = TRUE)

# export as csv and rds
print(paste0("saving geoCleaned ",n))
if (!file.exists(paste0(clean_dir,buoy,"/"))) {dir.create((paste0(clean_dir,buoy,"/")))}
# write.table(dat_clean,
paste0(clean_dir,buoy,"/","s_",buoy,"_",dat_source,"_",name_export,"_geoclean.csv"),
row.names=FALSE, col.names = TRUE, sep = ",")
# saveRDS(dat_clean, file =
paste0(clean_dir,buoy,"/","s_",buoy,"_",dat_source,"_",name_export,"_geoclean.rds"))

# build table
dat_count <- t(data.frame(dat_orig_count, dat_orig_start, dat_orig_end,
                          NA, NA, NA,
                          dat_geoClean_count, dat_geoClean_start,dat_geoClean_end))
colnames(dat_count) <- paste0("s_", buoy, "_",n)
dat_count <- data.frame(dat_count, stringsAsFactors = FALSE)
# add to table
data_count <- cbind(data_count,dat_count)
```

```
1118                 data_count <- data.frame(data_count, stringsAsFactors = FALSE)
1119
1120                 # # remove unclean data
1121                 # rm(list = ls(pattern = name_export))
1122                 # save for RData later
1123                 clean_dat <- paste0("s_",buoy,"_",dat_source,"_",name_export,"_geoClean")
1124                 assign(clean_dat, dat_clean)
1125                 rm(dat, clean_dat, positions, dat_clean, name_export, dat_count, dat_geoClean_count,
                     dat_orig_count)
1126             }
1127
1128             #-------------------------------------------------------------------------------------
1129             ## create station hull count table and save data_count table
1130             #-------------------------------------------------------------------------------------
1131             met_ls <- ls(pattern = "station_metadata")
1132             for(d in met_ls){
1133                 d_name <- paste0(d, "_count")
1134                 d_name <- gsub("ncei_","", d_name)
1135                 dat <- get(d)
1136                 # create empty df
1137                 df <- data.frame(matrix(ncol = 0, nrow = 30))
1138                 df$count <- 1:30
1139                 # find unique data per variable
1140                 depth <- data.frame(1:length(unique(dat$depth)),unique(dat$depth), stringsAsFactors = FALSE);
                     colnames(depth) <- c("count","depth")
1141                 mooring <- data.frame(1:length(unique(dat$mooring)),unique(dat$mooring), stringsAsFactors =
                     FALSE); colnames(mooring) <- c("count","mooring")
1142                 hull <- data.frame(1:length(unique(dat$hull)),unique(dat$hull), stringsAsFactors = FALSE);
                     colnames(hull) <- c("count","hull")
1143                 payload <- data.frame(1:length(unique(dat$payload)),unique(dat$payload), stringsAsFactors =
                     FALSE); colnames(payload) <- c("count","payload")
1144                 # build data frame
1145                 df <- full_join(df,depth, by = "count")
1146                 df <- full_join(df,mooring, by = "count")
1147                 df <- full_join(df,hull, by = "count")
1148                 df <- full_join(df,payload, by = "count")
1149                 df$count <- NULL
1150                 df <- df[rowSums(is.na(df)) != ncol(df),]
1151                 # save data
1152                 print("saving Station Count tables")
1153                 assign(d_name, df)
1154                 # write.table(df, paste0(clean_dir,buoy,"/",d_name,".csv"), row.names=FALSE, col.names = TRUE,
                     sep = ",")
1155                 # saveRDS(df, paste0(clean_dir,buoy,"/",d_name,".rds"))
1156                 # housekeeping
1157                 rm(d_name,dat,df)
1158             }
1159             rm(met_ls)
1160
1161             # Save data count tables
```

```r
print("saving Data Count tables")
colnames(data_count)[1] <- paste0("Station ",buoy)
# write.table(data_count, paste0(clean_dir,buoy,"/","s_",buoy,"_data_counts.csv"), row.names=FALSE,
col.names = TRUE, sep = ",")
data_count_name <- paste0("s_",buoy,"_ndbc_ncei_data_counts")
assign(data_count_name, data_count)

#-------------------------------------------------------------------------------------------
# exporting GeoCleaned datasets
#-------------------------------------------------------------------------------------------

if(switch_set_a == 1){
    print("Exporting Comp and GeoCleaned data")

    ndbc_list <- ls(pattern = "_ndbc_"); ncei_list <- ls(pattern = "_ncei_")
    ndbc_list <- ndbc_list[!grepl("color_", ndbc_list)]
    ndbc_list <- ndbc_list[grepl("_geoClean", ndbc_list)]
    print(paste0("NDBC datasets :", ndbc_list))
    print("")
    ncei_list <- ncei_list[grepl("_geoClean", ncei_list)]
    print(paste0("NCEI datasets :", ncei_list))


    # export and save geoClean dataset
    if (!file.exists(paste0(clean_dir,buoy,"/"))) {dir.create((paste0(clean_dir,buoy,"/")))}
    for(g in c(ndbc_list,ncei_list)){
        if(grepl("_stdmet",g)==TRUE){
            print(g)
            write.table(get(g), paste0(clean_dir,buoy,"/",g,".csv"), row.names=FALSE, col.names =
            TRUE, sep = ",")
            saveRDS(get(g), paste0(clean_dir,buoy,"/",g,".rds"))
        }
    }
    # export to RData
    save(list = ndbc_list, file = paste0(clean_dir,buoy,"/s_",buoy,"_ndbc_comp_geoClean.RData"))
    save(list = ncei_list, file = paste0(clean_dir,buoy,"/s_",buoy,"_ncei_comp_geoClean.RData"))
}else{print("switch set to don't export geoClean datasets")}

rm(ndbc_list,ncei_list, ndbc, ncei, dat_metadata, ncei_metadata)

#-------------------------------------------------------------------------------------------
# stats and plotting data
#-------------------------------------------------------------------------------------------

if(switch_set == 1){

    # if one or more dataset is present but may have different time periods
    print("Working on stats and plots...")

    data_lists <- ls(pattern = buoy)
```

```r
# remove preNDBC, station metadata, data counts and original netcdf data
data_lists <- data_lists[!grepl("_preNDBC_", data_lists)]
data_lists <- data_lists[!grepl("_station_metadata", data_lists)]
data_lists <- data_lists[!grepl("_ndbc_ncei_data_counts", data_lists)]
data_lists <- data_lists[!grepl("_netcdf", data_lists)]
print(data_lists)

#-------------------------------------------------------------------------------------------
# aggregating and plotting
#-------------------------------------------------------------------------------------------
if (!file.exists(paste0(stats_dir,buoy,"/"))) {dir.create((paste0(stats_dir,buoy,"/")))}
if (!file.exists(paste0(fig_dir,buoy,"/"))) {dir.create((paste0(fig_dir,buoy,"/")))}

for(df in data_lists){
    print(df)

    if(length(grep("_sensor_output",df))>0){
          dat_s <- get(df)
          png(paste0(fig_dir, paste0(buoy,"/",df,".png")), width = width1, height = height1)
          mainTitle <- paste0("NDBC Station ", buoy," NCEI Sensor Output")
          plot(dat_s$DateTime, dat_s$sensor_output,
               xlab = "Date", ylab = "1 = Displacement / 2 = Acceleration",
               lty = 5, pch = 0, col = "skyblue2", cex = 1, cex.lab = 1.2,
               panel.first=grid(), main = mainTitle)
          dev.off()
          print(paste0("printing... ",mainTitle))
          rm(dat_s)
    }else if(length(grep("_stdmet",df))>0){
          # GPS data
          print("working on GPS plot...")

          # get data
          dat <- get(df)

          if(dim(dat)[1]>0){

              # # formatting datasets
              if(length(grep("_stdmet",df))>0){
                  # library(tidyverse)
                  dat <- dat %>% dplyr::select(-contains("_metadata"))
                  if(grepl("_ncei",df)==TRUE){
                      # reducing primary and secondary data to one column
                      var_names <- names(dat[4:ncol(dat)])
                      var_names <- var_names[grepl("_1", var_names)]
                      # replace missing primary data with secondary data for plotting and stats
                      for(r in var_names){
                          for(q in 1:nrow(dat)){
                              if(is.na(dat[q,r]) & !is.na(dat[q,gsub("_1", "_2", r)])){
                                dat[q,r] <- dat[q,gsub("_1", "_2", r)]
                                dat[q,gsub("_1", "_2", r)] <- NA
```

```
                    }
                    if(!is.na(dat[q,r])){
                      dat[q,gsub("_1", "_2", r)] <- NA
                    }
                  }
                }
              # remove columns if empty
              dat <- dat[,colSums(is.na(dat))<nrow(dat)]
              # remove _1 from col names
              colnames(dat) <- sub("_1","",colnames(dat))
            }
          }

          if("lat" %in% names(dat)){
            # compute the bounding box
            library(ggplot2)
            library(ggmap)
            library(maps)
            library(mapdata)

            #-------------------------------------------------------------------------------
            --------
            gps <- dplyr::select(dat, lat,lon)
            gps <- gps[complete.cases(gps),]

            bc_bbox <- make_bbox(lat = lat, lon = lon, data = gps)
            bc_bbox
            tryCatch({
                  # grab the maps from google
                  bc_big <- get_map(location = bc_bbox, source = "google", maptype =
                  "satellite")
                  #> Warning: bounding box given to google - spatial extent only approximate.
                  #> converting bounding box to center/zoom specification. (experimental)
                  #> Map from URL :
                  http://maps.googleapis.com/maps/api/staticmap?center=51.86665,-127.98475&zo
                  om=6&size=640x640&scale=2&maptype=terrain&language=en-EN&sensor=false
                  # maptype = c("terrain","terrain-background", "satellite", "roadmap",
                  "hybrid", "toner",
                  #             "watercolor", "terrain-labels", "terrain-lines",
                  "toner-2010",
                  #             "toner-2011", "toner-background", "toner-hybrid",
                  "toner-labels",
                  #             "toner-lines", "toner-lite")
                  # source = c("google", "osm", "stamen")

                  # ggplot code
                  bp <- ggmap(bc_big) +
                    geom_point(data = ncei_positions, mapping = aes(x = lon, y = lat, color
                    = color_ndbc_raw), size = 2) +
                    geom_point(data = dat, mapping = aes(x = lon, y = lat, color =
```

```
                color_ndbc_orig), size = 2)

              # plotting
              b1 <- bp +  labs(title = paste0("Station ",buoy," NDBC GPS Positions\n"),
              x = "Longitude", y = "Latitude", color = "Data Sources") +
                scale_color_hue(labels = c("NCEI NetCDF Raw", "NDBC GeoCleaned
                Data"))#, values = c(color_ndbc_raw, color_ndbc_orig))
              # export the plot
              tiff(paste0(gps_dir,
              paste0("s_",buoy,"_Raw_vs_geoClean_NDBC_GPSPositions.tiff")), width =
              width, height = height, res = res)
              b1
              dev.off()
              # png
              png(paste0(gps_dir,
              paste0(df,"_Raw_vs_geoClean_NDBC_GPSPositions_ggplot.png")), width =
              width, height = height, res = res)
              b1
              dev.off()
              rm(bp,b1)
          }, error = function(e){
              print("no map box plot")
          })

          ## plot positions overall
          png(paste0(gps_dir, paste0(df,"_Raw_vs_geoClean_NDBC_GPSPositions.png")), width
          = width, height = height, res = res)
          mainTitle <- paste0("Station ",buoy," NDBC GPS Positions")
          print(paste0("printing... ",mainTitle))
          plot(ncei_positions$lon, ncei_positions$lat, col = color_ndbc_raw,
               main = mainTitle, ylab = "Latitude (N)", xlab = "Longitude (W)")
          points(dat$lon, dat$lat, col = color_ndbc_orig, pch = 1)
          par(xpd=TRUE)
          legend("topright", inset=c(0,-0.085),
                 box.lty = 0, legend=c("NCEI NetCDF data","NDBC GeoClean data"),
                 col=c(color_ndbc_raw, color_ndbc_orig),
                 pch = c(1,1),
                 cex=1)
          dev.off()
      }else{print(paste0("no gps data in ",df))}

      # standard meteorological data
      print("working on stdmet...")

      # set columns of interest list
      dataCols <-
      c("wind_direction","wind_speed","wind_gust","significant_wave_height","dominant_wave_
      period",

                  "average_wave_period","mean_wave_direction","air_pressure_at_sea_level"
```

```
                ,"air_temperature","sea_surface_temperature")
# remove `air_pressure_at_sea_level` in dataCols2
dataCols_2 <- dataCols[-8]

dat_stats <- dat
dat_stats$lat <-NULL; dat_stats$lon <-NULL; dat_stats$DateTime <- NULL
stats_all <- data.frame(do.call(cbind, lapply(dat_stats, summary)),
stringsAsFactors = FALSE)
# adding in a NA row if no NA's present in dataset (and therefore not added during
summary, which throws off row names)
if("NA's" %in% rownames(stats_all)==FALSE){stats_all[nrow(stats_all)+1,] <- 0}
stats_all[nrow(stats_all)+1,] <- nrow(dat_stats)
rownames(stats_all) <-
c("Min","1st_Qu","Median","Mean","3rd_Qu","Max","NA","Total_Count")
stats_all <- signif(stats_all,3)
stats_all[nrow(stats_all)-1,1:ncol(stats_all)]<-
trunc(stats_all[nrow(stats_all)-1,1:ncol(stats_all)])
stats_all[nrow(stats_all),1:ncol(stats_all)]<-
trunc(stats_all[nrow(stats_all),1:ncol(stats_all)])
write.csv(stats_all,paste0(stats_dir, buoy, "/",df,"_stats.csv"),row.names=TRUE)
rm(dat_stats, stats_all)


#-------------------------------------------------------------------------------
----
# plotting and stats

#-------------------------------------------------------------------------------
----
for(i in dataCols_2){
    # options(warn = -1)

    print(paste0("Working on... " , i))

    if(i %in% names(dat)){
          # print("plotting")
          library(dplyr)
          dat_df <- dplyr::select(dat, DateTime, all_of(i))
          # checking for empty columns
          if(sum(is.na(dat_df[,2])) != nrow(dat_df)){
                # export plot
                source(paste0(data_dir,"plot_stdmet.R"))
                png(paste0(fig_dir, buoy,"/",df,"_",i,".png"), width = width,
                height = height, res = res)
                plot_stdmet(dat_df, i, buoy)
                dev.off()
          }else{print(paste0(i," data are all NA for ",buoy))}
          rm(dat_df)

    }else{print(paste0("no ",i," data for ",buoy))
```

```
                                    tryCatch({
                                          dev.off()
                                    },error=function(cond) {
                                      message("")
                                    })
                              }
                        }
                        rm(dat)
                  }

            }else{
                  #-------------------------------------------------------------------------------------
                  # stats
                  #-------------------------------------------------------------------------------------
                  print(paste0("Working on... " , df))

                  # print("plotting")
                  dat <- get(df)
                  if(unique(sapply(dat[,2:ncol(dat)], class))=="character"){
                        dat[,2:ncol(dat)] <- sapply(dat[,2:ncol(dat)],as.numeric)
                  }
                  # plot_stats_name <- gsub("_geoClean", "",df)
                  plot_stats_name <- df

                  if(dim(dat)[1]>0){
                        # descriptive stats
                        if("lat" %in% names(dat)){
                              stats <- as.data.frame.matrix(summary(dat[,4:ncol(dat)]), stringsAsFactors
                              = FALSE)
                        }else{
                              stats <- as.data.frame.matrix(summary(dat[,2:ncol(dat)]), stringsAsFactors
                              = FALSE)
                        }
                        write.csv(stats,paste0(stats_dir,
                        buoy,"/",plot_stats_name,"_stats.csv"),row.names=FALSE)
                        rm(stats)


                        #-------------------------------------------------------------------------------------
                        ------
                        # aggregating and plotting

                        #-------------------------------------------------------------------------------------
                        ------
                        library(lubridate)
                        # accounting for single years of comparison data- daily, month or yearly grouping
                        if(length(unique(year(dat$DateTime))) >= 2){
                            agg_period <- "annual"
                            dat$grp <- year(dat$DateTime)
                        }
```

```r
if(length(unique(year(dat$DateTime))) < 2){
    if(length(unique(month(dat$DateTime))) <= 1){
        agg_period <- "daily"
        year_plot <- unique(year(dat$DateTime))
        dat$grp <- day(dat$DateTime)
    }else{
        agg_period <- "monthly"
        year_plot <- unique(year(dat$DateTime))
        dat$grp <- month(dat$DateTime)
    }
}
# descriptive stats
if("lat" %in% names(dat)){
    dat <- aggregate(dat[, 3:ncol(dat)-1], list(dat$grp), mean, na.rm = TRUE)
}else {
    dat <- aggregate(dat[, 2:ncol(dat)-1], list(dat$grp), mean, na.rm = TRUE)
}

if("lat" %in% names(dat)){
    dat$lat <- NULL; dat$lon <- NULL
}else{print(paste0("no lat/lon in ",df))}
if("DateTime" %in% names(dat)){
  dat$DateTime <- NULL
}else{print(paste0("no DateTime in ",df))}

# reformat for plotting purposes
dat1 <- dat
is.nan.data.frame <- function(x) do.call(cbind, lapply(x, is.nan))
dat1[is.nan.data.frame(dat1)] <- 0
dat1 <- data.frame(t(dat1),stringsAsFactors = FALSE)
names(dat1) <- dat1[1,]
col_name <- names(dat1)
dat1 <- data.frame(dat1[-1,],stringsAsFactors = FALSE)
names(dat1) <- col_name
library(dplyr)
library(tibble)
dat1 <- tibble::rownames_to_column(dat1, "Frequency")
if(agg_period == "daily" | agg_period == "monthly"){
    if(agg_period == "monthly"){
        colnames(dat1) <-
        c("Frequency",month.abb[as.numeric(gsub("X","",names(dat1[2:ncol(dat1)])))]
        )
    }else{
        colnames(dat1) <- c("Frequency",gsub("X","",names(dat1[2:ncol(dat1)])))
    }
}
dat1$Frequency <- as.numeric(gsub("X","",dat1$Frequency))
dat1$Frequency <- signif(dat1$Frequency, digits = 6)
if(agg_period != "monthly"){
    names_dat1 <- as.numeric(gsub("X","",names(dat1[2:ncol(dat1)])))
```

```
    colnames(dat1) <- c("Frequency",names_dat1)
}
# export as csv
write.table(dat1,
paste0(stats_dir,buoy,"/",plot_stats_name,"_",agg_period,"_mean.csv"),
row.names=FALSE, col.names = TRUE, sep = ",")

# formatting for plots
if(agg_period == "daily"){year_num <- names(dat1); year_num <-
as.numeric(year_num[-1])}
if(agg_period == "monthly"){year_num <- names(dat1);year_num <- year_num[-1]}
if(agg_period == "annual"){year_num <- names(dat1);year_num <-
as.numeric(year_num[-1])}
print(year_num)

for (year1 in year_num){
    source(paste0(data_dir,"plots_spec.R"))
    plot_titles <- paste0(plot_stats_name,"_mean")
    library(R.utils)
    if(unlist(strsplit(plot_titles,"_"))[3]=="ncei") {main_titles <-
    gsub("Ncei","NCEI",gsub("S ","NDBC Station ",str_to_title(gsub("_","
    ",plot_titles))))}
    if(unlist(strsplit(plot_titles,"_"))[3]=="ndbc") {main_titles <-
    gsub("Ndbc","NDBC",gsub("S ","NDBC Station ",str_to_title(gsub("_","
    ",plot_titles))))}
    if(agg_period == "monthly"){
        png(paste0(fig_dir,
        paste0(buoy,"/",plot_titles,year1,"_",year_plot,".png")), width = width1,
        height = height1)
        mainTitle <- paste0(main_titles," ",year1,", ",year_plot)
        plots_spec(dat1, year1, df, mainTitle)
        dev.off()
        print(paste0("printing... ",mainTitle))
    }else if(agg_period == "daily"){
        png(paste0(fig_dir,
        paste0(buoy,"/",plot_titles,year1,"_",year_plot,".png")), width = width1,
        height = height1)
        mainTitle <- paste0(main_titles," ",year1,", ",year_plot)
        plots_spec(dat1,year1, df, mainTitle)
        dev.off()
        print(paste0("printing... ",mainTitle))
    }else{
        png(paste0(fig_dir, buoy,"/",plot_titles,year1,".png"), width = width1,
        height = height1)
        mainTitle <- paste0(main_titles," ",year1)
        plots_spec(dat1,year1, df, mainTitle)
        dev.off()
        print(paste0("printing... ",mainTitle))
    }
}
```

```
                                }else{print(paste0(df," is an empty dataset"))}
                                rm(dat1)
                            } # end plot individual
                        }
                        #-------------------------------------------------------------------------------------

                    }else{print("switch set to don't plot this time")}

                    #-------------------------------------------------------------------------------------

                    # housekeeping
                    rm(list = ls(pattern = buoy))
                    rm(ncei_positions)
                    rm(data_count,a,c,d,i,n,g,p,name1,name2,r,var_index,idcols,index,j,lat_mean,
                    lat_tail,lon_mean,lon_tail,dir_column,q,dat_source,t,

                       dat_geoClean_start,dat_geoClean_end,met,ncei_dat_ls,ncei_ls,ndbc_ls,ndbc_name,ndbc_ord,dat_list,dat_
                       ls,column_names,single_sensor_ls,

                       start_date_ncei,start_date_ndbc,remove,dat_orig_end,dat_orig_start,col_index,col_names,colNames,outl
                       ier_table,var_list,depth,hull,
                       mooring, payload, gps, agg_period, bc_bbox, bc_big, col_name, dat_name, data_lists, dataCols,
                       dataCols_2, df, main, main_titles,
                       mainTitle, name_net, names_dat1, ndbc_col_num, ncei_col_num, plot_stats_name, plot_titles,
                       var,var_names, year_num, xlab,year_num, year1)
                    if(exists("ncei_pre")){rm(ncei_pre)}

                    # start writing to an output file
                    print(paste0("Finishing with geoClean: ",buoy))
                    # sink()

                }else{print("no new buoy data")}

                # print(paste0("Finishing with geoClean: ",buoy))
            }
        }
        # #-------------------------------------------------------------------------------------
        # #-------------------------------------------------------------------------------------
```

US Army Corps
of Engineers®

plot_stdmet.R

```r
plot_stdmet <- function(dat = "dataset", i = "i", buoy = "buoy"){

        ## set plot types
        lty_main <- 5
        pch_main <- 2

        # inputs for main title and ylabel
        xlablist <- c("Significant Wave Height",  "Average Period",  "Dominant Period",  "Mean Wave Direction",
                      "Wind Speed", "Wind Direction", "Wind Gust Speed", "Sea Level Pressure",
                      "Air Temperature", "Water Temperature","Dew Point Temperature")

        ylablist <- c("WVHGT (m)", "AVGPD (s)", "DOMPD (s)", paste("MWDIR (", degree,")"),
                      "WSPD (m/s)", paste0("WDIR (", degree, ")"), "GST (m/s)", "BARO (hPa)",
                      paste0("ATMP (", degree, "C)"),paste0("WTMP (", degree, "C)"), paste0("DEWP (", degree, "C)"))
        #-------------------------------------------------------------------------------------------
        ## NDBC / NCEI inputs set
        #-------------------------------------------------------------------------------------------
        if(i=="significant_wave_height"){main = xlablist[1] ; ylab = ylablist[1]}
        if(i=="average_wave_period" ){main = xlablist[2] ; ylab = ylablist[2]}
        if(i=="dominant_wave_period" ){main = xlablist[3] ; ylab = ylablist[3]}
        if(i=="mean_wave_direction" ){main = xlablist[4] ; ylab = ylablist[4]}
        if(i=="wind_speed"){main = xlablist[5] ; ylab = ylablist[5]}
        if(i=="wind_direction"){main = xlablist[6] ; ylab = ylablist[6]}
        if(i=="wind_gust" ){main = xlablist[7] ; ylab = ylablist[7]}
        if(i=="air_pressure_at_sea_level"){main = xlablist[8] ; ylab = ylablist[8]}
        if(i=="air_temperature"){main = xlablist[9] ; ylab = ylablist[9]}
        if(i=="sea_surface_temperature"){main = xlablist[10]; ylab = ylablist[10]}
        if(i=="dew_point_temperature"){main = xlablist[11]; ylab = ylablist[11]}
        #-------------------------------------------------------------------------------------------

        # plot
        plot(dat[[1]], dat[[2]], main = paste0("NDBC Station ",buoy," ", main), xlab = "Date", ylab = ylab,
             col = "darkblue", type = type, pch = pch, cex = cex)#,  xaxt="n") #lwd = lwd,
        if(i == "significant_wave_height") {abline(v= 0, h = 0.25, col = "black", lty = 2)}
        # my.grid(z, "years", "%Y")
        grid()
}
```

US Army Corps
of Engineers®

plots_spec.R

```r
plots_spec <- function(dat = "dat1",year1 = "year1", df = "df", mainTitle = "mainTitle"){
        par(mar=par1)

        dat_index <- which(names(dat)==year1)
        type_dat <- unlist(strsplit(df,"_"))[4]
        freq1 <- unlist(strsplit(df,"_"))[6]

        dir_type <- c("alpha1", "alpha2", "r1", "r2", "gamma2", "gamma3", "a1", "a2", "b1", "b2","phih", "rhq")
        c11_type <- c("c11", "c11m","C12", "Q12", "C13", "Q13","C22", "C23","C33", "Q23")

        if(unlist(strsplit(type_dat,"_"))[1] %in% dir_type){
                if(sum(dir_type ==  "a1"| dir_type ==  "a2"| dir_type ==  "b1"| dir_type ==  "b2" | dir_type ==
                "phih" | dir_type ==  "rhq")>0){
                        ylab = type_dat
                        ylim <- c(floor(min(dat[,dat_index], na.rm = TRUE)),ceiling(max(dat[,dat_index], na.rm =
                        TRUE)))
                }else{
                        ylab = paste0("Directional Spectrum (",degree,")")
                        ylim <- c(floor(min(dat[,dat_index], na.rm = TRUE))-20,ceiling(max(dat[,dat_index], na.rm =
                        TRUE))+20)
                }
                if(freq1 == "new"){xlim = c(0,0.5)
                }else if(freq1 == "old"){xlim = c(0,0.4)
                }else{xlim = c(0,0.5)
                }

        }else if(unlist(strsplit(type_dat,"_"))[1] %in% c11_type){
                if(unlist(strsplit(type_dat,"_"))[1] == "c11"){ylab = expression('C'[11]*' (m'^2*'/Hz) '['mean'])
                }else if(unlist(strsplit(type_dat,"_"))[1] == "c11m"){ylab = expression('C'[11]*''^m*'
                (m/s'^2*')'^2*'/Hz '['mean'])
                }else if(unlist(strsplit(type_dat,"_"))[1] == "C12"){ylab = expression('C'[12]*' (m/Hz) '['mean'])
                }else if(unlist(strsplit(type_dat,"_"))[1] == "Q12"){ylab = expression('Q'[12]*' (m/Hz) '['mean'])
                }else if(unlist(strsplit(type_dat,"_"))[1] == "C13"){ylab = expression('C'[13]*' (m/Hz) '['mean'])
                }else if(unlist(strsplit(type_dat,"_"))[1] == "Q13"){ylab = expression('Q'[13]*' (m/Hz) '['mean'])
                }else if(unlist(strsplit(type_dat,"_"))[1] == "C22"){ylab = expression('C'[22]*' (1/Hz) '['mean'])
                }else if(unlist(strsplit(type_dat,"_"))[1] == "C23"){ylab = expression('C'[23]*' (1/Hz) '['mean'])
                }else if(unlist(strsplit(type_dat,"_"))[1] == "C33"){ylab = expression('C'[33]*' (1/Hz) '['mean'])
                }else if(unlist(strsplit(type_dat,"_"))[1] == "Q23"){ylab = expression('Q'[23]*' (1/Hz) '['mean'])}
                ylim <- c(floor(min(dat[,dat_index], na.rm = TRUE))-0.2,ceiling(max(dat[,dat_index], na.rm =
                TRUE))+0.2)
                if(freq1 == "new"){xlim = c(0,0.5)
                }else if(freq1 == "old"){xlim = c(0,0.4)
                }else{xlim = c(0,0.5)}
        }

        if(ylim[1] != Inf){
                # dat data
                if(unlist(strsplit(type_dat,"_"))[1] %in% dir_type){
                  plot(dat$Frequency, dat[,dat_index],
                       ylab = ylab, xlab = "Frequency (Hz)",
```

```
                        lty = 5, pch = 0, col = "skyblue2", cex = 1, cex.lab = 1.2,
                        xaxt="n", xlim = xlim, ylim = ylim, panel.first=grid(), main = mainTitle)
                }
                if(unlist(strsplit(type_dat,"_"))[1] %in% c11_type){
                  plot(dat$Frequency, dat[,dat_index],
                        ylab = ylab, xlab = "Frequency (Hz)",
                        type = "o", lty = 5, pch = 0, col = "skyblue2", cex = 1, cex.lab = 1.2,
                        xaxt="n", yaxt="n", xlim = xlim,ylim = ylim, panel.first=grid(), main = mainTitle)
                }

                # axes
                axis(1, seq(xlim[1],xlim[2], by = 0.05))
                if(unlist(strsplit(type_dat,"_"))[1] %in% c11_type){
                        tryCatch(
                                {aty <- seq(ylim[1],ylim[2],by = 0.5)
                                labels <- sapply(aty,function(i)
                                        as.expression(bquote(10^ .(i)))
                                )
                                axis(2,at=aty,labels=labels, las=1)
                                },
                                error = function(cond){
                                        aty <- c(ylim[1],mean(ylim[2]-ylim[1]),ylim[2])
                                        labels <- sapply(aty,function(i)
                                                as.expression(bquote(10^ .(i)))
                                        )
                                        axis(2,at=aty,labels=labels, las=1)
                                }
                        )
                }
        }
}
```

US Army Corps
of Engineers®

create_best_data_5.R

```r
create_best_data_5 <- function(buoys = "list of buoys", data_dir = "data dir"){

      ##-------------------------------------------------------------------------------------------
      ## script to select the best available data from the compared, geographically quality controlled and self-describing datasets
      ## Hall, Candice
      ##-------------------------------------------------------------------------------------------

      ## Actions:
      ## The 'create_best_data_5.R' script loads the 's_buoy#_ndbc/ncei_comp_geoClean.Rdata' container files created in the previous step.
      ## The script tests for any pre-NDBC data that may have been included within the NCEI data sources and reduces any multiple sensor data to a single, best value for each variable.
      ## The script then selects all available NDBC data files, and any non-common NCEI data files for inclusion in the best available dataset.
      ## The selected datasets with self-describing metadata are saved in 's_buoy#_best_data.Rdata' container file within buoy station specific folders.

      #-------------------------------------------------------------------------------------------
      #-------------------------------------------------------------------------------------------
      ## load libraries (local run)
      library(lubridate)
      library(dplyr)
      library(gridExtra)
      library(data.table)
      library(oce)
      library(naniar)
      library(tidyverse)
      library(broom)
      library(openair)

      ## load libraries (HPC run)
      # library(lubridate, lib="/p/home/candice/Rlibs/")
      # library(dplyr, lib="/p/home/candice/Rlibs/")
      # library(gridExtra, lib="/p/home/candice/Rlibs/")
      # library(data.table, lib="/p/home/candice/Rlibs/")
      # # library(oce, lib="/p/home/candice/Rlibs/")
      # library(naniar, lib="/p/home/candice/Rlibs/")
      # library(tidyverse, lib="/p/home/candice/Rlibs/")
      # library(broom, lib="/p/home/candice/Rlibs/")
      # library(openair, lib="/p/home/candice/Rlibs/")

      ##-------------------------------------------------------------------------------------------
      ## set paths
      ##-------------------------------------------------------------------------------------------
      # drive <- "E:/Candice/"
      # drive <- "/p/work/candice/"

      setwd(data_dir)

```

```r
# set input directories
geoClean_dir <- paste0(data_dir,"geoClean_data/data/")

# set new output directories for datasets, stats and figures
if (!file.exists(paste0(data_dir,"best_data/"))) {dir.create((paste0(data_dir,"best_data/")))}
best_data_dir <- paste0(data_dir,"best_data/")

if (!file.exists(paste0(best_data_dir,"data/"))) {dir.create((paste0(best_data_dir,"data/")))}
best_dir <- paste0(best_data_dir,"data/")

##-------------------------------------------------------------------------------------------
## set set parameters common to all plots
##-------------------------------------------------------------------------------------------
xlab = "Date"   # label for x axis
width = 2000    # width of exported plot
height = 1500   # height of exported plot
res = 300
# set plot parameters
width1 = 1000
height1 = 700
par1 = c(5,5,4,4)

# colors for each buoy
color_sd <- "red"
color_ref <- "black"
color_test <- "blue"
# plot type, symbol, size
type = "p"
pch = 20
lwd = 1
cex = 0.5
# set parameters
Delta <- '\U0394'
degree <- '\U00B0'
# my.grid function formats
my.format <- "%m-%d-%Y" # "%m-%Y" (long datasets) or "%m-%d-%Y" (short datasets)
my.period <- "weeks" # "months" (long datasets) or "weeks (short datasets)
# my grid function for plots
my.grid <-function(dataset, my.period = "year", my.format = "%Y"){
     grid(nx=NA, ny=NULL)
     abline(v=axis.POSIXct(1, at=seq(min(dataset[1,1]), max(dataset[nrow(dataset),1]),
                                     by= my.period), format=my.format),
            col = "lightgray", lty = "dotted", lwd = par("lwd"))
}
# function to capitalize string
simpleCap <- function(x) {
     s <- strsplit(x, " ")[[1]]
     paste(toupper(substring(s, 1,1)), substring(s, 2),
           sep="", collapse=" ")
}
```

```
96
97        ##-------------------------------------------------------------------------------------------
98        ## Create single, corrected data set from NDBC and NCEI data sources
99        ##-------------------------------------------------------------------------------------------
100
101       print(buoys)
102
103       for(buoy in buoys){
104
105            # start writing to an output file
106            print(paste0("Starting on ",buoy))
107
108            if(file.exists(paste0(geoClean_dir,buoy,"/s_",buoy,"_ndbc_comp_geoClean.RData"))){
109
110                  sink(paste0(data_dir,"4_create_best_data_",buoy,"_",Sys.Date(),".txt"))
111                  print(paste0("Starting on ",buoy))
112
113                  #-------------------------------------------------------------------------------------
114                  ## read in data if not loaded in global environ
115                  #-------------------------------------------------------------------------------------
116
117                  # Load data
118                  load(paste0(geoClean_dir,buoy,"/s_",buoy,"_ndbc_comp_geoClean.RData"))
119                  load(paste0(geoClean_dir,buoy,"/s_",buoy,"_ncei_comp_geoClean.RData"))
120
121                  #-------------------------------------------------------------------------------------
122                  ## reduce multiple sensors in pre_NDBC NCEI stdmet data to best single sensor values
123                  #-------------------------------------------------------------------------------------
124
125                  # load stdmet data
126                  preNDBC_ls <- ls(pattern = "_ncei_stdmet_preNDBC_")
127                  if(length(preNDBC_ls)>0){
128                        dat <- get(preNDBC_ls[1])
129                        # clear loaded data from input vector
130                        col_names <- names(dat)
131                        secondary_sensor_ls <- col_names[col_names %like% '_2']; secondary_sensor_ls <-
                          secondary_sensor_ls[!grepl('_metadata_', secondary_sensor_ls)]
132                        if("wind_direction_2" %in% col_names){if(sum(is.na(dat$wind_direction_2)) ==
                          nrow(dat)){dat$wind_direction_2 <-NULL; dat$wind_direction_metadata_2 <- NULL; dat <-
                          dplyr::rename(dat, wind_direction=wind_direction_1,
                          wind_direction_metadata=wind_direction_metadata_1)
133                       }else{for(i in 1:nrow(dat)){if(is.na(dat$wind_direction_1[i]) &
                          !is.na(dat$wind_direction_2[i])){dat$wind_direction_1[i]=dat$wind_direction_2[i];
                          dat$wind_direction_metadata_1[i]=dat$wind_direction_metadata_2[i]
134                       dat$wind_direction_2 <-NULL; dat$wind_direction_metadata_2 <- NULL;dat <- dplyr::rename(dat,
                          wind_direction=wind_direction_1, wind_direction_metadata=wind_direction_metadata_1)}}}
135                       }
136                       if("wind_speed_2" %in% col_names){if(sum(is.na(dat$wind_speed_2)) ==
                          nrow(dat)){dat$wind_speed_2 <-NULL; dat$wind_speed_metadata_2 <- NULL; dat <-
                          dplyr::rename(dat, wind_speed=wind_speed_1, wind_speed_metadata=wind_speed_metadata_1)
```

```
}else{for(i in 1:nrow(dat)){if(is.na(dat$wind_speed_1[i]) &
!is.na(dat$wind_speed_2[i])){dat$wind_speed_1[i]=dat$wind_speed_2[i];
dat$wind_speed_metadata_1[i]=dat$wind_speed_metadata_2[i]
dat$wind_speed_2 <-NULL; dat$wind_speed_metadata_2 <- NULL; dat <- dplyr::rename(dat,
wind_speed=wind_speed_1, wind_speed_metadata=wind_speed_metadata_1)}}}
}
if("wind_gust_2" %in% col_names){if(sum(is.na(dat$wind_gust_2)) == nrow(dat)){dat$wind_gust_2
<-NULL; dat$wind_gust_metadata_2 <- NULL; dat <- dplyr::rename(dat, wind_gust=wind_gust_1,
wind_gust_metadata=wind_gust_metadata_1)
}else{for(i in 1:nrow(dat)){if(is.na(dat$wind_gust_1[i]) &
!is.na(dat$wind_gust_2[i])){dat$wind_gust_1[i]=dat$wind_gust_2[i];
dat$wind_gust_metadata_1[i]=dat$wind_gust_metadata_2[i]
dat$wind_gust_2 <-NULL; dat$wind_gust_metadata_2 <- NULL; dat <- dplyr::rename(dat,
wind_gust=wind_gust_1, wind_gust_metadata=wind_gust_metadata_1)}}}
}
if("air_pressure_at_sea_level_2" %in% col_names){if(sum(is.na(dat$air_pressure_at_sea_level_2))
== nrow(dat)){dat$air_pressure_at_sea_level_2 <-NULL; dat$air_pressure_at_sea_level_metadata_2
<- NULL; dat <- dplyr::rename(dat, air_pressure_at_sea_level=air_pressure_at_sea_level_1,
air_pressure_at_sea_level_metadata=air_pressure_at_sea_level_metadata_1)
}else{for(i in 1:nrow(dat)){if(is.na(dat$air_pressure_at_sea_level_1[i]) &
!is.na(dat$air_pressure_at_sea_level_2[i])){dat$air_pressure_at_sea_level_1[i]=dat$air_pressure_a
t_sea_level_2[i];
dat$air_pressure_at_sea_level_metadata_1[i]=dat$air_pressure_at_sea_level_metadata_2[i]
dat$air_pressure_at_sea_level_2 <-NULL; dat$air_pressure_at_sea_level_metadata_2 <- NULL; dat
<- dplyr::rename(dat, air_pressure_at_sea_level=air_pressure_at_sea_level_1,
air_pressure_at_sea_level_metadata=air_pressure_at_sea_level_metadata_1)}}}
}
if("air_temperature_2" %in% col_names){if(sum(is.na(dat$air_temperature_2)) ==
nrow(dat)){dat$air_temperature_2 <-NULL; dat$air_temperature_metadata_2 <- NULL; dat <-
dplyr::rename(dat, air_temperature=air_temperature_1,
air_temperature_metadata=air_temperature_metadata_1)
}else{for(i in 1:nrow(dat)){if(is.na(dat$air_temperature_1[i]) &
!is.na(dat$air_temperature_2[i])){dat$air_temperature_1[i]=dat$air_temperature_2[i];
dat$air_temperature_metadata_1[i]=dat$air_temperature_metadata_2[i]
dat$air_temperature_2 <-NULL; dat$air_temperature_metadata_2 <- NULL; dat <- dplyr::rename(dat,
air_temperature=air_temperature_1, air_temperature_metadata=air_temperature_metadata_1)}}}
}
if("dew_point_temperature_2" %in% col_names){if(sum(is.na(dat$dew_point_temperature_2)) ==
nrow(dat)){dat$dew_point_temperature_2 <-NULL; dat$dew_point_temperature_metadata_2 <- NULL;
dat <- dplyr::rename(dat, dew_point_temperature=dew_point_temperature_1,
dew_point_temperature_metadata=dew_point_temperature_metadata_1)
}else{for(i in 1:nrow(dat)){if(is.na(dat$dew_point_temperature_1[i]) &
!is.na(dat$dew_point_temperature_2[i])){dat$dew_point_temperature_1[i]=dat$dew_point_temperature_
2[i]; dat$dew_point_temperature_metadata_1[i]=dat$dew_point_temperature_metadata_2[i]
dat$dew_point_temperature_2 <-NULL; dat$dew_point_temperature_metadata_2 <- NULL; dat <-
dplyr::rename(dat, dew_point_temperature=dew_point_temperature_1,
dew_point_temperature_metadata=dew_point_temperature_metadata_1)}}}
}
if("sea_surface_temperature_2" %in% col_names){if(sum(is.na(dat$sea_surface_temperature_2)) ==
nrow(dat)){dat$sea_surface_temperature_2 <-NULL; dat$sea_surface_temperature_metadata_2 <-
```

```
          NULL; dat <- dplyr::rename(dat, sea_surface_temperature=sea_surface_temperature_1,
          sea_surface_temperature_metadata=sea_surface_temperature_1)
157       }else{for(i in 1:nrow(dat)){if(is.na(dat$sea_surface_temperature_1[i]) &
          !is.na(dat$sea_surface_temperature_2[i])){dat$sea_surface_temperature_1[i]=dat$sea_surface_temper
          ature_2[i]; dat$sea_surface_temperature_metadata_1[i]=dat$sea_surface_temperature_metadata_2[i]
158       dat$sea_surface_temperature_2 <-NULL; dat$sea_surface_temperature_2 <- NULL; dat <-
          dplyr::rename(dat, sea_surface_temperature=sea_surface_temperature_1,
          sea_surface_temperature_metadata=sea_surface_temperature_metadata_1)}}}
159       }
160  }
161
162  #-----------------------------------------------------------------------------------------
163  ## build best available dataset
164  #-----------------------------------------------------------------------------------------
165  # list ndbc data
166  ndbc_ls <- ls(pattern = paste0("s_",buoy,"_ndbc_"))
167  ndbc_ls <- ndbc_ls[ndbc_ls %in% grep(paste0("_geoClean", collapse = "|"), ndbc_ls, value = T)]
168  print(ndbc_ls)
169  # find common datasets
170  ndbc_ncei_common <- gsub("_ndbc_", "_ncei_", ndbc_ls)
171  print(ndbc_ncei_common)
172  # find ncei datasets
173  ncei_ls <- ls(pattern = paste0("s_",buoy,"_ncei_"))
174  print(ncei_ls)
175  # remove data already in ndbc list from ncei list
176  ncei_ls <- ncei_ls[!ncei_ls %in% grep(paste0(ndbc_ncei_common, collapse = "|"), ncei_ls, value = T)]
177  print(ncei_ls)
178  # select only geoClean data
179  ncei_ls <- ncei_ls[ncei_ls %in% grep(paste0("_geoClean", collapse = "|"), ncei_ls, value = T)]
180  print(ncei_ls)
181  # add station metadata and sensor output data
182  ncei_ls_met <- ls(pattern = "station_metadata_geoClean")
183  ncei_ls_sensor <- ls(pattern = "sensor_output_geoClean")
184
185  best_ls <- unique(c(ndbc_ls, ncei_ls, ncei_ls_met,ncei_ls_sensor))
186  print(best_ls)
187
188  if((length(ndbc_ls) + length(ncei_ls))-length(best_ls) == 0){
189       print("Succesful best data collection")
190  }else{
191       print("WARNING: check best data collection")
192  }
193
194  #-----------------------------------------------------------------------------------------
195  # exporting best datasets
196  #-----------------------------------------------------------------------------------------
197
198  # export and save geoClean dataset
199  if (!file.exists(paste0(best_dir,buoy,"/"))) {dir.create((paste0(best_dir,buoy,"/")))}
200  for(g in best_ls){
```

```
                        print(g)
                        write.table(get(g), paste0(best_dir,buoy,"/",g,".csv"), row.names=FALSE, col.names = TRUE, sep
                        = ",")
                        saveRDS(get(g), paste0(best_dir,buoy,"/",g,".rds"))
                    }
                    # export to RData
                    save(list = best_ls, file = paste0(best_dir,buoy,"/s_",buoy,"_best_data.RData"))
                    rm_ls <- ls(pattern = buoy)
                    rm(list = rm_ls)

                    #-----------------------------------------------------------------------------------------

                    print(paste0("Finished with Best data: ",buoy))
                    sink()
                    print(paste0("Finished with Best data: ",buoy))

                }else{print('no new buoy data')}

        }
    }
    #-----------------------------------------------------------------------------------------
```

US Army Corps
of Engineers®

build_thredds_netcdf_6.R

```r
build_thredds_netcdf_6 <- function(buoys = "list of buoys"){

    ##-------------------------------------------------------------------------------------------
    ## script to create netCDF data from combined NDBC web files and NCEI netcdf files
    ## Hall, Candice
    ##-------------------------------------------------------------------------------------------

    ## netCDF structure is cf compliant as per netCDF NCEI point reference:
    ## https://www.nodc.noaa.gov/data/formats/netcdf/v2.0/point.cdl
    ## cf compliant standard names:
    https://cfconventions.org/Data/cf-standard-names/78/build/cf-standard-name-table.html

    ## Actions:
    ## 1.  Sets data locations
    ## 2.  This step creates monthly netCDF NDBC data files that contains the best available data and metadata
    ##     variables that were selected in the step above.
    ## 3.  The 'build_thredds_netcdf_6.R' function requires a buoy list and a data directory. The script also
    ##     needs access to the 'NDBC_buoy.csv' and 'NDBC_buoy_descriptions.csv' spreadsheets that are found in the
    /data directory.
    ## 4.  NetCDF files are subset to create individual station files for each month and year (year_month)
    ## 5.  NetCDF structure is cf compliant as per netCDF NCEI point reference:
    https://www.nodc.noaa.gov/data/formats/netcdf/v2.0/point.cdl
    ## 6.  NetCDF standard names are cf compliant as per:
    https://cfconventions.org/Data/cf-standard-names/78/build/cf-standard-name-table.html
    ## 7.  Flag conventions are consistent with: Paris. Intergovernmental Oceanographic Commission of UNESCO. 2013.
    Ocean Data Standards,Vol.3:
    ##     Recommendation for a Quality Flag Scheme for the Exchange of Oceanographic and Marine Meteorological
    Data. (IOC Manuals and Guides, 54,
    ##     Vol. 3.) 12 pp. (English.)(IOC/2013/MG/54-3)

    #-------------------------------------------------------------------------------------------
    #-------------------------------------------------------------------------------------------
    # load libraries (local run)
    library(lubridate)
    library(plyr)
    library(dplyr)
    library(gridExtra)
    library(data.table)
    library(oce)
    library(naniar)
    library(tidyverse)
    library(broom)
    library(openair) # polarplots
    library(plotly)
    library(magrittr)
    library(tidyr)
    library(grid)
    library(devtools)
    library(lsr)
    library(stringr)
```

```r
library(RColorBrewer)
library(viridis)
library(colorRamps)
library(ggplot2)
library(ggmap)
library(maps)
library(mapdata)
library(broman)
library(ncdf4)

# # load libraries (HPC run)
# library(lubridate, lib="/p/home/candice/Rlibs/")
# library(plyr, lib="/p/home/candice/Rlibs/")
# library(crayon, lib="/p/home/candice/Rlibs/")
# library(pillar, lib="/p/home/candice/Rlibs/")
# library(dplyr, lib="/p/home/candice/Rlibs/")
# library(gridExtra, lib="/p/home/candice/Rlibs/")
# library(data.table, lib="/p/home/candice/Rlibs/")
# library(oce, lib="/p/home/candice/Rlibs/")
# library(naniar, lib="/p/home/candice/Rlibs/")
# library(tidyverse, lib="/p/home/candice/Rlibs/")
# library(broom, lib="/p/home/candice/Rlibs/")
# library(openair, lib="/p/home/candice/Rlibs/") # polarplots
# library(plotly, lib="/p/home/candice/Rlibs/")
# library(magrittr, lib="/p/home/candice/Rlibs/")
# library(tidyr, lib="/p/home/candice/Rlibs/")
# library(grid, lib="/p/home/candice/Rlibs/")
# library(devtools, lib="/p/home/candice/Rlibs/")
# library(lsr, lib="/p/home/candice/Rlibs/")
# library(stringr, lib="/p/home/candice/Rlibs/")
# library(RColorBrewer, lib="/p/home/candice/Rlibs/")
# library(viridis, lib="/p/home/candice/Rlibs/")
# library(colorRamps, lib="/p/home/candice/Rlibs/")
# library(ggplot2, lib="/p/home/candice/Rlibs/")
# library(ggmap, lib="/p/home/candice/Rlibs/")
# library(maps, lib="/p/home/candice/Rlibs/")
# library(mapdata, lib="/p/home/candice/Rlibs/")
# library(broman, lib="/p/home/candice/Rlibs/")
# library(ncdf4, lib="/p/home/candice/Rlibs/")


#-------------------------------------------------------------------------------------------
#-------------------------------------------------------------------------------------------
# set latest NDBC spreadsheet update from Steven DiNapoli, NDBC contract scientist
DiNapoli_year= 2021

##-----------------------------------------------------------------------------------------
## set paths
##-----------------------------------------------------------------------------------------
drive <- "E:/Candice/"
```

```r
# main dir
data_dir <- paste0(drive, "projects/WaveTrends/data/")
setwd(data_dir)
# set input directories
input_dir <- paste0(data_dir,"best_data/data/")
# set new output directories for datasets
if (!file.exists(paste0(data_dir,"best_netCDF_1/"))){dir.create((paste0(data_dir,"best_netCDF_1/")))}
best_netCDF_dir <- paste0("D:/best_netCDF/")

# function to perform capitalization
simpleCap <- function(x) {s <- strsplit(x, " ")[[1]]; paste(toupper(substring(s, 1,1)), substring(s, 2),sep="", collapse=" ")}
lowerFirst <- function(x) {substr(x, 1, 1) <- tolower(substr(x, 1, 1)); x}

##-------------------------------------------------------------------------------------------
## set presets
##-------------------------------------------------------------------------------------------

# setting flags
# Ref: Paris. Intergovernmental Oceanographic Commission of UNESCO. 2013.Ocean Data Standards,Vol.3:
# Recommendation for a Quality Flag Scheme for the Exchange of Oceanographic and Marine Meteorological
# Data. (IOC Manuals and Guides, 54, Vol. 3.) 12 pp. (English.)(IOC/2013/MG/54-3)
flag_descrip = "Good [1] = Passed documented required QC tests; Not Evaluated [2] = Used for data when no QC test performed or the information on quality is not available; Questionable/Suspect [3] = Failed non-critical documented metric or subjective test(s);
Bad [4] = Failed critical documented QC test(s) or as assigned by the data provider; Missing Data [5] = Used as place holder when data are missing [UNESCO 2013 Ocean Data Standards Vol. 3.]."
flag_good = 1; flag_good_desc = "Passed documented required QC tests"
# flag_notEvaluated = 2; flag_notEvaluated_desc = "Used for data when no QC test performed or the information on quality is not available"
flag_questionableSuspect = 3; flag_questionableSuspect_desc = "Failed non-critical documented metric or subjective test(s)"
flag_bad = 4; flag_bad_desc = "Failed critical documented QC test(s) or as assigned by the data provider"
flag_missingData = 5; flag_missingData_desc = "Used as place holder when data are missing"

# set precision
miss_value <- -999.99
variable_prec_df <- 'double'
variable_prec_flg <- 'integer'
variable_prec_metadata <- 'char'
date_range_increment <- 1

# load buoy stations
list_ndbc <- read.csv(paste0(data_dir,"NDBC_buoys.csv"),header = TRUE)
list_ndbc <- dplyr::filter(list_ndbc, list_ndbc$owner == "NDBC"); list_ndbc_buoy <- as.character(list_ndbc$station)
buoys <- list_ndbc_buoy; print(buoys)
# load buoy descriptions
buoy_desc <- read.csv(paste0(data_dir,"NDBC_buoy_descriptions.csv"),header = TRUE)
buoy_desc$Buoy <- as.numeric(buoy_desc$Buoy); buoy_desc$Description <- as.character(buoy_desc$Description)
```

```
138
139       #-------------------------------------------------------------------------------------------
140       # Creating netCDF files and add flags
141       #-------------------------------------------------------------------------------------------
142
143       for(buoy in buoys){
144            Sys.time()
145
146            print(paste0("starting build thredds on buoy: ", buoy))
147
148            # load buoy description
149            station_info = dplyr::filter(buoy_desc, buoy_desc$Buoy==buoy)
150            station_name= station_info$Description
151            rm(station_info)
152
153            # load buoy data
154            load(file = paste0(input_dir,buoy,"/s_",buoy,"_best_data.RData"))
155
156            # clear unnecessary spectral data from input vector
157            rm(list = ls(pattern = "_gamma"));rm(list = ls(pattern = "_rhq")); rm(list = ls(pattern = "_phih"))
158            rm(list = ls(pattern = "_Q"));rm(list = ls(pattern = "_C12_")); rm(list = ls(pattern = "_C13")); rm(list =
               ls(pattern = "_C22")); rm(list = ls(pattern = "_C33"))
159
160            # list data files
161            data_ls <- ls(pattern = buoy)
162            print(data_ls)
163
164            #-------------------------------------------------------------------------------------------
165            # subset data for time periods and create netcdf files
166            #-------------------------------------------------------------------------------------------
167
168            # find date range
169            # load stdmet data
170            dat <- get(data_ls[data_ls %like% 'ndbc_stdmet'])
171            date_year_start <- year(dat$DateTime[1])
172            date_year_end <- year(dat$DateTime[nrow(dat)])
173            ## for subset runs
174            # date_year_start <- as.integer(2016)
175            # date_year_end <- as.integer(2021)
176            rm(dat)
177            date_range <- seq(date_year_start,date_year_end,date_range_increment)
178            print(date_range)
179
180            for(dateRange in date_range){
181                print(dateRange)
182                # subset per month
183                monthRange <- 1:12
184
185                for(m in monthRange){
186                    print(m)
```

```
m = sprintf("%02d", as.numeric(m))

for(df in data_ls){
    dat <-get(df)
    dat <-dplyr::filter(dat, year(DateTime) == as.numeric(dateRange))
    dat <-dplyr::filter(dat, month(DateTime)== as.numeric(m))
    new_name <- paste0("data_",as.numeric(dateRange),"_",m,"_",df)
    if(dim(dat)[1] > 0){if(grepl("cols_",df)==FALSE){assign(new_name, dat)}}
    rm(dat)
}
rm(new_name,df)

# Create and write a netCDF file

# set new output directories for datasets
if (!file.exists(paste0(best_netCDF_dir,buoy,"/"))){dir.create((paste0(best_netCDF_dir,buoy,"/")))}
ncname <- paste0("s_",buoy,"_best_ndbc_ncei_", as.numeric(dateRange),"_",m)
print(ncname)
ncfname <- paste0(best_netCDF_dir, buoy,"/",ncname, ".nc")
miss_values <- miss_value

# subset relevant datasets
dat_ls <- ls(pattern = paste0("_",dateRange,"_"))
# delete list if no stdmet data present
if(sum(str_count(dat_ls, "_stdmet"))==0){
     rm(list = ls(pattern = paste0("data_",dateRange,"_",m,"_s_",buoy)))
     dat_ls <- vector()
}
print(dat_ls)

#-------------------------------------------------------------------------------------------
# load relevant datasets for dateRange
#-------------------------------------------------------------------------------------------
if(length(dat_ls)>0){
    # load stdmet data
    dat <- get(dat_ls[dat_ls %like% 'ndbc_stdmet'])

    # dealing with pre_NDBC NCEI stdmet data
    if(sum(str_count(dat_ls, "_ncei_stdmet_preNDBC_"))>0){
        col_names <- names(dat)
        secondary_sensor_ls <- col_names[col_names %like% '_2']; secondary_sensor_ls <-
        secondary_sensor_ls[!grepl('_metadata_', secondary_sensor_ls)]
        if("wind_direction_2" %in% col_names){if(sum(is.na(dat$wind_direction_2)) ==
        nrow(dat)){dat$wind_direction_2 <-NULL; dat$wind_direction_metadata_2 <- NULL; dat <-
        dplyr::rename(dat, wind_direction=wind_direction_1,
        wind_direction_metadata=wind_direction_metadata_1)
            }else{for(i in 1:nrow(dat)){if(is.na(dat$wind_direction_1[i]) &
            !is.na(dat$wind_direction_2[i])){dat$wind_direction_1[i]=dat$wind_direction_2[i];
            dat$wind_direction_metadata_1[i]=dat$wind_direction_metadata_2[i]
            dat$wind_direction_2 <-NULL; dat$wind_direction_metadata_2 <- NULL;dat <-
```

```
    dplyr::rename(dat, wind_direction=wind_direction_1,
    wind_direction_metadata=wind_direction_metadata_1)}}}
}
if("wind_speed_2" %in% col_names){if(sum(is.na(dat$wind_speed_2)) ==
nrow(dat)){dat$wind_speed_2 <-NULL; dat$wind_speed_metadata_2 <- NULL; dat <-
dplyr::rename(dat, wind_speed=wind_speed_1, wind_speed_metadata=wind_speed_metadata_1)
    }else{for(i in 1:nrow(dat)){if(is.na(dat$wind_speed_1[i]) &
    !is.na(dat$wind_speed_2[i])){dat$wind_speed_1[i]=dat$wind_speed_2[i];
    dat$wind_speed_metadata_1[i]=dat$wind_speed_metadata_2[i]
                dat$wind_speed_2 <-NULL; dat$wind_speed_metadata_2 <- NULL; dat <-
                dplyr::rename(dat, wind_speed=wind_speed_1,
                wind_speed_metadata=wind_speed_metadata_1)}}}
}
if("wind_gust_2" %in% col_names){if(sum(is.na(dat$wind_gust_2)) ==
nrow(dat)){dat$wind_gust_2 <-NULL; dat$wind_gust_metadata_2 <- NULL; dat <-
dplyr::rename(dat, wind_gust=wind_gust_1, wind_gust_metadata=wind_gust_metadata_1)
    }else{for(i in 1:nrow(dat)){if(is.na(dat$wind_gust_1[i]) &
    !is.na(dat$wind_gust_2[i])){dat$wind_gust_1[i]=dat$wind_gust_2[i];
    dat$wind_gust_metadata_1[i]=dat$wind_gust_metadata_2[i]
                dat$wind_gust_2 <-NULL; dat$wind_gust_metadata_2 <- NULL; dat <-
                dplyr::rename(dat, wind_gust=wind_gust_1,
                wind_gust_metadata=wind_gust_metadata_1)}}}
}
if("air_pressure_at_sea_level_2" %in%
col_names){if(sum(is.na(dat$air_pressure_at_sea_level_2)) ==
nrow(dat)){dat$air_pressure_at_sea_level_2 <-NULL; dat$air_pressure_at_sea_level_metadata_2
<- NULL; dat <- dplyr::rename(dat, air_pressure_at_sea_level=air_pressure_at_sea_level_1,
air_pressure_at_sea_level_metadata=air_pressure_at_sea_level_metadata_1)
    }else{for(i in 1:nrow(dat)){if(is.na(dat$air_pressure_at_sea_level_1[i]) &
    !is.na(dat$air_pressure_at_sea_level_2[i])){dat$air_pressure_at_sea_level_1[i]=dat$air_pr
    essure_at_sea_level_2[i];
    dat$air_pressure_at_sea_level_metadata_1[i]=dat$air_pressure_at_sea_level_metadata_2[i]
                dat$air_pressure_at_sea_level_2 <-NULL;
                dat$air_pressure_at_sea_level_metadata_2 <- NULL; dat <- dplyr::rename(dat,
                air_pressure_at_sea_level=air_pressure_at_sea_level_1,
                air_pressure_at_sea_level_metadata=air_pressure_at_sea_level_metadata_1)}}}
}
if("air_temperature_2" %in% col_names){if(sum(is.na(dat$air_temperature_2)) ==
nrow(dat)){dat$air_temperature_2 <-NULL; dat$air_temperature_metadata_2 <- NULL; dat <-
dplyr::rename(dat, air_temperature=air_temperature_1,
air_temperature_metadata=air_temperature_metadata_1)
    }else{for(i in 1:nrow(dat)){if(is.na(dat$air_temperature_1[i]) &
    !is.na(dat$air_temperature_2[i])){dat$air_temperature_1[i]=dat$air_temperature_2[i];
    dat$air_temperature_metadata_1[i]=dat$air_temperature_metadata_2[i]
                dat$air_temperature_2 <-NULL; dat$air_temperature_metadata_2 <- NULL; dat
                <- dplyr::rename(dat, air_temperature=air_temperature_1,
                air_temperature_metadata=air_temperature_metadata_1)}}}
}
if("dew_point_temperature_2" %in% col_names){if(sum(is.na(dat$dew_point_temperature_2)) ==
nrow(dat)){dat$dew_point_temperature_2 <-NULL; dat$dew_point_temperature_metadata_2 <-
```

```
                    NULL; dat <- dplyr::rename(dat, dew_point_temperature=dew_point_temperature_1,
                    dew_point_temperature_metadata=dew_point_temperature_metadata_1)
249                     }else{for(i in 1:nrow(dat)){if(is.na(dat$dew_point_temperature_1[i]) &
                        !is.na(dat$dew_point_temperature_2[i])){dat$dew_point_temperature_1[i]=dat$dew_point_temp
                        erature_2[i];
                        dat$dew_point_temperature_metadata_1[i]=dat$dew_point_temperature_metadata_2[i]
250                                 dat$dew_point_temperature_2 <-NULL; dat$dew_point_temperature_metadata_2 <-
                                    NULL; dat <- dplyr::rename(dat,
                                    dew_point_temperature=dew_point_temperature_1,
                                    dew_point_temperature_metadata=dew_point_temperature_metadata_1)}}}
251                 }
252                 if("sea_surface_temperature_2" %in% col_names){if(sum(is.na(dat$sea_surface_temperature_2))
                    == nrow(dat)){dat$sea_surface_temperature_2 <-NULL; dat$sea_surface_temperature_metadata_2
                    <- NULL; dat <- dplyr::rename(dat, sea_surface_temperature=sea_surface_temperature_1,
                    sea_surface_temperature_metadata=sea_surface_temperature_1)
253                     }else{for(i in 1:nrow(dat)){if(is.na(dat$sea_surface_temperature_1[i]) &
                        !is.na(dat$sea_surface_temperature_2[i])){dat$sea_surface_temperature_1[i]=dat$sea_surfac
                        e_temperature_2[i];
                        dat$sea_surface_temperature_metadata_1[i]=dat$sea_surface_temperature_metadata_2[i]
254                                 dat$sea_surface_temperature_2 <-NULL; dat$sea_surface_temperature_2 <-
                                    NULL; dat <- dplyr::rename(dat,
                                    sea_surface_temperature=sea_surface_temperature_1,
                                    sea_surface_temperature_metadata=sea_surface_temperature_metadata_1)}}}
255                 }
256             }
257
258             # remove stdmet data from list
259             dat_ls <- dat_ls[!grepl('_stdmet', dat_ls)]
260
261             # load station metadata
262             if(sum(str_count(dat_ls, "_station_metadata"))>0){
263                 dat_station <- get(dat_ls[dat_ls %like% '_station_metadata'])
264                 dat_station$lat <- NULL; dat_station$lon <- NULL
265                 # concatenate df
266                 dat <- left_join(dat, dat_station, by = "DateTime")
267                 rm(dat_station)
268                 # clear loaded data from input vector
269                 dat_ls <- dat_ls[!grepl('_station_metadata', dat_ls)]
270             }else{print(paste0("no station metadata data for ",ncname))}
271
272             # check for wave sensor data - common in old datasets
273             if(sum(str_count(dat_ls, "_sensor_output"))>0){
274                 dat_sensor <- get(dat_ls[dat_ls %like% '_sensor_output'])
275                 if("lat" %in% names(dat_sensor)){dat_sensor$lat <- NULL; dat_sensor$lon <-NULL}
276                 colnames(dat_sensor) <- c("DateTime", "wave_sensor_output")
277                 # concatenate df
278                 dat <- left_join(dat, dat_sensor, by = "DateTime")
279                 rm(dat_sensor)
280                 # clear loaded data from input vector
281                 dat_ls <- dat_ls[!grepl('_sensor_output', dat_ls)]
```

```
}else{print(paste0("no sensor output data for ",ncname))}

# subset time variable for spec matching
dat_time <- dplyr::select(dat, DateTime, lat,lon)
dat_start_date <- as.character(dat_time$DateTime[1])
dat_end_date <- as.character(dat_time$DateTime[nrow(dat_time)])

# load spectral data if present
if(sum(str_count(dat_ls, "_freq"))>0){
    for(df in dat_ls){
        # df <- dat_ls[1]
        df_name <-
        paste0("dat_spec_",gsub("_geoClean","",unlist(strsplit(df,paste0("_",buoy,"_")))[2]))
        dat_spec <- get(df)
        dat_spec$lat <- NULL; dat_spec$lon <- NULL
        # remove any old frequency columns previously added to support data ingestion
        if(grepl("_old_",df)){
             if("0.0100" %in% names(dat_spec)){dat_spec$`0.0100`<-NULL}
             if("0.0200" %in% names(dat_spec)){dat_spec$`0.0200`<-NULL}
        }
        dat_spec <- left_join(dat_time,dat_spec, by = "DateTime")
        assign(df_name,dat_spec)
        assign(df,dat_spec)
        # clear loaded data
        rm(dat_spec, df_name)
    }
}else{print(paste0("no spectral data for ",ncname))}
rm(dat_time)

# clear loaded data from input vector
dat_ls <- dat_ls[!grepl('_freq_', dat_ls)]
# housekeeping
if(length(dat_ls)==0){
    rm(list = ls(pattern = paste0("_",dateRange,"_")))
    rm(ncname)
    print("all data are assimilated")
}else{print("EXTRA DATA - check")}

#-------------------------------------------------------------------------------------
# prep data for netcdf creation
#-------------------------------------------------------------------------------------
print("starting with time data")

# create time variable
data.time <- julian(dat$DateTime, origin = as.POSIXct("1900-01-01 00:00:00", tz = "UTC"))
tunits1 <- "days since 1900-01-01 00:00:00"
dimTime <- ncdim_def("time",tunits1,as.double(data.time))
rm(tunits1, data.time)

# create character variables
```

```
331                     dimnchar <- ncdim_def(name="nchar", unit="", vals=1:nrow(dat), create_dimvar=FALSE )
332
333                     # select variables
334                     var_ls <- names(dat)
335                     var_ls <- var_ls[!(var_ls %in% "DateTime")]
336
337                     #-------------------------------------------------------------------------------------------
338                     # prep data to netcdf file
339                     #-------------------------------------------------------------------------------------------
340                     freq_ls <- ls(pattern = "dat_spec_")
341                     # collate time from the various old and new freq
342                     if(length(freq_ls)>0){
343                         # select wave frequency bands for spec data
344                         if(sum(str_count(freq_ls, "_freq_new"))>0){
345                             dat_spec <- get(freq_ls[grep("_freq_new",freq_ls)[1]])
346                             wave_wpm_new <- as.numeric(names(dat_spec[4:ncol(dat_spec)]))
347                             wave_new_length <- length(wave_wpm_new)
348                             dimwave_wpm_new <-
                                ncdim_def(paste0("waveFrequency_",as.character(length(wave_wpm_new))),"Hz",as.double(wave
                                _wpm_new))
349                             rm(dat_spec, wave_wpm_new)
350                         }
351                         if(sum(str_count(freq_ls, "_freq_old"))>0){
352                             # looping through all old spec to account for 0.01 Hz
353                             old_spec <- freq_ls[grep("_freq_old",freq_ls)]
354                             wave_wpm_old <- vector()
355                             for(old in old_spec){
356                                 df <- get(old)
357                                 if("0.0100" %in% names(df)==TRUE){df$`0.0100`<-NULL}
358                                 if("0.0200" %in% names(df)==TRUE){df$`0.0200`<-NULL}
359                                 cols_old <- names(df)
360                                 wave_wpm_old <- c(wave_wpm_old,cols_old)
361                                 rm(df)
362                             }
363                             wave_wpm_old <- unique(wave_wpm_old);removed <- c("DateTime","lat","lon")
364                             wave_wpm_old <- wave_wpm_old[!wave_wpm_old %in% removed]
365                             wave_old_length <- length(wave_wpm_old)
366                             dimwave_wpm_old <-
                                ncdim_def(paste0("waveFrequency_",as.character(length(wave_wpm_old))),"Hz",as.double(wave
                                _wpm_old))
367                             rm(wave_wpm_old,removed)
368                         }
369                         rm(old_freq, df1,old_spec, old,cols_old)
370
371                         # select variables
372                         var_ls <- c(var_ls,freq_ls)
373                     }else(print(paste0("no freq data for ", dateRange, "_", m)))
374
375                     # create variable dimensions for the netcdf file
376                     for(var_df in var_ls){
```

```
 print(var_df)
# set spectral constants
if(var_df != "lat" & var_df != "lon"){
    if(grepl("_freq_", var_df)){
        # load the data if its frequency
        dat_spec <- get(var_df)
        if(exists("wave_new_length")){name_var <- gsub("new",wave_new_length,var_df)}
        if(exists("wave_old_length")){name_var <- gsub("old",wave_old_length,var_df)}
        var_longname = gsub("_"," ",gsub("dat_spec_","",name_var))
        var_name = lowerFirst(gsub(" ","",simpleCap(var_longname)))
    }else{
        var_longname = gsub("_"," ",var_df)
        var_name = lowerFirst(gsub(" ","",simpleCap(var_longname)))
    }
}
miss_values <- miss_value
variable_prec <- variable_prec_df

# set variable constants
if(var_df == "lat") {var_name = "latitude"; var_units = 'degreesNorth';
var_longname='latitude'; variable_prec= variable_prec_df}
if(var_df == "lon") {var_name = "longitude"; var_units = 'degreesEast';
var_longname='longitude'; variable_prec= variable_prec_df}
if(var_df == "wind_direction"|var_df == "mean_wave_direction"|var_df ==
"wind_direction_metadata"|var_df == "mean_wave_direction_metadata") {var_units = 'degT';
variable_prec = variable_prec_flg}
if(var_df == "wind_speed"|var_df == "wind_gust"|var_df == "wind_speed_metadata"|var_df ==
"wind_gust_metadata") {var_units <- 'm/s'}
if(var_df == "significant_wave_height"|var_df == "significant_wave_height_metadata")
{var_name = "waveHs"; var_units = 'm'}
if(var_df == "dominant_wave_period"|var_df == "dominant_wave_period_metadata") {var_name =
"waveTp"; var_units = 's'}
if(var_df == "average_wave_period"|var_df == "average_wave_period_metadata") {var_name =
"waveTm"; var_units = 's'}
if(var_df == "air_pressure_at_sea_level"|var_df == "air_pressure_at_sea_level_metadata")
{var_units = 'hPa'}
if(var_df == "air_temperature"|var_df == "sea_surface_temperature"|var_df ==
"dew_point_temperature"|var_df == "air_temperature_metadata"|var_df ==
"sea_surface_temperature_metadata"|var_df == "dew_point_temperature_metadata") {var_units =
'Celsius'}
if(var_df == "air_pressure_at_sea_level") {var_name = "surfaceAirPressure"};if(var_df ==
"air_pressure_at_sea_level_metadata") {var_name = "surfaceAirPressureMetadata"}
if(var_df == "air_temperature") {var_name = "surfaceAirTemperature"};if(var_df ==
"air_temperature_metadata") {var_name = "surfaceAirTemperatureMetadata"}
if(var_df == "sea_surface_temperature"){var_name = "surfaceSeaTemperature"};if(var_df ==
"sea_surface_temperature_metadata") {var_name = "surfaceSeaTemperatureMetadata"}
if(var_df == "dew_point_temperature") {var_name = "surfaceDewPointTemperature"};if(var_df
== "dew_point_temperature_metadata") {var_name = "surfaceDewPointTemperatureMetadata"}
if(grepl("_metadata", var_df)==TRUE){variable_prec = variable_prec_metadata; var_units =
''; rm(miss_values)}
```

```
410                 if(var_df == "mooring"|var_df == "hull"|var_df == "payload"){variable_prec =
                    variable_prec_metadata; var_units = ''; rm(miss_values); var_longname = paste0("NDBC
                    ",var_df, " type")}
411                 if(var_df == "significant_wave_height_metadata"|var_df ==
                    "dominant_wave_period_metadata"|var_df == "average_wave_period_metadata"){var_name =
                    paste0(var_name,"Metadata")}
412                 if(var_df == "depth"){var_units = 'm'; var_longname = "sea_floor_depth_below_sea_level";
                    variable_prec = variable_prec_flg}
413                 if(var_df == "wave_sensor_output"){var_units = ''; var_longname =
                    "wave_sensor_output:1=Displacement_2=Acceleration"; variable_prec = variable_prec_flg}
414                 # set freq constants, standard and long names
415                 if(grepl("_freq_", var_df)){
416                     if(grepl("_c11_", var_df)){var_units <- 'm2/Hz'; var_name="waveEnergyDensity";
                        variable_prec=variable_prec_df;var_longname =
                        "sea_surface_wave_variance_spectral_density"}
417                     if(grepl("_c11m_", var_df)){var_units <- '(m/s2)2/Hz';
                        var_name="waveEnergyDensityUncorrected"; variable_prec=variable_prec_df;var_longname =
                        "sea_surface_wave_variance_spectral_density_uncorrected"}
418                     if(grepl("_alpha1_", var_df)){var_units <- 'degT'; var_name="waveAlpha1";var_longname =
                        "mean_wave_direction_for_each_spectrum_frequency_bin_of_the_sea_surface"}
419                     if(grepl("_alpha2_", var_df)){var_units <- 'degT'; var_name="waveAlpha2";var_longname =
                        "principal_wave_direction_for_each_spectrum_frequency_bin_of_the_sea_surface"}
420                     if(grepl("_r1_", var_df)){var_units <- 'unitless'; var_name="waveR1"; var_longname =
                        "first_normalized_polar_coordinate_of_the_Fourier_coefficients"}
421                     if(grepl("_r2_", var_df)){var_units <- 'unitless'; var_name="waveR2"; var_longname =
                        "second_normalized_polar_coordinate_of_the_Fourier_coefficients"}
422                 }
423                 # set long names
424                 if(var_df == "wind_direction") {var_longname='wind_from_direction'}
425                 if(var_df == "wind_direction_metadata") {var_longname='wind_from_direction_metadata'}
426                 if(var_df == "wind_speed") {var_longname='wind_speed'}
427                 if(var_df == "wind_speed_metadata") {var_longname='wind_speed_metadata'}
428                 if(var_df == "wind_gust") {var_longname='wind_speed_of_gust'}
429                 if(var_df == "wind_gust_metadata") {var_longname='wind_speed_of_gust_metadata'}
430                 if(var_df == "significant_wave_height") {var_longname='sea_surface_significant_wave_height'}
431                 if(var_df == "significant_wave_height_metadata")
                    {var_longname='sea_surface_significant_wave_height_metadata'}
432                 if(var_df == "dominant_wave_period")
                    {var_longname='sea_surface_wave_period_at_variance_spectral_density_maximum'}
433                 if(var_df == "dominant_wave_period_metadata")
                    {var_longname='sea_surface_wave_period_at_variance_spectral_density_maximum_metadata'}
434                 if(var_df == "average_wave_period")
                    {var_longname='sea_surface_wave_mean_period_from_variance_spectral_density_second_frequency_m
                    oment'}
435                 if(var_df == "average_wave_period_metadata")
                    {var_longname='sea_surface_wave_mean_period_from_variance_spectral_density_second_frequency_m
                    oment_metadata'}
436                 if(var_df == "mean_wave_direction") {var_longname='sea_surface_wave_from_direction'}
437                 if(var_df == "mean_wave_direction_metadata")
                    {var_longname='sea_surface_wave_from_direction_metadata'}
```

```
438                     if(var_df == "air_pressure_at_sea_level") {var_longname='air_pressure'}
439                     if(var_df == "air_pressure_at_sea_level_metadata") {var_longname='air_pressure_metadata'}
440                     if(var_df == "air_temperature") {var_longname='air_temperature'}
441                     if(var_df == "air_temperature_metadata") {var_longname='air_temperature_metadata'}
442                     if(var_df == "sea_surface_temperature") {var_longname='sea_surface_temperature'}
443                     if(var_df == "sea_surface_temperature_metadata")
                        {var_longname='sea_surface_temperature_metadata'}
444                     if(var_df == "dew_point_temperature") {var_longname='dew_point_temperature'}
445                     if(var_df == "dew_point_temperature_metadata")
                        {var_longname='dew_point_temperature_metadata'}
446                     # set frequency bands
447                     if(grepl("_new", var_df)){var_name = paste0(var_name,"_",wave_new_length, "Frequencies")}
448                     if(grepl("_old", var_df)){var_name = paste0(var_name,"_",wave_old_length, "Frequencies")}
449
450                     # create matrix
451                     if(grepl("_freq_", var_df)){
452                         # subset flag field
453                         df_df <- dplyr::select(dat_spec,DateTime)
454                         # format matrix df
455                         dat_spec$lat <- NULL; dat_spec$lon <- NULL; dat_spec$DateTime <- NULL
456                         # create matrix
457                         mat_df <- matrix(t(dat_spec), nrow=nrow(dat_spec),ncol=ncol(dat_spec))
458                         df_name <- paste0("df_",var_name)
459                         # assign(df_name, mat_df)
460                         assign(df_name, mat_df)
461                         rm(df_name, mat_df)
462                         # create flag field
463                         df_df$flag <-flag_good
464                         for(r in 1:nrow(df_df)){if(is.na(df_df[r,1])){df_df[r,2]=flag_missingData}}
465                         if(grepl("_preNDBC", var_df)){df_df$flag <-flag_questionableSuspect}
466                         # subset df
467                         df_df_flag <- dplyr::select(df_df,flag)
468                         # rename flag df
469                         df_name <- paste0("df_",var_name,"_flag")
470                         assign(df_name, df_df_flag)
471                         rm(df_df, df_df_flag, df_name,r)
472                     }else{
473                         # create df
474                         df_df <- dplyr::select(dat,all_of(var_df))
475                         # create flag field
476                         df_df$flag <-flag_good
477                         for(r in 1:nrow(df_df)){if(is.na(df_df[r,1])){df_df[r,2]=flag_missingData}}
478                         # subset df
479                         df_df_flag <- dplyr::select(df_df,flag)
480                         df_df$flag <- NULL
481                         # rename df
482                         df_name <- paste0("df_",var_name)
483                         assign(df_name, df_df)
484                         df_name <- paste0("df_",var_name,"_flag")
485                         assign(df_name, df_df_flag)
```

```
                rm(df_df, df_df_flag, df_name,r)
            }

            # Add dimensions and variables which accompany the dimensions (avoided by create_dimvar =
            FALSE)
            if(grepl("_metadata", var_df)){
                vari_df <- ncvar_def(name=var_name, units=var_units, longname=var_longname,
                dim=list(dimTime,dimnchar), prec=variable_prec)
            }else if(var_df == "mooring"|var_df == "hull"|var_df == "payload"){
                vari_df <- ncvar_def(name=var_name, units=var_units, longname=var_longname,
                dim=list(dimTime,dimnchar), prec=variable_prec)
            }else if(var_df == "lat" | var_df == "lon"){
                vari_df <- ncvar_def(name=var_name, units=var_units, longname=var_longname,
                dim=list(dimTime), missval=miss_values, prec=variable_prec)
            }else if(grepl("_freq_old", var_df)){
                vari_df <- ncvar_def(name=var_name, units=var_units, longname=var_longname,
                dim=list(dimwave_wpm_old,dimTime), missval=miss_values, prec=variable_prec)
            }else if(grepl("_freq_new", var_df)){
                vari_df <- ncvar_def(name=var_name, units=var_units, longname=var_longname,
                dim=list(dimwave_wpm_new,dimTime), missval=miss_values,  prec=variable_prec)
            }else{
                vari_df <- ncvar_def(name=var_name, units=var_units, longname=var_longname,
                dim=list(dimTime), missval=miss_values)
            }
            df_name <- paste0("var_",var_name)
            assign(df_name, vari_df)
            rm(df_name, vari_df)
            rm(var_longname, var_units, var_name)
        }
        rm(var_df)
        rm(list = ls(pattern = "dim_"))
        if(exists("dat_spec")){rm(dat_spec)}
        rm(list = ls(pattern = "dat_spec_"))
        if(exists("wave_old_length")){rm(wave_old_length)}
        if(exists("wave_new_length")){rm(wave_new_length)}

        #-------------------------------------------------------------------------------------------
        # prep flag data for netcdf file
        #-------------------------------------------------------------------------------------------
        flag_ls <- ls(pattern = "_flag")
        miss_values <- miss_value

        # function to perform capitalization
        simpleCap <- function(x) {
            s <- strsplit(x, " ")[[1]]
            paste(toupper(substring(s, 1,1)), substring(s, 2),
                  sep="", collapse=" ")
        }
        lowerFirst <- function(x) {substr(x, 1, 1) <- tolower(substr(x, 1, 1));x}

```

```r
# create dimensions for the flag data
for(fl in flag_ls){
    var_name = gsub("df_","",fl)
    var_longname = gsub("_"," ",tolower(gsub("([a-z])([A-Z])", "\\1 \\2", var_name)))
    var_longname <- gsub("frequencies", " frequencies", var_longname)
    var_name = gsub("_flag","Flag",var_name)
    vari_df <- ncvar_def(name=var_name, units="none", longname=var_longname, dim=list(dimTime),
    missval=miss_values, prec=variable_prec_flg)
    df_name <- paste0("var_",var_name)
    assign(df_name, vari_df)
    rm(df_name, vari_df,var_longname, var_name)
}
rm(fl, flag_ls, dimTime)

#-----------------------------------------------------------------------------------------
# build netcdf file
#-----------------------------------------------------------------------------------------
rm(var__flag, var_df, var_list, var_units)

# create a list of all variables to add them all at once
var_list <- ls(pattern = "var_"); var_list
var_list <-var_list[!var_list %in% "var_ls"]; var_list
if("var_list" %in% var_list){var_list<- var_list[!var_list %in% "var_list"]}
var_list

# create list for quick add to con nc file
vars <- list()
for(i in var_list){vars[[gsub("var_","",i)]] <- get(i)}#print(i)
rm(i)

# Make the file
# Create a new empty netcdf file.
con <- nc_create(ncfname, vars)#, verbose = TRUE)
rm(ncfname, vars)

# This variable was implicitly created by the dimension, so just specifying it by name
ncatt_put(con, 'time', 'standard_name', 'time')
ncatt_put(con, var_longitude, 'axis', 'X')
ncatt_put(con, var_latitude, 'axis', 'Y')
ncatt_put(con, 'time', 'axis', 'T')

# Add some extra attributes
for(var_df in var_list){
    print(var_df)
    std_name <- gsub("var_","",var_df)
    print(std_name)
    ndbc_blurb <- '(NDBC,2018). https://www.ndbc.noaa.gov/measdes.shtml'
    DiNapoli_blurb <- 'Source:NDBC metadata spreadsheet (DiNapoli, '
    metadata_blurb <- paste0(DiNapoli_blurb, DiNapoli_year,') / NDBC NCEI netCDF metadata.')
    # set standard names
```

```
578                 if(std_name == "latitude") {var_standard = "latitude"; desc_name="latitude"}
579                 if(std_name == "longitude") {var_standard = "longitude"; desc_name="longitude"}
580                 if(std_name == "mooring"){var_standard = "NDBC_mooring_type";
                    desc_name=paste0(DiNapoli_blurb, DiNapoli_year)}
581                 if(std_name == "hull"){var_standard = "NDBC_hull_type"; desc_name=paste0(DiNapoli_blurb,
                    DiNapoli_year)}
582                 if(std_name == "payload"){var_standard = "NDBC_payload_type";
                    desc_name=paste0(DiNapoli_blurb, DiNapoli_year)}
583                 if(std_name == "depth"){var_standard = "sea_floor_depth_below_mean_sea_level";
                    desc_name=paste0(DiNapoli_blurb, DiNapoli_year)}
584                 if(std_name == "waveSensorOutput"){var_standard = ''; desc_name = "1=Displacement;
                    2=Acceleration. If the sensor output is displacement, waveEnergyDensity units m2/Hz. If the
                    sensor output is acceleration, waveEnergyDensityUncorrected units are (m/s2)2/Hz."}
585
586                 if(std_name == "windDirection") {var_standard = 'wind_from_direction';
                    desc_name=paste0('Wind direction (the direction the wind is coming from in degrees
                    clockwise from true N) during the same period used for wind speed ',ndbc_blurb)}
587                 if(std_name == "windDirectionMetadata") {var_standard = 'wind_from_direction_metadata';
                    desc_name=paste0('wind direction metadata. ',metadata_blurb)}
588                 if(std_name == "windSpeed") {var_standard = 'wind_speed'; desc_name=paste0('wind speed
                    (m/s) ',ndbc_blurb)}
589                 if(std_name == "windSpeedMetadata") {var_standard = 'wind_speed_metadata';
                    desc_name=paste0('wind speed metadata. ',metadata_blurb)}
590                 if(std_name == "windGust") {var_standard = 'wind_speed_of_gust'; desc_name=paste0('Peak 5
                    or 8 second gust speed (m/s).  ',ndbc_blurb)}
591                 if(std_name == "windGustMetadata") {var_standard = 'wind_speed_of_gust_metadata';
                    desc_name=paste0('wind gust metadata. ',metadata_blurb)}
592
593                 if(std_name == "waveHs") {var_standard = 'sea_surface_significant_wave_height';
                    desc_name=paste0('Significant wave height (meters) is calculated as the average of the
                    highest one-third of all of the wave heights during the 20-minute sampling period
                    ',ndbc_blurb)}
594                 if(std_name == "waveHsMetadata") {var_standard =
                    'sea_surface_significant_wave_height_metadata'; desc_name=paste0('sea surface significant
                    wave height metadata. ',metadata_blurb)}
595                 if(std_name == "waveTp") {var_standard =
                    'sea_surface_wave_period_at_variance_spectral_density_maximum'; desc_name=paste0('Dominant
                    wave period (seconds) is the period with the maximum wave energy ',ndbc_blurb)}
596                 if(std_name == "waveTpMetadata") {var_standard =
                    'sea_surface_wave_period_at_variance_spectral_density_maximum_metadata';
                    desc_name=paste0('sea surface wave period at variance spectral density maximum metadata.
                    ',metadata_blurb)}
597                 if(std_name == "waveTm") {var_standard =
                    'sea_surface_wave_mean_period_from_variance_spectral_density_second_frequency_moment';
                    desc_name=paste0('Average wave period (seconds) of all waves during the 20-minute period
                    ',ndbc_blurb)}
598                 if(std_name == "waveTmMetadata") {var_standard =
                    'sea_surface_wave_mean_period_from_variance_spectral_density_second_frequency_moment_metadata
                    '; desc_name=paste0('sea surface wave mean period from variance spectral density second
                    frequency moment metadata. ',metadata_blurb)}
```

```
599                     if(std_name == "meanWaveDirection") {var_standard = 'sea_surface_wave_from_direction';
                        desc_name=paste0('The direction from which the waves at the dominant period are coming. The
                        units are degrees from true North, increasing clockwise, with North as 0 (zero) degrees and
                        East as 90 degrees ',ndbc_blurb)}
600                     if(std_name == "meanWaveDirectionMetadata") {var_standard =
                        'sea_surface_wave_from_direction_metadata'; desc_name=paste0('sea surface wave from
                        direction metadata. ',metadata_blurb)}
601
602                     if(std_name == "surfaceAirPressure") {var_standard = 'air_pressure'; desc_name=paste0('Sea
                        level pressure (hPa). For the Great Lakes buoys, the recorded pressure is reduced to sea
                        level using the method described in NWS Technical Procedures Bulletin 291 (11/14/80)
                        ',ndbc_blurb)}
603                     if(std_name == "surfaceAirPressureMetadata") {var_standard = 'air_pressure_metadata';
                        desc_name=paste0('air pressure metadata. ',metadata_blurb)}
604                     if(std_name == "surfaceAirTemperature") {var_standard = 'air_temperature';
                        desc_name=paste0('Air temperature (Celsius) ',ndbc_blurb)}
605                     if(std_name == "surfaceAirTemperatureMetadata") {var_standard = 'air_temperature_metadata';
                        desc_name=paste0('air temperature metadata. ',metadata_blurb)}
606                     if(std_name == "surfaceSeaTemperature") {var_standard = 'sea_surface_temperature';
                        desc_name=paste0('Sea surface temperature (Celsius) ',ndbc_blurb)}
607                     if(std_name == "surfaceSeaTemperatureMetadata") {var_standard =
                        'sea_surface_temperature_metadata'; desc_name=paste0('sea surface temperature metadata.
                        ',metadata_blurb)}
608                     if(std_name == "surfaceDewPointTemperature") {var_standard = 'dew_point_temperature';
                        desc_name=paste0('Dewpoint temperature taken at the same height as the air temperature
                        measurement ',ndbc_blurb)}
609                     if(std_name == "surfaceDewPointTemperatureMetadata") {var_standard =
                        'dew_point_temperature_metadata'; desc_name=paste0('dew point temperature metadata.
                        ',metadata_blurb)}
610                     # set freq standard names
611                     if(grepl("Frequencies", std_name)){
612                           if(grepl("EnergyDensity", std_name)){var_standard =
                              'sea_surface_wave_variance_spectral_density'; desc_name="Energy density, displacement
                              in m2/Hz, for each frequency bin"}
613                           if(grepl("waveEnergyDensityUncorrected", std_name)){var_standard =
                              'sea_surface_wave_variance_spectral_density_uncorrected'; desc_name="Uncorrected
                              energy density, acceleration in (m/s2)2/Hz, for each frequency bin"}
614                           if(grepl("Alpha1",
                              std_name)){var_standard='mean_wave_direction_at_specified_frequency';
                              desc_name="alpha1 is the mean wave direction, in degrees from true North, for each
                              spectrum frequency bin of the sea surface"}
615                           if(grepl("Alpha2",
                              std_name)){var_standard='principal_wave_direction_at_specified_frequency';
                              desc_name="alpha2 is the principal wave direction, in degrees from true North, for
                              each spectrum frequency bin of the sea surface. alpha2 has ambiguous results in using
                              the arctangent function with the Fourier Coefficients (b2,a2). When necessary, NDBC
                              adds 180 degrees to alpha2 in order to minimize the difference between alpha1 and
                              alpha2"}
616                           if(grepl("R1",
                              std_name)){var_standard='first_normalized_polar_coordinate_of_the_Fourier_coefficients'
```

```
                    ; desc_name="r1 is the nondimensional first normalized polar coordinates of the
                    Fourier coefficients"}
617                 if(grepl("R2",
                    std_name)){var_standard='second_normalized_polar_coordinate_of_the_Fourier_coefficients
                    '; desc_name="r2 is the nondimensional second normalized polar coordinates of the
                    Fourier coefficients"}
618             }
619             # overwrite std name if a flag
620             if(grepl("Flag", std_name)){var_standard = "quality_flag"; desc_name=
                paste0(std_name,"QualityFlag. ",flag_descrip)}
621
622             # add standard_name
623             ncatt_put(con, gsub("var_","",var_df), 'standard_name', var_standard)
624             rm(var_standard)
625
626             # add if present (i.e. listed above)
627             if(exists('desc_name')){
628                 # add description name
629                 ncatt_put(con, gsub("var_","",var_df), 'description_name', desc_name)
630                 rm(desc_name)
631             }
632             # add coordinates
633             if(var_df != "var_latitude" & var_df != "var_longitude"){
634                 if(grepl("Metadata", var_df)==FALSE){
635                     ncatt_put(con, gsub("var_","",var_df), 'coordinates', 'latitude longitude')
636                 }
637             }
638             # add data values to nc file
639             df_name <- gsub("var_", "",var_df)
640             if(grepl("Flag", df_name)){df_name = gsub("Flag","_flag",df_name)}
641             df <- get(paste0("df_",df_name))
642             if(grepl("_flag", df_name)){df_name = gsub("_flag","Flag",df_name)}
643             # print(paste0("add data: ",df_name))
644             if(grepl("Frequencies", df_name)){
645                 if(grepl("Flag", df_name)){
646                     ncvar_put(con, df_name, df[,1])
647                 }else{
648                     ncvar_put(con, df_name, df)
649                 }
650             }else{
651                 ncvar_put(con, df_name, df[,1])
652             }
653             rm(df_name, df)
654             if(exists("var_standard")){rm(var_standard)}
655             if(exists("desc_name")){rm(desc_name)}
656         }
657         rm(var_df)
658
659         # housekeeping
660         rm(list = ls(pattern = "var_"))
```

```
661                 rm(list = ls(pattern = "df_"))
662
663                 #-------------------------------------------------------------------------------------
664                 # add global attributes data for netcdf file
665                 #-------------------------------------------------------------------------------------
666
667                 ncatt_put(nc=con,varid=0, attname="id", attval=as.character(buoy), prec="int")
668                 ncatt_put(nc=con,varid=0, attname="naming_authority", attval = "WMO", prec="char")
669                 ncatt_put(nc=con,varid=0, attname="ioos_id", attval =
                    paste0("urn:ioos:station:wmo:",as.character(buoy)), prec="char")
670                 ncatt_put(nc=con,varid=0, attname="wmo_id", attval = as.character(buoy), prec="char")
671                 ncatt_put(nc=con,varid=0, attname="institution", attval = "National Data Buoy Center",
                    prec="char")
672                 ncatt_put(nc=con,varid=0, attname="institution_abbreviation", attval = "NDBC", prec="char")
673                 ncatt_put(nc=con,varid=0, attname="title", attval = "NDBC description: Meteorological and
                    Oceanographic Data Collected from the National Data Buoy Center\'s Weather Buoys", prec="char")
674                 ncatt_put(nc=con,varid=0, attname="summary", attval = "NDBC description: Over 100 moored
                    weather buoys have been deployed in U.S. coastal and offshore waters. Weather buoy data
                    typically include barometric pressure, wind direction, speed and gust, air temperature, sea
                    water temperature, waves, and relative humidity. Weather buoys also measure wave energy spectra
                    from which significant wave height, dominant wave period, average wave period and mean wave
                    direction are derived.", prec="char")
675                 if(length(station_name)>0){ncatt_put(nc=con,varid=0, attname="station_name", attval =
                    station_name, prec="char")}
676                 ncatt_put(nc=con,varid=0, attname="history", attval = 'The data were collected by the National
                    Data Buoy Center (NDBC) and archived on their website (https://www.ndbc.noaa.gov/) and in the
                    official National Oceanic and Atmospheric Administation (NOAA) archive at National Center for
                    Environmental Information (NCEI;
                    https://www.ncei.noaa.gov/access/marine-environmental-buoy-database/). Each data source has
                    their own idiosyncrasies (Hall and Jensen, 2021, http://dx.doi.org/10.21079/11681/40059) that
                    need to be accounted for to accurately use these NBDC data within U.S. Army Corps of Engineers
                    (USACE) Engineers and Research Development Center (ERDC) products. USACE sponsored a Coastal
                    Ocean Data Systems (CODS) National Coastal Wave Climate (NCWC) project that developed in-house
                    USACE quality control checks and metadata corrections to develop a best available measurement
                    archive (herewith called the USACE QCC measurement archive). Of note is that integral wave data
                    are imported directly from the NDBC data sources, and are not corrected for calculation errors
                    that occurred during NDBC processing from spectral wave data. The self-described, USACE QCC
                    measurement archive data is stored in netCDF format alongside the USACE Coastal and Hydraulic
                    Laboratory (CHL) Thredds Wave Information Study (WIS) long-term hindcast, accessible to both
                    the USACE and the public.', prec="char")
677                 ncatt_put(nc=con,varid=0, attname="geospatial_lat_max", attval = "variable: see latitude data",
                    prec="char")
678                 ncatt_put(nc=con,varid=0, attname="geospatial_lat_min", attval = "variable: see latitude data",
                    prec="char")
679                 ncatt_put(nc=con,varid=0, attname="geospatial_lat_units", attval = "degrees", prec="char")
680                 ncatt_put(nc=con,varid=0, attname="geospatial_lon_max", attval = "variable: see longitude
                    data", prec="char")
681                 ncatt_put(nc=con,varid=0, attname="geospatial_lon_min", attval = "variable: see longitude
                    data", prec="char")
682                 ncatt_put(nc=con,varid=0, attname="geospatial_lon_units", attval = "degrees", prec="char")
```

```
683                 ncatt_put(nc=con,varid=0, attname="geospatial_vertial_units", attval = "meters above mean sea
                    level", prec="char")
684                 ncatt_put(nc=con,varid=0, attname="geospatial_vertical_datum", attval =
                    "urn:x-noaa:def:datum:noaa::MSL", prec="char")
685                 ncatt_put(nc=con,varid=0, attname="qc_manual", attval =
                    "https://www.ndbc.noaa.gov/NDBCHandbookofAutomatedDataQualityControl2009.pdf", prec="char" )
686                 ncatt_put(nc=con,varid=0, attname="keywords", attval = "Atmospheric Pressure, Sea level
                    Pressure, Atmospheric Temperature, Surface Temperature, Dewpoint Temperature, Humidity, Surface
                    Winds, Ocean Winds, Ocean Temperature, Sea Surface Temperature, Ocean Waves,  Wave Height, Wave
                    Period, Wave Spectra", prec="char")
687                 ncatt_put(nc=con,varid=0, attname="keywords_vocabulary", attval = "GCMD Science Keywords",
                    prec="char")
688                 ncatt_put(nc=con,varid=0, attname="restrictions", attval = "There are no restrictions placed on
                    these data.", prec="char")
689                 ncatt_put(nc=con,varid=0, attname="scientific_project", attval = "None", prec="char")
690                 ncatt_put(nc=con,varid=0, attname="flag_Conventions", attval = "Paris. Intergovernmental
                    Oceanographic Commission of UNESCO. 2013.Ocean Data Standards, Vol.3: Recommendation for a
                    Quality Flag Scheme for the Exchange of Oceanographic and Marine Meteorological Data. (IOC
                    Manuals and Guides, 54, Vol. 3.) 12 pp. (IOC/2013/MG/54-3). http://dx.doi.org/10.25607/OBP-6",
                    prec="char")
691                 ncatt_put(nc=con,varid=0, attname="flag_descriptions", attval = flag_descrip, prec="char")
692                 ncatt_put(nc=con,varid=0, attname="citation", attval = "The National Data Buoy Center should be
                    cited as the source of these data if used in any publication.", prec="char" )
693                 ncatt_put(nc=con,varid=0, attname="distribution_statement", attval = "There are no restrictions
                    placed on these data.", prec="char")
694                 ncatt_put(nc=con,varid=0, attname="time_coverage_start", attval = dat_start_date, prec="char")
695                 ncatt_put(nc=con,varid=0, attname="time_coverage_end", attval = dat_end_date, prec="char")
696                 ncatt_put(nc=con,varid=0, attname="date_created", attval = as.character(Sys.time()), prec="char")
697                 ncatt_put(nc=con,varid=0, attname="date_created", attval = as.character(Sys.time()), prec="char")
698                 ncatt_put(nc=con,varid=0, attname="processing_level", attval = "0", prec="char")
699                 ncatt_put(nc=con,varid=0, attname="publisher_name", attval = "U.S. Army Corps of Engineers
                    (USACE) Engineers and Research Development Center (ERDC) Coastal Ocean Data Systems (CODS)
                    Program", prec="char")
700                 ncatt_put(nc=con,varid=0, attname="publisher_email", attval = "candice.hall@usace.army.mil",
                    prec="char")
701
702                 if(as.numeric(substr(buoy, start = 1, stop = 2))==45){
703                      if(buoy == "45001"){attval1 = "183 m above mean sea level"}
704                      if(buoy == "45002"){attval1 = "176 m above mean sea level"}
705                      if(buoy == "45003"){attval1 = "177 m above mean sea level"}
706                      if(buoy == "45004"){attval1 = "183 m above mean sea level"}
707                      if(buoy == "45005"){attval1 = "174 m above mean sea level"}
708                      if(buoy == "45006"){attval1 = "183 m above mean sea level"}
709                      if(buoy == "45007"){attval1 = "176 m above mean sea level"}
710                      if(buoy == "45008"){attval1 = "177 m above mean sea level"}
711                      if(buoy == "45010"){attval1 = "177 m above mean sea level"}
712                      if(buoy == "45011"){attval1 = "unknown"}
713                      if(buoy == "45012"){attval1 = "74.7 m above mean sea level"}
714                      ncatt_put(nc=con,varid=0, attname="site_elevation", attval = attval1, prec="char")
715                 }else{ncatt_put(nc=con,varid=0, attname="site_elevation", attval = "sea level", prec="char")}
```

```
                    ncatt_put(nc=con,varid=0, attname="standard_name_vocabulary", attval = "Standard Name Table
                    (current version, v78, 21 September 2021); https://cfconventions.org/standard-names.html",
                    prec="char")

                    #-------------------------------------------------------------------------------------------
                    # closing nc file
                    #-------------------------------------------------------------------------------------------
                    nc_close(con)
                    rm(con,dat,dat_ls, name_var)
                }else{
                    print(paste0("no data for ", dateRange,"_",m))
                    rm(m, ncname, dat_ls)
                }
            } # end month range loop
            rm(list = ls(pattern = paste0("var_")))
            rm(list = ls(pattern = paste0("dat_")))
            rm(list = ls(pattern = paste0("dat_spec_")))

        } # end of yearly date range loop
        print(paste0("finished build thredds on buoy: ", buoy))
        rm(list = ls(pattern = paste0("s_",buoy)))
        rm(buoy, data_ls, date_year_end, date_year_start, dateRange, freq_ls, std_name)
        Sys.time()
    } # end of buoy loop
}
```