TWELVE YEARS

of the Solar Radiation Data Archive – Citizen Weather Observer Program • February 2009 – April 2021 •

A Technical Report Presented to the CWOP Community

Lucy Hancock and Russ Chadwick
October 2021

FOREWORD

In the State of the Union address of February 9, 2009, President Barack Obama set out a vision in which solar energy assumed a greater role America’s future mix of energy sources. At that time, the Citizen Weather Observer Program (CWOP) had been founded 8.5 years earlier and was already collecting weather data from interested citizens - but not saving solar radiation data. If the solar radiation data being sent to CWOP were archived permanently and made available, it would over time comprise a useful observational basis for research and planning of solar energy generation.

On this consideration, automatic collection of CWOP reports of solar radiation was programmed. On February 18, 2009, the new archive began to collect data.

As of April 30, 2021, the archive had collected about 0.93 billion reports.

This ***Twelve Year Report*** aims to do the following:

- *to present the dimensions of the data archive as of April 30, 2021;*
- *to illustrate some new means of exploratory data analysis, with code and examples;*
- *to update guidance toward user setup of a Redshift archive of the CWOP Solar Radiation Data Archive.*

Background: APRS – CWOP – CWOP Solar Radiation Data Archive

The CWOP solar radiation data archive is a subset of the Citizen Weather Observer Program, or CWOP, in turn an aspect of APRS.

APRS. The APRS (Automatic Position Reporting System or Automatic Packet Reporting System) data sharing infrastructure is a digital communications system that grew out of amateur radio. (For the origin of APRS, see http://aprs.org/.)

CWOP. In 2000, the Citizen Weather Observer Program came into being using APRS as a platform for the sharing of weather station reports, initially from hams to NOAA and shortly thereafter from other citizens to NOAA. (See http://wxqa.com.) CWOP uses the APRS protocols for sharing of its data.

CWOP provides standards, technical support and a community to interested weather observers making observations and willing to share them with NOAA and other users.

Several web-based pages enable CWOP members to view data from their own stations and those of other members, together with relevant information.

- The findu.com site shows station location
  http://www.findu.com/cgi-bin/find.cgi?call=CW3318
  and the links on the left side of the page show graphs and displays relating to the weather data.
- The gladstonefamily.com site gives information, including quality checking, relating to any specific site
  http://weather.gladstonefamily.net/site/C0003

- The wxqa.com site has a search tool that does searches for results and other information relating to any of the CWOP stations, http://wxqa.com/search.htm

Other services relating to data quality are listed here:
http://wxqa.com/aprswxnetqc.html

including an e-mail reflector by which volunteer experts provide support to new observers.

CWOP proved popular and it grew rapidly. Today there are more than 14,000 contributors to CWOP.

CWOP Solar Radiation Data Archive. CWOP collects and shares a torrent of weather data, but at the outset did not undertake to archive all of it. In that context, the role of the CWOP Solar Radiation Data Archive was to secure CWOP measurements of solar radiation and make them available to all. Since February 18, 2009, almost all CWOP reports containing a non-zero value for solar radiation have been routinely collected each day at 11 pm UTC, gzipped, and made available on Google Docs via the clicks provided at http://wxqa.com/lum_search.htm.

UPDATED DASHBOARD
Dimensions of the Data Archive:
Data Received February 18, 2009 to April 30, 2021

This report describes the observations stored in the CWOP Solar Radiation Archive as of April 30, 2021. As such, it is an update on the ***Six Year Report*** and the ***Ten Year Report*** (these are linked at http://wxqa.com/lum_search.htm). This report expands on the topic of code for exploratory data analysis.

Number of observations. About 0.93 billion observations were in the archive as of April 30, 2021. Their accession to the archive over time is as shown in Figure 1. Seasonal variation is evident in Figure 1 because the network is denser in the Northern Hemisphere; the number of weather packets reporting non-zero solar radiation is greater during Northern Hemisphere summer when daylight is longer.

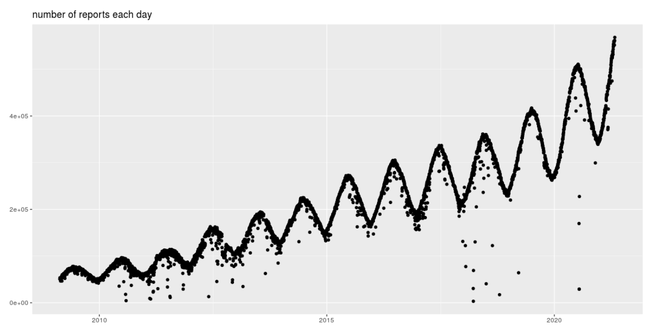


Figure 1. *Number of reports collected each day from February 18, 2009 through April 30, 2021, for operational days.*

Number of days. From inception to April 30, 2021, the archive captured data on 4422 days, i.e., all but 33 days. (See Annex A for Table of Missing Days.) The missing days occurred mainly in the first few years of the Archive as server operational procedures were being stabilized.

Number of stations. The number of stations reporting solar radiation has risen steadily over the twelve-year period. Figure 2 presents a time series of the number of stations reporting each day.

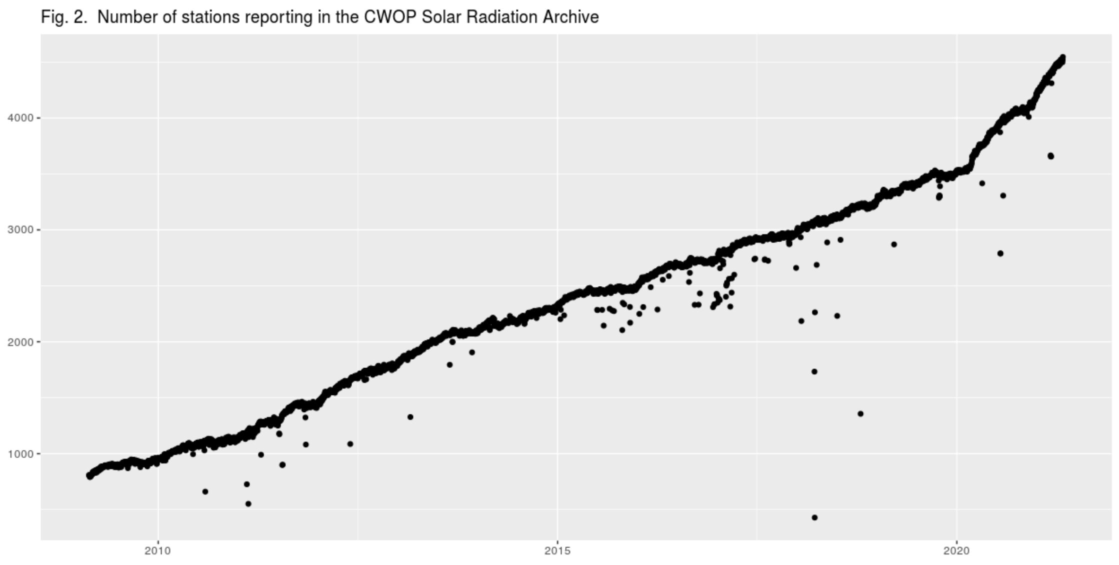


Figure 2. *For each day in the reporting period, the number of stations in the archive.*

The number of stations that have reported for five years or more is now in the thousands, as Figure 3 shows.

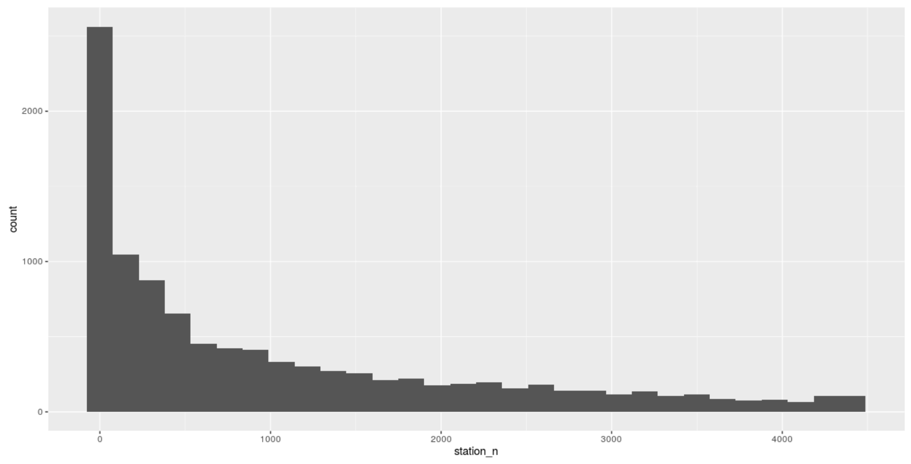


Figure. 3 As of April 30, 2021, distribution over all stations of the number of days reported. This distribution omits stations that reported for ten days or fewer. The slight bump at highest N shows the stations that were online on day one of the archive and are still operating.

Reporting programs. Most stations report once/five minutes, once/ten minutes or once/15 minutes. See Figure 4 for the estimated distribution of reporting intervals on March 21, 2021.

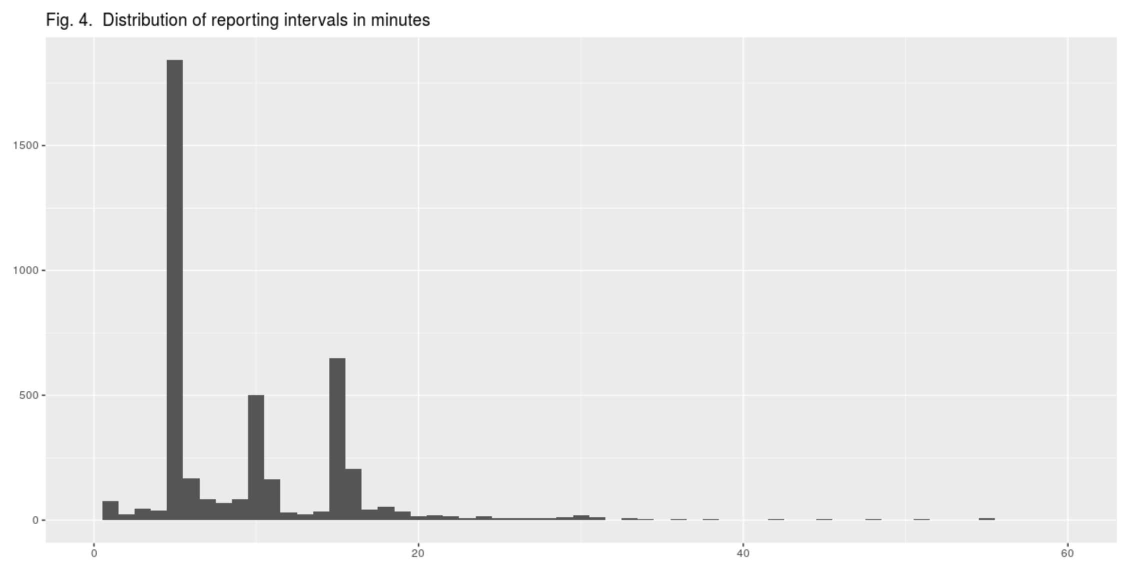
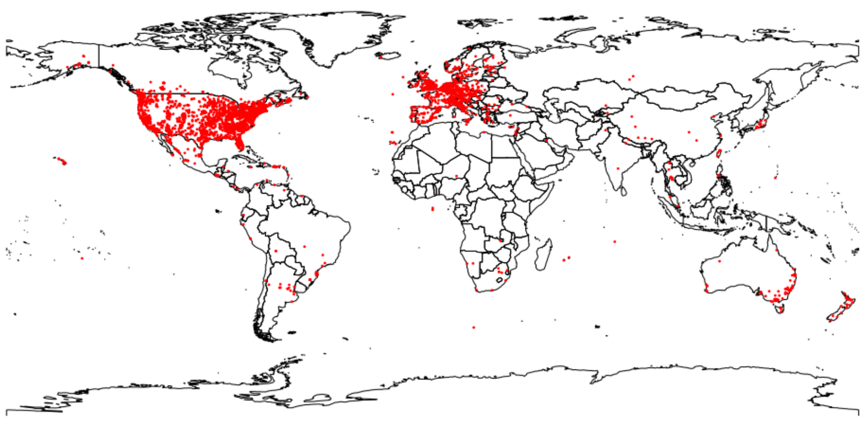


Figure 4. For all stations that reported at least twelve times on March 21, 2021, the distribution of estimated reporting intervals. The reporting frequency is estimated by a count of reports archived on March 21, 2021. As this was approximately the equinox, all stations were illuminated for about twelve hours. Thus 48 reports from one station that day would indicate four reports per hour, or a reporting interval of 15 minutes; 72 reports from one station a reporting interval of ten minutes; 144 an interval of five minutes..

Distribution of stations. The distribution of stations is global. See Map 1 for stations reporting April 30, 2021. The distribution of data remains skewed toward the Northern Hemisphere but coverage elsewhere is expanding. The vast majority of stations are located in populated areas.



Map 1. *Nominal positions of the stations that reported solar radiation data on April 30, 2021.*

Accompaniment by met data. Most solar radiation stations are also met stations reporting all the basic variables.

Table 1. Availability of Met Data

On April 30, 2021

```
98% of observations include wind direction.
98% of observations include wind speed in knots.
96% of observations include wind gust.
100% of observations include temperature.
96% of observations include rainfall in the last hour.
94% of observations include rainfall in the last 24h.
97% of observations include rainfall today.
100% of observations include relative humidity.
99% of observations include barometric pressure."
```

Hardware and software used in the network. In the final field of each observational report, the APRS protocol provides for a free-form field that does not have mandatory content. CWOP stations are encouraged to use this field to report on station hardware and software, and indeed the majority of users do provide at least some information. To see samples, click any of these links and then select one of the dates, to see a raw APRS report from a CWOP station. The field that ends the report is the information provided about hardware and software.

http://wxqa.com/activecwd/index_activecwd.html
http://wxqa.com/activedwd/index_activedwd.html
http://wxqa.com/activeewd/index_activeewd.html
http://wxqa.com/activefwd/index_activefwd.html
http://wxqa.com/callsminmax/index_callsminmax.html
http://wxqa.com/activegwd/index_activegwd.html

Table 2. The 10 most-used "TECH" strings on April 30, 2021

Rank	"TECH" String	Number of Stations	% of Stations (N=4272)
1	DsIP	1104	24.3
2	AmbientCWOP.com	653	14.4
3	eMB51	430	9.46
4	.WD 31	332	7.30
5	.DsWLL	316	6.95
6	.DsVP	294	6.47
7	eCumulusDsVP	210	4.62
8	WeatherCatV312B34H31	65	1.43
9	.weewx-4.5.1-Vantage	61	1.34
10	eMB50	55	1.21

METHODS FOR EXPLORATORY DATA ANALYSIS

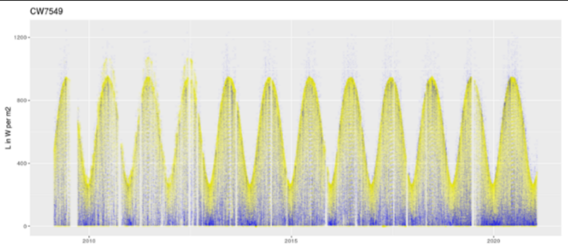
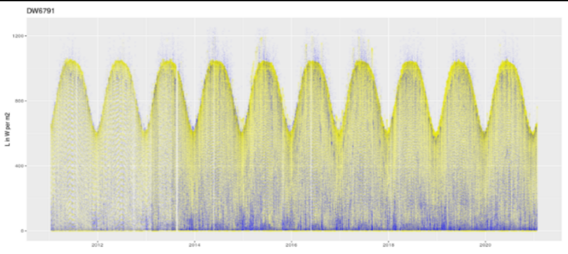
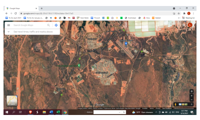
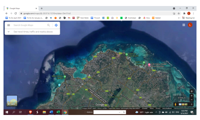
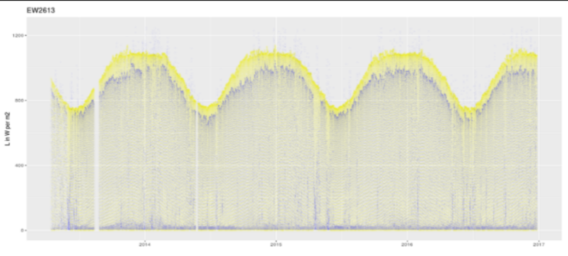
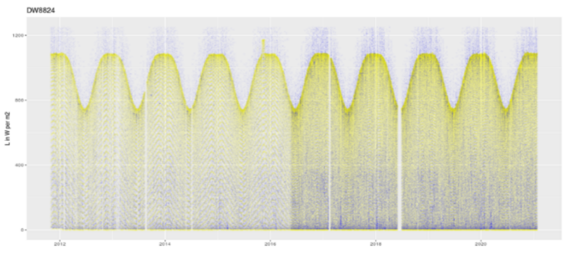
The ***Ten Year Report*** provided a recipe for setup of an Amazon relational database, a Redshift archive, comprising all observations, parsed data, time/astronomical data for each observation, and modeled L values for each observation per various assumptions (re albedo, ozone in the air column, atmospheric visibility, sensor tilt, and a few other things). In Annex B of this report, changes in the computing environment are noted as they affect some details of that recipe.

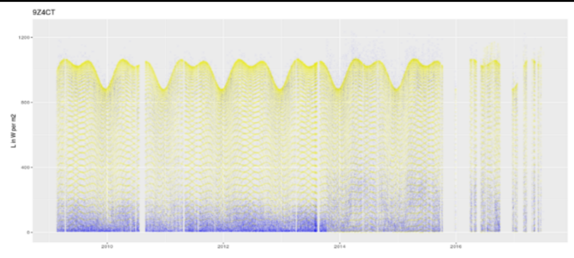
For a person in possession of such an archive, either because you made it or because you have simply requested a clone of the one we made, we have developed code examples in R/tidyverse to support you in quality control and exploratory data analysis of the archive. See Annex B, Table 1.

In this report, three useful approaches to exploratory data analysis are detailed.

Approach 1: Time Series Overlaying L Observations with Model Values

The figures below are presented as illustrations of latitude dependence, glinting, and cloud effects. Yellow dots are model, blue dots are observations.

Figure 5. Time series of observed values of L (blue) overlaid on modeled clear-sky values (yellow). (a) Latitude 50.4 N, Longitude 4.07W (near Plymouth, UK), Altitude 83 m	
(b) Latitude 31.7 N, Longitude 94.5W (Texas, US), Altitude 110 m	
(c) Latitude 20.4 S, Longitude 118.6 E (north coast of Australia), Altitude 18.2 m  (d) Latitude 20.0 S, Longitude 57.6E (north coast of Mauritius), Altitude 32.3 m 	 

(e) Latitude 10.7N, Longitude 61.5W (near Port of Spain, Trinidad and Tobago), Altitude 119 m	

Time series of observed L values overlying modeled values can flag certain issues:

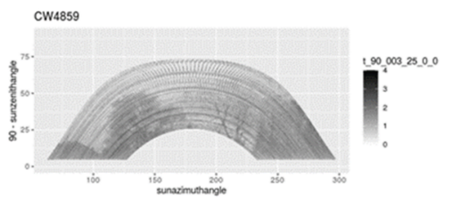
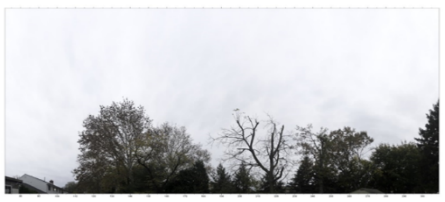
- Typos in station coordinates can be flagged by large mismatch between observation and model;
- Data gaps in the observations are apparent as gaps in the time series;
- Changes in the observing program are apparent as varying density of observations;
- Software can sometimes be identified in terms of the maximum L that is recorded.

Approach 2: Sunprint of Optical Depth, Other Statistics

Figure 5 above illustrated the difference between $L_{observed}$ and $L_{modeled}$. The difference can be compactly presented using the systematic comparison "optical depth" τ (see box, "Optical Depth"), which is just the log of the ratio $L_{observed}/L_{modeled}$.

$$\tau \equiv \ln (L_{model}/L_{observed})$$

To see what "optical depth" means, consider Figure 6, where, for one station, every τ is plotted at the Sun's position for that observation. Plotting τ amounts to a kind of "sunprint" of the station environment, a picture of all the shading that solar radiation encountered on its way to the sensor. This interpretation is validated by the accompanying photograph. The "sunprint" of course extends only over the arc of positions that the Sun has ever occupied, while the photograph is a wider, taller view. Another difference is that the photograph is taken at a single moment, while the sunprint records the difference in shading vegetation between summer (upper part of the figure) to midwinter (lowest). It appears the deciduous tree to the left is leafy at the top, in summertime data, but bare in its lowest branches, midwinter data.

	**Figure 6.** (a) Optical depth values at one CWOP station. Every value of τ over twelve years of operation is plotted here as a dot whose gray color is scaled to the value of τ. The dots are drawn on an altitude-azimuth grid, each one at the Sun's sky position at the time of the observation. See Annex B Table 1 for code.
	(b) For the same station, a photograph from the station of the view to the south. This is a wider and taller view – the sunprint occupies only 47 degrees in altitude (2 x Earth obliquity) and about 180 degrees in azimuth, while the photograph is both wider and taller than the sunprint dimensions.

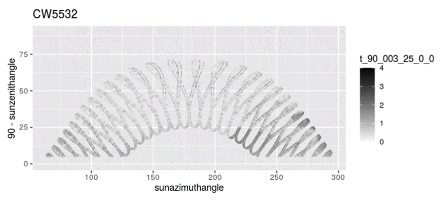
The uppermost/outermost arc is data from midsummer; the lowest/innermost arc is midwinter.

In the summertime sector, we see strong shading from the trees but in the winter the bare trunks and branches do not cast as much shade.

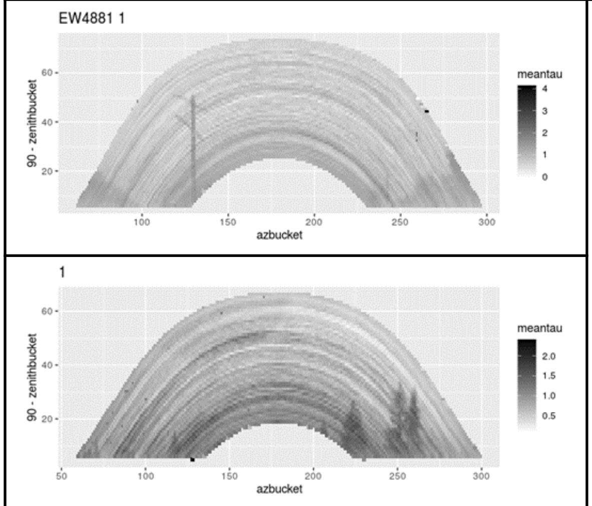
[illegible]

This figure shows us the station environment, suggesting a way to identify data that is shaded by vegetation and infrastructure.

There is a problem to solve: Figure 6 is affected by the fact that observations do not sample the sky evenly. Instead, the sunprint is composed of the figure 8, drawn over and over. This composition results from the fact that most stations take observations at the same times every day. But the sky position of the Sun taken at the same time every day for one year traces out figure called the analemma. To see a clearer illustration of the analemmas traced out by the sky position of the Sun at one station, consider Figure 7(a), the sunprint for a station that observes once/30 minutes. This station program comprises fewer observations per day, so the analemmas can be distinguished; the uneven sky sampling that results is clear.

	**Figure 7. Each Daily Observing Time Traces Out the Analemma Once/Year** Plot of all τ values from a CWOP station that observes once per thirty minutes (same time daily), each plotted at the respective sky position. The sky positions of the Sun trace out parts of 28 separate analemmas. For code to undertake such plots, see Annex B, Table 1, line 16.

To smooth out artifacts due to the analemma, we can group all observations into tiles and plot average tile values. Optimum tile size depends on the sky spacing of daily observations, which varies station to station, but we will omit that complication and use tiles that are one degree in azimuth and one degree in altitude.

	**Figure 8. Smoothed sunprints from a selection of CWOP stations**. In these smoothed sunprints, we can easily identify which data sectors are shaded by trees and infrastructure. For code to undertake such a plot, see Annex B, Table 1, line 30.

We can plot other statistical moments of τ as well. Figure 9 presents mean τ, standard deviation of τ, skew of τ, and kurtosis of τ, tile by tile. These depictions distinguish types of shading.

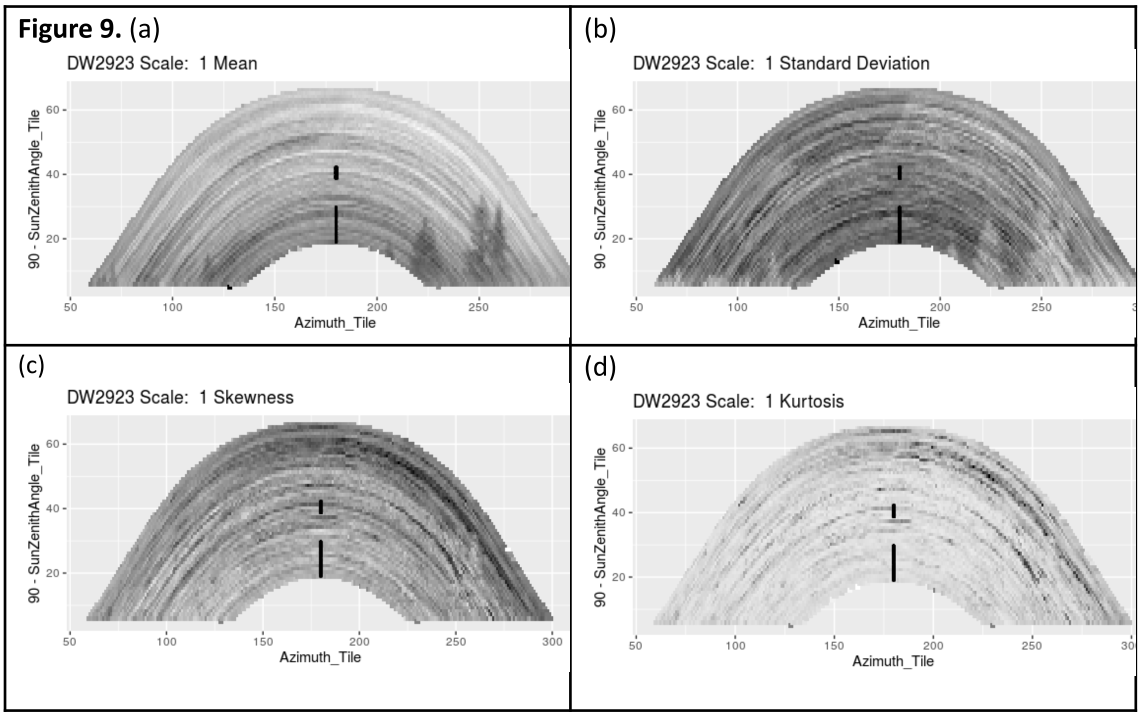


Figure 9. Statistical moments of τ . (a) Sunprint obtained from tile-wise mean values of τ. **(b, c, d)** Tile-wise values for the standard deviation of τ, the skewness of τ, and the kurtosis of τ. The semicircle seen along the upper edge is a shadow of an instrument on the weather station which cast a shadow during some years and was moved away in other years. Its statistical signature is different from that of the pine trees. For code to do this see Annex B, Table 1, line 31.

Using a combination of statistics, we can define masks that will pick out local shading. In the example below, a mask is defined so that vegetation and infrastructure are excluded.

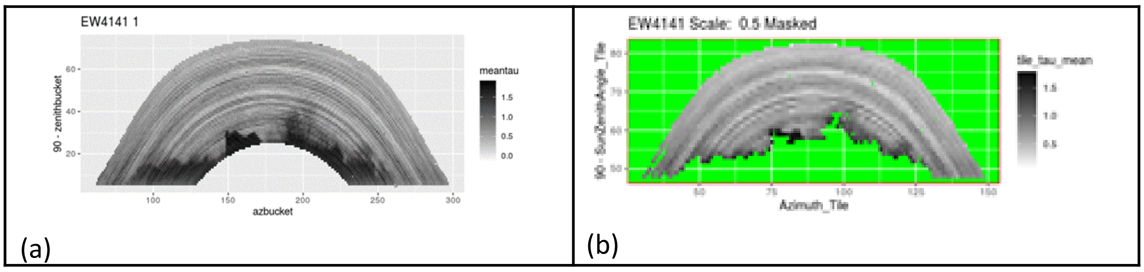


Figure 10. Statistical mask to remove shading. (a) Sunprint of τ, smoothed. (b) Sunprint after data has been masked.

Almost every station is partly shaded; however, valuable data can be obtained and selected at all of these locations if data is provided that is sufficient for development of a statistical mask. Masking depends on a long (or frequently-sampled) time series.

Approach 3: Histogram of optical depth

Histograms of optical depth are another summary of the difference between model and observation.

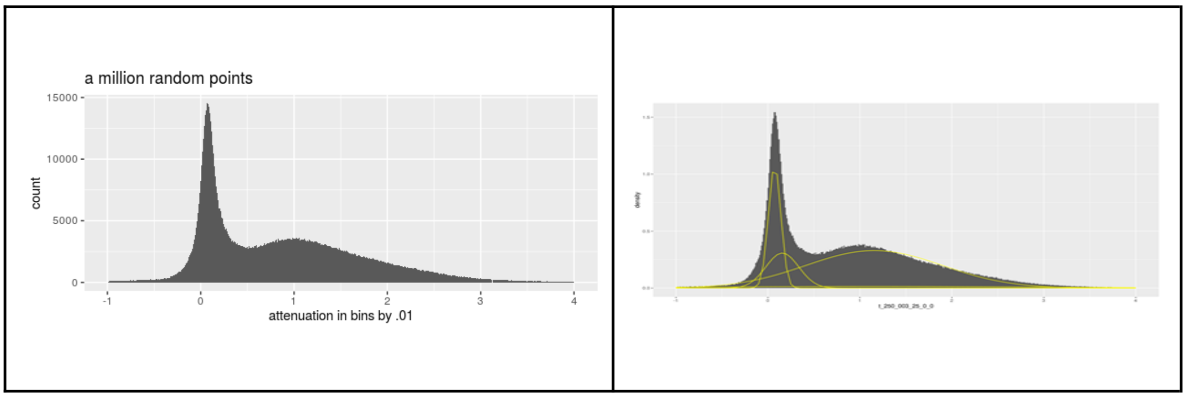


Figure 11. (a) Histogram of optical depth values, obtained from a million-point random extract from the database. (A subset is used for this figure because the database now comprises almost a billion observations. A subset is more tractable for statistical operations. The subset is scaled at one million to make it large enough to be representative for figures such as this one, but small enough to be tractable for statistical tests. Statistical operations other than this figure generally do not use the subset; rather, they use filters on the whole database.)

(b) Probability density function. Yellow lines present an optimized fit of the data to a Gaussian mixture model. See Annex B, Table 1, line 10 a,b for code to do this.

Conclusion

As CWOP data collects, it comprises an increasingly valuable means of putting to the test models of solar irradiance of the Earth's surface.

We are confident that this vast database will contribute to understanding the flux of solar radiation to the Earth' s surface.

Annex A
Tables

Table 1: Missing Days

2009 – None missing (from 2/18 forward)
2010 - 03/31, 05/25, 05/26, 8/1, 8/2, 8/3 (6 days)
2011 - 7/23, 7/28, 7/29, 7/30, 11/6 (5 days)
2012 - 1/7, 7/25 (2 days)
2013 - 2/26, 8/13 to 8/25, 8/31 (15 days)
2014 - 5/23, 5/24, 5/25, 5/26, 5/27 (5 days)
2015 – None missing
2016 – None missing
2017 – None missing
2018 – 1/18, 1/19, 10/19 (3 days)
2019 – None missing
2020 – None missing
2021 – None missing (through April 30, 2021)

Table 2: Typical values for τ

If model L is	and observed L is	then attenuation ratio is	and τ is
100	100	0	0.00
	95	0.05	0.05
	90	0.1	0.11
	80	0.2	0.22
	70	0.3	0.36
	60	0.4	0.51
	50	0.5	0.69
	37	0.63	0.99
	14	0.86	1.97
	5	0.95	3.00
	2	0.98	3.91
	1	0.99	4.61

Annex B
Making and Addressing an Archive in Scientific Units

Incremental changes to software and facilities used in the recipe for a Redshift database:

A recipe enabling users to set up a Redshift archive was provided in the ***Ten Year Report, Annex B.*** That recipe enabled users to set up a Redshift archive on Amazon Web Services, comprising the CWOP Solar Radiation Data Archive in parsed units, plus model comparators, Sun/Earth configuration data, and various other helper variables. Since March 2020, the computing environment has changed in a few respects such that the recipe provided a year ago would not work smoothly today. The key changes are listed below.

(1) Google Drive has changed, such that it is no longer possible to use links to the CWOP Archive to access data from your Amazon EC2.

Instead, what works now, is to download data from the Archive and move it to your MyDrive. If the daily archive files are in your own MyDrive, then the code here will work.

(2) Google now requires that the link between your EC2 and MyDrive be refreshed daily. This amounts to answering a query once/day.

(3) GPS Visualizer has changed, such that it is no longer possible to look up elevations for the entire CWOP dataset in a single query.

What GPSVisualizer objects to is to a single lookup submission that wanders over all the Earth's DEM tiles. Instead, a single lookup must be more geographically confined. It appears also that a single lookup is now limited to 1000 queries or fewer and those 1000 may not wander over all DEM tiles.

What works is to sort the data by latitude and longitude, as below:

```
latlon4lookup_sorted <- latlon4lookup[order(latitude,longitude),]
```

and write that to a file:

```
write.table(latlon4lookup_sorted,'~/CWOPSolar
Update/data/latlon4lookup_sorted.txt',sep=',',row.names=FALSE
,quote=FALSE,col.names=TRUE)
```

Export `latlon4lookup_sorted.txt` to the hard drive (instead of `latlon4lookup.txt` as in the ***Ten Year Report*** directions).

Working in the hard drive, break the sorted list down into ten or so files comprising tranches of fewer than 1000 lines each. There are about 8138 lines as of spring 2021; thus nine tranches. Each tranche should be composed with the header line `latitude,longitude`

The tranches comprise acceptable geographic subsets that GPS Visualizer will process. See directions in ***Ten Year Report.***

The results this year have yet another change from directions in the ***Ten Year Report*** -- namely, instead of a column named `elevation`, the Visualizer returns the same column now entitled `altitude`. If this column is renamed as `elevation` in each results file, and the results are assembled, then the assembled file can be returned to the EC2 as input to the parse process. See ***Ten Year Report.***

(4) Use of Amazon S3 storage has changed, such that addressing a bucket now requires designation of its region, either as the Amazon default (Virginia), your own default that you name as a variable set in startup.R, or within the call to place a file in the S3 bucket.

(5) Aginity Workbench has been superseded by Aginity Pro. I have not used it so I have no advice about the differences.

Meantime an alternative to Aginity has emerged, namely the Amazon SQL Query option that is attached to the Redshift service. The Amazon SQL Query option existed at the time of the Ten Year Report in 2020, but was not recommended, due to its impracticality. At that time, was impractical to use since it allowed only one query at a time. It appears that it now allows multiple queries, though it does so by treating multiple queries as one. That is, if you issue a hundred upcopy commands, and it finds a typo anywhere, it will do none of them.

A further limitation of the Amazon SQL IDE compared to Aginity Workbench is that permissions problems emerge if during one single session you use queries in the Amazon IDE, ***and also*** queries run from the EC2. In a single session, you can run queries from one or the other but alternating between them encounters barriers.

(6) The tidyverse aws.s3 package now deprecates use of the command src_postgres, used to make the connection between EC2 and Redshift. However, for the time being it works despite being deprecated.

Cookbook of code examples

In the ***Ten Year Report,*** Annex B provided at Table 1 a cookbook of usable code with which users could query a Redshift database made per the recipe. The same Table 1 is provided below with some revisions and additions.

Table 1: Code Examples

	Code snippet	*Output*
SAMPLE CODE FOR STARTUP.R: TO LOAD PACKAGES, MAKE CONNECTIONS		

1	`# Here is a startup.R file that will load the needed packages for all the code examples in this table.  NOTE that this is the startup.R file for data analysis, not for data parsing.`
	
	`substrRight <- function(x, n){`
	`  substr(x, nchar(x)-n+1, nchar(x))`	

```
} # that routine I found on stackoverflow

fun_prop <- function(x, mean, sd, proportion){
  proportion * dnorm(x = x, mean = mean, sd = sd)
} # this routine is from datacamp course on GMM

# install.packages("rmarkdown")
library(rmarkdown)
# install.packages("insol")
library(insol)
#install.packages("plyr")
library(plyr)
# install.packages("marelac")
library(marelac)
# install.packages("sqldf")
library(sqldf)
# install.packages("RODBC")
library(RODBC)
# install.packages("lubridate")
library(lubridate)
# install.packages("RPostgreSQL")
library(RPostgreSQL)

library(reshape2)
library(tibble)

# install.packages("rworldmap")
library(rworldmap)
# install.packages("gdata")
library(gdata)
# install.packages("ggmap")
library(ggmap)
# install.packages("broom")
library(broom)

library(grid)
library(png)
library(e1071)
library(xml2)
library(XML)
library(RCurl)
library(xts)

#install.packages("HelpersMG")
library(HelpersMG)

library(rlang)
library(curl)
library(crul)
library(tidyverse)
library(googledrive)
library(rgbif)
library(tidyverse)
library(aws.s3)

# install.packages("LaplacesDemon")
library(LaplacesDemon)
# install.packages("flexmix")
Library(flexmix)

Sys.setenv(
 "AWS_ACCESS_KEY_ID" = 'your access key',
  "AWS_SECRET_ACCESS_KEY" = 'your secret access
key',
  "AWS_REGION"="region you are using")

cwop_db<-src_postgres(host='your endpoint',
port='5439', dbname='dev', user='your username',
password='your password')

cwoparchive <- tbl(cwop_db, "cwoparchive")
```

```
millionpoints <- tbl(cwop_db, "millionpoints")

# The startup routine will result in the following
error messages:

Warning messages:
1: In fun(libname, pkgname) : couldn't connect to
display ":0"
2: `src_postgres()` is deprecated as of dplyr
1.0.0.
Please use `tbl()` directly with a database
connection
This warning is displayed once every 8 hours.
Call `lifecycle::last_warnings()` to see where this
warning was generated.


# This means that the tidyverse aws.s3 package now
deprecates use of the command src_postgres, used to
make the connection between EC2 and Redshift in the
below.  However, for the time being it works
despite being deprecated.
```

SAMPLE CODE FOR DATABASE-WIDE VISUALIZATION, STATISTICS, ANALYSES

2

```
# Find the last day of data in the archive
and store it in the environment, call it
displaydate, then display it.

# This code has no prerequisite except the
startup code in line 1.

latest<-cwoparchive %>%
  select(archivedate) %>%
  summarise(maxdate=max(archivedate))
latest_df<-data.frame(latest)
displaydate<-latest_df$maxdate
displaydate
```

```
"2021-04-30"
```

3

```
# Plot a time series of the number of
stations reporting to the archive each day
from beginning to final day in archive.

# This code has no prerequisite except the
startup code in line 1.

n_stations<-cwoparchive %>%
  group_by (archivedate) %>%
  select(stationname) %>%
  distinct()%>%
  summarise(daily_stations=n())
n_stations<-collect(n_stations)
n_stations_df<-data.frame(n_stations)

p<-ggplot(n_stations_df,aes(archivedate,daily
_stations))
p2<-p+geom_point()
p2+xlab("")+ylab("")+ggtitle("Fig. 2.  Number
of stations reporting in the CWOP Solar
Radiation Archive")
```

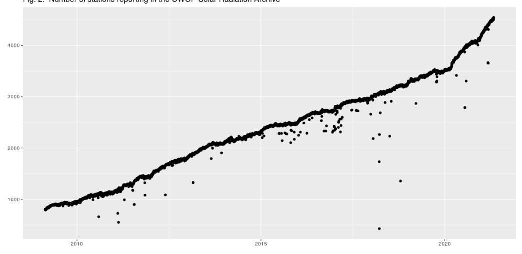
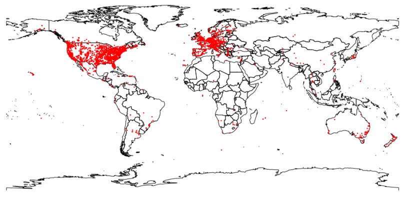
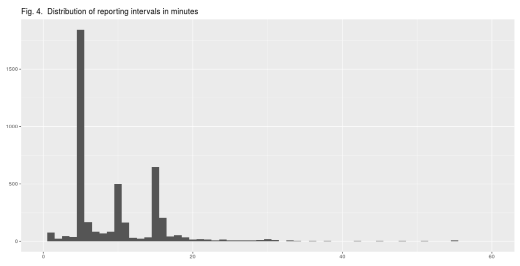


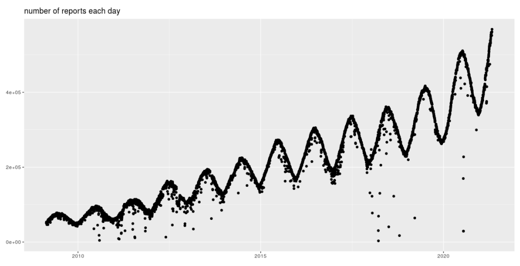
Fig. 2. Number of stations reporting in the CWOP Solar Radiation Archive

4

```
# Map of stations reporting on the last date
in the archive.

# This code uses the value of displaydate
computed above at line 2.

stations<-cwoparchive %>%
  filter(archivedate==displaydate) %>%
  select(latitude,longitude) %>%
  distinct()
```

	stations<-collect(stations)
	stations_df<-data.frame(stations)
	
	# plot the map
	
	stationmap<-getMap(resolution="low")
	plot(stationmap,xlim=c(-180,180),ylim=c(-80,80),asp=1)
	points(stations$longitude, stations$latitude, col = "red", pch=19, cex = 0.25)
		

5	# On a chosen equinox (replace bold date), identify distribution of reporting intervals. The selected date must be an equinox for this approach to work.
	# This code has no prerequisite except the startup code in line 1.
	
	n_reports <- cwoparchive %>%
	  filter(archivedate=='**2021-03-21**') %>%
	  group_by (stationname) %>%
	  summarise (daily_station_n=n())
	n_reports <- collect(n_reports)
	n_reports_df<-data.frame(n_reports)
	
	preports<-ggplot(n_reports_df,aes(720/daily_station_n))
	preports+geom_histogram(binwidth=1)+xlim(0,60)+xlab("Reporting interval in minutes")+ggtitle("**March 21, 2021** - Those reporting once/hour or more")+ylab("Number of stations")+xlab("")+ylab("")+ggtitle("Fig. 4. Distribution of reporting intervals in minutes")
		

6	# Plot a time series of the number of observations added to the archive each day.
	
	n_obs<-cwoparchive %>%
	  group_by (archivedate) %>%
	  summarise(daily_obs=n())
	n_obs<-collect(n_obs)
	n_obs_df<-data.frame(n_obs)
	
	pp<-ggplot(n_obs_df,aes(archivedate,daily_obs))
	pp2<-pp+geom_point()
	pp2+xlab("")+ylab("")+ggtitle("number of reports each day")
		

7	# The number of observations in the archive:
	
	latest<-cwoparchive %>%
	  summarise(totaln=n()) %>%
	  collect() %>%
	  data.frame ()
	total_n<-latest$totaln
	total_n
		926249415

8	#The number of stations that have ever reported:
	
	nstations <-cwoparchive %>%
	  summarise(a=n_distinct(stationname)) %>%
	  collect() %>%
	  data.frame()
	totalstatns <- nstations$a
	totalstatns
		10189

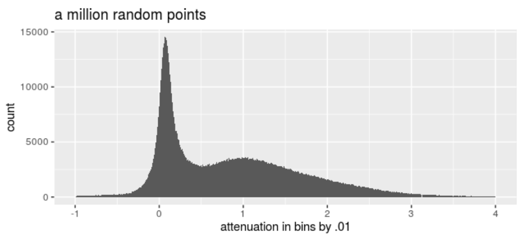
9	#The number of station-days:	

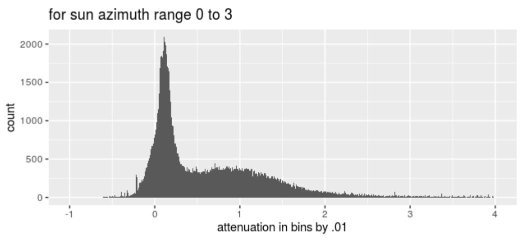
```
nstationdays <-cwoparchive %>%
  select(stationname,archivedate) %>%
  distinct() %>%
  summarise(count=n()) %>%
  collect() %>%
  data.frame ()
totstatndays<- nstationdays$count
totstatndays
```

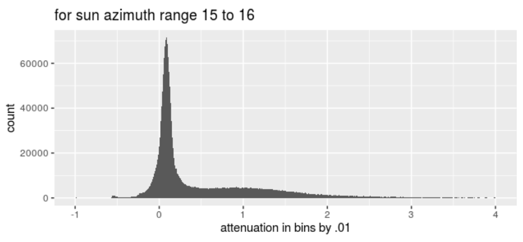
```
10453085
```

10
a

```
# Plot a histogram of the values of optical depth,
for a million randomly selected observations. For
model value for solar irradiance, it uses
t_250_003_25_0_0 (see Attachment 6, Table 1).

# This only works in a database where you have
created the table millionpoints

histodata <- millionpoints %>%
filter(sunzenithangle<90) %>%
select (t_250_003_25_0_0) %>%
collect(n=Inf) %>%
data.frame()
p<-ggplot(histodata,aes(t_250_003_25_0_0))
p+geom_histogram(binwidth=.01)+ggtitle("a million
random points")+xlim(-1,4)+xlab("attenuation in
bins by .01")

## Same as above but instead of a million random
observations, start with the whole database and
make a selection based on sun zenith angle.

szamin <- 0
szamax <- 3

histodata <- cwoparchive %>%

filter(sunzenithangle>szamin,
    sunzenithangle<szamax) %>%
select (t_250_003_25_0_0) %>%
collect(n=Inf) %>%
data.frame()
p<-ggplot(histodata,aes(t_250_003_25_0_0))
ggt <- paste("for sun azimuth
range",szamin,"to",szamax)

p+geom_histogram(binwidth=.01)+ggtitle(ggt)+xlim(-1
,4)+xlab("attenuation in bins by .01")
```



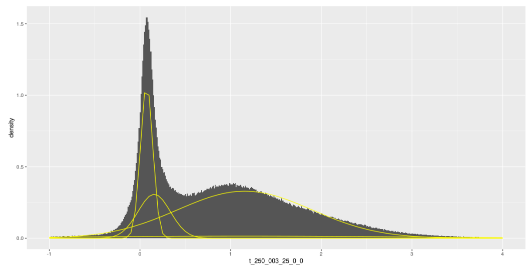


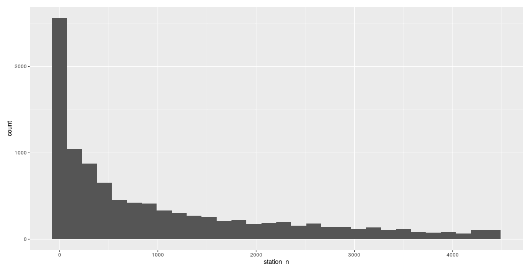


10
b

```
# Assuming you have collected data as above for a
histogram, and therefore that you have in the
environment a data frame called histodata with just
one column, t_250_003_25_0_0, this code analyzes
the resulting probability density function, by
modeling it as a mixture of Gaussians and
identifying their means, standard deviations, and
share of the whole dataset.

# It then superimposes those Gaussians on the
histogram.

mydata <- histodata %>%
  filter(is.finite(t_250_003_25_0_0))
decomposet <- flexmix (t_250_003_25_0_0 ~ 1,
   data=mydata, k=5,
   model=FLXMCnorm1(),
   control = list(tolerance = 1e-15, verbose = 1,
      iter = 1e4))
proportions <- prior(decomposet)
proportions
```

```
[…]
converged
[…]
proportions
[1] 0.13771338 0.06980049 0.18243321
0.61005292
[…]
        Comp.1
mean 0.1565659
sd   0.1783749
[…]
       Comp.2
mean 1.152506
sd   1.837425
[…]
         Comp.3
mean 0.07346395
sd   0.06730198
[…]
        Comp.4
mean 1.1554457
sd   0.7385999
```

	# The call to flexmix looks for five components although it usually finds only three or four. The point is not to prejudge the number of components. This way, The call above will tell you how many components it actually found. The routines below should be modified from three components as shown, to the number necessary.
	
	comp_1 <- parameters(decomposet, component = 1)
	comp_2 <- parameters(decomposet, component = 2)
	comp_3 <- parameters(decomposet, component = 3)
	comp_4 <- parameters(decomposet, component = 4)
	
	comp_1
	comp_2
	comp_3
	comp_4
	
	ggplot(mydata) + geom_histogram(aes(x = t_250_003_25_0_0, y = ..density..),binwidth=.01) +
	    stat_function(geom = "line",
	       color="yellow",fun = fun_prop,
	    args = list(mean = comp_1[1], sd = comp_1[2],
	    proportion = proportions[1])) +
	    stat_function(geom = "line",
	       color="yellow",fun = fun_prop,
	    args = list(mean = comp_2[1], sd = comp_2[2],
	    proportion = proportions[2]))+
	    stat_function(geom = "line",
	       color="yellow",fun = fun_prop,
	    args = list(mean = comp_3[1], sd = comp_3[2],
	    proportion = proportions[3]))+
	    stat_function(geom = "line",
	       color="yellow",fun = fun_prop,
	    args = list(mean = comp_4[1], sd = comp_4[2],
	    proportion = proportions[4]))+
	xlim(-1,4)
		

11	# Histogram showing distribution of stations by the number of days they have reported to the database.
	
	n_reports <- cwoparchive %>%
	  group_by (stationname) %>%
	  select(archivedate) %>%
	  distinct()%>%
	  summarise(station_n=n())
	# n_reports<-collect(station_n)
	n_reports_df<-data.frame(n_reports)
	ggplot(n_reports_df,aes(station_n)) +
	geom_histogram()
		

12	# The number of stations that reported on fewer than 10 days
	# Note: The following code example draws on the database n_reports_df computed above (line 11).
	
	lessthanten <- n_reports_df %>%
	  filter(station_n <= 10) %>%
	  summarise(n())
	lessthanten
		1531

13	# Count what percent of stations reported each met variable, on 2021-04-30
	
	mettable_df <- cwoparchive %>%
	    filter(archivedate =='2021-04-30') %>%
	    data.frame()
	
	aa<- mettable_df[!is.na(mettable_df$winddir),]
	winddirpct <- (dim(aa)[1]/dim(mettable_df)[1])*100
	
	aa<- mettable_df[!is.na(mettable_df$windknots),]
		[1] "98% of observations include wind direction. 98% of observations include wind speed in knots. 96% of observations include wind gust. 100% of observations include temperature. 96% of observations include rainfall in the last hour. 94% of observations include rainfall in the last 24h. 97% of observations include rainfall today. 100% of observations include relative humidity. 99% of observations include barometric pressure."

```
windknotspct <-
(dim(aa)[1]/dim(mettable_df)[1])*100

aa<- mettable_df[!is.na(mettable_df$gust),]
gustpct <- (dim(aa)[1]/dim(mettable_df)[1])*100

aa<- mettable_df[!is.na(mettable_df$temp),]
temppct <- (dim(aa)[1]/dim(mettable_df)[1])*100

aa<- mettable_df[!is.na(mettable_df$rainfallhour),]
rainfallhourpct <-
(dim(aa)[1]/dim(mettable_df)[1])*100

aa<- mettable_df[!is.na(mettable_df$rainfall24h),]
rainfall24hpct <-
(dim(aa)[1]/dim(mettable_df)[1])*100

aa<-
mettable_df[!is.na(mettable_df$rainfalltoday),]
rainfalltodaypct <-
(dim(aa)[1]/dim(mettable_df)[1])*100

aa<-
mettable_df[!is.na(mettable_df$relativehumidity),]
relativehumiditypct <-
(dim(aa)[1]/dim(mettable_df)[1])*100

aa<- mettable_df[!is.na(mettable_df$baropressure),]
baropressurepct <-
(dim(aa)[1]/dim(mettable_df)[1])*100

a1 <- paste(round(winddirpct),"%",' of observations
include wind direction.',sep="")
a2 <- paste(round(windknotspct),"%",' of
observations include wind speed in knots.',sep="")
a3 <- paste(round(gustpct),"%",' of observations
include wind gust.',sep="")
a4 <- paste(round(temppct),"%",' of observations
include temperature.',sep="")
a5 <- paste(round(rainfallhourpct),"%",' of
observations include rainfall in the last
hour.',sep="")
a6 <- paste(round(rainfall24hpct),"%",' of
observations include rainfall in the last
24h.',sep="")
a7 <- paste(round(rainfalltodaypct),"%",' of
observations include rainfall today.',sep="")
a8 <- paste(round(relativehumiditypct),"%",' of
observations include relative humidity.',sep="")
a9 <- paste(round(baropressurepct),"%",' of
observations include barometric pressure.',sep="")

paste(a1, a2, a3, a4, a5, a6, a7, a8, a9, sep="  ")
```

14

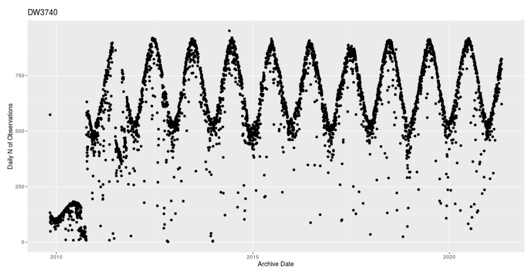
```
# Identify the text strings used by the largest
number of stations in the “tech” field, on a given
day of the archive.

thatday <- as.Date("2021-04-30")

n_stations<-cwoparchive %>%
  group_by (archivedate) %>%
  select(stationname) %>%
  distinct()%>%
  summarise(daily_stations=n())
n_stations<-collect(n_stations)
n_stations_df<-data.frame(n_stations)

nstationsthatday <- n_stations_df
[which(n_stations_df$archivedate==thatday),2]

mettable_df <- cwoparchive %>%
    filter(archivedate ==thatday) %>%
    data.frame()

maintechfields2 <- mettable_df %>%
  select(stationname,tech) %>%
  distinct() %>%
  group_by(tech) %>%
  summarize(nstationspertech = n()) %>%

mutate(tshare=100*(nstationspertech/nstationsthatda
y)) %>%
  arrange(desc(tshare))

maintechfields2
```

```
   tech               nstationspertech tshare
   <chr>                         <int>  <dbl>
 1 .DsIP                          1104   24.3
 2 AmbientCWOP.com                 653   14.4
 3 eMB51                           430   9.46
 4 .WD 31                          332   7.30
 5 .DsWLL                          316   6.95
 6 .DsVP                           294   6.47
 7 eCumulusDsVP                    210   4.62
 8 WeatherCatV312B34H31             65   1.43
 9 .weewx-4.5.1-Vantage             61   1.34
10 eMB50                            55   1.21
# … with 4,027 more rows
```

SAMPLE CODE FOR SINGLE-STATION VISUALIZATION, STATISTICS, ANALYSES

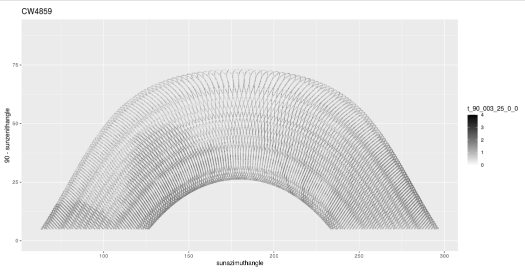
15

```
# For a single station, plot a time series of the
daily count of observations in the archive.

station<-"DW3740"
dailyn<-cwoparchive %>%
  filter(stationname==station) %>%
  group_by (archivedate) %>%
  summarise(ncount=n(),meanlat=mean(latitude),
meanlon=mean(longitude))
dailyn<-collect(dailyn)
dailyn_df<-data.frame(dailyn)
np<-ggplot(dailyn_df,aes(x=archivedate))
nplabels<-labs(x="Archive Date", y="Daily N of
Observations",size="meanlon")
npdailyn<-geom_point(data=dailyn_df,aes(y=ncount))
np2<- np+npdailyn+nplabels+ ggtitle(station)
np2
```



16

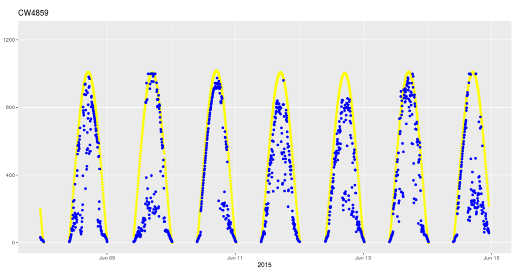
```
# A “sunprint” of site shading.  This code plots
attenuation of solar irradiance as a function of
the altitude and azimuth of the sun.
# The figure is from a station in suburban
Philadelphia for which we have a comparator
photograph.  The sunprint rightly depicts tree
trunks, branches, leaves, and a nearby building.

station<-"CW4859"
station_df <-cwoparchive %>%
filter(stationname==station,sunzenithangle<85.,l_90
_003_25_0_0<1000) %>%
select(z,linterpreted,l_90_003_25_0_0,sunzenithangl
e,sunazimuthangle,t_90_003_25_0_0,occultingeqr) %>%
collect(n=Inf) %>%
data.frame ()
p<-ggplot(station_df,aes(y=90-sunzenithangle,x=suna
zimuthangle))
q<-p+geom_point(aes(colour=t_90_003_25_0_0),size=0.
1,alpha=1/8)+ylim(0,90)+ggtitle(station)+scale_colo
```



```
ur_gradient(limits=c(0,4), low="white",
high="black", space="Lab")
q
```

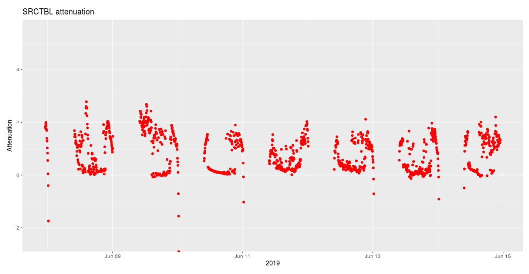
17

```
# Plot a few days of data from one station
(in blue), jointly with modeled clear-sky L
values for the same time and place (in
yellow).  Note that z is the time of the
data, but archivedate is not necessarily so
(archives are closed at 2300Z).

  station <- "CW4859"
  thisdataday_df<-cwoparchive %>%
    filter(stationname==station,archivedate >
'2015-06-07', archivedate < '2015-06-15') %>%

select(z,linterpreted,l_90_003_25_0_0,sunzeni
thangle,t_90_003_25_0_0) %>%
    collect(n=Inf) %>%
    data.frame()

p<-ggplot(thisdataday_df,aes(z,linterpreted))

lmodel<-geom_point(aes(y=l_90_003_25_0_0),col
our="yellow")
  lobs<-geom_point(colour="blue")
  plabs<-labs(x="Date", y="L in W per m2")
  p+lmodel+lobs+
ggtitle(station)+plabs+xlab("2015")+ylab("")+
ylim(0,1250)
```



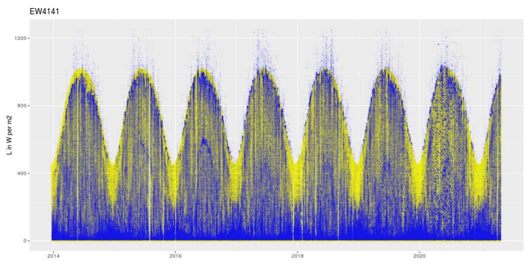
18

```
# For one station, a few days, plot a time series
of optical thickness, i.e., comparison between
observed and modeled L (see Attachment 6 to this
Annex for more about optical thickness).

  station <- "SRCTBL"
  thisdataday_df<-cwoparchive %>%
    filter(stationname==station,archivedate >
'2019-06-07', archivedate <
'2019-06-15',sunzenithangle<90.) %>%

select(z,linterpreted,insol90,sunzenithangle,t_90_0
03_25_0_0) %>%
    collect(n=Inf) %>%
    data.frame()
  p<-ggplot(thisdataday_df,aes(z,t_90_003_25_0_0))
  lobs<-geom_point(colour="red")
  plabs<-labs(x="2019", y="Attenuation")
  p+lobs+ ggtitle("SRCTBL attenuation")+plabs +
ylim(c(-2.5,5.5))
```

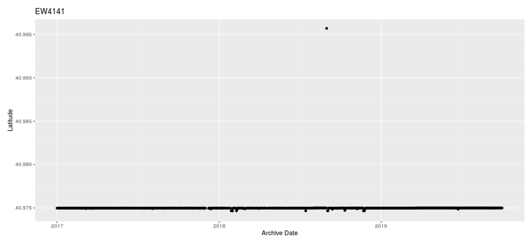
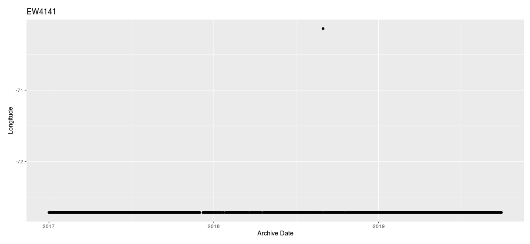


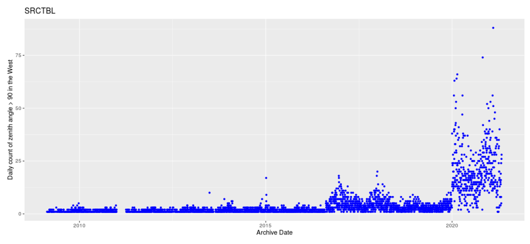
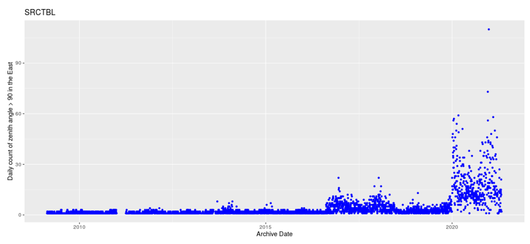
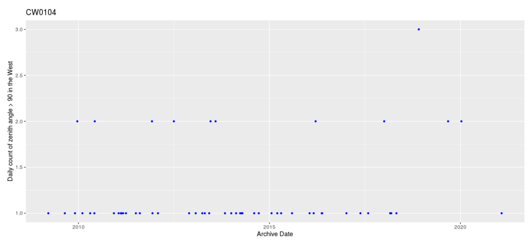
19

```
# Plot of data against model for one station over
its full archived time series of observations.

  station <- "EW4141"
  thisdata_df<-cwoparchive %>%
    filter(stationname==station) %>%

select(z,linterpreted,l_90_003_25_0_0,sunzenithangl
e,t_90_003_25_0_0) %>%
    collect(n=Inf) %>%
    data.frame()
  p<-ggplot(thisdata_df,aes(z,linterpreted))

lmodel<-geom_point(aes(y=l_90_003_25_0_0),colour="y
ellow",size=1,alpha=1/40)

lobs<-geom_point(colour="blue",size=0.25,alpha=1/40
)
  plabs<-labs(x="", y="L in W per m2")
```

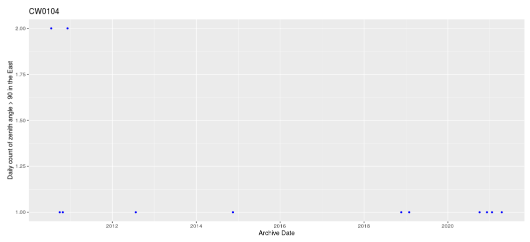


	p+lmodel+lobs+ plabs+ylim(0,1500)+ggtitle(station) +ylim(0,1250)	

20	# For a single station, plot a time series of the daily average value for reported latitude, then the same for longitude, **over a selected time range.** In this case, the tiny anomaly over a short period could pertain to a restart of the station, and a double check of coordinates.
	
	station<-"**EW4141**"
	dailyn<-cwoparchive %>%
	  filter(stationname==station) %>%
	  group_by (archivedate) %>%
	  summarise(ncount=n(),meanlat=mean(latitude),
	meanlon=mean(longitude))
	dailyn<-collect(dailyn)
	dailyn_df<-data.frame(dailyn)
	daystats<-ggplot(dailyn_df,aes(x=archivedate))
	
	latlabels<-labs(x="Archive Date", y="Latitude")
	latdailyn<-geom_point(data=dailyn_df,aes(y=meanlat)
	)
	lat2<- daystats+latdailyn+latlabels+
	ggtitle(station)
	lat2 +xlim(as.Date(c(**"2017-01-01","2019-09-30"**)))
	
	lonlabels<-labs(x="Archive Date", y="Longitude")
	londailyn<-geom_point(data=dailyn_df,aes(y=meanlon)
	)
	lon2<- daystats+londailyn+lonlabels+
	ggtitle(station)
	lon2 +xlim(as.Date(c(**"2017-01-01","2019-09-30"**)))
		
		

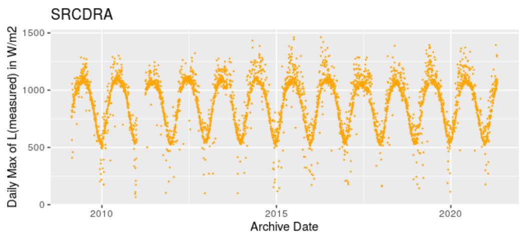
21	# For a single station, plot a time series of the daily count of the reports of solar irradiance that occur when the sun is not above the horizon. First figure is near sunset (a few are expected due to effect of archiving interval); second figure is before sunrise which ideally reflects only the limitations of APRS precision in location reporting. A high count on either side calls for examination of the timing of these reports. Station not calibrated? Reported coordinates are wrong?
	
	station<-"SRCTBL"
	zanomalies<-cwoparchive %>%
	    filter(stationname==station, sunzenithangle > 90.0, sunazimuthangle > 180) %>%
	    group_by (archivedate) %>%
	    summarise(nxzenithangle=n())
	zanomalies<-collect(zanomalies)
	zanomalies_df<-data.frame(zanomalies)
	zp <-ggplot(zanomalies_df,aes(x=archivedate))
	zplabels<-labs(x="Archive Date", y="Daily count of zenith angle > 90 in the West")
	zpanomalies<-geom_point(data=zanomalies_df,aes(x=archivedate,y=nxzenithangle),size=1,color="blue")
	zp2<- zp+zpanomalies+zplabels+ggtitle(station)
	zp2
	
	zanomalies<-cwoparchive %>%
	  filter(stationname==station, sunzenithangle > 90.0, sunazimuthangle < 180) %>%
	  group_by (archivedate) %>%
	  summarise(nxzenithangle=n())
	zanomalies<-collect(zanomalies)
	zanomalies_df<-data.frame(zanomalies)
	zp <-ggplot(zanomalies_df,aes(x=archivedate))
	zplabels<-labs(x="Archive Date", y="Daily count of zenith angle > 90 in the East")
		
		
		

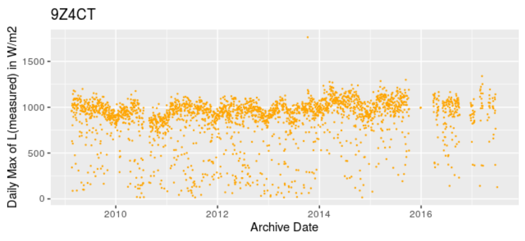
```
zpanomalies<-geom_point(data=zanomalies_df,aes(x=ar
chivedate,y=nxzenithangle),size=1,color="blue")
zp2<- zp+zpanomalies+zplabels+ggtitle(station)
zp2

# The number of L reports for times when the sun is
below the horizon varies from year to year,
suggesting calibration varies as pyranometers are
swapped annually. The number of such L reports
jumps in 2020, this time because data frequency is
increased.

# Same figures for CW0104
```



22

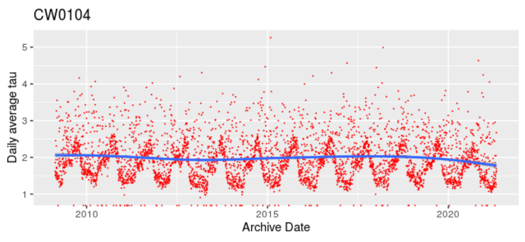
```
# Time series of the daily value of L-max for a
given station. The display is limited to L < 2000
to eliminate glitches. This display helps show what
software a station is using, because they vary in
how L > 1000 is depicted, folded over, or omitted.

station<-"SRCDRA"
dailylmax<-cwoparchive %>%
    filter(stationname==station,sunzenithangle
<=85.0, linterpreted<2000) %>%
    group_by (archivedate) %>%
    summarise(lmax=max(linterpreted))
dailylmax<-collect(dailylmax)
dailylmax_df<-data.frame(dailylmax)
mp<-ggplot(dailylmax_df,aes(x=archivedate))
mplabels<- labs(x="Archive Date", y="Daily Max of
L(measured) in W/m2")
mplmax<-geom_point(aes(y=lmax),
size=0.2,colour="orange")
mp2<-mp+mplabels+mplmax+ ggtitle(station)
mp2

# Same for 9Z4CT.  Neither of these stations shows
the folding problem – data L>1000 is not capped,
e.g., at 998 or 1000 as some software does.
```

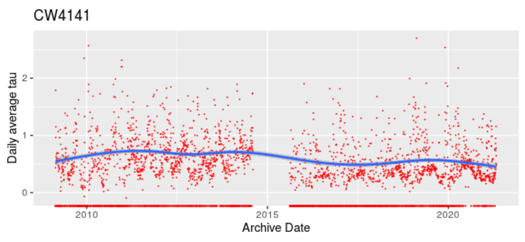


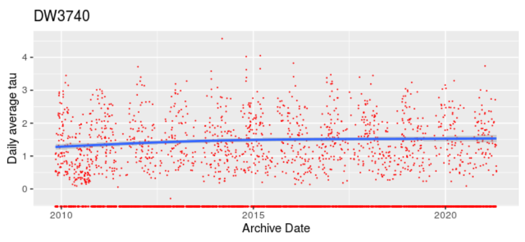


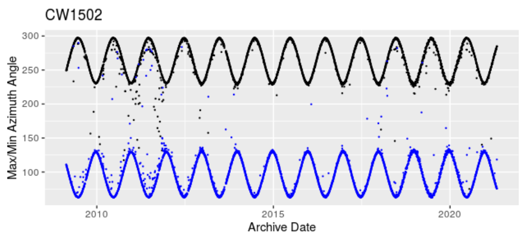
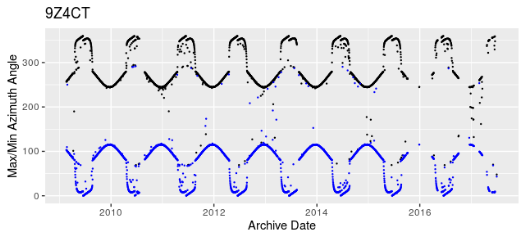
23

```
# Daily mean value of optical thickness.  In this
case we use the model value t_250_003_25_0_0.

station<-"CW0104"
dailymeantau<-cwoparchive %>%
  filter(stationname==station) %>%
  group_by (archivedate) %>%
  summarise
(meant_250_003_25_0_0=mean(t_250_003_25_0_0))

dailymeantau<-collect(dailymeantau)
dailymeantau_df<-data.frame(dailymeantau)

p<-ggplot(dailymeantau_df,aes(x=archivedate))
plabels<-labs(x="Archive Date", y="Daily average
tau")
pmeantau<- geom_point(aes(y=meant_250_003_25_0_0),
size=0.02,colour="red")
pplot <- p + plabels + pmeantau + ggtitle(station)
+stat_smooth(aes(y=meant_250_003_25_0_0))
pplot

# Same for CW4141
# Same for DW3740
```

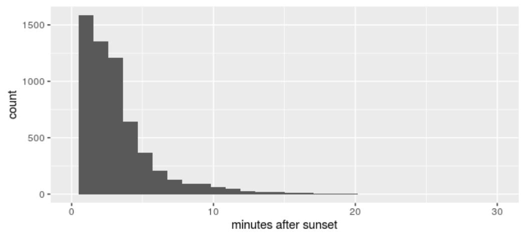


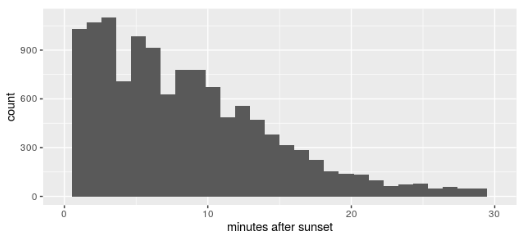


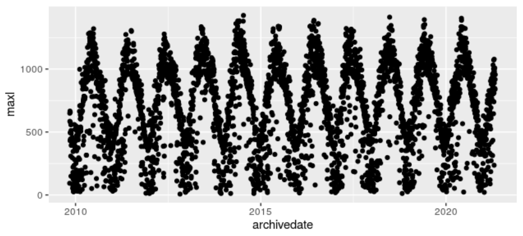
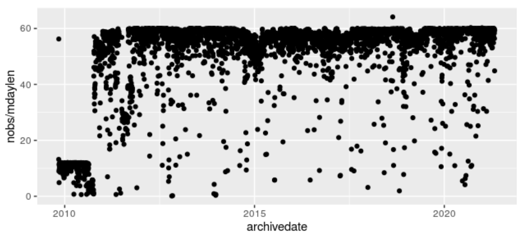
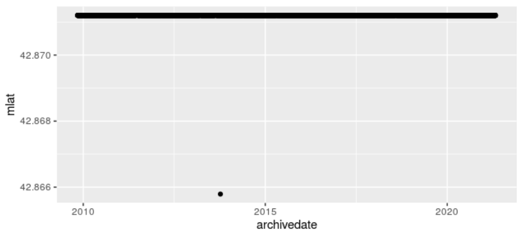
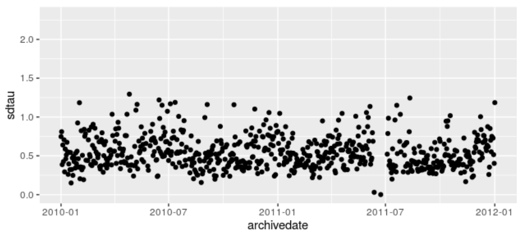


24	`# Daily maximum and minimum value of sunazimuthangle`
	
	`station<-"CW1502"`
	`dailymaxazimuth<-cwoparchive %>%`
	`  filter(stationname==station,sunzenithangle <= 85.0, linterpreted<2000) %>%`
	`  group_by (archivedate) %>%`
	`  summarise(max_azimuth=max(sunazimuthangle), min_azimuth=min(sunazimuthangle))`
	`dailymaxazimuth<-collect(dailymaxazimuth)`
	`dailymaxazimuth_df<-data.frame(dailymaxazimuth)`
	`p<-ggplot(dailymaxazimuth_df,aes(x=archivedate))`
	`plabels<-labs(x="Archive Date", y="Max/Min Azimuth Angle")`
	`pdailyazmax<-geom_point(aes(y=max_azimuth), size=0.2,colour="black")`
	`pdailyazmin<-geom_point(aes(y=min_azimuth), size=0.2,colour="blue")`
	`pplot <- p+plabels + pdailyazmax + pdailyazmin + ggtitle(station)`
	`pplot`
	
	`# Same for 9Z4CT, a tropical station 10.7N near Port of Spain.  This is why azimuth max/min has discontinuities - this has to do with the sun moving from northern sky to southern and back.`
		
		

25	`# For one station, a list of all the words ever used in the tech field.`
	
	`station <- "EW2020"`
	`mydata <- cwoparchive %>%`
	`  filter(stationname==station) %>%`
	`  select(tech) %>%`
	`  data.frame()`
	`mydatav2 <- str_replace_all(mydata[,],"([1234567890])"," ")`
	`mydatav3<-str_replace_all(mydatav2,"[[:punct:]]"," ")`
	`myuniquedata <-unique(unlist(str_split(mydatav3," ")))`
	`mydata[1:5,]`
	`myuniquedata`
	
	`# Same for EW4141`
	`# Same for 9Z4CT`
		`> mydata[1:5,]`
		`[1] ".DsIP-VP" ".DsIP-VP" ".DsIP-VP" ".DsIP-VP" ".DsIP-VP"`
		`> myuniquedata`
		`[1] ""      "DsIP" "VP"    "DsVP"`
		
		`> mydata[1:5,]`
		`[1] ".DsIP-VP"      ".DsIP-VP" "eCumulusDsVP" "eCumulusDsVP" "eCumulusDsVP"`
		`> myuniquedata`
		`[1] ""              "DsIP"          "VP" "eCumulusDsVP" "WFL"`
		`[6] "eMH"`
		
		`> mydata[1:5,]`
		`[1] ".DsVP" ".DsVP" ".DsVP" ".DsVP" ".DsVP"`
		`> myuniquedata`
		`[1] ""      "DsVP"`

26	`# At a single station, plotting the distribution of offsets between time of sunset and timestamp on observations that occur soon after sunset.  This plot may help establish the upper limit on archiving interval, which is otherwise hard to discover.`
	
	`station <- "DW3740"`
	`stationdata <- cwoparchive %>%`
	`  filter(stationname==station) %>%`
	`  data.frame()`
	
	`mydata <- stationdata %>%`
	`  filter(sunzenithangle > 90.0, sunazimuthangle>180 )`
	`basejd <- trunc(JD(mydata$z))`
	`basejdwaypoints <- data.frame(daylength(mydata$latitude,mydata$longitude, basejd,0)) # sunset in local time`
	`jdaddend <- ((basejdwaypoints$sunset)-12)/24`
	`jdsunset <- basejd+jdaddend`
	`jdsunsettime <- JD(jdsunset,inverse=TRUE)`
	`sunsetoffset <- mydata$z - jdsunsettime`
		DW3740
		
		
		SRCTBL

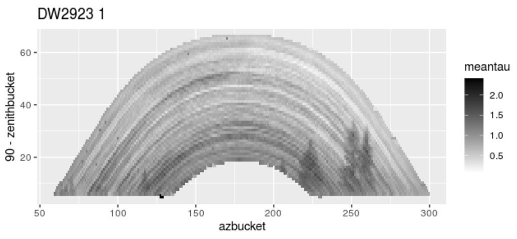
	sunsetoffsetmin <- data.frame(as.numeric(sunsetoffset)) names(sunsetoffsetmin) <- "offset" ggplot(data=sunsetoffsetmin, aes(offset/60))+geom_histogram()+xlim(0,30)+xlab("minutes after sunset") # Then same for SRCTBL # Note that for a station east of Greenwich, the adjustment from JD to UTC and back would be different…	

27
ok
	# For a single station, plot a time series of daily summary statistics. The final example includes a time filter to select a range of dates.
	
	station <- "DW3740"
	bagrovodata <- cwoparchive %>%
	  filter(stationname==station) %>%
	  data.frame()
	save(bagrovodata,file="bagrovodata.RData")
	
	bagrovodays <- bagrovodata %>%
	  filter(sunzenithangle < 90) %>%
	  group_by(archivedate) %>%
	
	summarize(meantau=mean(t_90_003_25_0_0),maxl=max(linterpreted),minl=min(linterpreted),
	
	maxeqr=max(occultingeqr),mineqr=min(occultingeqr),sdtau=sd(t_90_003_25_0_0),
	
	mjd=mean(jd),mlat=mean(latitude),mlon=mean(longitude),mhite=mean(hite),
	            mdaylen=mean(daylength),
	            nobs=n()) %>%
	  data.frame()
	
	ggplot(data=bagrovodays,
	aes(x=archivedate,y=maxl))+geom_point()
	ggplot(data=bagrovodays,
	aes(x=archivedate,y=nobs/mdaylen))+geom_point()
	ggplot(data=bagrovodays,
	aes(x=archivedate,y=mlat))+geom_point()
	d1<-as.Date('2010-01-01')
	d2<- as.Date('2012-01-01')
	ggplot(data=bagrovodays,
	aes(x=archivedate,y=sdtau))+geom_point()+xlim(d1,d2)
	)
		
		
		
		

28	# To re-parse a station which reports out-of-format position data, ***See Ten Year Report, Annex B, Attachment 7.*** The code is not general. To be updated.	
29	# Using Bayes predictive methods to repair stations whose data is afflicted by the protocol error that folds down L > 1000, , ***See Ten Year Report, Annex B, Attachment 7.*** The code is not general. To be updated.	

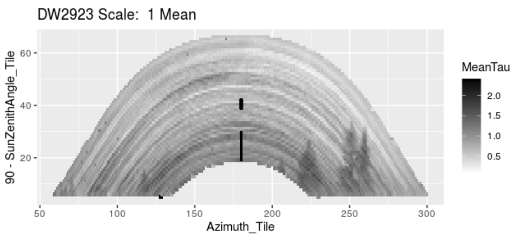
30

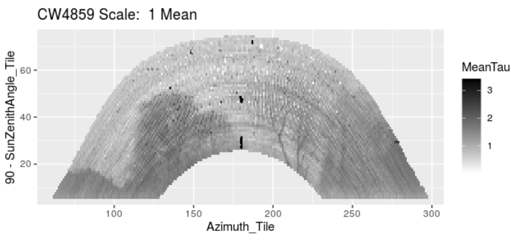
```
# Smooth a sunprint
# Collect the station’s data

station<-"DW2923"
station_df <-cwoparchive %>%
filter(stationname==station,sunzenithangle<85.) %>%
select(z,sunzenithangle,sunazimuthangle,t_250_003_2
5_0_0,occultingeqr) %>%
collect(n=Inf) %>%
data.frame ()

# embed a scale for the tiles
pscale <- 1

# Bin observations per sky position of sun.
by_sun_position<-
group_by(station_df,azbucket=floor(pscale*(station_
df$sunazimuthangle)),zenithbucket=floor(pscale*(sta
tion_df$sunzenithangle)))

# For each bin, record the mean
tile_the_sky<-summarise(by_sun_position,meantau=mea
n(t_250_003_25_0_0))

# And plot the means
p<-ggplot(tile_the_sky,aes(y=90-zenithbucket,x=azbu
cket)) # now we plot it
ptitl <- paste (station, pscale)
p+geom_point(aes(colour=meantau),shape=15)
+scale_colour_gradient(low="white",high="black",spa
ce="Lab")+ggtitle(ptitl)
```



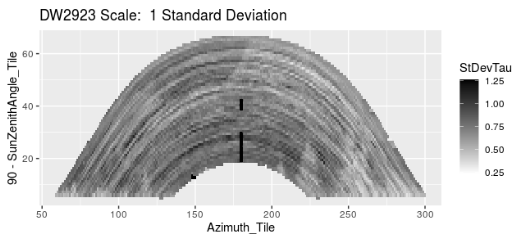
31

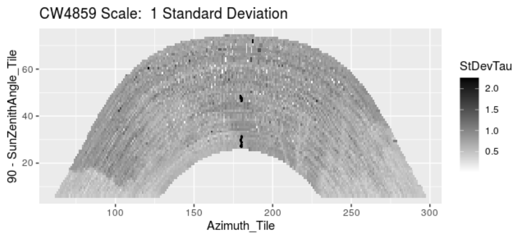
```
# Plot statistical moments and add meridian symbols
for data where occultingeqr is between 2.3 and 4
Rearth or else is greater than 75000 km.

# Collect station data

station<-"DW2923"
station_df <-cwoparchive %>%
filter(stationname==station,sunzenithangle<85.)%>%
  collect(n=Inf) %>%
  data.frame ()

# Pick out data where occultingeqr is between 2.3
and 4 Rearth or else is greater than 75000 km.
All data is already flagged in the system by the
variable eqr – which equatorial radius are we
looking through. (See Ten Year Report Annex B for
the geometry of eqr.)  Be careful because use of
eqr depends on whether station is NH or SH; we are
going to choose only NH just here.  And the
geometry is only right at noon, and it assumes
circular orbits.  Even so.  … Set a few scales we
can change in case I want to look at some other
range.

Re <- 6.371
Band2p5 <- 2.3*Re
Band5 <- 4*Re

# Subset the station data to pick out data where
occulting eqr is between 2.3 and 4 Rearth

station_df_paint_band1 <- station_df %>%
  filter(occultingeqr>Band2p5 & occultingeqr <
Band5 & sunazimuthangle > 179.75 & sunazimuthangle
< 180.25) %>%
  data.frame()
```

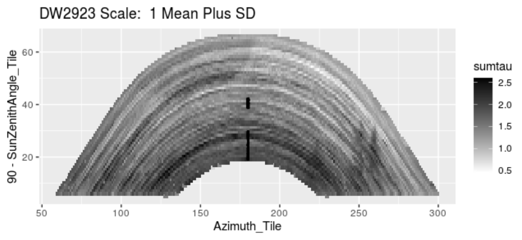


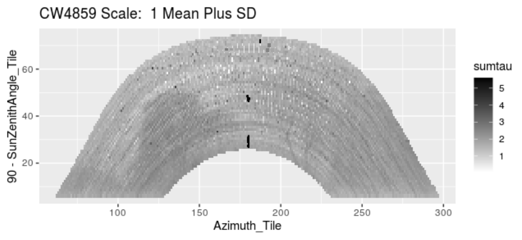


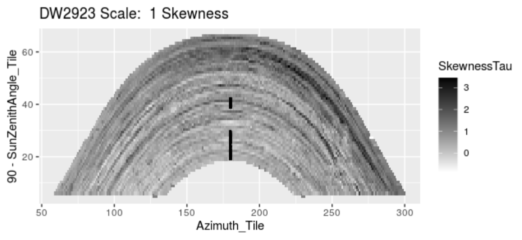
```
# A plot element to make a symbol on that band

qband1 <-
geom_point(data=station_df_paint_band1,aes(x=sunazi
muthangle,y=90-sunzenithangle),size=0.1)

# Subset the station data near noontimes, to pick
out eqr greater than 75000 km.  This band ends at
the spring equinox.

station_df_paint_band3 <- station_df %>%
  filter(occultingeqr>75 & sunazimuthangle > 179.5
& sunazimuthangle < 180.5) %>%
  data.frame()

# A plot element to make a symbol on that band

qband3 <-
geom_point(data=station_df_paint_band3,aes(x=sunazi
muthangle,y=90-sunzenithangle), size=0.1)

# Embed a scale for the tiles
pscale <- 1

# Bin observations per sky position of sun.

by_sun_position<-
group_by(station_df,Azimuth_Tile=floor(pscale*(stat
ion_df$sunazimuthangle)),SunZenithAngle_Tile=floor(
pscale*(station_df$sunzenithangle)))

# For each tile, compute mean value of tau.

tile_the_sky<-summarise(by_sun_position, MeanTau
=(mean(t_90_003_25_0_0)))

# plot each tile with a gray scale color related to
the mean value of tau.  Add in symbols for bands 1
and 2 defined above

p<-ggplot(tile_the_sky,aes(y=90-SunZenithAngle_Tile
,x=Azimuth_Tile))
ptitl <- paste (station, "Scale: ",pscale, "Mean")
p+ geom_point(aes(colour=MeanTau),shape=15) +
ggtitle(ptitl) + qband1 + qband3 +
scale_colour_gradient(low="white",high="black",spac
e="Lab")

# For each tile, compute sd of tau.

tile_the_sky<-summarise(by_sun_position, StDevTau
=(sd(t_90_003_25_0_0)))

# plot each tile with a gray scale color related to
the sd of tau.  Add in symbols for bands 1 and 2
defined above

p<-ggplot(tile_the_sky,aes(y=90-SunZenithAngle_Tile
,x=Azimuth_Tile))
ptitl <- paste (station, "Scale: ",pscale,
"Standard Deviation")
p+ geom_point(aes(colour=StDevTau),shape=15) +
ggtitle(ptitl) + qband1 + qband3 +
scale_colour_gradient(low="white",high="black",spac
e="Lab")

# For each tile, compute sum of mean and sd of tau.
This is just an example of how a person might do
channel math to distinguish object types.  It is
clumsy – it is just a skeleton of how to do it.
```

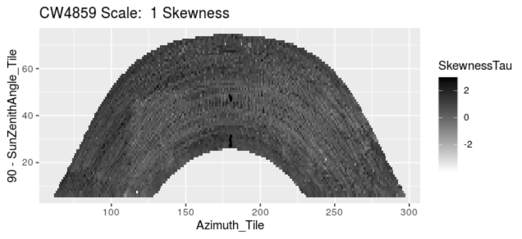




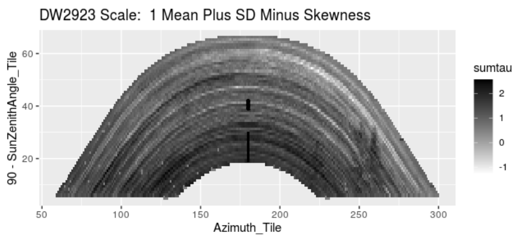


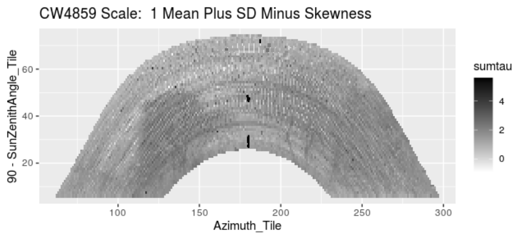


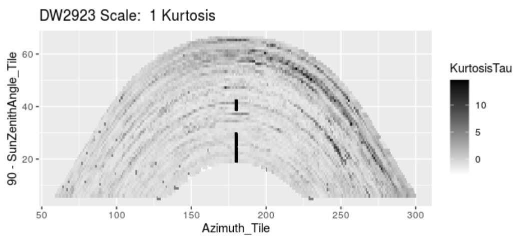


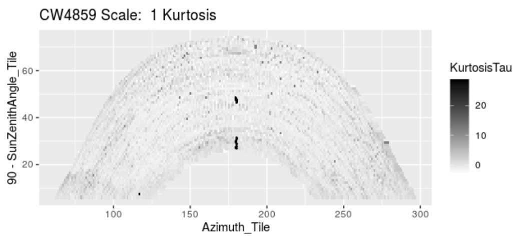


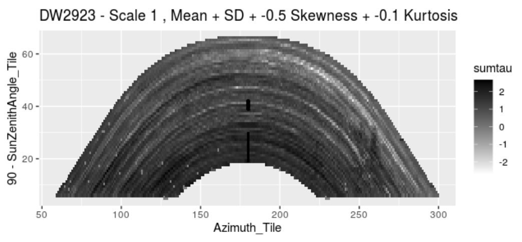
```
tile_the_sky<-summarise(by_sun_position,sumtau=(sd(
t_90_003_25_0_0)+mean(t_90_003_25_0_0)))

# plot each tile with a gray scale color related to
the sum of mean and sd of tau.  Add in symbols for
bands 1 and 2 defined above

p<-ggplot(tile_the_sky,aes(y=90-SunZenithAngle_Tile
,x=Azimuth_Tile))
ptitl <- paste (station, "Scale: ",pscale, "Mean
Plus SD")
p+ geom_point(aes(colour=sumtau),shape=15) +
ggtitle(ptitl) + qband1 + qband3 +
scale_colour_gradient(low="white",high="black",spac
e="Lab")

# For each tile, compute skewness of tau.

tile_the_sky<-summarise(by_sun_position,
SkewnessTau =(skewness(t_90_003_25_0_0)))



# plot each tile with a gray scale color related to
the skewness of tau.  Add in symbols for bands 1
and 2 defined above

p<-ggplot(tile_the_sky,aes(y=90-SunZenithAngle_Tile
,x=Azimuth_Tile))
ptitl <- paste (station, "Scale: ",pscale,
"Skewness")
p+ geom_point(aes(colour=SkewnessTau),shape=15) +
ggtitle(ptitl) + qband1 + qband3 +
scale_colour_gradient(low="white",high="black",spac
e="Lab")

# For each tile, compute the sum of mean and sd
minus 0.5* skewness.  Just one combination, not all
that carefully optimized, an option that could be
used for a mask for trees and infrastructure

tile_the_sky<-summarise(by_sun_position,sumtau=(sd(
t_90_003_25_0_0)+mean(t_90_003_25_0_0)-0.5*skewness
(t_90_003_25_0_0)))

# plot each tile with a gray scale color related to
the combination just noted (mean plus sd minus 0.5
* skewness).  Add in symbols for bands 1 and 2
defined above

p<-ggplot(tile_the_sky,aes(y=90-SunZenithAngle_Tile
,x=Azimuth_Tile))
ptitl <- paste (station, "Scale: ",pscale, "Mean
Plus SD Minus Skewness")
p+ geom_point(aes(colour=sumtau),shape=15) +
ggtitle(ptitl) + qband1 + qband3 +
scale_colour_gradient(low="white",high="black",spac
e="Lab")
```

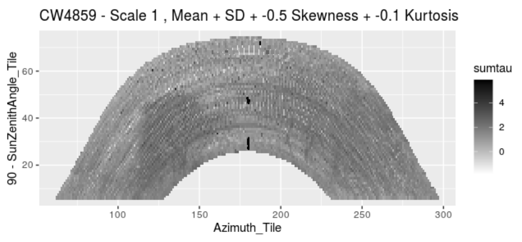








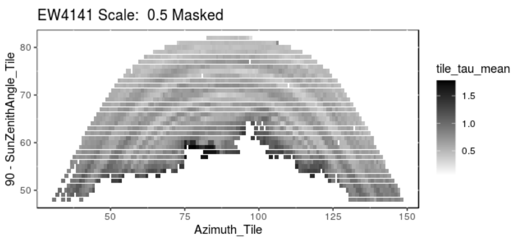




```
# For each tile, compute kurtosis

tile_the_sky<-summarise(by_sun_position,
KurtosisTau =(kurtosis(t_90_003_25_0_0)))

# plot each tile with a gray scale color related to
the kurtosis of tau.  Add in symbols for bands 1
and 2 defined above

p<-ggplot(tile_the_sky,aes(y=90-SunZenithAngle_Tile
,x=Azimuth_Tile))
ptitl <- paste (station, "Scale: ",pscale,
"Kurtosis")
p+ geom_point(aes(colour=KurtosisTau),shape=15) +
ggtitle(ptitl) + qband1 + qband3 +
scale_colour_gradient(low="white",high="black",spac
e="Lab")

# The stats have very different ranges. For the
next combination we introduce a bit of scaling for
flexibility.

skewscale <- -0.5
kurtosisscale <- -0.1

# And with that, we devise a combination, sd plus
mean plus scaled skewness plus scaled kurtosis

tile_the_sky<-summarise(by_sun_position,sumtau=(sd(
t_90_003_25_0_0)+mean(t_90_003_25_0_0)+skewscale*sk
ewness(t_90_003_25_0_0)+kurtosisscale*kurtosis(t_90
_003_25_0_0)))

# And plot the result

p<-ggplot(tile_the_sky,aes(y=90-SunZenithAngle_Tile
,x=Azimuth_Tile))
ptitl <- paste (station, "- Scale",pscale, ", Mean
+ SD +",skewscale,"Skewness
+",kurtosisscale,"Kurtosis")
p+ geom_point(aes(colour=sumtau),shape=15) +
ggtitle(ptitl) + qband1 + qband3 +
scale_colour_gradient(low="white",high="black",spac
e="Lab")
```

32

```
# This is example code for defining and applying a
mask.  (This is not a recommended mask or a
particularly good one.)

station<-"EW4141"
station_df <-cwoparchive %>%
filter(stationname==station,sunzenithangle<85.) %>%
select(z,sunzenithangle,sunazimuthangle,t_90_003_25
_0_0,occultingeqr) %>%
collect(n=Inf) %>%
data.frame ()

pscale <- 1/2
station_stats_df <- station_df %>%
group_by(Azimuth_Tile=floor(pscale*(station_df$suna
zimuthangle)),
SunZenithAngle_Tile=floor(pscale*(station_df$sunzen
ithangle))) %>%

mutate (
  tile_tau_mean = mean(t_90_003_25_0_0),
  tile_tau_sd = sd(t_90_003_25_0_0),
```



```
  tile_tau_skewness = skewness(t_90_003_25_0_0),
  tile_tau_kurtosis = kurtosis(t_90_003_25_0_0) )

station_masked <- station_stats_df %>%
  filter(
    tile_tau_mean < 1.0
   |(tile_tau_mean > 1.0 & tile_tau_sd > .7 ))

p<-ggplot(station_masked,
aes(y=90-SunZenithAngle_Tile,x=Azimuth_Tile))
ptitl <- paste (station, "Scale: ",pscale,
"Masked")
p+
geom_point(aes(colour=tile_tau_mean),shape=15) +
ggtitle(ptitl)+ scale_colour_gradient(
low="white",high="black",space="Lab") +
theme(panel.background =
element_rect(fill = 'white', colour = 'black'))
```

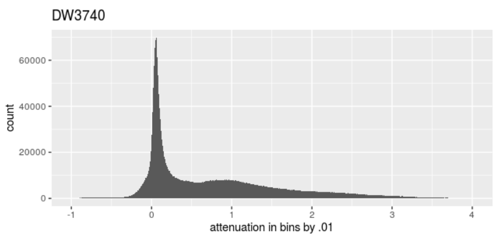
33

```
#  Select and extract a set of APRS reports and
archive dates for a station and a period of time.

station <- "SRCSXF"
thisdata_df<-cwoparchive %>%
  filter(stationname==station,archivedate >
'2019-12-15', archivedate < '2020-01-15') %>%
  select(report,archivedate) %>%
  collect(n=Inf) %>%
  data.frame()
thisdata_df <- arrange(thisdata_df,archivedate)
write.csv(thisdata_df,'srcsxfdata.csv')
```

```
   Report     Archivedate
1
SRCSXF>161848z4343.80N/09637.20W_196/007t022L263h9
2  2019-12-16
2
SRCSXF>161800z4343.80N/09637.20W_256/006t022L335h9
2  2019-12-16
3
SRCSXF>161742z4343.80N/09637.20W_265/006t021L240h9
3  2019-12-16


[… and so on…]
```

34

```
# Plot a histogram of the values of optical depth
for one station.

station <- "DW3740"

histodata <-cwoparchive %>%
  filter(stationname==station) %>%
  select (t_90_003_25_0_0) %>%
  collect(n=Inf) %>%
  data.frame()
p<-ggplot(histodata,aes(t_90_003_25_0_0))
p+geom_histogram(binwidth=.01)+ggtitle(station)+xli
m(-1,4)+xlab("attenuation in bins by .01")

# Note:  to identify the modes of histogram,
# or alternatively to decompose a histogram into
components assumed to be Gaussian distributions,
plot the result, and present the parameters of
those distributions,
# see lines 10b and 10c above, where the code is
set out for a multi-station data collection.
```



35

```
# Filter the database.

mydata2009 <- cwoparchive %>%
  filter(linterpreted != 998,
         latitude >= 0.0,
         sunazimuthangle >= 175,
         sunazimuthangle <= 185,
         archivedate > '2009-06-09',archivedate <
'2009-06-21') %>%
 # select (t_250_003_25_0_0) %>%
  collect(n=Inf) %>%
  data.frame()

# You can select dates and times by using z instead
of archivedate – they are different by a few hours,
since z goes by GMT while archivedate stops
collecting an hour before that.

myzdata2009 <- mydata2009 %>%
filter(z > '2009-06-11',z < '2009-06-12')
```

36

```
# Fit a GMM model to a PDF of optical depth values,
where the PDF includes all the values for a single
data’s data.  Do that for a batch of days.  This
has three steps.

# 1.  Source the following function.

###################################
gmm_one_day <- function(testdate) {

histodata <- cwoparchive %>%
  filter(archivedate == testdate) %>%
  select (t_250_003_25_0_0) %>%
  collect(n=Inf) %>%
  data.frame()

histodata <- histodata %>%
filter(is.finite(t_250_003_25_0_0))

decomposet <- flexmix (t_250_003_25_0_0 ~ 1,
data=histodata, k=4, model=FLXMCnorm1(),control =
list(tolerance = 1e-15, verbose = 1, iter = 1e4))

proportions <- prior(decomposet)
comp_1 <- parameters(decomposet, component = 1)
comp_2 <- parameters(decomposet, component = 2)
if(dim(parameters(decomposet))[2] >=3) {comp_3 <-
parameters(decomposet, component = 3)} else {comp_3
<- t(data.frame(NA,NA))}
if(dim(parameters(decomposet))[2] >=4) {comp_4 <-
parameters(decomposet, component = 4)} else {comp_4
<- t(data.frame(NA,NA))}

n<- dim(histodata)[1]

gmm_date_outcomes <- data.frame(testdate,n=n,

prop1=proportions[1],prop2=proportions[2],prop3=pro
portions[3],prop4=proportions[4],

k1mean=comp_1[1],k1sd=comp_1[2],

k2mean=comp_2[1],k2sd=comp_2[2],

k3mean=comp_3[1],k3sd=comp_3[2],

k4mean=comp_4[1],k4sd=comp_4[2])
return(gmm_date_outcomes)

}

############################

# 2. Initialize a collector

gmm_date_summary <- data.frame(

testdate=char,
n=integer,
prop1 = numeric, prop2=numeric, prop3=numeric,
prop4 = numeric,
k1mean=numeric, k1sd = numeric, k2mean = numeric,
k2sd = numeric, k3mean = numeric, k3sd = numeric,
k4mean = numeric, k4sd = numeric

)


#3.  Run a batch job, such as:
```

```
gmm_date_summary <-
rbind(gmm_date_summary,gmm_one_day('2009-02-18'))
gmm_date_summary <-
rbind(gmm_date_summary,gmm_one_day('2009-02-19'))
gmm_date_summary <-
rbind(gmm_date_summary,gmm_one_day('2009-02-20'))
```

37

```
# Fit a GMM model to a PDF of optical depth values,
where the PDF includes all the values from the
database obtained in a given range of sun zenith
angles.  Do that for a batch of such ranges.  This
has three steps.  Results are deposited in a bucket
at S3, in a file that continuously grows.  The
purpose of depositing to S3 is to simplify use of
spot instances.

# Source the following function.  It deposits its
results to S3, so first tune it for an S3 bucket of
yours.

###################################

gmm_by_sza <- function(szamin,delta) {

write.csv(gmm_sza_summary,"gmm_sza_summary.csv")
aws.s3::put_object(file="gmm_sza_summary.csv",bucke
t="lpic",region="us-west-2")

szamax <- szamin+delta

histodata <- cwoparchive %>%

  filter(sunzenithangle >= szamin,
  sunzenithangle <= szamax,
  linterpreted != 998) %>%
  select (t_250_003_25_0_0) %>%
  collect(n=Inf) %>%
  data.frame()

histodata <- histodata %>%
filter(is.finite(t_250_003_25_0_0)) # It appears
illogical to separate this step from the filters
above but because of the size of the subset, this
step requires to be separated.

decomposet <- flexmix (t_250_003_25_0_0 ~ 1,
data=histodata, k=4, model=FLXMCnorm1(),control =
list(tolerance = 1e-15, verbose = 1, iter = 1e4))

proportions <- prior(decomposet)
comp_1 <- parameters(decomposet, component = 1)
comp_2 <- parameters(decomposet, component = 2)
if(dim(parameters(decomposet))[2] >=3) {comp_3 <-
parameters(decomposet, component = 3)} else {comp_3
<- t(data.frame(NA,NA))}
if(dim(parameters(decomposet))[2] >=4) {comp_4 <-
parameters(decomposet, component = 4)} else {comp_4
<- t(data.frame(NA,NA))}

n<- dim(histodata)[1]

gmm_sza_outcomes <- data.frame(szamin,szamax,n=n,

prop1=proportions[1],prop2=proportions[2],prop3=pro
portions[3],prop4=proportions[4],

k1mean=comp_1[1],k1sd=comp_1[2],

k2mean=comp_2[1],k2sd=comp_2[2],
k3mean=comp_3[1],k3sd=comp_3[2],

k4mean=comp_4[1],k4sd=comp_4[2])

return(gmm_sza_outcomes)

}

##################################

# 2.  Initialize a collector

gmm_sza_summary  <- data.frame(

szamin=integer, szamin=numeric,
n=integer,
prop1 = numeric, prop2=numeric, prop3=numeric,
prop4 = numeric,
k1mean=numeric, k1sd = numeric, k2mean = numeric,
k2sd = numeric, k3mean = numeric, k3sd = numeric,
k4mean = numeric, k4sd = numeric

)

# 3.  Launch a batch job, such as:

gmm_sza_summary <-
rbind(gmm_sza_summary,gmm_by_sza(0,2.))
  gmm_sza_summary <-
rbind(gmm_sza_summary,gmm_by_sza(5,1.))
  gmm_sza_summary <-
rbind(gmm_sza_summary,gmm_by_sza(10,0.5))

# And So On…

# The reason for transferring results to an S3
bucket is that it saves money to use of spot
instances for this calculation, but there is a risk
of lost results if spot instances are terminated.
By uploading results continuously to S3, you can
keep the work a spot instance has completed even if
the spot instance is terminated unexpectedly.
```

SAMPLE SQL QUERIES

	`# Count rows`
	
	`select count (*) from cwoparchive;`
		879476443

	`# Count the number of very-high-L (glitches)`
	
	`select count (*) from cwoparchive where linterpreted>2000;`
		241614

	`# Delete all observations from a specific day`
	
	`delete from cwoparchive where archivedate='2020-06-16'`	

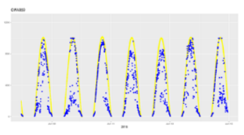
	`# To turn off a rogue process, first find the ID of a running process`
	
	`select pid,`
	`trim(user_name),`
	`starttime,`
	`substring(query,1,20)`
	`from stv_recents`
	`where`
	`status='Running';`	

	`# then stop the process whose index nnnn is returned by the above`
	
	`cancel nnnn`	

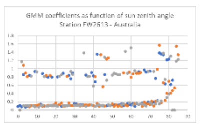
	`# Drop a column – in this example, the column lcharerr`
	
	`alter table public.cwoparchive drop column lcharerr;`	

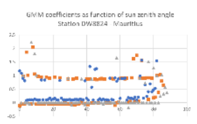
Annex C
Data idiosyncrasies

Table 1: Idiosyncrasies in the Solar Radiation Archive and Database

APRS reporting anomalies and idiosyncrasies	**Frequency**	**Means of Detection**	**Remedy/ Systematic effect of uncorrected error**
Reported station coordinates may have typos or be out of APRS format.	This occurs occasionally in the early years or two of station stabilization. Thereafter errors are rare, but mis-formatting does affect some long-term, valuable stations.	The Gladstone quality control program flags location errors implicitly, since a wrong location will yield errors in all the met data. Alternatively, large persistent errors are apparent in exploratory data analysis, e.g., via comparison of real to modeled data.  As for occasional errors (e.g., moving day) Annex B Table 1 provides code to detect changes in reported latitude and longitude, so that occasional errors can be picked up. The correct value is in general associated with a populated area. When several values are provided , e.g. by a station with several	This can be easily remedied via re-parsing, if the issue is flagged and a better value is known. If it is not flagged, then in general it can be reasonably expected that the errors will be random, except that most stations are in the North and so sign errors systematically send stations to the South.

		observing programs in parallel, distinction of errors from correct values can sometimes by identified by inspection, such as sign errors in lat/lon, or by observing the values used the user finds and corrects it.	
Station times may be out of format	Small clock errors do not appear to occur at all, since software picks up date-time from global clocks. Format errors do occur, albeit rarely.	If the date-time data does not appear similar to the APRS format, the observation is not parsed. If format looks right but the date provided is wrong, this is discovered during parsing, when dates supplied are crosschecked with the title of the daily data file. When these conflict, the parser does not analyze the data. If the format and date are right but the time provided is wrong, such as for a time zone error, the error would be apparent in exploratory data analysis.	These can be remedied via re-parsing. If a format or date error is not remedied, the data is simply absent from the database. If a time zone is mis-reported, and goes uncorrected, this contributes a serious systematic error to the single station, but not to the database so far as we can see.
Sensors tilted	A few stations deploy sensors tilted toward the southern horizon, usually on a temporary basis. A survey in 2016 detected on the	Large tilts can be identified in exploratory data analysis. Small tilts are difficult to distinguish from	If the tilt is detected, the data can be reparsed. (The package insol provides means to do so.)

	order of a dozen such stations. Smaller sensor tilts, on the order of a degree or two, are not yet measured; they are expected to be ubiquitous but random.	other potential anomalies. If the spike in optical depth at around five to ten percent turns out to be predictable, it could help to distinguish these errors.	If it is not corrected, the effect of uncorrected small tilts is expected be systematic at a given station, creating spurious diurnal and seasonal effects, but to be random over the database as a whole.
Shading of the sensor by vegetation, infrastructure	Occurs often.	Easily detected in exploratory data analysis	The data can be masked to remove these anomalies. If the data is not masked, then the effect must be considered, and analyses that are robust must be selected. Of note, not all analyses are strongly affected. For example, steady shading will affect the mean value of tau but not its statistical moments. To take another example, shading will tend to drive observations of tau to higher values; thus a lesser effect is expected on phenomena at the low-tau end of the spectrum
Cosine error	Per spec sheet for the widely used Davis sensor, expect cosine error at large sun zenith angles in a substantial fraction of the network. The cosine response for TEPT5700 sensor that is recently	It appears in the GMM decomposition of the PDF. See the two which follow. The first uses the Davis sensor. 	For sun zenith angles above 75 degrees expect data to be significantly darkened by this effect, increasingly so. Don't overinterpret data in that range.

	increasingly in use does not seem to report cosine response;	Sensor not identified for the station below. 	See figures in cell to the left; these suggest the magnitude of the error (presented as optical depth).
	Unknown. The network is a mix of new and old stations. Tarnishing should be considered as a possible effect on data at the older stations using SiO_2 sensors.	A projected analysis, not yet undertaken, is a study of all the station data from Boulder, CO, for one year, such as 2020. Values of optical depth would be plotted against year the station entered the network, and a functional relationship would be determined.	A remedy is not yet calculated; however, once the effect is measured a fix can be applied to data as a function of station age. If the data is not remedied, tarnishing is expected to contribute shading, more so at old stations The spike apparent in the PDF of optical depth values has a width that must comprise all random errors as well as the natural variation in that spike. That peak accordingly sets an upper limit on this error. This estimate is not yet made.
Software error that folds over L>1000 by subtracting 1000	No count has been made, but a rough estimate is that thousands of stations have some segment affected by this issue. The respective manufacturers are undertaking correction, so the frequency of this error is declining	Stretches of data that is capped at L=1000. See code examples in lines 19 or 22 of Annex B, Table 2. Note the software manufacturer, either by reviewing Gladstone's site or by using the code in line 25 of Annex B, Table 2.	It is possible to choose only datasets that are unaffected, by examination. It is possible to choose only later data, verifying by examination. It is also possible to simply excise all data recorded when predicted L is above L=999. In that case, results would not

			reflect data obtained in the tropics at the zenith. A Bayes fix is under development to repair data from affected stations. This would use segments of unaffected station data to learn a system for identifying errors in affected data. This was presented in Annex 2 to the Ten Year Report and it worked well, but it is not yet generalized. If the fix is not made, then the highest L observations are recorded as deeply shaded when they are not.
Software error that reports all values of L>998 as 998	No count has been made, but a rough estimate is that several hundred stations have data segments affected by this software issue	Look for stretches of data that is capped at L=998. See code examples in lines 19 or 22 of Annex B, Table 2. Or look at the software manufacturer, either by reviewing Gladstone's site or by using the code in line 25 of Annex B, Table 2.	Choose unaffected data segments. A quick fix is to delete all observations where L=998. This removes erroneous observations but does not replace the missing high-L data from affected stations. It is systematically not present. If no fix is undertaken, then data will appear more shaded than it is, and this effect will be latitude and climate-dependent.

Albedo is not reported.	This is not an error.	Data is parsed on the basis of an assumption about albedo. As for spotting cases where that assumption is wrong, if the spike in optical depth at around five to ten percent turns out to be predictable, it could help to highlight this error.	Data can be re-parsed on the basis of correct albedo if known. Several alternative models with varying values of albedo are provided already in the dataset. It appears the effect of mis-estimated albedo on the optical depth calculation can be as much as 0.1. For a single station this is systematic. For the database as a whole it is not systematic provided the albedo value used is a reasonable average.
Atmospheric visibility is not known	Same	This does not affect measured L values. It does affect estimates of optical depth. The database provides estimates of optical depth based on a range of assumptions. if the spike in optical depth at around five to ten percent turns out to be predictable, it could help to establish which value is correct.	This is not an error in L nor an uncertainty in L. If the value is known, data can be re-parsed, or can be parsed without this correction if a correct value cannot be confidently assumed. To estimate the uncertainty in tau that results from this uncertainty, it may be useful to consider the range of models supplied in the database, which include models with varying values of assumed atmospheric visibility.

Ozone is not reported	Same	Same	This is not an error in L nor an uncertainty in L. To estimate the uncertainty in tau that results from this uncertainty, run insol using a range of assumptions, or use the range of models supplied in the database, which include models with varying values of assumed ozone.
Observing programs vary	Very often	Apparent in exploratory data analysis.	Choose analyses robust to variations in the selected datasets
Stations at times have reporting gaps	Very often	Apparent in exploratory data analysis	Choose analyses robust to variations in the selected datasets
Station name changes or nonstandard use of the station name field	Occasionally happens	Annex 2, Table B provides ways to estimate the length of data segments using constant lat/lon	This is not an error and it does not contribute error. But where persistent stations are not recognizable from station names, statistical methods of data correction are constrained, since all segments appear short.
The network is geographically uneven	NA	It is possible to map data availabity. See Line 4, and to map availability on a selected date, replace the value of "displaydate"	Choose analyses robust to the data distribution and the selections you can make. Choose analyses robust to uneven data distribution. If this is not addressed, statistical analyses will over-represent some areas.

Archiving interval not known	This interval is never known except at the NOAA stations.	Exploratory data analysis may be able to pin this down – Annex 2 Table 1 proposes use of the number of reports pre-dawn and post-sunset as a means to this end. However, these are not elaborated.	This is an unknown. Choose analyses that are robust to this uncertainty. If this is not addressed, glinting will be reported differently from station to station, an artifact with a systematic aspect due to the regional differences in software.
Sensors / software not known	A small share of stations do not report their hardware or their software	There is code in Annex B, Table 2, by which to assemble all comments provided in the tech field, ever, by a given station. This usually produces information about the hardware and software	It is possible to select only stations for which hardware and/or software are known. If that is not done, then analyses should be chosen that are robust to the constraints of hardware. For example, many sensors are not calibrated much above L=1000; data loggers have a range of cutoff values of L.
Observations are quantized, L=1, 2, 3 and so on	APRS protocol requires this for all observations in the database	All observations observe this protocol	Watch out for artifacts when analyzing low L or small differences
Instrumental uncertainty		Of note, if the mean value of the histogram Spike does turn out to be a steady predictable function of known variables, then it can be used to help with estimate of instrumental uncertainty.	See spec sheets and take into account expected instrumental uncertainty Or if no correction is made, then select analyses for which random error does not matter
Calibration drift	All stations	Station age is some indicator of drift, but	Select stations where hardware is named,

		it is not reliable, because sometimes new names are given to existing stations. Where hardware is named either at the Gladstone site or in the text field (in which case it can be extracted using code in line 25 of Table 1 above), then spec sheets can estimate drift If the spike value turns out to be steady, it could be a means to identify drift.	and apply that estimate of drift. Use only the younger stations, which must be in most cases truly new because the network as a whole is growing, and apply some reasonable correction for cases that are not new. If no correction is made, then select analyses for which growing uncertainty that is random in nature does not matter.