Check COM Ports with R serial

Christopher Swingley, ABR

2022-03-25

Introduction

If you aren't seeing latitude and longitude coordinates in SeaLog at any of the COM ports Windows is displaying in the app, you may need to confirm that the GPS unit you are using is actually streaming NMEA locations. You can use a terminal program like PuTTY, or the following script, which uses the `serial` R library to communicate with serial ports on your computer.

The script itself is included with this PDF, `read_serial.R`. Open the script in RStudio and follow along by highlighting the sections shown and running them in the console.

Before running the code, you need to make sure you have the `tidyverse`, `lubridate`, and `serial` packages installed in R. You can install the latest version of all three with the code below, but if you are already successfully using QA/QSea, `serial` is probably the only package you aren't likely to already have installed.

```r
install.packages(c("tidyverse", "lubridate", "serial"))
```

Libraries and Ports

The first part of the script loads the required R packages, and then prints a list of what ports are available on your system.

```r
library(tidyverse)
library(lubridate)
options(pillar.sigfig = 8)

# See what ports are available:
listPorts()
```

```
## COM1
## COM3
```

If you get an error running `listPorts()` you may not have permission to access the serial ports, which could be why SeaLog isn't getting data. You might try logging in as a user with administrative priviledges, starting R and/or SeaLog “as an administrator”, or consulting with your IT Department to get user-level access to the serial ports.

Choosing a port

Once you have the list of ports on your system, change the value of the `port` in `serialConnection` to match the COM port you want to test. The `mode` option controls the speed used, and other connection properties. The NMEA standard sets the speed to 4800 baud, so you shouldn’t need to change this setting with any commercial GPS unit. If you are connecting to an external NMEA feed, such as may be provided by the vessel, you may need to change this to whatever baud rate they are using.

```
# Change the port here (COM1, etc.)
#
# Only change 4800 in the mode if you know for sure the NMEA stream is using a
# different (and non-standard) baud rate
con <- serialConnection(
  port = "COM1",
  mode = "4800,n,8,1"
)
```

Reading data

The script opens the serial port and attempts to read 60 seconds of data, or the first 100 lines, whichever comes first. A normal NMEA stream will typically get to 100 lines much faster than 60 seconds, so you should see results from this code very quickly.

```
open(con)

# Read from the port for 60 seconds or once we've received 100 lines of data
num_lines = 0
time_seconds = 0
all_lines <- c()
while (num_lines < 100 & time_seconds < 60) {
    data <- read.serialConnection(con)
    lines <- str_split(data, "\\$")[[1]]
    lines <- subset(lines, grepl("^GP", lines))
    print(lines)
    num_lines <<- num_lines + length(lines)
    time_seconds <<- time_seconds + 1
    all_lines <<- c(all_lines, lines)
    Sys.sleep(1)
}
```

```
## [1] "GPGGA,181528.000,6454.7439,N,14755.9877,W,1,10,0.8,200.2,M,4.7,M,,0000*7A"
## [2] "GPGSA,A,3,14,13,15,30,23,08,07,17,05,24,,,1.4,0.8,1.1*34"
## [3] "GPRMC,181528.000,A,6454.7439,N,14755.9877,W,0.00,199.96,250322,,,A*78"
## [1] "GPGGA,181529.000,6454.7439,N,14755.9877,W,1,10,0.8,200.2,M,4.7,M,,0000*7B"
## [2] "GPGSA,A,3,14,13,15,30,23,08,07,17,05,24,,,1.4,0.8,1.1*34"
## [3] "GPRMC,181529.000,A,6454.7439,N,14755.9877,W,0.00,199.96,250322,,,A*79"
## [1] "GPGGA,181530.000,6454.7439,N,14755.9877,W,1,10,0.8,200.2,M,4.7,M,,0000*73"
## [2] "GPGSA,A,3,14,13,15,30,23,08,07,17,05,24,,,1.4,0.8,1.1*34"
## [3] "GPRMC,181530.000,A,6454.7439,N,14755.9877,W,0.00,199.96,250322,,,A*71"
## [1] "GPGGA,181531.000,6454.7439,N,14755.9877,W,1,10,0.8,200.2,M,4.7,M,,0000*72"
## [2] "GPGSA,A,3,14,13,15,30,23,08,07,17,05,24,,,1.4,0.8,1.1*34"
## [3] "GPGSV,3,1,12,14,65,145,29,13,63,218,34,15,48,277,36,30,39,109,28*7A"
## [1] "GPGSV,3,2,12,23,24,324,20,08,22,034,09,07,16,102,17,17,10,149,34*74"
## [2] "GPGSV,3,3,12,05,08,227,25,24,07,275,26,28,48,189,30,57,39,268,*73"
## [3] "GPRMC,181531.000,A,6454.7439,N,14755.9877,W,0.00,199.96,250322,,,A*70"
## [4] "GPGGA,181532.000,6454.7439,N,14755.9877,W,1,10,0.8,200.2,M,4.7,M,,0000*71"
## [5] "GPGSA,A,3,14,13,15,30,23,08,07,17,05,24,,,1.4,0.8,1.1*34"
## [6] "GPRMC,181532.000,A,6454.7439,N,14755.9877,W,0.00,199.96,250322,,,A*73"
## [1] "GPGGA,181533.000,6454.7439,N,14755.9877,W,1,10,0.8,200.2,M,4.7,M,,0000*70"
## [2] "GPGSA,A,3,14,13,15,30,23,08,07,17,05,24,,,1.4,0.8,1.1*34"
```

```r
close(con)
```

Parsing NMEA

If the script received data from the port, it goes on to parse each line as though it was NMEA data. If you get to this point in the script, but errors are printed, you may have chosen a port that is sending serial data, but not NMEA data.

The last section of the code prints the 10 most recent coordinates from the GPS. If you get to that point without errors, you have found the COM port you need for SeaLog to retrieve your position data.

```r
# Was there data from the port?
if (num_lines == 0) {
  print("Error: no data resembling NMEA on this port")
} else {
  # Do we have NMEA position data (GPGGA)
  nmea_data <- tibble(raw = all_lines) %>%
    filter(grepl("GPGGA", raw))

  if (nrow(nmea_data) == 0) {
    print("Error: no NMEA data")
  } else {
    # Does the data contain GPS coordinates?
    nmea_data <- nmea_data %>%
      mutate(
        fields = str_split(raw, ","),
        hhmmss = map_chr(fields, ~ pluck(., 2)),
        latddmm = map_chr(fields, ~ pluck(., 3)),
        ns = map_chr(fields, ~ pluck(., 4)),
        londdmm = map_chr(fields, ~ pluck(., 5)),
        ew = map_chr(fields, ~ pluck(., 6)),
      ) %>%
      filter(
        str_length(latddmm) > 4,
        str_length(londdmm) > 5
      )
    if (nrow(nmea_data) == 0) {
      print("WARNING: no valid locations in data")
    } else {
      nmea_data <- nmea_data %>%
        mutate(
          ts_utc = strptime(
            paste(date(now(tzone = "UTC")), hhmmss),
            "%Y-%m-%d %H%M%OS",
            tz = "UTC"
          ),
          latdd = as.numeric(gsub("([0-9]{2}).*", "\\1", latddmm)),
          latmm = as.numeric(gsub("[0-9]{2}([0-9.]+)", "\\1", latddmm)),
          lat = if_else(
            ns == "S",
            -1 * latdd + latmm / 60.0,
            latdd + latmm / 60.0
          ),
          londd = as.numeric(gsub("([0-9]{3}).*", "\\1", londdmm)),
          lonmm = as.numeric(gsub("[0-9]{3}([0-9.]+)", "\\1", londdmm)),
          lon = if_else(
            ns == "W",
            -1 * londd + lonmm / 60.0,
            londd + lonmm / 60.0
          )
        ) %>%
        select(ts_utc, lat, lon, raw)

      # Print the 10 most recent locations
      print(nmea_data %>% tail(10))
    }
  }
}
```

Successful Results

If the port has a valid NMEA stream attached to it, you should see results similar to the following output.

```
## # A tibble: 10 x 4
##    ts_utc                    lat       lon raw
##    <dttm>                  <dbl>     <dbl> <chr>
##  1 2022-03-25 18:15:47 64.912397 147.93313 GPGGA,181547.000,6454.7438,N,14755.9~
##  2 2022-03-25 18:15:48 64.912397 147.93313 GPGGA,181548.000,6454.7438,N,14755.9~
##  3 2022-03-25 18:15:49 64.912397 147.93313 GPGGA,181549.000,6454.7438,N,14755.9~
##  4 2022-03-25 18:15:50 64.912397 147.93313 GPGGA,181550.000,6454.7438,N,14755.9~
##  5 2022-03-25 18:15:51 64.912397 147.93313 GPGGA,181551.000,6454.7438,N,14755.9~
##  6 2022-03-25 18:15:52 64.912397 147.93313 GPGGA,181552.000,6454.7438,N,14755.9~
##  7 2022-03-25 18:15:53 64.912397 147.93313 GPGGA,181553.000,6454.7438,N,14755.9~
##  8 2022-03-25 18:15:54 64.912397 147.93313 GPGGA,181554.000,6454.7438,N,14755.9~
##  9 2022-03-25 18:15:55 64.912397 147.93313 GPGGA,181555.000,6454.7438,N,14755.9~
## 10 2022-03-25 18:15:56 64.912397 147.93313 GPGGA,181556.000,6454.7438,N,14755.9~
```