



**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

```
1  #-----
2  #-----
3
4  ## master script
5
6  # This script initiates the function scripts within this product suite within the correct order.
7  # This script can run all the buoy stations in serial, or can be copied to run the function scripts
8  # on the HPC in parallel.
9
10 #-----
11 #-----
12 ## install libraries
13 # install.packages("lubridate")
14 # install.packages("R.utils")
15 # install.packages("qpcR")
16 # install.packages("plyr")
17 # install.packages("dplyr")
18 # install.packages("ncdf4")
19 # install.packages("tidyverse")
20 # install.packages("readxl") # dependent on tidyverse
21 # install.packages("stringr")
22 # install.packages("data.table")
23 # install.packages("tidyr")
24 # install.packages("gridExtra")
25 # install.packages("oce")
26 # install.packages("naniar")
27 # install.packages("broom")
28 # install.packages("openair") # polarplots
29 # install.packages("plotly")
30 # install.packages("magrittr")
31 # install.packages("grid")
32 # install.packages("devtools")
33 # install.packages("lsr")
34 # install.packages("RColorBrewer")
35 # install.packages("viridis")
36 # install.packages("colorRamps")
37 # install.packages("ggplot2")
38 # install.packages("ggmap")
39 # install.packages("maps")
40 # install.packages("mapdata")
41
42 ## load libraries (local runs)
43 library(lubridate)
44 library(R.utils)
45 library(qpcR)
46 library(plyr)
47 library(dplyr)
48 library(tibble)
49 # library(ncdf4)
50 library(tidyverse)
```

```
51 library(readxl) # dependent on tidyverse
52 library(stringr)
53 library(data.table)
54 library(tidyr)
55 library(gridExtra)
56 # library(sf)
57 # library(oce)
58 library(naniar)
59 library(broom)
60 library(openair) # polarplots
61 library(plotly)
62 library(magrittr)
63 # library(grid)
64 # library(devtools)
65 library(lsr)
66 library(RColorBrewer)
67 library(viridis)
68 library(colorRamps)
69 library(ggplot2)
70 library(ggmap)
71 library(maps)
72 library(mapdata)
73 library(modeest)
74
75 ## load libraries (HPC runs)
76
77 # library(lubridate, lib="/p/home/candice/Rlibs/")
78 # library(R.utils, lib="/p/home/candice/Rlibs/")
79 # library(qpcR, lib="/p/home/candice/Rlibs/")
80 # library(plyr, lib="/p/home/candice/Rlibs/")
81 # library(dplyr, lib="/p/home/candice/Rlibs/")
82 # library(ncdf4, lib="/p/home/candice/Rlibs/")
83 # library(tidyverse, lib="/p/home/candice/Rlibs/")
84 # library(readxl, lib="/p/home/candice/Rlibs/") # dependent on tidyverse
85 # library(stringr, lib="/p/home/candice/Rlibs/")
86 # library(data.table, lib="/p/home/candice/Rlibs/")
87 # library(tidyr, lib="/p/home/candice/Rlibs/")
88 # library(gridExtra, lib="/p/home/candice/Rlibs/")
89 # library(sf, lib="/p/home/candice/Rlibs/")
90 # library(oce, lib="/p/home/candice/Rlibs/")
91 # library(naniar, lib="/p/home/candice/Rlibs/")
92 # library(broom, lib="/p/home/candice/Rlibs/")
93 # library(openair, lib="/p/home/candice/Rlibs/") # polarplots
94 # library(plotly, lib="/p/home/candice/Rlibs/")
95 # library(magrittr, lib="/p/home/candice/Rlibs/")
96 # library(grid, lib="/p/home/candice/Rlibs/")
97 # library(devtools, lib="/p/home/candice/Rlibs/")
98 # library(lsr, lib="/p/home/candice/Rlibs/")
99 # library(RColorBrewer, lib="/p/home/candice/Rlibs/")
100 # library(viridis, lib="/p/home/candice/Rlibs/")
```

```

101 # library(colorRamps, lib="/p/home/candice/Rlibs/")
102 # library(ggplot2, lib="/p/home/candice/Rlibs/")
103 # library(ggmap, lib="/p/home/candice/Rlibs/")
104 # library(maps, lib="/p/home/candice/Rlibs/")
105 # library(mapdata, lib="/p/home/candice/Rlibs/")
106 # library(modeest, lib="/p/home/candice/Rlibs/")
107
108
109 ## select the following data locations and start date
110 start_date <- 1970
111 drive <- "F:/Candice/"
112 # drive <- "/p/work/candice/"
113 data_dir <- paste0(drive, "projects/WaveTrends/data/")
114 # data_dir <- paste0(drive, "projects/WaveTrends/annual_runs/data/")
115 setwd(data_dir)
116
117 ## set colors for each buoy
118 plot_colors <- viridis(n=6)
119 # plot_colors <- matlab.like2(4)
120 color_ndbc_raw <- plot_colors[1]
121 color_ndbc_orig <- plot_colors[5] # "#440154FF" # purple
122 color_ndbc_recalc <- plot_colors[3] # "#3B528BFF" # "5 = #FDE725FF" # yellow
123 color_chl_calc <- plot_colors[4] # "#21908CFF" # "#21908CFF" # green
124 color_WIS <- "red" # "#5DC863FF"
125
126 ## set colors for wind/wave polar plots
127 # brewer.pal(n = 8, name = "RdBu")
128 colour1 <- viridis(n=4)
129 cols1 <- c(colour1[4], colour1[3], colour1[2], colour1[1])
130 colour2 <- viridis(n=5)
131 cols2 <- c(colour2[5], colour2[4], colour2[3], colour2[2], colour2[1])
132 # cols1 <- c("black", "green", "blue", "red")
133 # plot type, symbol, size
134 type = "l"
135 pch = "."
136 lwd = 0.5
137 cex = 0.5
138
139 ## set plot parameters
140 ## set parameters common to all plots
141 xlab = "Date" # label for x axis
142 width = 2000+2000 # width of exported plot
143 height = 1500+1500 # height of exported plot
144 res = 300 # plot resolution
145 width1 = 1000
146 height1 = 700
147 par1 = c(5, 5, 4, 4)
148 Delta <- '\U0394'
149 degree <- '\U00B0'
150 ## my.grid function formats

```

```

151 my.format <- "%m-%d-%Y" # "%m-%Y" (long datasets) or "%m-%d-%Y" (short datasets)
152 my.period <- "weeks" # "months" (long datasets) or "weeks" (short datasets)
153 ## my grid function for plots
154 my.grid <-function(dataset, my.period = "year", my.format = "%Y"){
155     grid(nx=NA, ny=NULL)
156     abline(v=axis.POSIXct(1, at=seq(min(dataset[1,1]), max(dataset[nrow(dataset),1]),
157                                     by= my.period), format=my.format),
158           col = "lightgray", lty = "dotted", lwd = par("lwd"))
159 }
160 ## function to capitalize string
161 simpleCap <- function(x) {
162     s <- strsplit(x, " ")[[1]]
163     paste(toupper(substring(s, 1,1)), substring(s, 2),
164           sep="", collapse=" ")
165 }
166
167 ## The choice of significance level at which you reject H0 is arbitrary.
168 sig <- 0.01
169
170 ## to run all of the buoy stations in serial, read in the NDBC buoy list as follows:
171 list_ndbc <- read.csv(paste0(data_dir,"NDBC_buoys.csv"),header = TRUE)
172 list_ndbc <- dplyr::filter(list_ndbc, list_ndbc$owner == "NDBC")
173 list_ndbc_buoy <- as.character(list_ndbc$station)
174 buoy_ls <- list_ndbc_buoy
175 rm(list_ndbc_buoy, list_ndbc)
176 print(buoy_ls)
177
178 # ## to run all of the buoy stations in serial, use the following code that assigns the names of this master
179 # ## files to the scripts to isolate a specific buoy number:
180 # buoy_ls <- gsub(".R","",unlist(strsplit(rstudioapi::getSourceEditorContext()$path,"__"))[2])
181 # print(buoy_ls)
182 # print(Sys.time())
183
184 #-----
185 #-----
186
187 ## download ndbc and ncei datasets from ndbc and ncei websites
188 print("starting data download...")
189 start_time <- Sys.time()
190 print(Sys.time())
191 ## Be aware that NDBC changes 'Dxx' extensions in NCEI NetCDF data! The code is currently set to test for 'D'
192 ## extensions between 1-20.
193 for (buoys in buoy_ls){
194     print(buoys)
195     source(paste0(data_dir,"download_data_1.R"))
196     download_data_1(buoys, start_year = start_date, data_dir)
197     rm(download_data_1, read_NetCDF_post_2011)
198 }
199 print(Sys.time())
200 end_time <- Sys.time()

```

```
200 print("finished data download")
201
202 #-----
203
204 ## look for empty folders in each data directory and delete
205
206 ## ndbc unzipped folder
207 # Get vector of all folder names
208 dirlist <- list.dirs(unzip_ndbc_dir)
209 length(dirlist)
210 # Extract vector of empty folder names
211 empty_dir <- dirlist[sapply(dirlist, function(x) length(list.files(x))==0)]
212 length(empty_dir)
213 print(empty_dir)
214 # Remove empty folders
215 unlink(empty_dir, recursive=TRUE, force=FALSE)
216 # Get vector of all file names
217 ff <- dir(unzip_ndbc_dir, recursive=TRUE, full.names=TRUE)
218 length(ff)
219 # Extract vector of empty files' names
220 eff <- ff[file.info(ff)[["size"]]==0]
221 length(eff)
222 print(eff)
223 # Remove empty files
224 unlink(eff, recursive=TRUE, force=FALSE)
225 rm(dirlist,empty_dir,ff, eff)
226
227 ## ndbc zipped folder
228 # Get vector of all folder names
229 dirlist <- list.dirs(zip_ndbc_dir)
230 length(dirlist)
231 # Extract vector of empty folder names
232 empty_dir <- dirlist[sapply(dirlist, function(x) length(list.files(x))==0)]
233 length(empty_dir)
234 print(empty_dir)
235 # Remove empty folders
236 unlink(empty_dir, recursive=TRUE, force=FALSE)
237 # Get vector of all file names
238 ff <- dir(zip_ndbc_dir, recursive=TRUE, full.names=TRUE)
239 length(ff)
240 # Extract vector of empty files' names
241 eff <- ff[file.info(ff)[["size"]]==0]
242 length(eff)
243 print(eff)
244 # Remove empty files
245 unlink(eff, recursive=TRUE, force=FALSE)
246 rm(dirlist,empty_dir,ff, eff)
247
248 ##-----
249
```

```

250 ## ncei netCDF folder
251 # Get vector of all folder names
252 dirlist <- list.dirs(netCDF_ncei_dir)
253 length(dirlist)
254 # Extract vector of empty folder names
255 empty_dir <- dirlist[apply(dirlist, function(x) length(list.files(x))==0)]
256 length(empty_dir)
257 print(empty_dir)
258 # Remove empty folders
259 unlink(empty_dir, recursive=TRUE, force=FALSE)
260 # Get vector of all file names
261 ff <- dir(netCDF_ncei_dir, recursive=TRUE, full.names=TRUE)
262 length(ff)
263 # Extract vector of empty files' names
264 eff <- ff[file.info(ff)[["size"]]==0]
265 length(eff)
266 print(eff)
267 # Remove empty files
268 unlink(eff, recursive=TRUE, force=FALSE)
269 rm(dirlist,empty_dir,ff, eff)
270
271 ## ncei ASCII folder
272 # Get vector of all folder names
273 dirlist <- list.dirs(ascii_ncei_dir)
274 length(dirlist)
275 # Extract vector of empty folder names
276 empty_dir <- dirlist[apply(dirlist, function(x) length(list.files(x))==0)]
277 length(empty_dir)
278 print(empty_dir)
279 # Remove empty folders
280 unlink(empty_dir, recursive=TRUE, force=FALSE)
281 # Get vector of all file names
282 ff <- dir(ascii_ncei_dir, recursive=TRUE, full.names=TRUE)
283 length(ff)
284 # Extract vector of empty files' names
285 eff <- ff[file.info(ff)[["size"]]==0]
286 length(eff)
287 print(eff)
288 # Remove empty files
289 unlink(eff, recursive=TRUE, force=FALSE)
290 rm(dirlist,empty_dir,ff, eff)
291
292 #-----
293 #-----
294
295 ## If this is an annual run, merge the previous and updated metadata sheets for use in concat step.
296 print("starting merge metadata...")
297 start_time <- Sys.time()
298 print(Sys.time())
299 source(paste0(data_dir,"merge_metadata_1a.R"))

```



```

300 merge_metadata_1a(data_dir)
301 rm(merge_metadata_1a)
302 print(Sys.time())
303 end_time <- Sys.time()
304 print("finished merge metadata")
305
306 #-----
307 #-----
308
309 ## concatenate ndbc and ncei datasets from downloaded data
310 print("starting data concat...")
311 start_time <- Sys.time()
312 print(Sys.time())
313 for (buoys in buoy_ls){
314     source(paste0(data_dir,"concat_data_2.R"))
315     concat_data_2(buoys, start_year = start_date, data_dir)
316     rm(concat_data_2,concat_ndbc, concat_ncei)
317     rm_ls = ls(pattern = paste0("s_", buoys))
318     rm(list = rm_ls)
319     rm(rm_ls)
320 }
321 print(Sys.time())
322 end_time <- Sys.time()
323 print("finished data concat")
324
325 #-----
326 #-----
327
328 ## ** update NDBC google spreadsheets downloads **
329 ## Download latest NDBC metadata sheets (download folders)
330
331 ## verify ncei netcdf metadata with NDBC Google spreadsheets
332 print("starting ncei metadata verification...")
333 start_time <- Sys.time()
334 print(Sys.time())
335 for (buoys in buoy_ls){
336     source(paste0(data_dir,"verify_netcdf_3.R"))
337     verify_netcdf_3(buoys, data_dir)
338     rm(verify_netcdf_3)
339 }
340 print(Sys.time())
341 end_time <- Sys.time()
342 print("finished ncei metadata verification")
343
344 #-----
345 #-----
346
347 ## geoclean the ndbc and ncei data
348 print("starting data geoClean...")
349 start_time <- Sys.time()

```

```
350 print(Sys.time())
351 for (buoys in buoy_ls){
352     source(paste0(data_dir,"geoClean_data_4.R"))
353     geoClean_data_4(buoys, data_dir)
354     rm(geoClean_data_4, plot_stdmet, plots_spec)
355 }
356 print(Sys.time())
357 end_time <- Sys.time()
358 print("finished data geoClean")
359
360 #-----
361 #-----
362
363 ## create best available ndbc dataset with ncei metadata
364 print("starting best data...")
365 start_time <- Sys.time()
366 print(Sys.time())
367 for (buoys in buoy_ls){
368     source(paste0(data_dir,"create_best_data_5.R"))
369     create_best_data_5(buoys, data_dir)
370     rm(create_best_data_5)
371 }
372 print(Sys.time())
373 end_time <- Sys.time()
374 print("finished best data")
375
376 #-----
377 #-----
378
379 # build CHL Thredds NDBC netcdf file
380 start_time <- Sys.time()
381 print(Sys.time())
382 for (buoys in buoy_ls){
383     source(paste0(data_dir,"build_thredds_netcdf_6.R"))
384     build_thredds_netcdf_6(buoys, data_dir)
385     rm(build_thredds_netcdf_6)
386 }
387 print(Sys.time())
388 end_time <- Sys.time()
389
390 #-----
391 #-----
392 #-----
393
394
```



US Army Corps  
of Engineers®

download\_data\_1.R

```

1 download_data_1 <- function(buoys = "list of buoys", start_year = "start year", data_dir = "data_dir"){
2
3     ##-----
4     ## script to download NDBC web files and NCEI netcdf files
5     ## Hall 12/28/2019
6     ##-----
7
8     ## Actions:
9     ## 1.   Sets data locations
10    ## 2.   Lists buoy ID's for downloading
11    ## 3.   Finds current end date for monthly and yearly data downloads
12    ## 4.   Downloads NDBC website historical data from the NDBC website
13    ## 5.   Unzips NDBC files
14    ## 6.   Downloads NDBC NetCDF data from the NCEI website
15    ## 7.   Calls 'read_NetCDF_post_2011.R and read_NetCDF_pre_2011.R' to extract NetCDF files to ASCII
16    ## 8.   Attaches datetime, lat and lon to all datasets - including gps/latitude and gps/longitude = repeated
17    ## 9.   Calls 'read_NetCDF_metadata.R' script to extract all netCDF metadata from global and variable
18    ##        attributes
19
20    ##-----
21    ##-----
22
23    # load libraries (local run)
24    library(lubridate)
25    library(R.utils)
26    # library("ncdf4")
27    library(qpcR)
28    library(plyr)
29    library(dplyr)
30    library(ncdf4)
31
32    ## load libraries (HPC run)
33    # library(lubridate, lib="/p/home/candice/Rlibs/")
34    # library(R.utils, lib="/p/home/candice/Rlibs/")
35    # library("ncdf4", lib="/p/home/candice/Rlibs/")
36    # library(qpcR, lib="/p/home/candice/Rlibs/")
37    # library(plyr, lib="/p/home/candice/Rlibs/")
38    # library(dplyr, lib="/p/home/candice/Rlibs/")
39    # library(ncdf4, lib="/p/home/candice/Rlibs/")
40
41    ##-----
42    ## set paths
43    ##-----
44
45    setwd(data_dir)
46    # create new folders for data
47    if (!file.exists(paste0(data_dir,"raw_data/"))) {dir.create((paste0(data_dir,"raw_data/")))}
48    input_dir <- paste0(data_dir,"raw_data/")
49    ## set new output directories for raw and zipped datasets
50    # ndbc

```

```

49   if (!file.exists(paste0(input_dir,"ndbc/"))) {dir.create((paste0(input_dir,"ndbc/")))}
50   ndbc_dir <- paste0(input_dir,"ndbc/")
51   if (!file.exists(paste0(ndbc_dir,"zipped/"))) {dir.create((paste0(ndbc_dir,"zipped/")))}
52   zip_ndbc_dir <- paste0(ndbc_dir,"zipped/")
53   if (!file.exists(paste0(ndbc_dir,"unzipped/"))) {dir.create((paste0(ndbc_dir,"unzipped/")))}
54   unzip_ndbc_dir <- paste0(ndbc_dir,"unzipped/")
55   # ncei
56   if (!file.exists(paste0(input_dir,"ncei/"))) {dir.create((paste0(input_dir,"ncei/")))}
57   ncei_dir <- paste0(input_dir,"ncei/")
58   if (!file.exists(paste0(ncei_dir,"netCDF/"))) {dir.create((paste0(ncei_dir,"netCDF/")))}
59   netCDF_ncei_dir <- paste0(ncei_dir,"netCDF/")
60   if (!file.exists(paste0(ncei_dir,"ascii/"))) {dir.create((paste0(ncei_dir,"ascii/")))}
61   ascii_ncei_dir <- paste0(ncei_dir,"ascii/")
62   if (!file.exists(paste0(ncei_dir,"metadata/"))) {dir.create((paste0(ncei_dir,"metadata/")))}
63   metadata_ncei_dir <- paste0(ncei_dir,"metadata/")
64
65   ##-----
66   ## set buoy stations for downloading (for stand-alone use)
67   ##-----
68
69   # list_ndbc <- read.csv(paste0(data_dir,"NDBC_buoys.csv"),header = TRUE)
70   # list_ndbc <- dplyr::filter(list_ndbc, list_ndbc$owner == "NDBC")
71   # list_ndbc_buoy <- as.character(list_ndbc$station)
72   #
73   # buoys <- list_ndbc_buoy
74   # rm(list_ndbc)
75
76   #-----
77   #-----
78
79   # downloading NDBC website historical data
80
81   #-----
82   ## sample web addresses for all data related to each buoy station
83
84   ## yearly Standard meteorological data:
85   https://www.ndbc.noaa.gov/data/historical/stdmet/44014h1990.txt.gz
86   ## yearly Spectral wave density data:
87   https://www.ndbc.noaa.gov/data/historical/swden/44014w1996.txt.gz
88   ## yearly Spectral wave (alpha1) direction data:
89   https://www.ndbc.noaa.gov/data/historical/swdir/44014d1996.txt.gz
90   ## yearly Spectral wave (alpha2) direction data:
91   https://www.ndbc.noaa.gov/data/historical/swdir2/44014i1998.txt.gz
92   ## yearly Spectral wave (r1) direction data:
93   https://www.ndbc.noaa.gov/data/historical/swr1/44014j1998.txt.gz
94   ## yearly Spectral wave (r2) direction data:
95   https://www.ndbc.noaa.gov/data/historical/swr2/44014k1998.txt.gz
96
97   ## monthly Standard meteorological data:
98   https://www.ndbc.noaa.gov/data/stdmet/Jan/44014l2019.txt.gz

```

```

92     ## monthly Spectral wave density data:
93     https://www.ndbc.noaa.gov/data/swden/Jan/4401412019.txt.gz
94     ## monthly Spectral wave (alpha1) direction data:
95     https://www.ndbc.noaa.gov/data/swdir/Jan/4401412019.txt.gz
96     ## monthly Spectral wave (alpha2) direction data:
97     https://www.ndbc.noaa.gov/data/swdir2/Jan/4401412019.txt.gz
98     ## monthly Spectral wave (r1) direction data:
99     https://www.ndbc.noaa.gov/data/swr1/Jan/4401412019.txt.gz
100    ## monthly Spectral wave (r2) direction data:
101    https://www.ndbc.noaa.gov/data/swr2/Jan/4401412019.txt.gz
102
103    ##-----
104    ## set lists of downloading parameters
105    ##-----
106
107    # end date for monthly and yearly data downloads
108    library(lubridate)
109    if(month(Sys.Date())== 1 | month(Sys.Date())== 2){
110        current_year <- as.numeric(unlist(strsplit(as.character(Sys.Date()),"-"))[1])-1
111    }else{
112        current_year <- as.numeric(unlist(strsplit(as.character(Sys.Date()),"-"))[1])
113    }
114    # yearly files
115    start_year <- as.numeric(start_year)
116    years <- seq(start_year,current_year,1)
117    month <- month.abb[c(1:12)]
118
119    # dataset types (Standard meteorological, Spectral wave density, Spectral wave (alpha1) direction,
120    # Spectral wave (alpha2) direction, Spectral wave (r1) direction, Spectral wave (r2) direction)
121    dataTypes <- c("stdmet", "swden", "swdir", "swdir2", "swr1","swr2")
122    dataType_ab <- c("h", "w", "d", "i","j","k")
123
124    ##-----
125    ## download code
126    ##-----
127
128    # looping through the listed buoy ID's
129    for (buoy in buoys){
130        print(paste0("Starting NDBC download: ",buoy))
131
132        # # start writing to an output file
133        # sink(paste0(data_dir,"0_ndbc_download_buoy_",buoy,"_",Sys.Date(),".txt"))
134        # print(paste0("Starting NDBC download: ",buoy))
135
136        ## create buoy specific download folder
137        if (!file.exists(paste0(zip_ndbc_dir,buoy,"/"))) {dir.create((paste0(zip_ndbc_dir,buoy,"/")))}
138        ## for each dataset type, set the data type
139        for (dataT in dataTypes){
140            # dataT <- dataTypes[1]
141            ab <- dataType_ab[match(dataT,dataTypes)]

```

```

137 ## downloading yearly files
138 ## https://www.ndbc.noaa.gov/data/historical/stdmet/44014h1990.txt.gz
139 ## https://www.ndbc.noaa.gov/data/historical/swden/44014wb1997.txt.gz
140 for (year in years){
141     # year <- years[8]
142     # URL website
143     fileUrl1 <-
144     paste0("https://www.ndbc.noaa.gov/data/historical/",dataT,"/",buoy,ab,year,".txt.gz")
145     fileUrl2 <-
146     paste0("https://www.ndbc.noaa.gov/data/historical/",dataT,"/",buoy,ab,"b",year,".txt.gz")
147     #https://www.ndbc.noaa.gov/data/historical/stdmet/44014h2001.txt.gz
148     print(fileUrl1)
149     # set download location and file name
150     destfile = paste0(zip_ndbc_dir,buoy,"/",buoy,ab,year,"_",dataT,".txt.gz")
151     destfile2 = paste0(zip_ndbc_dir,buoy,"/",buoy,ab,"b",year,"_",dataT,".txt.gz")
152     # Download data from the website
153     if (!file.exists(destfile)) {
154         try(download.file(fileUrl1, mode = "wb", destfile = destfile))
155         print(destfile)
156     } else {print("Data are already downloaded: yearly")}
157     if (!file.exists(destfile2)) {
158         try(download.file(fileUrl2, mode = "wb", destfile = destfile2))
159         print(destfile2)
160     } else {print("Data are already downloaded: yearly")}
161 }
162 library(lubridate)
163 if(month(Sys.Date()) == 1) {
164     year1 <- current_year-1
165     ## downloading monthly files of the current year
166     ## https://www.ndbc.noaa.gov/data/stdmet/Jan/4401412019.txt.gz
167     for(m in month){
168         # m <- month[11]
169         num_mth <- match(m, month.abb)
170         print(m)
171         # URL website
172         fileUrl2 <-
173         paste0("https://www.ndbc.noaa.gov/data/",dataT,"/",m,"/",buoy,num_mth,year1,".txt.gz") # normal monthly NDBC file name
174         fileUrl3 <-
175         paste0("https://www.ndbc.noaa.gov/data/",dataT,"/",m,"/",buoy,"a",year1,".txt.gz") # new name??
176         fileUrl4 <-
177         paste0("https://www.ndbc.noaa.gov/data/",dataT,"/",m,"/",buoy,".txt") #
178         another new name???
179         print(fileUrl2)
180         # set download location and file name
181         # stdmet/Jan/4401412019.txt.gz
182         destfile2 =
183         paste0(zip_ndbc_dir,buoy,"/",buoy,ab,year1,"_",match(m,month.abb),"_",dataT,".t

```

```

xt.gz")
# Download data from the website
if (!file.exists(destfile2)) {
  try(download.file(fileUrl2, mode = "wb", destfile = destfile2))
  print(destfile2)
} else {print("Data are already downloaded: fileUrl2")}
if (!file.exists(destfile2)) {
  try(download.file(fileUrl3, mode = "wb", destfile = destfile2))
  print(destfile2)
} else {print("Data are already downloaded: fileUrl3")}
if (!file.exists(destfile2)) {
  try(download.file(fileUrl4, mode = "wb", destfile = destfile2))
  print(destfile2)
} else {print("Data are already downloaded: fileUrl4")}
}
}else{ year1 <- current_year
## 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")
  # Download data from the website
  if (!file.exists(destfile2)) {
    try(download.file(fileUrl2, mode = "wb", destfile = destfile2))
    print(destfile2)
  } else {print("Data are already downloaded: fileUrl2")}
  if (!file.exists(destfile2)) {
    try(download.file(fileUrl3, mode = "wb", destfile = destfile2))
    print(destfile2)
  } else {print("Data are already downloaded: fileUrl3")}
  if (!file.exists(destfile2)) {
    try(download.file(fileUrl4, mode = "wb", destfile = destfile2))
    print(destfile2)
  }
}
}

```



```

218             } else {print("Data are already downloaded: fileUrl4")}
219
220         }
221     }
222 }
223 print(paste0("Finished NDBC download: ",buoy))
224 # end writing to an output file
225 # sink()
226 # print(paste0("Finished NDBC download: ",buoy))
227 }
228
229 ## Different NDBC formats examples...
230 # https://www.ndbc.noaa.gov/data/stdmet/Oct/44014a2019.txt.gz
231 # https://www.ndbc.noaa.gov/data/stdmet/Nov/44014.txt
232 # https://www.ndbc.noaa.gov/data/stdmet/Oct/44014a2019.txt.gz
233 # https://www.ndbc.noaa.gov/data/swr2/Oct/44014a2019.txt.gz
234
235 ##-----
236 ## copy and unzip datafiles code
237 ##-----
238
239 library(R.utils)
240 # copy files to new folder and unzip
241 for (buoy in buoys){
242     print(paste0("Starting NDBC unzip: ",buoy))
243
244     # copy files to new folder for unzipping
245     list.zip <- list.files(path = paste0(zip_ndbc_dir,buoy,"/"), pattern = ".gz", full.names = TRUE)
246
247     if(length(list.zip)> 0){
248
249         # start writing to an output file
250         # sink(paste0(data_dir,"0_ndbc_unzip_buoy_",buoy,"_",Sys.Date(),".txt"))
251         # print(paste0("Starting NDBC unzip: ",buoy))
252
253         ## create buoy specific unzip folder
254         if (!file.exists(paste0(unzip_ndbc_dir,buoy,"/"))){
255             {dir.create((paste0(unzip_ndbc_dir,buoy,"/")))}
256
257             # copy files to new folder for unzipping
258             list.zip <- list.files(path = paste0(zip_ndbc_dir,buoy,"/"), pattern = ".gz", full.names =
TRUE)
259
260             list.unzipped <- list.files(path = paste0(unzip_ndbc_dir,buoy,"/"), pattern = ".txt",
full.names = FALSE)
261             # copy new files over
262             for (file in list.zip){
263                 file_present <- gsub(".gz","",unlist(strsplit(file,paste0(zip_ndbc_dir,buoy,"/")))[2])
264                 if(file_present %in% list.unzipped){
265                     print("file already exists")
266                 }else{

```

```

265         file.copy(file, paste0(unzip_ndbc_dir,buoy,"/"))
266     }
267 }
268 # unzip files in unzipped folder
269 list.unzip <- list.files(path = paste0(unzip_ndbc_dir,buoy,"/"), pattern = ".gz", full.names
= TRUE)
270 for(i in list.unzip){
271     print(i)
272     tryCatch({
273         gunzip(i, remove = TRUE)
274     }, error = function(e){
275         print("already unzipped")
276         unlink(i)
277     })
278 }
279
280 print(paste0("Finished NDBC upzip: ",buoy))
281 # end writing to an output file
282 # sink()
283 }else{print("no ndbc data to unzip")}
284
285 print(paste0("Finished NDBC unzip: ",buoy))
286
287 }
288
289 #-----
290 #-----
291
292 # downloading NCEI NDBC NetCDF data
293
294 #-----
295
296 # ncei website
297
298 ## sample web addresses for all data related to each buoy station
299
300 ## ftp://ftp.nodc.noaa.gov/pub/f291/200301/44014_200301.Z
301 ## https://data.nodc.noaa.gov/thredds/fileServer/ndbc/cmanwx/2011/01/NDBC_44014_201101_D1_v00.nc
302 ## https://data.nodc.noaa.gov/thredds/fileServer/ndbc/cmanwx/2019/01/NDBC_44014_201901_D5_v00.nc
303
304 ## https://www.ncei.noaa.gov/data/oceans/ndbc/cmanwx/2020/03/NDBC_41002_202003_D7_v00.nc
305 ## https://www.ncei.noaa.gov/data/oceans/ndbc/cmanwx/2020/04/NDBC_41001_202004_D4_v00.nc
306
307 ## NCEI NODC THREDDIS NDBC CMANWX server
308 #https://data.nodc.noaa.gov/thredds/fileServer/ndbc/cmanwx/2006/12/44014_200612.nc
309
310 #-----
311 ## download code
312 #-----
313 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             ##-----
344             # # for pre-2011 F291 file formats (no longer available but retaining code for
345             # # historical value)
346             # if(year <2011){
347             #     # URL website
348             #     # ftp://ftp.nodc.noaa.gov/pub/f291/201001/44014_201001.Z
349             #     fileUrl1 <- paste0("ftp://ftp.nodc.noaa.gov/pub/f291/",year,sprintf("%02d",
350             num_mth),"/",buoy,"_",year,sprintf("%02d", num_mth),".Z") # pre-2011
351             #     print(fileUrl1)
352             #     # set download location and file name
353             #     destfile1 = paste0(netCDF_ncei_dir,buoy,"/",buoy,"_",year,sprintf("%02d",
354             num_mth),".Z")
355             #
356             #     # Download data from the website
357             #     if (!file.exists(destfile1)) {
358             #         try(download.file(fileUrl1, mode = "wb", destfile = destfile1))
359             #         print(destfile1)
360             #     } else {print("Data are already downloaded: fileUrl4")}
361             # }

```

```

359 ##-----
360 ---
361 # # post 2011
362 if(year >= 2011){
363     for(d in 1:20){
364         # print(d)
365         # fileUrl1 <-
366         paste0("https://data.nodc.noaa.gov/thredds/fileServer/ndbc/cmanwx/",year,"/",sprintf("%02d", num_mth),"/NDBC_",buoy,"_",year,sprintf("%02d", num_mth),"_D",d,"_v00.nc") # post 2011
367         fileUrl1 <-
368         paste0("https://www.ncei.noaa.gov/data/oceans/ndbc/cmanwx/",year,"/",sprintf("%02d", num_mth),"/NDBC_",buoy,"_",year,sprintf("%02d", num_mth),"_D",d,"_v00.nc") # post 2011
369         print(fileUrl1)
370         destfile1 =
371         paste0(netCDF_ncei_dir,buoy,"/",buoy,"_",year,sprintf("%02d", num_mth),"_D",d,".nc")
372         print(destfile1)
373         # Download data from the website
374         if (!file.exists(destfile1)) {
375             try(download.file(fileUrl1, mode = "wb", destfile = destfile1))
376             print(destfile1)
377         } else {print("Post 2011 data are already downloaded")}
378     }
379 }
380 # pre 2011
381 if(year < 2011){
382     # fileUrl2 <-
383     paste0("https://data.nodc.noaa.gov/thredds/fileServer/ndbc/cmanwx/",year,"/",sprintf("%02d", num_mth),"/",buoy,"_",year,sprintf("%02d", num_mth),".nc") # post 2011
384     post 2011
385     fileUrl2 <-
386     paste0("https://www.ncei.noaa.gov/data/oceans/ndbc/cmanwx/",year,"/",sprintf("%02d", num_mth),"/",buoy,"_",year,sprintf("%02d", num_mth),".nc") # post 2011
387     destfile2 = paste0(netCDF_ncei_dir,buoy,"/",buoy,"_",year,sprintf("%02d", num_mth),".nc")
388     print(destfile2)
389     if (!file.exists(destfile2)) {
390         try(download.file(fileUrl2, mode = "wb", destfile = destfile2))
391         print(destfile2)
392     } else {print("Pre 2011 data are already downloaded")}
393 }
394 }
395 print(paste0("Finished NCEI download: ",buoy))
396 # # end writing to an output file
397 # sink()
398 # print(paste0("Finished NCEI download: ",buoy))
399 }

```

```

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") #
431                     seconds since 1970-01-01 00:00:00 UTC"
432                     colnames(time) <- c("time")
433                     # save wave spectral frequency bands
434                     wave_wpm_available <-
435                     t(data.frame(c(as.character(time[1,1]),signif(wave_wpm,6)),
436                         stringsAsFactors = FALSE))
437                     library(plyr)
438                     wave_wpm_available_ALL <- qpcR::rbind.na(wave_wpm_available_ALL,
439                         wave_wpm_available)
440                     rownames(wave_wpm_available_ALL) <- NULL
441                     rm(wave_wpm_available)
442                 }, error = function(e){print("no wave_wpm")})

```

```

439         )
440
441         # r data check
442         dat_list <- NULL
443
444         for (i in 1:nc$nvars){
445             j<-nc$var[[i]]$name
446             # print(j)
447             dat_list <- c(dat_list,j)
448         }
449         # find r1 data
450         if(any(grepl("*/r1$", dat_list))){
451             r_num <- grep("*/r1$", dat_list)[1]
452             r_dat <- data.frame(t(ncvar_get(nc, dat_list[r_num])),
453                                stringsAsFactors = FALSE)
454             # human readable time
455             time[,1] <- as.POSIXct(time[,1], origin = "1970-01-01", tz = "uct") #
456             seconds since 1970-01-01 00:00:00 UTC"
457             colnames(time) <- c("time")
458             # save wave spectral frequency bands
459             r_available <- data.frame(c(as.character(time[1,1]), r_dat[1,]),
460                                     stringsAsFactors = FALSE)
461             library(plyr)
462             spec_r_type_ALL <- qpcR::rbind.na(spec_r_type_ALL, r_available)
463             rownames(spec_r_type_ALL) <- NULL
464             rm(r_available)
465         }else{print(paste0("has no R: ", file))}
466         nc_close(nc)
467         rm(nc,time,dat_list, j)
468     }, error = function(e){print(paste0("can't open: ", file))}
469 )
470 }
471 write.csv(wave_wpm_available_ALL, paste0(ascii_ncei_dir,buoy,"_spec_freq_available_ALL.csv"),
472           row.names=FALSE, na = "NaN")
473 rm(wave_wpm_available_ALL)
474 write.csv(spec_r_type_ALL, paste0(ascii_ncei_dir,buoy,"_spec_r_type_ALL.csv"),
475           row.names=FALSE, na = "NaN")
476 rm(spec_r_type_ALL)
477
478 ## 2011 onward NetCDF format
479
480 ## create buoy specific unzip folder
481 if (!file.exists(paste0(ascii_ncei_dir,buoy,"/"))){
482     dir.create((paste0(ascii_ncei_dir,buoy,"/")))
483     print(paste0("working on post-2011: ", file))
484
485     # copy files to new folder for unzipping
486     list.zip <- list.files(path = paste0(netCDF_ncei_dir,buoy,"/"), pattern = "_D", full.names =
487                             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     ## pre-2011 NetCDF format
521
522     # reset directory
523     setwd(data_dir)
524     print(paste0("working on pre-2011: ", file))
525
526

```

```

527 # # copy files to new folder for unzipping
528 # list.zip <- list.files(path = paste0(netCDF_ncei_dir,buoy,"/"), pattern = buoy, full.names
= TRUE)
529 list.netcdf <- list.files(path = paste0(netCDF_ncei_dir,buoy,"/"), full.names = TRUE)
530 list.ascii <- list.files(path = paste0(ascii_ncei_dir,buoy,"/"), full.names = TRUE)
531
532 if (start_year <= 2010){
533     dates <- c(1970:2010)
534     ## find the files for this data setup in netcdf folders
535     files <- list()
536     for (i in 1:length(dates)){
537         files[[i]] <- list.netcdf[grepl(paste0("_",dates[i]),list.netcdf)]
538     }
539     list.netcdf <- unlist(files)
540     ## find the files for this data setup in ascii folders
541     files <- list()
542     for (i in 1:length(dates)){
543         files[[i]] <- list.ascii[grepl(paste0("_",dates[i]),list.ascii)]
544     }
545     list.ascii <- unlist(files)
546     # look for extracted files
547     list.extracted <- NULL
548     for(l in list.ascii){
549         # l <- list.ascii[1]
550         stripped <- unlist(strsplit(l, paste0("/",buoy,"/")))[2]
551         list.extracted <- c(list.extracted,stripped)
552     }
553     # copy new files over
554     for (file in list.netcdf){
555         # file <- list.netcdf[1]
556         file_present <-
557         gsub(".nc","",unlist(strsplit(file,paste0(netCDF_ncei_dir,buoy,"/")))[2])
558         if(file_present %in% list.extracted){
559             print("file already exists")
560         }else{
561             file.copy(file, paste0(ascii_ncei_dir,buoy,"/"))
562             print(paste0("copying file over: ", file))
563         }
564     }
565
566     # Use 'read_NetCDF_pre_2011.R' script and extract NetCDF files to ASCII
567     source(paste0(data_dir,"read_NetCDF_pre_2011.R"))
568     files <- list.files(path = paste0(ascii_ncei_dir,buoy,"/"), pattern = ".nc",
569     full.names = TRUE)
570
571     for(file in files){
572         # file <- files[202]
573         file
574         read_NetCDF_pre_2011(file, buoy, ncei_dir, ascii_ncei_dir)
575     }

```



```

574         # remove files if extracted
575         ext_files <- list.files(path = paste0(ascii_ncei_dir,buoy,"/"), pattern = ".nc",
                                full.names = FALSE)
576         for (e in ext_files){
577             # e <- ext_files[1]
578             e_full <- paste0(ascii_ncei_dir,buoy,"/",e)
579             destfile <- paste0(ascii_ncei_dir,buoy,"/", unlist(strsplit(e, ".nc"))[1])
580             destfile2 <- paste0(ascii_ncei_dir,buoy,"/", unlist(strsplit(e, "_D"))[1])
581             if(file.exists(destfile)){try(file.remove(e_full))}
582             if(file.exists(destfile2)){try(file.remove(e_full))}
583         }
584         print(paste0("finished pre-2011: ", file))
585
586         # buoy <- buoys[1]
587         print(paste0("Finished NCEI read_NetCDF: ",buoy))
588     }
589     # end writing to an output file
590     # sink()
591     print(paste0("Finished NCEI read_NetCDF: ",buoy))
592 }else{print("empty folder - no data")}
593 # # end writing to an output file
594 print(paste0("Finished NCEI read_NetCDF: ",buoy))
595 }
596 rm(r_dat)
597
598 #-----
599 #-----
600 #
601 # Extract all metadata
602 #
603 #-----
604 # buoys with no post-2011 data:
605 # buoys <- c(41006, 41018, 41021, 41022, 41023, 42007, 42038, 42041, 42042, 42053, 42054, 42080,
606 #           44004, 44028, 44070, 45011, 46003, 46023, 46030, 46045, 46051, 46062, 46063, 46079,
607 #           46106, 46107, 48011, 51026, 51028)
608
609 for (buoy in buoys){
610     print(paste0("Starting NCEI metadata dump: ",buoy))
611     time_pause <- 18 # seconds
612
613     # look for data
614     list.zip <- list.files(path = paste0(netCDF_ncei_dir,buoy,"/"), pattern = "_D", full.names = TRUE)
615
616     if(length(list.zip)>0){
617         # # start writing to an output file
618         # sink(paste0(data_dir,"0_ncei_metadata_buoy_",buoy,"_",Sys.Date(),".txt"))
619         # print(paste0("Starting NCEI read_NetCDF: ",buoy))
620
621         # create buoy folder
622         if (!file.exists(paste0(metadata_ncei_dir,buoy,"/")))

```

```

623     {dir.create((paste0(metadata_ncei_dir,buoy,"/")))}
624     source(paste0(data_dir,"read_NetCDF_metadata.R"))
625     read_NetCDF_metadata(buoy, list.zip, netCDF_ncei_dir, metadata_ncei_dir, time_pause)
626
627     # # # end writing to an output file
628     # sink()
629     # print(paste0("Finished metadata: ",buoy))
630 }else{
631     print("no netCDF data available for download")
632     # sink()
633     print(paste0("Finished metadata: ",buoy))
634 }
635 # # preventative
636 # sink()
637 }
638
639
640 #-----
641 #-----
642 # clean glob environ
643 rm(list = ls())
644 #-----
645 #-----
646 }
647
648

```



US Army Corps  
of Engineers®

read\_NetCDF\_pre\_2011.R

```

1 read_NetCDF_pre_2011 <- function(x = "list of files", buoy = "buoy number", ncei_dir = "ncei_dir", ascii_ncei_dir =
  "ascii_ncei_dir"){
2
3     print(x)
4     library("ncdf4")
5     ## create buoy specific extract folder
6     folder_name <- gsub(".nc","",unlist(strsplit(x, paste0("/",buoy,"/")))[2])
7     if (!file.exists(paste0(ascii_ncei_dir,buoy,"/",folder_name)))
8     {dir.create((paste0(ascii_ncei_dir,buoy,"/",folder_name)))}
9     tryCatch({
10         nc <- nc_open(x)                                # Reading the netcdf data
11         # identify variables in .nc file
12         dat_list <- NULL
13         for (i in 1:nc$nvars){
14             j<-nc$var[[i]]$name
15             # print(j)
16             dat_list <- c(dat_list,j)
17         }
18         # extract data
19         lat <- ncvar_get(nc, "lat")
20         lon <- ncvar_get(nc, "lon")
21         time <- data.frame(ncvar_get(nc, "time"), stringsAsFactors = FALSE)
22         # human readable time
23         time[,1] <- as.POSIXct(time[,1],origin = "1970-01-01",tz = "uct") # seconds since 1970-01-01 00:00:00
24         UTC"
25         colnames(time) <- c("time")
26         # export lat / lon and time
27         lat_t <- merge(time,lat)
28         colnames(lat_t) <- c("time", "lat")
29         lon_t <- merge(time,lon)
30         colnames(lon_t) <- c("time", "lon")
31         # concat time, lat, lon
32         library(dplyr)
33         time_position <- full_join(lat_t, lon_t, by = "time")
34         # consistency with post 2011 files
35         lat_t$qc_flag <- NA; lat_t$release_flag <- NA
36         lon_t$qc_flag <- NA; lon_t$release_flag <- NA
37         # export files
38         write.csv(lat_t, paste0(ascii_ncei_dir,buoy,"/",folder_name, "/",folder_name,"_lat.csv"),
39             row.names=FALSE, na = "NaN")
40         write.csv(lon_t, paste0(ascii_ncei_dir,buoy,"/",folder_name, "/",folder_name,"_lon.csv"),
41             row.names=FALSE, na = "NaN")
42         write.csv(time, paste0(ascii_ncei_dir,buoy,"/",folder_name, "/",folder_name,"_time.csv"),
43             row.names=FALSE, na = "NaN")
44         write.csv(time_position, paste0(ascii_ncei_dir,buoy,"/",folder_name,
45             "/",folder_name,"_time_position.csv"), row.names=FALSE, na = "NaN")
46         rm(lat_t, lon_t, time)
47         # extract data through variable names, attached time stamp and export
48         for(j in dat_list[5: length(dat_list)]){
49             print(j)

```

```

44 dat <- data.frame(ncvar_get(nc, j), stringsAsFactors = FALSE)
45 if(dim(dat)[1]==dim(time_position)[1]){
46     dat <- cbind(time_position,dat)
47     colnames(dat) <- c("time","lat",'lon',j)
48     # to match post 2011 dataset
49     dat$qc_flag <- NA; dat$release_flag <- NA
50
51 }else if(dim(dat)[2] == dim(time_position)[1]){ # spectral parameters
52     if(dim(dat)[1] == 2){
53         dat <- dat
54     }else{ dat <- data.frame(t(dat), stringsAsFactors = FALSE)
55         dat <- cbind(time_position,dat)
56         if("wave_wpm_bnds" %in% dat_list){
57             wave_wpm <- ncvar_get(nc, "wave_wpm")
58             wave_col <- c("time","lat",'lon',signif(wave_wpm,4))
59             if(dim(dat)[2]==length(wave_col)){
60                 colnames(dat) <- wave_col
61             }
62         }
63         if("depth_bnds" %in% dat_list){
64             depth <- ncvar_get(nc, "depth")
65             sub_col <- c("time","lat",'lon',depth)
66             if(dim(dat)[2]==length(sub_col)){
67                 colnames(dat) <- sub_col
68             }
69         }
70     }
71 }else{
72     dat <- dat
73 }
74 write.csv(dat, paste0(ascii_ncei_dir,buoy,"/",folder_name, "/",folder_name,"_",j,".csv"),
75 row.names=FALSE, na = "NaN")
76 rm(dat)
77 if("depth_bnds" %in% dat_list){
78     tryCatch({
79         rm(depth)
80     }, error = function(e){
81         print("no depth")
82     })
83 }
84 nc_close(nc)
85 rm(nc, time_position)
86 }, error = function(e){print(paste0("can't open: ",file))}
87 )
88 }
89

```



US Army Corps  
of Engineers®

read\_NetCDF\_post\_2011.R

```

1 read_NetCDF_post_2011 <- function(x = "list of files", buoy = "buoy number", ascii_ncei_dir = "ascii_ncei_dir"){
2     library("ncdf4")
3     library("stringr")
4     ## create buoy specific extract folder
5     folder_name <- gsub(".nc","",unlist(strsplit(x, paste0("/",buoy,"/")))[2])
6     dir.create((paste0(ascii_ncei_dir,buoy,"/",folder_name)))
7
8     tryCatch({
9         nc <- nc_open(x) # Reading the netcdf data
10        print(nc$filename)
11        # extract time data
12        time <- data.frame(ncvar_get(nc, "time"), stringsAsFactors = FALSE)
13        # test for 10 second time field
14        tryCatch({time_10min = data.frame(ncvar_get(nc, "time10"), stringsAsFactors = FALSE)
15        }, error = function(e) {print("no 10-second time field")
16        })
17        # time_10min <- data.frame(ncvar_get(nc, "time10"), stringsAsFactors = FALSE)
18        # human readable time
19        time[,1] <- as.POSIXct(time[,1],origin = "1970-01-01",tz = "uct") # seconds since 1970-01-01 00:00:00
20        UTC"
21        colnames(time) <- c("time")
22        time_list <- ls(pattern = "time")
23        if("time_10min" %in% time_list){
24            time_10min[,1] <- as.POSIXct(time_10min[,1],origin = "1970-01-01",tz = "uct") # seconds since
25            1970-01-01 00:00:00 UTC"
26            colnames(time_10min) <- c("time")
27        }
28        # check for empty NetCDF file before continuing.
29        if(nc$nvars != 0){
30            # Get a list of the nc variable names.
31            dat_list <- attributes(nc$var)$names
32            # lat and lon
33            if("gps_1/latitude" %in% dat_list){
34                lat <- cbind(time,data.frame(ncvar_get(nc, "gps_1/latitude"), stringsAsFactors =
35                FALSE),
36                data.frame(ncvar_get(nc, "gps_1/latitude_qc"), stringsAsFactors = FALSE),
37                data.frame(ncvar_get(nc, "gps_1/latitude_release"), stringsAsFactors =
38                FALSE))
39
40                lon <- cbind(time, data.frame(ncvar_get(nc, "gps_1/longitude"), stringsAsFactors =
41                FALSE),
42                data.frame(ncvar_get(nc, "gps_1/longitude_qc"), stringsAsFactors = FALSE),
43                data.frame(ncvar_get(nc, "gps_1/longitude_release"), stringsAsFactors =
44                FALSE))
45
46                colnames(lat) <- c("time", "lat", "qc_flag", "release_flag")
47                colnames(lon) <- c("time", "lon","qc_flag", "release_flag")
48            }else{
49                lat <- time
50                lat$lat <- NA; lat$qc_flag <- NA; lat$release_flag <- NA
51                lon <- time

```

```

45         lon$lon <- NA; lon$qc_flag <- NA; lon$release_flag <- NA
46     }
47     # concat time, lat, lon
48     library(dplyr)
49     time_position <- full_join(lat[,1:2], lon[,1:2], by = "time")
50     # export files
51     write.csv(lat, paste0(ascii_ncei_dir,buoy,"/",folder_name, "/",folder_name,"_lat.csv"),
52               row.names=FALSE, na = "NaN")
53     write.csv(lon, paste0(ascii_ncei_dir,buoy,"/",folder_name, "/",folder_name,"_lon.csv"),
54               row.names=FALSE, na = "NaN")
55     write.csv(time, paste0(ascii_ncei_dir,buoy,"/",folder_name, "/",folder_name,"_time.csv"),
56               row.names=FALSE, na = "NaN")
57     write.csv(time_position, paste0(ascii_ncei_dir,buoy,"/",folder_name,
58                                     "/",folder_name,"_time_position.csv"), row.names=FALSE, na = "NaN")
59
60     if("time_10min" %in% time_list){
61         time_10min_position <- full_join(time_10min, lat[,1:2], by = "time")
62         time_10min_position <- full_join(time_10min_position, lon[,1:2], by = "time")
63         colnames(time_10min_position) <- c("time", "lat", "lon")
64         # fill in missing lat/lon
65         library(tidyr)
66         time_10min_position <- time_10min_position %>% fill(c(lat,lon),.direction = "downup")
67         write.csv(time_10min, paste0(ascii_ncei_dir,buoy,"/",folder_name,
68                                     "/",folder_name,"_time_10min.csv"), row.names=FALSE, na = "NaN")
69         write.csv(time_10min_position, paste0(ascii_ncei_dir,buoy,"/",folder_name,
70                                                 "/",folder_name,"_time_10min_position.csv"), row.names=FALSE, na = "NaN")
71         rm(time_10min)
72     }
73
74     rm(lat,lon, time)
75     # remove qc and release fields from main dat list
76     dat_use <- Filter(function(x) !any(grepl("_qc", x)), dat_list)
77     dat_use <- Filter(function(x) !any(grepl("_release", x)), dat_use)
78
79     a = 1; g = 1; b = 1; d = 1; w = 1
80     a_ls <- NULL; g_ls <- NULL; b_ls <- NULL; d_ls <- NULL; w_ls <- NULL
81     # extract data through variable names, attached time and position, and export
82     for(j in dat_use){
83         dat <- data.frame(ncvar_get(nc, j), stringsAsFactors = FALSE)
84         # find qc and release code
85         if(paste0(j,"_qc") %in% dat_list){
86             qc <- data.frame(ncvar_get(nc, paste0(j,"_qc")), stringsAsFactors = FALSE)
87         }
88         if(paste0(j,"_release") %in% dat_list){
89             rl <- data.frame(ncvar_get(nc, paste0(j,"_release")), stringsAsFactors = FALSE)
90         }
91         # define column name
92         if(!is.na(unlist(strsplit(j, "/"))[2])){
93             dat_name <- unlist(strsplit(j, "/"))[2]
94         }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 =
122                                     "time")
123             dat <- left_join(dat,time_new, by = "time")
124         }else{
125             dat <- cbind(time_10min_position,dat, qc, rl)
126         }
127         colnames(dat) <- c("time","lat","lon",dat_name,
128                             "qc_flag","release_flag")
129     }
130     if(dim(dat)[1] < dim(time_10min_position)[1] & dim(dat)[1] >
131         dim(time_position)[1]){ # spectral parameters
132         if(j != "time_wpm_40_bnds"){
133             if(dim(qc)[1]==dim(rl)[1]){
134                 dat <- cbind(dat, qc, rl)
135                 colnames(dat) <- c(dat_name, "qc_flag","release_flag")
136             }else{
137                 dat <- cbind(dat, qc)

```

```

135                                     colnames(dat) <- c(dat_name, "qc_flag")
136                                     dat$release_flag <- NA
137                                     }
138                                     }else{
139                                     dat <- dat
140                                     colnames(dat) <- dat_name
141                                     }
142                                     }
143
144 # define export name and export
145 dat_name <- gsub("/", "_", j)
146 find_dat <- unlist(strsplit(dat_name, "_"))[1]
147 payload <- names(unlist(nc$fqgn2Rindex))
148 # print(dat_name)
149 if(find_dat == "gps"){
150     if(dat_name %in% g_ls){g <- sum(str_count(g_ls, paste0(dat_name, "$")))+1}
151     g_ls <- c(g_ls, dat_name)
152     payload_num <- unlist(payload[grepl(find_dat, payload, ignore.case = TRUE)])
153     payload_num <- unlist(strsplit(payload_num[g], paste0("/", find_dat)))[1]
154 }else if(find_dat == "anemometer"){
155     if(dat_name %in% a_ls){a <- sum(str_count(a_ls, paste0(dat_name, "$")))+1}
156     a_ls <- c(a_ls, dat_name)
157     payload_num <- unlist(payload[grepl(find_dat, payload, ignore.case = TRUE)])
158     payload_num <- unlist(strsplit(payload_num[a], paste0("/", find_dat)))[1]
159 }else if(find_dat == "barometer"){
160     if(dat_name %in% b_ls){b <- sum(str_count(b_ls, paste0(dat_name, "$")))+1}
161     b_ls <- c(b_ls, dat_name)
162     payload_num <- unlist(payload[grepl(find_dat, payload, ignore.case = TRUE)])
163     payload_num <- unlist(strsplit(payload_num[b], paste0("/", find_dat)))[1]
164 }else if(find_dat == "air"){
165     if(dat_name %in% d_ls){d <- sum(str_count(d_ls, paste0(dat_name, "$")))+1}
166     d_ls <- c(d_ls, dat_name)
167     payload_num <- unlist(payload[grepl(find_dat, payload, ignore.case = TRUE)])
168     payload_num <- unlist(strsplit(payload_num[d], paste0("/", find_dat)))[1]
169 }else if(find_dat == "wave"){
170     if(dat_name %in% w_ls){w <- sum(str_count(w_ls, paste0(dat_name, "$")))+1}
171     w_ls <- c(w_ls, dat_name)
172     payload_num <- unlist(payload[grepl(find_dat, payload, ignore.case = TRUE)])
173     payload_num <- unlist(strsplit(payload_num[w], paste0("/", find_dat)))[1]
174 }else{
175     payload_num <- unlist(strsplit(payload[grepl(find_dat, payload, ignore.case =
176                                     TRUE)], "/"))[1]
177 }
178 print(paste0(payload_num, "_", dat_name))
179 write.csv(dat, paste0(ascii_ncei_dir, buoy, "/", folder_name,
180 "/", folder_name, "_", payload_num, "_", dat_name, ".csv"), row.names=FALSE, na = "NaN")
181 rm(dat)
182 if(exists("qc")){rm(qc)}
183 if(exists("rl")){rm(rl)}
184 }

```

```
183         nc_close(nc)
184         rm(nc, time_position)
185         if("time_10min" %in% time_list){
186             rm(time_10min_position)
187         }
188     }else{
189         print("No data in NetCDF")
190         if(exists("nc")){nc_close(nc)}
191     }
192 }, error = function(e){print(paste0("can't open: ", file))}
193 )
194 }
195
```



US Army Corps  
of Engineers®

concat\_data\_2.R

```

1 concat_data_2 <- function(buoys = "list of buoys", start_year = "2020", data_dir = "data_dir"){
2
3     ##-----
4     ## concat NDBC web files and NCEI netCDF files
5     ## Hall 12/23/2019
6     ##-----
7
8     ## Actions:
9     ## 1. Sets data locations
10
11     ## To handle the downloaded NDBC website data, this step allows for the management of:
12     ## Differing yearly file formats and spectral frequencies;
13     ## Concatenates multiple date and time columns into one field;
14     ## Removes redundant dates, as well as visibility and tide columns in stdmet data;
15     ## If necessary (not needed for NDBC data but code provision is in place), allocates spectral data
16     ## into appropriate 38 frequencies (old wave sensors), and 47 frequencies (new wave sensors); and
17     ## Converts NDBC r1 and r2 values to their correct values (NDBC stored data are scaled by 100 to
18     ## reduce storage requirements, so data are multiplied by 0.01).
19
20     ## To handle the NCEI data, this step:
21     ## Selects only relevant data for concatenation;
22     ## Concatenates stdmet data to match NDBC website data nomenclature;
23     ## Applies NDBC netCDF QC flags to the extracted data;
24     ## Converts air, water and dew point temperatures from Kelvin to degree Celsius to match NDBC data;
25     ## Removes zero ('0') wind gust values when no wind speed is present (data that weren't flagged in file).
26     ## To handle the erroneous netCDF spectral frequency data, this part of the code steps through each
27     ## row of spectral data and attempt to match the available spectral frequency data to the appropriate
28     ## 38 frequencies (old wave sensors) or 47 frequencies (new wave sensors).
29     ## Removes redundant netCDF data points that are ~5-10 seconds apart.
30
31     #-----
32     #-----
33
34     # load libraries (local run)
35     library(lubridate)
36     library(plyr)
37     library(dplyr)
38     library(data.table)
39     library(naniar)
40
41     ## load libraries (HPC run)
42     # library(lubridate, lib="/p/home/candice/Rlibs/")
43     # library(plyr, lib="/p/home/candice/Rlibs/")
44     # library(dplyr, lib="/p/home/candice/Rlibs/")
45     # library(data.table, lib="/p/home/candice/Rlibs/")
46     # library(naniar, lib="/p/home/candice/Rlibs/")
47
48     ##-----
49     ## set paths
50     ##-----

```

```

51 # drive <- "E:/Candice/"
52 # drive <- "/p/work/candice/"
53 # data_dir <- paste0(drive, "projects/WaveTrends/annual_runs/data/")
54 setwd(data_dir)
55
56 raw_dir <- paste0(data_dir, "raw_data/")
57 setwd(data_dir)
58 # set input directories
59 unzip_ndbc_dir <- paste0(raw_dir, "ndbc/unzipped/")
60 ascii_ncei_dir <- paste0(raw_dir, "ncei/ascii/")
61
62 # set new out_dir
63 if (!file.exists(paste0(data_dir, "concat_data/"))) {dir.create((paste0(data_dir, "concat_data/")))}
64 out_dir <- paste0(data_dir, "concat_data/")
65 # ndbc
66 if (!file.exists(paste0(out_dir, "ndbc/"))) {dir.create((paste0(out_dir, "ndbc/")))}
67 ndbc_dir <- paste0(out_dir, "ndbc/")
68 # ncei
69 if (!file.exists(paste0(out_dir, "ncei/"))) {dir.create((paste0(out_dir, "ncei/")))}
70 ncei_dir <- paste0(out_dir, "ncei/")
71
72 # data types
73 dataTypes <- c("stdmet", "swden", "swdir", "swdir2", "swr1", "swr2")
74 dataType_ab <- c("h", "w", "d", "i", "j", "k")
75 data_types_stdmet <- c("lat", "lon", "wind direction", "wind speed", "wind gust",
76 "significant_wave_height", "dominant_period", "average_period", "mean_wave_direction",
77 "air_pressure", "air_temperature", "sea_surface_temperature", "dew_point_temperature")
78 data_types_spec <- c("c11", "c11m", "alpha1", "alpha2", "r1", "r2", "C12", "C13", "C22", "C33",
79 "Q12", "Q13", "gamma2", "gamma3", "phih", "rhq", "sensor_output") # c11 = spectral
80 energy, c11m = uncorrected spectral energy
81
82 ##-----
83 ## set buoy stations for stand-alone use
84 ##-----
85
86 # list_ndbc <- read.csv(paste0(data_dir, "NDBC_buoys.csv"), header = TRUE)
87 # list_ndbc <- dplyr::filter(list_ndbc, list_ndbc$owner == "NDBC")
88 # list_ndbc_buoy <- as.character(list_ndbc$station)
89 # buoys <- buoy_ls[1]
90 # print(buoys)
91
92 #-----
93
94 ## ndbc web data files
95
96 #-----
97
98 # looping through the listed buoy ID's
99 for (buoy in buoys){

```

```

100 print(paste0("Starting on ndbc... ",buoy))
101
102 #-----
103
104 # selecting files to concatenate, then concatenating
105 file_list <- list.files(path = paste0(unzip_ndbc_dir,buoy), pattern = ".txt", full.names = TRUE)
106
107 if(length(file_list)>0){
108
109     # # start writing to an output file
110     # sink(paste0(data_dir,"1_ndbc_concat_",buoy,"_",Sys.Date(),".txt"))
111     #
112     print(paste0("Starting on ndbc... ",buoy))
113     print(file_list)
114
115     ## create buoy specific concat folder
116     if (!file.exists(paste0(ndbc_dir,buoy,"/"))) {dir.create((paste0(ndbc_dir,buoy,"/")))}
117
118     ## merging yearly and monthly datafiles of each type
119     # run loops
120     for (t in dataType_ab){
121         print(t)
122         files <- file_list[grepl(pattern = paste0("/",buoy,t), file_list)]
123         print(files)
124         if(t == "w"){
125             spec_freq_available_ALL <- data.frame(matrix(NA, nrow = 0, ncol = 48))
126             for(spec_file in files){
127                 con_1 <- file(spec_file,"r")
128                 first_line <- readLines(con_1,n=3)
129                 close(con_1)
130                 dateString <-unlist(strsplit(first_line[[3]]," "))
131                 df <- paste0(dateString[1],"/",dateString[2],"/",dateString[3])
132                 df <- paste0(df, " ", first_line[[1]])
133                 df <- t(data.frame(unlist(strsplit(df, " ")), stringsAsFactors =
FALSE))
134                 rownames(df) <- NULL
135                 library(plyr)
136                 spec_freq_available_ALL <- qpcR::rbind.na(spec_freq_available_ALL, df)
137             }
138             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)
139         }
140         if(t == "j"){
141             spec_r_type_ALL <- data.frame(matrix(NA, nrow = 0, ncol = 48))
142             for(spec_file in files){
143                 con_1 <- file(spec_file,"r")
144                 first_line <- readLines(con_1, n = 2)
145                 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,

```



```

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

```

```

199     print(paste0("empty sea_surface_temperature column added to NDBC stdmet for
    ", buoy)))
    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))

```

```

238 dataset <- get(d)
239 if(dim(dataset)[1] != 0){
240   # save and export dataset
241   year_range1 <- paste0(year(min(dataset$DateTime, na.rm
= TRUE)), "_", year(max(dataset$DateTime, na.rm =
TRUE)))
242   records1 <- nrow(dataset)
243   if(unlist(strsplit(d, "_"))[3] != "new" &
unlist(strsplit(d, "_"))[3] != "old"){
244     count_cols <- ncol(dataset)
245     data_name <-
paste0("s_", buoy, "_ndbc", file_name, "_freq_", cou
nt_cols, "cols")
246
247   }else{
248     data_name <-
paste0("s_", buoy, "_ndbc", file_name, "_freq_", unl
ist(strsplit(d, "_"))[3])
249
250   }
251   # write.csv(dataset,
paste0(unlist(strsplit(concat_ind_dir, "s_"))[1], data_na
me, "_", year_range1, "_", records1, "_records.csv"),
row.names=FALSE)
252   # saveRDS(dataset, file =
paste0(unlist(strsplit(concat_ind_dir, "s_"))[1], data_na
me, "_", year_range1, ".rds"))
253   rm(year_range1, records1)
254   }
255   rm(dataset)
256   }
257   rm(d, data_list)
258 }else{print(paste0("no ", t, " data for ", buoy))}
259
260
261 #-----
262 # reformat odd data
263
264 #-----
265 data_list <- ls(pattern = "dataset_")
266 data_list <- data_list[data_list %in%
c("dataset_spec_new", "dataset_spec_old") == FALSE]
267 print(data_list)
268
269 if(length(data_list)>0){
270   for(d in data_list){
271     df <- get(d)
272     print(d)

```

```

271         if(dim(df)[1] != 0){
272             # remove rows with no dates
273             df <- df[!is.na(df$DateTime), ]
274             # remove blank data rows
275             df <- df[rowSums(is.na(df)) != ncol(df)-1,]
276             if(dim(df)[1] > 0){
277                 # remove columns with zero or NA
278                 df <- df[, colSums(df != 0, na.rm = TRUE) > 0]
279                 # reorder columns numerically
280                 library(data.table)
281                 df <- setcolorder(df, c(1,
282                                     order(as.numeric(names(df)[-1])) + 1))
283                 # ordering the df by date and selecting
284                 # unique values only
285                 df <- df[order(df$DateTime),]
286                 df <- unique(df)
287                 # rename the rows to reflect unique data
288                 row.names(df) <- 1:nrow(df)
289                 # assign data
290                 assign(d,df)
291             }else{
292                 rm(list = ls()[grepl(d, ls())])
293                 print("dataset removed: all row NA or zero")
294             }
295         }
296         rm(df)
297     }
298     }else{print("no unusual frequency data A")}
299
300     #-----
301     # testing data merge with old and new frequency dfs
302     #-----
303
304     data_list <- ls(pattern = "dataset_")
305     data_list <- data_list[data_list %in%
306     c("dataset_spec_new", "dataset_spec_old") == FALSE]
307     print(data_list)
308
309     if(length(data_list)>0){
310
311         # dealing with multiple formats for the same frequencies
312         if("dataset_spec_old" %in% data_avail){
313             print("dataset_spec_old already exists")
314
315         }else{
316             colNames_spec_old <- c("DateTime",
317                                   "0.0100", "0.0200", "0.0300", "0.0400", "0.0500", "0.0600", "0.0700",

```

```

313         "0.0800", "0.0900", "0.1000", "0.1100", "0.1200",
314         "0.1300", "0.1400", "0.1500", "0.1600", "0.1700", "0.1800", "0.1900", "0.2000", "0.2100",
315         "0.2200",
316         "0.2300", "0.2400", "0.2500", "0.2600", "0.2700", "0.2800", "0.2900", "0.3000", "0.3100",
317         "0.3200",
318         "0.3300", "0.3400", "0.3500", "0.3600", "0.3700", "0.3800", "0.3900", "0.4000")
319     dataset_spec_old <- data.frame(matrix(NA, nrow = 1, ncol =
320     length(colNames_spec_old)))
321     colnames(dataset_spec_old) <- colNames_spec_old
322     dataset_spec_old$DateTime <-
323     lubridate::ymd_hms(dataset_spec_old$DateTime)
324 }
325
326 if("dataset_spec_new" %in% data_avail){
327     print("dataset_spec_new already exists")
328 }else{
329     colNames_spec_new <- c("DateTime", "0.0200", "0.0325",
330     "0.0375", "0.0425", "0.0475", "0.0525",
331     "0.0575", "0.0625", "0.0675",
332     "0.0725", "0.0775", "0.0825",
333     "0.0875", "0.0925", "0.1000", "0.1100",
334     "0.1200", "0.1300", "0.1400",
335     "0.1500", "0.1600", "0.1700",
336     "0.1800", "0.1900", "0.2000", "0.2100",
337     "0.2200", "0.2300", "0.2400",
338     "0.2500", "0.2600", "0.2700",
339     "0.2800", "0.2900", "0.3000", "0.3100",
340     "0.3200", "0.3300", "0.3400",
341     "0.3500", "0.3650", "0.3850",
342     "0.4050", "0.4250", "0.4450", "0.4650",
343     "0.4850")
344     dataset_spec_new <- data.frame(matrix(NA, nrow = 1, ncol =
345     length(colNames_spec_new)))
346     colnames(dataset_spec_new) <- colNames_spec_new
347     dataset_spec_new$DateTime <-
348     lubridate::ymd_hms(dataset_spec_new$DateTime)
349 }
350
351 # loop through odd frequencies to test if df can be merged with new
352 # and odd frequencies
353 # testing data merge
354 data_list <- ls(pattern = "dataset_")
355 data_list <- data_list[data_list %in%
356 c("dataset_spec_new", "dataset_spec_old") == FALSE]

```

```

339     print(data_list)
340
341     for (l in data_list){
342         print(l)
343         dat <- get(l)
344         spec_list <- ls(pattern = "dataset_spec_")
345         spec_list <- spec_list[spec_list %in%
346             c("dataset_spec_new", "dataset_spec_old") == TRUE]
347         print(spec_list)
348
349         for(spec in spec_list){
350             if(exists("dat")){
351                 spec_dat <- get(spec)
352                 ndbc_freq <- names(spec_dat)
353                 dat_names <- names(dat)
354                 setdiff(ndbc_freq, dat_names)
355                 print(unique(dat_names %in% ndbc_freq))
356                 # looping through data
357                 if(any(unique(dat_names %in%
358                     ndbc_freq)==FALSE)){
359                     print(paste0(spec, " didn't match"))
360                 }else{
361                     tryCatch({
362                         library(plyr)
363                         spec_dat <-
364                             rbind.fill(spec_dat, dat)
365                         print(paste0("concat ", spec,
366                             " and ", l, " match"))
367                         assign(spec, spec_dat)
368                         rm(dat, spec_dat)
369                         rm(list = ls()[grepl(l, ls())])
370                     }, error = function(e) {
371                         print("dataset doesn't match")
372                     })
373                     if(exists("spec_dat")){rm(spec_dat)}
374                 }
375                 rm(spec_dat, ndbc_freq, dat_names)
376             }
377             if(exists("dat")){
378                 assign(l, dat)
379                 rm(dat)
380             }
381         }else{print("no unusual frequency data B")}
382
383     #-----
384     -----

```

```

383 # if still present, loop through odd frequencies to attempt a merge by row
384
385 #-----
386
387 data_list <- ls(pattern = "dataset_")
388 data_list <- data_list[data_list %in%
389 c("dataset_spec_new", "dataset_spec_old") == FALSE]
390 print(data_list)
391
392 if(length(data_list)>0){
393   # loop through odd frequencies to attempt a merge by row
394   # testing data merge
395   data_list <- ls(pattern = "dataset_")
396   if("dataset_spec_new" %in% data_list){df_freq_new <-
397     dataset_spec_new; rm(dataset_spec_new)}
398   if("dataset_spec_old" %in% data_list){df_freq_old <-
399     dataset_spec_old; rm(dataset_spec_old)}
400   data_list <- data_list[data_list %in%
401     c("dataset_spec_new", "dataset_spec_old") == FALSE]
402   print(data_list)
403
404   # try per row
405   for (l in data_list){
406     print(l)
407     dat <- get(l)
408     # loop through each row and try to match to new and old
409     frequencies
410     for(i in 1:nrow(dat)) {
411       print(i)
412       # Extract row and all columns
413       dat_row <- dat[i, ]
414       # remove cells with zero or NA - selects ANY NA and
415       removes column
416       library(dplyr)
417       dat_row <- dat_row %>% select_if(~ !any(is.na(.)))
418       # match data rows
419       spec_list <- ls(pattern = "df_freq_")
420       print(spec_list)
421
422       for(spec in spec_list){
423         if(exists("dat_row")){
424           spec_dat <- get(spec)
425           ndbc_freq <- names(spec_dat)
426           dat_names <- names(dat_row)
427           setdiff(ndbc_freq, dat_names)
428           print(unique(dat_names %in% ndbc_freq))
429           # looping through data
430           if(any(unique(dat_names %in%
431             ndbc_freq)==FALSE)) {
432             print(paste0(spec, " didn't

```

```

424                                     match"))
425                                     }else{
426                                         tryCatch({
427                                             library(plyr)
428                                             spec_dat <-
429                                                 rbind.fill(spec_dat,dat
430                                                 _row)
431                                             print(paste0("concat
432                                                 ",spec, " and row
433                                                 ",i," match"))
434                                             assign(spec,spec_dat)
435                                             rm(dat_row,spec_dat)
436                                         }, error = function(e) {
437                                             print("dataset
438                                                 doesn't match")
439                                         })
440                                     if(exists("spec_dat")){rm(spec_
441                                     dat)}
442                                     }
443                                     }
444                                     }
445                                     if(exists("dat_row")){
446                                         count <- length(spec_list)+1
447                                         df_freq <- data.frame(matrix(NA, nrow = 0,
448                                         ncol = dim(dat_row)[2]))
449                                         df_freq<-rbind(df_freq, dat_row)
450                                         new_name <- paste0("df_freq_",count)
451                                         assign(new_name, df_freq)
452                                         print(paste0("data added to NEW DF::
453                                         ",new_name))
454                                         rm(dat_row, df_freq,new_name)
455                                     }
456                                     }# end of row loop
457                                     rm(dat,i,count)
458                                     rm(list = ls()[grepl(1, ls())])
459                                     }
460                                     }else{print("no usual frequency data C")}
461
462                                     #-----
463                                     -----
464                                     # renaming frequency datasets
465                                     #-----
466                                     -----
467                                     data_list <- ls(pattern = "df_freq_")
468                                     if("df_freq_new" %in% data_list){dataset_spec_new <- df_freq_new;

```



```

460   rm(df_freq_new)}
461   if("df_freq_old" %in% data_list){dataset_spec_old <- df_freq_old;
462   rm(df_freq_old)}
463   data_list <- ls(pattern = "df_freq_")
464   if(length(data_list)>0){
465     for(d in data_list){
466       dat_name <- gsub("df_freq_", "dataset_spec_", d)
467       print(dat_name)
468       assign(dat_name, get(d))
469       rm(list = ls()[grepl(d, ls())])
470       rm(d)
471     }
472   }else{print("no usual frequency data D")}

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

#-----

475   data_list <- ls(pattern = "dataset_")
476   data_list <- data_list[data_list %in%
477   c("dataset_spec_new", "dataset_spec_old") == FALSE]
478   print(data_list)
479   # n <- 0
480   # if there are any odd data left
481   if(length(data_list)>0){
482     # test loop three times
483     for (n in 1:3){
484       print(n)
485       data_list <- ls(pattern = "dataset_")
486       data_list <- data_list[data_list %in%
487       c("dataset_spec_new", "dataset_spec_old") == FALSE]
488       print(data_list)
489       if(length(data_list)>1){
490         df_count <- data.frame(matrix(NA, nrow = 0, ncol = 2))
491         colnames(df_count) <- c("Name", "Count")
492         # calculate and save column dimensions
493         for(c in data_list){
494           df <- data.frame(matrix(NA, nrow = 0, ncol =
495           2))
496           colnames(df) <- c("Name", "Count")
497           df[1,] <- c(as.character(c), dim(get(c))[2])
498           df_count <- rbind(df_count, df)
499           rm(df)
500         }
501         # find the df with the most columns
502         df_longest <- df_count[match(max(df_count$Count,
503         na.rm = TRUE), df_count$Count), 1]

```

```

500 print(df_longest)
501
502 # set main with which to merge shorter datasets
503 df_orig <- get(df_longest)
504 rm(list = ls()[grepl(df_longest, ls())])
505
506 # search for remaining odd frequency datasets
507 data_list <- ls(pattern = "dataset_")
508 data_list <- data_list[data_list %in%
509   c("dataset_spec_new", "dataset_spec_old") == FALSE]
510 print(data_list)
511
512 for (l in data_list){
513   print(l)
514   dat <- get(l)
515   if(exists("dat")){
516     ndbc_freq <- names(df_orig)
517     dat_names <- names(dat)
518     setdiff(ndbc_freq, dat_names)
519     print(unique(dat_names %in% ndbc_freq))
520     # looping through data
521     if(any(unique(dat_names %in%
522       ndbc_freq)==FALSE)){
523       print(paste0(spec, " didn't
524         match"))
525     }else{
526       tryCatch({
527         library(plyr)
528         df_orig <-
529         rbind.fill(df_orig, dat)
530         print(paste0("concat
531           ", df_longest, " and
532           ", l, " match"))
533         dat_name <-
534         paste0("df_freq_",
535           unlist(strsplit(df_long
536             est, "_"))[3])
537
538         assign(dat_name, df_orig
539           )
540         rm(dat)
541         rm(list =
542           ls()[grepl(l, ls())])
543       }, error = function(e) {
544         print("dataset
545           doesn't match")
546       })
547     }
548     rm(ndbc_freq, dat_names)
549   }
550 }

```

```

537                                     if(exists("dat")){
538                                         assign(l,dat)
539                                         rm(dat)
540                                     }
541                                 }
542                                 rm(df_count, df_orig)
543
544                                 }else{print("only one odd frequency data left")}
545                            }
546                    }else{print("no remaining odd frequency data E")}}
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576

```

```

#-----
# 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()[grep1(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

```

```

577 # to the archive centers. The units are hundredths,
578 # so the R1 and R2 values in those
579 # files should be multiplied by 0.01. ref:
580 # https://www.ndbc.noaa.gov/measdes.shtml"
581 if(t == "j" | t == "k"){
582     dataset[,2:ncol(dataset)] <-
583     dataset[,2:ncol(dataset)]*0.01
584 }
585 # remove empty rows - accounting for datasets with no
586 # lat/lon
587 dataset <- dataset[rowSums(is.na(dataset)) !=
588 ncol(dataset)-1,]
589 column_names <- names(dataset)
590 # remove columns with zero or NA in old freq
591 if("0.0100" %in% column_names){
592     if(sum(dataset$`0.0100`, na.rm =
593 TRUE)==0){dataset$`0.0100` <- NULL}
594     if(sum(dataset$`0.0200`, na.rm =
595 TRUE)==0){dataset$`0.0200` <- NULL}
596     print("removing 0.0100 and 0.0200 frequency
597 from dataset")
598 }else{print("no 0.0100 frequency in dataset")}
599 # ordering the dataset by date and selecting unique
600 # values only
601 dataset <- dataset[order(dataset$DateTime),]
602 dataset <- unique(dataset)
603 # rename the rows to reflect unique data
604 row.names(dataset) <- 1:nrow(dataset)
605 # save and export dataset
606 year_range <- paste0(year(min(dataset$DateTime, na.rm
607 = TRUE)), "_", year(max(dataset$DateTime, na.rm =
608 TRUE)))
609 records <- nrow(dataset)
610 if(unlist(strsplit(d, "_"))[3] != "new" &
611 unlist(strsplit(d, "_"))[3] != "old"){
612     count_cols <- ncol(dataset)-3
613     data_name <-
614     paste0("s_", buoy, "_ndbc", file_name, "_freq_", cou
615 nt_cols, "cols")
616 }else{
617     data_name <-
618     paste0("s_", buoy, "_ndbc", file_name, "_freq_", unl
619 ist(strsplit(d, "_"))[3])
620 }
621 # export
622 write.csv(dataset,
623 paste0(ndbc_dir, buoy, "/", data_name, "_", year_range, "_", r
624 ecords, "_records.csv"), row.names=FALSE)
625 # saveRDS(dataset, file =
626 paste0(ndbc_dir, buoy, "/", data_name, "_", year_range, ".rds

```

```

608                                     ")))
609                                     # assign(data_name, dataset, envir=globalenv())
610                                     assign(data_name, dataset)
611                                     rm(year_range, records)
612                                     rm(dataset)
613                                     }else{
614                                     rm(dataset)
615                                     rm(list = ls()[grep1(d, ls())])
616                                     }
617                                     }
618                                     }else{print("no data")}
619                                     # housekeeping
620                                     rm_ls <- ls(pattern = "dataset_")
621                                     rm(list = rm_ls)
622                                     rm(rm_ls,data_name,d,data_list)
623
624                                     } # end of individual dataType_ab loop
625
626                                     } # end of dataType_ab loop
627
628                                     # Save multiple objects
629                                     datasets <- ls(pattern = paste0(buoy, "_ndbc"))
630                                     print(datasets)
631                                     save(list = datasets, file = paste0(ndbc_dir, buoy, "/s_",buoy,"_ndbc_ALL.RData"))
632                                     # housekeeping
633                                     print(paste0("Completed data concat for ",buoy))
634                                     # rm(list = ls())
635                                     rm_ls <- ls(pattern = buoy)
636                                     rm(list = rm_ls)
637                                     rm(d, data_list,data_name, file_name, first_line, records, rm_ls, count_cols, t, datasets)
638
639                                     print(paste0("Finished ndbc... ",buoy))
640
641                                     # # Stop writing to the file
642                                     # sink()
643
644                                     }else{print("no new ndbc data for this buoy")}
645
646                                     #-----
647
648                                     print(paste0("Finished ndbc... ",buoy))
649                                     }
650
651                                     #-----
652
653                                     ## ncei ndbc netCDF data files
654
655                                     #-----
656

```

```

657 # selecting files to concatenate, then concatenating
658 for(buoy in buoys){
659     print(paste0("Starting on ncei... ",buoy))
660
661     #-----
662
663     # selecting files to concatenate, then concatenating
664     # find all files
665     file_list <- list.files(path = paste0(ascii_ncei_dir,buoy), pattern = ".csv", full.names = TRUE,
666                             recursive = TRUE)
667
668     # only execute if buoy data available
669     if(length(file_list)>0){
670
671         # # start writing to an output file
672         # sink(paste0(data_dir,"1_ncei_concat_",buoy,"_",Sys.Date(),".txt"))
673
674         print(paste0("Starting on ncei... ",buoy))
675         print(file_list)
676
677         ## create buoy specific concat folder
678         if (!file.exists(paste0(ncei_dir,buoy,"/"))) {dir.create((paste0(ncei_dir,buoy,"/")))}
679
680         #-----
681         # met data
682         #-----
683
684         print(paste0("Starting stdmet data concat for ",buoy))
685
686         source(paste0(data_dir,"concat_ncei.R"))
687         print("initializing concat_ncei for stdmet...")
688         concat_ncei(file_list, start_year, "stdmet", drive, buoy, file_list)
689         print("finished concat_ncei for stdmet...")
690
691         library(dplyr)
692         # remove any duplicate rows
693         dataset <- unique(dataset)
694         # rename the rows to reflect unique data
695         row.names(dataset) <- 1:nrow(dataset)
696
697         # checking for different data nomenclature
698         df_names <- names(dataset)
699         if(length(grep("dominant_period",df_names, value = TRUE))>0){colnames(dataset) =
700             gsub("dominant_period", "dominant_wave_period", colnames(dataset))}
701         if(length(grep("average_period",df_names, value = TRUE))>0){colnames(dataset) =
702             gsub("average_period", "average_wave_period", colnames(dataset))}
703         if(length(grep("_pressure",df_names, value = TRUE))>0){colnames(dataset) = gsub("_pressure",
704             "_pressure_at_sea_level", colnames(dataset))}
705         # if("air_pressure_2" %in% names(dataset)){dataset <- dplyr::rename(dataset,
706             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]) &
726                     !is.na(dataset[p+1,c])){dataset[p,c] <- dataset[p-1,c]}},
727                     error = function(cond){print("no lat/lon inserted")})
728                 tryCatch({if(!is.na(dataset[p-1,c]) &
729                     !is.na(dataset[p+2,c])){dataset[p,c] <- dataset[p-1,c]}},
730                     error = function(cond){print("no lat/lon inserted")})
731                 tryCatch({if(!is.na(dataset[p-2,c]) &
732                     !is.na(dataset[p+2,c])){dataset[p,c] <- dataset[p-2,c]}},
733                     error = function(cond){print("no lat/lon inserted")})
734             }
735             find_NA <- which(is.na(dataset[,c]))
736             NA_count <- length(find_NA)
737             print(paste0("Post count of NA: ",NA_count, ", run: ",loops))
738         }
739     }else{
740         dataset <- left_join(dataset, gps_positions, by = "DateTime")
741         print(paste0("adding gps positions to ",d))
742     }
743
744     # reset order of df
745     df_names <- names(dataset)
746     dataCols <- c("DateTime","lat","lat_metadata","lon","lon_metadata")
747     dataCols <- c(dataCols,grep("wind_direction",df_names, value = TRUE))
748     dataCols <- c(dataCols,grep("wind_speed",df_names, value = TRUE))
749     dataCols <- c(dataCols,grep("wind_gust",df_names, value = TRUE))
750     dataCols <- c(dataCols,grep("significant_wave_height",df_names, value = TRUE))
751     dataCols <- c(dataCols,grep("dominant_wave_period",df_names, value = TRUE))
752     dataCols <- c(dataCols,grep("average_wave_period",df_names, value = TRUE))
753     dataCols <- c(dataCols,grep("mean_wave_direction",df_names, value = TRUE))
754     dataCols <- c(dataCols,grep("air_pressure_at_sea_level",df_names, value = TRUE))
755     dataCols <- c(dataCols,grep("air_temperature",df_names, value = TRUE))
756     dataCols <- c(dataCols,grep("sea_surface_temperature",df_names, value = TRUE))
757     dataCols <- c(dataCols,grep("dew_point_temperature",df_names, value = TRUE))
758     dataCols <- c(dataCols,grep("air_pressure_at_sea_level",df_names, value = TRUE))
759     # reorder columns
760     dataset <- dplyr::select(dataset, all_of(dataCols))
761     rm(dataCols)
762
763     # save and export
764     year_range <- paste0(year(min(dataset$DateTime,na.rm = TRUE)),"_", year(max(dataset$DateTime,
765     na.rm = TRUE)))

```



```

761 write.csv(dataset, paste0(ncei_dir,buoy,"/s_",buoy, "_ncei_h_stdmet_",year_range,".csv"),
762           row.names=FALSE)
763 # saveRDS(dataset, file = paste0(ncei_dir,buoy,"/s_",buoy,"_ncei_stdmet_",year_range,".rds"))
764 data_name <- paste0("s_",buoy,"_ncei_stdmet")
765 assign(data_name, dataset)
766
767 gps_positions <- dplyr::select(dataset, "DateTime", "lat", "lon")
768 rm(dataset, year_range)
769 print(paste0("Completed stdmet data concat for ",buoy))
770
771 #-----
772 # spectral data
773 #-----
774 print(paste0("Starting spectral data concat for ",buoy))
775
776 for (t in data_types_spec){
777   print(t)
778   # run concat_ncei function code
779   source(paste0(data_dir,"concat_ncei.R"))
780
781   print("initializing concat_ncei for spec...")
782   concat_ncei(file_list, start_year, "spec", drive, buoy, file_list, t)
783   print("finishing concat_ncei for spec...")
784
785   data_avail <- ls(pattern = "dataset")
786
787   if(length(data_avail)>0){
788     if(t == "sensor_output"){
789       # ordering the dataset by date and selecting unique values only
790       dataset <- dataset[order(dataset$DateTime),]
791       dataset <- unique(dataset)
792       # rename the rows to reflect unique data
793       row.names(dataset) <- 1:nrow(dataset)
794       # save and export dataset
795       year_range <- paste0(year(min(dataset$DateTime,na.rm = TRUE)), "_",
796                             year(max(dataset$DateTime, na.rm = TRUE)))
797       records <- nrow(dataset)
798       data_name <- paste0("s_",buoy,"_ncei_",t)
799       print("exporting dataset")
800       # write.csv(dataset,
801         paste0(ncei_dir,buoy,"/",data_name,"_",year_range,"_",records,"_records
802               .csv"), row.names=FALSE)
803       # saveRDS(dataset, file =
804         paste0(ncei_dir,buoy,"/",data_name,"_",year_range,".rds"))
805       assign(data_name, dataset)
806       rm(dataset, year_range)
807     }else{
808       # formatting the data
809       data_list <- ls(pattern = "dataset_")
810       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 #-----
829 # reformat odd data
830
831 #-----
832 data_list <- ls(pattern = "dataset_")
833 data_list <- data_list[data_list %in%
c("dataset_spec_new","dataset_spec_old") == FALSE]
834 print(data_list)
835
836 if(length(data_list)>0){
837     for(d in data_list){
838         df <- get(d)
839         print(d)
840         if(dim(df)[1] != 0){

```

```

839 # remove rows with no dates
840 df <- df[!is.na(df$DateTime), ]
841 # remove blank data rows
842 df <- df[rowSums(is.na(df)) != ncol(df)-3,]
843 if(dim(df)[2]>3){ # to handle two column data
844     df$count <- rowSums(df[4:ncol(df)],
845                         na.rm = TRUE)
846     df <- dplyr::filter(df, count != 0)
847     df$count <- NULL
848 }
849 if(dim(df)[1] > 0){
850     # format data
851     df1 <- dplyr::select(df, DateTime,
852                         lat,lon)
853     df2 <- df
854     df2$lat <- NULL; df2$lon <- NULL
855     # remove columns with zero or NA
856     df2 <- df2[, colSums(df2 != 0, na.rm
857                         = TRUE) > 0]
858     # reorder columns numerically
859     library(data.table)
860     df2 <- setcolorder(df2, c(1,
861                             order(as.numeric(names(df2)[-1])) + 1))
862     df <- full_join(df1,df2, by =
863                     "DateTime")
864     rm(df1,df2)
865     # ordering the df by date and
866     # selecting unique values only
867     df <- df[order(df$DateTime),]
868     df <- unique(df)
869     # rename the rows to reflect unique
870     # data
871     row.names(df) <- 1:nrow(df)
872     # assign data
873     assign(d,df)
874 }else{
875     rm(list = ls()[grepl(d, ls())])
876     print("dataset removed: all row NA or
877         zero")
878 }
879 }
880 rm(df)
881 }
882 }else{print("no unusual data A")}
883
884 #-----
885 -----
886 # testing data merge with old and new frequency dfs

```

```

#-----
-----
879 data_list <- ls(pattern = "dataset_")
880 data_list <- data_list[data_list %in%
c("dataset_spec_new", "dataset_spec_old") == FALSE]
881 print(data_list)
882
883 if(length(data_list)>0){
884
885     # dealing with multiple formats for the same frequencies
886     if("dataset_spec_old" %in% data_avail){
887         print("dataset_spec_old already exists")
888
889     }else{
890         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)
891
892
893
894
895
896
897     }
898
899     if("dataset_spec_new" %in% data_avail){
900         print("dataset_spec_new already exists")
901
902     }else{
903         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",
904

```

```

905                                     "0.2200", "0.2300", "0.2400",
                                           "0.2500", "0.2600", "0.2700",
                                           "0.2800", "0.2900", "0.3000",
                                           "0.3100",
906                                     "0.3200", "0.3300", "0.3400",
                                           "0.3500", "0.3650", "0.3850",
                                           "0.4050", "0.4250", "0.4450",
                                           "0.4650",
907                                     "0.4850")
908 dataset_spec_new <- data.frame(matrix(NA, nrow = 1,
ncol = length(colNames_spec_new)))
909 colnames(dataset_spec_new) <- colNames_spec_new
910 dataset_spec_new$DateTime <-
lubridate::ymd_hms(dataset_spec_new$DateTime)
911 }
912
913 # loop through odd frequencies to test if df can be merged
with new and odd frequencies
914 # testing data merge
915 data_list <- ls(pattern = "dataset_")
916 data_list <- data_list[data_list %in%
c("dataset_spec_new", "dataset_spec_old") == FALSE]
917 print(data_list)
918
919 for (l in data_list){
920   print(l)
921   dat <- get(l)
922   spec_list <- ls(pattern = "dataset_spec_")
923   spec_list <- spec_list[spec_list %in%
c("dataset_spec_new", "dataset_spec_old") == TRUE]
924   print(spec_list)
925
926   for(spec in spec_list){
927     if(exists("dat")){
928       spec_dat <- get(spec)
929       ndbc_freq <- names(spec_dat)
930       dat_names <- names(dat)
931       setdiff(ndbc_freq, dat_names)
932       print(unique(dat_names %in% ndbc_freq))
933       # looping through data
934       if(any(unique(dat_names %in%
ndbc_freq)==FALSE)){
935         print(paste0(spec, " didn't
match"))
936       }else{
937         tryCatch({
938           library(plyr)
939           spec_dat <-
rbind.fill(spec_dat, dat
)

```

```

940                                     print(paste0("concat
", spec, " and ", l, "
match"))
941                                     assign(spec, spec_dat)
942                                     rm(dat, spec_dat)
943                                     rm(list =
ls()[grepl(l, ls())])
944                                     }, error = function(e) {
945                                     print("dataset
doesn't match")
946                                     })
947
if(exists("spec_dat")){rm(spec_
dat)}
948                                     }
949                                     rm(spec_dat, ndbc_freq, dat_names)
950                                     }
951
952                                     }
953                                     if(exists("dat")){
954                                     assign(l, dat)
955                                     rm(dat)
956                                     }
957                                     }
958                                     }else{print("no unusual frequency data B")}
959
960
#-----
# 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)

```

```

975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009

# 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("c
              oncat ",spec,
                " and row
                ",i," match"))

            assign(spec,spe
              c_dat)
          }

```

[illegible]



```

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()[grep1(d, ls())])
1048             rm(d)
1049         }
1050     }else{print("no usual frequency data D")}
1051
1052
1053     #-----
1054     # trying to merge remaining odd frequencies within themselves
1055
1056     #-----
1057     data_list <- ls(pattern = "dataset_")
1058     data_list <- data_list[data_list %in%
1059     c("dataset_spec_new", "dataset_spec_old") == FALSE]
1060     print(data_list)
1061     # if there are any odd data left
1062     if(length(data_list)>0){
1063         # test loop three times
1064         for (n in 1:3){
1065             print(n)
1066             data_list <- ls(pattern = "dataset_")
1067             data_list <- data_list[data_list %in%
1068             c("dataset_spec_new", "dataset_spec_old") == FALSE]
1069             print(data_list)
1070             if(length(data_list)>1){
1071                 df_count <- data.frame(matrix(NA, nrow = 0,
1072                 ncol = 2))
1073                 colnames(df_count) <- c("Name", "Count")
1074                 # calculate and save column dimensions
1075                 for(c in data_list){
1076                     df <- data.frame(matrix(NA, nrow = 0,
1077                     ncol = 2))
1078                     colnames(df) <- c("Name", "Count")
1079                     df[1,] <- c(as.character(c),
1080                     dim(get(c))[2])
1081                     df_count <- rbind(df_count, df)
1082                     rm(df)
1083                 }
1084                 # find the df with the most columns
1085                 df_longest <-
1086                 df_count[match(max(df_count$Count, na.rm =
1087                 TRUE), df_count$Count),1]
1088                 print(df_longest)
1089
1090                 # set main with which to merge shorter datasets

```

```

1082 df_orig <- get(df_longest)
1083 rm(list = ls()[grepl(df_longest, ls())])
1084
1085 # search for remaining odd frequency datasets
1086 data_list <- ls(pattern = "dataset_")
1087 data_list <- data_list[data_list %in%
c("dataset_spec_new", "dataset_spec_old") ==
FALSE]
print(data_list)
1088
1089
1090 for (l in data_list){
1091   print(l)
1092   dat <- get(l)
1093   if(exists("dat")){
1094     ndbc_freq <- names(df_orig)
1095     dat_names <- names(dat)
1096     setdiff(ndbc_freq, dat_names)
1097     print(unique(dat_names %in%
ndbc_freq))
1098     # looping through data
1099     if(any(unique(dat_names %in%
ndbc_freq)==FALSE)){
1100       print(paste0(spec, "
didn't match"))
1101     }else{
1102       tryCatch({
1103         library(plyr)
1104         df_orig <-
rbind.fill(df_o
rig, dat)
1105
1106         print(paste0("c
oncat
", df_longest,
" and ", l, "
match"))
1107         dat_name <-
paste0("df_freq
_",
unlist(strsplit
(df_longest, "_"
))[3])
1108         assign(dat_name
, df_orig)
1109         rm(dat)
1110         rm(list =
ls()[grepl(l,
ls())])
}, error =

```

```

1111                                     function(e) {
1112                                             print("dataset
1113                                             doesn't match")
1114                                             })
1115                                             #if(exists("spec_dat"))
1116                                             {rm(spec_dat)}
1117                                             }
1118                                             rm(ndbc_freq, dat_names)
1119                                             }
1120                                             if(exists("dat")){
1121                                             assign(l, dat)
1122                                             rm(dat)
1123                                             }
1124                                             }
1125                                             rm(df_count, df_orig)
1126                                     }else{print("only one odd frequency data left")}
1127                                     }
1128                                     }else{print("no remaining odd frequency data E")}
1129
1130                                     #-----
1131                                     # renaming frequency datasets after trying to merge odd frequencies
1132                                     # with each other
1133
1134                                     #-----
1135                                     data_list <- ls(pattern = "df_freq_")
1136                                     print(data_list)
1137
1138                                     if(length(data_list)>0){
1139                                     for(d in data_list){
1140                                     dat_name <- gsub("df_freq_", "dataset_spec_", d)
1141                                     print(dat_name)
1142                                     assign(dat_name, get(d))
1143                                     rm(list = ls()[grepl(d, ls())])
1144                                     rm(d)
1145                                     }
1146                                     }else{print("no usual frequency data F")}
1147
1148                                     #-----
1149                                     # formatting the merged data
1150
1151                                     #-----

```

```

1147 data_list <- ls(pattern = "dataset_")
1148 print(data_list)
1149
1150 if(length(data_list)>0){
1151   for(d in data_list){
1152     print(paste0("Saving concat... ",d))
1153     dataset <- get(d)
1154     # remove empty rows across df
1155     dataset <- dataset[rowSums(is.na(dataset)) !=
1156       ncol(dataset),]
1157     # remove rows with no datetimes
1158     dataset <- dataset[!is.na(dataset$DateTime),]
1159
1160     if(dim(dataset)[1] != 0){
1161       # remove empty rows - accounting for datasets
1162       # with no lat/lon
1163       column_names <- names(dataset)
1164       if("lat" %in% column_names){
1165         dataset <-
1166           dataset[rowSums(is.na(dataset)) !=
1167             ncol(dataset)-3,]
1168         dataset <-
1169           dataset[rowSums(is.na(dataset)) !=
1170             ncol(dataset)-1,]
1171       }else {dataset <-
1172         dataset[rowSums(is.na(dataset)) !=
1173           ncol(dataset)-1,]}
1174       # remove columns with zero or NA in old freq
1175       if("0.0100" %in% column_names){
1176         if(sum(dataset$`0.0100`, na.rm =
1177           TRUE)==0){dataset$`0.0100` <- NULL}
1178         if(sum(dataset$`0.0200`, na.rm =
1179           TRUE)==0){dataset$`0.0200` <- NULL}
1180       }
1181       if(dim(dataset)[1] != 0){
1182         # deal with skipped GPS positions not
1183         # lasting more than 4 hours
1184         if("lat" %in% names(dataset)){
1185           for(c in
1186             match(c("lat", "lon"), names(data
1187               set))) {
1188             for(loops in 1:3){
1189               # c <- 3
1190               find_NA <-
1191                 which(is.na(dat
1192                   aset[,c]))
1193               NA_count <-
1194                 length(find_NA)

```

1181  
1182  
1183  
1184  
1185  
1186  
1187  
1188  
1189  
1190  
1191  
  
1192  
1193  
1194  
1195  
1196  
1197  
1198  
1199  
1200  
1201  
1202

```
<- 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")})
```

```
print(paste0("Pre  
count of  
NA:  
",NA_count,  
", run:  
",loops))  
for(p in  
find_NA){  
  # p  
  
  #
```

```
}  
find_NA <-  
which(is.na(dat  
aset[,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)  
}
```

```

1203 # ordering the dataset by date and selecting
1204 unique values only
1205 dataset <- dataset[order(dataset$DateTime),]
1206 dataset <- unique(dataset)
1207 # rename the rows to reflect unique data
1208 row.names(dataset) <- 1:nrow(dataset)
1209 # save and export dataset
1210 year_range <-
1211 paste0(year(min(dataset$DateTime, na.rm =
1212 TRUE)), "_", year(max(dataset$DateTime, na.rm
1213 = TRUE)))
1214 records <- nrow(dataset)
1215 if(unlist(strsplit(d, "_"))[3] != "new" &
1216 unlist(strsplit(d, "_"))[3] != "old"){
1217     count_cols <- ncol(dataset)-3
1218     data_name <-
1219     paste0("s_", buoy, "_ncei_", t, "_freq_", co
1220 unt_cols, "cols")
1221 }else{
1222     data_name <-
1223     paste0("s_", buoy, "_ncei_", t, "_freq_", un
1224 list(strsplit(d, "_"))[3])
1225 }
1226 print("exporting datasets")
1227 write.csv(dataset,
1228 paste0(ncei_dir, buoy, "/", data_name, "_", year_ran
1229 ge, "_", records, "_records.csv"),
1230 row.names=FALSE)
1231 # saveRDS(dataset, file =
1232 paste0(ncei_dir, buoy, "/", data_name, "_", year_ran
1233 ge, ".rds"))
1234 # assign(data_name, dataset,
1235 envir=parent.frame())
1236 assign(data_name, dataset)
1237 rm(year_range, records)
1238 rm(dataset)
1239 }else{
1240     rm(dataset)
1241     rm(list = ls()[grepl(d, ls())])
1242 }
1243 }else{print("no data to merge")}
1244
1245 rm_ls <- ls(pattern = "dataset_")
1246 rm(list = rm_ls)
1247 rm(rm_ls, data_name, d, data_list)
1248 } # end of no sensor loop
1249
1250 }else{print(paste0("no dataset:", t))} # end of individual t
1251
1252
1253

```

```

1238     } # end of all t loop
1239
1240     # Save multiple objects
1241     datasets <- ls(pattern = paste0(buoy, "_ncei"))
1242     # return(datasets)
1243     print(datasets)
1244
1245     save(list = datasets, file = paste0(ncei_dir, buoy, "/s_", buoy, "_ncei_ALL.RData"))
1246     # housekeeping
1247     rm(data_name, datasets, file_list, t, dataset)
1248     print(paste0("Completed spectral data concat for ", buoy))
1249     print(paste0("Completed data concat for ", buoy))
1250
1251     rm_ls <- ls(pattern = buoy)
1252     rm(list = rm_ls)
1253     rm(gps_positions)
1254
1255     print(paste0("Finished ncei buoy... ", buoy))
1256
1257     # # Stop writing to the file
1258     # sink()
1259     #-----
1260
1261     print(paste0("Finished ncei buoy... ", buoy))
1262
1263     }else{
1264         print(paste0("No new ncei data for buoy ", buoy))
1265     }
1266 }
1267
1268 #-----
1269 #-----
1270 # clean glob environ
1271 # rm(list = ls())
1272 #-----
1273 #-----
1274 }
1275
1276

```



US Army Corps  
of Engineers®

concat\_ndbc.R



```

1 concat_ndbc <- function(files = "list of files", start_year = "earliest dataset year", t = t, buoy = buoy){
2
3   print(paste0("Working on...",t))
4
5   colNames_stdmet <- c("YYYY", "MM", "DD", "hh", "mm", "WDIR", "WSPD", "GST", "WVHT", "DPD", "APD", "MWD", "PRES", "ATMP", "WTMP",
6                        "DEWP", "VIS", "TIDE")
7
8   NA_strings <- c(9.96920996838687E+36, "9.96920996838687E+36", "9.96921e+36", 9.96921e+36, 999.00, 999,
9                  "-32767", -32767, "-2147483647", -2147483647, NA,
10                  NaN, "NA", "NaN", "9969209968386869046778552952102584320.000",
11                  "9.96920996838686E+36", "99.0", "9999.0", "999", "999.0", "99.00", "999.00")
12
13   date_formatted <- function(df){
14     # find and removed rows with names/text
15     selectedRows <- df[grepl("YY", df[,1]), ]
16     if(nrow(selectedRows) > 0){
17       df <- df[!(df[,1] %in% selectedRows[,1]),]
18     }
19     rm(selectedRows)
20     # convert characters to numeric
21     df[,] <- apply(df[,], 2, function(x) as.numeric(as.character(x)))
22     # adding leading 0 to month, day and hour
23     df$MM <- formatC(df$MM, width = 2, format = "d", flag = "0")
24     df$DD <- formatC(df$DD, width = 2, format = "d", flag = "0")
25     df$hh <- formatC(df$hh, width = 2, format = "d", flag = "0")
26     df$mm <- formatC(df$mm, width = 2, format = "d", flag = "0")
27     if(names(df[1]) == "YY"){
28       if(nchar(df[1,1]) == 2){
29         df$year <- as.integer(19)
30         colY = c(grep("year", colnames(df)), grep("YY", colnames(df)))
31         df <- cbind(YYYY = do.call(paste0, df[colY]), df)
32         df <- dplyr::select(df, -"YY", -"year")
33       }else{
34         df <- dplyr::rename(df, "YYYY" = "YY")
35       }
36     }
37     # Creating concatenated individual and combined date and time columns
38     colst = c(grep("YYYY", colnames(df)), grep("MM", colnames(df)),
39              grep("DD", colnames(df)), grep("hh", colnames(df)),
40              grep("mm", colnames(df)))
41     df <- cbind(DateTime = do.call(paste0, df[colst]), df)
42     ## setting the date and time format
43     library(lubridate)
44     df$DateTime <- ymd_hm(df$DateTime)
45     # ordering the dataset by date and selecting unique values only
46     df <- df[order(df$DateTime),]
47     df <- unique(df)
48     # rename the rows to reflect unique data
49     row.names(df) <- 1:nrow(df)
50     # 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",
82                             "0.1000", "0.1100",
83                             "0.1200", "0.1300", "0.1400", "0.1500", "0.1600", "0.1700", "0.1800", "0.1900",
84                             "0.2000", "0.2100",
85                             "0.2200", "0.2300", "0.2400", "0.2500", "0.2600", "0.2700", "0.2800", "0.2900",
86                             "0.3000", "0.3100",
87                             "0.3200", "0.3300", "0.3400", "0.3500", "0.3650", "0.3850", "0.4050", "0.4250",
88                             "0.4450", "0.4650",
89                             "0.4850")
90     colNames_spec_old <- c("DateTime",
91                             "0.0300", "0.0400", "0.0500", "0.0600", "0.0700", "0.0800", "0.0900", "0.1000", "0.1100", "0.1200",
92                             "0.1300", "0.1400", "0.1500", "0.1600", "0.1700", "0.1800", "0.1900", "0.2000", "0.2100",
93                             "0.2200",
94                             "0.2300", "0.2400", "0.2500", "0.2600", "0.2700", "0.2800", "0.2900", "0.3000", "0.3100",
95                             "0.3200",
96                             "0.3300", "0.3400", "0.3500", "0.3600", "0.3700", "0.3800", "0.3900", "0.4000")

```

```

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 ##                   ##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
108 ##                   .0775
109 ##                   .1700
110 ##                   .1800 .1900 .2000 .2100 .2200 .2300 .2400 .2500 .2600 .2700
111 ##                   .2800
112 ##                   .2900 .3000 .3100 .3200 .3300 .3400 .3500 .3650 .3850 .4050
113 ##                   .4250
114 ##                   .4450 .4650 .4850
115
116 ## selecting files to concatenate, then concatenating
117 # loop through files and concat
118 if(t == "h"){
119
120     library(lubridate)
121     if(buoy == "46029"){
122         dates <- c(2006:year(Sys.Date()))
123     }else{
124         dates <- c(2007:year(Sys.Date()))
125     }
126     ## find the files for this data setup
127     file_dates <- list()
128     for (i in 1:length(dates)){
129         file_dates[[i]] <- files[grepl(paste0(dates[i], "_"), files)]
130     }
131     # remove list function from subset data list
132     file_dates <- unlist(file_dates)
133
134     # print(file_dates)
135     if(length(file_dates) > 0){
136         for(file in file_dates){
137             # if the merged dataset doesn't exist, create it
138             if (!exists("dataset")){
139                 dataset <- read.table(file, header=FALSE, na.strings = NA_strings, fill = T, skip = 2)

```

```

137     }
138     # if the merged dataset does exist, append to it
139     if (exists("dataset")){
140       temp_dataset <- read.table(file, header=FALSE, na.strings = NA_strings, fill = T, skip = 2)
141       dataset<-rbind(dataset, temp_dataset)
142       rm(temp_dataset)
143     }
144   }
145   colnames(dataset) <- colNames_stdmet
146   dataset <- date_formatted(dataset)
147   dataset_master <- dataset
148   rm(dataset)
149   # return(dataset_master)
150 }
151 }else{
152   library(lubridate)
153   dates <- c(2007:year(Sys.Date()))
154   ## find the files for this data setup
155   file_dates <- list()
156   for (i in 1:length(dates)){
157     file_dates[[i]] <- files[grepl(paste0(dates[i], "_"), files)]
158   }
159   # remove list function from subset data list
160   file_dates <- unlist(file_dates)
161   # print(file_dates)
162   if(length(file_dates) > 0){
163     for(file in file_dates){
164       # check for empty files
165       if(file.info(file)$size !=0){
166         # read header data
167         header <- read_header(file)
168         # read file
169         dataset <- read.table(file, header=FALSE, na.strings = NA_strings, fill = T, skip = 0)
170         names(dataset) <- header
171         # format date
172         dataset <- date_formatted(dataset)
173         # find all available frames
174         dataset_list <- ls(pattern = "dataset_spec")
175
176         for(matchable in dataset_list){
177           dat <- get(matchable)
178           if(exists("dataset")){
179             if(identical(names(dat), names(dataset))){
180               dat<-rbind(dat, dataset)
181               print(paste0("added to: ", matchable))
182               assign(matchable, dat)
183               rm(dataset)
184             }
185           }
186           rm(dat)

```

```

187     }
188     if(exists("dataset")){
189         count <- count + 1
190         dataset_spec <- data.frame(matrix(NA, nrow = 0, ncol = dim(dataset)[2]))
191         dataset_spec<-rbind(dataset_spec, dataset)
192         new_name <- paste0("dataset_spec_",count)
193         assign(new_name, dataset_spec)
194         print(paste0("data added to NEW DF:: ",new_name))
195         rm(dataset, dataset_spec)
196     }
197     }else{print("no data in these files")}
198 }
199 }
200 }
201
202 ##-----
203
204 ## 2005 & 2006 - skip first line, has no # flag
205
206 ## stdmet data format:   YYYY MM DD hh mm   WD   WSPD GST   WVHT   DPD   APD   MWD   BAR   ATMP   WTMP   DEWP   VIS   TIDE
207 ##                      units: no units in files
208 ## spectral data format: YYYY MM DD hh mm   .030   .040   .050   .060   .070   .080   .090   .100   .110   .120
209 ##                      .130   .140   .150   .160   .170   .180   .190   .200   .210   .220
210 ##                      .230   .240   .250   .260   .270   .280   .290   .300   .310   .320
211 ##                      .330   .340   .350   .360   .370   .380   .390   .400
212
213 # loop through files and concat
214 if(t == "h"){
215     ## selecting files to concatenate, then concatenating
216     library(lubridate)
217     if(buoy == "46029"){
218         dates <- c(2004,2005)
219     }else{
220         dates <- c(2005,2006)
221     }
222     ## find the files for this data setup
223     file_dates <- list()
224     for (i in 1:length(dates)){
225         file_dates[[i]] <- files[grepl(paste0(dates[i],"_"),files)]
226     }
227     # remove list function from subset data list
228     file_dates <- unlist(file_dates)
229     # print(file_dates)
230
231     if(length(file_dates) > 0){
232         for(file in file_dates){
233             # if the merged dataset doesn't exist, create it
234             if (!exists("dataset")){
235                 dataset <- read.table(file, header=FALSE, na.strings = NA_strings, fill = T, skip = 1)
236             }

```

```

237         # if the merged dataset does exist, append to it
238         if (exists("dataset")){
239             temp_dataset <-read.table(file, header=FALSE, na.strings = NA_strings, fill = T, skip =
240                                     1)
241             dataset<-rbind(dataset, temp_dataset)
242             rm(temp_dataset)
243         }
244         colnames(dataset) <- colNames_stdmet
245         dataset <- date_formatted(dataset)
246         ## combining datasets
247         if(exists("dataset_master")){
248             dataset_master<-rbind(dataset_master, dataset)
249         }else{dataset_master<-dataset}
250         rm(dataset)
251         # return(dataset_master)
252     }
253 }
254 }else{
255     dates <- c(2005,2006)
256     ## find the files for this data setup
257     file_dates <- list()
258     for (i in 1:length(dates)){
259         file_dates[[i]] <- files[grepl(paste0(dates[i], "_"), files)]
260     }
261     # remove list function from subset data list
262     file_dates <- unlist(file_dates)
263     # print(file_dates)
264     if(length(file_dates) > 0){
265         for(file in file_dates){
266             # check for empty files
267             if(file.info(file)$size !=0){
268                 header <- read_header(file)
269                 # read file
270                 dataset <- read.table(file, header=FALSE, na.strings = NA_strings, fill = T, skip = 0)
271                 names(dataset) <- header
272                 # format date
273                 dataset <- date_formatted(dataset)
274                 # find all available frames
275                 dataset_list <- ls(pattern = "dataset_spec")
276
277                 for(matchable in dataset_list){
278                     dat <- get(matchable)
279                     if(exists("dataset")){
280                         if(identical(names(dat), names(dataset))){
281                             dat<-rbind(dat, dataset)
282                             print(paste0("added to: ",matchable))
283                             assign(matchable,dat)
284                             rm(dataset)
285                         }

```

```

286     }
287     rm(dat)
288   }
289   if(exists("dataset")){
290     count <- count + 1
291     dataset_spec <- data.frame(matrix(NA, nrow = 0, ncol = dim(dataset)[2]))
292     dataset_spec<-rbind(dataset_spec, dataset)
293     new_name <- paste0("dataset_spec_",count)
294     assign(new_name, dataset_spec)
295     print(paste0("data added to NEW DF:: ",new_name))
296     rm(dataset, dataset_spec)
297   }
298 }
299 }
300 }
301 }
302
303
304 ##-----
305
306 ## 2000 & 2004 - no minute column - don't skip lines, missing tide data in some sets
307
308 ## stdmet data format:   YYYY MM DD hh WD   WSPD GST   WVHT   DPD   APD   MWD   BAR   ATMP   WTMP   DEWP   VIS   TIDE
309 ##                      units: no units in files
310 ## spectral data format: YYYY MM DD hh   .030   .040   .050   .060   .070   .080   .090   .100   .110   .120
311 ##                      .130   .140   .150   .160   .170   .180   .190   .200   .210   .220
312 ##                      .230   .240   .250   .260   .270   .280   .290   .300   .310   .320
313 ##                      .330   .340   .350   .360   .370   .380   .390   .400
314
315 Names <- c("YYYY", "MM", "DD", "hh", "WDIR", "WSPD", "GST", "WVHT", "DPD", "APD", "MWD", "PRES", "ATMP", "WTMP",
"DEWP", "VIS", "TIDE")
316
317 # loop through files and concat
318 if(t == "h"){
319   ## selecting files to concatenate, then concatenating
320   library(lubridate)
321   if(buoy == "46029"){
322     dates <- c(2000:2003)
323   }else{
324     dates <- c(2000:2004)
325   }
326   ## find the files for this data setup
327   file_dates <- list()
328   for (i in 1:length(dates)){
329     file_dates[[i]] <- files[grepl(paste0(dates[i], "_"),files)]
330   }
331   # remove list function from subset data list
332   file_dates <- unlist(file_dates)
333   # print(file_dates)
334   if(length(file_dates) > 0){

```

```

335     for(file in file_dates){
336         # if the merged dataset doesn't exist, create it
337         if (!exists("dataset")){
338             dataset <- read.table(file, header=FALSE, na.strings = NA_strings, fill = T)
339         }
340         # if the merged dataset does exist, append to it
341         if (exists("dataset")){
342             temp_dataset <-read.table(file, header=FALSE, na.strings = NA_strings, fill = T)
343             dataset<-rbind(dataset, temp_dataset)
344             rm(temp_dataset)
345         }
346     }
347     # rename columns
348     colnames(dataset) <- Names
349     ## adding in minutes column
350     dataset$mm <- as.integer(0)
351     dataset <- date_formatted(dataset)
352     ## combining datasets
353     if(exists("dataset_master")){
354         dataset_master<-rbind(dataset_master, dataset)
355     }else{dataset_master<-dataset}
356     rm(dataset)
357     # return(dataset_master)
358
359 }
360 }else{
361     dates <- c(2000:2004)
362     ## find the files for this data setup
363     file_dates <- list()
364     for (i in 1:length(dates)){
365         file_dates[[i]] <- files[grepl(paste0(dates[i], "_"),files)]
366     }
367     # remove list function from subset data list
368     file_dates <- unlist(file_dates)
369     if(length(file_dates) > 0){
370         for(file in file_dates){
371             # check for empty files
372             if(file.info(file)$size !=0){
373                 header <- read_header(file)
374                 # read file
375                 dataset <- read.table(file, header=FALSE, na.strings = NA_strings, fill = T, skip = 0)
376                 names(dataset) <- header
377                 # add missing minute column
378                 dataset$mm <- as.integer(0)
379                 # format date
380                 dataset <- date_formatted(dataset)
381                 # find all available frames
382                 dataset_list <- ls(pattern = "dataset_spec")
383
384                 for(matchable in dataset_list){

```



```

385     dat <- get(matchable)
386     if(exists("dataset")){
387         if(identical(names(dat), names(dataset))){
388             dat<-rbind(dat, dataset)
389             print(paste0("added to: ", matchable))
390             assign(matchable, dat)
391             rm(dataset)
392         }
393     }
394     rm(dat)
395 }
396 if(exists("dataset")){
397     count <- count + 1
398     dataset_spec <- data.frame(matrix(NA, nrow = 0, ncol = dim(dataset)[2]))
399     dataset_spec<-rbind(dataset_spec, dataset)
400     new_name <- paste0("dataset_spec_", count)
401     assign(new_name, dataset_spec)
402     print(paste0("data added to NEW DF:: ", new_name))
403     rm(dataset, dataset_spec)
404 }
405 }
406 }
407 }
408 }
409
410 ##-----
411
412 ## 1999 - no TIDE or minute column
413
414 ## stdmet data format:   YYYY MM DD hh WD WSPD GST  WVHT  DPD   APD  MWD  BAR   ATMP  WTMP  DEWP  VIS
415 ##                      units: no units in files
416 ## spectral data format: YYYY MM DD hh   .030  .040  .050  .060  .070  .080  .090  .100  .110  .120
417 ##                      .130  .140  .150  .160  .170  .180  .190  .200  .210  .220
418 ##                      .230  .240  .250  .260  .270  .280  .290  .300  .310  .320
419 ##                      .330  .340  .350  .360  .370  .380  .390  .400
420
421 Names <- c("YYYY", "MM", "DD", "hh", "WDIR", "WSPD", "GST", "WVHT", "DPD", "APD", "MWD", "PRES", "ATMP", "WTMP",
"DEWP", "VIS")
422
423 ## selecting files to concatenate, then concatenating
424 library(lubridate)
425 dates <- c(1999)
426 ## find the files for this data setup
427 file_dates <- list()
428 for (i in 1:length(dates)){
429     file_dates[[i]] <- files[grepl(paste0(dates[i], "_"), files)]
430 }
431 # remove list function from subset data list
432 file_dates <- unlist(file_dates)
433 # 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)

```

```

484     # format date
485     dataset <- date_formatted(dataset)
486     # find all available frames
487     dataset_list <- ls(pattern = "dataset_spec")
488
489     for(matchable in dataset_list){
490         dat <- get(matchable)
491         if(exists("dataset")){
492             if(identical(names(dat), names(dataset))){
493                 dat<-rbind(dat, dataset)
494                 print(paste0("added to: ", matchable))
495                 assign(matchable, dat)
496                 rm(dataset)
497             }
498         }
499         rm(dat)
500     }
501     if(exists("dataset")){
502         count <- count + 1
503         dataset_spec <- data.frame(matrix(NA, nrow = 0, ncol = dim(dataset)[2]))
504         dataset_spec<-rbind(dataset_spec, dataset)
505         new_name <- paste0("dataset_spec_", count)
506         assign(new_name, dataset_spec)
507         print(paste0("data added to NEW DF:: ", new_name))
508         rm(dataset, dataset_spec)
509     }
510 }
511 }
512 }
513 }
514
515 ##-----
516
517 ##1979 & 1998 - no tide and minute columns
518
519 ## stdmet data format:   YY MM DD hh WD   WSPD GST   WVHT   DPD   APD   MWD   BAR   ATMP   WTMP   DEWP   VIS
520 ##                      units: no units in files
521 ## spectral data format: YY MM DD hh   .0200   .0325   .0375   .0425   .0475   .0525   .0575   .0625   .0675   .0725
522 ##                      .0775   .0825   .0875   .0925   .1000   .1100   .1200   .1300   .1400   .1500   .1600   .1700
523 ##                      .1800   .1900   .2000   .2100   .2200   .2300   .2400   .2500   .2600   .2700   .2800   .2900
524 ##                      .3000   .3100   .3200   .3300   .3400   .3500   .3650   .3850   .4050   .4250   .4450   .4650   .4850
525
526 Names <- c("YY", "MM", "DD", "hh", "WDIR", "WSPD", "GST", "WVHT", "DPD", "APD", "MWD", "PRES", "ATMP", "WTMP",
"DEWP", "VIS")
527
528 ## selecting files to concatenate, then concatenating
529 library(lubridate)
530 dates <- c(start_year:1998)
531 ## find the files for this data setup
532 file_dates <- list()

```

```

533 for (i in 1:length(dates)){
534   file_dates[[i]] <- files[grepl(paste0(dates[i], "_"), files)]
535 }
536 # remove list function from subset data list
537 file_dates <- unlist(file_dates)
538
539 # loop through files and concat
540 if(t == "h"){
541   if(length(file_dates) > 0){
542     for(file in file_dates){
543       # if the merged dataset doesn't exist, create it
544       if (!exists("dataset")){
545         dataset <- read.table(file, header=FALSE, na.strings = NA_strings, fill = T, skip = 1)
546       }
547       # if the merged dataset does exist, append to it
548       if (exists("dataset")){
549         temp_dataset <- read.table(file, header=FALSE, na.strings = NA_strings, fill = T, skip = 1)
550         dataset <- rbind(dataset, temp_dataset)
551         rm(temp_dataset)
552       }
553     }
554     ## rename columns
555     colnames(dataset) <- Names
556     ## adding in missing column
557     dataset$TIDE <- as.logical(NA)
558     dataset$Year <- as.integer(19)
559     ## Creating new column with '19', and combining YY column to be 4 digits.
560     colst = c(grep("Year", colnames(dataset)), grep("YY", colnames(dataset)))
561     dataset <- cbind(YYYY = do.call(paste0, dataset[colst]),
562                      dataset)
563     dataset$YYYY <- as.numeric(as.character(dataset$YYYY))
564     ## delete working columns of YY and YY values
565     dataset$Year <- NULL; dataset$YY <- NULL
566     ## adding in minutes column
567     if(buoy == 41009){ # using minute data from NetCDF files
568       dataset$mm <- as.integer(0)
569       dataset <- date_formatted(dataset)
570       dat1 <- filter(dataset, dataset$DateTime < as.Date("1992-08-01 00:00:00"))
571       dat1$mm <- as.numeric(rep(c(00, 30), nrow(dat1)/2))
572       minute(dat1$DateTime) <- as.numeric(dat1$mm)
573       dat1 <- dplyr::select(dat1, -"mm")
574       dat2 <- filter(dataset, dataset$DateTime >= as.Date("1992-08-01 00:00:00"))
575       dat2$mm <- as.numeric(rep(c(20, 50), nrow(dat2)/2))
576       minute(dat2$DateTime) <- as.numeric(dat2$mm)
577       dat2 <- dplyr::select(dat2, -"mm")
578       dataset <- rbind(dat1, dat2)
579       rm(dat1, dat2)
580     }else{
581       dataset$mm <- as.integer(0)
582       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

```

1  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"){
2
3  NA_strings <- c(9.96920996838687E+36,"9.96920996838687E+36","9.96921e+36",9.96921e+36,999.00,999,
4                "-32767",-32767,"-2147483647",-2147483647,NA,
                    NaN,"NA","NaN","9969209968386869046778552952102584320.000",
5                "9.96920996838686E+36","99.0","9999.0","999","999.0","99.00","999.00","", " ")
6  # format header
7  read_header <- function(df){
8    years <- df[1:3]
9    freq <- df[4:length(df)]
10   freq <- as.numeric(freq)
11   freq <- sprintf("%1.4f", freq)
12   freq <- as.character(freq)
13   df <- c(years, freq)
14 }
15 # functions to test for date formats
16 library(lubridate)
17 IsDate_dmy_hms <- function(mydate, date.format = "%d-%m-%Y %h:%m:%s") {
18   tryCatch(!is.na(dmy_hms(mydate, date.format)),
19           error = function(err) {FALSE})
20 }
21 IsDate_ymd_hms <- function(mydate, date.format = "%Y-%m-%d %h:%m:%s") {
22   tryCatch(!is.na(ymd_hms(mydate, date.format)),
23           error = function(err) {FALSE})
24 }
25 IsDate_mdy_hm <- function(mydate, date.format = "%m/%d/%Y %h:%m") {
26   tryCatch(!is.na(mdy_hm(mydate, date.format)),
27           error = function(err) {FALSE})
28 }
29
30 setwd(data_dir)
31
32 raw_dir <- paste0(data_dir,"raw_data/")
33 setwd(data_dir)
34 # set input directories
35 unzip_ndbc_dir <- paste0(raw_dir,"ndbc/unzipped/")
36 ascii_ncei_dir <- paste0(raw_dir,"ncei/ascii/")
37
38 # set new out_dir
39 if (!file.exists(paste0(data_dir,"concat_data/"))) {dir.create((paste0(data_dir,"concat_data/")))}
40 out_dir <- paste0(data_dir,"concat_data/")
41 # ndbc
42 if (!file.exists(paste0(out_dir,"ndbc/"))) {dir.create((paste0(out_dir,"ndbc/")))}
43 ndbc_dir <- paste0(out_dir,"ndbc/")
44 # ncei
45 if (!file.exists(paste0(out_dir,"ncei/"))) {dir.create((paste0(out_dir,"ncei/")))}
46 ncei_dir <- paste0(out_dir,"ncei/")
47
48 # data types

```



```

49 dataTypes <- c("stdmet", "swden", "swdir", "swdir2", "swr1", "swr2")
50 dataType_ab <- c("h", "w", "d", "i", "j", "k")
51 data_types_stdmet <- c("lat", "lon", "wind_direction", "wind_speed", "wind_gust",
52                        "significant_wave_height", "dominant_period", "average_period", "mean_wave_direction",
53                        "air_pressure", "air_temperature", "sea_surface_temperature", "dew_point_temperature")
54 data_types_spec <- c("c11", "c11m", "alpha1", "alpha2", "r1", "r2", "C12", "C13", "C22", "C33",
55                      "Q12", "Q13", "gamma2", "gamma3", "phih", "rhq", "sensor_output") # c11 = spectral energy,
56                                           c11m = uncorrected spectral energy
57
58 if(input == "stdmet"){
59     #-----
60     # met data
61     #-----
62     # # merging yearly and monthly datafiles of each type
63     library(lubridate)
64     library(tidyr)
65     # standard met data - create NCEI matching NDBC web file nomenclature
66     for (t in data_types_stdmet){
67         print(paste0("Working on...",t))
68
69
70         if (!file.exists(paste0(ncei_dir,buoy,"/data_availability")))
71         {dir.create((paste0(ncei_dir,buoy,"/data_availability")))}
72         data_avail_dir <- paste0(ncei_dir,buoy,"/data_availability/s_")
73
74         # add sensor information
75         metadata <- read.csv(paste0(raw_dir,"ncei/metadata/",buoy,"_metadata_ALL.csv"))
76
77         # subset conditions for each variable - because NDBC loves duplication and format change for no reason
78         # handle different netcdf variable names for waves
79         if(grepl("dominant", t)){t1 <- "dominant_wave_period"; files2 <- file_list[grepl(pattern = paste0("_",t1),
80         file_list)]}
81         if(grepl("average", t)){t1 <- "average_wave_period"; files2 <- file_list[grepl(pattern = paste0("_",t1),
82         file_list)]}
83         files <- file_list[grepl(pattern = paste0("_",t), file_list)]
84         if(exists("files2")){files <- c(files2, files); rm(files2)}
85
86         # available files ALL
87         files <- sort(files)
88         df_files <- gsub(paste0(ascii_ncei_dir, buoy,"/",buoy,"_"),"",files)
89         df_files <- df_files[order(files)]
90         df_files <- unique(df_files)
91         df_files <- data.frame(df_files, stringsAsFactors = F)
92         df_files <- separate(df_files, df_files, into = c('date', 'file'),sep = "/", remove = TRUE)
93         write.csv(df_files, paste0(data_avail_dir,buoy, "_",t,"_available_files_all.csv"), row.names=FALSE)
94
95         if(t == "lat"){
96             # used:
97             df_files <- gsub(paste0(ascii_ncei_dir, buoy,"/",buoy,"_"),"",files)

```

```

95     df_files <- df_files[order(files)]
96     df_files <- unique(df_files)
97     df_files <- data.frame(df_files, stringsAsFactors = F)
98     df_files <- separate(df_files, df_files, into = c('date', 'file'), sep = "/", remove = TRUE)
99     write.csv(df_files, paste0(data_avail_dir, buoy, "_", t, "_available_files_USED.csv"), row.names=FALSE)
100     rm(df_files)
101
102   }else if(t == "lon"){
103     # removed:
104     variable2 <- files[grepl(pattern = "_longwave_", files)]
105     if(length(variable2) != 0){files <- files[!grepl(paste(variable2, collapse="|"), files)]}
106     # used:
107     df_files <- gsub(paste0(ascii_ncei_dir, buoy, "/", buoy, "_"), "", files)
108     df_files <- df_files[order(files)]
109     df_files <- unique(df_files)
110     df_files <- data.frame(df_files, stringsAsFactors = F)
111     df_files <- separate(df_files, df_files, into = c('date', 'file'), sep = "/", remove = TRUE)
112     write.csv(df_files, paste0(data_avail_dir, buoy, "_", t, "_available_files_USED.csv"), row.names=FALSE)
113     rm(df_files)
114
115   }else if(t == "wind_direction"){
116     # removed:
117     continuous <- files[grepl(pattern = paste0("_continuous_", t), files)]
118     if(length(continuous) != 0){files <- files[!grepl(paste(continuous, collapse="|"), files)]}
119     wind_dir_58 <- files[grepl(pattern = "_58", files)]
120     if(length(wind_dir_58) != 0){files <- files[!grepl(paste(wind_dir_58, collapse="|"), files)]}
121     # isolate secondary files
122     files_secondary <- files[grepl(pattern = "_2_", files)]
123     if(length(files_secondary) != 0){if(unique(files == files_secondary)==TRUE){files_secondary <- NULL}}
124     if(length(files_secondary) != 0){files <- files[!grepl(paste(files_secondary, collapse="|"), files)]}
125     if(length(files_secondary) == 0){rm(files_secondary)}
126     # used:
127     df_files <- gsub(paste0(ascii_ncei_dir, buoy, "/", buoy, "_"), "", files)
128     df_files <- df_files[order(files)]
129     df_files <- unique(df_files)
130     df_files <- data.frame(df_files, stringsAsFactors = F)
131     df_files <- separate(df_files, df_files, into = c('date', 'file'), sep = "/", remove = TRUE)
132     write.csv(df_files, paste0(data_avail_dir, buoy, "_", t, "_available_files_USED.csv"), row.names=FALSE)
133     rm(df_files)
134
135     # filter metadata
136     metadata <- dplyr::filter(metadata, grepl("anemometer", payload_sensor))
137
138   }else if(t == "wind_speed"){
139     # removed:
140     continuous <- files[grepl(pattern = paste0("_continuous_", t), files)]
141     if(length(continuous) != 0){files <- files[!grepl(paste(continuous, collapse="|"), files)]}
142     max_1 <- files[grepl(pattern = "_max_1", files)]
143     if(length(max_1) != 0){files <- files[!grepl(paste(max_1, collapse="|"), files)]}
144     peak <- files[grepl(pattern = "_peak_", files)]

```

```

145     if(length(peak) != 0){files <- files[!grepl(paste(peak, collapse="|"), files)]}
146     wind_speed_58 <- files[grepl(pattern = "_58", files)]
147     if(length(wind_speed_58) != 0){files <- files[!grepl(paste(wind_speed_58, collapse="|"), files)]}
148     # isolate secondary files
149     files_secondary <- files[grepl(pattern = "_2_", files)]
150     if(length(files_secondary) != 0){if(unique(files == files_secondary)==TRUE){files_secondary <- NULL}}
151     if(length(files_secondary) != 0){files <- files[!grepl(paste(files_secondary, collapse="|"), files)]}
152     if(length(files_secondary) == 0){rm(files_secondary)}
153     # used:
154     df_files <- gsub(paste0(ascii_ncei_dir, buoy, "/", buoy, "_"), "", files)
155     df_files <- df_files[order(files)]
156     df_files <- unique(df_files)
157     df_files <- data.frame(df_files, stringsAsFactors = F)
158     df_files <- separate(df_files, df_files, into = c('date', 'file'), sep = "/", remove = TRUE)
159     write.csv(df_files, paste0(data_avail_dir, buoy, "_", t, "_available_files_USED.csv"), row.names=FALSE)
160     rm(df_files)
161
162     # filter metadata
163     metadata <- dplyr::filter(metadata, grepl("anemometer", payload_sensor))
164
165   }else if(t == "wind_gust"){
166     # removed:
167     ave <- files[grepl(pattern = paste0(t, "_averaging_period"), files)]
168     if(length(ave) != 0){files <- files[!grepl(paste(ave, collapse="|"), files)]}
169     windgust_2 <- files[grepl(pattern = "_wind_gust_2", files)]
170     if(length(windgust_2) != 0){files <- files[!grepl(paste(windgust_2, collapse="|"), files)]}
171     wind_gust_58 <- files[grepl(pattern = "_58", files)]
172     if(length(wind_gust_58) != 0){files <- files[!grepl(paste(wind_gust_58, collapse="|"), files)]}
173     # isolate secondary files
174     files_secondary <- files[grepl(pattern = "_2_", files)]
175     if(length(files_secondary) != 0){if(unique(files == files_secondary)==TRUE){files_secondary <- NULL}}
176     if(length(files_secondary) != 0){files <- files[!grepl(paste(files_secondary, collapse="|"), files)]}
177     if(length(files_secondary) == 0){rm(files_secondary)}
178     # used:
179     df_files <- gsub(paste0(ascii_ncei_dir, buoy, "/", buoy, "_"), "", files)
180     df_files <- df_files[order(files)]
181     df_files <- unique(df_files)
182     df_files <- data.frame(df_files, stringsAsFactors = F)
183     df_files <- separate(df_files, df_files, into = c('date', 'file'), sep = "/", remove = TRUE)
184     write.csv(df_files, paste0(data_avail_dir, buoy, "_", t, "_available_files_USED.csv"), row.names=FALSE)
185     rm(df_files)
186
187     # filter metadata
188     metadata <- dplyr::filter(metadata, grepl("anemometer", payload_sensor))
189
190   }else if(t == "dominant_period"){
191     # isolate secondary files
192     files_secondary <- files[grepl(pattern = "_2_", files)]
193     if(length(files_secondary) != 0){if(unique(files == files_secondary)==TRUE){files_secondary <- NULL}}
194     if(length(files_secondary) != 0){files <- files[!grepl(paste(files_secondary, collapse="|"), files)]}

```

```

195     if(length(files_secondary) == 0){rm(files_secondary)}
196     # used:
197     df_files <- gsub(paste0(ascii_ncei_dir, buoy,"/",buoy,"_"), "", files)
198     df_files <- df_files[order(files)]
199     df_files <- unique(df_files)
200     df_files <- data.frame(df_files, stringsAsFactors = F)
201     df_files <- separate(df_files, df_files, into = c('date', 'file'), sep = "/", remove = TRUE)
202     write.csv(df_files, paste0(data_avail_dir, buoy, "_", t, "_available_files_USED.csv"), row.names=FALSE)
203     rm(df_files)
204
205     # filter metadata
206     metadata <- dplyr::filter(metadata, grepl("wave_sensor", payload_sensor))
207
208   }else if(t == "average_period"){
209     # isolate secondary files
210     files_secondary <- files[grepl(pattern = "_2_", files)]
211     if(length(files_secondary) != 0){if(unique(files == files_secondary)==TRUE){files_secondary <- NULL}}
212     if(length(files_secondary) != 0){files <- files[!grepl(paste(files_secondary, collapse="|"), files)]}
213     if(length(files_secondary) == 0){rm(files_secondary)}
214     # used:
215     df_files <- gsub(paste0(ascii_ncei_dir, buoy,"/",buoy,"_"), "", files)
216     df_files <- df_files[order(files)]
217     df_files <- unique(df_files)
218     df_files <- data.frame(df_files, stringsAsFactors = F)
219     df_files <- separate(df_files, df_files, into = c('date', 'file'), sep = "/", remove = TRUE)
220     write.csv(df_files, paste0(data_avail_dir, buoy, "_", t, "_available_files_USED.csv"), row.names=FALSE)
221     rm(df_files)
222
223     # filter metadata
224     metadata <- dplyr::filter(metadata, grepl("wave_sensor", payload_sensor))
225
226   }else if(t == "air_pressure"){
227     # isolate secondary files
228     files_secondary <- files[grepl(pattern = "_2_", files)]
229     if(length(files_secondary) != 0){if(unique(files == files_secondary)==TRUE){files_secondary <- NULL}}
230     if(length(files_secondary) != 0){files <- files[!grepl(paste(files_secondary, collapse="|"), files)]}
231     if(length(files_secondary) == 0){rm(files_secondary)}
232     # # used:
233     df_files <- gsub(paste0(ascii_ncei_dir, buoy,"/",buoy,"_"), "", files)
234     df_files <- df_files[order(files)]
235     df_files <- unique(df_files)
236     df_files <- data.frame(df_files, stringsAsFactors = F)
237     df_files <- separate(df_files, df_files, into = c('date', 'file'), sep = "/", remove = TRUE)
238     write.csv(df_files, paste0(data_avail_dir, buoy, "_", t, "_available_files_USED.csv"), row.names=FALSE)
239     rm(df_files)
240
241     # filter metadata
242     metadata <- dplyr::filter(metadata, grepl("barometer", payload_sensor))
243
244   }else if(t == "air_temperature"){

```

```

245     # removing dew point temperature
246     dew <- files[grep(pattern = "_dew_", files)]
247     if(length(dew) != 0){files <- files[!grepl(paste(dew, collapse="|"), files)]}
248     # isolate secondary files
249     files_secondary <- files[grep(pattern = "_2_", files)]
250     if(length(files_secondary) != 0){if(unique(files == files_secondary)==TRUE){files_secondary <- NULL}}
251     if(length(files_secondary) != 0){files <- files[!grepl(paste(files_secondary, collapse="|"), files)]}
252     if(length(files_secondary) == 0){rm(files_secondary)}
253     # used:
254     df_files <- gsub(paste0(ascii_ncei_dir, buoy, "/", buoy, "_"), "", files)
255     df_files <- df_files[order(files)]
256     df_files <- unique(df_files)
257     df_files <- data.frame(df_files, stringsAsFactors = F)
258     df_files <- separate(df_files, df_files, into = c('date', 'file'), sep = "/", remove = TRUE)
259     write.csv(df_files, paste0(data_avail_dir, buoy, "_", t, "_available_files_USED.csv"), row.names=FALSE)
260     rm(df_files)
261
262     # filter metadata
263     metadata <- dplyr::filter(metadata, grepl("air_temperature", payload_sensor))
264
265   }else { # significant wave height, mean wave direction, sea surface temperature, dew point temperature
266     # # isolate secondary files
267     files_secondary <- files[grep(pattern = "_2_", files)]
268     if(length(files_secondary) != 0){if(unique(files == files_secondary)==TRUE){files_secondary <- NULL}}
269     if(length(files_secondary) != 0){files <- files[!grepl(paste(files_secondary, collapse="|"), files)]}
270     if(length(files_secondary) == 0){rm(files_secondary)}
271     # used:
272     df_files <- gsub(paste0(ascii_ncei_dir, buoy, "/", buoy, "_"), "", files)
273     df_files <- df_files[order(files)]
274     df_files <- unique(df_files)
275     df_files <- data.frame(df_files, stringsAsFactors = F)
276     df_files <- separate(df_files, df_files, into = c('date', 'file'), sep = "/", remove = TRUE)
277     write.csv(df_files, paste0(data_avail_dir, buoy, "_", t, "_available_files_USED.csv"), row.names=FALSE)
278     rm(df_files)
279
280     if(t == "dew_point_temperature"){metadata <- dplyr::filter(metadata,
281       grepl("air_temperature", payload_sensor))
282     }else if(t == "sea_surface_temperature"){metadata <- dplyr::filter(metadata,
283       grepl("ocean_temperature", payload_sensor))
284     }else {metadata <- dplyr::filter(metadata, grepl("wave_sensor", payload_sensor))}
285   }
286   print(head(files))
287   print(tail(files))
288
289   if(length(files)>0){
290     # loop for each data type
291     if(exists("files_secondary")){
292       data_sets <- c("files", "files_secondary")
293     } else{
294       data_sets <- "files"
295     }
296   }

```

```

293 }
294 # order metadata file
295 if(dim(metadata)[1]>0){
296   metadata <- metadata[order(metadata$file),]
297   # rename the rows to reflect unique data
298   row.names(metadata) <- 1:nrow(metadata)
299 }
300 for (df in data_sets){
301   df_2 <- get(df)
302   for (file in df_2){
303     print(file)
304     # if the merged dataset doesn't exist, create it
305     if (!exists("dataset")){
306       dataset <- read.table(file, header=TRUE, na.strings = NA_strings, fill = T, sep =
307         ",", stringsAsFactors = FALSE)
308       # set date format
309       if(IsDate_dmy_hms(dataset[,1])==TRUE){dataset[,1] <- dmy_hms(dataset[,1]);
310         print("dmy_hms dates converted")}
311       if(IsDate_ymd_hms(dataset[,1])==TRUE){dataset[,1] <- ymd_hms(dataset[,1]);
312         print("ymd_hms dates converted")}
313       if(IsDate_mdy_hm(dataset[,1])==TRUE){dataset[,1] <- mdy_hm(dataset[,1]);
314         print("mdy_hm dates converted")}
315
316       # reformatting the data structure
317       if("qc_flag" %in% names(dataset)){print("qc data available")}else{dataset$qc_flag <-
318         NA}
319       if("release_flag" %in% names(dataset)){print("release_flag
320         available")}else{dataset$release_flag <- NA}
321
322       if(dim(dataset)[2]>4){
323         if(t == "lat"){dataset <- dplyr::select(dataset, time, lat, qc_flag, release_flag)}
324         if(t == "lon"){dataset <- dplyr::select(dataset, time, lon, qc_flag, release_flag)}
325       }
326       # converting Kelvin to deg C if necessary
327       if(t != "lat" & t != "lon"){
328         if("temperature" %in% unlist(strsplit(t, "_"))){
329           dataset[,4] <- ifelse(dataset[,4] >= 100,
330             dataset[,4] - 273.15,
331             dataset[,4])
332           print("temperature converted from Kelvin to C")
333         }
334         # converting pressure to bars if necessary
335         if("pressure" %in% unlist(strsplit(t, "_"))){
336           dataset[,4] <- ifelse(dataset[,4] > 10000,
337             dataset[,4] / 100,
338             dataset[,4])
339           print("pressure convert to bars")
340         }
341         if("lat" %in% names(dataset)){dataset <- dplyr::select(dataset, -"lat");
342           print("lat removed before merge")}

```

```

336         if("lon" %in% names(dataset)){dataset <- dplyr::select(dataset, ~"lon");
337         print("lon removed before merge")}
338     }
339     colnames(dataset) <- c("DateTime", t, "qc_flag", "release_flag")
340     # save only unique data
341     dataset <- unique(dataset)
342     # add sensor information
343     dataset$payload_sensor <-
344     gsub(".csv", "", unlist(strsplit(file, "/"))[length(unlist(strsplit(file, "/")))])
345 }
346 # if the merged dataset does exist, append to it
347 if (exists("dataset")){
348     temp_dataset <- read.table(file, header=TRUE, na.strings = NA_strings, fill = T, sep =
349     ",", stringsAsFactors = FALSE)
350
351     # set date format
352     if(IsDate_dmy_hms(temp_dataset[,1])==TRUE){temp_dataset[,1] <-
353     dmy_hms(temp_dataset[,1]); print("dmy_hms dates converted")}
354     if(IsDate_ymd_hms(temp_dataset[,1])==TRUE){temp_dataset[,1] <-
355     ymd_hms(temp_dataset[,1]); print("ymd_hms dates converted")}
356     if(IsDate_mdy_hm(temp_dataset[,1])==TRUE){temp_dataset[,1] <-
357     mdy_hm(temp_dataset[,1]); print("mdy_hm dates converted")}
358
359     # reformatting the data structure
360     if("QC_Flags" %in% names(temp_dataset)){temp_dataset <- dplyr::rename(temp_dataset,
361     "qc_flag" = "QC_Flags")}
362     if("qc_flag" %in% names(temp_dataset)){print("qc data
363     available")}else{temp_dataset$qc_flag <- NA}
364     if("release_flag" %in% names(temp_dataset)){print("release_flag
365     available")}else{temp_dataset$release_flag <- NA}
366
367     if(dim(temp_dataset)[2]>4){
368         if(t == "lat"){temp_dataset <- dplyr::select(temp_dataset, time, lat, qc_flag,
369         release_flag)}
370         if(t == "lon"){temp_dataset <- dplyr::select(temp_dataset, time, lon, qc_flag,
371         release_flag)}
372     }
373     # converting Kelvin to deg C if necessary
374     if(t != "lat" & t != "lon"){
375         if("temperature" %in% unlist(strsplit(t, "_"))){
376             temp_dataset[,4] <- ifelse(temp_dataset[,4] > 100,
377             temp_dataset[,4] - 273.15,
378             temp_dataset[,4])
379             print("temperature converted from Kelvin to C")
380         }
381         # converting pressure to bars if necessary
382         if("pressure" %in% unlist(strsplit(t, "_"))){
383             temp_dataset[,4] <- ifelse(temp_dataset[,4] > 10000,
384             temp_dataset[,4] / 100,
385             temp_dataset[,4])
386         }
387     }
388 }

```

```

375         print("pressure convert to bars")
376     }
377     if("lat" %in% names(temp_dataset)){temp_dataset <- dplyr::select(temp_dataset,
378         -"lat"); print("lat removed before merge")}
379     if("lon" %in% names(temp_dataset)){temp_dataset <- dplyr::select(temp_dataset,
380         -"lon"); print("lon removed before merge")}
381     }
382     colnames(temp_dataset) <- c("DateTime", t, "qc_flag", "release_flag")
383     # save only unique data
384     temp_dataset <- unique(temp_dataset)
385     # add sensor information
386     temp_dataset$payload_sensor <-
387     gsub(".csv", "", unlist(strsplit(file, "/"))[length(unlist(strsplit(file, "/")))])
388     # merge datasets
389     dataset<-rbind(dataset, temp_dataset)
390     rm(temp_dataset)
391 }
392 # ordering the dataset by date and selecting unique values only
393 dataset <- dataset[order(dataset$DateTime),]
394 dataset <- unique(dataset)
395 # rename the rows to reflect unique data
396 row.names(dataset) <- 1:nrow(dataset)
397 # for saving
398 year_range <- paste0(year(dataset$DateTime[1]), "_", year(dataset$DateTime[nrow(dataset)]))
399 # sep primary and secondary data
400 if(df == "files"){concat_name <- paste0(buoy, "_", t, "_", year_range, "_concat_primary")}
401 if(df == "files_secondary"){concat_name <- paste0(buoy, "_", t, "_", year_range, "_concat_secondary")}
402 # save individual datasets
403 library(lubridate)
404 # write.csv(dataset, paste0(concat_ind_dir, concat_name, ".csv"), row.names=FALSE)
405 # saveRDS(dataset, file = paste0(concat_ind_dir, concat_name, ".rds"))
406
407 # remove flagged data
408 # Looping through all NDBC QC flags and removing flagged raw data
409 dataset[,3][is.na(dataset[,3])] <- 0 # replace NA with zero when flags not
410 included in original dataset
411 dataset[,4][is.na(dataset[,4])] <- 0
412 flags <- c("W", "R", "V", "M", "T", "D", "U", "L", "H", "2", "3", "_") # ignore S flags
413 for (i in flags){
414     # i <- flags[1]
415     index <- dataset$qc_flag == i
416     dataset[index,2] <- NA
417     # dataset <- subset(dataset, qc_flag != i)
418 }
419 # order for released data and remove duplicated date rows
420 if(nrow(dataset[!duplicated(dataset$DateTime),]) != nrow(dataset)){
421     if(length(unique(dataset$release_flag))>2){
422         dataset = dataset[order(dataset[, 'DateTime'], -dataset[, 'release_flag']),]
423         dataset = dataset[!duplicated(dataset$DateTime),]

```



```

421         print("removing duplicated data based on release flag")
422     }else{
423         dataset = dataset[!duplicated(dataset$DateTime),]
424         print("removing duplicated data NOT based on release flag")
425     }
426 }
427
428 # remove redundant columns
429 library(dplyr)
430 dataset <- dplyr::select(dataset,-"qc_flag")
431 dataset <- dplyr::select(dataset,-"release_flag")
432
433 # sep primary and secondary data
434 if(df == "files"){dataset1 <- dataset}
435 if(df == "files_secondary"){dataset2 <- dataset}
436 rm(dataset)
437 } # loop through datasets
438
439 # loop through multiple payloads
440 if(length(data_sets) > 1){
441     if(dim(metadata)[1]>0){
442         # prepping the metadata
443         metadata1 <- metadata
444         metadata1$join <- gsub("/", "_", paste0(gsub(".nc", "", metadata1$file),
445             "_", metadata1$payload_sensor, "_", t))
446         metadata1 <- metadata1[,7:ncol(metadata1)]
447         # handling different air pressure nomenclature
448         if(t == "air_pressure"){
449             metadata1$join <- gsub("_air_pressure_at_sea_level", "_air_pressure", metadata1$join)
450             dataset1$payload_sensor <- gsub("_air_pressure_at_sea_level", "_air_pressure",
451                 dataset1$payload_sensor)
452             dataset2$payload_sensor <- gsub("_air_pressure_at_sea_level", "_air_pressure",
453                 dataset2$payload_sensor)
454         }
455         # add primary sensor information
456         dataset1 <- dplyr::rename(dataset1, join = payload_sensor)
457         dataset1 <- left_join(dataset1, metadata1, by = "join")
458         # add secondary sensor information
459         dataset2 <- dplyr::rename(dataset2, join = payload_sensor)
460         dataset2 <- left_join(dataset2, metadata1, by = "join")
461         # rename columns
462         colnames(dataset1) <- c("DateTime", paste0(colnames(dataset1)[2:ncol(dataset1)], "_1"))
463         colnames(dataset2) <- c("DateTime", paste0(colnames(dataset2)[2:ncol(dataset2)], "_2"))
464         rm(metadata, metadata1)
465     }else{
466         dataset$payload_sensor <- "no information available"
467         colnames(dataset) <- c("DateTime", t, paste0(t, "_payload_sensor"))
468     }
469     # Prep for secondary sensor data if primary sensor data is NA
470     dataset <- full_join(dataset1, dataset2, by = "DateTime")

```

```

468 rm(dataset1,dataset2)
469 # rename columns
470 colnames(dataset) = gsub("\\\\.x", "_1", colnames(dataset))
471 colnames(dataset) = gsub("\\\\.y", "_2", colnames(dataset))
472 # check for duplicates
473 dataset <- dataset[order(dataset$DateTime),]
474 dataset <- unique(dataset)
475 # rename the rows to reflect unique data
476 row.names(dataset) <- 1:nrow(dataset)
477 # export data
478 year_range <- paste0(year(min(dataset$DateTime,na.rm = TRUE)), "_", year(max(dataset$DateTime,
na.rm = TRUE)))
479 # write.csv(dataset, paste0(ncei_dir,buoy,"/s_",buoy,
"_ncei_",t,"_primary_secondary_",year_range,".csv"), row.names=FALSE)
480 # saveRDS(dataset, file =
paste0(ncei_dir,buoy,"/s_",buoy,"_ncei_",t,"_primary_secondary_",year_range,".rds"))
481
482 # concat metadata
483 dataset$metadata_1 <-
paste0(dataset[,3],"_",dataset[,4],"_",dataset[,5],"_",dataset[,7],"_",dataset[,8],"_",dataset
[,9],"_",dataset[,10],"_",dataset[,11],"_",dataset[,12])
484 dataset$metadata_1 <- gsub("__NA","",dataset$metadata_1)
485 dataset$metadata_2 <-
paste0(dataset[,14],"_",dataset[,15],"_",dataset[,16],"_",dataset[,18],"_",dataset[,19],"_",da
taset[,20],"_",dataset[,21],"_",dataset[,22],"_",dataset[,23])
486 dataset$metadata_2 <- gsub("__NA","",dataset$metadata_2)
487 # subset relevant data
488 dataset <- dataset[,c(1,2,24,13,25)]
489 # rename columns
490 if(dim(dataset)[2]==3){
491   colnames(dataset)<-c("DateTime",t,paste0(t, "_metadata"))
492 }else if(dim(dataset)[2]==2){
493   if(t %in% names(dataset)){print("")}
494   }else{dataset$dat <- NA; dat_names <- names(dataset); dataset <- dplyr::select(dataset,
DateTime, dat,dat_names[2])
495   colnames(dataset)<-c("DateTime",t,paste0(t, "_metadata"))
496   }
497 }else{
498   colnames(dataset) <- c("DateTime", paste0(t,"_1"), paste0(t,"_metadata_1"),paste0(t,"_2"),
paste0(t,"_metadata_2"))
499 }
500 # export data
501 year_range <- paste0(year(min(dataset$DateTime,na.rm = TRUE)), "_", year(max(dataset$DateTime,
na.rm = TRUE)))
502 # write.csv(dataset, paste0(ncei_dir,buoy,"/s_",buoy,
"_ncei_",t,"_concat_primary_secondary_metadata_",year_range,".csv"), row.names=FALSE)
503 # saveRDS(dataset, file =
paste0(ncei_dir,buoy,"/s_",buoy,"_ncei_",t,"_concat_primary_secondary_metadata_",year_range,".rds")
)
504 # 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),
512             "_", metadata1$payload_sensor, "_", t))
513             metadata1 <- metadata1[,7:ncol(metadata1)]
514             # handling different air pressure nomenclature
515             if(t == "air_pressure"){
516                 metadata1$join <- gsub("_air_pressure_at_sea_level", "_air_pressure", metadata1$join)
517                 dataset$payload_sensor <- gsub("_air_pressure_at_sea_level", "_air_pressure",
518                 dataset$payload_sensor)
519             }
520             # add primary sensor information
521             dataset <- dplyr::rename(dataset, join = payload_sensor)
522             dataset <- left_join(dataset, metadata1, by = "join")
523             # export data
524             year_range <- paste0(year(min(dataset$DateTime, na.rm = TRUE)), "_",
525             year(max(dataset$DateTime, na.rm = TRUE)))
526             # write.csv(dataset, paste0(ncei_dir, buoy, "/s_", buoy,
527             "_ncei_", t, "_primary_", year_range, ".csv"), row.names=FALSE)
528             # saveRDS(dataset, file =
529             paste0(ncei_dir, buoy, "/s_", buoy, "_ncei_", t, "_primary_", year_range, ".rds"))
530
531             # convert for concat
532             dataset$join <- gsub(paste0("_", t), "", dataset$join)
533             dataset$metadata <-
534             paste0(dataset[,3], "__", dataset[,4], "__", dataset[,5], "__", dataset[,7], "__", dataset[,8], "__", d
535             ataset[,9], "__", dataset[,10], "__", dataset[,11], "__", dataset[,12])
536             dataset$metadata <- gsub("__NA", "", dataset$metadata)
537             dataset <- dataset[,c(1,2,13)]
538             # rename columns
539             colnames(dataset) <- c("DateTime", t, paste0(t, "_metadata"))
540             rm(dataset1, metadata1)
541         }else{
542             dataset$payload_sensor <- "no information available"
543             colnames(dataset) <- c("DateTime", t, paste0(t, "_metadata"))
544         }
545         rm(dataset1)
546     }
547
548     # removing duplicated date/times added from secondary dataset (accounts for slightly different data
549     values caused by significant places)
550     dataset <- dataset[!duplicated(dataset$DateTime),]
551     # ordering the dataset by date and selecting unique values only
552     dataset <- dataset[order(dataset$DateTime),]
553     dataset <- unique(dataset)
554     # rename the rows to reflect unique data
555     row.names(dataset) <- 1:nrow(dataset)

```

```

547     # concat files
548     if (!exists("ncei_concat_stdmet")){
549         ncei_concat_stdmet <- dataset
550     }else{
551         ncei_concat_stdmet<-dplyr::full_join(ncei_concat_stdmet, dataset, by = "DateTime")
552         ncei_concat_stdmet <- ncei_concat_stdmet[order(ncei_concat_stdmet$DateTime),]
553     }
554     rm(dataset)
555     }else{print(paste("no data for ",t))}
556 } # end of concat data loop
557
558 dataset <- ncei_concat_stdmet
559 # # save dataset
560 # year_range <- paste0(year(dataset$DateTime[1]), "_", year(dataset$DateTime[nrow(dataset)]))
561 # saveRDS(dataset, file = paste0(ncei_dir,buoy,"/s_",buoy,"_ncei_concat_stdmet_",year_range,"_v1.rds"))
562
563 # remove '0' wind gust data when no wind speed is present (not flagged in orig files)
564 name_list <- names(dataset)
565 if("lat_1" %in% name_list){
566     dataset$lat_2 <- NULL; dataset$lat_metadata_2 <- NULL; dataset$lon_2 <- NULL; dataset$lon_metadata_2 <-
NULL
567     dataset <- dplyr::rename(dataset, lat = lat_1, lon = lon_1,lat_metadata = lat_metadata_1, lon_metadata =
lon_metadata_1 )
568 }
569 # account for erroneous data
570 if("wind_gust" %in% name_list){
571     library(stringr)
572     if(sum(str_count(name_list,"wind_gust"))>2){ # accounts for wind_gust and wind_gust_metadata
573         dataset$wind_gust_1 <- ifelse(dataset$wind_gust_1 == 0 & dataset$wind_speed_1 != 0,NA,
dataset$wind_gust_1)
574         dataset$wind_gust_2 <- ifelse(dataset$wind_gust_2 == 0 & dataset$wind_speed_2 != 0,NA,
dataset$wind_gust_2)
575     }else{dataset$wind_gust <- ifelse(dataset$wind_gust == 0 & dataset$wind_speed != 0,NA, dataset$wind_gust)}
576 }
577 assign("dataset", dataset, envir=parent.frame())
578
579 }else if(input == "spec"){
580     #-----
581     # spectral data -
582     #-----
583     # data_types_spec <- c("c11", "c11m", "alpha1", "alpha2", "r1", "r2", "C12", "C13", "C22", "C33",
584     # "Q12", "Q13", "gamma2", "gamma3", "phih", "rhq", "sensor_output") # c11 = spectral
energy, c11m = uncorrected spectral energy
585
586     # start_year <- 1979
587     # create data avail dir and folder
588     if (!file.exists(paste0(ncei_dir,buoy,"/data_availability"))){
589         {dir.create((paste0(ncei_dir,buoy,"/data_availability")))}
590     data_avail_dir <- paste0(ncei_dir,buoy,"/data_availability/s_")
591     print(paste0("Working on...",t))

```

```

591 library(lubridate)
592 library(tidyr)
593 # subset conditions for each variable - because NBDC loves duplication and format change for no reason
594 if(t == "c11"){ # I believe k is corrected
595   # see what's available:
596   files1 <- file_list[grepl(pattern = "spectral_density", file_list)]
597   files2 <- file_list[grepl(pattern = t, file_list)]
598   files3 <- file_list[grepl(pattern = "C11", file_list)]
599   files4 <- file_list[grepl(pattern = "c11", file_list)]
600   files <- c(files1, files2, files3, files4)
601   c11m <- files[grepl(pattern = "_c11m.csv", files)]
602   if(length(c11m) != 0){files <- files[!grepl(paste(c11m, collapse="|"), files)]}
603   df_files <- gsub(paste0(ascii_ncei_dir, buoy, "/", buoy, "_"), "", files)
604   df_files <- df_files[order(files)]
605   df_files <- unique(df_files)
606   df_files <- data.frame(df_files, stringsAsFactors = F)
607   df_files <- separate(df_files, df_files, into = c('date', 'file'), sep = "/", remove = TRUE)
608   write.csv(df_files, paste0(data_avail_dir, buoy, "_c11_nondir_spectral_available_files_all.csv"),
609     row.names=FALSE)
610   # used:
611   files1 <- file_list[grepl(pattern = "spectral_density", file_list)]
612   files2 <- file_list[grepl(pattern = t, file_list)]
613   files <- c(files1, files2)
614   files <- sort(files)
615   c11m <- files[grepl(pattern = "_c11m.csv", files)]
616   if(length(c11m) != 0){files <- files[!grepl(paste(c11m, collapse="|"), files)]}
617   c11i <- files[grepl(pattern = "_c11i.csv", files)]
618   if(length(c11i) != 0){files <- files[!grepl(paste(c11i, collapse="|"), files)] }
619   # payload_2 <- files[grepl(pattern = "_payload_2", files)]
620   df_files <- gsub(paste0(ascii_ncei_dir, buoy, "/", buoy, "_"), "", files)
621   df_files <- df_files[order(files)]
622   df_files <- unique(df_files)
623   df_files <- data.frame(df_files, stringsAsFactors = F)
624   df_files <- separate(df_files, df_files, into = c('date', 'file'), sep = "/", remove = TRUE)
625   write.csv(df_files, paste0(data_avail_dir, buoy, "_c11_nondir_spectral_available_files_USED.csv"),
626     row.names=FALSE)
627 } else if(t == "c11m"){ # I believe i is uncorrected
628   # see what's available:
629   files1 <- file_list[grepl(pattern = "c11", file_list)]
630   files2 <- file_list[grepl(pattern = "C11", file_list)]
631   files <- c(files1, files2)
632   c11 <- files[grepl(pattern = "_c11.csv", files)]
633   if(length(c11) != 0){files <- files[!grepl(paste(c11, collapse="|"), files)]}
634   df_files <- gsub(paste0(ascii_ncei_dir, buoy, "/", buoy, "_"), "", files)
635   df_files <- df_files[order(files)]
636   df_files <- unique(df_files)
637   df_files <- data.frame(df_files, stringsAsFactors = F)
638   df_files <- separate(df_files, df_files, into = c('date', 'file'), sep = "/", remove = TRUE)
639   write.csv(df_files, paste0(data_avail_dir, buoy,

```

```

639     "_c11m_uncorrected_nondir_spectral_available_files_all.csv"), row.names=FALSE)
640     # used:
641     files1 <- file_list[grepl(pattern = "c11", file_list)]
642     files2 <- file_list[grepl(pattern = "C11", file_list)]
643     files <- c(files1, files2)
644     files <- sort(files)
645     c11 <- files[grepl(pattern = "_c11.csv", files)]
646     if(length(c11) != 0){files <- files[!grepl(paste(c11, collapse="|"), files)]}
647     c11k <- files[grepl(pattern = "_c11_k.csv", files)]
648     if(length(c11k) != 0){files <- files[!grepl(paste(c11k, collapse="|"), files)]}
649     c11_i <- files[grepl(pattern = "_c11_i.csv", files)]
650     if(length(c11_i) != 0){files <- files[!grepl(paste(c11_i, collapse="|"), files)]}
651     df_files <- gsub(paste0(ascii_ncei_dir, buoy,"/",buoy,"_"), "",files)
652     df_files <- df_files[order(files)]
653     df_files <- unique(df_files)
654     df_files <- data.frame(df_files, stringsAsFactors = F)
655     df_files <- separate(df_files, df_files, into = c('date', 'file'),sep = "/", remove = TRUE)
656     write.csv(df_files, paste0(data_avail_dir,buoy,
657     "_c11m_uncorrected_nondir_spectral_available_files_USED.csv"), row.names=FALSE)
658
659 }else if(t == "alpha1"){
660     # see what's available:
661     files <- file_list[grepl(pattern = t, file_list)]
662     files <- sort(files)
663     df_files <- gsub(paste0(ascii_ncei_dir, buoy,"/",buoy,"_"), "",files)
664     df_files <- df_files[order(files)]
665     df_files <- unique(df_files)
666     df_files <- data.frame(df_files, stringsAsFactors = F)
667     df_files <- separate(df_files, df_files, into = c('date', 'file'),sep = "/", remove = TRUE)
668     write.csv(df_files, paste0(data_avail_dir,buoy, "_alpha1_available_files_all.csv"), row.names=FALSE)
669     # used:
670     files <- file_list[grepl(pattern = t, file_list)]
671     df_files <- gsub(paste0(ascii_ncei_dir, buoy,"/",buoy,"_"), "",files)
672     df_files <- df_files[order(files)]
673     df_files <- unique(df_files)
674     df_files <- data.frame(df_files, stringsAsFactors = F)
675     df_files <- separate(df_files, df_files, into = c('date', 'file'),sep = "/", remove = TRUE)
676     write.csv(df_files, paste0(data_avail_dir,buoy, "_alpha1_available_files_USED.csv"), row.names=FALSE)
677
678 }else if(t == "alpha2"){
679     # see what's available:
680     files <- file_list[grepl(pattern = t, file_list)]
681     files <- sort(files)
682     df_files <- gsub(paste0(ascii_ncei_dir, buoy,"/",buoy,"_"), "",files)
683     df_files <- df_files[order(files)]
684     df_files <- unique(df_files)
685     df_files <- data.frame(df_files, stringsAsFactors = F)
686     df_files <- separate(df_files, df_files, into = c('date', 'file'),sep = "/", remove = TRUE)
687     write.csv(df_files, paste0(data_avail_dir,buoy, "_alpha2_available_files_all.csv"), row.names=FALSE)
688     # used:

```

```

687 files <- file_list[grep(pattern = t, file_list)]
688 df_files <- gsub(paste0(ascii_ncei_dir, buoy, "/", buoy, "_"), "", files)
689 df_files <- df_files[order(files)]
690 df_files <- unique(df_files)
691 df_files <- data.frame(df_files, stringsAsFactors = F)
692 df_files <- separate(df_files, df_files, into = c('date', 'file'), sep = "/", remove = TRUE)
693 write.csv(df_files, paste0(data_avail_dir, buoy, "_alpha2_available_files_USED.csv"), row.names=FALSE)
694
695 }else if(t == "r1"){
696   # see what's available:
697   files <- file_list[grep(pattern = t, file_list)]
698   files <- sort(files)
699   df_files <- gsub(paste0(ascii_ncei_dir, buoy, "/", buoy, "_"), "", files)
700   df_files <- df_files[order(files)]
701   df_files <- unique(df_files)
702   df_files <- data.frame(df_files, stringsAsFactors = F)
703   df_files <- separate(df_files, df_files, into = c('date', 'file'), sep = "/", remove = TRUE)
704   write.csv(df_files, paste0(data_avail_dir, buoy, "_r1_available_files_all.csv"), row.names=FALSE)
705   # used:
706   files <- file_list[grep(pattern = t, file_list)]
707   df_files <- gsub(paste0(ascii_ncei_dir, buoy, "/", buoy, "_"), "", files)
708   df_files <- df_files[order(files)]
709   df_files <- unique(df_files)
710   df_files <- data.frame(df_files, stringsAsFactors = F)
711   df_files <- separate(df_files, df_files, into = c('date', 'file'), sep = "/", remove = TRUE)
712   write.csv(df_files, paste0(data_avail_dir, buoy, "_r1_available_files_USED.csv"), row.names=FALSE)
713
714 }else if(t == "r2"){
715   # see what's available:
716   files <- file_list[grep(pattern = t, file_list)]
717   files <- sort(files)
718   df_files <- gsub(paste0(ascii_ncei_dir, buoy, "/", buoy, "_"), "", files)
719   df_files <- df_files[order(files)]
720   df_files <- unique(df_files)
721   df_files <- data.frame(df_files, stringsAsFactors = F)
722   df_files <- separate(df_files, df_files, into = c('date', 'file'), sep = "/", remove = TRUE)
723   write.csv(df_files, paste0(data_avail_dir, buoy, "_r2_available_files_all.csv"), row.names=FALSE)
724   # used:
725   files <- file_list[grep(pattern = t, file_list)]
726   df_files <- gsub(paste0(ascii_ncei_dir, buoy, "/", buoy, "_"), "", files)
727   df_files <- df_files[order(files)]
728   df_files <- unique(df_files)
729   df_files <- data.frame(df_files, stringsAsFactors = F)
730   df_files <- separate(df_files, df_files, into = c('date', 'file'), sep = "/", remove = TRUE)
731   write.csv(df_files, paste0(data_avail_dir, buoy, "_r2_available_files_USED.csv"), row.names=FALSE)
732
733 }else{ # "C12", "C13", "C22", "C33", "Q12", "Q13", "gamma2", "gamma3", "phih", "rhq", "sensor_output"
734   # see what's available:
735   files <- file_list[grep(pattern = t, file_list)]
736   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 = ",",
762                               stringsAsFactors = FALSE)
763         # set date format
764         if(IsDate_dmy_hms(dataset[,1])==TRUE){dataset[,1] <- dmy_hms(dataset[,1]);
765           print("dmy_hms dates converted")}
766         if(IsDate_ymd_hms(dataset[,1])==TRUE){dataset[,1] <- ymd_hms(dataset[,1]);
767           print("ymd_hms dates converted")}
768         if(IsDate_mdy_hm(dataset[,1])==TRUE){dataset[,1] <- mdy_hm(dataset[,1]); print("mdy_hm
769           dates converted")}
770         dataset <- dplyr::select(dataset, -"lat")
771         dataset <- dplyr::select(dataset, -"lon")
772         dataset <- dplyr::rename(dataset, "DateTime" = "time")
773       }
774       # if the merged dataset does exist, append to it
775       if (exists("dataset")){
776         temp_dataset <- read.table(file, header=TRUE, na.strings = NA_strings, fill = T, sep =
777           ",", stringsAsFactors = FALSE)
778         # set date format
779         if(IsDate_dmy_hms(temp_dataset[,1])==TRUE){temp_dataset[,1] <-
780           dmy_hms(temp_dataset[,1]); print("dmy_hms dates converted")}
781         if(IsDate_ymd_hms(temp_dataset[,1])==TRUE){temp_dataset[,1] <-
782           ymd_hms(temp_dataset[,1]); print("ymd_hms dates converted")}
783         if(IsDate_mdy_hm(temp_dataset[,1])==TRUE){temp_dataset[,1] <-
784           mdy_hm(temp_dataset[,1]); print("mdy_hm dates converted")}
785         # temp_dataset[,1] <- ymd_hms(temp_dataset[,1]) # set date format
786         temp_dataset <- dplyr::select(temp_dataset, -"lat")

```



```

779         temp_dataset <- dplyr::select(temp_dataset, ~lon")
780         colnames(temp_dataset) <- names(dataset)
781         # merge datasets
782         dataset<-rbind(dataset, temp_dataset)
783         rm(temp_dataset)
784     }
785 }
786 # ordering the dataset by date and selecting unique values only
787 dataset <- dataset[order(dataset$DateTime),]
788 dataset <- unique(dataset)
789 # rename the rows to reflect unique data
790 row.names(dataset) <- 1:nrow(dataset)
791 # save individual datasets
792 library(lubridate)
793 year_range <- paste0(year(dataset$DateTime[1]), "_", year(dataset$DateTime[nrow(dataset)]))
794 # write.csv(dataset, paste0(concat_ind_dir,buoy, "_",t,"_",year_range,"_concat.csv"), row.names=FALSE)
795 # saveRDS(dataset, file = paste0(concat_ind_dir,buoy, "_",t,"_",year_range,"_concat.rds"))
796 # remove redundant rows
797 dataset <- unique(dataset)
798 library(dplyr)
799 dataset <- dplyr::select(dataset,~"qc_flag")
800 dataset <- dplyr::select(dataset,~"release_flag")
801 dataset <- dataset[rowSums(is.na(dataset)) != ncol(dataset)-1, ]
802 assign("dataset", dataset, envir=parent.frame())
803
804 }else{
805
806     colNames_spec_new <- c("DateTime","lat","lon","0.0200", "0.0325", "0.0375", "0.0425", "0.0475", "0.0525",
807                           "0.0575", "0.0625", "0.0675", "0.0725", "0.0775", "0.0825", "0.0875", "0.0925",
808                           "0.1000", "0.1100",
809                           "0.1200", "0.1300", "0.1400", "0.1500", "0.1600", "0.1700", "0.1800", "0.1900",
810                           "0.2000", "0.2100",
811                           "0.2200", "0.2300", "0.2400", "0.2500", "0.2600", "0.2700", "0.2800", "0.2900",
812                           "0.3000", "0.3100",
813                           "0.3200", "0.3300", "0.3400", "0.3500", "0.3650", "0.3850", "0.4050", "0.4250",
814                           "0.4450", "0.4650",
815                           "0.4850")
816
817     colNames_spec_old <-
818     c("DateTime","lat","lon","0.0100","0.0200","0.0300","0.0400","0.0500","0.0600","0.0700","0.0800","0.0900",
819       ,"0.1000","0.1100","0.1200",
820
821       "0.1300","0.1400","0.1500","0.1600","0.1700","0.1800","0.1900","0.2000","0.2100",""
822       0.2200",
823
824       "0.2300","0.2400","0.2500","0.2600","0.2700","0.2800","0.2900","0.3000","0.3100",""
825       0.3200",
826       "0.3300","0.3400","0.3500","0.3600","0.3700","0.3800","0.3900","0.4000")
827
828     dataset_spec_new <- data.frame(matrix(NA, nrow = 0, ncol = length(colNames_spec_new)))
829     colnames(dataset_spec_new) <- colNames_spec_new

```

```

819 dataset_spec_new$DateTime <- lubridate::ymd_hms(dataset_spec_new$DateTime)
820
821 dataset_spec_old <- data.frame(matrix(NA, nrow = 0, ncol = length(colNames_spec_old)))
822 colnames(dataset_spec_old) <- colNames_spec_old
823 dataset_spec_old$DateTime <- lubridate::ymd_hms(dataset_spec_old$DateTime)
824
825 count <- 0
826
827 for (file in files){
828   print(file)
829   # load dataset
830   dataset <- read.table(file, header=TRUE, na.strings = NA_strings, fill = T, sep = ",",
831     stringsAsFactors = FALSE)
832   # set date format
833   if(IsDate_dmy_hms(dataset[,1])==TRUE){dataset[,1] <- dmy_hms(dataset[,1]); print("dmy_hms dates
834     converted")}
835   if(IsDate_ymd_hms(dataset[,1])==TRUE){dataset[,1] <- ymd_hms(dataset[,1]); print("ymd_hms dates
836     converted")}
837   if(IsDate_mdy_hm(dataset[,1])==TRUE){dataset[,1] <- mdy_hm(dataset[,1]); print("mdy_hm dates
838     converted")}
839   library(lubridate)
840   # remove the X from the column headers
841   colnames(dataset) <- sub("X", "", colnames(dataset))
842   header <- names(dataset)
843   header <- read_header(header)
844   colnames(dataset) <- header
845   if("time" %in% names(dataset)){dataset <- dplyr::rename(dataset,"DateTime" = "time")}
846   if("lat" %in% names(dataset)){
847     print("lat and lon present")
848   }else{
849     dataset$lat <- NA; dataset$lon <- NA
850     print("adding lat and lon")
851   }
852
853   # find datasets
854   dataset_list <- ls(pattern = "dataset_spec_")
855   dataset_list <- dataset_list[dataset_list %in% c("dataset_spec_new", "dataset_spec_old") == TRUE]
856   print(dataset_list)
857   # build datasets
858
859   for(matchable in dataset_list){
860     print(matchable)
861     dat <- get(matchable)
862
863     if(exists("dataset")){
864       ndbc_freq <- names(dat)
865       dat_names <- names(dataset)
866       setdiff(ndbc_freq, dat_names)
867       print(unique(dat_names %in% ndbc_freq))
868       # test if columns match

```

```

865         if(any(unique(dat_names %in% ndbc_freq)==FALSE)){
866             print(paste0(matchable, " didn't match"))
867         }else{
868             tryCatch({
869                 library(plyr)
870                 dat <- rbind.fill(dat,dataset)
871                 print(paste0("added to: ",matchable))
872                 assign(matchable,dat)
873                 rm(dataset)
874             }, error = function(e) {
875                 print("dataset doesn't match")
876             })
877         }
878     }
879     rm(dat)
880 }
881 if(exists("dataset")){
882     dataset_list <- ls(pattern = "dataset_spec_")
883     dataset_list <- dataset_list[dataset_list %in% c("dataset_spec_new","dataset_spec_old") ==
FALSE]
884     print(dataset_list)
885     # build datasets
886
887     for(matchable in dataset_list){
888         print(matchable)
889         dat <- get(matchable)
890
891         if(exists("dataset")){
892             ndbc_freq <- names(dat)
893             dat_names <- names(dataset)
894             setdiff(ndbc_freq, dat_names)
895             print(unique(dat_names %in% ndbc_freq))
896             # test if columns match
897             if(any(unique(dat_names %in% ndbc_freq)==FALSE)){
898                 print(paste0(matchable, " didn't match"))
899             }else{
900                 tryCatch({
901                     library(plyr)
902                     dat <- rbind.fill(dat,dataset)
903                     print(paste0("added to: ",matchable))
904                     assign(matchable,dat)
905                     rm(dataset)
906                 }, error = function(e) {
907                     print("dataset doesn't match")
908                 })
909             }
910         }
911         rm(dat)
912     }
913 }

```

```

914         if(exists("dataset")){
915             count <- count + 1
916             dataset_spec <- data.frame(matrix(NA, nrow = 0, ncol = dim(dataset)[2]))
917             dataset_spec<-rbind(dataset_spec, dataset)
918             new_name <- paste0("dataset_spec_",count)
919             assign(new_name, dataset_spec)
920             print(paste0("data added to NEW DF:: ",new_name))
921             rm(dataset, dataset_spec)
922         }
923     }
924
925     # export from function
926     dat_list <- ls(pattern = "dataset_spec")
927     print(dat_list)
928     for(d in dat_list){
929         df <- get(d)
930         if(dim(df)[1] > 0){
931             # remove rows with no data
932             completeFun <- function(data, desiredCols) {completeVec <- complete.cases(data[,
desiredCols]); return(data[completeVec, ])}
933             # remove empty rows across df
934             df <- df[rowSums(is.na(df)) != ncol(df),]
935             # remove rows with no DateTime
936             df <- df[!is.na(df$DateTime),]
937             # remove rows with multiple version of missing data
938             df <- completeFun(df,1:3) # no DateTime,lat,lon
939             for(desiredCount in 3:5){df <- df[rowSums(is.na(df)) != ncol(df)-desiredCount,]} # no data
in data columns
940             # ordering the dataset by date and selecting unique values only
941             if(dim(df)[1]>0){
942                 df <- df[order(df$DateTime),]
943                 df <- unique(df)
944                 # rename the rows to reflect unique data
945                 row.names(df) <- 1:nrow(df)
946             }
947         }
948         # export from function
949         if(dim(df)[1] !=0){assign(d, df, envir=parent.frame())}
950     }
951     rm(d)
952 }
953 }
954 }
955 }

```



US Army Corps  
of Engineers®

verify\_netcdf\_3.R

```

1 verify_netcdf_3 <- function(buoys = "list of buoys", data_dir = "data_dir"){
2
3     ##-----
4     ## validates netCDF metadata with NDBC google spreadsheet metadata as captured by NDBC, DiNapoli,2020
5     ## Hall, Candice
6     ##-----
7
8     ## Actions:
9     ## 1. Sets data locations
10    ## 2. Read in NDBC and NCEI data if not already loaded in global environ
11    ## 3. Loads and formats the station specific metadata spreadsheet that was concatenated in step 1a above.
12    ## 4. Selects the station specific NCEI stdmet data and removes NA and duplicate data
13    ## 5. Concatenates and verifies wave metadata with NDBC metadata spreadsheets
14    ## 6. Concatenates and verifies other dual meteorological sensor metadata with the NDBC metadata spreadsheets
15    ## 7. Performs housekeeping tasks to remove excess sensor information labels that were included within the
16    metadata extraction process.
17    ## 8. Removes excess hull info and verify sensor height information
18    ## 9. Double-checks that NCEI netCDF extracted metadata fields no longer contain 'no available information'.
19    ## 10. Filters for unique date/time data and reorders the datasets by date/time, before reordering the
20    datasets to columns structures that match NDBC stdmet datasets.
21    ## 11. The verified stdmet with newly verified metadata, as well as all of the individual spectral wave
22    variable datasets, are saved in an 's_buoy#_ncei_ALL_verified.Rdata' container file within buoy station
23    specific folders
24
25    #-----
26    #-----
27    # library(NCmisc)
28    # list.functions.in.file(rstudioapi::getSourceEditorContext()$path, alphabetic = TRUE)
29    ## libraries required
30    # install.packages("tidyverse", lib="/p/home/candice/Rlibs/")
31    # install.packages("readxl", lib="/p/home/candice/Rlibs/") # dependent on tidyverse
32    # install.packages("lubridate", lib="/p/home/candice/Rlibs/")
33    # install.packages("stringr", lib="/p/home/candice/Rlibs/")
34    # install.packages("data.table", lib="/p/home/candice/Rlibs/")
35    # install.packages("dplyr", lib="/p/home/candice/Rlibs/")
36    # install.packages("tidyr", lib="/p/home/candice/Rlibs/")
37
38    # load libraries (HPC run)
39    library(readxl, lib="/p/home/candice/Rlibs/") # dependent of tidyverse
40    library(backports, lib="/p/home/candice/Rlibs/") # tidyverse
41    library(withr, lib="/p/home/candice/Rlibs/") # tidyverse
42    library(cli, lib="/p/home/candice/Rlibs/") # tidyverse
43    library(tzdb, lib="/p/home/candice/Rlibs/") # tidyverse
44    library(readr, lib="/p/home/candice/Rlibs/") # tidyverse
45    library(rstudioapi, lib="/p/home/candice/Rlibs/") # tidyverse
46    library(tidyverse, lib="/p/home/candice/Rlibs/")
47    library(readxl, lib="/p/home/candice/Rlibs/") # dependent on tidyverse
48    library(lubridate, lib="/p/home/candice/Rlibs/")
49    library(stringr, lib="/p/home/candice/Rlibs/")
50    library(data.table, lib="/p/home/candice/Rlibs/")

```

```

47 library(crayon, lib="/p/home/candice/Rlibs/") # dplyr
48 library(pillar, lib="/p/home/candice/Rlibs/") # dplyr
49 library(dplyr, lib="/p/home/candice/Rlibs/")
50 library(tidyr, lib="/p/home/candice/Rlibs/")
51 # library(stats, lib="/p/home/candice/Rlibs/")
52
53 # # load libraries (local run)
54 # library(tidyverse)
55 # library(readxl) # dependent on tidyverse
56 # library(lubridate)
57 # library(stringr)
58 # library(data.table)
59 # library(dplyr)
60 # library(tidyr)
61 # library(stats)
62
63
64 ##-----
65 ## set paths
66 ##-----
67 # drive <- "E:/Candice/"
68 # drive <- "/p/work/candice/"
69
70 # data_dir <- paste0(drive, "projects/WaveTrends/annual_runs/data/")
71 setwd(data_dir)
72
73 # set input directories
74 input_dir <- paste0(data_dir, "concat_data/ncei/")
75
76 # metadata sheets
77 metadata_dir <- paste0(data_dir, "NDBC_metadata_sheets/")
78 # if (!file.exists(metadata_dir)) {dir.create(metadata_dir)}
79
80 ##-----
81 ## set buoy stations for downloading (for stand-alone use)
82 ##-----
83 # list_ndbc <- read.csv("NDBC_buoys.csv", header = TRUE)
84 # list_ndbc <- dplyr::filter(list_ndbc, list_ndbc$owner == "NDBC")
85 # list_ndbc_buoy <- as.character(list_ndbc$station)
86 # buoys <- list_ndbc_buoy
87 # rm(list_ndbc, list_ndbc_buoy)
88 # print(buoys)
89
90 ##-----
91 ## set sensor heights
92 ##-----
93 ## NDBC published sensor heights (August 3, 2016): https://www.ndbc.noaa.gov/bht.shtml
94 ## At what heights are the sensors located on moored buoys?
95 ## Meteorological sensors are normally located at the ten meter level for the 10-meter
96 ## and the 12-meter discus buoys and are at a nominal height of five meters for the

```

```

97     ## 3-meter discus buoys and the 6-meter NOMAD buoys. However, barometers are located
98     ## inside the hull at the water level.
99     ## Sea surface temperature sensors are located at a depth of 1.5 meters for 10-m and 12-m
100    ## buoys and at 1 meter for all 3-m and 6-m moored buoys. Sea surface temperature sensors
101    ## on NDBC buoys are located near one meter below the water line but they vary by hull type.
102    ## Current hull configurations for water temperature sensors are: for 2.4- and 3-meter hulls
103    ## at 0.7 meters; for 6-meter hulls at 0.8 meters; for 10-meter hulls at 1.1 meters; and
104    ## for 12-meter hulls at 0.9 meters. Historically, water temperature sensors for 3-meter
105    ## "foam" hulls were at 0.75 meters and 1.8-meter hulls were at 0.35 meters.
106
107    # wind                # sst                # ref
108    wind_12m <- 10;       sst_12m <- 0.9    # NDBC, 2016, https://www.ndbc.noaa.gov/bht.shtml
109    wind_10m <- 10;       sst_10m <- 1.1    # NDBC, 2016, https://www.ndbc.noaa.gov/bht.shtml
110    wind_6m  <- 5;        sst_6m  <- 0.8    # NDBC, 2016, https://www.ndbc.noaa.gov/bht.shtml
111    wind_5m  <- 5;        sst_5m  <- 0.7    # No information, Rodney Riley, NDBC Engineer (pers. comms.
112    03/02/2021)
113    wind_3m  <- 5;        sst_3m  <- 0.7    # NDBC, 2016, https://www.ndbc.noaa.gov/bht.shtml
114    wind_elb <- 5.2;      sst_elb <- 1.8    # 1985 Smith
115    wind_lnb <- 10;       sst_lnb <- 0.9    # Large navigational buoy: NDBC, 1992,
116    ftp://ftp.library.noaa.gov/noaa\_documents.lib/NWS/National\_Data\_Buoy\_Center/technical\_bulletin/June-1992\_vol-18
117    .pdf
118    wind_2_8m <- NA;      sst_2_8m <- NA    # No information, Rodney Riley, NDBC Engineer (pers. comms.
119    03/02/2021)
120    wind_2_6m <- NA;      sst_2_6m <- NA    # No information, Rodney Riley, NDBC Engineer (pers. comms.
121    03/02/2021)
122    wind_2_4m <- 3.3;     sst_2_4m <- 0.7    # NDBC 'BUOY COMPARISONS Final - with sensor heights.pdf'
123    wind_2_3m <- 3.2;     sst_2_3m <- 1.3    # 2017 Bouchard et al., NDBC 'BUOY COMPARISONS Final - with sensor
124    heights.pdf'
125    wind_2_1m <- 3.2;     sst_2_1m <- 1.1    # NDBC, 2021,
126    https://www.ndbc.noaa.gov/station\_page.php?station=45001
127    wind_1_8m <- 2.1;     sst_1_8m <- 0.4    # 2008 Crout et al.
128
129    for(buoy in buoys){ #
130
131        # start writing to an output file
132        print(paste0("Starting on ",buoy))
133
134        # # start writing to an output file
135        sink(paste0(data_dir,"3_ncei_metadata_verification_",buoy,"_",Sys.Date(),".txt"))
136        print(paste0("Starting on...",buoy))
137        #
138        #-----
139        ## read in data if not loaded in global environ
140        #-----
141
142        if(file.exists(paste0(input_dir,buoy,"/s_",buoy,"_ncei_ALL.RData"))){
143
144            print("Loading datasets")
145            load(paste0(input_dir,buoy,"/s_",buoy,"_ncei_ALL.RData"))
146

```



```

140 # load spreadsheet metadata
141 # /p/work/candice/projects/WaveTrends/data/NDBC_metadata_sheets
142 met_dat <- list.files(metadata_dir,full.names = TRUE); met_dat <-
met_dat[grepl("-Meta-Data-",met_dat)]
143 met_dat <- met_dat[grepl(buoy,met_dat)]
144 metadata <- read_excel(met_dat, col_names = TRUE, skip = 2, na = c("?", "N/A"))
145 if("AIO Met Sensor" %in% names(metadata)){
146     metadata <- dplyr::select(metadata, `Date Start`,Dep:`AIO Met Sensor`)
147 }else{metadata <- dplyr::select(metadata, `Date Start`,Dep:Sensor); metadata$`AIO Met Sensor` <-
NA}
148 # remove column name spaces and caps, replace wave NA, merge wave metadata columns and remove
redundant columns
149 metadata <- dplyr::rename(metadata, Date_Start = `Date Start`,depth = Dep,mooring = Mooring,hull
= Hull,payload = Payload, waveSys = WaveSys,waveSensor = Sensor, windSensor = `AIO Met Sensor`)
150 for(rn in 1:nrow(metadata)){if(is.na(metadata$waveSys[rn])){metadata$waveSys[rn] <-
"unknown"};if(is.na(metadata$waveSensor[rn])){metadata$waveSensor[rn] <- "unknown"}}
151 metadata$waveSys_Sensor <- paste0(metadata$waveSys,"; ",metadata$waveSensor)
152 # reformat date
153 metadata$DateTime <- ymd_hms(paste0(metadata$Date_Start," 00:00:00"))
154 metadata$waveSys <- NULL; metadata$sensor <- NULL; metadata$Date_Start <- NULL
155 # fill in missing data
156 metadata <- fill(metadata, depth,mooring,hull,payload, .direction = "up")
157 # reorder - spreadsheets don't have the same column order (?)
158 metadata <- dplyr::select(metadata, DateTime,depth,hull,mooring,payload,waveSys_Sensor)
159 # handle unknowns
160 index_hull <- which(metadata$hull=="unknown");
if(length(index_hull)>0){metadata$hull[index_hull] <- metadata$hull[index_hull+1]}
161
162 # set as data.table for date merge
163 metadata <- data.table(metadata)
164 setkey(metadata, DateTime)
165
166 # find df's to update
167 stdmet_ls <- ls(pattern = "ncei_stdmet")
168 stdmet_ls
169
170 for(df in stdmet_ls){
171     dat <- get(df)
172
173     print(paste0("working on ",df))
174     print(Sys.time())
175
176     # removing any character NA's
177     dat[, 2:ncol(dat)][dat[, 2:ncol(dat)]=="NA"] <- NA
178
179     # remove duplicate wave data
180     if("significant_wave_height_1" %in% names(dat) & "significant_wave_height_2" %in%
names(dat)){
181         print("working on redundant wave data")
182         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 =

```

```

207 TRUE)==0){dat$significant_wave_height_2 <- NULL;
dat$significant_wave_height_metadata_2 <- NULL;
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)

```

```

237 dat_merge_wave$wave_payload_info <- NULL; dat_merge_wave$waveSys_Sensor <- NULL
238 dat_merge_hull <-
239   setkey(data.table(dplyr::select(dat_merge,DateTime,depth,mooring,hull,payload)), DateTime)
240   # merge with original data
241   dat <- data.table(dat)
242   setkey(dat, DateTime)
243   dat <- dat_merge_hull[dat, roll = "nearest"]
244   dat$significant_wave_height_metadata <- NULL; dat$dominant_wave_period_metadata <- NULL;
245   dat$average_wave_period_metadata <- NULL; dat$mean_wave_direction_metadata <- NULL
246   dat <- left_join(dat,dat_merge_wave, by = "DateTime")
247   rm(dat_merge, dat_merge_hull, dat_merge_wave, dat_sub)
248   # set wave metadata
249   dat$significant_wave_height_metadata <- dat$wave_payload_info_sensor
250   dat$dominant_wave_period_metadata <- NA; dat$average_wave_period_metadata <- NA;
251   dat$mean_wave_direction_metadata <- NA
252   dat$wave_payload_info_sensor <- NULL
253
254   # concat other sensor metadata
255   print("concatenating other sensors metadata")
256   # wind data
257   col_ls <- names(dat)[grep("wind_speed_metadata",names(dat))]
258   if(length(col_ls) == 2){
259     dat$wind_direction_metadata_1 <- NA; dat$wind_gust_metadata_1 <- NA
260     dat$wind_direction_metadata_2 <- NA; dat$wind_gust_metadata_2 <- NA
261   }else{dat$wind_direction_metadata <- NA; dat$wind_gust_metadata <- NA}
262
263   # remove excess sensor info
264   print("removing excess sensor info")
265   print(Sys.time())
266   if(grepl("ncei",df,ignore.case = TRUE)){
267     if("wind_speed_metadata_1" %in% names(dat)){
268       dat$wind_speed_metadata_1 <- gsub("_wind_speed","",dat$wind_speed_metadata_1);
269       dat$wind_speed_metadata_2 <- gsub("_wind_speed","",dat$wind_speed_metadata_2)
270     }else{dat$wind_speed_metadata <- gsub("_wind_speed","",dat$wind_speed_metadata)}
271
272     if("air_pressure_at_sea_level_metadata_1" %in% names(dat)){
273       dat$air_pressure_at_sea_level_metadata_1 <-
274       gsub("_air_pressure_at_sea_level","",dat$air_pressure_at_sea_level_metadata_1);dat
275       $air_pressure_at_sea_level_metadata_2 <-
276       gsub("_air_pressure_at_sea_level","",dat$air_pressure_at_sea_level_metadata_2)
277     }else{dat$air_pressure_at_sea_level_metadata <-
278     gsub("_air_pressure_at_sea_level","",dat$air_pressure_at_sea_level_metadata)}
279
280     if("air_temperature_metadata_1" %in% names(dat)){
281       dat$air_temperature_metadata_1 <-
282       gsub("air_temperature__","",dat$air_temperature_metadata_1);dat$air_temperature
283       metadata_2 <- gsub("air_temperature__","",dat$air_temperature_metadata_2)
284     }else{dat$air_temperature_metadata_1 <-
285     gsub("air_temperature__","",dat$air_temperature_metadata_1);dat$air_temperature_metadata_2
286     <- gsub("air_temperature__","",dat$air_temperature_metadata_2)

```

```

275     dat$air_temperature_metadata_1 <-
      gsub("___", "", dat$air_temperature_metadata_1); dat$air_temperature_metadata_2
276     <- gsub("___", "", dat$air_temperature_metadata_2)
      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}

```

```

306   if(hull_type == "ELB"){wind_height <- wind_elb; sst_height <- sst_elb}; if(hull_type
307   == 2.8){wind_height <- wind_2_8m; sst_height <- sst_2_8m}
308   if(hull_type == "LNB"){wind_height <- wind_lnb; sst_height <- sst_lnb}; if(hull_type
309   == 2.8){wind_height <- wind_2_8m; sst_height <- sst_2_8m}
310   if(hull_type == 2.6){wind_height <- wind_2_6m; sst_height <- sst_2_6m}; if(hull_type
311   == 2.4){wind_height <- wind_2_4m; sst_height <- sst_2_4m}
312   if(hull_type == 2.3){wind_height <- wind_2_3m; sst_height <- sst_2_3m}; if(hull_type
313   == 1.8){wind_height <- wind_1_8m; sst_height <- sst_1_8m}
314   if(hull_type == 2.1){wind_height <- wind_2_1m; sst_height <- sst_2_1m}
315
316   # air pressure
317   if("air_pressure_at_sea_level_metadata_1" %in% names(dat)){
318     if(!is.na(dat$air_pressure_at_sea_level_metadata_1[rn])){
319       rm_pattern <-
320       stringr::str_extract(dat$air_pressure_at_sea_level_metadata_1[rn],
321       paste0(".{0,2}",hull_id,".{0,7}"))
322       if(!is.na(rm_pattern)){dat$air_pressure_at_sea_level_metadata_1[rn] <-
323       gsub(rm_pattern,"",dat$air_pressure_at_sea_level_metadata_1[rn])}
324       rm(rm_pattern)
325     }
326   }
327   if("air_pressure_at_sea_level_metadata_2" %in% names(dat)){
328     if(!is.na(dat$air_pressure_at_sea_level_metadata_2[rn])){
329       rm_pattern <-
330       stringr::str_extract(dat$air_pressure_at_sea_level_metadata_2[rn],
331       paste0(".{0,2}",hull_id,".{0,7}"))
332       if(!is.na(rm_pattern)){dat$air_pressure_at_sea_level_metadata_2[rn] <-
333       gsub(rm_pattern,"",dat$air_pressure_at_sea_level_metadata_2[rn])}
334       rm(rm_pattern)
335     }
336   }
337   if("air_pressure_at_sea_level_metadata" %in% names(dat)){
338     if(!is.na(dat$air_pressure_at_sea_level_metadata[rn])){
339       rm_pattern <-
340       stringr::str_extract(dat$air_pressure_at_sea_level_metadata[rn],
341       paste0(".{0,2}",hull_id,".{0,7}"))
342       if(!is.na(rm_pattern)){dat$air_pressure_at_sea_level_metadata[rn] <-
343       gsub(rm_pattern,"",dat$air_pressure_at_sea_level_metadata[rn])}
344       rm(rm_pattern)
345     }
346   }
347
348   # air temp
349   if("air_temperature_metadata_1" %in% names(dat)){
350     if(!is.na(dat$air_temperature_metadata_1[rn])){
351       rm_pattern <- stringr::str_extract(dat$air_temperature_metadata_1[rn],
352       paste0(".{0,2}",hull_id,".{0,7}"))
353       if(!is.na(rm_pattern)){dat$air_temperature_metadata_1[rn] <-
354       gsub(rm_pattern,"",dat$air_temperature_metadata_1[rn])}
355       rm(rm_pattern)
356     }
357   }

```

```

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],
345       paste0("{0,2}", hull_id, "{0,7}"))
346     if(!is.na(rm_pattern)){dat$air_temperature_metadata_2[rn] <-
347       gsub(rm_pattern, "", dat$air_temperature_metadata_2[rn])}
348     rm(rm_pattern)
349   }
350 }
351 if("air_temperature_metadata" %in% names(dat)){
352   if(!is.na(dat$air_temperature_metadata[rn])){
353     rm_pattern <- stringr::str_extract(dat$air_temperature_metadata[rn],
354       paste0("{0,2}", hull_id, "{0,7}"))
355     if(!is.na(rm_pattern)){dat$air_temperature_metadata[rn] <-
356       gsub(rm_pattern, "", dat$air_temperature_metadata[rn])}
357     rm(rm_pattern)
358   }
359 }
360 # sea temp
361 if("sea_surface_temperature_metadata_1" %in% names(dat)){
362   if(!is.na(dat$sea_surface_temperature_metadata_1[rn])){
363     rm_pattern <-
364     stringr::str_extract(dat$sea_surface_temperature_metadata_1[rn],
365       paste0("{0,2}", hull_id, "{0,7}"))
366     if(!is.na(rm_pattern)){dat$sea_surface_temperature_metadata_1[rn] <-
367       gsub(rm_pattern, "", dat$sea_surface_temperature_metadata_1[rn])}
368     rm(rm_pattern)
369     dat$sea_surface_temperature_metadata_1[rn] <- gsub("no available height
370     data", paste0(sst_height, "m"), dat$sea_surface_temperature_metadata_1[rn])
371   }
372 }
373 if("sea_surface_temperature_metadata_2" %in% names(dat)){
374   if(!is.na(dat$sea_surface_temperature_metadata_2[rn])){
375     rm_pattern <-
376     stringr::str_extract(dat$sea_surface_temperature_metadata_2[rn],
377       paste0("{0,2}", hull_id, "{0,7}"))
378     if(!is.na(rm_pattern)){dat$sea_surface_temperature_metadata_2[rn] <-
379       gsub(rm_pattern, "", dat$sea_surface_temperature_metadata_2[rn])}
380     rm(rm_pattern)
381     dat$sea_surface_temperature_metadata_2[rn] <- gsub("no available height
382     data", paste0(sst_height, "m"), dat$sea_surface_temperature_metadata_2[rn])
383   }
384 }
385 if("sea_surface_temperature_metadata" %in% names(dat)){
386   if(!is.na(dat$sea_surface_temperature_metadata[rn])){
387     rm_pattern <- stringr::str_extract(dat$sea_surface_temperature_metadata[rn],
388       paste0("{0,2}", hull_id, "{0,7}"))
389     if(!is.na(rm_pattern)){dat$sea_surface_temperature_metadata[rn] <-
390       gsub(rm_pattern, "", dat$sea_surface_temperature_metadata[rn])}

```

```

377         rm(rm_pattern)
378         dat$sea_surface_temperature_metadata[rn] <- gsub("no available height data",
379             paste0(sst_height, "m"), dat$sea_surface_temperature_metadata[rn])
380     }
381     # dew pt temp
382     if("dew_point_temperature_metadata_1" %in% names(dat)){
383         if(!is.na(dat$dew_point_temperature_metadata_1[rn])){
384             rm_pattern <- stringr::str_extract(dat$dew_point_temperature_metadata_1[rn],
385                 paste0(".{0,2}", hull_id, ".{0,7}"))
386             if(!is.na(rm_pattern)){dat$dew_point_temperature_metadata_1[rn] <-
387                 gsub(rm_pattern, "", dat$dew_point_temperature_metadata_1[rn])}
388             rm(rm_pattern)
389         }
390     }
391     if("dew_point_temperature_metadata_2" %in% names(dat)){
392         if(!is.na(dat$dew_point_temperature_metadata_2[rn])){
393             rm_pattern <- stringr::str_extract(dat$dew_point_temperature_metadata_2[rn],
394                 paste0(".{0,2}", hull_id, ".{0,7}"))
395             if(!is.na(rm_pattern)){dat$dew_point_temperature_metadata_2[rn] <-
396                 gsub(rm_pattern, "", dat$dew_point_temperature_metadata_2[rn])}
397             rm(rm_pattern)
398         }
399     }
400     if("dew_point_temperature_metadata" %in% names(dat)){
401         if(!is.na(dat$dew_point_temperature_metadata[rn])){
402             rm_pattern <- stringr::str_extract(dat$dew_point_temperature_metadata[rn],
403                 paste0(".{0,2}", hull_id, ".{0,7}"))
404             if(!is.na(rm_pattern)){dat$dew_point_temperature_metadata[rn] <-
405                 gsub(rm_pattern, "", dat$dew_point_temperature_metadata[rn])}
406             rm(rm_pattern)
407         }
408     }
409     # wind sensor metadata
410     if("wind_speed_metadata_1" %in% names(dat)){
411         if(!is.na(dat$wind_speed_metadata_1[rn])){
412             rm_pattern <- stringr::str_extract(dat$wind_speed_metadata_1[rn],
413                 paste0(".{0,2}", hull_id, ".{0,7}"))
414             if(!is.na(rm_pattern)){dat$wind_speed_metadata_1[rn] <-
415                 gsub(rm_pattern, "", dat$wind_speed_metadata_1[rn])}
416             rm(rm_pattern)
417         }
418     }
419     if("wind_speed_metadata_2" %in% names(dat)){
420         if(!is.na(dat$wind_speed_metadata_2[rn])){
421             rm_pattern <- stringr::str_extract(dat$wind_speed_metadata_2[rn],
422                 paste0(".{0,2}", hull_id, ".{0,7}"))
423             if(!is.na(rm_pattern)){dat$wind_speed_metadata_2[rn] <-
424                 gsub(rm_pattern, "", dat$wind_speed_metadata_2[rn])}
425             rm(rm_pattern)
426         }
427     }

```



```

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],
421         paste0(".{0,2}", "hull_id", ".{0,7}"))
422       if(!is.na(rm_pattern)){dat$wind_speed_metadata[rn] <-
423         gsub(rm_pattern, "", dat$wind_speed_metadata[rn])}
424       rm(rm_pattern)
425     }
426   }
427   # remove unknown from redundant sensor heights
428   # find sensor and associated heights
429   rm(wind_s1, wind_s1_height, wind_s2, wind_s2_height)
430   if("wind_speed_metadata_1" %in% names(dat)){
431     if(!is.na(dat$wind_speed_metadata_1[rn])){
432       wind_sensor <- unlist(strsplit(dat$wind_speed_metadata_1[rn], "__"));
433       wind_s1 <- wind_sensor[length(wind_sensor)-1]; if(length(wind_s1)==0){wind_s1
434         <- dat$wind_speed_metadata_1[rn]}
435       wind_sensor_height <- unlist(strsplit(dat$wind_speed_metadata_1[rn], "__"));
436       #wind_s1_height <- wind_sensor_height[length(wind_sensor_height)]
437       if(wind_sensor_height != wind_s1){wind_s1_height <-
438         wind_sensor_height[length(wind_sensor_height)]}else{wind_s1 <- NA;
439         wind_s1_height <- NA}
440       if(!is.na(wind_s1_height) & !is.na(wind_s1)){
441         if(wind_s1_height == "no available height data" & wind_s1 == "no
442           available manufacturer data"){wind_s1_height <-
443             wind_height; dat$wind_speed_metadata_1[rn] <- gsub("no available
444             height data", paste0(wind_s1_height, "m"),
445             dat$wind_speed_metadata_1[rn])}
446       }
447       rm(wind_sensor, wind_sensor_height)
448     }else{wind_s1 <- NA; wind_s1_height <- NA}
449   }
450   if("wind_speed_metadata_2" %in% names(dat)){
451     if(!is.na(dat$wind_speed_metadata_2[rn])){
452       wind_sensor <- unlist(strsplit(dat$wind_speed_metadata_2[rn], "__"));
453       wind_s2 <- wind_sensor[length(wind_sensor)-1]; if(length(wind_s2)==0){wind_s2
454         <- dat$wind_speed_metadata_2[rn]}
455       wind_sensor_height <- unlist(strsplit(dat$wind_speed_metadata_2[rn], "__"));
456       if(wind_sensor_height != wind_s2){wind_s2_height <-
457         wind_sensor_height[length(wind_sensor_height)]}else{wind_s2 <- NA;
458         wind_s2_height <- NA}
459       if(!is.na(wind_s2_height) & !is.na(wind_s2)){
460         if(wind_s2_height == "no available height data" & wind_s2 == "no
461           available manufacturer data"){wind_s2_height <-
462             wind_height; dat$wind_speed_metadata_2[rn] <- gsub("no available
463             height data", paste0(wind_s2_height, "m"),
464             dat$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$air_pressure_at_sea_level_metadata_1[rn])){
471         AP_1 <-
472         unlist(strsplit(dat$air_pressure_at_sea_level_metadata_1[rn],
473           "__"));AP_s1 <- AP_1[length(AP_1)-1]
474         if(grepl(wind_s1, AP_s1, ignore.case = TRUE, fixed =
475           FALSE)){dat$air_pressure_at_sea_level_metadata_1[rn] <- gsub("no
476           available height
477           data",wind_s1_height,dat$air_pressure_at_sea_level_metadata_1[rn])}
478       }
479       # sensor 2
480       if(!is.na(dat$air_pressure_at_sea_level_metadata_2[rn])){
481         AP_2 <-
482         unlist(strsplit(dat$air_pressure_at_sea_level_metadata_2[rn],
483           "__"));AP_s2 <- AP_2[length(AP_2)-1]
484         if(grepl(wind_s1, AP_s2, ignore.case = TRUE, fixed =
485           FALSE)){dat$air_pressure_at_sea_level_metadata_2[rn] <- gsub("no
486           available height
487           data",wind_s1_height,dat$air_pressure_at_sea_level_metadata_2[rn])}
488       }
489     }
490   }

```

```

480     if("air_pressure_at_sea_level_metadata" %in% names(dat)){
481         # sensor 1
482         if(!is.na(dat$air_pressure_at_sea_level_metadata[rn])){
483             AP_1 <-
484                 unlist(strsplit(dat$air_pressure_at_sea_level_metadata[rn],
485                                 "__"));AP_s1 <- AP_1[length(AP_1)-1]
486             if(exists("wind_s1")){if(grepl(wind_s1, AP_s1, ignore.case = TRUE,
487                                           fixed = FALSE)){dat$air_pressure_at_sea_level_metadata[rn] <-
488                 gsub("no available height
489                     data",wind_s1_height,dat$air_pressure_at_sea_level_metadata[rn])}}
490         }
491     }
492
493     # air temperature
494     if("air_temperature_metadata_1" %in% names(dat)){
495         # sensor 1
496         if(!is.na(dat$air_temperature_metadata_1[rn])){
497             AP_1 <- unlist(strsplit(dat$air_temperature_metadata_1[rn],
498                                     "__"));AP_s1 <- AP_1[length(AP_1)-1]
499             if(grepl(wind_s1, AP_s1, ignore.case = TRUE, fixed =
500                     FALSE)){dat$air_temperature_metadata_1[rn] <- gsub("no available
501                     height data",wind_s1_height,dat$air_temperature_metadata_1[rn])}
502         }
503         # sensor 2
504         if(!is.na(dat$air_temperature_metadata_2[rn])){
505             AP_2 <- unlist(strsplit(dat$air_temperature_metadata_2[rn],
506                                     "__"));AP_s2 <- AP_2[length(AP_2)-1]
507             if(grepl(wind_s1, AP_s2, ignore.case = TRUE, fixed =
508                     FALSE)){dat$air_temperature_metadata_2[rn] <- gsub("no available
509                     height data",wind_s1_height,dat$air_temperature_metadata_2[rn])}
510         }
511     }
512     if(buoy == 41001 & year(dat$DateTime[rn])==2019){
513         if(grepl("no available description data__no available manufacturer
514                 data__no available height
515                 data",dat$air_temperature_metadata_1[rn])){
516             if(is.na(dat$air_temperature_metadata_2[rn]) &
517                 !is.na(dat$air_pressure_at_sea_level_2[rn])){
518                 dat$air_temperature_metadata_1[rn] <-
519                     dat$air_pressure_at_sea_level_metadata_2[rn]
520                 dat$air_temperature_metadata_1[rn] <-
521                     gsub("barometer","air_temperature_sensor",
522                         dat$air_temperature_metadata_1[rn])
523             }
524         }
525     }
526 }
527
528 if("air_temperature_metadata" %in% names(dat)){
529     # sensor 1
530     if(!is.na(dat$air_temperature_metadata[rn])){
531         AP_1 <- unlist(strsplit(dat$air_temperature_metadata[rn],

```

```

513     "___");AP_s1 <- AP_1[length(AP_1)-1]
        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],
522         "___"));AP_s1 <- AP_1[length(AP_1)-1]
        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],
527         "___"));AP_s2 <- AP_2[length(AP_2)-1]
        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$dew_point_temperature_metadata_2[rn]) &
532           !is.na(dat$air_pressure_at_sea_level_2[rn])){
533           dat$dew_point_temperature_metadata_1[rn] <-
            dat$air_temperature_metadata_1[rn]
            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],
542         "___"));AP_s1 <- AP_1[length(AP_1)-1]
        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$air_pressure_at_sea_level_metadata_1[rn],
553             "_"));AP_s1 <- AP_1[length(AP_1)-1]
          if(grepl(wind_s2, AP_s1, ignore.case = TRUE, fixed =
554             FALSE)){dat$air_pressure_at_sea_level_metadata_1[rn] <-
            gsub("no available height
555             data",wind_s2_height,dat$air_pressure_at_sea_level_metadata_1[r
556             n])}
557       }
558       # sensor 2
559       if(!is.na(dat$air_pressure_at_sea_level_metadata_2[rn])){
560         AP_2 <-
          unlist(strsplit(dat$air_pressure_at_sea_level_metadata_2[rn],
561             "_"));AP_s2 <- AP_2[length(AP_2)-1]
          if(grepl(wind_s2, AP_s2, ignore.case = TRUE, fixed =
562             FALSE)){dat$air_pressure_at_sea_level_metadata_2[rn] <-
            gsub("no available height
563             data",wind_s2_height,dat$air_pressure_at_sea_level_metadata_2[r
564             n])}
565       }
566     }
567   }
568   # air temperature
569   if("air_temperature_metadata_1" %in% names(dat)){
570     # sensor 1
571

```

```

572         if(!is.na(dat$air_temperature_metadata_1[rn])){
573             AP_1 <- unlist(strsplit(dat$air_temperature_metadata_1[rn],
574                                   "___"));AP_s1 <- AP_1[length(AP_1)-1]
575             if(grepl(wind_s2, AP_s1, ignore.case = TRUE, fixed =
576                     FALSE)){dat$air_temperature_metadata_1[rn] <- gsub("no
577                               available height
578                               data",wind_s2_height,dat$air_temperature_metadata_1[rn])}
579         }
580         # sensor 2
581         if(!is.na(dat$air_temperature_metadata_2[rn])){
582             AP_2 <- unlist(strsplit(dat$air_temperature_metadata_2[rn],
583                                   "___"));AP_s2 <- AP_2[length(AP_2)-1]
584             if(grepl(wind_s2, AP_s2, ignore.case = TRUE, fixed =
585                     FALSE)){dat$air_temperature_metadata_2[rn] <- gsub("no
586                               available height
587                               data",wind_s2_height,dat$air_temperature_metadata_2[rn])}
588         }
589     }
590     if("air_temperature_metadata" %in% names(dat)){
591         # sensor 1
592         if(!is.na(dat$air_temperature_metadata[rn])){
593             AP_1 <- unlist(strsplit(dat$air_temperature_metadata[rn],
594                                   "___"));AP_s1 <- AP_1[length(AP_1)-1]
595             if(exists("wind_s2")){if(grepl(wind_s2, AP_s1, ignore.case =
596                                         TRUE, fixed = FALSE)){dat$air_temperature_metadata[rn] <-
597                               gsub("no available height
598                               data",wind_s2_height,dat$air_temperature_metadata[rn])}}
599         }
600     }
601     # dew point temperature
602     if("dew_point_temperature_metadata_1" %in% names(dat)){
603         # sensor 1
604         if(!is.na(dat$dew_point_temperature_metadata_1[rn])){
605             AP_1 <-
606             unlist(strsplit(dat$dew_point_temperature_metadata_1[rn],
607                               "___"));AP_s1 <- AP_1[length(AP_1)-1]
608             if(grepl(wind_s2, AP_s1, ignore.case = TRUE, fixed =
609                     FALSE)){dat$dew_point_temperature_metadata_1[rn] <- gsub("no
610                               available height
611                               data",wind_s2_height,dat$dew_point_temperature_metadata_1[rn])}
612         }
613         # sensor 2
614         if(!is.na(dat$dew_point_temperature_metadata_2[rn])){
615             AP_2 <-
616             unlist(strsplit(dat$dew_point_temperature_metadata_2[rn],
617                               "___"));AP_s2 <- AP_2[length(AP_2)-1]
618             if(grepl(wind_s2, AP_s2, ignore.case = TRUE, fixed =
619                     FALSE)){dat$dew_point_temperature_metadata_2[rn] <- gsub("no
620                               available height

```

```

601         data", wind_s2_height, dat$dew_point_temperature_metadata_2[rn]))}
602     }
603     if("dew_point_temperature_metadata" %in% names(dat)){
604         # sensor 1
605         if(!is.na(dat$dew_point_temperature_metadata[rn])){
606             AP_1 <-
607             unlist(strsplit(dat$dew_point_temperature_metadata[rn],
608                             " ")); AP_s1 <- AP_1[length(AP_1)-1]
609             if(exists("wind_s2")){if(grepl(wind_s2, AP_s1, ignore.case =
610             TRUE, fixed = FALSE)){dat$dew_point_temperature_metadata[rn]
611             <- gsub("no available height
612             data", wind_s2_height, dat$dew_point_temperature_metadata[rn])}}
613         }
614     }
615 }
616 # cycle through one more time looking for 'no available height data'
617 # wind
618 if("wind_speed_metadata_1" %in% names(dat)){
619     if(!is.na(dat$wind_speed_metadata_1[rn])){
620         if(grepl("no available height data",
621         dat$wind_speed_metadata_1[rn])==TRUE){
622             dat$wind_speed_metadata_1[rn] <- gsub("no available height
623             data", paste0(wind_height, "m"), dat$wind_speed_metadata_1[rn])
624         }
625     }
626     if(!is.na(dat$wind_speed_metadata_2[rn])){
627         if(grepl("no available height data",
628         dat$wind_speed_metadata_2[rn])==TRUE){
629             dat$wind_speed_metadata_2[rn] <- gsub("no available height
630             data", paste0(wind_height, "m"), dat$wind_speed_metadata_2[rn])
631         }
632     }
633 }
634 if("wind_speed_metadata" %in% names(dat)){
635     if(!is.na(dat$wind_speed_metadata[rn])){
636         if(grepl("no available height data",
637         dat$wind_speed_metadata[rn])==TRUE){
638             dat$wind_speed_metadata[rn] <- gsub("no available height

```

```

639         dat$air_pressure_at_sea_level_metadata_1[rn])==TRUE){
            dat$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"),

```



```

672         dat$air_temperature_metadata[rn])
673     }
674 }
675 # dew point temperature
676 if("dew_point_temperature_metadata_1" %in% names(dat)){
677     if(!is.na(dat$dew_point_temperature_metadata_1[rn])){
678         if(grepl("no available height data",
679             dat$dew_point_temperature_metadata_1[rn])==TRUE){
680             dat$dew_point_temperature_metadata_1[rn] <- gsub("no available
681             height data", paste0(wind_height,"m"),
682             dat$dew_point_temperature_metadata_1[rn])
683         }
684     }
685     if(!is.na(dat$dew_point_temperature_metadata_2[rn])){
686         if(grepl("no available height data",
687             dat$dew_point_temperature_metadata_2[rn])==TRUE){
688             dat$dew_point_temperature_metadata_2[rn] <- gsub("no available
689             height data", paste0(wind_height,"m"),
690             dat$dew_point_temperature_metadata_2[rn])
691         }
692     }
693 }
694 if("dew_point_temperature_metadata" %in% names(dat)){
695     if(!is.na(dat$dew_point_temperature_metadata[rn])){
696         if(grepl("no available height data",
697             dat$dew_point_temperature_metadata[rn])==TRUE){
698             dat$dew_point_temperature_metadata[rn] <- gsub("no available
699             height data", paste0(wind_height,"m"),
700             dat$dew_point_temperature_metadata[rn])
701         }
702     }
703 }
704 # copy metadata across wave variables
705 if(!is.na(dat$dominant_wave_period[rn])){dat$dominant_wave_period_metadata[rn] <-
706 dat$significant_wave_height_metadata[rn]}
707 if(!is.na(dat$average_wave_period[rn])){dat$average_wave_period_metadata[rn] <-
708 dat$significant_wave_height_metadata[rn]}
709 if(!is.na(dat$mean_wave_direction[rn])){dat$mean_wave_direction_metadata[rn] <-
710 dat$significant_wave_height_metadata[rn]}
711 if("wind_direction_1" %in% names(dat)){
712     if(!is.na(dat$wind_speed_metadata_1[rn])){dat$wind_direction_metadata_1[rn] <-
713     dat$wind_speed_metadata_1[rn]; dat$wind_gust_metadata_1[rn] <-
714     dat$wind_speed_metadata_1[rn]}
715     if(!is.na(dat$wind_speed_metadata_2[rn])){dat$wind_direction_metadata_2[rn] <-
716     dat$wind_speed_metadata_2[rn]; dat$wind_gust_metadata_2[rn] <-
717     dat$wind_speed_metadata_2[rn]}
718 }
719 } # end of row checks

```

```

705 # ordering the df by date and selecting unique values only
706 dat <- dat[order(dat$DateTime),]
707 # rename the rows to reflect unique data
708 row.names(dat) <- 1:nrow(dat)
709
710 if(ncol(dat)== 45){
711     print("ncei data format - dual sst")
712     dat <- dplyr::select(dat, DateTime, depth, mooring, hull, payload, lat,
713         lat_metadata, lon, lon_metadata,
714         wind_direction_1, wind_direction_metadata_1, wind_direction_2,
715         wind_direction_metadata_2,
716         wind_speed_1, wind_speed_metadata_1, wind_speed_2,
717         wind_speed_metadata_2,
718         wind_gust_1, wind_gust_metadata_1, wind_gust_2, wind_gust_metadata_2,
719         significant_wave_height, significant_wave_height_metadata,
720         dominant_wave_period, dominant_wave_period_metadata,
721         average_wave_period, average_wave_period_metadata,
722         mean_wave_direction, mean_wave_direction_metadata,
723         air_pressure_at_sea_level_1, air_pressure_at_sea_level_metadata_1,
724         air_pressure_at_sea_level_2, air_pressure_at_sea_level_metadata_2,
725         air_temperature_1, air_temperature_metadata_1, air_temperature_2,
726         air_temperature_metadata_2,
727         sea_surface_temperature_1,
728         sea_surface_temperature_metadata_1, sea_surface_temperature_2,
729         sea_surface_temperature_metadata_2,
730         dew_point_temperature_1, dew_point_temperature_metadata_1,
731         dew_point_temperature_2, dew_point_temperature_metadata_2)
732 }else if(ncol(dat)== 43){
733     print("ncei data format - single sst or single dew point temperature")
734     if("sea_surface_temperature" %in% names(dat)){
735         dat <- dplyr::select(dat, DateTime, depth, mooring, hull, payload, lat,
736             lat_metadata, lon, lon_metadata,
737             wind_direction_1, wind_direction_metadata_1,
738             wind_direction_2, wind_direction_metadata_2,
739             wind_speed_1, wind_speed_metadata_1, wind_speed_2,
740             wind_speed_metadata_2,
741             wind_gust_1, wind_gust_metadata_1, wind_gust_2,
742             wind_gust_metadata_2,
743             significant_wave_height,
744             significant_wave_height_metadata,
745             dominant_wave_period, dominant_wave_period_metadata,
746             average_wave_period, average_wave_period_metadata,
747             mean_wave_direction, mean_wave_direction_metadata,
748             air_pressure_at_sea_level_1,
749             air_pressure_at_sea_level_metadata_1,
750             air_pressure_at_sea_level_2,
751             air_pressure_at_sea_level_metadata_2,
752             air_temperature_1, air_temperature_metadata_1,
753             air_temperature_2, air_temperature_metadata_2,
754             sea_surface_temperature,

```

```

734         sea_surface_temperature_metadata,
        dew_point_temperature_1,
        dew_point_temperature_metadata_1,
        dew_point_temperature_2,
        dew_point_temperature_metadata_2)
735     }else if("dew_point_temperature" %in% names(dat)){
736         dat <- dplyr::select(dat, DateTime, depth, mooring, hull, payload, lat,
        lat_metadata, lon, lon_metadata,
737         wind_direction_1, wind_direction_metadata_1,
        wind_direction_2, wind_direction_metadata_2,
738         wind_speed_1, wind_speed_metadata_1, wind_speed_2,
        wind_speed_metadata_2,
739         wind_gust_1, wind_gust_metadata_1, wind_gust_2,
        wind_gust_metadata_2,
740         significant_wave_height,
        significant_wave_height_metadata,
741         dominant_wave_period, dominant_wave_period_metadata,
        average_wave_period, average_wave_period_metadata,
742         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,
743         air_temperature_1, air_temperature_metadata_1,
        air_temperature_2, air_temperature_metadata_2,
744         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)
745     }else{print("error: dim = 43 but dat names don't match")}
746 }else if(ncol(dat)== 41){
747     print("ncei data format - single dew point")
748     dat <- dplyr::select(dat, DateTime, depth, mooring, hull, payload, lat,
        lat_metadata, lon, lon_metadata,
750     wind_direction_1, wind_direction_metadata_1, wind_direction_2,
        wind_direction_metadata_2,
751     wind_speed_1, wind_speed_metadata_1, wind_speed_2,
        wind_speed_metadata_2,
752     wind_gust_1, wind_gust_metadata_1, wind_gust_2, wind_gust_metadata_2,
753     significant_wave_height, significant_wave_height_metadata,
        dominant_wave_period, dominant_wave_period_metadata,
754     average_wave_period, average_wave_period_metadata,
        mean_wave_direction, mean_wave_direction_metadata,
755     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,
756     air_temperature_1, air_temperature_metadata_1, air_temperature_2,
        air_temperature_metadata_2,
757     sea_surface_temperature, sea_surface_temperature_metadata,
        dew_point_temperature, dew_point_temperature_metadata)
758 }else if(ncol(dat)== 31){

```

```

760         print("ncei data format - single sensors for all")
761         dat <- dplyr::select(dat, DateTime, depth, mooring, hull, payload, lat,
762                             lat_metadata, lon, lon_metadata,
763                             wind_direction, wind_direction_metadata, wind_speed,
764                             wind_speed_metadata, wind_gust, wind_gust_metadata,
765                             significant_wave_height, significant_wave_height_metadata,
766                             dominant_wave_period, dominant_wave_period_metadata,
767                             average_wave_period, average_wave_period_metadata,
768                             mean_wave_direction, mean_wave_direction_metadata,
769                             air_pressure_at_sea_level, air_pressure_at_sea_level_metadata,
770                             air_temperature, air_temperature_metadata,
771                             sea_surface_temperature, sea_surface_temperature_metadata,
772                             dew_point_temperature, dew_point_temperature_metadata)
773     }
774
775     # save data
776     dfv <- paste0(df, "_verified")
777     assign(dfv, dat)
778     rm(dat, AP_1, AP_2, AP_s1, AP_s2, hull_type, index_hull, wind_s1, wind_s1_height, wind_s2,
779         wind_s2_height, metadata)
780 }
781
782 #-----
783 # exporting GeoCleaned datasets
784 #-----
785
786 print("Exporting new verified data")
787 ncei_list <- ls(pattern = "verified")
788 print(paste0("NCEI datasets :", ncei_list))
789
790 # export and save verified dataset
791 for(g in ncei_list){
792     if(grepl("_verified", g) == TRUE){
793         print(g)
794         # write.table(get(g), paste0(input_dir, buoy, "/", g, ".csv"), row.names=FALSE, col.names
795         = TRUE, sep = ",")
796         # saveRDS(get(g), paste0(input_dir, buoy, "/", g, ".rds"))
797     }
798 }
799 # export to RData
800 ncei_list <- ls(pattern = buoy)
801 save(list = ncei_list, file = paste0(input_dir, buoy, "/s_", buoy, "_ncei_ALL_verified.RData"))
802
803 # # Stop writing to the file
804 sink()
805 print("test complete")
806
807 print(paste0("finishing metadata verification for buoy ", buoy))
808 }else{print("no new data for this buoy")}
809 }

```



US Army Corps  
of Engineers®

# geoClean\_data\_4.R

```

1 geoClean_data_4 <- function(buoys = "list of buoys", data_dir = "data_dir"){
2
3     ##-----
4     ## Compares and geographically cleans data before verifying metadata by combining NDBC web files and NCEI
5     netcdf files
6     ## Hall, Candice
7     ##-----
8
9     ## Actions:
10    ## 1. Sets data locations
11    ## 2. Read in NDBC and NCEI data if not already loaded in global environ
12    ## 3. Compares the NDBC and NCEI sourced data by matching the datasets on 'nearest' date and time.
13    ## 4. This step geographically quality controls each dataset by removing GPS positions and
14    ##      associated data that are not within a pre-selected radius of the NDBC station watch circles.
15    ## 5. This step assigns verified metadata to the NDBC stdmet datasets.
16    ## 6. If desired, the final section of code within the 'geoClean_data_4.R' script produces statistical
17    ##      comparisons and plots of the NDBC and NCEI datasets. A 'switch_set' turns this functionality
18    ##      on and off during stand-alone use of this script. To activate this functionality in HPC runs, the
19    ##      'geoClean_data_4.R' will require and additional 'switch-set' input value to function.
20
21    ##-----
22    ##-----
23    ## load libraries (local run)
24    library(lubridate)
25    library(plyr)
26    library(dplyr)
27    library(gridExtra)
28    library(data.table)
29    # library(oce)
30    library(naniar)
31    library(tidyverse)
32    library(broom)
33    library(openair) # polarplots
34    library(plotly)
35    library(magrittr)
36    library(tidyr)
37    # library(grid)
38    # library(devtools)
39    library(lsr)
40    library(stringr)
41    library(RColorBrewer)
42    library(viridis)
43    library(colorRamps)
44    library(ggplot2)
45    library(ggmap)
46    library(maps)
47    library(mapdata)
48    library(modeest)
49    library(tibble)

```

```

50 ## load libraries (HPC run)
51 # library(lubridate, lib="/p/home/candice/Rlibs/")
52 # library(plyr, lib="/p/home/candice/Rlibs/")
53 # library(dplyr, lib="/p/home/candice/Rlibs/")
54 # library(gridExtra, lib="/p/home/candice/Rlibs/")
55 # library(data.table, lib="/p/home/candice/Rlibs/")
56 # # library(oce, lib="/p/home/candice/Rlibs/")
57 # library(naniar, lib="/p/home/candice/Rlibs/")
58 # library(tidyverse, lib="/p/home/candice/Rlibs/")
59 # library(broom, lib="/p/home/candice/Rlibs/")
60 # library(openair, lib="/p/home/candice/Rlibs/") # polarplots
61 # library(plotly, lib="/p/home/candice/Rlibs/")
62 # library(magrittr, lib="/p/home/candice/Rlibs/")
63 # library(tidyr, lib="/p/home/candice/Rlibs/")
64 # # library(grid, lib="/p/home/candice/Rlibs/")
65 # # library(devtools, lib="/p/home/candice/Rlibs/")
66 # library(lsr, lib="/p/home/candice/Rlibs/")
67 # library(stringr, lib="/p/home/candice/Rlibs/")
68 # library(RColorBrewer, lib="/p/home/candice/Rlibs/")
69 # library(viridis, lib="/p/home/candice/Rlibs/")
70 # library(colorRamps, lib="/p/home/candice/Rlibs/")
71 # library(ggplot2, lib="/p/home/candice/Rlibs/")
72 # library(ggmap, lib="/p/home/candice/Rlibs/")
73 # library(maps, lib="/p/home/candice/Rlibs/")
74 # library(mapdata, lib="/p/home/candice/Rlibs/")
75 # library(modeest, lib="/p/home/candice/Rlibs/")
76 # library(tibble, lib="/p/home/candice/Rlibs/")
77
78 #-----
79 #-----
80
81 # set switch for script actions
82 # 1 == plot/stats for geoClean dataset creation, 2 == don't plot/stats
83 switch_set <- 2
84 # GPS buffer
85 GPS_buffer <- 1 # allows for a 1 degree radius
86
87 ##-----
88 ## set paths
89 ##-----
90 # drive <- "E:/Candice/"
91 # drive <- "/p/work/candice/"
92 # data_dir <- paste0(drive, "projects/WaveTrends/annual_runs/data/")
93 setwd(data_dir)
94
95 # set input directories
96 input_dir <- paste0(data_dir, "concat_data/")
97 # ndbc
98 ndbc_dir <- paste0(input_dir, "ndbc/")
99 # ncei

```

```

100 ncei_dir <- paste0(input_dir,"ncei/")
101
102 # set new output directories for datasets, stats and figures
103 if (!file.exists(paste0(data_dir,"geoClean_data/"))) {dir.create((paste0(data_dir,"geoClean_data/")))}
104 clean_data_dir <- paste0(data_dir,"geoClean_data/")
105
106 # set new output directories for datasets, stats and figures
107 if (!file.exists(paste0(clean_data_dir,"results/"))) {dir.create((paste0(clean_data_dir,"results/")))}
108 out_dir <- paste0(clean_data_dir,"results/")
109 if (!file.exists(paste0(clean_data_dir,"data/"))) {dir.create((paste0(clean_data_dir,"data/")))}
110 clean_dir <- paste0(clean_data_dir,"data/")
111 if (!file.exists(paste0(out_dir,"stats/"))) {dir.create((paste0(out_dir,"stats/")))}
112 stats_dir <- paste0(out_dir,"stats/")
113 if (!file.exists(paste0(out_dir,"figures/"))) {dir.create((paste0(out_dir,"figures/")))}
114 fig_dir <- paste0(out_dir,"figures/")
115 if (!file.exists(paste0(out_dir,"GPS_plots/"))) {dir.create((paste0(out_dir,"GPS_plots/")))}
116 gps_dir <- paste0(out_dir,"GPS_plots/")
117
118 ##-----
119 ## set set parameters common to all plots
120 ##-----
121 xlab = "Date" # label for x axis
122 width = 2000+2000 # width of exported plot
123 height = 1500+1500 # height of exported plot
124 res = 300
125 # res2 = 500
126 # set plot parameters
127 width1 = 1000
128 height1 = 700
129 par1 = c(5,5,4,4)
130
131 # # colors for each buoy
132 plot_colors <- viridis(n=6)
133 color_ndbc_raw <- plot_colors[1]
134 color_ndbc_orig <- plot_colors[5] #"#440154FF" # purple
135 color_ndbc_recalc <- plot_colors[3] # "#3B528BFF" # "5 = #FDE725FF" # yellow
136 color_chl_calc <- plot_colors[4] # "#21908CFF" # "#21908CFF" # green
137 color_WIS <- "red" # "#5DC863FF"
138
139 # wind/wave polar plots
140 colour1 <- viridis(n=4)
141 cols1 <- c(colour1[4], colour1[3], colour1[2], colour1[1])
142 colour2 <- viridis(n=5)
143 cols2 <- c(colour2[5], colour2[4], colour2[3], colour2[2], colour2[1])
144 type = "l"
145 pch = "."
146 lwd = 0.5
147 cex = 0.5
148 # set parameters
149 Delta <- '\U0394'

```



```

150 degree <- '\U00B0'
151 # my.grid function formats
152 my.format <- "%m-%d-%Y" # "%m-%Y" (long datasets) or "%m-%d-%Y" (short datasets)
153 my.period <- "weeks" # "months" (long datasets) or "weeks" (short datasets)
154 # my grid function for plots
155 my.grid <-function(dataset, my.period = "year", my.format = "%Y"){
156   grid(nx=NA, ny=NULL)
157   abline(v=axis.POSIXct(1, at=seq(min(dataset[1,1]), max(dataset[nrow(dataset),1]),
158                                   by= my.period), format=my.format),
159          col = "lightgray", lty = "dotted", lwd = par("lwd"))
160 }
161 # function to capitalize string
162 simpleCap <- function(x) {
163   s <- strsplit(x, " ")[[1]]
164   paste(toupper(substring(s, 1,1)), substring(s, 2),
165         sep="", collapse=" ")
166 }
167
168 # The choice of significance level at which you reject H0 is arbitrary.
169 sig <- 0.01
170
171 ##-----
172 ## set buoy stations for downloading (for stand-alone use)
173 ##-----
174
175 # list_ndbc <- read.csv(paste0(data_dir,"NDBC_buoys.csv"),header = TRUE)
176 # list_ndbc <- dplyr::filter(list_ndbc, list_ndbc$owner == "NDBC")
177 # list_ndbc_buoy <- as.character(list_ndbc$station)
178 #
179 # buoys <- list_ndbc_buoy
180 # rm(list_ndbc, list_ndbc_buoy)
181 # print(buoys)
182 ##-----
183
184 for(buoy in buoys){
185   # buoy <- buoys[1]
186
187   # start writing to an output file
188   print(paste0("Starting on ",buoy))
189
190   #-----
191   ## read in data if not loaded in global environ
192   #-----
193
194   if(file.exists(paste0(ndbc_dir,buoy,"/s_",buoy,"_ndbc_ALL.RData"))){
195
196     # # start writing to an output file
197     # sink(paste0(data_dir,"3_geoClean_data_",buoy,"_",Sys.Date(),".txt"))
198     # print(paste0("Starting on... ",buoy))
199

```

```

200     print("Loading datasets")
201
202     load(paste0(ndbc_dir,buoy,"/s_",buoy,"_ndbc_ALL.RData"))
203     load(paste0(ncei_dir,buoy,"/s_",buoy,"_ncei_ALL_verified.RData"))
204
205     #-----
206     # check to remove empty dataframes, if any
207     #-----
208     print("checking for empty dataframes")
209     dat_list <- ls(pattern = buoy)
210     print(dat_list)
211     for(d in dat_list){
212         dat <- get(d)
213         dat <- data.frame(dat, stringsAsFactors = FALSE)
214         if(dim(dat)[1]<= 1){
215             rm(list = ls()[grep1(d, ls())])
216             print(paste0("removing empty df: ",d))
217         }else{print(paste0(d, " is not empty"))}
218         rm(dat)
219     }
220
221     #-----
222     # isolating common metadata from verified ncei df
223     #-----
224     print("Handling verified ncei metadata")
225     # rename non verified data
226     ncei_list <- ls(pattern = "_ncei_stdmet")
227     ncei_list <- ncei_list[!ncei_list %in% grep(paste0("_verified", collapse = "|"), ncei_list, value = T)]
228     print(ncei_list)
229     dat <- get(ncei_list)
230     name_net <- paste0("s_",buoy,"_ncei_sdmet_netcdf")
231     assign(name_net, dat)
232
233     print("isolating common metadata from verified ncei df")
234     metadata <- ls(pattern = "_verified")
235     print(metadata)
236     df <- data.frame(get(metadata), stringsAsFactors = FALSE)
237     drop_cols <- c("DateTime", "lat", "lon", "depth", "mooring", "hull", "payload")
238     meta_df <- dplyr::select(df,all_of(drop_cols))
239     df <- df[, !names(df) %in% c("depth", "mooring", "hull", "payload")]
240     # save data
241     assign(ncei_list, df)
242     metadata <- gsub("_stdmet_verified", "_station_metadata", metadata)
243     assign(metadata, meta_df)
244     # remove original data
245     rm(list = ls(pattern = "_verified"))
246     rm(ncei_list,dat,df, meta_df, metadata, drop_cols)
247
248     #-----
249     ## create data count table

```

```

250 #-----
251 data_count <- t(data.frame("Original_count", "Original_start", "Original_end",
252                           "Comparison_count", "Comparison_start", "Comparison_end",
253                           "GeoCleaned_count", "GeoCleaned_start", "GeoCleaned_end"))
254
255 # rm(Original_count, Original_start,
256     Original_end, Comparison_count, Comparison_start, Comparison_end, GeoCleaned_count, GeoCleaned_start, GeoClea
257     ned_end)
258 data_count <- data.frame(data_count, stringsAsFactors = FALSE)
259
260 #-----
261 ## geoclean matching data
262 #-----
263 print("GeoCleaning matching datasets")
264 ncei_list <- ls(pattern = "_ncei_")
265 ncei_list <- ncei_list[!ncei_list %in% grep(paste0("sensor_output", collapse = "|"), ncei_list, value
266     = T)]
267 print(ncei_list)
268 colNames <- c("DateTime", "lat", "lon")
269 ncei_positions <- data.frame(matrix(NA, nrow = 0, ncol = length(colNames)))
270 colnames(ncei_positions) <- colNames
271 ncei_positions$DateTime <- lubridate::ymd_hms(ncei_positions$DateTime)
272
273 # selection matching positions
274 for(n in ncei_list){
275   print(n)
276   dat <- get(n)
277   column_names <- names(dat)
278   if("lat" %in% column_names){
279     dat <- dplyr::select(dat, DateTime, lat, lon)
280     dat$lat <- as.numeric(dat$lat)
281     dat$lon <- as.numeric(dat$lon)
282     ncei_positions <- rbind(ncei_positions, dat)
283   }else{print(paste0("no GPS data in ", n))}
284   rm(dat)
285 }
286 # find unique
287 ncei_positions <- unique(ncei_positions)
288 # ordering the df by date and selecting unique values only
289 ncei_positions <- ncei_positions[order(ncei_positions$DateTime),]
290 ncei_positions <- unique(ncei_positions)
291 # rename the rows to reflect unique data
292 row.names(ncei_positions) <- 1:nrow(ncei_positions)
293 # remove bad data
294 ncei_positions <- ncei_positions[complete.cases(ncei_positions),]
295
296 # remove bad netcdf GPS positions and correct ncei datafiles
297 library(modeest)
298 # using a sorted table of value occurrences to find most common lat/lon
299 lat_tail <- tail(names(sort(table(ncei_positions$lat))), 1)

```

```

297 lon_tail <- tail(names(sort(table(ncei_positions$lon))),1)
298 print(paste0("sorted table method - lat: ", lat_tail, "; lon: ", lon_tail))
299 # check positions in ndbc bulk
300 if(abs(range(ncei_positions$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$lon, na.rm =
TRUE)[2]) > GPS_buffer){
301   print(paste0("range - lat: ", range(ncei_positions$lat, na.rm = TRUE)[1]," ",
range(ncei_positions$lat, na.rm = TRUE)[2]))
302   print(paste0("range - lon: ", range(ncei_positions$lon, na.rm = TRUE)[1]," ",
range(ncei_positions$lon, na.rm = TRUE)[2]))
303   ncei_positions <- dplyr::filter(ncei_positions, lat >=
as.numeric(lat_tail)-as.numeric(GPS_buffer) & lat <= as.numeric(lat_tail)+as.numeric(GPS_buffer))
304   ncei_positions <- dplyr::filter(ncei_positions, lon >=
as.numeric(lon_tail)-as.numeric(GPS_buffer) & lon <= as.numeric(lon_tail)+as.numeric(GPS_buffer))
305   # ordering the df by date and selecting unique values only
306   ncei_positions <- ncei_positions[order(ncei_positions$DateTime),]
307   ncei_positions <- unique(ncei_positions)
308   # rename the rows to reflect unique data
309   row.names(ncei_positions) <- 1:nrow(ncei_positions)
310   # remove bad data
311   ncei_positions <- ncei_positions[complete.cases(ncei_positions),]
312
313   # apply corrected positions for ncei data
314   for(n in ncei_list){
315     print(n)
316     dat <- get(n)
317     column_names <- names(dat)
318     if("lat" %in% column_names){
319       dat$lat <- NULL; dat$lon <- NULL
320       dat <- left_join(dat, ncei_positions, by = "DateTime")
321       dat <- dplyr::select(dat, all_of(column_names))
322       assign(n,dat)
323     }else{print(paste0("no GPS data in ", n))}
324     rm(dat)
325   }
326 }
327
328 # export as csv and rds
329 print("saving original GPS data")
330 if (!file.exists(paste0(clean_dir,buoy,"/"))) {dir.create((paste0(clean_dir,buoy,"/")))}
331 # write.table(ncei_positions, paste0(clean_dir,buoy,"/s_",buoy,"_GPS_ALL.csv"), row.names=FALSE,
col.names = TRUE, sep = ",")
332 # saveRDS(ncei_positions, file = paste0(clean_dir,buoy,"/s_",buoy,"_GPS_ALL.rds"))
333 # rm(ncei_positions)
334
335 # list out the NDBC and NCEI data
336 ndbc_list <- ls(pattern = paste0("s_",buoy,"_ndbc_"))
337 print(ndbc_list)
338 ndbc_ls <- vector()
339 remainder <- vector()

```

```

340
341 # load individual datasets and geoclean pairs
342 for(n in ndbc_list){
343   name_export <- n
344   print(name_export)
345
346   # find matching datasets
347   ndbc <- get(n)
348   ncei_name <- gsub("ndbc_", "ncei_",n)
349   tryCatch({dat <- get(ncei_name)},error=function(cond){print(paste0("no matching ncei data for ",
n))})
350   if(exists("dat")==TRUE){
351     if(grepl("cols", ncei_name, fixed = TRUE)){
352       if("lat" %in% names(ndbc)){
353         ncei <- get(ls(pattern = gsub("ndbc_", "ncei_",n)))
354       }else{
355         ncei_count <- as.numeric(gsub("cols","",unlist(strsplit(n,"_freq_"))[2]))+2
356         ncei_name2 <- gsub("ndbc_",
"ncei_",paste0(unlist(strsplit(n,"_freq_"))[1],"_freq_",ncei_count,"cols"))
357         # check if ncei data exists
358         if(exists(ncei_name2)){
359           ncei <- get(ls(pattern = ncei_name2))
360         }else{
361           print("no matching ncei data")
362           ndbc_ls <- c(ndbc_ls,n)
363         }
364         rm(ncei_count, ncei_name2)
365       }
366     }else{
367       ncei <- get(ls(pattern = gsub("ndbc_", "ncei_",n)))
368     }
369     rm(ncei_name)
370   }else {remainder <- c(remainder,n)}
371
372   # handling ndbc data that has no matching ncei data
373   if(exists("ncei")){
374     # remove blank rows
375     ndbc <- ndbc[rowSums(is.na(ndbc)) != ncol(ndbc)-1,]
376     ncei <- ncei[rowSums(is.na(ncei)) != ncol(ncei)-1,]
377     positions <- dplyr::select(ncei, DateTime, lat,lon)
378     # original dfs
379     ncei_orig <- ncei #<- ncei_orig
380     ndbc_orig <- ndbc #<- ndbc_orig
381     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)
382     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)
383
384     # checking geographical positions for service visits and buoy adrift, i.e. not in watch
circle

```

```

385 # removing multiple significant places that skew the sorted table results
386 positions$lat <- as.numeric(positions$lat); positions$lon <- as.numeric(positions$lon)
387 positions$lat <- signif(positions$lat,3)
388 positions$lon <- signif(positions$lon,3)
389
390 # two methods of finding mode
391 library(modeest)
392 lat_mean <- mfv(positions$lat, na_rm = TRUE)
393 lon_mean <- mfv(positions$lon, na_rm = TRUE)
394 print(paste0("mean method - lat: ", lat_mean, "; lon: ", lon_mean))
395 # using a sorted table of value occurrences to find most common lat/lon
396 lat_tail <- tail(names(sort(table(positions$lat))),1)
397 lon_tail <- tail(names(sort(table(positions$lon))),1)
398 print(paste0("sorted table method - lat: ", lat_tail, "; lon: ", lon_tail))
399 # check positions in ndbc bulk
400 print(paste0("range - lat: ", range(positions$lat, na.rm = TRUE)[1]," ",
401         range(positions$lat, na.rm = TRUE)[2]))
402 print(paste0("range - lon: ", range(positions$lon, na.rm = TRUE)[1]," ",
403         range(positions$lon, na.rm = TRUE)[2]))
404 rm(positions)
405
406 # matching time setkeys to only include geoClean data times from NDBC website
407 # set as data.tables
408 library(data.table)
409 ncei_dt <- data.table(dplyr::select(ncei,DateTime,lat,lon))
410 ndbc_dt <- data.table(dplyr::select(ndbc,DateTime))
411 setkey(ncei_dt,DateTime)
412 setkey(ndbc_dt,DateTime)
413
414 # manipulate and subset to include common data, using NDBC as ref dataset
415 ncei_dt$date <- ncei_dt$DateTime
416 match_dt <- ncei_dt[ndbc_dt, roll = "nearest" ]
417 # removing rows with no dates
418 match_dt <- match_dt[!with(match_dt,is.na(DateTime)| is.na(date)),]
419 match_dt$DateTime <- match_dt$date
420 match_dt <- match_dt[,1:3]
421 # subset
422 ncei2 <- data.table(ncei)
423 ndbc2 <- data.table(ndbc)
424 setkey(ncei2,DateTime)
425 setkey(ndbc2,DateTime)
426 ncei <- data.frame(ncei2[match_dt, roll = "nearest" ], stringsAsFactors = FALSE)
427 ndbc <- data.frame(ndbc2[match_dt, roll = "nearest" ], stringsAsFactors = FALSE)
428 # remove extra lat/lon in ndbc data
429 ncei <- ncei[ , -which(names(ncei) %in% c("i.lat", "i.lon"))]
430 # remove x in column names
431 colnames(ndbc) <- gsub("X", "", colnames(ndbc))
432 colnames(ncei) <- gsub("X", "", colnames(ncei))
433 # remove 0.0100 column if empty
434 if("0.0100" %in% names(ncei)) {if(sum(is.na(ncei$`0.0100`))==nrow(ncei)) {ncei$`0.0100` <-

```

```

NULL}}
# re-order datasets
idcols <- names(ncei)
# formatting datasets
if(unlist(strsplit(n, "_"))[4]== "stdmet"){
  ncei_metadata <- ncei
  idcols <- str_subset(idcols, "_metadata", negate = TRUE)
  idcols <- str_subset(idcols, "_2", negate = TRUE)
  idcols <- gsub("_1", "", idcols)
}
if(grepl("_freq_", name_export)){
  if("0.0200" %in% names(ndbc)){ndbc <- dplyr::select(ndbc, all_of(idcols))
  }else{ndbc$`0.0200` <- NA; ndbc <- dplyr::select(ndbc, all_of(idcols))}
}

# save comparable datasets
ncei_comp <- ncei; ndbc_comp <- ndbc
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)
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)
name_comp <- paste0(name_export, "_comp")
assign(name_comp, ndbc_comp)
name_comp <- gsub("ndbc", "ncei", paste0(name_export, "_comp"))
assign(name_comp, ncei_comp)
# housekeeping
rm(ncei_dt, ndbc_dt, ncei2, ndbc2, match_dt, name_comp)

# export as csv and rds
print(paste0("saving comps ", n))
if (!file.exists(paste0(clean_dir, buoy, "/"))) {dir.create((paste0(clean_dir, buoy, "/")))}
# write.table(ncei_comp, paste0(clean_dir, buoy, "/", gsub("ndbc_", "ncei_", n), "_comp.csv"),
row.names=FALSE, col.names = TRUE, sep = ",")
# saveRDS(ncei_comp, file = paste0(clean_dir, buoy, "/", gsub("ndbc_", "ncei_", n), "_comp.rds"))
# write.table(ndbc_comp, paste0(clean_dir, buoy, "/", n, "_comp.csv"), row.names=FALSE,
col.names = TRUE, sep = ",")
# saveRDS(ndbc_comp, file = paste0(clean_dir, buoy, "/", n, "_comp.rds"))

# filter for pre-matching data
start_date_ncei <- min(ncei_comp$DateTime, na.rm = TRUE)
ncei_pre <- dplyr::filter(ncei_orig, ncei_orig$DateTime < start_date_ncei)
# quick check = should == 0
start_date_ndbc <- min(ndbc_comp$DateTime, na.rm = TRUE)
ndbc_pre <- dplyr::filter(ndbc_orig, ndbc_orig$DateTime < start_date_ndbc)
# removing empty dfs
for(i in c("ndbc_pre", "ncei_pre")){
  dat <- get(i)
  if(dim(dat)[1]>0){
    # print(paste0("Error: pre filter data available in ", n))
    print(paste0("Pre filter data available in ", i))
  }
}

```

```

478         if(i == "ndbc_pre"){
479             # if pre-ndbc exists, there are no geographical positions that match.
480             # export and delete
481             # export as csv and rds
482             print(paste0("saving pre NDBC data: ",name_export))
483             if (!file.exists(paste0(clean_dir,buoy,"/")))
484                 {dir.create((paste0(clean_dir,buoy,"/")))}
485             if (!file.exists(paste0(clean_dir,buoy,"/no_gps/")))
486                 {dir.create((paste0(clean_dir,buoy,"/no_gps/")))}
487             # write.table(ndbc_pre,
488             # paste0(clean_dir,buoy,"/no_gps/",name_export,"_pre_NDBC.csv"),
489             # row.names=FALSE, col.names = TRUE, sep = ",")
490             # saveRDS(ndbc_pre, file =
491             # paste0(clean_dir,buoy,"/no_gps/",name_export,"_pre_NDBC.rds"))
492             rm(ndbc_pre)
493         }
494     }else{
495         rm(dat)
496         rm(list = ls()[grepl(i, ls())])
497         print(paste0("removing empty df: ",i))
498     }
499 }
500 rm(i, ndbc_orig, ncei_orig)
501
502 # filter the data on these geographical conditions
503 dat_ls <- c("ncei_comp", "ncei_pre", "ndbc_comp")
504 for(i in dat_ls){
505     if(exists(i)){
506         print(paste0("geoCleaning: ",i))
507         dat <- get(i)
508         if(dim(dat)[1]>2){
509             dat <- dplyr::filter(dat, lat >= as.numeric(lat_tail)-as.numeric(GPS_buffer) &
510             lat <= as.numeric(lat_tail)+as.numeric(GPS_buffer))
511             dat <- dplyr::filter(dat, lon >= as.numeric(lon_tail)-as.numeric(GPS_buffer) &
512             lon <= as.numeric(lon_tail)+as.numeric(GPS_buffer))
513
514             dat <- unique(dat)
515
516             if(unlist(strsplit(n, "_"))[4]== "stdmet"){
517
518                 #-----
519                 # using a sorted table of value occurrences to find outliers in stdmet data
520
521                 #-----
522
523                 print("Re-ordering matching stdmet columns")
524                 # preserving metadata
525                 met <- names(dat)
526                 if(length(grep("_metadata",met))>0){

```



```

517         dat_metadata <- dplyr::select(dat, c(DateTime, contains("_metadata")))
518         dat <- dplyr::select(dat, !contains("_metadata"))
519     }
520
521     # identify and remove outlier
522     for(c in 4:ncol(dat)){
523         # c <- 5
524         print(colnames(dat[c]))
525         # find outliers
526         outlier_table <- sort(as.numeric(names(sort(table(dat[,c])))))
527         if(length(outlier_table)>1){
528             if(max(outlier_table, na.rm = TRUE) >=
529                 2*outlier_table[length(outlier_table)-1]){
530                 index <- which(dat[,c] == max(outlier_table, na.rm = TRUE))
531                 dat[index,c] <- NA
532                 print(paste0("outlier of ", max(outlier_table, na.rm = TRUE), "
533                     removed from ", colnames(dat[c])))
534             }else{print(paste0("no outliers for ", colnames(dat[c])))}
535         }else{print(paste0("no outliers for ", colnames(dat[c])))}
536     }
537     # correcting for directional data outliers
538     dir_list <- grep("_direction", names(dat))
539     for(dir_column in dir_list){
540         print(names(dat)[dir_column])
541         direction_ls <- which(dat[,dir_column] > 360)
542         print(paste0("outlier indices: ", direction_ls))
543         if(length(direction_ls)>0){for(d in direction_ls){dat[d,dir_column] <-
544             NA}}
545         rm(direction_ls)
546     }
547     rm(dir_list)
548
549     # rejoining ncei stdmet data and metadata
550     if(length(grep("_metadata", met))>0){
551         dat <- full_join(dat, dat_metadata, by = "DateTime")
552         dat <- dplyr::select(dat, all_of(met))
553     }
554
555     # export as csv and rds
556     if(i == "ncei_pre"){
557         name1 <- gsub("ndbc", "ncei", name_export)
558         name2 <- paste0(name1, "_preNDBC_geoClean")
559     }else if(i == "ndbc_pre"){
560         name2 <- paste0(name_export, "_preNDBC_geoClean")
561     }else if(i == "ncei_comp"){
562         name1 <- gsub("ndbc", "ncei", name_export)
563         name2 <- paste0(name1, "_geoClean")
564     }else{
565         name2 <- paste0(name_export, "_geoClean")
566     }

```

```

564     }
565     # export as csv and rds
566     print("")
567     print(paste0("saving geoCleaned: ",name2))
568     if (!file.exists(paste0(clean_dir,buoy,"/")))
569     {dir.create((paste0(clean_dir,buoy,"/")))}
570     # write.table(dat, paste0(clean_dir,buoy,"/",name2,".csv"), row.names=FALSE,
571     col.names = TRUE, sep = ",")
572     # saveRDS(dat, file = paste0(clean_dir,buoy,"/",name2,".rds"))
573     # save for RData later
574     assign(name2,dat)
575     # housekeeping
576     rm(dat, dat_metadata)
577     }else{print("df dim less than 2 rows")}
578     rm(dat)
579     }else{print(paste0("no preNDBC data for ",i))}
580     rm(i,dat)
581   }
582   # count
583   ncei_geoClean_count <- nrow(ncei_comp); ncei_geoClean_start <- min(ncei_comp$DateTime,
584   na.rm = TRUE); ncei_geoClean_end <- max(ncei_comp$DateTime, na.rm = TRUE)
585   ndbc_geoClean_count <- nrow(ndbc_comp); ndbc_geoClean_start <- min(ndbc_comp$DateTime,
586   na.rm = TRUE); ndbc_geoClean_end <- max(ndbc_comp$DateTime, na.rm = TRUE)
587
588   # build table
589   ncei_count <- t(data.frame(ncei_orig_count, ncei_orig_start, ncei_orig_end,
590   ncei_comp_count, ncei_comp_start, ncei_comp_end,
591   ncei_geoClean_count, ncei_geoClean_start, ncei_geoClean_end))
592   colnames(ncei_count) <- gsub("ndbc", "ncei", n)
593   ncei_count <- data.frame(ncei_count, stringsAsFactors = FALSE)
594
595   ndbc_count <- t(data.frame(ndbc_orig_count, ndbc_orig_start, ndbc_orig_end,
596   ndbc_comp_count, ndbc_comp_start, ndbc_comp_end,
597   ndbc_geoClean_count, ndbc_geoClean_start, ndbc_geoClean_end))
598   colnames(ndbc_count) <- n
599   ndbc_count <- data.frame(ndbc_count, stringsAsFactors = FALSE)
600
601   # add to table
602   data_count <- cbind(data_count,ncei_count)
603   data_count <- cbind(data_count,ndbc_count)
604   # remove unclean data
605   rm(ncei, ndbc, ncei_comp, ndbc_comp, ncei_count, ndbc_count,
606   ncei_orig_count, ncei_comp_count, ncei_geoClean_count,ncei_orig_start, ncei_orig_end,
607   ncei_comp_start, ncei_comp_end,ncei_geoClean_start,ncei_geoClean_end,
608   ndbc_orig_count, ndbc_comp_count, ndbc_geoClean_count, ndbc_orig_start, ndbc_orig_end,
609   ndbc_comp_start, ndbc_comp_end,ndbc_geoClean_start, ndbc_geoClean_end)
610 }
611 if(exists("ndbc")){rm(ndbc)}; if(exists("ncei")){rm(ncei)}
612 }

```

```

608 #-----
609 ## attach stdmet metadata to geoCleaned ndbc stdmet dataset
610 #-----
611 ncei_met <- get(ls(pattern = "ncei_stdmet_geoClean"))
612 ndbc_met <- get(ls(pattern = "ndbc_stdmet_geoClean"))
613
614 # deal with significant places that skew alignment
615 ndbc_col_num <- grep("wind_direction", colnames(ndbc_met))
616 ncei_col_num <- grep("wind_direction", colnames(ncei_met))
617 for(n in ndbc_col_num:ncol(ndbc_met)){if(is.numeric(ndbc_met[,n]) == TRUE){ndbc_met[,n] <-
round(ndbc_met[n],2)}}
618 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)}}
619
620 # select variables to run
621 var_list <- names(ndbc_met)
622 # selecting only main variable per sensor to increase computational efficiency
623 var_list_main <- var_list[!var_list %in% c("DateTime", "lat", "lon", "wind_direction", "wind_gust",
624                                           "dominant_wave_period", "average_wave_period",
                                           "mean_wave_direction")]
625
626 print(var_list_main)
627
628 print(Sys.time())
629
630 for(i in var_list_main){
631   print(paste0("matching ncei metadata to ndbc data for ",i," for ",buoy))
632   print(Sys.time())
633
634   # subset data
635   library(dplyr)
636   ndbc <- dplyr::select(ndbc_met, DateTime, contains(i))
637   ncei <- dplyr::select(ncei_met, DateTime, contains(i))
638
639   # if no data for this variable, skip the following...
640   if(sum(is.na(ndbc[i])) != nrow(ndbc)){
641
642     # single sensor
643     if(dim(ncei)[2]==3){
644       print("handling single sensor")
645       ncei$two <- NA; ncei$met <- NA
646       colnames(ncei) <- c("DateTime",
647                           paste0(i,"_1"),paste0(i,"_metadata_1"),paste0(i,"_2"),paste0(i,"_metadata_2"))
648     }
649
650     # multiple sensors
651     ndbc <- left_join(ndbc,ncei,by = "DateTime")
652     # remove rows with no data
653     ndbc <- ndbc[!is.na(ndbc[,2]), ]
654     # order and re-index rows
655     ndbc <- ndbc[order(ndbc$DateTime),]

```

```

654 ndbc <- unique(ndbc)
655 row.names(ndbc) <- 1:nrow(ndbc)
656 # set up metadata column and select relevant data columns
657 ndbc$metadata <- NA
658 ncei_dat_ls <- names(dplyr::select(ncei, contains(i)))
659
660 # matching ndbc && ncei primary sensor where secondary data is na
661 var_index <- which(ndbc[i] == ndbc[ncei_dat_ls[1]] & is.na(ndbc[ncei_dat_ls[3]]))# |
is.na(ndbc[ncei_dat_ls[3]]))
length(var_index)
if(length(var_index)>0){
  print("check for published duplicate metadata")
  for(d in var_index){
    # insert metadata, accounting for NDBC practice of inserting duplicate data
    ndbc$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)

```

```

697     if(length(var_index)>0){
698         print("check for secondary sensor metadata with non-matching primary metadata")
699         for(d in var_index){
700             # insert metadata
701             ndbc$metadata[d] <- ndbc[d,ncei_dat_ls[4]]
702         }
703     }else{print("no only secondary sensor metadata")}
704
705     # ndbc data that matches ncei primary but DOESN'T match secondary sensor - logical primary
706     # sensor usage
707     var_index <- which(is.na(ndbc$metadata) & ndbc[i] == ndbc[ncei_dat_ls[1]] &
708     ndbc[ncei_dat_ls[1]] != ndbc[ncei_dat_ls[3]])
709     length(var_index)
710     if(length(var_index)>0){
711         print("check for primary sensor metadata with non-matching secondary metadata")
712         for(d in var_index){
713             # insert metadata
714             ndbc$metadata[d] <- ndbc[d,ncei_dat_ls[2]]
715         }
716     }else{print("no only primary sensor data")}
717
718     # Test if exists: ndbc data that DOESN'T match ncei primary or secondary but is still present
719     # - possible NDBC data QC
720     var_index <- which(is.na(ndbc$metadata) & is.na(ndbc[ncei_dat_ls[1]]) &
721     is.na(ndbc[ncei_dat_ls[3]]))
722     length(var_index)
723     if(length(var_index)>0){
724         print("check for remaining primary sensor metadata")
725         for(d in var_index){
726             # insert metadata
727             ndbc$metadata[d] <- ndbc[d,ncei_dat_ls[2]]
728             ndbc[d,ncei_dat_ls[1]] <- ndbc[d,i]
729         }
730     }else{print("no remaining primary sensor metadata")}
731
732     # Test if exists: ndbc data that DOESN'T match ncei primary but is still present - possible
733     # NDBC rounding issues or data QC
734     var_index <- which(is.na(ndbc$metadata) & !is.na(ndbc[i]) & !is.na(ndbc[ncei_dat_ls[1]]) &
735     !is.na(ndbc[ncei_dat_ls[2]]) & is.na(ndbc[ncei_dat_ls[3]]) & is.na(ndbc[ncei_dat_ls[4]]))
736     length(var_index)
737     if(length(var_index)>0){
738         print("check for remaining primary sensor metadata")
739         for(d in var_index){
740             # insert metadata
741             ndbc$metadata[d] <- ndbc[d,ncei_dat_ls[2]]
742         }
743     }else{print("no remaining primary sensor metadata")}
744
745     # Test if exists: ndbc data that DOESN'T match ncei secondary but is still present - possible
746     # NDBC rounding issues or data QC

```

```

740 var_index <- which(is.na(ndbc$metadata) & !is.na(ndbc[ncei_dat_ls[3]]) &
741 !is.na(ndbc[ncei_dat_ls[4]]) & is.na(ndbc[ncei_dat_ls[1]]) & is.na(ndbc[ncei_dat_ls[2]]))
742 length(var_index)
743 if(length(var_index)>0){
744     print("check for remaining secondary sensor metadata")
745     for(d in var_index){
746         # insert metadata
747         ndbc$metadata[d] <- ndbc[d,ncei_dat_ls[4]]
748     }
749 }else{print("no remaining secondary sensor metadata")}
750
751 # Test if exists: remaining primary metadata
752 var_index <- which(is.na(ndbc$metadata) & is.na(ndbc[ncei_dat_ls[1]]) &
753 !is.na(ndbc[ncei_dat_ls[2]]) & is.na(ndbc[ncei_dat_ls[3]]) & is.na(ndbc[ncei_dat_ls[4]]))
754 length(var_index)
755 if(length(var_index)>0){
756     print("check for remaining ndbc blank metadata")
757     for(d in var_index){
758         # insert metadata
759         ndbc$metadata[d] <- ndbc[d,ncei_dat_ls[2]]#}
760     }
761 }else{print("no remaining ndbc blank primary metadata")}
762
763 # Test if exists: look for remaining secondary metadata
764 var_index <- which(is.na(ndbc$metadata) & is.na(ndbc[ncei_dat_ls[1]]) &
765 is.na(ndbc[ncei_dat_ls[2]]) & is.na(ndbc[ncei_dat_ls[3]]) & !is.na(ndbc[ncei_dat_ls[4]]))
766 length(var_index)
767 if(length(var_index)>0){
768     print("check for remaining ndbc blank metadata")
769     for(d in var_index){
770         # insert metadata
771         ndbc$metadata[d] <- ndbc[d,ncei_dat_ls[4]]#}
772     }
773 }else{print("no remaining ndbc blank secondary metadata")}
774
775 # Test if exists: remaining metadata where no data matches... possible shoreside corrections?
776 var_index <- which(is.na(ndbc$metadata) & !is.na(ndbc[ncei_dat_ls[1]]) &
777 !is.na(ndbc[ncei_dat_ls[2]]) &
778 !is.na(ndbc[ncei_dat_ls[3]]) & !is.na(ndbc[ncei_dat_ls[4]]) & ndbc[i] !=
779 ndbc[ncei_dat_ls[1]] &
780 ndbc[i] != ndbc[ncei_dat_ls[3]] & ndbc[ncei_dat_ls[1]] !=
781 ndbc[ncei_dat_ls[3]])
782 length(var_index)
783 if(length(var_index)>0){
784     print("check for remaining ndbc blank metadata")
785     for(d in var_index){
786         # insert metadata
787         ndbc$metadata[d] <- ndbc[d,ncei_dat_ls[2]]#}
788     }
789 }else{print("no remaining blank metadata")}

```

```

# 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] ==

```

```

828         df[j+2,3]){df[j,3] <- df[j-2,3]}},error=function(cond){print("")})
      tryCatch({if(df[j,3] != df[j-3,3] & df[j,3] != df[j+3,3] & df[j-3,3] ==
829         df[j+3,3]){df[j,3] <- df[j-3,3]}},error=function(cond){print("")})
      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 <-
880     c("DateTime", "lat", "lon", "wind_direction", "wind_direction_metadata", "wind_speed", "wind_speed_metadata",
881       "wind_gust", "wind_gust_metadata",
882
883       "significant_wave_height", "significant_wave_height_metadata", "dominant_wave_period", "domi-
884       nant_wave_period_metadata", "average_wave_period",
885       "average_wave_period_metadata", "mean_wave_direction", "mean_wave_direction_metadata"
886       , "air_pressure_at_sea_level", "air_pressure_at_sea_level_metadata",
887
888       "air_temperature", "air_temperature_metadata", "sea_surface_temperature", "sea_surface_tempe-
889       rature_metadata", "dew_point_temperature", "dew_point_temperature_metadata")
890     ndbc_met <- dplyr::select(ndbc_met, all_of(ndbc_ord))
891
892     #-----
893
894     # correct ncei
895     var_list_rem <- c("wind_direction", "wind_gust")
896     print(var_list_rem)
897     for(i in var_list_rem){
898         print(i)
899         if(any(grepl(i, colnames(ncei_met)))==TRUE){
900             # subset data
901             library(dplyr)
902             ndbc <- dplyr::select(ndbc_met, DateTime, contains(i))
903             ncei <- dplyr::select(ncei_met, DateTime, contains(i))
904             # join df
905             merged_df <- merge(ndbc, ncei, by = "DateTime")
906             # select relevant data columns
907             ncei_dat_ls <- names(dplyr::select(ncei, contains(i)))
908
909             # ndbc data that DOESN't match ncei primary and secondary sensor - to account for possible
910             build errors
911             var_index <- which(!is.na(merged_df[i]) != !is.na(merged_df[ncei_dat_ls[1]]) &
912               !is.na(merged_df[i]) != !is.na(merged_df[ncei_dat_ls[3]]))
913             length(var_index)
914             if(length(var_index)>0){
915                 print("check for non-matching sensor data")
916                 for(d in var_index){merged_df[d,ncei_dat_ls[1]] <- merged_df[d,i]}
917             }else{print("no non-matching sensor data")}
918
919             # ndbc data that DOESN't match ncei primary and secondary sensor - to account for possible
920             build errors
921             var_index <- which(merged_df[i] != merged_df[ncei_dat_ls[1]] & merged_df[ncei_dat_ls[1]] ==
922               merged_df[ncei_dat_ls[3]])
923             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
919     build errors
920     var_index <- which(!is.na(merged_df[i]) != merged_df[ncei_dat_ls[1]] &
921     merged_df[ncei_dat_ls[1]] == merged_df[ncei_dat_ls[3]])
922     length(var_index)
923     if(length(var_index)>0){
924         print("check for non-matching sensor data")
925         for(d in var_index){merged_df[d,ncei_dat_ls[1]] <- merged_df[d,i]}
926     }else{print("no non-matching sensor data")}
927
928     # remove ndbc and ncei data
929     merged_df[,2:3] <- NULL
930     drop_columns <- grep(i, colnames(ncei_met))
931     ncei_met[drop_columns] <- NULL
932     ncei_met <- left_join(ncei_met, merged_df, by = "DateTime")
933     rm(merged_df, ndbc, ncei, var_index, drop_columns)
934 }
935 # re-order columns
936 if(ncol(ncei_met)== 41){
937     print("ncei data format - dual sst")
938     ncei_met <- dplyr::select(ncei_met, DateTime, lat, lat_metadata, lon, lon_metadata,
939     wind_direction_1, wind_direction_metadata_1, wind_direction_2,
940     wind_direction_metadata_2,
941     wind_speed_1, wind_speed_metadata_1, wind_speed_2, wind_speed_metadata_2,
942     wind_gust_1, wind_gust_metadata_1, wind_gust_2, wind_gust_metadata_2,
943     significant_wave_height, significant_wave_height_metadata,
944     dominant_wave_period, dominant_wave_period_metadata,
945     average_wave_period, average_wave_period_metadata, mean_wave_direction,
946     mean_wave_direction_metadata,
947     air_pressure_at_sea_level_1, air_pressure_at_sea_level_metadata_1,
948     air_pressure_at_sea_level_2, air_pressure_at_sea_level_metadata_2,
949     air_temperature_1, air_temperature_metadata_1, air_temperature_2,
950     air_temperature_metadata_2,
951     sea_surface_temperature_1,
952     sea_surface_temperature_metadata_1, sea_surface_temperature_2,
953     sea_surface_temperature_metadata_2,
954     dew_point_temperature_1, dew_point_temperature_metadata_1,
955     dew_point_temperature_2, dew_point_temperature_metadata_2)
956 }else if(ncol(ncei_met)== 39){
957     if("sea_surface_temperature" %in% names(ncei_met)){
958         print("ncei data format - single sst")
959         ncei_met <- dplyr::select(ncei_met, DateTime, lat, lat_metadata, lon, lon_metadata,
960         wind_direction_1, wind_direction_metadata_1, wind_direction_2,
961         wind_direction_metadata_2,

```

```

952         wind_speed_1, wind_speed_metadata_1, wind_speed_2, wind_speed_metadata_2,
953         wind_gust_1, wind_gust_metadata_1, wind_gust_2, wind_gust_metadata_2,
954         significant_wave_height, significant_wave_height_metadata,
955         dominant_wave_period, dominant_wave_period_metadata,
956         average_wave_period, average_wave_period_metadata, mean_wave_direction,
957         mean_wave_direction_metadata,
958         air_pressure_at_sea_level_1, air_pressure_at_sea_level_metadata_1,
959         air_pressure_at_sea_level_2, air_pressure_at_sea_level_metadata_2,
960         air_temperature_1, air_temperature_metadata_1, air_temperature_2,
961         air_temperature_metadata_2,
962         sea_surface_temperature, sea_surface_temperature_metadata,
963         dew_point_temperature_1, dew_point_temperature_metadata_1,
964         dew_point_temperature_2, dew_point_temperature_metadata_2)
965     }else if("dew_point_temperature" %in% names(ncei_met)){
966         print("ncei data format - single dew point temperature")
967         dat <- dplyr::select(dat, DateTime, lat, lat_metadata, lon, lon_metadata,
968         wind_direction_1, wind_direction_metadata_1, wind_direction_2,
969         wind_direction_metadata_2,
970         wind_speed_1, wind_speed_metadata_1, wind_speed_2, wind_speed_metadata_2,
971         wind_gust_1, wind_gust_metadata_1, wind_gust_2, wind_gust_metadata_2,
972         significant_wave_height, significant_wave_height_metadata,
973         dominant_wave_period, dominant_wave_period_metadata,
974         average_wave_period, average_wave_period_metadata, mean_wave_direction,
975         mean_wave_direction_metadata,
976         air_pressure_at_sea_level_1, air_pressure_at_sea_level_metadata_1,
977         air_pressure_at_sea_level_2, air_pressure_at_sea_level_metadata_2,
978         air_temperature_1, air_temperature_metadata_1, air_temperature_2,
979         air_temperature_metadata_2,
980         sea_surface_temperature_1,
981         sea_surface_temperature_metadata_1, sea_surface_temperature_2,
982         sea_surface_temperature_metadata_2,
983         dew_point_temperature, dew_point_temperature_metadata)
984     }else{print("error: dim = 39 but dat names don't match")}
985 }else if(ncol(dat)== 37){
986     print("ncei data format - single dew point")
987     dat <- dplyr::select(dat, DateTime, lat, lat_metadata, lon, lon_metadata,
988     wind_direction_1, wind_direction_metadata_1, wind_direction_2,
989     wind_direction_metadata_2,
990     wind_speed_1, wind_speed_metadata_1, wind_speed_2, wind_speed_metadata_2,
991     wind_gust_1, wind_gust_metadata_1, wind_gust_2, wind_gust_metadata_2,
992     significant_wave_height, significant_wave_height_metadata,
993     dominant_wave_period, dominant_wave_period_metadata,
994     average_wave_period, average_wave_period_metadata, mean_wave_direction,
995     mean_wave_direction_metadata,
996     air_pressure_at_sea_level_1, air_pressure_at_sea_level_metadata_1,
997     air_pressure_at_sea_level_2, air_pressure_at_sea_level_metadata_2,
998     air_temperature_1, air_temperature_metadata_1, air_temperature_2,
999     air_temperature_metadata_2,
1000     sea_surface_temperature, sea_surface_temperature_metadata,
1001     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,
989                        wind_gust, wind_gust_metadata,
990                        significant_wave_height, significant_wave_height_metadata,
991                        dominant_wave_period, dominant_wave_period_metadata,
992                        average_wave_period, average_wave_period_metadata, mean_wave_direction,
993                        mean_wave_direction_metadata,
994                        air_pressure_at_sea_level, air_pressure_at_sea_level_metadata,
995                        air_temperature, air_temperature_metadata,
996                        sea_surface_temperature, sea_surface_temperature_metadata,
997                        dew_point_temperature, dew_point_temperature_metadata)
998 }
999
1000 #-----
1001 # find unique rows only
1002 ndbc_met <- unique(ndbc_met)
1003 ncei_met <- unique(ncei_met)
1004 # rename the rows to reflect unique data
1005 row.names(ndbc_met) <- 1:nrow(ndbc_met)
1006 row.names(ncei_met) <- 1:nrow(ncei_met)
1007
1008 # save file in glob environment
1009 ndbc_name <- ls(pattern = "ndbc_stdmet_geoClean")
1010 ncei_name <- ls(pattern = "ncei_stdmet_geoClean")
1011
1012 assign(ndbc_name, ndbc_met)
1013 assign(ncei_name, ncei_met)
1014
1015 # housekeeping
1016 rm(ncei_met, ndbc_met, ncei_metadata, ndbc_name, ncei_name)
1017
1018 #-----
1019 ## geoclean unique data
1020 #-----
1021 print(paste0("GeoCleaning unique datasets for ", buoy))
1022 ncei_list <- ls(pattern = "_ncei_")
1023 remove <- gsub("_ndbc_", "_ncei_", ndbc_list)
1024 ncei_list <- ncei_list[!grepl("_geoClean", ncei_list)]
1025 ncei_list <- ncei_list[!grepl("_comp", ncei_list)]
1026 ncei_list <- ncei_list[!grepl("_sdmet_netcdf", ncei_list)]
1027 ncei_list <- ncei_list[str_remove_all(ncei_list, paste(remove, collapse = "|"))!=""]
1028 # add extra ndbc data with no matching ncei
1029 if(length(ndbc_ls)>0){ncei_list <- c(ncei_list, ndbc_ls)}
1030 if(length(remainder)>0){ncei_list <- c(ncei_list, remainder)}
1031 # remove buoy info
1032 ncei_list <- gsub(paste0("s_",buoy,"_"), "", ncei_list)
1033 print(ncei_list)

```

```

1031   if("ncei_sensor_output" %in% ncei_list){
1032       dat_name <- paste0("s_",buoy,"_ncei_sensor_output")
1033       dat <- get(dat_name)
1034       dat <- left_join(dat, ncei_positions, by = "DateTime")
1035       assign(dat_name,dat)
1036       rm(dat)
1037   }
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$DateTime, na.rm = TRUE); dat_orig_end <-
1053       max(dat$DateTime, na.rm = TRUE)
1054
1055       # export as csv and rds
1056       print(paste0("saving orig ",n))
1057       if (!file.exists(paste0(clean_dir,buoy,"/"))) {dir.create((paste0(clean_dir,buoy,"/")))}
1058       # write.table(dat,
1059       paste0(clean_dir,buoy,"/","s_",buoy,"_",dat_source,"_",name_export,"_orig.csv"),
1060       row.names=FALSE, col.names = TRUE, sep = ",")
1061       # saveRDS(dat, file =
1062       paste0(clean_dir,buoy,"/","s_",buoy,"_",dat_source,"_",name_export,"_orig.rds"))
1063
1064       # checking geographical positions for service visits and buoy adrift, i.e. not in watch circle
1065       # using a sorted table of value occurrences to find most common lat/lon
1066       if("lat" %in% names(dat)){
1067           positions <- dplyr::select(dat,DateTime,lat,lon)
1068           positions$lat <- as.numeric(positions$lat); positions$lon <- as.numeric(positions$lon)
1069           positions$lat <- signif(positions$lat,3)
1070           positions$lon <- signif(positions$lon,3)
1071       }else{
1072           print("no gps data, adding data from master positions data")
1073           positions <- ncei_positions
1074           ## Use round.Date to round, then format to format
1075           positions$DateTime <- format(round(positions$DateTime, units="hours"), format="%Y-%m-%d

```

```

1076         positions <- dplyr::select(positions,DateTime,lat,lon)
1077         positions$lat <- as.numeric(positions$lat); positions$lon <- as.numeric(positions$lon)
1078         positions$lat <- signif(positions$lat,3); positions$lon <- signif(positions$lon,3)
1079         # positions <- unique(positions)
1080     }
1081
1082     # two methods of finding mode
1083     library(modeest)
1084     lat_mean <- mfv(positions$lat, na_rm = TRUE)
1085     lon_mean <- mfv(positions$lon, na_rm = TRUE)
1086     print(paste0("mean method - lat: ", lat_mean, "; lon: ", lon_mean))
1087     # using a sorted table of value occurrences to find most common lat/lon
1088     lat_tail <- tail(names(sort(table(positions$lat))),1)
1089     lon_tail <- tail(names(sort(table(positions$lon))),1)
1090     print(paste0("sorted table method - lat: ", lat_tail, "; lon: ", lon_tail))
1091     # check positions in ndbc bulk
1092     print(paste0("range - lat: ", range(positions$lat, na.rm = TRUE)[1]," ", range(positions$lat,
na.rm = TRUE)[2]))
1093     print(paste0("range - lon: ", range(positions$lon, na.rm = TRUE)[1]," ", range(positions$lon,
na.rm = TRUE)[2]))
1094
1095     # filter the data on these geographical conditions
1096     if(dat_source == "ndbc"){dat <- left_join(dat, positions, by = "DateTime")}
1097     dat <- dplyr::filter(dat, lat >= as.numeric(lat_tail)-as.numeric(GPS_buffer) & lat <=
as.numeric(lat_tail)+as.numeric(GPS_buffer))
1098     dat <- dplyr::filter(dat, lon >= as.numeric(lon_tail)-as.numeric(GPS_buffer) & lon <=
as.numeric(lon_tail)+as.numeric(GPS_buffer))
1099     dat_clean <- unique(dat)
1100     # count
1101     dat_geoClean_count <- nrow(dat_clean)
1102     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)
1103
1104     # export as csv and rds
1105     print(paste0("saving geoCleaned ",n))
1106     if (!file.exists(paste0(clean_dir,buoy,"/"))) {dir.create((paste0(clean_dir,buoy,"/")))}
1107     # write.table(dat_clean,
paste0(clean_dir,buoy,"/","s_",buoy,"_",dat_source,"_",name_export,"_geoclean.csv"),
row.names=FALSE, col.names = TRUE, sep = ",")
1108     # saveRDS(dat_clean, file =
paste0(clean_dir,buoy,"/","s_",buoy,"_",dat_source,"_",name_export,"_geoclean.rds"))
1109
1110     # build table
1111     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))
1112
1113     colnames(dat_count) <- paste0("s_", buoy, "_",n)
1114     dat_count <- data.frame(dat_count, stringsAsFactors = FALSE)
1115     # add to table
1116     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

```

```

1162 print("saving Data Count tables")
1163 colnames(data_count)[1] <- paste0("Station ",buoy)
1164 # write.table(data_count, paste0(clean_dir,buoy,"/", "s_",buoy,"_data_counts.csv"), row.names=FALSE,
col.names = TRUE, sep = ",")
1165 data_count_name <- paste0("s_",buoy,"_ndbc_ncei_data_counts")
1166 assign(data_count_name, data_count)
1167
1168 #-----
1169 # exporting GeoCleaned datasets
1170 #-----
1171
1172 if(switch_set_a == 1){
1173   print("Exporting Comp and GeoCleaned data")
1174
1175   ndbc_list <- ls(pattern = "_ndbc_"); ncei_list <- ls(pattern = "_ncei_")
1176   ndbc_list <- ndbc_list[!grepl("color_", ndbc_list)]
1177   ndbc_list <- ndbc_list[grepl("_geoClean", ndbc_list)]
1178   print(paste0("NDBC datasets :", ndbc_list))
1179   print("")
1180   ncei_list <- ncei_list[grepl("_geoClean", ncei_list)]
1181   print(paste0("NCEI datasets :", ncei_list))
1182
1183
1184   # export and save geoClean dataset
1185   if (!file.exists(paste0(clean_dir,buoy,"/"))) {dir.create((paste0(clean_dir,buoy,"/")))}
1186   for(g in c(ndbc_list,ncei_list)){
1187     if(grepl("_stdmet",g)==TRUE){
1188       print(g)
1189       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"))
1190     }
1191   }
1192   # export to RData
1193   save(list = ndbc_list, file = paste0(clean_dir,buoy,"/s_",buoy,"_ndbc_comp_geoClean.RData"))
1194   save(list = ncei_list, file = paste0(clean_dir,buoy,"/s_",buoy,"_ncei_comp_geoClean.RData"))
1195 }else{print("switch set to don't export geoClean datasets")}
1196
1197 rm(ndbc_list,ncei_list, ndbc, ncei, dat_metadata, ncei_metadata)
1198
1199 #-----
1200 # stats and plotting data
1201 #-----
1202
1203
1204 if(switch_set == 1){
1205
1206   # if one or more dataset is present but may have different time periods
1207   print("Working on stats and plots...")
1208
1209   data_lists <- ls(pattern = buoy)

```



```

1210 # remove preNDBC, station metadata, data counts and original netcdf data
1211 data_lists <- data_lists[!grepl("_preNDBC_", data_lists)]
1212 data_lists <- data_lists[!grepl("_station_metadata", data_lists)]
1213 data_lists <- data_lists[!grepl("_ndbc_ncei_data_counts", data_lists)]
1214 data_lists <- data_lists[!grepl("_netcdf", data_lists)]
1215 print(data_lists)
1216
1217 #-----
1218 # aggregating and plotting
1219 #-----
1220 if (!file.exists(paste0(stats_dir,buoy,"/"))) {dir.create((paste0(stats_dir,buoy,"/")))}
1221 if (!file.exists(paste0(fig_dir,buoy,"/"))) {dir.create((paste0(fig_dir,buoy,"/")))}
1222
1223 for(df in data_lists){
1224   print(df)
1225
1226   if(length(grep("_sensor_output",df))>0){
1227     dat_s <- get(df)
1228     png(paste0(fig_dir, paste0(buoy,"/",df,".png")), width = width1, height = height1)
1229     mainTitle <- paste0("NDBC Station ", buoy, " NCEI Sensor Output")
1230     plot(dat_s$DateTime, dat_s$sensor_output,
1231          xlab = "Date", ylab = "1 = Displacement / 2 = Acceleration",
1232          lty = 5, pch = 0, col = "skyblue2", cex = 1, cex.lab = 1.2,
1233          panel.first=grid(), main = mainTitle)
1234     dev.off()
1235     print(paste0("printing... ",mainTitle))
1236     rm(dat_s)
1237   }else if(length(grep("_stdmet",df))>0){
1238     # GPS data
1239     print("working on GPS plot...")
1240
1241     # get data
1242     dat <- get(df)
1243
1244     if(dim(dat)[1]>0){
1245
1246       # # formatting datasets
1247       if(length(grep("_stdmet",df))>0){
1248         # library(tidyverse)
1249         dat <- dat %>% dplyr::select(-contains("_metadata"))
1250         if(grepl("_ncei",df)==TRUE){
1251           # reducing primary and secondary data to one column
1252           var_names <- names(dat[4:ncol(dat)])
1253           var_names <- var_names[grepl("_1", var_names)]
1254           # replace missing primary data with secondary data for plotting and stats
1255           for(r in var_names){
1256             for(q in 1:nrow(dat)){
1257               if(is.na(dat[q,r]) & !is.na(dat[q,gsub("_1", "_2", r)])){
1258                 dat[q,r] <- dat[q,gsub("_1", "_2", r)]
1259                 dat[q,gsub("_1", "_2", r)] <- NA

```

```

1260     }
1261     if(!is.na(dat[q,r])){
1262         dat[q,gsub("_1", "_2", r)] <- NA
1263     }
1264 }
1265 }
1266 # remove columns if empty
1267 dat <- dat[,colSums(is.na(dat))<nrow(dat)]
1268 # remove _1 from col names
1269 colnames(dat) <- sub("_1","",colnames(dat))
1270 }
1271 }
1272
1273 if("lat" %in% names(dat)){
1274     # compute the bounding box
1275     library(ggplot2)
1276     library(ggmap)
1277     library(maps)
1278     library(mapdata)
1279
1280     #-----
1281     gps <- dplyr::select(dat, lat,lon)
1282     gps <- gps[complete.cases(gps),]
1283
1284     bc_bbox <- make_bbox(lat = lat, lon = lon, data = gps)
1285     bc_bbox
1286     tryCatch({
1287         # grab the maps from google
1288         bc_big <- get_map(location = bc_bbox, source = "google", maptype =
1289             "satellite")
1290         #> Warning: bounding box given to google - spatial extent only approximate.
1291         #> converting bounding box to center/zoom specification. (experimental)
1292         #> Map from URL :
1293         http://maps.googleapis.com/maps/api/staticmap?center=51.86665,-127.98475&zo
1294         om=6&size=640x640&scale=2&maptype=terrain&language=en-EN&sensor=false
1295         # maptype = c("terrain","terrain-background", "satellite", "roadmap",
1296         "hybrid", "toner",
1297         # "watercolor", "terrain-labels", "terrain-lines",
1298         "toner-2010",
1299         # "toner-2011", "toner-background", "toner-hybrid",
1300         "toner-labels",
1301         # "toner-lines", "toner-lite")
1302         # source = c("google", "osm", "stamen")
1303
1304         # ggplot code
1305         bp <- ggmap(bc_big) +
1306             geom_point(data = ncei_positions, mapping = aes(x = lon, y = lat, color
1307                 = color_ndbc_raw), size = 2) +
1308             geom_point(data = dat, mapping = aes(x = lon, y = lat, color =

```

```

1301         color_ndbc_orig), size = 2)
1302
1303     # plotting
1304     b1 <- bp + labs(title = paste0("Station ",buoy," NDBC GPS Positions\n"),
1305                    x = "Longitude", y = "Latitude", color = "Data Sources") +
1306       scale_color_hue(labels = c("NCEI NetCDF Raw", "NDBC GeoCleaned
1307                                Data"))#, values = c(color_ndbc_raw, color_ndbc_orig))
1308     # export the plot
1309     tiff(paste0(gps_dir,
1310                paste0("s_",buoy,"_Raw_vs_geoClean_NDBC_GPSPositions.tiff")), width =
1311         width, height = height, res = res)
1312     b1
1313     dev.off()
1314     # png
1315     png(paste0(gps_dir,
1316                paste0(df,"_Raw_vs_geoClean_NDBC_GPSPositions_ggplot.png")), width =
1317         width, height = height, res = res)
1318     b1
1319     dev.off()
1320     rm(bp,b1)
1321   }, error = function(e){
1322     print("no map box plot")
1323   })
1324
1325   ## plot positions overall
1326   png(paste0(gps_dir, paste0(df,"_Raw_vs_geoClean_NDBC_GPSPositions.png")), width
1327       = width, height = height, res = res)
1328   mainTitle <- paste0("Station ",buoy," NDBC GPS Positions")
1329   print(paste0("printing... ",mainTitle))
1330   plot(ncei_positions$lon, ncei_positions$lat, col = color_ndbc_raw,
1331        main = mainTitle, ylab = "Latitude (N)", xlab = "Longitude (W)")
1332   points(dat$lon, dat$lat, col = color_ndbc_orig, pch = 1)
1333   par(xpd=TRUE)
1334   legend("topright", inset=c(0,-0.085),
1335        box.lty = 0, legend=c("NCEI NetCDF data","NDBC GeoClean data"),
1336        col=c(color_ndbc_raw, color_ndbc_orig),
1337        pch = c(1,1),
1338        cex=1)
1339   dev.off()
1340 }else{print(paste0("no gps data in ",df))}
1341
1342 # standard meteorological data
1343 print("working on stdmet...")
1344
1345 # set columns of interest list
1346 dataCols <-
1347 c("wind_direction","wind_speed","wind_gust","significant_wave_height","dominant_wave_
1348    period",
1349
1350    "average_wave_period","mean_wave_direction","air_pressure_at_sea_level"

```

```

1340         , "air_temperature", "sea_surface_temperature")
1341 # remove `air_pressure_at_sea_level` in dataCols2
1342 dataCols_2 <- dataCols[-8]
1343
1344 dat_stats <- dat
1345 dat_stats$lat <-NULL; dat_stats$lon <-NULL; dat_stats$DateTime <- NULL
1346 stats_all <- data.frame(do.call(cbind, lapply(dat_stats, summary)),
1347 stringsAsFactors = FALSE)
1348 # adding in a NA row if no NA's present in dataset (and therefore not added during
1349 summary, which throws off row names)
1350 if("NA's" %in% rownames(stats_all)==FALSE){stats_all[nrow(stats_all)+1,] <- 0}
1351 stats_all[nrow(stats_all)+1,] <- nrow(dat_stats)
1352 rownames(stats_all) <-
1353 c("Min", "1st_Qu", "Median", "Mean", "3rd_Qu", "Max", "NA", "Total_Count")
1354 stats_all <- signif(stats_all, 3)
1355 stats_all[nrow(stats_all)-1, 1:ncol(stats_all)] <-
1356 trunc(stats_all[nrow(stats_all)-1, 1:ncol(stats_all)])
1357 stats_all[nrow(stats_all), 1:ncol(stats_all)] <-
1358 trunc(stats_all[nrow(stats_all), 1:ncol(stats_all)])
1359 write.csv(stats_all, paste0(stats_dir, buoy, "/", df, "_stats.csv"), row.names=TRUE)
1360 rm(dat_stats, stats_all)
1361
1362 #-----
1363 ----
1364 # plotting and stats
1365
1366 #-----
1367 ----
1368
1369 for(i in dataCols_2){
1370   # options(warn = -1)
1371
1372   print(paste0("Working on... " , i))
1373
1374   if(i %in% names(dat)){
1375     # print("plotting")
1376     library(dplyr)
1377     dat_df <- dplyr::select(dat, DateTime, all_of(i))
1378     # checking for empty columns
1379     if(sum(is.na(dat_df[,2])) != nrow(dat_df)){
1380       # export plot
1381       source(paste0(data_dir, "plot_stdmet.R"))
1382       png(paste0(fig_dir, buoy, "/", df, "_", i, ".png"), width = width,
1383           height = height, res = res)
1384       plot_stdmet(dat_df, i, buoy)
1385       dev.off()
1386     }else{print(paste0(i, " data are all NA for ", buoy))}
1387     rm(dat_df)
1388
1389   }else{print(paste0("no ", i, " data for ", buoy))}

```

```

1379         tryCatch({
1380             dev.off()
1381         },error=function(cond) {
1382             message("")
1383         })
1384     }
1385 }
1386 rm(dat)
1387 }
1388
1389 }else{
1390     #-----
1391     # stats
1392     #-----
1393     print(paste0("Working on... " , df))
1394
1395     # print("plotting")
1396     dat <- get(df)
1397     if(unique(sapply(dat[,2:ncol(dat)], class))=="character"){
1398         dat[,2:ncol(dat)] <- sapply(dat[,2:ncol(dat)],as.numeric)
1399     }
1400     # plot_stats_name <- gsub("_geoClean", "",df)
1401     plot_stats_name <- df
1402
1403     if(dim(dat)[1]>0){
1404         # descriptive stats
1405         if("lat" %in% names(dat)){
1406             stats <- as.data.frame.matrix(summary(dat[,4:ncol(dat)]), stringsAsFactors
1407                 = FALSE)
1408         }else{
1409             stats <- as.data.frame.matrix(summary(dat[,2:ncol(dat)]), stringsAsFactors
1410                 = FALSE)
1411         }
1412         write.csv(stats,paste0(stats_dir,
1413             buoy,"/",plot_stats_name,"_stats.csv"),row.names=FALSE)
1414         rm(stats)
1415
1416         #-----
1417         # aggregating and plotting
1418         #-----
1419         library(lubridate)
1420         # accounting for single years of comparison data- daily, month or yearly grouping
1421         if(length(unique(year(dat$DateTime))) >= 2){
1422             agg_period <- "annual"
1423             dat$grp <- year(dat$DateTime)
1424         }

```

```

1422     if(length(unique(year(dat$DateTime))) < 2){
1423         if(length(unique(month(dat$DateTime))) <= 1){
1424             agg_period <- "daily"
1425             year_plot <- unique(year(dat$DateTime))
1426             dat$grp <- day(dat$DateTime)
1427         }else{
1428             agg_period <- "monthly"
1429             year_plot <- unique(year(dat$DateTime))
1430             dat$grp <- month(dat$DateTime)
1431         }
1432     }
1433     # descriptive stats
1434     if("lat" %in% names(dat)){
1435         dat <- aggregate(dat[, 3:ncol(dat)-1], list(dat$grp), mean, na.rm = TRUE)
1436     }else {
1437         dat <- aggregate(dat[, 2:ncol(dat)-1], list(dat$grp), mean, na.rm = TRUE)
1438     }
1439
1440     if("lat" %in% names(dat)){
1441         dat$lat <- NULL; dat$lon <- NULL
1442     }else{print(paste0("no lat/lon in ",df))}
1443     if("DateTime" %in% names(dat)){
1444         dat$DateTime <- NULL
1445     }else{print(paste0("no DateTime in ",df))}
1446
1447     # reformat for plotting purposes
1448     dat1 <- dat
1449     is.nan.data.frame <- function(x) do.call(cbind, lapply(x, is.nan))
1450     dat1[is.nan.data.frame(dat1)] <- 0
1451     dat1 <- data.frame(t(dat1), stringsAsFactors = FALSE)
1452     names(dat1) <- dat1[1,]
1453     col_name <- names(dat1)
1454     dat1 <- data.frame(dat1[-1,], stringsAsFactors = FALSE)
1455     names(dat1) <- col_name
1456     library(dplyr)
1457     library(tibble)
1458     dat1 <- tibble::rownames_to_column(dat1, "Frequency")
1459     if(agg_period == "daily" | agg_period == "monthly"){
1460         if(agg_period == "monthly"){
1461             colnames(dat1) <-
1462                 c("Frequency", month.abb[as.numeric(gsub("X", "", names(dat1[2:ncol(dat1)])))]
1463                 )
1464         }else{
1465             colnames(dat1) <- c("Frequency", gsub("X", "", names(dat1[2:ncol(dat1)])))
1466         }
1467     }
1468     dat1$Frequency <- as.numeric(gsub("X", "", dat1$Frequency))
1469     dat1$Frequency <- signif(dat1$Frequency, digits = 6)
1470     if(agg_period != "monthly"){
1471         names_dat1 <- as.numeric(gsub("X", "", names(dat1[2:ncol(dat1)])))

```

```

1470         colnames(dat1) <- c("Frequency",names_dat1)
1471     }
1472     # export as csv
1473     write.table(dat1,
1474                 paste0(stats_dir,buoy,"/",plot_stats_name,"_",agg_period,"_mean.csv"),
1475                 row.names=FALSE, col.names = TRUE, sep = ",")
1476
1477     # formatting for plots
1478     if(agg_period == "daily"){year_num <- names(dat1); year_num <-
1479     as.numeric(year_num[-1])}
1480     if(agg_period == "monthly"){year_num <- names(dat1);year_num <- year_num[-1]}
1481     if(agg_period == "annual"){year_num <- names(dat1);year_num <-
1482     as.numeric(year_num[-1])}
1483     print(year_num)
1484
1485     for (year1 in year_num){
1486         source(paste0(data_dir,"plots_spec.R"))
1487         plot_titles <- paste0(plot_stats_name,"_mean")
1488         library(R.utils)
1489         if(unlist(strsplit(plot_titles,"_"))[3]=="ncei") {main_titles <-
1490         gsub("Ncei","NCEI",gsub("S ","NDBC Station ",str_to_title(gsub("_"," ",
1491         plot_titles))))}
1492         if(unlist(strsplit(plot_titles,"_"))[3]=="ndbc") {main_titles <-
1493         gsub("Ndbc","NDBC",gsub("S ","NDBC Station ",str_to_title(gsub("_"," ",
1494         plot_titles))))}
1495         if(agg_period == "monthly"){
1496             png(paste0(fig_dir,
1497                 paste0(buoy,"/",plot_titles,year1,"_",year_plot,".png")), width = width1,
1498                 height = height1)
1499             mainTitle <- paste0(main_titles," ",year1," ",year_plot)
1500             plots_spec(dat1, year1, df, mainTitle)
1501             dev.off()
1502             print(paste0("printing... ",mainTitle))
1503         }else if(agg_period == "daily"){
1504             png(paste0(fig_dir,
1505                 paste0(buoy,"/",plot_titles,year1,"_",year_plot,".png")), width = width1,
1506                 height = height1)
1507             mainTitle <- paste0(main_titles," ",year1," ",year_plot)
1508             plots_spec(dat1,year1, df, mainTitle)
1509             dev.off()
1510             print(paste0("printing... ",mainTitle))
1511         }else{
1512             png(paste0(fig_dir, buoy,"/",plot_titles,year1,".png"), width = width1,
1513                 height = height1)
1514             mainTitle <- paste0(main_titles," ",year1)
1515             plots_spec(dat1,year1, df, mainTitle)
1516             dev.off()
1517             print(paste0("printing... ",mainTitle))
1518         }
1519     }
1520 }

```

```

1507
1508         }else{print(paste0(df, " is an empty dataset"))}
1509         rm(dat1)
1510     } # end plot individual
1511 }
1512 #-----
1513
1514 }else{print("switch set to don't plot this time")}
1515
1516 #-----
1517
1518 # housekeeping
1519 rm(list = ls(pattern = buoy))
1520 rm(ncei_positions)
1521 rm(data_count,a,c,d,i,n,g,p,name1,name2,r,var_index,idcols,index,j,lat_mean,
1522 lat_tail,lon_mean,lon_tail,dir_column,q,dat_source,t,
1523
1524     dat_geoClean_start,dat_geoClean_end,met,ncei_dat_ls,ncei_ls,ndbc_ls,ndbc_name,ndbc_ord,dat_list,dat_
1525     ls,column_names,single_sensor_ls,
1526
1527     start_date_ncei,start_date_ndbc,remove,dat_orig_end,dat_orig_start,col_index,col_names,colNames,outl
1528     ier_table,var_list,depth,hull,
1529     mooring,payload,gps,agg_period,bc_bbox,bc_big,col_name,dat_name,data_lists,dataCols,
1530     dataCols_2,df,main,main_titles,
1531     mainTitle,name_net,names_dat1,ndbc_col_num,ncei_col_num,plot_stats_name,plot_titles,
1532     var,var_names,year_num,xlab,year_num,year1)
1533 if(exists("ncei_pre")){rm(ncei_pre)}
1534
1535 # start writing to an output file
1536 print(paste0("Finishing with geoClean: ",buoy))
1537 # sink()
1538
1539 }else{print("no new buoy data")}
1540
1541 # print(paste0("Finishing with geoClean: ",buoy))
1542 }
1543 # #-----
1544 # #-----
1545
1546
1547
1548
1549

```



1550  
1551  
1552  
1553  
1554  
1555  
1556  
1557



US Army Corps  
of Engineers®

plot\_stdmet.R

```

1 plot_stdmet <- function(dat = "dataset", i = "i", buoy = "buoy"){
2
3     ## set plot types
4     lty_main <- 5
5     pch_main <- 2
6
7     # inputs for main title and ylabel
8     xlablist <- c("Significant Wave Height", "Average Period", "Dominant Period", "Mean Wave Direction",
9                  "Wind Speed", "Wind Direction", "Wind Gust Speed", "Sea Level Pressure",
10                 "Air Temperature", "Water Temperature", "Dew Point Temperature")
11
12     ylablist <- c("WVHGT (m)", "AVGPD (s)", "DOMPD (s)", paste("MWDIR (", degree, "°)",
13                  "WSPD (m/s)", paste0("WDIR (", degree, "°)", "GST (m/s)", "BARO (hPa)",
14                  paste0("ATMP (", degree, "°C)", paste0("WTMP (", degree, "°C)", paste0("DEWP (", degree, "°C)"))
15     #-----
16     ## NDBC / NCEI inputs set
17     #-----
18     if(i=="significant_wave_height"){main = xlablist[1] ; ylab = ylablist[1]}
19     if(i=="average_wave_period" ){main = xlablist[2] ; ylab = ylablist[2]}
20     if(i=="dominant_wave_period" ){main = xlablist[3] ; ylab = ylablist[3]}
21     if(i=="mean_wave_direction" ){main = xlablist[4] ; ylab = ylablist[4]}
22     if(i=="wind_speed"){main = xlablist[5] ; ylab = ylablist[5]}
23     if(i=="wind_direction"){main = xlablist[6] ; ylab = ylablist[6]}
24     if(i=="wind_gust" ){main = xlablist[7] ; ylab = ylablist[7]}
25     if(i=="air_pressure_at_sea_level"){main = xlablist[8] ; ylab = ylablist[8]}
26     if(i=="air_temperature"){main = xlablist[9] ; ylab = ylablist[9]}
27     if(i=="sea_surface_temperature"){main = xlablist[10]; ylab = ylablist[10]}
28     if(i=="dew_point_temperature"){main = xlablist[11]; ylab = ylablist[11]}
29     #-----
30
31     # plot
32     plot(dat[[1]], dat[[2]], main = paste0("NDBC Station ", buoy, " ", main), xlab = "Date", ylab = ylab,
33          col = "darkblue", type = type, pch = pch, cex = cex)#, xaxt="n") #lwd = lwd,
34     if(i == "significant_wave_height") {abline(v= 0, h = 0.25, col = "black", lty = 2)}
35     # my.grid(z, "years", "%Y")
36     grid()
37 }

```



US Army Corps  
of Engineers®

plots\_spec.R

```

1  plots_spec <- function(dat = "dat1", year1 = "year1", df = "df", mainTitle = "mainTitle"){
2      par(mar=par1)
3
4      dat_index <- which(names(dat)==year1)
5      type_dat <- unlist(strsplit(df, "_"))[4]
6      freq1 <- unlist(strsplit(df, "_"))[6]
7
8      dir_type <- c("alpha1", "alpha2", "r1", "r2", "gamma2", "gamma3", "a1", "a2", "b1", "b2", "phih", "rhq")
9      c11_type <- c("c11", "c11m", "C12", "Q12", "C13", "Q13", "C22", "C23", "C33", "Q23")
10
11     if(unlist(strsplit(type_dat, "_"))[1] %in% dir_type){
12         if(sum(dir_type == "a1" | dir_type == "a2" | dir_type == "b1" | dir_type == "b2" | dir_type ==
13             "phih" | dir_type == "rhq")>0){
14             ylab = type_dat
15             ylim <- c(floor(min(dat[, dat_index], na.rm = TRUE)), ceiling(max(dat[, dat_index], na.rm =
16                 TRUE)))
17         }else{
18             ylab = paste0("Directional Spectrum (", degree, ")")
19             ylim <- c(floor(min(dat[, dat_index], na.rm = TRUE))-20, ceiling(max(dat[, dat_index], na.rm =
20                 TRUE))+20)
21         }
22         if(freq1 == "new"){xlim = c(0, 0.5)}
23         else if(freq1 == "old"){xlim = c(0, 0.4)}
24         else{xlim = c(0, 0.5)}
25     }
26
27     }else if(unlist(strsplit(type_dat, "_"))[1] %in% c11_type){
28         if(unlist(strsplit(type_dat, "_"))[1] == "c11"){ylab = expression('C'[11]*' (m'^2*/Hz) '['mean']')}
29         else if(unlist(strsplit(type_dat, "_"))[1] == "c11m"){ylab = expression('C'[11]*' 'm*'
30             (m/s'^2*') '^2*/Hz '['mean']')}
31         else if(unlist(strsplit(type_dat, "_"))[1] == "C12"){ylab = expression('C'[12]*' (m/Hz) '['mean']')}
32         else if(unlist(strsplit(type_dat, "_"))[1] == "Q12"){ylab = expression('Q'[12]*' (m/Hz) '['mean']')}
33         else if(unlist(strsplit(type_dat, "_"))[1] == "C13"){ylab = expression('C'[13]*' (m/Hz) '['mean']')}
34         else if(unlist(strsplit(type_dat, "_"))[1] == "Q13"){ylab = expression('Q'[13]*' (m/Hz) '['mean']')}
35         else if(unlist(strsplit(type_dat, "_"))[1] == "C22"){ylab = expression('C'[22]*' (1/Hz) '['mean']')}
36         else if(unlist(strsplit(type_dat, "_"))[1] == "C23"){ylab = expression('C'[23]*' (1/Hz) '['mean']')}
37         else if(unlist(strsplit(type_dat, "_"))[1] == "C33"){ylab = expression('C'[33]*' (1/Hz) '['mean']')}
38         else if(unlist(strsplit(type_dat, "_"))[1] == "Q23"){ylab = expression('Q'[23]*' (1/Hz) '['mean']')}
39         ylim <- c(floor(min(dat[, dat_index], na.rm = TRUE))-0.2, ceiling(max(dat[, dat_index], na.rm =
40             TRUE))+0.2)
41         if(freq1 == "new"){xlim = c(0, 0.5)}
42         else if(freq1 == "old"){xlim = c(0, 0.4)}
43         else{xlim = c(0, 0.5)}
44     }
45
46     if(ylim[1] != Inf){
47         # dat data
48         if(unlist(strsplit(type_dat, "_"))[1] %in% dir_type){
49             plot(dat$Frequency, dat[, dat_index],
50                 ylab = ylab, xlab = "Frequency (Hz)",

```

```

46         lty = 5, pch = 0, col = "skyblue2", cex = 1, cex.lab = 1.2,
47         xaxt="n", xlim = xlim, ylim = ylim, panel.first=grid(), main = mainTitle)
48     }
49     if(unlist(strsplit(type_dat,"_"))[1] %in% c11_type){
50         plot(dat$Frequency, dat[,dat_index],
51             ylab = ylab, xlab = "Frequency (Hz)",
52             type = "o", lty = 5, pch = 0, col = "skyblue2", cex = 1, cex.lab = 1.2,
53             xaxt="n", yaxt="n", xlim = xlim,ylim = ylim, panel.first=grid(), main = mainTitle)
54     }
55
56     # axes
57     axis(1, seq(xlim[1],xlim[2], by = 0.05))
58     if(unlist(strsplit(type_dat,"_"))[1] %in% c11_type){
59         tryCatch(
60             {aty <- seq(ylim[1],ylim[2],by = 0.5)
61               labels <- sapply(aty,function(i)
62                 as.expression(bquote(10^ .(i))))
63             },
64             axis(2,at=aty,labels=labels, las=1)
65         ),
66         error = function(cond){
67             aty <- c(ylim[1],mean(ylim[2]-ylim[1]),ylim[2])
68             labels <- sapply(aty,function(i)
69               as.expression(bquote(10^ .(i))))
70             }
71             axis(2,at=aty,labels=labels, las=1)
72         }
73     )
74 }
75 }
76 }
77
78
79
80

```



US Army Corps  
of Engineers®

create\_best\_data\_5.R

```

1 create_best_data_5 <- function(buoys = "list of buoys", data_dir = "data dir"){
2
3     ##-----
4     ## script to select the best available data from the compared, geographically quality controlled and
5     ## self-describing datasets
6     ## Hall, Candice
7     ##-----
8
9     ## Actions:
10    ## The 'create_best_data_5.R' script loads the 's_buoy#_ndbc/ncei_comp_geoClean.Rdata' container files created
11    ## in the previous step.
12    ## The script tests for any pre-NDBC data that may have been included within the NCEI data sources and reduces
13    ## any multiple sensor data to a single, best value for each variable.
14    ## The script then selects all available NDBC data files, and any non-common NCEI data files for inclusion in
15    ## the best available dataset.
16    ## The selected datasets with self-describing metadata are saved in 's_buoy#_best_data.Rdata' container file
17    ## within buoy station specific folders.
18
19    #-----
20    #-----
21    ## load libraries (local run)
22    library(lubridate)
23    library(dplyr)
24    library(gridExtra)
25    library(data.table)
26    library(oce)
27    library(naniar)
28    library(tidyverse)
29    library(broom)
30    library(openair)
31
32    ## load libraries (HPC run)
33    # library(lubridate, lib="/p/home/candice/Rlibs/")
34    # library(dplyr, lib="/p/home/candice/Rlibs/")
35    # library(gridExtra, lib="/p/home/candice/Rlibs/")
36    # library(data.table, lib="/p/home/candice/Rlibs/")
37    # # library(oce, lib="/p/home/candice/Rlibs/")
38    # library(naniar, lib="/p/home/candice/Rlibs/")
39    # library(tidyverse, lib="/p/home/candice/Rlibs/")
40    # library(broom, lib="/p/home/candice/Rlibs/")
41    # library(openair, lib="/p/home/candice/Rlibs/")
42
43    ##-----
44    ## set paths
45    ##-----
46    # drive <- "E:/Candice/"
47    # drive <- "/p/work/candice/"
48
49    setwd(data_dir)

```



```

46 # set input directories
47 geoClean_dir <- paste0(data_dir,"geoClean_data/data/")
48
49 # set new output directories for datasets, stats and figures
50 if (!file.exists(paste0(data_dir,"best_data/"))) {dir.create((paste0(data_dir,"best_data/")))}
51 best_data_dir <- paste0(data_dir,"best_data/")
52
53 if (!file.exists(paste0(best_data_dir,"data/"))) {dir.create((paste0(best_data_dir,"data/")))}
54 best_dir <- paste0(best_data_dir,"data/")
55
56 ##-----
57 ## set set parameters common to all plots
58 ##-----
59 xlab = "Date"      # label for x axis
60 width = 2000      # width of exported plot
61 height = 1500     # height of exported plot
62 res = 300
63 # set plot parameters
64 width1 = 1000
65 height1 = 700
66 par1 = c(5,5,4,4)
67
68 # colors for each buoy
69 color_sd <- "red"
70 color_ref <- "black"
71 color_test <- "blue"
72 # plot type, symbol, size
73 type = "p"
74 pch = 20
75 lwd = 1
76 cex = 0.5
77 # set parameters
78 Delta <- '\U0394'
79 degree <- '\U00B0'
80 # my.grid function formats
81 my.format <- "%m-%d-%Y" # "%m-%Y" (long datasets) or "%m-%d-%Y" (short datasets)
82 my.period <- "weeks" # "months" (long datasets) or "weeks" (short datasets)
83 # my grid function for plots
84 my.grid <-function(dataset, my.period = "year", my.format = "%Y"){
85     grid(nx=NA, ny=NULL)
86     abline(v=axis.POSIXct(1, at=seq(min(dataset[1,1]), max(dataset[nrow(dataset),1]),
87                                     by= my.period), format=my.format),
88           col = "lightgray", lty = "dotted", lwd = par("lwd"))
89 }
90 # function to capitalize string
91 simpleCap <- function(x) {
92     s <- strsplit(x, " ")[[1]]
93     paste(toupper(substring(s, 1,1)), substring(s, 2),
94           sep="", collapse=" ")
95 }

```

```

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 <-
132             secondary_sensor_ls[!grepl('_metadata_', secondary_sensor_ls)]
133             if("wind_direction_2" %in% col_names){if(sum(is.na(dat$wind_direction_2)) ==
134             nrow(dat)){dat$wind_direction_2 <-NULL; dat$wind_direction_metadata_2 <- NULL; dat <-
135             dplyr::rename(dat, wind_direction=wind_direction_1,
136             wind_direction_metadata=wind_direction_metadata_1)
137             }else{for(i in 1:nrow(dat)){if(is.na(dat$wind_direction_1[i]) &
138             !is.na(dat$wind_direction_2[i])){dat$wind_direction_1[i]=dat$wind_direction_2[i];
139             dat$wind_direction_metadata_1[i]=dat$wind_direction_metadata_2[i]
140             dat$wind_direction_2 <-NULL; dat$wind_direction_metadata_2 <- NULL;dat <- dplyr::rename(dat,
141             wind_direction=wind_direction_1, wind_direction_metadata=wind_direction_metadata_1)}}}
142             }
143             if("wind_speed_2" %in% col_names){if(sum(is.na(dat$wind_speed_2)) ==
144             nrow(dat)){dat$wind_speed_2 <-NULL; dat$wind_speed_metadata_2 <- NULL; dat <-
145             dplyr::rename(dat, wind_speed=wind_speed_1, wind_speed_metadata=wind_speed_metadata_1)

```

```

137 }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]
138 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)}}}
139 }
140 if("wind_gust_2" %in% col_names){if(sum(is.na(dat$wind_gust_2)) == nrow(dat)){dat$wind_gust_2
<-NULL; dat$wind_gust_metadata_2 <- NULL; dat <- dplyr::rename(dat, wind_gust=wind_gust_1,
wind_gust_metadata=wind_gust_metadata_1)
141 }else{for(i in 1:nrow(dat)){if(is.na(dat$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$wind_gust_metadata_2[i]
142 dat$wind_gust_2 <-NULL; dat$wind_gust_metadata_2 <- NULL; dat <- dplyr::rename(dat,
wind_gust=wind_gust_1, wind_gust_metadata=wind_gust_metadata_1)}}}
143 }
144 if("air_pressure_at_sea_level_2" %in% col_names){if(sum(is.na(dat$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)
145 }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]
146 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)}}}
147 }
148 if("air_temperature_2" %in% col_names){if(sum(is.na(dat$air_temperature_2)) ==
nrow(dat)){dat$air_temperature_2 <-NULL; dat$air_temperature_metadata_2 <- NULL; dat <-
dplyr::rename(dat, air_temperature=air_temperature_1,
air_temperature_metadata=air_temperature_metadata_1)
149 }else{for(i in 1:nrow(dat)){if(is.na(dat$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$air_temperature_metadata_2[i]
150 dat$air_temperature_2 <-NULL; dat$air_temperature_metadata_2 <- NULL; dat <- dplyr::rename(dat,
air_temperature=air_temperature_1, air_temperature_metadata=air_temperature_metadata_1)}}}
151 }
152 if("dew_point_temperature_2" %in% col_names){if(sum(is.na(dat$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)
153 }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]
154 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)}}}
155 }
156 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 <-

```

```

157     NULL; dat <- dplyr::rename(dat, sea_surface_temperature=sea_surface_temperature_1,
    sea_surface_temperature_metadata=sea_surface_temperature_1)
    }else{for(i in 1:nrow(dat)){if(is.na(dat$sea_surface_temperature_1[i]) &
158     !is.na(dat$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]
    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("Successful 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){

```

```
201         print(g)
202         write.table(get(g), paste0(best_dir,buoy,"/",g,".csv"), row.names=FALSE, col.names = TRUE, sep
                = ",")
203         saveRDS(get(g), paste0(best_dir,buoy,"/",g,".rds"))
204     }
205     # export to RData
206     save(list = best_ls, file = paste0(best_dir,buoy,"/s_",buoy,"_best_data.RData"))
207     rm_ls <- ls(pattern = buoy)
208     rm(list = rm_ls)
209
210     #-----
211
212     print(paste0("Finished with Best data: ",buoy))
213     sink()
214     print(paste0("Finished with Best data: ",buoy))
215
216     }else{print('no new buoy data')}
217
218     }
219 }
220 #-----
221
222
223
224
225
226
227
228
229
230
231
232
```



US Army Corps  
of Engineers®

# build\_thredds\_netcdf\_6.R

```

1 build_thredds_netcdf_6 <- function(buoys = "list of buoys"){
2
3     ##-----
4     ## script to create netCDF data from combined NDBC web files and NCEI netcdf files
5     ## Hall, Candice
6     ##-----
7
8     ## netCDF structure is cf compliant as per netCDF NCEI point reference:
9     ## https://www.nodc.noaa.gov/data/formats/netcdf/v2.0/point.cdl
10    ## cf compliant standard names:
11    ## https://cfconventions.org/Data/cf-standard-names/78/build/cf-standard-name-table.html
12
13    ## Actions:
14    ## 1. Sets data locations
15    ## 2. This step creates monthly netCDF NDBC data files that contains the best available data and metadata
16    ##    variables that were selected in the step above.
17    ## 3. The 'build_thredds_netcdf_6.R' function requires a buoy list and a data directory. The script also
18    ##    needs access to the 'NDBC_buoy.csv' and 'NDBC_buoy_descriptions.csv' spreadsheets that are found in the
19    ##    /data directory.
20    ## 4. NetCDF files are subset to create individual station files for each month and year (year_month)
21    ## 5. NetCDF structure is cf compliant as per netCDF NCEI point reference:
22    ## https://www.nodc.noaa.gov/data/formats/netcdf/v2.0/point.cdl
23    ## 6. NetCDF standard names are cf compliant as per:
24    ## https://cfconventions.org/Data/cf-standard-names/78/build/cf-standard-name-table.html
25    ## 7. Flag conventions are consistent with: Paris. Intergovernmental Oceanographic Commission of UNESCO. 2013.
26    ##    Ocean Data Standards, Vol.3:
27    ##    Recommendation for a Quality Flag Scheme for the Exchange of Oceanographic and Marine Meteorological
28    ##    Data. (IOC Manuals and Guides, 54,
29    ##    Vol. 3.) 12 pp. (English.) (IOC/2013/MG/54-3)
30
31    #-----
32    #-----
33    # load libraries (local run)
34    library(lubridate)
35    library(plyr)
36    library(dplyr)
37    library(gridExtra)
38    library(data.table)
39    library(oce)
40    library(naniar)
41    library(tidyverse)
42    library(broom)
43    library(openair) # polarplots
44    library(plotly)
45    library(magrittr)
46    library(tidyr)
47    library(grid)
48    library(devtools)
49    library(lsr)
50    library(stringr)

```

```

45 library(RColorBrewer)
46 library(viridis)
47 library(colorRamps)
48 library(ggplot2)
49 library(ggmap)
50 library(maps)
51 library(mapdata)
52 library(broman)
53 library(ncdf4)
54
55 # # load libraries (HPC run)
56 # library(lubridate, lib="/p/home/candice/Rlibs/")
57 # library(plyr, lib="/p/home/candice/Rlibs/")
58 # library(crayon, lib="/p/home/candice/Rlibs/")
59 # library(pillar, lib="/p/home/candice/Rlibs/")
60 # library(dplyr, lib="/p/home/candice/Rlibs/")
61 # library(gridExtra, lib="/p/home/candice/Rlibs/")
62 # library(data.table, lib="/p/home/candice/Rlibs/")
63 # library(oce, lib="/p/home/candice/Rlibs/")
64 # library(naniar, lib="/p/home/candice/Rlibs/")
65 # library(tidyverse, lib="/p/home/candice/Rlibs/")
66 # library(broom, lib="/p/home/candice/Rlibs/")
67 # library(openair, lib="/p/home/candice/Rlibs/") # polarplots
68 # library(plotly, lib="/p/home/candice/Rlibs/")
69 # library(magrittr, lib="/p/home/candice/Rlibs/")
70 # library(tidyr, lib="/p/home/candice/Rlibs/")
71 # library(grid, lib="/p/home/candice/Rlibs/")
72 # library(devtools, lib="/p/home/candice/Rlibs/")
73 # library(lsr, lib="/p/home/candice/Rlibs/")
74 # library(stringr, lib="/p/home/candice/Rlibs/")
75 # library(RColorBrewer, lib="/p/home/candice/Rlibs/")
76 # library(viridis, lib="/p/home/candice/Rlibs/")
77 # library(colorRamps, lib="/p/home/candice/Rlibs/")
78 # library(ggplot2, lib="/p/home/candice/Rlibs/")
79 # library(ggmap, lib="/p/home/candice/Rlibs/")
80 # library(maps, lib="/p/home/candice/Rlibs/")
81 # library(mapdata, lib="/p/home/candice/Rlibs/")
82 # library(broman, lib="/p/home/candice/Rlibs/")
83 # library(ncdf4, lib="/p/home/candice/Rlibs/")
84
85
86 #-----
87 #-----
88 # set latest NDBC spreadsheet update from Steven DiNapoli, NDBC contract scientist
89 DiNapoli_year= 2021
90
91 ##-----
92 ## set paths
93 ##-----
94 drive <- "E:/Candice/"

```



```

95 # main dir
96 data_dir <- paste0(drive, "projects/WaveTrends/data/")
97 setwd(data_dir)
98 # set input directories
99 input_dir <- paste0(data_dir, "best_data/data/")
100 # set new output directories for datasets
101 if (!file.exists(paste0(data_dir, "best_netCDF_1/"))){dir.create((paste0(data_dir, "best_netCDF_1/")))}
102 best_netCDF_dir <- paste0("D:/best_netCDF/")
103
104 # function to perform capitalization
105 simpleCap <- function(x) {s <- strsplit(x, " ")[[1]]; paste(toupper(substring(s, 1,1)), substring(s, 2), sep="",
collapse=" ")}
106 lowerFirst <- function(x) {substr(x, 1, 1) <- tolower(substr(x, 1, 1)); x}
107
108 ##-----
109 ## set presets
110 ##-----
111
112 # setting flags
113 # Ref: Paris. Intergovernmental Oceanographic Commission of UNESCO. 2013.Ocean Data Standards,Vol.3:
114 # Recommendation for a Quality Flag Scheme for the Exchange of Oceanographic and Marine Meteorological
115 # Data. (IOC Manuals and Guides, 54, Vol. 3.) 12 pp. (English.) (IOC/2013/MG/54-3)
116 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);
117 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]."
118 flag_good = 1; flag_good_desc = "Passed documented required QC tests"
119 # flag_notEvaluated = 2; flag_notEvaluated_desc = "Used for data when no QC test performed or the information on
quality is not available"
120 flag_questionableSuspect = 3; flag_questionableSuspect_desc = "Failed non-critical documented metric or
subjective test(s)"
121 flag_bad = 4; flag_bad_desc = "Failed critical documented QC test(s) or as assigned by the data provider"
122 flag_missingData = 5; flag_missingData_desc = "Used as place holder when data are missing"
123
124 # set precision
125 miss_value <- -999.99
126 variable_prec_df <- 'double'
127 variable_prec_flg <- 'integer'
128 variable_prec_metadata <- 'char'
129 date_range_increment <- 1
130
131 # load buoy stations
132 list_ndbc <- read.csv(paste0(data_dir, "NDBC_buoys.csv"), header = TRUE)
133 list_ndbc <- dplyr::filter(list_ndbc, list_ndbc$owner == "NDBC"); list_ndbc_buoy <-
as.character(list_ndbc$station)
134 buoys <- list_ndbc_buoy; print(buoys)
135 # load buoy descriptions
136 buoy_desc <- read.csv(paste0(data_dir, "NDBC_buoy_descriptions.csv"), header = TRUE)
137 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 = "_phi_h"))
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)

```

```

187 m = sprintf("%02d", as.numeric(m))
188
189 for(df in data_ls){
190   dat <-get(df)
191   dat <-dplyr::filter(dat, year(DateTime) == as.numeric(dateRange))
192   dat <-dplyr::filter(dat, month(DateTime)== as.numeric(m))
193   new_name <- paste0("data_",as.numeric(dateRange),"_",m,"_",df)
194   if(dim(dat)[1] > 0){if(grepl("cols_",df)==FALSE){assign(new_name, dat)}}
195   rm(dat)
196 }
197 rm(new_name,df)
198
199 # Create and write a netCDF file
200
201 # set new output directories for datasets
202 if (!file.exists(paste0(best_netCDF_dir,buoy,"/"))){dir.create((paste0(best_netCDF_dir,buoy,"/")))}
203 ncname <- paste0("s_",buoy,"_best_ndbc_ncei_", as.numeric(dateRange),"_",m)
204 print(ncname)
205 ncfname <- paste0(best_netCDF_dir, buoy,"/",ncname, ".nc")
206 miss_values <- miss_value
207
208 # subset relevant datasets
209 dat_ls <- ls(pattern = paste0("_",dateRange,"_"))
210 # delete list if no stdmet data present
211 if(sum(str_count(dat_ls, "_stdmet")==0){
212   rm(list = ls(pattern = paste0("data_",dateRange,"_",m,"_s_",buoy)))
213   dat_ls <- vector()
214 }
215 print(dat_ls)
216
217 #-----
218 # load relevant datasets for dateRange
219 #-----
220 if(length(dat_ls)>0){
221   # load stdmet data
222   dat <- get(dat_ls[dat_ls %like% 'ndbc_stdmet'])
223
224   # dealing with pre_NDBC NCEI stdmet data
225   if(sum(str_count(dat_ls, "_ncei_stdmet_preNDBC_"))>0){
226     col_names <- names(dat)
227     secondary_sensor_ls <- col_names[col_names %like% '_2']; secondary_sensor_ls <-
228     secondary_sensor_ls[!grepl('_metadata_', secondary_sensor_ls)]
229     if("wind_direction_2" %in% col_names){if(sum(is.na(dat$wind_direction_2)) ==
230 nrow(dat)){dat$wind_direction_2 <-NULL; dat$wind_direction_metadata_2 <- NULL; dat <-
231 dplyr::rename(dat, wind_direction=wind_direction_1,
232 wind_direction_metadata=wind_direction_metadata_1)
233 }else{for(i in 1:nrow(dat)){if(is.na(dat$wind_direction_1[i]) &
234 !is.na(dat$wind_direction_2[i])){dat$wind_direction_1[i]=dat$wind_direction_2[i];
235 dat$wind_direction_metadata_1[i]=dat$wind_direction_metadata_2[i]
236 dat$wind_direction_2 <-NULL; dat$wind_direction_metadata_2 <- NULL;dat <-

```

```

231     dplyr::rename(dat, wind_direction=wind_direction_1,
232     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$wind_speed_metadata_2 <- NULL; dat <-
dplyr::rename(dat, wind_speed=wind_speed_1, wind_speed_metadata=wind_speed_metadata_1)
233     }else{for(i in 1:nrow(dat)){if(is.na(dat$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$wind_speed_metadata_2[i]
234     dat$wind_speed_2 <-NULL; dat$wind_speed_metadata_2 <- NULL; dat <-
dplyr::rename(dat, wind_speed=wind_speed_1,
wind_speed_metadata=wind_speed_metadata_1))}}}
235   }
236   if("wind_gust_2" %in% col_names){if(sum(is.na(dat$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)
237     }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]
238     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))}}}
239   }
240   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$air_pressure_at_sea_level_metadata_2
<- NULL; dat <- dplyr::rename(dat, air_pressure_at_sea_level=air_pressure_at_sea_level_1,
air_pressure_at_sea_level_metadata=air_pressure_at_sea_level_metadata_1)
241     }else{for(i in 1:nrow(dat)){if(is.na(dat$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$air_pressure_at_sea_level_metadata_2[i]
242     dat$air_pressure_at_sea_level_2 <-NULL;
dat$air_pressure_at_sea_level_metadata_2 <- NULL; dat <- dplyr::rename(dat,
air_pressure_at_sea_level=air_pressure_at_sea_level_1,
air_pressure_at_sea_level_metadata=air_pressure_at_sea_level_metadata_1))}}}
243   }
244   if("air_temperature_2" %in% col_names){if(sum(is.na(dat$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)
245     }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]
246     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))}}}
247   }
248   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 <-

```

```

249 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_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)]

```

```

282 }else{print(paste0("no sensor output data for ",ncname))}
283
284 # subset time variable for spec matching
285 dat_time <- dplyr::select(dat, DateTime, lat,lon)
286 dat_start_date <- as.character(dat_time$DateTime[1])
287 dat_end_date <- as.character(dat_time$DateTime[nrow(dat_time)])
288
289 # load spectral data if present
290 if(sum(str_count(dat_ls, "_freq"))>0){
291   for(df in dat_ls){
292     # df <- dat_ls[1]
293     df_name <-
294       paste0("dat_spec_",gsub("_geoClean","",unlist(strsplit(df,paste0("_",buoy,"_")))[2]))
295     dat_spec <- get(df)
296     dat_spec$lat <- NULL; dat_spec$lon <- NULL
297     # remove any old frequency columns previously added to support data ingestion
298     if(grepl("_old_",df)){
299       if("0.0100" %in% names(dat_spec)){dat_spec$`0.0100`<-NULL}
300       if("0.0200" %in% names(dat_spec)){dat_spec$`0.0200`<-NULL}
301     }
302     dat_spec <- left_join(dat_time,dat_spec, by = "DateTime")
303     assign(df_name,dat_spec)
304     assign(df,dat_spec)
305     # clear loaded data
306     rm(dat_spec, df_name)
307   }
308 }else{print(paste0("no spectral data for ",ncname))}
309 rm(dat_time)
310
311 # clear loaded data from input vector
312 dat_ls <- dat_ls[!grepl('_freq_', dat_ls)]
313 # housekeeping
314 if(length(dat_ls)==0){
315   rm(list = ls(pattern = paste0("_",dateRange,"_")))
316   rm(ncname)
317   print("all data are assimilated")
318 }else{print("EXTRA DATA - check")}
319
320 #-----
321 # prep data for netcdf creation
322 #-----
323 print("starting with time data")
324
325 # create time variable
326 data.time <- julian(dat$DateTime, origin = as.POSIXct("1900-01-01 00:00:00", tz = "UTC"))
327 tunits1 <- "days since 1900-01-01 00:00:00"
328 dimTime <- ncdim_def("time",tunits1,as.double(data.time))
329 rm(tunits1, data.time)
330
331 # 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[grepl("_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 <-
349       ncdim_def(paste0("waveFrequency_",as.character(length(wave_wpm_new))), "Hz",as.double(wave
350         _wpm_new))
351     rm(dat_spec, wave_wpm_new)
352   }
353   if(sum(str_count(freq_ls, "_freq_old"))>0){
354     # looping through all old spec to account for 0.01 Hz
355     old_spec <- freq_ls[grepl("_freq_old",freq_ls)]
356     wave_wpm_old <- vector()
357     for(old in old_spec){
358       df <- get(old)
359       if("0.0100" %in% names(df)==TRUE){df$`0.0100`<-NULL}
360       if("0.0200" %in% names(df)==TRUE){df$`0.0200`<-NULL}
361       cols_old <- names(df)
362       wave_wpm_old <- c(wave_wpm_old,cols_old)
363       rm(df)
364     }
365     wave_wpm_old <- unique(wave_wpm_old);removed <- c("DateTime","lat","lon")
366     wave_wpm_old <- wave_wpm_old[!wave_wpm_old %in% removed]
367     wave_old_length <- length(wave_wpm_old)
368     dimwave_wpm_old <-
369       ncdim_def(paste0("waveFrequency_",as.character(length(wave_wpm_old))), "Hz",as.double(wave
370         _wpm_old))
371     rm(wave_wpm_old,removed)
372   }
373   rm(old_freq, df1,old_spec, old,cols_old)
374
375   # select variables
376   var_ls <- c(var_ls,freq_ls)
377 }else(print(paste0("no freq data for ", dateRange, "_", m)))
378
379 # create variable dimensions for the netcdf file
380 for(var_df in var_ls){

```

```

377     print(var_df)
378     # set spectral constants
379     if(var_df != "lat" & var_df != "lon"){
380         if(grepl("_freq_", var_df)){
381             # load the data if its frequency
382             dat_spec <- get(var_df)
383             if(exists("wave_new_length")){name_var <- gsub("new",wave_new_length,var_df)}
384             if(exists("wave_old_length")){name_var <- gsub("old",wave_old_length,var_df)}
385             var_longname = gsub("_", " ", gsub("dat_spec_", "", name_var))
386             var_name = lowerFirst(gsub(" ", "", simpleCap(var_longname)))
387         }else{
388             var_longname = gsub("_", " ", var_df)
389             var_name = lowerFirst(gsub(" ", "", simpleCap(var_longname)))
390         }
391     }
392     miss_values <- miss_value
393     variable_prec <- variable_prec_df
394
395     # set variable constants
396     if(var_df == "lat") {var_name = "latitude"; var_units = 'degreesNorth';
397     var_longname='latitude'; variable_prec= variable_prec_df}
398     if(var_df == "lon") {var_name = "longitude"; var_units = 'degreesEast';
399     var_longname='longitude'; variable_prec= variable_prec_df}
400     if(var_df == "wind_direction"|var_df == "mean_wave_direction"|var_df ==
401     "wind_direction_metadata"|var_df == "mean_wave_direction_metadata") {var_units = 'degT';
402     variable_prec = variable_prec_flg}
403     if(var_df == "wind_speed"|var_df == "wind_gust"|var_df == "wind_speed_metadata"|var_df ==
404     "wind_gust_metadata") {var_units <- 'm/s'}
405     if(var_df == "significant_wave_height"|var_df == "significant_wave_height_metadata")
406     {var_name = "waveHs"; var_units = 'm'}
407     if(var_df == "dominant_wave_period"|var_df == "dominant_wave_period_metadata") {var_name =
408     "waveTp"; var_units = 's'}
409     if(var_df == "average_wave_period"|var_df == "average_wave_period_metadata") {var_name =
410     "waveTm"; var_units = 's'}
411     if(var_df == "air_pressure_at_sea_level"|var_df == "air_pressure_at_sea_level_metadata")
412     {var_units = 'hPa'}
413     if(var_df == "air_temperature"|var_df == "sea_surface_temperature"|var_df ==
414     "dew_point_temperature"|var_df == "air_temperature_metadata"|var_df ==
415     "sea_surface_temperature_metadata"|var_df == "dew_point_temperature_metadata") {var_units =
416     'Celsius'}
417     if(var_df == "air_pressure_at_sea_level") {var_name = "surfaceAirPressure"};if(var_df ==
418     "air_pressure_at_sea_level_metadata") {var_name = "surfaceAirPressureMetadata"}
419     if(var_df == "air_temperature") {var_name = "surfaceAirTemperature"};if(var_df ==
420     "air_temperature_metadata") {var_name = "surfaceAirTemperatureMetadata"}
421     if(var_df == "sea_surface_temperature"){var_name = "surfaceSeaTemperature"};if(var_df ==
422     "sea_surface_temperature_metadata") {var_name = "surfaceSeaTemperatureMetadata"}
423     if(var_df == "dew_point_temperature") {var_name = "surfaceDewPointTemperature"};if(var_df
424     == "dew_point_temperature_metadata") {var_name = "surfaceDewPointTemperatureMetadata"}
425     if(grepl("_metadata", var_df)==TRUE){variable_prec = variable_prec_metadata; var_units =
426     ''; 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")
444     {var_longname='sea_surface_temperature_metadata'}
445   if(var_df == "dew_point_temperature") {var_longname='dew_point_temperature'}
446   if(var_df == "dew_point_temperature_metadata")
447     {var_longname='dew_point_temperature_metadata'}
448   # set frequency bands
449   if(grepl("_new", var_df)){var_name = paste0(var_name, "_", wave_new_length, "Frequencies")}
450   if(grepl("_old", var_df)){var_name = paste0(var_name, "_", wave_old_length, "Frequencies")}
451
452   # create matrix
453   if(grepl("_freq_", var_df)){
454     # subset flag field
455     df_df <- dplyr::select(dat_spec, DateTime)
456     # format matrix df
457     dat_spec$lat <- NULL; dat_spec$lon <- NULL; dat_spec$DateTime <- NULL
458     # create matrix
459     mat_df <- matrix(t(dat_spec), nrow=nrow(dat_spec), ncol=ncol(dat_spec))
460     df_name <- paste0("df_", var_name)
461     # assign(df_name, mat_df)
462     assign(df_name, mat_df)
463     rm(df_name, mat_df)
464     # create flag field
465     df_df$flag <- flag_good
466     for(r in 1:nrow(df_df)){if(is.na(df_df[r,1])){df_df[r,2]=flag_missingData}}
467     if(grepl("_preNDBC", var_df)){df_df$flag <- flag_questionableSuspect}
468     # subset df
469     df_df_flag <- dplyr::select(df_df, flag)
470     # rename flag df
471     df_name <- paste0("df_", var_name, "_flag")
472     assign(df_name, df_df_flag)
473     rm(df_df, df_df_flag, df_name, r)
474   }else{
475     # create df
476     df_df <- dplyr::select(dat, all_of(var_df))
477     # create flag field
478     df_df$flag <- flag_good
479     for(r in 1:nrow(df_df)){if(is.na(df_df[r,1])){df_df[r,2]=flag_missingData}}
480     # subset df
481     df_df_flag <- dplyr::select(df_df, flag)
482     df_df$flag <- NULL
483     # rename df
484     df_name <- paste0("df_", var_name)
485     assign(df_name, df_df)
486     df_name <- paste0("df_", var_name, "_flag")
487     assign(df_name, df_df_flag)

```

```

486         rm(df_df, df_df_flag, df_name,r)
487     }
488
489     # Add dimensions and variables which accompany the dimensions (avoided by create_dimvar =
490     FALSE)
491     if(grepl("_metadata", var_df)){
492         vari_df <- ncvar_def(name=var_name, units=var_units, longname=var_longname,
493             dim=list(dimTime,dimnchar), prec=variable_prec)
494     }else if(var_df == "mooring"|var_df == "hull"|var_df == "payload"){
495         vari_df <- ncvar_def(name=var_name, units=var_units, longname=var_longname,
496             dim=list(dimTime,dimnchar), prec=variable_prec)
497     }else if(var_df == "lat" | var_df == "lon"){
498         vari_df <- ncvar_def(name=var_name, units=var_units, longname=var_longname,
499             dim=list(dimTime), missval=miss_values, prec=variable_prec)
500     }else if(grepl("_freq_old", var_df)){
501         vari_df <- ncvar_def(name=var_name, units=var_units, longname=var_longname,
502             dim=list(dimwave_wpm_old,dimTime), missval=miss_values, prec=variable_prec)
503     }else if(grepl("_freq_new", var_df)){
504         vari_df <- ncvar_def(name=var_name, units=var_units, longname=var_longname,
505             dim=list(dimwave_wpm_new,dimTime), missval=miss_values, prec=variable_prec)
506     }else{
507         vari_df <- ncvar_def(name=var_name, units=var_units, longname=var_longname,
508             dim=list(dimTime), missval=miss_values)
509     }
510     df_name <- paste0("var_",var_name)
511     assign(df_name, vari_df)
512     rm(df_name, vari_df)
513     rm(var_longname, var_units, var_name)
514 }
515 rm(var_df)
516 rm(list = ls(pattern = "dim_"))
517 if(exists("dat_spec")){rm(dat_spec)}
518 rm(list = ls(pattern = "dat_spec_"))
519 if(exists("wave_old_length")){rm(wave_old_length)}
520 if(exists("wave_new_length")){rm(wave_new_length)}
521
522 #-----
523 # prep flag data for netcdf file
524 #-----
525 flag_ls <- ls(pattern = "_flag")
526 miss_values <- miss_value
527
528 # function to perform capitalization
529 simpleCap <- function(x) {
530     s <- strsplit(x, " ")[[1]]
531     paste(toupper(substring(s, 1,1)), substring(s, 2),
532         sep="", collapse=" ")
533 }
534 lowerFirst <- function(x) {substr(x, 1, 1) <- tolower(substr(x, 1, 1));x}

```

```

529 # create dimensions for the flag data
530 for(fl in flag_ls){
531   var_name = gsub("df_", "", fl)
532   var_longname = gsub("_", " ", tolower(gsub("([a-z])([A-Z])", "\\1 \\2", var_name)))
533   var_longname <- gsub("frequencies", " frequencies", var_longname)
534   var_name = gsub("_flag", "Flag", var_name)
535   vari_df <- ncvar_def(name=var_name, units="none", longname=var_longname, dim=list(dimTime),
536     missval=miss_values, prec=variable_prec_flg)
537   df_name <- paste0("var_", var_name)
538   assign(df_name, vari_df)
539   rm(df_name, vari_df, var_longname, var_name)
540 }
541 rm(fl, flag_ls, dimTime)
542
543 #-----
544 # build netcdf file
545 #-----
546 rm(var__flag, var_df, var_list, var_units)
547
548 # create a list of all variables to add them all at once
549 var_list <- ls(pattern = "var_"); var_list
550 var_list <- var_list[!var_list %in% "var_ls"]; var_list
551 if("var_list" %in% var_list){var_list<- var_list[!var_list %in% "var_list"]}
552 var_list
553
554 # create list for quick add to con nc file
555 vars <- list()
556 for(i in var_list){vars[[gsub("var_", "", i)]] <- get(i)}#print(i)
557 rm(i)
558
559 # Make the file
560 # Create a new empty netcdf file.
561 con <- nc_create(ncfname, vars)#, verbose = TRUE)
562 rm(ncfname, vars)
563
564 # This variable was implicitly created by the dimension, so just specifying it by name
565 ncatt_put(con, 'time', 'standard_name', 'time')
566 ncatt_put(con, var_longitude, 'axis', 'X')
567 ncatt_put(con, var_latitude, 'axis', 'Y')
568 ncatt_put(con, 'time', 'axis', 'T')
569
570 # Add some extra attributes
571 for(var_df in var_list){
572   print(var_df)
573   std_name <- gsub("var_", "", var_df)
574   print(std_name)
575   ndbc_blurb <- '(NDBC,2018). https://www.ndbc.noaa.gov/measdes.shtml'
576   DiNapoli_blurb <- 'Source:NDBC metadata spreadsheet (DiNapoli, '
577   metadata_blurb <- paste0(DiNapoli_blurb, DiNapoli_year,') / NDBC NCEI netCDF metadata.')
578   # 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'

```

```

617         ; desc_name="r1 is the nondimensional first normalized polar coordinates of the
        Fourier coefficients"}
        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 =
670         paste0("urn:ioos:station:wmo:",as.character(buoy)), prec="char")
671         ncatt_put(nc=con,varid=0, attname="wmo_id", attval = as.character(buoy), prec="char")
672         ncatt_put(nc=con,varid=0, attname="institution", attval = "National Data Buoy Center",
673         prec="char")
674         ncatt_put(nc=con,varid=0, attname="institution_abbreviation", attval = "NDBC", prec="char")
675         ncatt_put(nc=con,varid=0, attname="title", attval = "NDBC description: Meteorological and
676         Oceanographic Data Collected from the National Data Buoy Center\'s Weather Buoys", prec="char")
677         ncatt_put(nc=con,varid=0, attname="summary", attval = "NDBC description: Over 100 moored
678         weather buoys have been deployed in U.S. coastal and offshore waters. Weather buoy data
679         typically include barometric pressure, wind direction, speed and gust, air temperature, sea
680         water temperature, waves, and relative humidity. Weather buoys also measure wave energy spectra
681         from which significant wave height, dominant wave period, average wave period and mean wave
682         direction are derived.", prec="char")
683         if(length(station_name)>0){ncatt_put(nc=con,varid=0, attname="station_name", attval =
684         station_name, prec="char")}
685         ncatt_put(nc=con,varid=0, attname="history", attval = 'The data were collected by the National
686         Data Buoy Center (NDBC) and archived on their website (https://www.ndbc.noaa.gov/) and in the
687         official National Oceanic and Atmospheric Administration (NOAA) archive at National Center for
688         Environmental Information (NCEI;
689         https://www.ncei.noaa.gov/access/marine-environmental-buoy-database/). Each data source has
690         their own idiosyncrasies (Hall and Jensen, 2021, http://dx.doi.org/10.21079/11681/40059) that
691         need to be accounted for to accurately use these NBDC data within U.S. Army Corps of Engineers
692         (USACE) Engineers and Research Development Center (ERDC) products. USACE sponsored a Coastal
693         Ocean Data Systems (CODS) National Coastal Wave Climate (NCWC) project that developed in-house
694         USACE quality control checks and metadata corrections to develop a best available measurement
695         archive (herewith called the USACE QCC measurement archive). Of note is that integral wave data
696         are imported directly from the NDBC data sources, and are not corrected for calculation errors
697         that occurred during NDBC processing from spectral wave data. The self-described, USACE QCC
698         measurement archive data is stored in netCDF format alongside the USACE Coastal and Hydraulic
699         Laboratory (CHL) Thredds Wave Information Study (WIS) long-term hindcast, accessible to both
700         the USACE and the public.', prec="char")
701         ncatt_put(nc=con,varid=0, attname="geospatial_lat_max", attval = "variable: see latitude data",
702         prec="char")
703         ncatt_put(nc=con,varid=0, attname="geospatial_lat_min", attval = "variable: see latitude data",
704         prec="char")
705         ncatt_put(nc=con,varid=0, attname="geospatial_lat_units", attval = "degrees", prec="char")
706         ncatt_put(nc=con,varid=0, attname="geospatial_lon_max", attval = "variable: see longitude
707         data", prec="char")
708         ncatt_put(nc=con,varid=0, attname="geospatial_lon_min", attval = "variable: see longitude
709         data", prec="char")
710         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"){attvall1 = "183 m above mean sea level"}
704     if(buoy == "45002"){attvall1 = "176 m above mean sea level"}
705     if(buoy == "45003"){attvall1 = "177 m above mean sea level"}
706     if(buoy == "45004"){attvall1 = "183 m above mean sea level"}
707     if(buoy == "45005"){attvall1 = "174 m above mean sea level"}
708     if(buoy == "45006"){attvall1 = "183 m above mean sea level"}
709     if(buoy == "45007"){attvall1 = "176 m above mean sea level"}
710     if(buoy == "45008"){attvall1 = "177 m above mean sea level"}
711     if(buoy == "45010"){attvall1 = "177 m above mean sea level"}
712     if(buoy == "45011"){attvall1 = "unknown"}
713     if(buoy == "45012"){attvall1 = "74.7 m above mean sea level"}
714     ncatt_put(nc=con,varid=0, attname="site_elevation", attval = attvall1, prec="char")
715 }else{ncatt_put(nc=con,varid=0, attname="site_elevation", attval = "sea level", prec="char")}

```

```

716
717         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")

718
719         #-----
720         # closing nc file
721         #-----
722         nc_close(con)
723         rm(con,dat,dat_ls, name_var)
724     }else{
725         print(paste0("no data for ", dateRange,"_",m))
726         rm(m, ncname, dat_ls)
727     }
728     } # end month range loop
729     rm(list = ls(pattern = paste0("var_")))
730     rm(list = ls(pattern = paste0("dat_")))
731     rm(list = ls(pattern = paste0("dat_spec_")))
732
733     } # end of yearly date range loop
734     print(paste0("finished build thredds on buoy: ", buoy))
735     rm(list = ls(pattern = paste0("s_",buoy)))
736     rm(buoy, data_ls, date_year_end, date_year_start, dateRange, freq_ls, std_name)
737     Sys.time()
738 } # end of buoy loop
739 }
740
741
742
743
744

```