RNOAA R Script for extracting ISD/DS3505 Datasets for output to CSV and netCDF

References

- https://www.ncdc.noaa.gov/isd
- ftp://ftp.ncdc.noaa.gov/pub/data/noaa/ish-tech-report.pdf

Pardon the typos; they are legion.

User expectations: Set the following variables

- target_year
- file_title_string
- name_of_station
- station_list_number

The following libraries are required

- rnoaa : API Interface for Accessing data from NOAA NCDC/NCEI
- isdparser : Tools for parsing 'NOAA' Integrated Surface Data
- lubridate : Functions to work with date-times and time-spans
- dplyr : A fast, consistent tool for working with data frame like objects
- ncdf4 : API Interface to Unidata netCDF
- openair : Tools for the Analysis of Air Pollution Data

Something to be aware of before we start.

These are an merger of several report message types, METARs (Hourlies), SPECIs (special reports that supplement the METARs), [before modern data era these were SAOs] and 3-hrly but more comprehensive SYNOPs. Most are derived for use at airports and thus have more than one cloud field (low, middle and high). The data archiving is designed to house any of the messages in one single record.

For some fields like temperature, pressure, humidity and wind speed, this isn't too much of a problem. But for other fields like cloud and precip and significant weather that will vary with the type of report message.

```
library("rnoaa")
library("isdparser")
library("lubridate")
```

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

```
## The following object is masked from 'package:base':
##
##     date
```

```
library("ncdf4")
library("dplyr")
```

```
##
## Attaching package: 'dplyr'
```

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

```
## The following objects are masked from 'package:stats':
##
```

```
##     filter, lag

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

```
library("openair")
```

To target a given station location you will need two catalog codes

The USAF Code and the WBAN code.

If you didn't work in the old Asheville Federal Building in the 80s before they moved and don't know what file cabinet the station history info is kept, there is hope.

If you have the latitude and longitude and radius in KM you can pull those fields from using the isd_station_search function.

```
stations_near_targ = isd_stations_search(lat    =   44.0444325,  # degrees_north
                                         lon    = -103.0565652,  # degrees_east
                                         radius =   10.)  # km



target_year = 2017     # add manually

file_title_string = "KRAP"
name_of_station   = "Rapid City Regional Airport"

print(stations_near_targ)
```

```
## # A tibble: 2 x 12
##   usaf   wban  station_name    ctry  state icao  latitude longitude elev_m
##   <chr>  <chr> <chr>           <chr> <chr> <chr>    <dbl>     <dbl>  <dbl>
## 1 726620 24090 RAPID CITY REG~ US    SD    KRAP      44.0      -103    963
## 2 999999 24090 RAPID CITY REG~ US    SD    KRAP      44.0      -103    966
## # ... with 3 more variables: begin <dbl>, end <dbl>, distance <dbl>
```

So for Rapid City Regional Airport (KRAP) which probably has the best reporting fidelity since it's a First Order Station for NOAA, and for your period, you will want to use the following stationID pair.

I am not sure as to the quality or reliability of Elsworth (KRCA), Custer (KCUT) or Spearfish (KSPF) since they are not affiliated with an NWS office.

```
station_list_number = 1 # indicing starts at 1   # add manually


target_usaf = stations_near_targ$usaf[station_list_number]
target_wban = stations_near_targ$wban[station_list_number]

station_lon = stations_near_targ$longitude[station_list_number]
station_lat = stations_near_targ$latitude[station_list_number]
station_alt = stations_near_targ$elev_m[station_list_number]

station_name_label = paste(name_of_station,
                           target_year)

output_file_name = paste(file_title_string,
                         target_year,
```

```
                          ".csv",
                          sep="")
```

To extract the data (and this can take a few minutes since you are digging on the NOAA servers...)

use the isd command following this example. The original data is kept in a compressed "tarball" with one tarball per year.

This also has the option for multiprocessing but you probably don't need it.

```
targ_data = isd(usaf     = target_usaf,  # your usaf number
                wban     = target_wban,  # your wban number
                year     = target_year,  # your year
                progress = TRUE)         # shows prograss as you go
```

```
## found in cache
```

```
colnames(targ_data)
```

```
##   [1] "total_chars"
##   [2] "usaf_station"
##   [3] "wban_station"
##   [4] "date"
##   [5] "time"
##   [6] "date_flag"
##   [7] "latitude"
##   [8] "longitude"
##   [9] "type_code"
##  [10] "elevation"
##  [11] "call_letter"
##  [12] "quality"
##  [13] "wind_direction"
##  [14] "wind_direction_quality"
##  [15] "wind_code"
##  [16] "wind_speed"
##  [17] "wind_speed_quality"
##  [18] "ceiling_height"
##  [19] "ceiling_height_quality"
##  [20] "ceiling_height_determination"
##  [21] "ceiling_height_cavok"
##  [22] "visibility_distance"
##  [23] "visibility_distance_quality"
##  [24] "visibility_code"
##  [25] "visibility_code_quality"
##  [26] "temperature"
##  [27] "temperature_quality"
##  [28] "temperature_dewpoint"
##  [29] "temperature_dewpoint_quality"
##  [30] "air_pressure"
##  [31] "air_pressure_quality"
##  [32] "AA1_precipitation_liquid"
##  [33] "AA1_period_quantity_hrs"
##  [34] "AA1_depth"
##  [35] "AA1_condition_quality"
##  [36] "AA1_quality_code"
##  [37] "AO2_precipitation_liquid"
```

```
##  [38] "AO2_period_quantity_minutes"
##  [39] "AO2_depth_dimension"
##  [40] "AO2_condition_code"
##  [41] "AO2_quality_code"
##  [42] "GA1_sky_cover_layer_identifier"
##  [43] "GA1_coverage_code"
##  [44] "GA1_coverage_quality_code"
##  [45] "GA1_base_height_dimension"
##  [46] "GA1_base_height_quality_code"
##  [47] "GA1_cloud_type_code"
##  [48] "GA1_cloud_type_quality_code"
##  [49] "GD1_sky_cover_summation_state_identifier"
##  [50] "GD1_coverage_code_1"
##  [51] "GD1_coverage_code_2"
##  [52] "GD1_coverage_quality_code"
##  [53] "GD1_height_dimension"
##  [54] "GD1_height_dimension_quality_code"
##  [55] "GD1_characteristic_code"
##  [56] "MA1_atmospheric_pressure"
##  [57] "MA1_altimeter_setting_rate"
##  [58] "MA1_altimeter_quality_code"
##  [59] "MA1_station_pressure_rate"
##  [60] "MA1_station_pressure_quality_code"
##  [61] "REM_remarks"
##  [62] "REM_identifier"
##  [63] "REM_length_quantity"
##  [64] "REM_comment"
##  [65] "MD1_atmospheric_change"
##  [66] "MD1_tendency"
##  [67] "MD1_tendency_quality"
##  [68] "MD1_three_hr"
##  [69] "MD1_three_hr_quality"
##  [70] "MD1_twentyfour_hr"
##  [71] "MD1_twentyfour_hr_quality"
##  [72] "AJ1_snow_depth"
##  [73] "AJ1_depth_dimension"
##  [74] "AJ1_condition_code"
##  [75] "AJ1_quality_code"
##  [76] "AJ1_equivalent_water_depth"
##  [77] "AJ1_equivalent_water_condition_code"
##  [78] "AJ1_equivalent_water_condition_quality_code"
##  [79] "AN1_snow_accumulation_day_month"
##  [80] "AN1_period_quantity"
##  [81] "AN1_depth_dimension"
##  [82] "AN1_condition_code"
##  [83] "AN1_quality_code"
##  [84] "NO5_original_observation"
##  [85] "NO5_original_value_text"
##  [86] "NO5_units_code"
##  [87] "NO5_parameter_code"
##  [88] "N10_original_observation"
##  [89] "N10_original_value_text"
##  [90] "N10_units_code"
##  [91] "N10_parameter_code"
```

```
##  [92] "KA1_extreme_temp"
##  [93] "KA1_period_quantity"
##  [94] "KA1_max_min"
##  [95] "KA1_temp"
##  [96] "KA1_temp_quality"
##  [97] "KA2_extreme_temp"
##  [98] "KA2_period_quantity"
##  [99] "KA2_max_min"
## [100] "KA2_temp"
## [101] "KA2_temp_quality"
## [102] "KG1_average_dew_point_wet_bulb_trt"
## [103] "KG1_time_quantity"
## [104] "KG1_code"
## [105] "KG1_avg_temp"
## [106] "KG1_derived_code"
## [107] "KG1_quality_code"
## [108] "KG2_average_dew_point_wet_bulb_trt"
## [109] "KG2_time_quantity"
## [110] "KG2_code"
## [111] "KG2_avg_temp"
## [112] "KG2_derived_code"
## [113] "KG2_quality_code"
## [114] "MF1_atmospheric_pressure"
## [115] "MF1_avg_pressure"
## [116] "MF1_avg_pressure_quality"
## [117] "MF1_avg_sea_level_pressure"
## [118] "MF1_avg_sea_level_pressure_quality"
## [119] "MG1_atmospheric_pressure"
## [120] "MG1_avg_pressure"
## [121] "MG1_avg_pressure_quality"
## [122] "MG1_min_sea_level_pressure"
## [123] "MG1_min_sea_level_pressure_quality"
## [124] "OE1_summary_of_day_wind_observation_identifier"
## [125] "OE1_type_code"
## [126] "OE1_period_quantity"
## [127] "OE1_speed_rate"
## [128] "OE1_direction_of_wind"
## [129] "OE1_time_of_occurrence_in_z_time_utc"
## [130] "OE1_quality_code"
## [131] "OE2_summary_of_day_wind_observation_identifier"
## [132] "OE2_type_code"
## [133] "OE2_period_quantity"
## [134] "OE2_speed_rate"
## [135] "OE2_direction_of_wind"
## [136] "OE2_time_of_occurrence_in_z_time_utc"
## [137] "OE2_quality_code"
## [138] "OE3_summary_of_day_wind_observation_identifier"
## [139] "OE3_type_code"
## [140] "OE3_period_quantity"
## [141] "OE3_speed_rate"
## [142] "OE3_direction_of_wind"
## [143] "OE3_time_of_occurrence_in_z_time_utc"
## [144] "OE3_quality_code"
## [145] "RH1_relative_humidity"
```

```
## [146] "RH1_period_quantity"
## [147] "RH1_code"
## [148] "RH1_relative_humidity_percent"
## [149] "RH1_relative_humidity_derived_code"
## [150] "RH1_relative_humidity_quality_code"
## [151] "RH2_relative_humidity"
## [152] "RH2_period_quantity"
## [153] "RH2_code"
## [154] "RH2_relative_humidity_percent"
## [155] "RH2_relative_humidity_derived_code"
## [156] "RH2_relative_humidity_quality_code"
## [157] "RH3_relative_humidity"
## [158] "RH3_period_quantity"
## [159] "RH3_code"
## [160] "RH3_relative_humidity_percent"
## [161] "RH3_relative_humidity_derived_code"
## [162] "RH3_relative_humidity_quality_code"
## [163] "AB1_precipitation_liquid_monthly_total"
## [164] "AB1_depth_dimension"
## [165] "AB1_condition_code"
## [166] "AB1_quality_code"
## [167] "AD1_precipitation_liquid_greatest_24_hrs"
## [168] "AD1_depth_dimension"
## [169] "AD1_condition_code"
## [170] "AD1_dates_occurrence_1"
## [171] "AD1_dates_occurrence_2"
## [172] "AD1_dates_occurrence_3"
## [173] "AD1_quality_code"
## [174] "AE1_precipitation_liquid_number_days_amt"
## [175] "AE1_number_days_.01inch"
## [176] "AE1_quality_code_.01inch"
## [177] "AE1_number_days_.10inch"
## [178] "AE1_quality_code_.10inch"
## [179] "AE1_number_days_.50inch"
## [180] "AE1_quality_code_.50inch"
## [181] "AE1_number_days_1inch"
## [182] "AE1_quality_code_1inch"
## [183] "AK1_snow_depth_greatest"
## [184] "AK1_depth_dimension"
## [185] "AK1_condition_code"
## [186] "AK1_date_occurrence"
## [187] "AK1_quality_code"
## [188] "AM1_snow_accumulation_greatest_24_hrs"
## [189] "AM1_depth_dimension"
## [190] "AM1_condition_code"
## [191] "AM1_dates_occurrence_1"
## [192] "AM1_dates_occurrence_2"
## [193] "AM1_dates_occurrence_3"
## [194] "AM1_quality_code"
## [195] "EQD_observation_identifier"
## [196] "EQD_observation_text"
## [197] "EQD_reason_code"
## [198] "EQD_parameter"
## [199] "KB1_avg_temp"
```

```
## [200] "KB1_period_quantity"
## [201] "KB1_temp_code"
## [202] "KB1_temp"
## [203] "KB1_temp_quality"
## [204] "KB2_avg_temp"
## [205] "KB2_period_quantity"
## [206] "KB2_temp_code"
## [207] "KB2_temp"
## [208] "KB2_temp_quality"
## [209] "KB3_avg_temp"
## [210] "KB3_period_quantity"
## [211] "KB3_temp_code"
## [212] "KB3_temp"
## [213] "KB3_temp_quality"
## [214] "KC1_extreme_temp_month"
## [215] "KC1_code"
## [216] "KC1_condition_code"
## [217] "KC1_temp"
## [218] "KC1_dates"
## [219] "KC1_temp_quality"
## [220] "KC2_extreme_temp_month"
## [221] "KC2_code"
## [222] "KC2_condition_code"
## [223] "KC2_temp"
## [224] "KC2_dates"
## [225] "KC2_temp_quality"
## [226] "KD1_heating_cooling_degree_days"
## [227] "KD1_period_quantity"
## [228] "KD1_code"
## [229] "KD1_days"
## [230] "KD1_days_quality"
## [231] "KD2_heating_cooling_degree_days"
## [232] "KD2_period_quantity"
## [233] "KD2_code"
## [234] "KD2_days"
## [235] "KD2_days_quality"
## [236] "KE1_extreme_temp_days_exceeding_criteria"
## [237] "KE1_number_days_max_temp_32lower"
## [238] "KE1_number_days_max_temp_32lower_quality"
## [239] "KE1_number_days_max_temp_90higher"
## [240] "KE1_number_days_max_temp_90higher_quality"
## [241] "KE1_number_days_min_temp_32lower"
## [242] "KE1_number_days_min_temp_32lower_quality"
## [243] "KE1_number_days_min_temp_0lower"
## [244] "KE1_number_days_min_temp_0lower_quality"
## [245] "MH1_atmospheric_pressure"
## [246] "MH1_avg_pressure"
## [247] "MH1_avg_pressure_quality"
## [248] "MH1_avg_sea_level_pressure"
## [249] "MH1_avg_sea_level_pressure_quality"
## [250] "MK1_atmospheric_pressure"
## [251] "MK1_max_sea_level_pressure"
## [252] "MK1_max_sea_level_pressure_datetime"
## [253] "MK1_max_sea_level_pressure_quality"
```

```
## [254] "MK1_min_sea_level_pressure"
## [255] "MK1_min_sea_level_pressure_datetime"
## [256] "MK1_min_sea_level_pressure_quality"
## [257] "NO3_original_observation"
## [258] "NO3_original_value_text"
## [259] "NO3_units_code"
## [260] "NO3_parameter_code"
## [261] "NO1_original_observation"
## [262] "NO1_original_value_text"
## [263] "NO1_units_code"
## [264] "NO1_parameter_code"
## [265] "GA2_sky_cover_layer_identifier"
## [266] "GA2_coverage_code"
## [267] "GA2_coverage_quality_code"
## [268] "GA2_base_height_dimension"
## [269] "GA2_base_height_quality_code"
## [270] "GA2_cloud_type_code"
## [271] "GA2_cloud_type_quality_code"
## [272] "GD2_sky_cover_summation_state_identifier"
## [273] "GD2_coverage_code_1"
## [274] "GD2_coverage_code_2"
## [275] "GD2_coverage_quality_code"
## [276] "GD2_height_dimension"
## [277] "GD2_height_dimension_quality_code"
## [278] "GD2_characteristic_code"
## [279] "GE1_sky_condition"
## [280] "GE1_connective_cloud_attribute"
## [281] "GE1_vertical_datum_attribute"
## [282] "GE1_base_height_upper_range_attribute"
## [283] "GE1_base_height_lower_range_attribute"
## [284] "GF1_sky_condition"
## [285] "GF1_coverage"
## [286] "GF1_opaque_coverage"
## [287] "GF1_coverage_quality"
## [288] "GF1_lowest_cover"
## [289] "GF1_lowest_cover_quality"
## [290] "GF1_low_cloud_genus"
## [291] "GF1_low_cloud_genus_quality"
## [292] "GF1_lowest_cloud_base_height"
## [293] "GF1_lowest_cloud_base_height_quality"
## [294] "GF1_mid_cloud_genus"
## [295] "GF1_mid_cloud_genus_quality"
## [296] "GF1_high_cloud_genus"
## [297] "GF1_high_cloud_genus_quality"
## [298] "OC1_wind_gust_observation_identifier"
## [299] "OC1_speed_rate"
## [300] "OC1_quality_code"
## [301] "AU1_present_weather_observation_identifier_asos_awos"
## [302] "AU1_intensity_and_proximity_code"
## [303] "AU1_descriptor_code"
## [304] "AU1_precipitation_code"
## [305] "AU1_obscuration_code"
## [306] "AU1_other_weather_phenomena_code"
## [307] "AU1_combination_indicator_code"
```

```
## [308] "AU1_quality_code"
## [309] "AW1_present_weather_observation_identifier"
## [310] "AW1_automated_atmospheric_condition_code"
## [311] "AW1_quality_automated_atmospheric_condition_code"
## [312] "WA1_platform_ice_accretion"
## [313] "WA1_source_code"
## [314] "WA1_thickness_dimension"
## [315] "WA1_tendency_code"
## [316] "WA1_quality_code"
## [317] "N08_original_observation"
## [318] "N08_original_value_text"
## [319] "N08_units_code"
## [320] "N08_parameter_code"
## [321] "MW1_first_weather_reported"
## [322] "MW1_manual_atmospheric_condition_code"
## [323] "MW1_condition_quality"
## [324] "N07_original_observation"
## [325] "N07_original_value_text"
## [326] "N07_units_code"
## [327] "N07_parameter_code"
## [328] "AX1_past_weather_observation_summaryofday"
## [329] "AX1_atmospheric_condition_code"
## [330] "AX1_quality_manual_atmospheric_condition_code"
## [331] "AX1_period_quantity"
## [332] "AX1_period_quality_code"
## [333] "N06_original_observation"
## [334] "N06_original_value_text"
## [335] "N06_units_code"
## [336] "N06_parameter_code"
## [337] "AU2_present_weather_observation_identifier_asos_awos"
## [338] "AU2_intensity_and_proximity_code"
## [339] "AU2_descriptor_code"
## [340] "AU2_precipitation_code"
## [341] "AU2_obscuration_code"
## [342] "AU2_other_weather_phenomena_code"
## [343] "AU2_combination_indicator_code"
## [344] "AU2_quality_code"
## [345] "AW2_present_weather_observation_identifier"
## [346] "AW2_automated_atmospheric_condition_code"
## [347] "AW2_quality_automated_atmospheric_condition_code"
## [348] "AA2_precipitation_liquid"
## [349] "AA2_period_quantity_hrs"
## [350] "AA2_depth"
## [351] "AA2_condition_quality"
## [352] "AA2_quality_code"
## [353] "N04_original_observation"
## [354] "N04_original_value_text"
## [355] "N04_units_code"
## [356] "N04_parameter_code"
## [357] "GA3_sky_cover_layer_identifier"
## [358] "GA3_coverage_code"
## [359] "GA3_coverage_quality_code"
## [360] "GA3_base_height_dimension"
## [361] "GA3_base_height_quality_code"
```

```
## [362] "GA3_cloud_type_code"
## [363] "GA3_cloud_type_quality_code"
## [364] "GD3_sky_cover_summation_state_identifier"
## [365] "GD3_coverage_code_1"
## [366] "GD3_coverage_code_2"
## [367] "GD3_coverage_quality_code"
## [368] "GD3_height_dimension"
## [369] "GD3_height_dimension_quality_code"
## [370] "GD3_characteristic_code"
## [371] "OD1_supplementary_wind_observation_identifier"
## [372] "OD1_type_code"
## [373] "OD1_period_quantity"
## [374] "OD1_speed_rate"
## [375] "OD1_speed_rate_quality_code"
## [376] "OD1_direction_quantity"
## [377] "AX2_past_weather_observation_summaryofday"
## [378] "AX2_atmospheric_condition_code"
## [379] "AX2_quality_manual_atmospheric_condition_code"
## [380] "AX2_period_quantity"
## [381] "AX2_period_quality_code"
## [382] "AX3_past_weather_observation_summaryofday"
## [383] "AX3_atmospheric_condition_code"
## [384] "AX3_quality_manual_atmospheric_condition_code"
## [385] "AX3_period_quantity"
## [386] "AX3_period_quality_code"
## [387] "AX4_past_weather_observation_summaryofday"
## [388] "AX4_atmospheric_condition_code"
## [389] "AX4_quality_manual_atmospheric_condition_code"
## [390] "AX4_period_quantity"
## [391] "AX4_period_quality_code"
## [392] "N02_original_observation"
## [393] "N02_original_value_text"
## [394] "N02_units_code"
## [395] "N02_parameter_code"
## [396] "N12_original_observation"
## [397] "N12_original_value_text"
## [398] "N12_units_code"
## [399] "N12_parameter_code"
## [400] "CT1_subhourly_temperature"
## [401] "CT1_average_air_temperature"
## [402] "CT1_average_air_temperature_quality_code"
## [403] "CT1_average_air_temperature_quality_flag"
## [404] "N09_original_observation"
## [405] "N09_original_value_text"
## [406] "N09_units_code"
## [407] "N09_parameter_code"
## [408] "N11_original_observation"
## [409] "N11_original_value_text"
## [410] "N11_units_code"
## [411] "N11_parameter_code"
## [412] "AW3_present_weather_observation_identifier"
## [413] "AW3_automated_atmospheric_condition_code"
## [414] "AW3_quality_automated_atmospheric_condition_code"
## [415] "AA3_precipitation_liquid"
```

```
## [416] "AA3_period_quantity_hrs"
## [417] "AA3_depth"
## [418] "AA3_condition_quality"
## [419] "AA3_quality_code"
## [420] "AH1_precipitation_liquid_max_short_duration"
## [421] "AH1_period_quantity"
## [422] "AH1_depth_dimension"
## [423] "AH1_condition_code"
## [424] "AH1_ending_date_time"
## [425] "AH1_quality_code"
## [426] "AH2_precipitation_liquid_max_short_duration"
## [427] "AH2_period_quantity"
## [428] "AH2_depth_dimension"
## [429] "AH2_condition_code"
## [430] "AH2_ending_date_time"
## [431] "AH2_quality_code"
## [432] "AH3_precipitation_liquid_max_short_duration"
## [433] "AH3_period_quantity"
## [434] "AH3_depth_dimension"
## [435] "AH3_condition_code"
## [436] "AH3_ending_date_time"
## [437] "AH3_quality_code"
## [438] "AH4_precipitation_liquid_max_short_duration"
## [439] "AH4_period_quantity"
## [440] "AH4_depth_dimension"
## [441] "AH4_condition_code"
## [442] "AH4_ending_date_time"
## [443] "AH4_quality_code"
## [444] "AH5_precipitation_liquid_max_short_duration"
## [445] "AH5_period_quantity"
## [446] "AH5_depth_dimension"
## [447] "AH5_condition_code"
## [448] "AH5_ending_date_time"
## [449] "AH5_quality_code"
## [450] "AH6_precipitation_liquid_max_short_duration"
## [451] "AH6_period_quantity"
## [452] "AH6_depth_dimension"
## [453] "AH6_condition_code"
## [454] "AH6_ending_date_time"
## [455] "AH6_quality_code"
## [456] "AI1_precipitation_liquid_max_short_duration"
## [457] "AI1_period_quantity"
## [458] "AI1_depth_dimension"
## [459] "AI1_condition_code"
## [460] "AI1_ending_date_time"
## [461] "AI1_quality_code"
## [462] "AI2_precipitation_liquid_max_short_duration"
## [463] "AI2_period_quantity"
## [464] "AI2_depth_dimension"
## [465] "AI2_condition_code"
## [466] "AI2_ending_date_time"
## [467] "AI2_quality_code"
## [468] "AI3_precipitation_liquid_max_short_duration"
## [469] "AI3_period_quantity"
```

```
## [470] "AI3_depth_dimension"
## [471] "AI3_condition_code"
## [472] "AI3_ending_date_time"
## [473] "AI3_quality_code"
## [474] "AI4_precipitation_liquid_max_short_duration"
## [475] "AI4_period_quantity"
## [476] "AI4_depth_dimension"
## [477] "AI4_condition_code"
## [478] "AI4_ending_date_time"
## [479] "AI4_quality_code"
## [480] "AI5_precipitation_liquid_max_short_duration"
## [481] "AI5_period_quantity"
## [482] "AI5_depth_dimension"
## [483] "AI5_condition_code"
## [484] "AI5_ending_date_time"
## [485] "AI5_quality_code"
## [486] "AI6_precipitation_liquid_max_short_duration"
## [487] "AI6_period_quantity"
## [488] "AI6_depth_dimension"
## [489] "AI6_condition_code"
## [490] "AI6_ending_date_time"
## [491] "AI6_quality_code"
## [492] "MV1_present_weather"
## [493] "MV1_atmospheric_condition_code"
## [494] "MV1_condition_quality"
## [495] "AU3_present_weather_observation_identifier_asos_awos"
## [496] "AU3_intensity_and_proximity_code"
## [497] "AU3_descriptor_code"
## [498] "AU3_precipitation_code"
## [499] "AU3_obscuration_code"
## [500] "AU3_other_weather_phenomena_code"
## [501] "AU3_combination_indicator_code"
## [502] "AU3_quality_code"
## [503] "AW4_present_weather_observation_identifier"
## [504] "AW4_automated_atmospheric_condition_code"
## [505] "AW4_quality_automated_atmospheric_condition_code"
## [506] "ED1_runway_visual_range"
## [507] "ED1_direction_angle"
## [508] "ED1_runway_designator_code"
## [509] "ED1_visibility_dimension"
## [510] "ED1_quality_code"
## [511] "MW2_first_weather_reported"
## [512] "MW2_manual_atmospheric_condition_code"
## [513] "MW2_condition_quality"
## [514] "N17_original_observation"
## [515] "N17_original_value_text"
## [516] "N17_units_code"
## [517] "N17_parameter_code"
## [518] "AL1_snow_accumulation"
## [519] "AL1_period_quantity"
## [520] "AL1_depth_dimension"
## [521] "AL1_condition_code"
## [522] "AL1_quality_code"
```

You will want to fix the date. In the ISD format it is split between calendar day (UTC time always) and clock hour (UTC time always)

This can be done with the lubridate function ymd_hm()

```
targ_data$date_time = ymd_hm(sprintf("%s %s",
                                     as.character(targ_data$date),
                                     targ_data$time))

targ_data$date = targ_data$date_time
```

The data is parsed as strings. So you'll have to use the as.numeric() a lot Missing fields are typically ID'ed as 9999 in the original data

```
targ_data$temperature[targ_data$temperature == "+9999"]                    = NA

targ_data$temperature_dewpoint[targ_data$temperature_dewpoint == "+9999"] = NA

targ_data$air_pressure[targ_data$air_pressure == "99999"]                  = NA

targ_data$wind_speed[targ_data$wind_speed == "9999"]                       = NA

targ_data$wind_direction[targ_data$wind_direction == "999"]                = NA
```

Some of these fields, but not all, are managed with the isd_transform() function with respect to scaling

temp/dew point (deg C) converted from 10ths of degree mean sea level pressure (hPa) converted from 10ths of hPa wind speed (m s-1) converted from 10ths of m s-1 wind direction (compass decimal degrees)

Precip is also processed but this data set is an merger of both hourly, 3-hrly, 6-hrly, 12-hrly, 24-hrly data depending on the report message that's being archived.

```
precip_workspace_time_interval = as.numeric(targ_data$AA1_period_quantity_hrs)

precip_workspace_depth = as.numeric(targ_data$AA1_depth)

precip_workspace_depth[precip_workspace_depth == 9999] = NA

precip_workspace_depth_01hrly =  precip_workspace_depth
precip_workspace_depth_03hrly =  precip_workspace_depth
precip_workspace_depth_06hrly =  precip_workspace_depth
precip_workspace_depth_12hrly =  precip_workspace_depth
precip_workspace_depth_24hrly =  precip_workspace_depth

precip_workspace_depth_01hrly[precip_workspace_time_interval != 01]  =  NA
precip_workspace_depth_03hrly[precip_workspace_time_interval != 03]  =  NA
precip_workspace_depth_06hrly[precip_workspace_time_interval != 06]  =  NA
precip_workspace_depth_12hrly[precip_workspace_time_interval != 12]  =  NA
precip_workspace_depth_24hrly[precip_workspace_time_interval != 24]  =  NA

targ_data$precip_01hr = precip_workspace_depth_01hrly
targ_data$precip_03hr = precip_workspace_depth_03hrly
targ_data$precip_06hr = precip_workspace_depth_06hrly
targ_data$precip_12hr = precip_workspace_depth_12hrly
targ_data$precip_24hr = precip_workspace_depth_24hrly
```

These fields that come from the ISD extraction are typically integer and aren't converted to their expected units. This function below scales them correctly.

```
targ_data = isd_transform(targ_data)
```

```
## Warning in as.POSIXlt.POSIXct(x, tz = tz): unknown timezone '%Y%m%d'
```

```
# patch the wind direction so it's 0 degrees when the wind speed is missing

targ_data$wind_direction[targ_data$wind_speed == 0]             = 0
```

Cloud Cover is held in several positions in the dataset. Once again this is because the data is designed for use for aerodrome use.

I am setting it up to give you the "GF1 group" which is the total cloud cover for all flight levels. These fields are often archived in "octaves" or eights and I'll be converting them to fractions for you. IN cases of "obscured sky" caused by fog or smoke, I will label it 100% for total obscured sky or 50% covered for partial obscured skies.
print

```
targ_data$GF1_total_cloud_cover_fraction = as.numeric(targ_data$GF1_coverage)

# 00: None, SKC or CLR

targ_data$GF1_total_cloud_cover_fraction[targ_data$GF1_total_cloud_cover_fraction==00] = 0.00

# 01: One okta - 1/10 or less but not zero

targ_data$GF1_total_cloud_cover_fraction[targ_data$GF1_total_cloud_cover_fraction==01] = 1.0 / 8.0

# 02: Two oktas - 2/10 - 3/10, or FEW

targ_data$GF1_total_cloud_cover_fraction[targ_data$GF1_total_cloud_cover_fraction==02] = 2.0 / 8.0

# 03: Three oktas - 4/10

targ_data$GF1_total_cloud_cover_fraction[targ_data$GF1_total_cloud_cover_fraction==03] = 3.0 / 8.0

# 04: Four oktas - 5/10, or SCT

targ_data$GF1_total_cloud_cover_fraction[targ_data$GF1_total_cloud_cover_fraction==04] = 4.0 / 8.0

# 05: Five oktas - 6/10

targ_data$GF1_total_cloud_cover_fraction[targ_data$GF1_total_cloud_cover_fraction==05] = 5.0 / 8.0

# 06: Six oktas - 7/10 - 8/10

targ_data$GF1_total_cloud_cover_fraction[targ_data$GF1_total_cloud_cover_fraction==06] = 6.0 / 8.0

# 07: Seven oktas - 9/10 or more but not 10/10, or BKN

targ_data$GF1_total_cloud_cover_fraction[targ_data$GF1_total_cloud_cover_fraction==07] = 7.0 / 8.0

# 08: Eight oktas - 10/10, or OVC

targ_data$GF1_total_cloud_cover_fraction[targ_data$GF1_total_cloud_cover_fraction==08] = 8.0 / 8.0
```

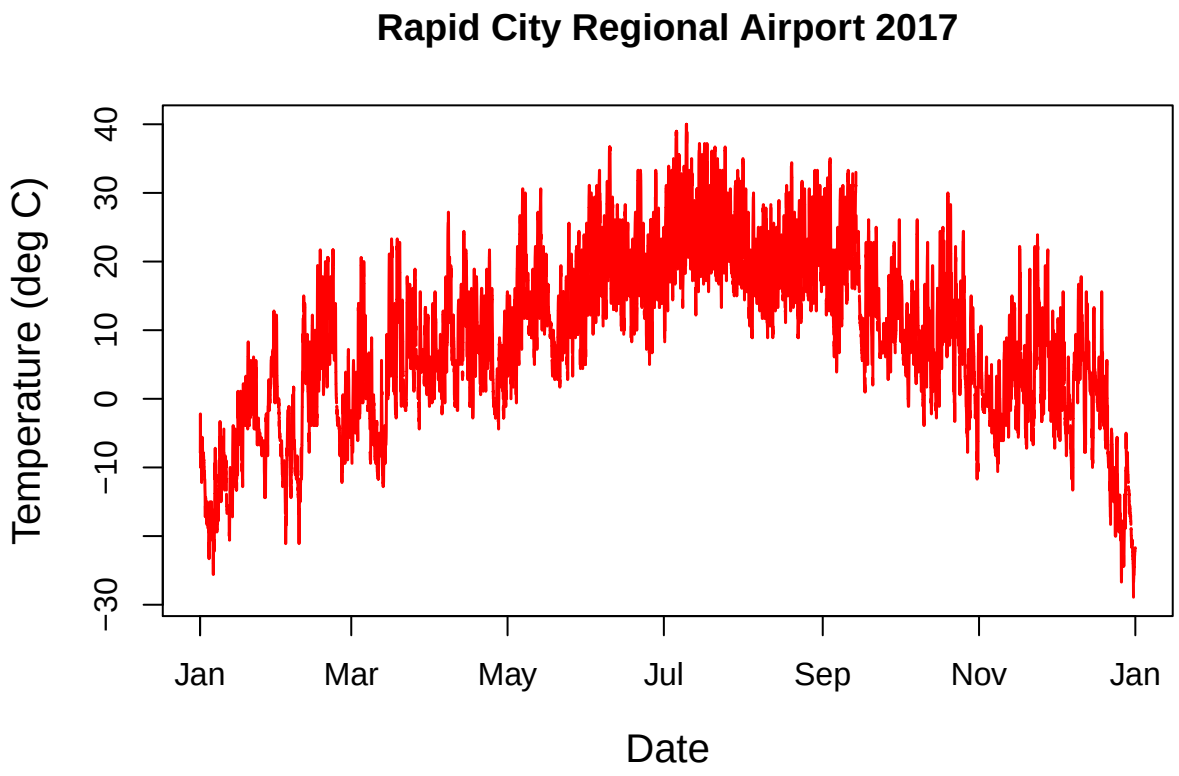
```
# 09: Sky obscured, or cloud amount cannot be estimated
targ_data$GF1_total_cloud_cover_fraction[targ_data$GF1_total_cloud_cover_fraction==09] = 8.0 / 8.0
# 10: Partial obscuration
targ_data$GF1_total_cloud_cover_fraction[targ_data$GF1_total_cloud_cover_fraction==10] = 4.0 / 8.0
# 11: Thin scattered
targ_data$GF1_total_cloud_cover_fraction[targ_data$GF1_total_cloud_cover_fraction==11] = 2.0 / 8.0
# 12: Scattered
targ_data$GF1_total_cloud_cover_fraction[targ_data$GF1_total_cloud_cover_fraction==12] = 4.0 / 8.0
# 13: Dark scattered
targ_data$GF1_total_cloud_cover_fraction[targ_data$GF1_total_cloud_cover_fraction==13] = 5.0 / 8.0
# 14: Thin broken
targ_data$GF1_total_cloud_cover_fraction[targ_data$GF1_total_cloud_cover_fraction==14] = 6.0 / 8.0
# 15: Broken
targ_data$GF1_total_cloud_cover_fraction[targ_data$GF1_total_cloud_cover_fraction==15] = 7.0 / 8.0
# 16: Dark broken
targ_data$GF1_total_cloud_cover_fraction[targ_data$GF1_total_cloud_cover_fraction==16] = 8.0 / 8.0
# 17: Thin overcast
targ_data$GF1_total_cloud_cover_fraction[targ_data$GF1_total_cloud_cover_fraction==17] = 4.0 / 8.0
# 18: Overcast
targ_data$GF1_total_cloud_cover_fraction[targ_data$GF1_total_cloud_cover_fraction==18] = 8.0 / 8.0
# 19: Dark overcast
targ_data$GF1_total_cloud_cover_fraction[targ_data$GF1_total_cloud_cover_fraction==19] = 8.0 / 8.0
# 99: Missing
targ_data$GF1_total_cloud_cover_fraction[targ_data$GF1_total_cloud_cover_fraction==99] = NA
```

Now that we have our data prepeared we can now print things out.

Plotting Temperature

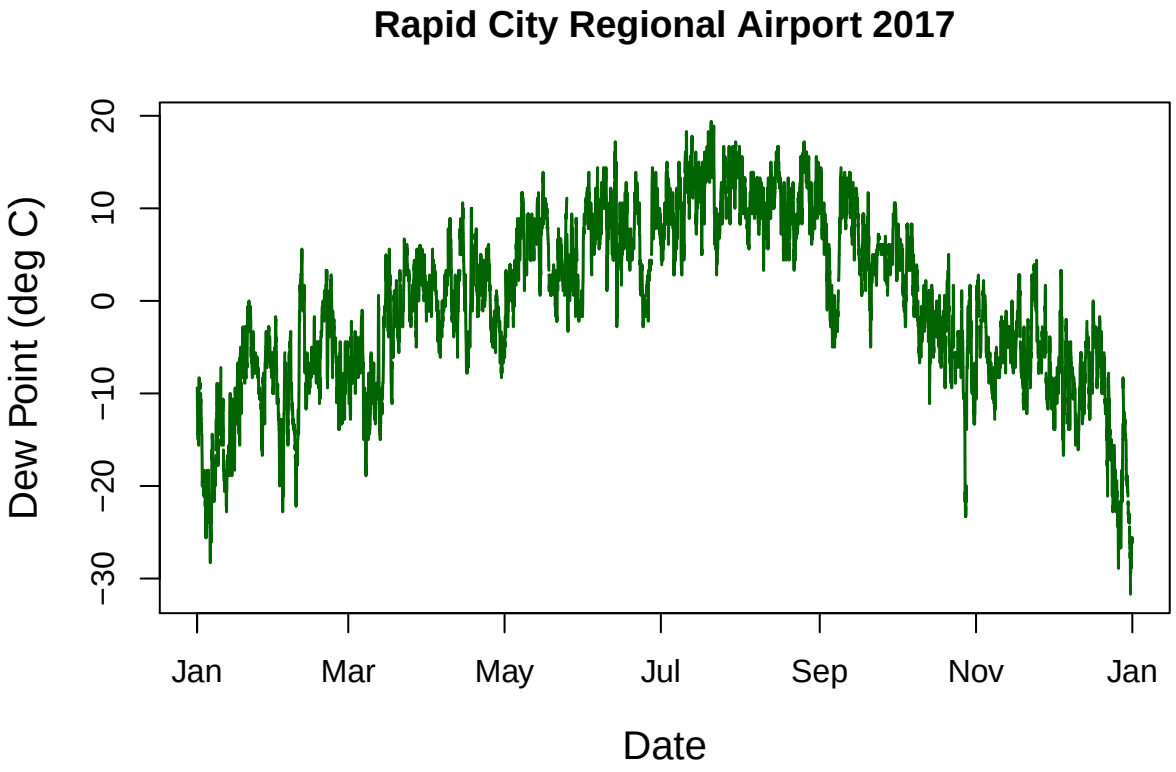
```
plot(x       = targ_data$date_time,
     y       = targ_data$temperature,
     type    = "l",
```

```
     col      = "red",
     lwd      = 1.5,
     cex.lab = 1.25,
     xlab     = "Date",
     ylab     = "Temperature (deg C)",
     main     = station_name_label)
```



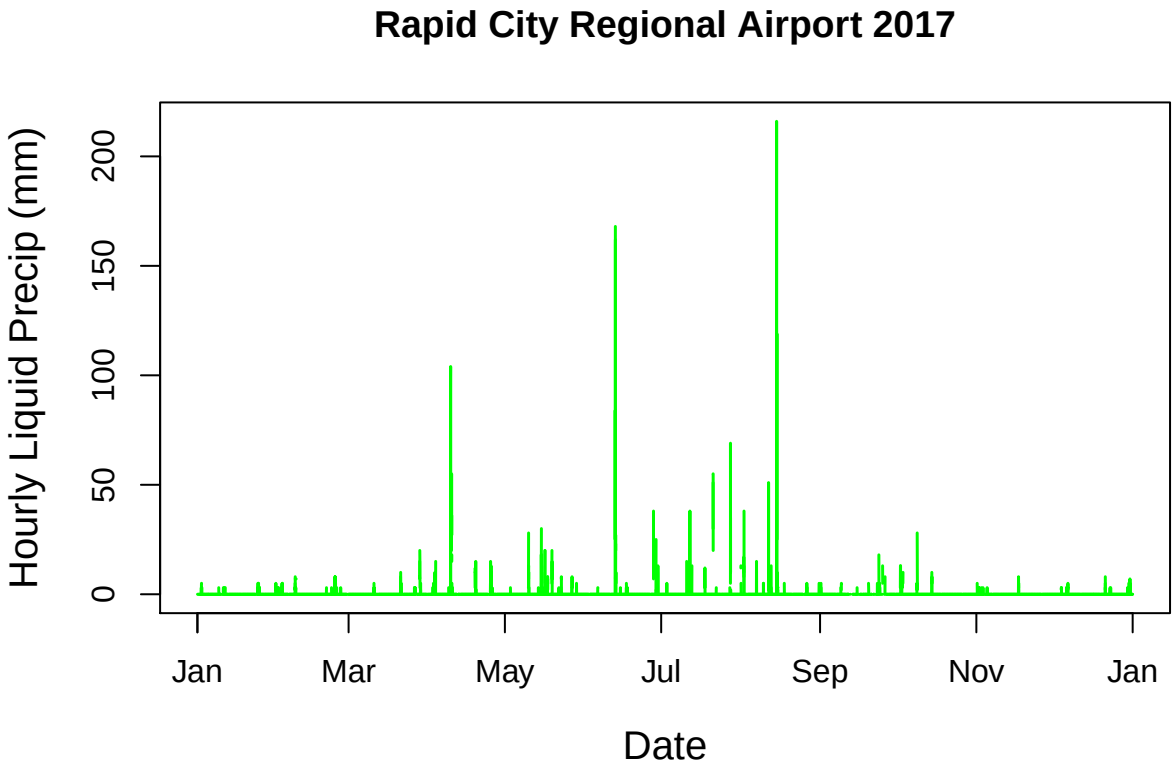
Plotting Dew Point

```
plot(x        = targ_data$date_time,
     y        = targ_data$temperature_dewpoint,
     type     = "l",
     col      = "darkgreen",
     lwd      = 1.5,
     cex.lab = 1.25,
     xlab     = "Date",
     ylab     = "Dew Point (deg C)",
     main     = station_name_label)
```



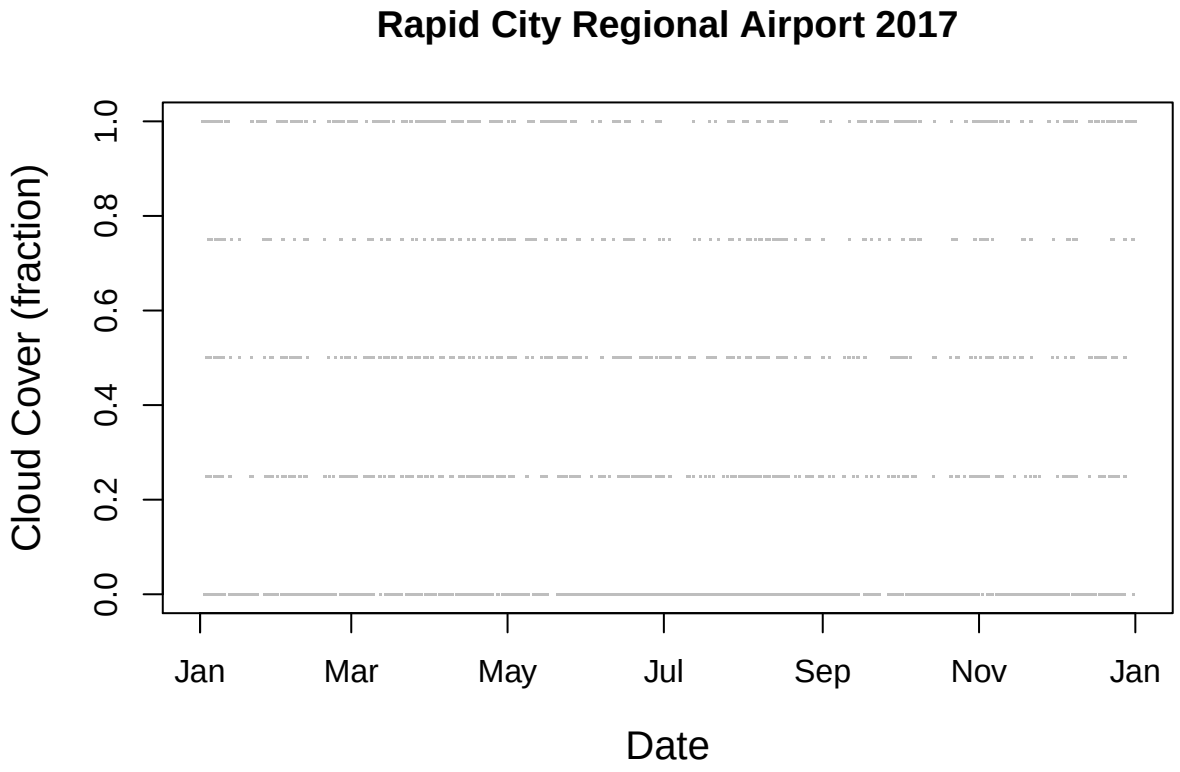
Plotting Rainfall Precipitation

```
plot(x        = targ_data$date_time,
     y        = targ_data$precip_01hr,
     type     = "l",
     col      = "green",
     lwd      = 1.5,
     cex.lab  = 1.25,
     xlab     = "Date",
     ylab     = "Hourly Liquid Precip (mm)",
     main     = station_name_label)
```



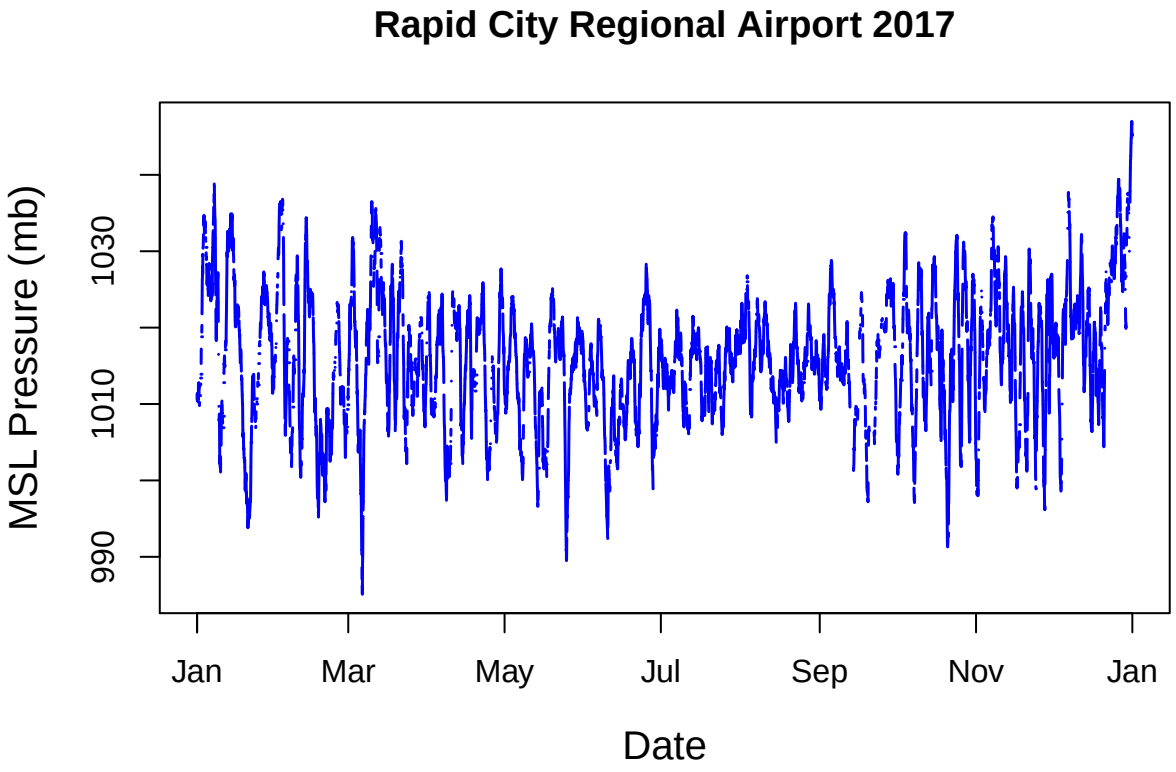
Plotting Cloud Fraction

```
plot(x       = targ_data$date_time,
     y       = targ_data$GF1_total_cloud_cover_fraction,
     type    = "p",
     pch     = ".",   # as points
     col     = "grey",
     lwd     = 1.5,
     cex.lab = 1.25,
     xlab    = "Date",
     ylab    = "Cloud Cover (fraction)",
     main    = station_name_label)
```



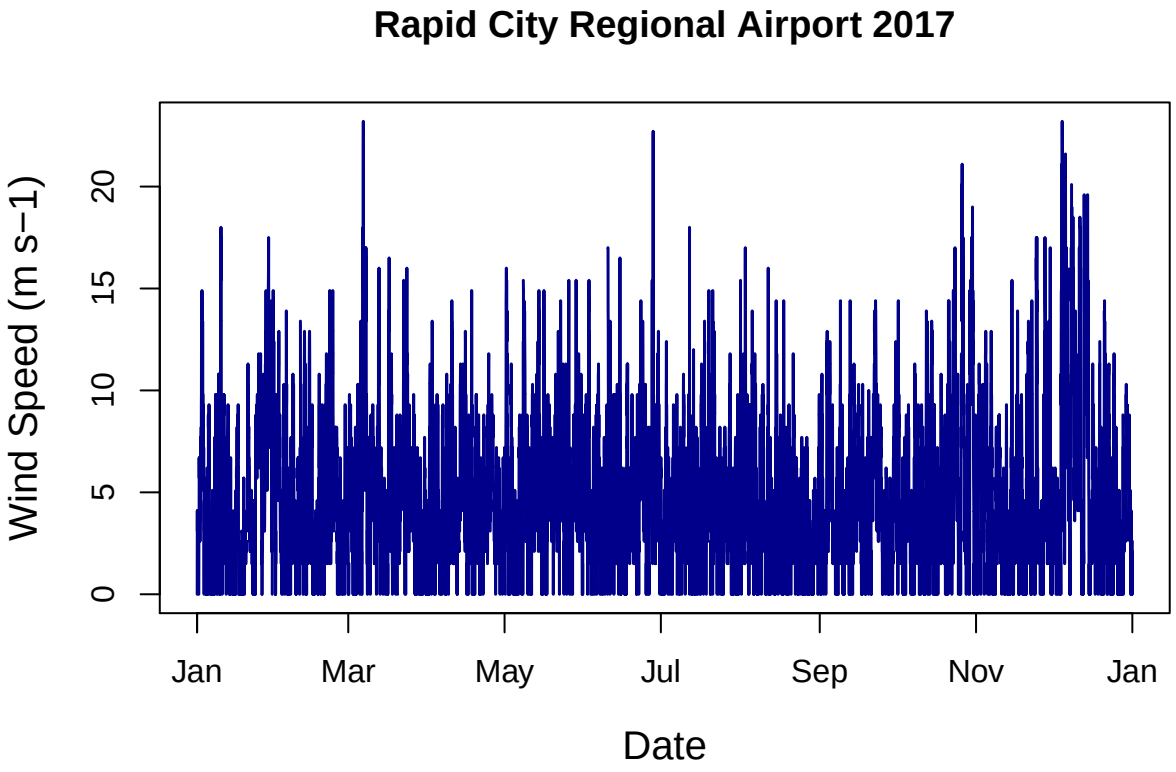
Plotting Pressure

```
plot(x        = targ_data$date_time,
     y        = targ_data$air_pressure,
     type     = "l",
     col      = "blue",
     lwd      = 1.5,
     cex.lab  = 1.25,
     xlab     = "Date",
     ylab     = "MSL Pressure (mb)",
     main     = station_name_label)
```



Plotting Wind Speed

```
plot(x        = targ_data$date_time,
     y        = targ_data$wind_speed,
     type     = "l",
     col      = "darkblue",
     lwd      = 1.5,
     cex.lab  = 1.25,
     xlab     = "Date",
     ylab     = "Wind Speed (m s-1)",
     main     = station_name_label)
```

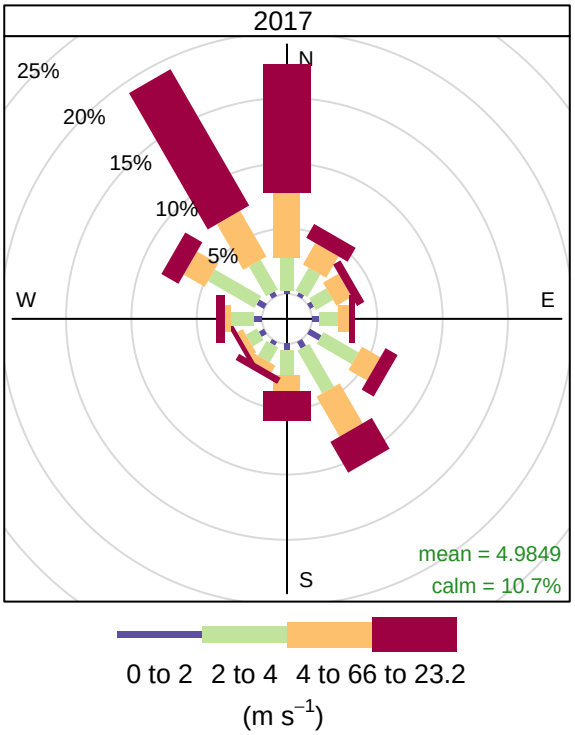


Plotting Wind Rose

```
windrose_frame = data.frame(date      = targ_data$date_time,
                            ws        = targ_data$wind_speed,
                            wd        = targ_data$wind_direction,
                            longitude = targ_data$longitude,
                            latitude  = targ_data$latitude)

windRose(mydata     = windrose_frame,
         ws         = "ws",
         wd         = "wd",
         type       = "year",
         hemisphere = "northern",
         main       = station_name_label)
```

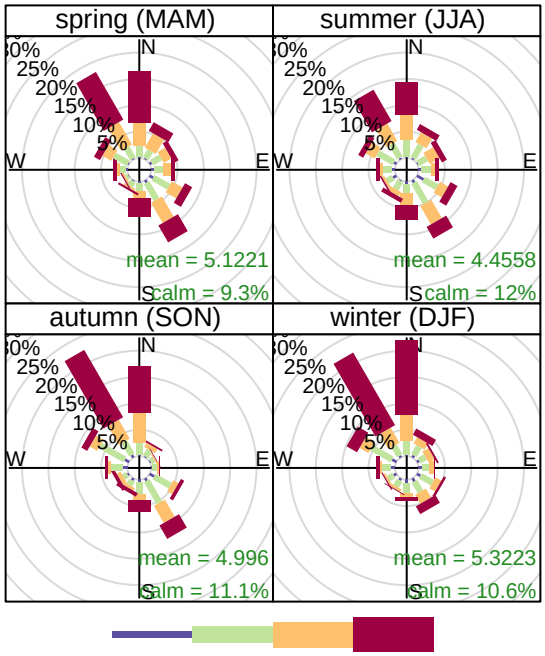
Rapid City Regional Airport 2017



Frequency of counts by wind direction (%)

```
windRose(mydata     = windrose_frame,
         ws         = "ws",
         wd         = "wd",
         type       = "season",
         hemisphere = "northern",
         main       = station_name_label)
```

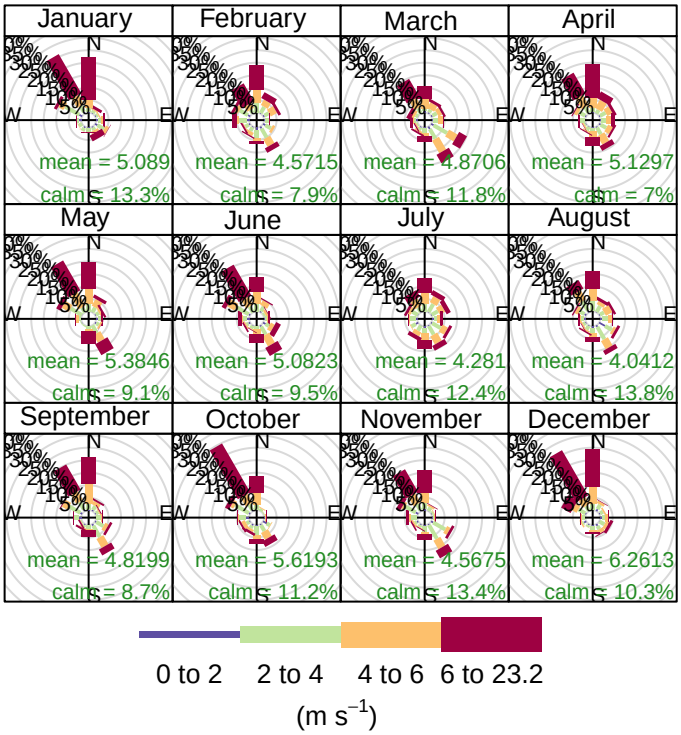
Rapid City Regional Airport 2017



Frequency of counts by wind direction (%)

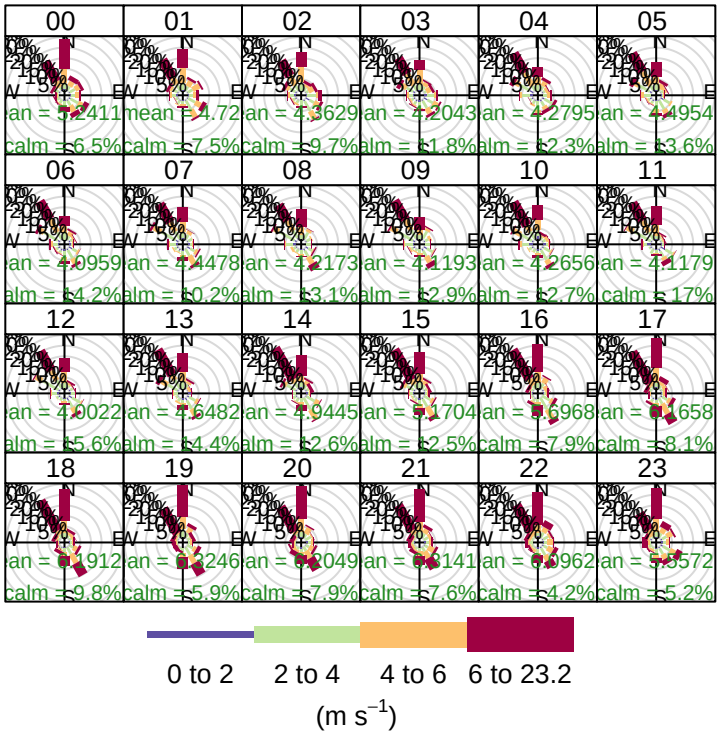
```
windRose(mydata     = windrose_frame,
         ws         = "ws",
         wd         = "wd",
         type       = "month",
         hemisphere = "northern",
         main       = station_name_label)
```

Rapid City Regional Airport 2017



Frequency of counts by wind direction (%)

```
windRose(mydata     = windrose_frame,
         ws         = "ws",
         wd         = "wd",
         type       = "hour",
         hemisphere = "northern",
         main       = station_name_label)
```



Finally let's put everything into a single time frame since we have observations at several times. We can interpolate the data to the nearest hour.

We do this by first creating a regularly spaced time vector

```
start_date = as.POSIXct(paste(target_year,
                              "-01-01 01:00:00 UTC",
                              sep=""),
                        tz = "UTC")

end_date   = as.POSIXct(paste((target_year+1),
                              "-01-01 00:00:00 UTC",
                              sep=""),
                        tz = "UTC")

hour_time = seq.POSIXt(from = start_date,
                       to   = end_date,
                       by   = "1 hour",
                       tz   = "UTC")


time_start_in_seconds = as.numeric(interval(start = "2006-01-01 00:00:00 UTC",
                                            end   = min(hour_time),
                                            tzone = tz(start)) )

time_end_in_seconds   = as.numeric(interval(start = "2006-01-01 00:00:00 UTC",
                                            end   = max(hour_time),
                                            tzone = tz(start)) )
```

```
hourly_interval = as.double(3600)

time_in_netcdf_units = seq(from = time_start_in_seconds,
                           to   = time_end_in_seconds,
                           by   =  hourly_interval)
```

We then interpolate between the various fields and frame them...

But first for precip we need a "nearest neighbor" intepolation function

```
targ_time_series                  = data.frame(date = hour_time)

targ_time_series$temperature_degC = approx(x      = targ_data$date_time,
                                           y      = targ_data$temperature,
                                           method = "linear",
                                           xout   = hour_time)$y

targ_time_series$dewpoint_degC    = approx(x      = targ_data$date_time,
                                           y      = targ_data$temperature_dewpoint,
                                           method = "linear",
                                           xout   = hour_time)$y

targ_time_series$cloud_fraction   = approx(x      = targ_data$date_time,
                                           y      = targ_data$GF1_total_cloud_cover_fraction,
                                           method = "linear",
                                           xout   = hour_time)$y

targ_time_series$press_msl_hPa    = approx(x      = targ_data$date_time,
                                           y      = targ_data$air_pressure,
                                           method = "linear",
                                           xout   = hour_time)$y

targ_time_series$wind_spd_ms      = approx(x      = targ_data$date_time,
                                           y      = targ_data$wind_speed,
                                           method = "linear",
                                           xout   = hour_time)$y

targ_time_series$wind_dir_degrees  = approx(x     = targ_data$date_time,
                                           y      = targ_data$wind_direction,
                                           method = "linear",
                                           xout   = hour_time)$y

targ_time_series$ISD_precip_01hr   = approx(x      = targ_data$date_time,
                                            y      = targ_data$precip_01hr,
                                            method = "constant",
                                            xout   = hour_time)$y

#targ_time_series$ISD_precip_03hr   = approx(x      = targ_data$date_time,
#                                            y      = targ_data$precip_03hr,
#                                            method = "constant",
#                                            xout   = hour_time)$y

#targ_time_series$ISD_precip_06hr   = approx(x      = targ_data$date_time,
#                                            y      = targ_data$precip_06hr,
```

```
#                                           method = "constant",
#                                           xout   = hour_time)$y

#targ_time_series$ISD_precip_12hr   = approx(x      = targ_data$date_time,
#                                           y      = targ_data$precip_12hr,
#                                           method = "constant",
#                                           xout   = hour_time)$y

#targ_time_series$ISD_precip_24hr   = approx(x      = targ_data$date_time,
#                                           y      = targ_data$precip_24hr,
#                                           method = "constant",
#                                           xout   = hour_time)$y

# print(targ_time_series)
```

And send it to an ASCII file

```
output_file_name = paste(file_title_string,
                         "_HOURLY_",
                         target_year,
                         ".csv",
                         sep="")

write.table(x    = targ_time_series,
            file = output_file_name,
            sep  =", ",
            row.names = FALSE)
```

Now let's make a "raw" datafile for all observations regardless what part of the hour they are taken.

```
targ_time_series_raw                      = data.frame(date = targ_data$date_time)

targ_time_series_raw$data_product_code    = targ_data$type_code

targ_time_series_raw$temperature          = targ_data$temperature
targ_time_series_raw$temperature_dewpoint = targ_data$temperature_dewpoint
targ_time_series_raw$cloud_cover_fraction = targ_data$GF1_total_cloud_cover_fraction
targ_time_series_raw$air_pressure         = targ_data$air_pressure

targ_time_series_raw$wind_speed       = targ_data$wind_speed
targ_time_series_raw$wind_dir_degrees = targ_data$wind_direction

targ_time_series_raw$ISD_precip_01hr = targ_data$precip_01hr


output_file_name = paste(file_title_string,
                         "_RAW_",
                         target_year,
                         ".csv",
                         sep="")

write.table(x    = targ_time_series_raw,
            file = output_file_name,
            sep  =", ",
            row.names = FALSE)
```

This section prepeares the output for a formal NetCDF for the hourly data. This will be followed by a second code block that will deal with the full data record.

We start by create the dimensions and the varialbles to which they are assigned. In this case, it's time. We also want to keep this variable as having an unlimited size so that we can concatenate several files together.

```
netcdf_time_dim  = ncdim_def(name  = "time",
                             units = "hours since 2006-01-01 00:00:00",
                             val   = as.double(time_in_netcdf_units)/as.double(3600),
                             unlim = TRUE,
                             calendar="standard")
```

Now we create the variables

```
fill_value = 9.96921e+36

netcdf_lat      = ncvar_def(nam      = "latitude",
                             units    = "degrees_north",
                             dim      = list(),
                             missval  = fill_value,
                             longname = "Latitude",
                             prec     ="single")

netcdf_lon      = ncvar_def(nam      = "longitude",
                             units    = "degrees_east",
                             dim      = list(),
                             missval  = fill_value,
                             longname = "Longitue",
                             prec     ="single")

netcdf_alt      = ncvar_def(nam      = "altitude",
                             units    = "m",
                             dim      = list(),
                             missval  = fill_value,
                             longname = "Elevation",
                             prec     ="single")

netcdf_temp      = ncvar_def(nam      = "air_temperature",
                             units    = "deg_C",
                             dim      = netcdf_time_dim,
                             missval  = fill_value,
                             longname = "2-m Air Temperature",
                             prec     ="single")

netcdf_dewpoint  = ncvar_def(nam      = "dew_point_temperature",
                             units    = "deg_C",
                             dim      = netcdf_time_dim,
                             missval  = fill_value,
                             longname = "2-m Dew Point Temperature",
                             prec     ="single")

netcdf_mslp      = ncvar_def(nam      = "air_pressure_at_mean_sea_level",
                             units    = "hPa",
                             dim      = netcdf_time_dim,
                             missval  = fill_value,
                             longname = "Air Pressure Reduced to Mean Sea Level",
```

```
                              prec     ="single")

netcdf_cloud     = ncvar_def(nam      = "cloud_area_fraction",
                              units    = "fraction",
                              dim      = netcdf_time_dim,
                              missval  = fill_value,
                              longname = "Message-Derived Cloud Cover Fraction",
                              prec     ="single")

netcdf_windspeed = ncvar_def(nam      = "wind_speed",
                              units    = "m s-1",
                              dim      = netcdf_time_dim,
                              missval  = fill_value,
                              longname = "10-m Wind Speed",
                              prec     ="single")

netcdf_winddir   = ncvar_def(nam      = "wind_from_direction",
                              units    = "degrees_from",
                              dim      = netcdf_time_dim,
                              missval  = fill_value,
                              longname = "10-m Wind Source Direction",
                              prec     ="single")

netcdf_prec      = ncvar_def(nam      = "precipitation_amount",
                              units    = "kg m-2",
                              dim      = netcdf_time_dim,
                              missval  = fill_value,
                              longname = "Hourly Precipitation",
                              prec     ="single")
```

With these basics done we can now create the empty file.

```
netcdf_output_file_name = paste(file_title_string,
                                "_HOURLY_",
                                target_year,
                                ".nc",
                                sep="")

nc_hourly = nc_create(filename = netcdf_output_file_name,
                      vars     = list(netcdf_lat,
                                      netcdf_lon,
                                      netcdf_alt,
                                      netcdf_temp,
                                      netcdf_dewpoint,
                                      netcdf_mslp,
                                      netcdf_cloud,
                                      netcdf_windspeed,
                                      netcdf_winddir,
                                      netcdf_prec),
                      force_v4 = FALSE,
                      verbose  = FALSE )
```

We can only assign at few attribute on file creation. We now add some of the other ones.

```
ncatt_put(nc         = nc_hourly,
          varid      = 0,
          attname    = "Title",
          attval     = paste("NCEI Data Hourly Output for ",
                             name_of_station,
                             sep=""),
          prec       = NA,
          verbose    = TRUE,
          definemode = FALSE )
```

```
## [1] "ncatt_put: entering"
## [1] "ncatt_put: checking for a global att"
## [1] "ncatt_put: IS a global att"
## [1] "ncatt_put_inner: entering with ncid= 65536 varid= -1 attname= Title attval= NCEI Data Hourly Ou
## [1] "ncatt_put_inner: no user-specified att type was given, figuring it out..."
## [1] "ncatt_put_inner: using deduced attribute prec of character"
## [1] "ncatt_put_inner: prec to create: character"
## [1] "ncatt_put: exiting"
```

```
ncatt_put(nc         = nc_hourly,
          varid      = 0,
          attname    = "WBAN_Number",
          attval     = as.integer(target_wban),
          prec       = NA,
          verbose    = FALSE,
          definemode = FALSE )

ncatt_put(nc         = nc_hourly,
          varid      = 0,
          attname    = "USAF_Number",
          attval     = as.integer(target_usaf),
          prec       = NA,
          verbose    = FALSE,
          definemode = FALSE )

ncatt_put(nc         = nc_hourly,
          varid      = 0,
          attname    = "Station_Name",
          attval     = name_of_station,
          prec       = NA,
          verbose    = FALSE,
          definemode = FALSE )

ncatt_put(nc         = nc_hourly,
          varid      = 0,
          attname    = "Station_Latitude",
          attval     = station_lat,
          prec       = NA,
          verbose    = FALSE,
          definemode = FALSE )

ncatt_put(nc         = nc_hourly,
          varid      = 0,
          attname    = "Station_Latitude",
```

```
          attval     = station_lon,
          prec       = NA,
          verbose    = FALSE,
          definemode = FALSE )
```

We should also add the standard names and suplementary descriptions for our variables.

```
ncatt_put(nc         = nc_hourly,
          varid      = netcdf_prec,
          attname    = "standard_name",
          attval     = "precipitation_amount",
          prec       = NA,
          verbose    = FALSE,
          definemode = FALSE )

ncatt_put(nc         = nc_hourly,
          varid      = netcdf_prec,
          attname    = "description",
          attval     = "Hourly Precipitation",
          prec       = NA,
          verbose    = FALSE,
          definemode = FALSE )

ncatt_put(nc         = nc_hourly,
          varid      = netcdf_prec,
          attname    = "comment",
          attval     = "Precipitation amounts taken from FM-15 & SAO Messages",
          prec       = NA,
          verbose    = FALSE,
          definemode = FALSE )


ncatt_put(nc         = nc_hourly,
          varid      = netcdf_winddir,
          attname    = "standard_name",
          attval     = "wind_from_direction",
          prec       = NA,
          verbose    = FALSE,
          definemode = FALSE )

ncatt_put(nc         = nc_hourly,
          varid      = netcdf_winddir,
          attname    = "description",
          attval     = "10-m Wind Source Direction",
          prec       = NA,
          verbose    = FALSE,
          definemode = FALSE )

ncatt_put(nc         = nc_hourly,
          varid      = netcdf_windspeed,
          attname    = "standard_name",
          attval     = "wind_speed",
          prec       = NA,
          verbose    = FALSE,
```

```
          definemode = FALSE )

ncatt_put(nc         = nc_hourly,
          varid      = netcdf_windspeed,
          attname    = "description",
          attval     = "10-m Wind Speed",
          prec       = NA,
          verbose    = FALSE,
          definemode = FALSE )

ncatt_put(nc         = nc_hourly,
          varid      = netcdf_cloud,
          attname    = "standard_name",
          attval     = "cloud_area_fraction",
          prec       = NA,
          verbose    = FALSE,
          definemode = FALSE )

ncatt_put(nc         = nc_hourly,
          varid      = netcdf_cloud,
          attname    = "description",
          attval     = "Message-Derived Cloud Cover Fraction",
          prec       = NA,
          verbose    = FALSE,
          definemode = FALSE )

ncatt_put(nc         = nc_hourly,
          varid      = netcdf_mslp,
          attname    = "standard_name",
          attval     = "air_pressure_at_mean_sea_level",
          prec       = NA,
          verbose    = FALSE,
          definemode = FALSE )

ncatt_put(nc         = nc_hourly,
          varid      = netcdf_mslp,
          attname    = "description",
          attval     = "Air Pressure Reduced to Mean Sea Level",
          prec       = NA,
          verbose    = FALSE,
          definemode = FALSE )

ncatt_put(nc         = nc_hourly,
          varid      = netcdf_dewpoint,
          attname    = "standard_name",
          attval     = "dew_point_temperature",
          prec       = NA,
          verbose    = FALSE,
          definemode = FALSE )

ncatt_put(nc         = nc_hourly,
          varid      = netcdf_dewpoint,
          attname    = "description",
```

```
          attval     = "2-m Dew Point Temperature",
          prec       = NA,
          verbose    = FALSE,
          definemode = FALSE )

ncatt_put(nc         = nc_hourly,
          varid      = netcdf_temp,
          attname    = "standard_name",
          attval     = "air_temperature",
          prec       = NA,
          verbose    = FALSE,
          definemode = FALSE )

ncatt_put(nc         = nc_hourly,
          varid      = netcdf_temp,
          attname    = "description",
          attval     = "2-m Air Temperature",
          prec       = NA,
          verbose    = FALSE,
          definemode = FALSE )

ncatt_put(nc         = nc_hourly,
          varid      = netcdf_alt,
          attname    = "standard_name",
          attval     = "altitude",
          prec       = NA,
          verbose    = FALSE,
          definemode = FALSE )

ncatt_put(nc         = nc_hourly,
          varid      = netcdf_alt,
          attname    = "description",
          attval     = "Elevation",
          prec       = NA,
          verbose    = FALSE,
          definemode = FALSE )

ncatt_put(nc         = nc_hourly,
          varid      = netcdf_lon,
          attname    = "standard_name",
          attval     = "longitiude",
          prec       = NA,
          verbose    = FALSE,
          definemode = FALSE )

ncatt_put(nc         = nc_hourly,
          varid      = netcdf_lon,
          attname    = "description",
          attval     = "Longitude",
          prec       = NA,
          verbose    = FALSE,
          definemode = FALSE )
```

```
ncatt_put(nc         = nc_hourly,
          varid      = netcdf_lat,
          attname    = "standard_name",
          attval     = "latitude",
          prec       = NA,
          verbose    = FALSE,
          definemode = FALSE )

ncatt_put(nc         = nc_hourly,
          varid      = netcdf_lat,
          attname    = "description",
          attval     = "Latitude",
          prec       = NA,
          verbose    = FALSE,
          definemode = FALSE )

#ncatt_put(nc         = nc_hourly,
#          varid      = netcdf_time_dim,
#          attname    = "standard_name",
#          attval     = "time",
#          prec       = NA,
#          verbose    = TRUE,
#          definemode = FALSE )

#ncatt_put(nc         = nc_hourly,
#         varid      = netcdf_time_dim,
#         attname    = "description",
#         attval     = "Time",
#          prec       = NA,
#          verbose    = TRUE,
#          definemode = FALSE )
```

Finally, we can populate these fields with Data.

```
ncvar_put(nc      = nc_hourly,
          varid   = netcdf_lat,
          vals    = station_lat,
          verbose = FALSE )

ncvar_put(nc      = nc_hourly,
          varid   = netcdf_lon,
          vals    = station_lon,
          verbose = FALSE )

ncvar_put(nc      = nc_hourly,
          varid   = netcdf_alt,
          vals    = station_alt,
          verbose = FALSE )

ncvar_put(nc      = nc_hourly,
          varid   = netcdf_temp,
          vals    = targ_time_series$temperature_degC,
          verbose = FALSE )
```

```
ncvar_put(nc      = nc_hourly,
          varid   = netcdf_dewpoint,
          vals    = targ_time_series$dewpoint_degC,
          verbose = FALSE )

ncvar_put(nc      = nc_hourly,
          varid   = netcdf_mslp,
          vals    = targ_time_series$press_msl_hPa,
          verbose = FALSE )

ncvar_put(nc      = nc_hourly,
          varid   = netcdf_cloud,
          vals    = targ_time_series$cloud_fraction,
          verbose = FALSE )

ncvar_put(nc      = nc_hourly,
          varid   = netcdf_windspeed,
          vals    = targ_time_series$wind_spd_ms,
          verbose = FALSE )

ncvar_put(nc      = nc_hourly,
          varid   = netcdf_winddir,
          vals    = targ_time_series$wind_dir_degrees,
          verbose = FALSE )

ncvar_put(nc      = nc_hourly,
          varid   = netcdf_prec,
          vals    = targ_time_series$ISD_precip_01hr,
          verbose = FALSE )
```