```
#   Data merging script for the Atmopsheric Tomography Experiment (ATom)
#   merge files are made at 1s and 10s (aggregated) time resoultion, plus files for sampling intervals for
each of WAS, TOGA, SAGA, and MEDUSA
#   The time base is from the MMS instrument
#
#   usage:  source("../RPROGS/merge.atom_annotated.r")
#   to be run from folder:  [base.dir]/YYYYMMDD      where YYYYMMDD denotes the flight of interest e.g.
20171027
#
#  ATom flights should be represented in a table "flts" in the base directory
##  the base dir should have subfolders:
#      RPROGS  -- contain this script and all ancillary scripts
#      MERGE   -- the merge files will be moved here
#      YYYYMMDD folder for each flight, e.g. 20160822.  Here they are:
 #          20160808         20170111         20170210         20170926         20171013         20171027
           20180429         20180514
 # 20160712      20160812         20170124         20170213         20170928         20171014
           20171215         20180501         20180517
 # 20160726      20160815         20170126         20170215         20171001         20171017
           20171223         20180503         20180518
 # 20160729      20160817         20170129         20170218         20171004         20171019
           20180410         20180505         20180519
 # 20160801      20160820         20170201         20170219         20171006         20171020
           20180413         20180506         20180521
 # 20160803      20160822         20170203         20170221         20171008         20171023
           20180424         20180509
 # 20160806      20160823         20170205         20170914         20171011         20171025
           20180427         20180512
#  These folders contain all icartt data files (e.g. NOAA-Picarro-CO2-CH4-CO_DC8_20160808_R4.ict )
#  Each should also contain a set of files that specify the current revision number for each merge file
type, as follows:
#           rev_curr_TOGA.txt rev_curr_MED.txt rev_curr_SAGA.AERO.txt rev_curr_WAS.txt
rev_curr_PFP.txt rev_curr.txt
#           This information is needed to ensure that updates increment the revision number by 1, and it is
needed for the icartt
#           header of the merge file.  If any of these should be missing, the merge file for that type will be
assigned R1.
#
#  The files Dists1.txt and Dists10.txt give along track ground distances between data points. If missing,
they will be generated (slow).
#
#   File layout:
#
#                    | [prof.YYYYMMDD.txt]
#                    | [rev_curr*txt]
#          |20160726     | icartt files for this flight
#          |20160729  ----| [Dists1.txt]
#base.dir ---|...          | [Dists10.txt]
```

```
#          |
#          |RPROGS    --  | (all requred R scripts go here)
#          |MERGE     --  | (merge files and merge RData objects will be put here
#          |flts (plain file with flight dates, RF number, ATom number)
#          | [other plan files in base.dir]:   L.PUBLIC ; X.OK ; keep.X.txt ; N2O_Offsets.txt ;
HRAMS_NewVariableNames_ATOm1vs4.csv
#
#    Format for plain ascii file "flts" :
#     RF atom flt
#     1 1 20160729
#     2 1 20160801
#     3 1 20160803
#     4 1 20160806
#                format for prof.[YYYYMMDD].txt
#                profile.no pr.start pr.end pr.z.start pr.z.end
#                1 52220 53959 840 9655
#                2 72921 74541 9381 904
#                3 74637 75304 925 5627
#                  <start or line numbers in MMS for each profile>  (if omitted, this merge will not designate
profile numbers)
#
#    Required scripts in RPROGS are shown in the source( ) function calls below, plus script
"make.CO.X_CO2.X.r" (run later)
#     .....
##
base.dir="/Users/scwofsy/B/DATA/ATOM_TEST/"
lpublic = scan(paste(base.dir,"L.PUBLIC",sep=""),what=logical())  #   publilc merge files made if T,
restricted if F (should always be T currently)
l.X=scan(paste(base.dir,"X.OK",sep=""),what=logical()) ## T = include CO.X and CO2.X in the merges; F =
only make sep files;  set always to T
keep.X=scan(paste(base.dir,"keep.X.txt",sep=""),what=character()) ## other parameters to put in the
CO.X and CO2.X distinct objects

print(paste("base.dir"=base.dir," lpublic=",lpublic) )
source(paste(sep="",base.dir,"RPROGS/read.1001_icartt_new.r") )  # script to read the icartt files
(format 1001 is the only one)
source(paste(sep="",base.dir,"RPROGS/merge.medusa_atom.r") )  # script to merge 1s data with
medusa, including averaging kernal
source(paste(sep="",base.dir,"RPROGS/repeated.segments.r"))   # script that detects and counts
sequential repeats within a vectur
source(paste(sep="",base.dir,"RPROGS/make.cloud_ind.r") )     # script to decode the cloud indicator
library(stringr)
require(sp)      # package needed only if the along track ground distances are being computed
#
# Directory and flight information by parsing the folder name for the working directory
pdir=unlist( str_split(getwd(),"/") )
flt=as.numeric(pdir[length(pdir)] )
    YYYY=substring(as.character(flt),1,4)
```

```
    MM=substring(as.character(flt),5,6)
    DD=substring(as.character(flt),7,8)
#filenames=list.files(pattern="ict")
flts=read.table("../flts",header=T)
#     Assign an ATom deployment number ( 1, 2, 3, or 4)
A.no= flts [flts,"flt"]==as.character(flt) ,"atom"]
#     Assign the number for the offset between PFP and QCLS N2O (no QCLS N2O for ATom 1)
if(A.no>1) n2o.offset=n2o.offsets[n2o.offsets,"AT.no"]==A.no,"offset"]
filenames=system("ls *.ict",intern=T)
#
##it is inappropriate to merge the 60s file in the basic merge procedure
## remove this file if it is present
if(is.finite(match("AMS-60s",substring(filenames,1,7)))){
   print("Deleting AMS-60s")
   system("rm AMS-60s*")
   filenames=list.files(pattern="ict") ## 60s deleted
 }
if(is.finite(match("GEOS5-2D",substring(filenames,1,8)))){
   print("Deleting GEOS5-2D")
   system("rm GEOS5-2D*")
   filenames=list.files(pattern="ict") ## 60s deleted
 }

##ULOD 77777 and LLOD =-8888 must be flagged and set
n2o.offsets=read.table("../N2O_Offsets.txt",header=T)
#  Feb 2021 added offset column for N2O
## trap AMS names -- they can be inconsistent
ams.trap=read.table("../HRAMS_NewVariableNames_ATOm1vs4.csv",header=T,sep=",")

## files giving long time averages will be merged separately; plus the merged files, which never should
be merged
integ.prefix=c("MER","WAS","MED","PFP","PAN","TOGA")
#these are the 1Hz data sets; MMS is the first to be read so it is excluded along with integ.prefix
f2br.0=filenames[!substring(filenames,1,3)%in%c("MMS",integ.prefix) &
(substring(filenames,1,9)!="SAGA-AERO") & (substring(filenames,1,9)!="CAFS-FLUX")] ## files to be read
#F2br.0=matrix(unlist(str_split(f2br.0,"_")) ,ncol=4,byrow=T)
F2br.0=t(apply(matrix(f2br.0,ncol=1),1,function(x)unlist(str_split(x,"_"))[1:4]))
  idd=rev( duplicated(rev(F2br.0[,1]) ) ) ##"filenames" is in ascending order; mark as duplicated any row
with earlier revision number
  F2br=F2br.0[!idd,]
#we only merge the most recent revision
#  case where there is only one file in addition to MMS (obsolete -- used right after a mission)
if(is.null(dim(F2br)[[1]]))F2br=as.matrix(F2br,nrow=1,ncol=length(F2br),byrow=T)
  f2br=f2br.0[!idd]

#MMS file name, if there are multiple versions take the last one
n.mms=filenames[grep("MMS",substring(filenames,1,3))]
n.mms=n.mms[length(n.mms)]
```

```
print(n.mms)
#
#### now start reading the files, MMS first.
#
###  read the MMS file
##  read.1001 reads icartt formated text files and returns a list. Component "dat" has the data.
      #  The other components are used to construct header and metadata information for data from the
particular file
MMS.0=read.1001(n.mms)
# the data frame "Dum" accumulates the merged data
Dum=MMS.0$dat
LLOD_FLAG_MMS=as.numeric(unlist(str_split(system(paste("grep
LLOD_FLAG",n.mms),intern=T),":"))[2])
ULOD_FLAG_MMS=as.numeric(unlist(str_split(system(paste("grep
ULOD_FLAG",n.mms),intern=T),":"))[2])
## NB:  in these merges, LLOD and ULOD flagged data are set to MISSING (!)
Dum[Dum == LLOD_FLAG_MMS | Dum == ULOD_FLAG_MMS ] =NA
### full range of acceptable times; time vector with 1 Hz resolution
Dum.24=Dum
#                   ---- PATCH ------------------
##          The following patch handles missing data in the MMS file.  Usually there are none, but a few
may show up
if(sum(is.finite(Dum[,"G_LONG"]+Dum[,"G_LAT"]))!=0){
lna=is.finite(Dum[,"G_LONG"]+Dum[,"G_LAT"])
print(paste(sum(!lna),"NAs found in MMS"))
## interpolate missing lat/lon;  not needed after MMS revisions
a7=approx(Dum[lna,1],Dum[lna,"G_LAT"],xout=Dum[!lna,1])$y
Dum[!lna,"G_LAT"]=a7
MMS.0$dat[!lna,"G_LAT"]=a7
a8=approx(Dum[lna,1],Dum[lna,"G_LONG"],xout=Dum[!lna,1])$y
Dum[!lna,"G_LONG"]=a8
MMS.0$dat[!lna,"G_LONG"]=a8
a9=approx(Dum[lna,1],Dum[lna,"G_ALT"],xout=Dum[!lna,1])$y
Dum[!lna,"G_ALT"]=a9
MMS.0$dat[!lna,"G_ALT"]=a9
}
#              end of patch for missing MMS data
UTC=(Dum[,1])

##
## 10 s merge
##cut into 10 sec intervals
  Dum.mms=Dum   ##saved for later use
  t1=trunc(UTC/10)*10       ## An index that labels each row with the start time of a 10s interval
  bb=unique(c(t1,t1[length(t1)]+10)) ## breaks for cut () -- 10s intervals for all unique t1, including all
data
  INDEX.start=bb[as.numeric(cut(UTC,right=F,breaks=bb))]  ## INDEX for each 10s interval, labelled at
START
```

```
 INDEX.start.mms=INDEX.start   ##save for later
##  This function aggregates MMS by averaging all data into the 10s intervals labelled by bb
MMS.1=aggregate(Dum,by=list(INDEX.start),mean,na.rm=T)
#       -- PATCH --
# dateline trap function -- corrects erroneous averaging of longitude if an interval spans the dateline
   funny=function(x){
     x1=mean(x,na.rm=T) ;
     if(abs(diff(range(x,na.rm=T)))> 10 | median(abs(trunc(x)),na.rm=T) ==179){
        X=x ; X[x<0]=X[x<0]+360 ; X1=mean(X,na.rm=T) ; x1=X1 ; if(X1>180)x1=X1-360
       }
    return(x1)}
# correct longitude in MER10 for dateline trap
MMS.1[,"G_LONG"]=tapply(Dum[,"G_LONG"],INDEX.start,FUN=funny     ) ## accounts for averaging
dateline issue
#       -- END of DATELINE PATCH --
UTC_Stop=MMS.1[,1]+10
MMS=cbind(MMS.1[,1],UTC_Stop,MMS.1[,2:ncol(MMS.1)])
colnames(MMS)[1:3]=c("UTC_Start","UTC_Stop","UTC_Mean")
#
##        ------------------------------------------------|
##         Data frame Mer will accumlate the merged 10s data|
          Mer=MMS            # MMS is the first one     #|
##        ------------------------------------------------|
#
#### These will be for the 10s files -- required metdata for the MERGE files to conform to ICARTT
Special.comments="File,YYYY,MM,DD,yyyy,mm,dd,R#"
Special.comments=c(Special.comments,paste(n.mms,paste(MMS.0$date.s,collapse=","),MMS.0$REV,se
p=",") )
Params=colnames(MMS)[2:ncol(MMS)]
Units=c("s","s",MMS.0$VUNITS )
 ALLNAMES = data.frame(params=colnames(Mer),data.set=rep("MMS",ncol(Mer)) )  ### accume all
params with source file for metadata
####

##
#############################################################################################
##
## 1 s merge -- condtioning of MMS data -- follows the logic for 10 s data, above, but
#  #make intervals every 1 s
Dum=MMS.0$dat
LLOD_FLAG_MMS=as.numeric(unlist(str_split(system(paste("grep
LLOD_FLAG",n.mms),intern=T),":"))[2])
ULOD_FLAG_MMS=as.numeric(unlist(str_split(system(paste("grep
ULOD_FLAG",n.mms),intern=T),":"))[2])
Dum[Dum == LLOD_FLAG_MMS | Dum == ULOD_FLAG_MMS ] =NA
### full range of acceptable times; time vector with 1 Hz resolution
UTC=(Dum[,1])
```

```
 UTC.1s=min(trunc(UTC)):(max(trunc(UTC)))
UTC_Stop.1s=UTC.1s+1
 UTC_Start.1s=trunc(UTC)
MMS.1s=aggregate(Dum,by=list(UTC_Start.1s),mean,na.rm=T)
MMS.1.1s=merge(cbind(UTC.1s,UTC_Stop.1s),MMS.1s,by=1,all=T)

colnames(MMS.1.1s)[1:3]=c("UTC_Start","UTC_Stop","UTC_Actual")
#
##        ------------------------------------------------|
#
#### This will be for the 1s files --|
Mer.1s=MMS.1.1s                # |
# ---------------------------------|
########################################  gap fill if any in MMS
#####################################3
#   gaps can occur because sampling is not *exactly* 1Hz, so an individual 1s interval can be skipped:
interpolate
lna=is.finite(Mer.1s[,"G_LONG"]+Mer.1s[,"G_LAT"])
print(paste(sum(!lna),"1s gap found"))
if(sum(lna)>0) {
a7=approx(Mer.1s[lna,1],Mer.1s[lna,"G_LAT"],xout=Mer.1s[!lna,1])$y
Mer.1s[!lna,"G_LAT"]=a7
a8=approx(Mer.1s[lna,1],Mer.1s[lna,"G_LONG"],xout=Mer.1s[!lna,1])$y
Mer.1s[!lna,"G_LONG"]=a8
a9=approx(Mer.1s[lna,1],Mer.1s[lna,"G_ALT"],xout=Mer.1s[!lna,1])$y
Mer.1s[!lna,"G_ALT"]=a9
}
########################################  gap fill ####################################3
###  DONE MMS conditioning for Mer.1s
##     Metadata for 1s file
Special.comments.1s="File,YYYY,MM,DD,yyyy,mm,dd,R#"
Special.comments.1s=c(Special.comments.1s,paste(n.mms,paste(MMS.0$date.s,collapse=","),MMS.0$R
EV,sep=",") )
Params.1s=colnames(MMS.1.1s)[2:ncol(MMS.1.1s)]
Units.1s=c("s","s",MMS.0$VUNITS )

#
##           carry out the 1Hz meerge in a loop over the available ict files
for(i2 in 1:length(f2br) ){
if(F2br[i2,1]%in%c("SAGA-MC","SAGA-AERO","TOGA")) next       ## don't merge TOGA or SAGA to 1s,
sample interval is 35s
print(paste("1Hz merge Reading:",f2br[i2]) )
is.n2o = substring(f2br[i2],1,15) == "QCLS-CH4-CO-N2O"  ; if(is.n2o & A.no> 1 )paste("N2O offset" )
# read the file for this realization of the loop over the ict files
Dum.0 = read.1001(f2br[i2] )
Dum=Dum.0$dat
if(is.null(dim(Dum)))Dum=t(as.matrix(Dum.0$dat))  ## trap error for 1 line of  null data in the file
l.ok=Dum[,1]> UTC.1s[1] & Dum[,1]< UTC_Stop.1s[length(UTC_Stop.1s)]
```

```
Dum=Dum[l.ok,]
if(is.null(dim(Dum)))Dum=t(as.matrix(Dum.0$dat))  ## trap error for 1 line of  null data in the file
#  determine values used for LLOD and ULOD flags
LLOD_FLAG=as.numeric(unlist(str_split(system(paste("grep LLOD_FLAG",f2br[i2]),intern=T),":"))[2])
ULOD_FLAG=as.numeric(unlist(str_split(system(paste("grep ULOD_FLAG",f2br[i2]),intern=T),":"))[2])
#changed 20191027  ; then changed back 20191106 per Pedro communication
Dum[Dum == LLOD_FLAG | Dum == ULOD_FLAG] =NA
#-#Dum[Dum == LLOD_FLAG] = -8888 ;  Dum[Dum == ULOD_FLAG] = 77777   #-# 2019-10-27
if(is.null(dim(Dum)))Dum=t(as.matrix(Dum.0$dat))  ## trap error for 1 line of  null data in the file
  t1=trunc(Dum[,1])
if(length(t1)<5) next   ## traps case with no entries
if(substring(F2br[i2,1],1,3)=="CAFS"){
  t1=sum(is.finite(Dum[,4]))
if(length(t1)<100) next   ## traps case with no entries
}
  Dum.1=aggregate(Dum,by=list(t1),mean,na.rm=T)
#  aggregates because there can be two readings in a particular 1s interval
  colnames(Dum.1)[2]=paste("UTC_Start",F2br[i2,1],sep=".")
if(F2br[i2,1] ==
"DOSE")colnames(Dum.1)[2:ncol(Dum.1)]=paste(colnames(Dum.1)[2:ncol(Dum.1)],"DOSE",sep=".")
##############    NAMES    NAMES    NAMES ##############    NAMES    NAMES    NAMES
#              The hard-coded traps below account for actual instances of anomalies in the ict files
## UCATS
if(F2br[i2,1]=="UCATS-GC"){
##check if names changed, if not, append _UCATS
print("UCATS name check")
N7=str_split(colnames(Dum.1),pattern="_")
##case nothing appended
if(is.na(N7[[4]][2])) { colnames(Dum.1)[2:ncol(Dum.1)] =
paste(colnames(Dum.1)[2:ncol(Dum.1)],"UCATS",sep="_")
   } else {
if(N7[[4]][2]!="UCATS"){                  #do nothing if it is already _UCATS
cx5=matrix(unlist(N7),ncol=2,byrow=T)[,1]    ## first element of each list
colnames(Dum.1)[2:ncol(Dum.1)] = paste(cx5,"UCATS",sep="_")
                  }
}
}

#
####  the following TRAPS for non-conforming names, a few errors in particular files, etc
#
## GMI   avoid dup variables
if(substring(F2br[i2,1],1,3)=="GMI" ){
### append _GMI to names
print("GMI file data found!")
print("**********************************************************************************
******")
Dummass.2=Dum.1
```

```
ncc=nchar(colnames(Dum.1))
if(substring(colnames(Dum.1)[6],ncc[6]-2)!="GMI"){
##case T, nothing appended
colnames(Dum.1)[2:ncol(Dum.1)] = paste(colnames(Dum.1)[2:ncol(Dum.1)],"GMI",sep="_")
}

}

## AMSSD
l.AMSSD=F
if(substring(F2br[i2,1],1,5)=="AMSSD" ){
###1st 12 columns only
l.AMSSD=T
print("AMSSD check")
Dum.1=Dum.1[,1:13]
## ams names trap
uu=match(colnames(Dum.1),ams.trap[,2])
colnames(Dum.1)[which(is.finite(uu))]=ams.trap[uu[is.finite(uu)],1]
Dum.0$VUNITS=Dum.0$VUNITS[1:11]
}

## AMS
if(substring(F2br[i2,1],1,3)=="AMS" ){
###1st 12 columns only
print("AMS names trap")
## ams names trap
uu=match(colnames(Dum.1),ams.trap[,2])
colnames(Dum.1)[which(is.finite(uu))]=ams.trap[uu[is.finite(uu)],1]
}

##############    NAMES    NAMES    NAMES ##############    NAMES    NAMES    NAMES
### n2o.offset
if(A.no>1 & is.n2o){
N2O_QCLS_adj = Dum.1[,"N2O_QCLS"]-n2o.offset
new.colname=c(colnames(Dum.1),"N2O_QCLS_adj")
Dum.1=cbind(Dum.1,N2O_QCLS_adj) ; colnames(Dum.1)=new.colname
Dum.0$VUNITS = c(Dum.0$VUNITS,"ppb")
}
#    end n2o.offset
###############       END of TRAPS for 1s merge

Mer.new=merge(Mer.1s,Dum.1,by=1,all=T)
Mer.1s=Mer.new
######
Special.comments.1s=c(Special.comments.1s,paste(f2br[i2],paste(Dum.0$date.s,collapse=","),Dum.0$RE
V,sep=",") )
Params.1s=c(Params.1s, colnames(Dum.1)[2:ncol(Dum.1)] )
Units.1s=c(Units.1s,"s",Dum.0$VUNITS )
```

```
#
}
##   make combinded data sets for CO and CO2, 1s data;  This script, when sources, creates the Haverd-NOAA merged data for CO and CO2 (no gaps)
source("../RPROGS/make.CO.X_CO2.X.r")
if(is.finite(match("CO_QCLS",colnames(Mer.1s))+match("CO_NOAA",colnames(Mer.1s)))){
print("Doing CO.X")
res.CO.X=make.co.x()
if(!is.null(res.CO.X)) {
if(I.X){
Mer.1s=cbind(Mer.1s,res.CO.X[,c("CO.X","Flag.CO.X")])
Units.1s = c(Units.1s,"ppb","NONE")
Params.1s=c(Params.1s,"CO.X","Flag.CO.X")
Special.comments.1s=c(Special.comments.1s,"CO_QCLS gap filled with smoothed CO_NOAA (Level3 data product)","Flag: CO_QCL=0, CO_NOAA=1")
 } #else {
keeper=keep.X[is.finite(match(keep.X,colnames(Mer)))]
#Dum.co=cbind(Mer.1s[,keeper],res.CO.X[,c("CO.X","Flag.CO.X")])
Dum.co=cbind(Mer.1s[,keeper],res.CO.X)
save(Dum.co,file=paste("CO.X",flt,"RData",sep="."))
 #}
#png(paste("Fig.",flt,"_CO.X.comp.png",sep=""),width=1200,height=1000,pointsize=30)
##  make figures for the combined data sets
##  Combining for CO first
pdf(paste("Fig.",flt,"_CO.X.comp.pdf",sep=""),width=10,height=8.5,pointsize=12)
par(mfrow=c(2,1),mar=c(2,4,.5,2))
plot(res.CO.X[,"UTC_Start"],res.CO.X[,"CO.X"],type="l",col="orange",lwd=3,xlab="")
points(res.CO.X[,"UTC_Start"],res.CO.X[,"CO_QCLS"],col="blue",pch=16,cex=.3)
title(paste(flt,"CO_NOAA and CO.X, 1s data"))
legend("top",legend=c("CO_QCLS","CO.X"),lty=1,lwd=3,col=c("blue","orange"),text.col=c("blue","orange"),text.font=2)
par(new=T) ; plot(Mer.1s[,"UTC_Start"],Mer.1s[,"G_ALT"],type="l",col="grey",axes=F,xlab="",ylab="")
axis(4,col="grey",col.axis="grey",col.lab="grey",font=2)

plot(res.CO.X[,"UTC_Start"],res.CO.X[,"CO.del"],type="l",ylab="Offset: QCL - Picarro",xlab="")
par(new=T) ; plot(Mer.1s[,"UTC_Start"],Mer.1s[,"G_ALT"],type="l",col="grey",axes=F,xlab="",ylab="")
axis(4,col="grey",col.axis="grey",col.lab="grey",font=2)

plot(res.CO.X[,"UTC_Start"],res.CO.X[,"CO_QCLS"]-res.CO.X[,"CO.X"],type="l",ylab="CO_QCLS - CO.X",xlab="")
par(new=T) ; plot(Mer.1s[,"UTC_Start"],Mer.1s[,"G_ALT"],type="l",col="grey",axes=F,xlab="",ylab="")
axis(4,col="grey",col.axis="grey",col.lab="grey",font=2)
dev.off()
}
 }
#####
if(is.finite(match("CO2_QCLS",colnames(Mer.1s))+match("CO2_NOAA",colnames(Mer.1s)))){
print("Doing CO2.X")
```

```
res.CO2.X=make.co2.x()
if(!is.null(res.CO2.X)) {
print(head(res.CO2.X))
print("Here")
 if(l.X){
Mer.1s=cbind(Mer.1s,res.CO2.X[,c("CO2.X","Flag.CO2.X")])
Units.1s = c(Units.1s,"ppm","NONE")
Params.1s=c(Params.1s,"CO2.X","Flag.CO2.X")
Special.comments.1s=c(Special.comments.1s,"CO2_NOAA gap filled with CO2_QCLS (level 3 data
product)","Flag: CO2_NOAA=0, CO2_QCLS=1")
 } # else {
keeper=keep.X[is.finite(match(keep.X,colnames(Mer)))]
#Dum.co2=cbind(Mer.1s[,keeper],res.CO2.X[,c("CO2.X","Flag.CO2.X")])
Dum.co2=cbind(Mer.1s[,keeper],res.CO2.X)
save(Dum.co2,file=paste("CO2.X",flt,"RData",sep="."))
 #}

#png(paste("Fig.",flt,"_CO2.X.comp.png",sep=""),width=1200,height=1000,pointsize=30)
pdf(paste("Fig.",flt,"_CO2.X.comp.pdf",sep=""),width=10,height=8.5,pointsize=12)
par(mfrow=c(2,1),mar=c(2,4,.5,2))
plot(res.CO2.X["UTC_Start"],res.CO2.X["CO2.X"],type="l",col="orange",lwd=3,xlab="")
points(res.CO2.X["UTC_Start"],res.CO2.X["CO2_NOAA"],col="blue",pch=16,cex=.3)
title(paste(flt,"CO2_NOAA and CO2.X, 1s data"))
legend("top",legend=c("CO2_NOAA","CO2.X"),lty=1,lwd=3,col=c("blue","orange"),text.col=c("blue","ora
nge"),text.font=2)
par(new=T) ; plot(Mer.1s["UTC_Start"],Mer.1s["G_ALT"],type="l",col="grey",axes=F,xlab="",ylab="")
axis(4,col="grey",col.axis="grey",col.lab="grey",font=2)

plot(res.CO2.X["UTC_Start"],res.CO2.X["CO2.del"],type="l",ylab="Offset: QCL - Picarro",xlab="")
par(new=T) ; plot(Mer.1s["UTC_Start"],Mer.1s["G_ALT"],type="l",col="grey",axes=F,xlab="",ylab="")
axis(4,col="grey",col.axis="grey",col.lab="grey",font=2)


plot(res.CO2.X["UTC_Start"],res.CO2.X["CO2.Xn"],type="l",ylab="Picarro and QCLS",xlab="")
points(res.CO2.X[,1],res.CO2.X["CO2_NOAA"],col="red",type="p",pch=16,cex=.2)
legend("topleft",legend=c("QCLS w/bias
cor","CO2_NOAA"),col=c("black","red"),lty=1,cex=.6,text.font=2)
par(new=T) ; plot(Mer.1s["UTC_Start"],Mer.1s["G_ALT"],type="l",col="grey",axes=F,xlab="",ylab="")
axis(4,col="grey",col.axis="grey",col.lab="grey",font=2)
dev.off()

}
}

##  does not do this if(!file.exists(paste("prof",flt,"txt",sep=".")) ){print("Locate profiles") ;
source("../RPROGS/make.profiles.r")}
######
##add profiles if they have been determined
```

```
if(file.exists(paste("prof",flt,"txt",sep=".")) ) {
prof.nums=read.table(paste("prof",flt,"txt",sep="."),header=T)
prof.no=rep(0,nrow(Mer.1s))
for(k in 1:nrow(Mer.1s)){ prof.no[ Mer.1s[,1]>prof.nums[k,"pr.start"]&Mer.1s[,1]<prof.nums[k,"pr.end"]
] = k}
prof.no[ prof.no == 0 & Mer.1s[,"G_ALT"] ] = -1  ## 0, level at alt, -1, level at bottom
Mer.1s=cbind(Mer.1s,prof.no)
Units.1s = c(Units.1s,"NONE")
Params.1s=c(Params.1s,"prof.no")
Special.comments.1s=c(Special.comments.1s,"Profile Number (>0) ; high level leg = 0 ; low level leg = -
1")
}
#### distances will be added later
#                    -----  end 1 Hz merge ----------------------------------------------
##############################################################################################
#############
####                    START 10s merge                  ################################
####                    Follows 1s merge but time intervals are 10s   ####################
##############################################################################################
#############
#  for convenience:
## note the file types and parameters for 10s include TOGA and SAGA with 30s sampling times
###    USE midpoints for TOGA and SAGA
#                     Loop over the file types
for(i2 in 1:length(f2br) ){
if(F2br[i2,1]%in%c("TOGA")) next
print(paste("Reading:",f2br[i2]) )
is.n2o = substring(f2br[i2],1,15) == "QCLS-CH4-CO-N2O"  ; if(is.n2o & A.no> 1 )paste("N2O offset" )
Dum.0 = read.1001(f2br[i2] )
Dum=Dum.0$dat
if(is.null(dim(Dum)))Dum=t(as.matrix(Dum.0$dat))  ## trap error for 1 line of  null data in the file
#              use the time stamp for start of the interval, except for averaging measurements
if(! F2br[i2,1]%in%c("TOGA","DOSE","SAGA-MC","SAGA-AERO")) { j.UTC = 1 } else {j.UTC =3 }  ## Mid
UTC (various names)
l.ok=Dum[,j.UTC ]> UTC[1] & Dum[,j.UTC ]< UTC[length(UTC)]
##    l.ok selects for measurements within the range of "UTC" which comes from MMS
Dum=Dum[l.ok,]
if(is.null(dim(Dum)))Dum=t(as.matrix(Dum.0$dat))  ## trap error for 1 line of  null data in the file
## handle ULOD and LLOD -- set to NA
LLOD_FLAG=as.numeric(unlist(str_split(system(paste("grep LLOD_FLAG",f2br[i2]),intern=T),":"))[2])
ULOD_FLAG=as.numeric(unlist(str_split(system(paste("grep ULOD_FLAG",f2br[i2]),intern=T),":"))[2])
#changed 20191027  change back 20191106
Dum[Dum == LLOD_FLAG | Dum == ULOD_FLAG] =NA
#Dum[Dum == LLOD_FLAG] = -8888 ;  Dum[Dum == ULOD_FLAG] = 77777   #-# 2019-10-27
if(is.null(dim(Dum)))Dum=t(as.matrix(Dum.0$dat))  ## trap error for 1 line of  null data in the file
   if(substring(f2br[i2],1,4) == "DOSE"){
      Dum[,j.UTC ]=Dum[,"Mid_UTC"]  ## a kluge, treating the midpoint as the startpoint for simple
merging
```

```
      }
  t1=trunc(Dum[,j.UTC ]/10)*10
if(length(t1)<5) next
if(substring(F2br[i2,1],1,3)=="CAFS"){
  t1=sum(is.finite(Dum[,4]))
if(length(t1)<100) next   ## traps case with no entries
}
     if(! F2br[i2,1]%in%c("TOGA","DOSE","SAGA-MC","SAGA-AERO")) {          ##do not do the aggregate
step for these
  bb=unique(c(t1,t1[length(t1)]+10)) ## breaks for cut ()
  INDEX.start=bb[as.numeric(cut(Dum[,j.UTC ],right=F,breaks=bb))]  ## INDEX for each 10s interval,
labelled at START
### this step aggregates to INDEX.start ******************
  Dum.1=aggregate(Dum,by=list(INDEX.start),mean,na.rm=T)
  colnames(Dum.1)[2]=paste("UTC_Start",F2br[i2,1],sep=".")
###    composite of cloud indicator
if(is.finite(match("cloudindicator",colnames(Dum.1)))){
cloud.x=tapply(Dum[,"cloudindicator"],INDEX.start,cloud.index)
Dum.1[,"cloudindicator"] = cloud.x
}
    } else {
t1 = round (Dum[,j.UTC],-1)   ##note rounding here, giving the max overalp with the interval it will round
to
#Dum.1 = Dum
#Dum.1[,1]=t1
Dum.1 = cbind(t1,Dum)   ## t1 replaces "Group.1"
  colnames(Dum.1)[2]=paste("UTC_Start",F2br[i2,1],sep=".")
   }

# -----------   NAMES    NAMES    NAMES    NAMES `
#### Name alterations
if(F2br[i2,1] ==
"DOSE")colnames(Dum.1)[2:ncol(Dum.1)]=paste(colnames(Dum.1)[2:ncol(Dum.1)],"DOSE",sep=".")
##
## UCATS
if(F2br[i2,1]=="UCATS-GC"){
print("UCATS name check, 1s")
##check if names changed, if not, append _UCATS
N7=str_split(colnames(Dum.1),pattern="_")
##case nothing appended
if(is.na(N7[[4]][2])) { colnames(Dum.1)[2:ncol(Dum.1)] =
paste(colnames(Dum.1)[2:ncol(Dum.1)],"UCATS",sep="_")
    } else {
if(N7[[4]][2]!="UCATS"){                    #do nothing if it is already _UCATS
cx5=matrix(unlist(N7),ncol=2,byrow=T)[,1]    ## first element of each list
colnames(Dum.1)[2:ncol(Dum.1)] = paste(cx5,"UCATS",sep="_")
                  }
}
```

```
}


## GMI   avoid dup variables
if(substring(F2br[i2,1],1,3)=="GMI" ){
### append _GMI to names
print("GMI file data found!")
Dumass.1=Dum.1
if(substring(colnames(Dum.1)[6],ncc[6]-2)!="GMI"){
##case T, nothing appended
colnames(Dum.1)[2:ncol(Dum.1)] = paste(colnames(Dum.1)[2:ncol(Dum.1)],"GMI",sep="_")
}

}

## end name alterations
## finish the traps
## AMSSD
l.AMSSD=F
if(substring(F2br[i2,1],1,5)=="AMSSD" ){
###1st 12 columns only
l.AMSSD=T
print("AMSSD check")
Dum.1=Dum.1[,1:13]
## ams names trap
uu=match(colnames(Dum.1),ams.trap[,2])
colnames(Dum.1)[which(is.finite(uu))]=ams.trap[uu[is.finite(uu)],1]
Dum.0$VUNITS=Dum.0$VUNITS[1:11]
}

## AMS
if(substring(F2br[i2,1],1,3)=="AMS" ){
###1st 12 columns only
print("AMS names trap")
## ams names trap
uu=match(colnames(Dum.1),ams.trap[,2])
colnames(Dum.1)[which(is.finite(uu))]=ams.trap[uu[is.finite(uu)],1]
}

# ---------- NAMES NAMES     NAMES       NAMES
### n2o.offset
if(A.no>1 & is.n2o){
N2O_QCLS_adj = Dum.1[,"N2O_QCLS"]-n2o.offset
new.colname=c(colnames(Dum.1),"N2O_QCLS_adj")
Dum.1=cbind(Dum.1,N2O_QCLS_adj) ; colnames(Dum.1)=new.colname
Dum.0$VUNITS = c(Dum.0$VUNITS,"ppb")
}
#    end n2o.offset
```

```
#
Mer.new=merge(Mer,Dum.1,by=1,all=T)
ALLNAMES=rbind(ALLNAMES,
data.frame(params=colnames(Dum.1)[2:ncol(Dum.1)],data.set=rep(f2br[i2],ncol(Dum.1)-1) ) )
Mer=Mer.new
###
Special.comments=c(Special.comments,paste(f2br[i2],paste(Dum.0$date.s,collapse=","),Dum.0$REV,sep
=",") )
Params=c(Params, colnames(Dum.1)[2:ncol(Dum.1)] )
Units=c(Units,"s",Dum.0$VUNITS )
#Units=c(Units,"s",Dum.0$VUNITS[1:(ncol(Dum.1)-1)] )
#
}

#fix colnames
n.dup=which(substring(colnames(Mer),1,8)=="Stop_UTC")
Start.dup=colnames(Mer)[n.dup-1]
Stop.dup=gsub(pattern="Start",replacement="Stop",Start.dup)
colnames(Mer)[n.dup] = Stop.dup
## CO.X and CO2.X
source("../RPROGS/make.CO.X_CO2.X.r")
if(is.finite(match("CO_QCLS",colnames(Mer))+match("CO_NOAA",colnames(Mer)))){
print("CO.X 10s")
res.CO.X.10=make.co.x(Mer)
if( ! is.null(res.CO.X.10 )){

if(l.X){
Mer=cbind(Mer,res.CO.X.10[,c("CO.X","Flag.CO.X")])
Units = c(Units,"ppb","NONE")
Params=c(Params,"CO.X","Flag.CO.X")
Special.comments=c(Special.comments,"CO_QCLS gap filled with smoothed CO_NOAA (Level3 data
product)","Flag: CO_QCL=0, CO_NOAA=1")
 } #else {
keeper=keep.X[is.finite(match(keep.X,colnames(Mer)))]
#Dum.co.10=cbind(Mer[,keeper],res.CO.X.10[,c("CO.X","Flag.CO.X")])
Dum.co.10=cbind(Mer[,keeper],res.CO.X.10)
save(Dum.co.10,file=paste("CO.X.10",flt,"RData",sep="."))
 #}
}
}

if(is.finite(match("CO2_QCLS",colnames(Mer))+match("CO2_NOAA",colnames(Mer)))){
print("CO2.X 10s")
res.CO2.X.10=make.co2.x(Mer)
if( ! is.null(res.CO2.X.10 )){
if(l.X){
```

```
Mer=cbind(Mer,res.CO2.X.10[,c("CO2.X","Flag.CO2.X")])
Units = c(Units,"ppm","NONE")
Params=c(Params,"CO2.X","Flag.CO2.X")
Special.comments=c(Special.comments,"CO2_NOAA gap filled with CO2_QCLS (level 3 data
product)","Flag: CO2_NOAA=0, CO2_QCLS=1")
 } #else {
keeper=keep.X[is.finite(match(keep.X,colnames(Mer)))]
#Dum.co2.10=cbind(Mer[,keeper],res.CO2.X.10[,c("CO2.X","Flag.CO2.X")])
Dum.co2.10=cbind(Mer[,keeper],res.CO2.X.10)
save(Dum.co2.10,file=paste("CO2.X.10",flt,"RData",sep="."))
 #}
}

}

##add profiles if they have been determined
if(file.exists(paste("prof",flt,"txt",sep=".")) ) {
prof.nums=read.table(paste("prof",flt,"txt",sep="."),header=T)
prof.no=rep(0,nrow(Mer))
for(k in 1:nrow(Mer)){ prof.no[ Mer[,1]>prof.nums[k,"pr.start"]&Mer[,1]<prof.nums[k,"pr.end"] ] = k}
prof.no[ prof.no == 0 & Mer[,"G_ALT"] ] = -1  ## 0, level at alt, -1, level at bottom
Mer=cbind(Mer,prof.no)
Units = c(Units,"NONE")
Params=c(Params,"prof.no")
Special.comments=c(Special.comments,"Profile Number (>0) ; high level leg = 0 ; low level leg = -1")
ALLNAMES=rbind(ALLNAMES,c("prof.no","computed"))
}

##add distance in km since departure -- use file if it exists
if(file.exists("Dists10.txt")){Dist=scan("Dists10.txt")} else {
## if the file Dists10.txt does not exist, create it (slow) us spDistsN1( )
#
d0=0
#
for(k in 2:nrow(Dum.mms)){
d0=c(d0,spDistsN1(pts=matrix(c(Dum.mms[k,"G_LONG"],Dum.mms[k,"G_LAT"]),ncol=2,nrow=1,byrow=
T),
    pt=c(Dum.mms[k-1,"G_LONG"],Dum.mms[k-1,"G_LAT"]),longlat=T) )
}
dd=tapply(d0,INDEX.start.mms,sum)
write(d0,file="d0.all.txt")
Dist=cumsum(dd)
print("Writing Dists10.txt")
write(Dist,file="Dists10.txt")
}
Mer=cbind(Mer,Dist)
Units = c(Units,"km")
Params=c(Params,"Dist")
```

```
Special.comments=c(Special.comments,"Distance in km along the flight track")
ALLNAMES=rbind(ALLNAMES,c("Dist","computed"))

#########################################################
##add Dists to Mer.1s by interpolating in the 10s file.#
if(file.exists("Dists1.txt")){                    #
                                  #
Dist=scan("Dists1.txt")} else {                    #
d0 = scan("d0.all.txt") # file created in 10s loop     # May not have exact 1s time stamp of UTC_1s     #
#                                  #
dd=tapply(d0,UTC_Start.1s,sum)                    #
if(length(dd)>=nrow(Mer.1s)) {dd=dd[1:nrow(Mer.1s)]    #
  } else {                                  #
dd=c(dd,rep(0,nrow(Mer.1s)-length(dd)) ) }           #
Dist=cumsum(dd)                                  #
print("Writing Dists1.txt")                      #
write(Dist,file="Dists1.txt")                    #
}                                  #
                                  #
Mer.1s=cbind(Mer.1s,Dist)                        #
                                  #
Units.1s = c(Units.1s,"km")                      #
Params.1s=c(Params.1s,"Dist")                    #_____________________________#
Special.comments.1s=c(Special.comments.1s,"Distance in km along the flight track")  #
####    end of adding distance to 1s file           #---------------------------#
#########################################################
# write out allnames
write.table(ALLNAMES,row.names=F,col.names=T,quote=F,file="ALLNAMES.txt")

#          vvvvvvvvvvvvvvvvvvvvvv
##   THE 1s and 10s merges are now complete !
#          ^^^^^^^^^^^^^^^^^^^^^^
## forst write out flat files, then write out ICARTT files
### remove redundant UTC columns  1s
lUTC.1s=1:(ncol(Mer.1s) )%in%grep(pattern="UTC",x=colnames(Mer.1s))
lUTC.1s=lUTC.1s[2:ncol(Mer.1s)]
lUTC.1s[1]=F    ## this is UTC_Stop; all the others containing "UTC" will be removed
l.UTC.1s= !lUTC.1s
l.keep.1s=unlist(str_split(paste("UTC_Start",paste(Params.1s[l.UTC.1s],collapse=","),sep=","),pattern=","
))
colnames(Mer.1s)[2]=l.keep.1s[2]
#print(l.keep.1s)
lUTC=1:(ncol(Mer) )%in%grep(pattern="UTC",x=colnames(Mer))
lUTC=lUTC[2:ncol(Mer)]
lUTC[1]=F    ## this is UTC_Stop; all the others containing "UTC" will be removed
l.UTC= !lUTC
l.keep=unlist(str_split(paste("UTC_Start",paste(Params[l.UTC],collapse=","),sep=","),pattern=","))
colnames(Mer)[2]=l.keep[2] ;
```

```
##
## flat tables --
write.table(Mer.1s,file="Mer.1s.tbl",row.names=F,col.names=T,quote=F)
write.table(Mer,file="Mer.tbl",row.names=F,col.names=T,quote=F)
system("gzip -f Mer.tbl Mer.1s.tbl")
## end flat tables

## construct the 10s ICARTT file
Normal.Comments=readLines(paste(sep="",base.dir,"RPROGS/Normal_Comments.txt" ))
## deal with revision number
if(file.exists("rev_curr.txt") ) {          ## case REV is specified
REV=scan("rev_curr.txt")
N.C=Normal.Comments[1:(length(Normal.Comments) -1 ) ]
N.C.addon=paste("REVISION: R",REV+1,sep="")
if(REV>=0) { for(i in 1:(REV+1)) { N.C.addon= c( N.C.addon , paste("R",i,": updated", sep="")) } }  ##
creating next rev
Normal.Comments=c(N.C,N.C.addon)
}
##
Normal.Comments=c(Normal.Comments,paste("UTC_Start",paste(Params[l.UTC],collapse=","),sep=","))
N.Normal.Comments=length(Normal.Comments)  ##add line of parameters
NV=length(Params[(l.UTC)])
Scale.Factors=paste(rep(1,NV),collapse=",")
Missing.Values=paste(rep(-99999,NV),collapse=",")
Var.names.units=paste(Params[l.UTC],Units[l.UTC],sep=", ")
NLINES=14+NV+length(Special.comments)+(N.Normal.Comments)
Params.units=Params[l.UTC] ; names(Params.units) = Units[l.UTC]  ## note: missing UTC_Start

##write the 10 s file
write(paste(NLINES,"1001",sep=", "),file="Mer10.icartt")
write(c("Wofsy,S", "Harvard_University", "10 second merge of ATom DC-8 data", "ATom", "1,1"),
ncol=1,file="Mer10.icartt",append=T)
write(paste(YYYY, MM,DD,
format(Sys.time(),tz="GMT","%Y,%m,%d"),sep=","),file="Mer10.icartt",append=T)
write("10",file="Mer10.icartt",append=T)
write("UTC_Start, s",file="Mer10.icartt",append=T)
write(NV ,file="Mer10.icartt",append=T)
write(Scale.Factors ,file="Mer10.icartt",append=T)
write(Missing.Values ,file="Mer10.icartt",append=T)
write(Var.names.units ,file="Mer10.icartt",ncol=1,append=T)
write(length(Special.comments),file="Mer10.icartt",append=T)
write(Special.comments,file="Mer10.icartt",ncol=1,append=T)
write(N.Normal.Comments,file="Mer10.icartt",append=T)
write(Normal.Comments,file="Mer10.icartt",ncol=1,append=T)
write.table(Mer[,l.keep],file="Mer10.icartt", append=T,row.names=F,col.names=F,sep=",",na="-
99999",quote=F)
if(file.exists("rev_curr.txt") ) {          ## case REV is specified
if(lpublic){
```

```
system(paste("cp Mer10.icartt MER10_DC8_",flt,"_R",REV+1,".ict",sep=""))
} else{
system(paste("cp Mer10.icartt MER10-restricted_DC8_",flt,"_R",REV+1,".ict",sep=""))
}
}
write.table(ALLNAMES[l.keep,],row.names=F,col.names=T,quote=F,file="ALLNAMES_keep.txt")
###########   start write 1s merge file
Normal.Comments.1s=readLines(paste(base.dir,"RPROGS/Normal_Comments.txt",sep="") )
if(file.exists("rev_curr.txt") ) {          ## case REV is specified
REV=scan("rev_curr.txt")
N.C=Normal.Comments.1s[1:(length(Normal.Comments.1s) -1 ) ]
N.C.addon=paste("REVISION: R",REV+1,sep="")
if(REV>=0) { for(i in 1:(REV+1)) { N.C.addon= c( N.C.addon , paste("R",i,": updated", sep="")) } }  ##
creating next rev
Normal.Comments.1s=c(N.C,N.C.addon)
}
Normal.Comments.1s=c(Normal.Comments.1s,paste("UTC_Start",paste(Params.1s[l.UTC.1s],collapse=",
"),sep=","))
#
N.Normal.Comments.1s=length(Normal.Comments.1s)  ##add line of parameters
NV.1s=length(Params.1s[(l.UTC.1s)])
Scale.Factors.1s=paste(rep(1,NV.1s),collapse=",")
Missing.Values.1s=paste(rep(-99999,NV.1s),collapse=",")
Var.names.units.1s=paste(Params.1s[l.UTC.1s],Units.1s[l.UTC.1s],sep=", ")
NLINES.1s=14+NV.1s+length(Special.comments.1s)+(N.Normal.Comments.1s)
Params.units.1s=Params.1s[l.UTC.1s] ; names(Params.units.1s) = Units.1s[l.UTC.1s]

write(paste(NLINES.1s,"1001",sep=", "),file="Mer01.icartt")
write(c("Wofsy,S", "Harvard_University", "1 second merge of ATom DC-8 data", "ATom", "1,1"),
ncol=1,file="Mer01.icartt",append=T)
write(paste(YYYY, MM,DD,
format(Sys.time(),tz="GMT","%Y,%m,%d"),sep=","),file="Mer01.icartt",append=T)
write("1",file="Mer01.icartt",append=T)
write("UTC_Start, s",file="Mer01.icartt",append=T)
write(NV.1s ,file="Mer01.icartt",append=T)
write(Scale.Factors.1s ,file="Mer01.icartt",append=T)
write(Missing.Values.1s ,file="Mer01.icartt",append=T)
write(Var.names.units.1s ,file="Mer01.icartt",ncol=1,append=T)
write(length(Special.comments.1s),file="Mer01.icartt",append=T)
write(Special.comments.1s,file="Mer01.icartt",ncol=1,append=T)
write(N.Normal.Comments.1s,file="Mer01.icartt",append=T)
write(Normal.Comments.1s,file="Mer01.icartt",ncol=1,append=T)
#print(colnames(Mer.1s));print(l.keep.1s); m8=match(l.keep.1s,colnames(Mer.1s)); print(m8)
write.table(Mer.1s[,l.keep.1s],file="Mer01.icartt", append=T,row.names=F,col.names=F,sep=",",na="-
99999",quote=F)
if(file.exists("rev_curr.txt") ) {          ## case REV is specified
if(lpublic){
system(paste("cp Mer01.icartt MER-1HZ_DC8_",flt,"_R",REV+1,".ict",sep=""))
```

```
} else {
system(paste("cp Mer01.icartt MER-1HZ-restricted_DC8_",flt,"_R",REV+1,".ict",sep=""))}
}
###########  end write 1s merge file

#                   vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv
######################  Begin the special interval merges, start with Whole Air Sampler ("WAS")
#                   ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
##Start: merge for WAS
##########################################################################################
####
#1. merge all the other data at 1s.
#2. go to the WAS data, and use the start and end time for each row to
#make an index denoting which rows of the 1s data correspond to each flask.
#3.  use aggregate( ) to find the mean of those 1Hz rows (in one go) that correspond to
#each WAS row, and finally I cbind( ) that aggregated data frame to the WAS data.
#4. Clean up longitude and categoricals
#  a. near the dateline -- the mean location for a flask
#acquired between +179.5 and -179.5 is not 0 longitude!
#  b. the profile numberdesignator needed to be cleaned up -- each flask should be put in the most
suitable profile,
#     not some mean designator. That would hold for any other categorical data in the 1s file.
#
#NOTE:  Medusa flasks, a filling rate file provided,
# NOT DONE:  did not apply some sort of minimum coverage threshold for non-missing data
#          did not attempt to find overlap among slow-sampling systems, e.g. TOGA and WAS.
## merge for WAS
##########################################################################################
####
file.WAS=system(paste("ls -1 WAS_DC8_",flt,"_R?.ict",sep=""),intern=T)
if(length(file.WAS)>0) {    ##proceed with WAS
f.WAS=file.WAS[length(file.WAS)] ## take the last revision
Dum.0 = read.1001(f.WAS )
Dum=Dum.0$dat
#               use the time stamp for start of the interval, except for averaging measurements
l.ok=Dum[, 1 ]> Mer.1s[1,1] & Dum[,1 ]< Mer.1s[nrow(Mer.1s),1]
Dum=Dum[l.ok,]
##fix the colnames which have "_ppt" at the end.  THIS SECTION TO CHANGE IF WAS FILES CHANGE
nom.cols=matrix(unlist(str_split(colnames(Dum),pattern="_")),ncol=2,byrow=T)
colnames(Dum)[4:ncol(Dum)]=paste(nom.cols[4:ncol(Dum),1],"WAS",sep="_")
colnames(Dum)[1:3]=c("UTC_Start_WAS","UTC_Stop_WAS","UTC_Mid_WAS")
#
LLOD_FLAG=as.numeric(unlist(str_split(system(paste("grep LLOD_FLAG",f.WAS),intern=T),":"))[2])
ULOD_FLAG=as.numeric(unlist(str_split(system(paste("grep ULOD_FLAG",f.WAS),intern=T),":"))[2])
#changed 20191027
#Dum[Dum == LLOD_FLAG | Dum == ULOD_FLAG] =NA
Dum[Dum == LLOD_FLAG] = -8888 ;  Dum[Dum == ULOD_FLAG] = 77777   #-# 2019-10-27
#
```

```
was.cuts=rep(0,nrow(Mer.1s))
for(i in 1:nrow(Dum)){
  was.cuts[Mer.1s[,"UTC_Start"]>=Dum[i,"UTC_Start_WAS"] &
Mer.1s[,"UTC_Start"]<=Dum[i,"UTC_Stop_WAS"]]=i
                    }
#was.cuts=as.numeric( cut(Mer.1s[,1],breaks=c(t(Dum[,1:2])) ,labels=as.character(1:(2*nrow(Dum) - 1)))
)
#was.cuts[was.cuts%%2==0] = NA
AA=as.matrix ( aggregate(Mer.1s,by=list(was.cuts),mean,na.rm=T) )
AA=AA[AA[,1]!=0,]
prf=( tapply(Mer.1s[was.cuts!=0,"prof.no"],list(was.cuts[was.cuts!=0]),max,na.rm=T) )
AA[,"prof.no"]=prf
## correct longitude for dateline trap
X.LONG=tapply(Mer.1s[,"G_LONG"],list(was.cuts),FUN=funny     )
AA[,"G_LONG"]=X.LONG[as.numeric(names(X.LONG))>0]
## correct longitude for dateline trap
## cloudindicator mapping
if(is.finite(match("cloudindicator",colnames(Mer.1s)))) {
X.cloud=tapply(Mer.1s[was.cuts!=0,"cloudindicator"],list(was.cuts[was.cuts!=0]),FUN=cloud.index     )
AA[,"cloudindicator"]=X.cloud
}

lU=grep(pattern="UTC",colnames(AA))
lU=lU[2:length(lU)]
lkk=rep(T,ncol(AA))
lkk[lU]=F
AB=AA[,lkk]
colnames(AB)[2]="UTC_Mean_1s"
Mer.WAS = cbind(Dum[,1:3],AB[,2:ncol(AB)],Dum[,4:ncol(Dum)])
#
Special.comments.WAS=c(Special.comments.1s,paste(f.WAS
,paste(Dum.0$date.s,collapse=","),Dum.0$REV,sep=",") )
# Params.WAS=c(Params.1s, colnames(Dum)[4:ncol(Dum)] )
# Units.WAS=c(Units.1s,Dum.0$VUNITS[4:ncol(Dum)] )
Params.WAS=colnames(Mer.WAS)[2:ncol(Mer.WAS)]
Units.WAS=c("s","s","s",Units.1s[lkk[3:length(lkk)] ],Dum.0$VUNITS[3:(ncol(Dum)-1) ] )
## do the averaging for each flask


############   start write WAS merge file
Normal.Comments.WAS=readLines(paste(base.dir,"RPROGS/Normal_Comments.txt",sep="") )
if(file.exists("rev_curr_WAS.txt") ) {          ## case REV is specified
REV=scan("rev_curr_WAS.txt")
N.C=Normal.Comments.WAS[1:(length(Normal.Comments.WAS) -1 ) ]
N.C.addon=paste("REVISION: R",REV+1,sep="")
if(REV>=0) { for(i in 1:(REV+1)) { N.C.addon= c( N.C.addon , paste("R",i,": updated", sep="")) } }  ##
creating next rev
Normal.Comments.WAS=c(N.C,N.C.addon)
```

```
}
Normal.Comments.WAS=c(Normal.Comments.WAS,paste("UTC_Start",paste(Params.WAS,collapse=","),
sep=","))
N.Normal.Comments.WAS=length(Normal.Comments.WAS)  ##add line of parameters
NV.WAS=length(Params.WAS )
Scale.Factors.WAS=paste(rep(1,NV.WAS),collapse=",")
Missing.Values.WAS=paste(rep(-99999,NV.WAS),collapse=",")
Var.names.units.WAS=paste(Params.WAS,Units.WAS,sep=", ")
NLINES.WAS=14+NV.WAS+length(Special.comments.WAS)+(N.Normal.Comments.WAS)

write(paste(NLINES.WAS,"1001",sep=", "),file="MER-WAS.icartt")
write(c("Wofsy,S", "Harvard_University", "merge of 1s ATom DC-8 data with WAS flask samples",
"ATom", "1,1"),
ncol=1,file="MER-WAS.icartt",append=T)
write(paste(YYYY, MM,DD, format(Sys.time(),tz="GMT","%Y,%m,%d"),sep=","),file="MER-
WAS.icartt",append=T)
write("0",file="MER-WAS.icartt",append=T) ## interval =0 , not regular apcing
write("UTC_Start, s",file="MER-WAS.icartt",append=T)
write(NV.WAS ,file="MER-WAS.icartt",append=T)
write(Scale.Factors.WAS ,file="MER-WAS.icartt",append=T)
write(Missing.Values.WAS ,file="MER-WAS.icartt",append=T)
write(Var.names.units.WAS ,file="MER-WAS.icartt",ncol=1,append=T)
write(length(Special.comments.WAS),file="MER-WAS.icartt",append=T)
write(Special.comments.WAS,file="MER-WAS.icartt",ncol=1,append=T)
write(N.Normal.Comments.WAS,file="MER-WAS.icartt",append=T)
write(Normal.Comments.WAS,file="MER-WAS.icartt",ncol=1,append=T)
write.table(Mer.WAS,file="MER-WAS.icartt", append=T,row.names=F,col.names=F,sep=",",na="-
99999",quote=F)
#
if(file.exists("rev_curr_WAS.txt") ) {          ## case REV is specified
if(lpublic){
system(paste("cp MER-WAS.icartt MER-WAS_DC8_",flt,"_R",REV+1,".ict",sep=""))
  } else{
system(paste("cp MER-WAS.icartt MER-WAS-restricted_DC8_",flt,"_R",REV+1,".ict",sep=""))
}
}
}
##End: merge for WAS
#################################################################################################
####
#
##Start: merge for PFP
#################################################################################################
####

file.PFP=system(paste("ls -1 PFP_DC8_",flt,"_R?.ict",sep=""),intern=T)
if(length(file.PFP)>0) {    ##proceed with PFP
f.PFP=file.PFP[length(file.PFP)] ## take the last revision
```

```
Dum.0 = read.1001(f.PFP )
Dum=Dum.0$dat
#              use the time stamp for start of the interval, except for averaging measurements
l.ok=Dum[, 1 ]> Mer.1s[1,1] & Dum[,1 ]< Mer.1s[nrow(Mer.1s),1]
Dum=Dum[l.ok,]
##fix the colnames
## PFP's do not have a mid time
UTC_Mid=(Dum[,1] + Dum[,2])/2
Dum=cbind(Dum[,1:2],UTC_Mid,Dum[,3:ncol(Dum)] )
##### nom.cols=sapply(str_split(colnames(Dum),pattern="_"),function(x)return(x[1]))  ## these lines
were holdovers/
##### colnames(Dum)[4:ncol(Dum)]=paste(nom.cols[4:ncol(Dum)],"PFP",sep="_")
colnames(Dum)[1:3]=c("UTC_Start_PFP","UTC_Stop_PFP","UTC_Mid_PFP")
#
LLOD_FLAG=as.numeric(unlist(str_split(system(paste("grep LLOD_FLAG",f.PFP),intern=T),":"))[2])
ULOD_FLAG=as.numeric(unlist(str_split(system(paste("grep ULOD_FLAG",f.PFP),intern=T),":"))[2])
#changed 20191027
#-# 20191027 Dum[Dum == LLOD_FLAG | Dum == ULOD_FLAG] =NA
Dum[Dum == LLOD_FLAG] = -8888 ;  Dum[Dum == ULOD_FLAG] = 77777   #-# 2019-10-27
#
#pfp.cuts=as.numeric( cut(Mer.1s[,1],breaks=c(t(Dum[,1:2])) ,labels=as.character(1:(2*nrow(Dum) - 1)))
)
#pfp.cuts[pfp.cuts%%2==0] = NA
pfp.cuts=rep(0,nrow(Mer.1s))
for(i in 1:nrow(Dum)){
  pfp.cuts[Mer.1s[,"UTC_Start"]>=Dum[i,1] & Mer.1s[,"UTC_Start"]<=Dum[i,2] ] =i
                    }
AA=as.matrix ( aggregate(Mer.1s,by=list(pfp.cuts),mean,na.rm=T) )
AA=AA[AA[,1]!=0,]
prf=( tapply(Mer.1s[pfp.cuts!=0,"prof.no"],list(pfp.cuts[pfp.cuts!=0]),max,na.rm=T) )
AA[,"prof.no"]=prf
## correct longitude for dateline trap
X.LONG=tapply(Mer.1s[,"G_LONG"],list(pfp.cuts),FUN=funny     )
AA[,"G_LONG"]=X.LONG[as.numeric(names(X.LONG))>0]
## correct longitude for dateline trap
## cloudindicator mapping
if(is.finite(match("cloudindicator",colnames(Mer.1s)))) {
X.cloud=tapply(Mer.1s[pfp.cuts!=0,"cloudindicator"],list(pfp.cuts[pfp.cuts!=0]),FUN=cloud.index     )
AA[,"cloudindicator"]=X.cloud
}

lU=grep(pattern="UTC",colnames(AA))
lU=lU[2:length(lU)]
lkk=rep(T,ncol(AA))
lkk[lU]=F
AB=AA[,lkk]
colnames(AB)[2]="UTC_Mean_1s"
Mer.PFP = cbind(Dum[,1:3],AB[,2:ncol(AB)],Dum[,4:ncol(Dum)])
```

```
#
Special.comments.PFP=c(Special.comments.1s,paste(f.PFP
,paste(Dum.0$date.s,collapse=","),Dum.0$REV,sep=",") )
# Params.PFP=c(Params.1s, colnames(Dum)[4:ncol(Dum)] )
# Units.PFP=c(Units.1s,Dum.0$VUNITS[4:ncol(Dum)] )
Params.PFP=colnames(Mer.PFP)[2:ncol(Mer.PFP)]
Units.PFP=c("s","s","s",Units.1s[lkk[3:length(lkk)] ],Dum.0$VUNITS[3:(ncol(Dum)-1) ] )
## do the averaging for each flask


###########   start write PFP merge file
Normal.Comments.PFP=readLines(paste(base.dir,"RPROGS/Normal_Comments.txt",sep="") )
if(file.exists("rev_curr_PFP.txt") ) {          ## case REV is specified
REV=scan("rev_curr_PFP.txt")
N.C=Normal.Comments.PFP[1:(length(Normal.Comments.PFP) -1 ) ]
N.C.addon=paste("REVISION: R",REV+1,sep="")
if(REV>=0) { for(i in 1:(REV+1)) { N.C.addon= c( N.C.addon , paste("R",i,": updated", sep="")) } }  ##
creating next rev
Normal.Comments.PFP=c(N.C,N.C.addon)
}
Normal.Comments.PFP=c(Normal.Comments.PFP,paste("UTC_Start",paste(Params.PFP,collapse=","),sep
=","))
N.Normal.Comments.PFP=length(Normal.Comments.PFP)  ##add line of parameters
NV.PFP=length(Params.PFP )
Scale.Factors.PFP=paste(rep(1,NV.PFP),collapse=",")
Missing.Values.PFP=paste(rep(-99999,NV.PFP),collapse=",")
Var.names.units.PFP=paste(Params.PFP,Units.PFP,sep=", ")
NLINES.PFP=14+NV.PFP+length(Special.comments.PFP)+(N.Normal.Comments.PFP)

write(paste(NLINES.PFP,"1001",sep=", "),file="MER-PFP.icartt")
write(c("Wofsy,S", "Harvard_University", "merge of 1s ATom DC-8 data with PFP flask samples",
"ATom", "1,1"),
ncol=1,file="MER-PFP.icartt",append=T)
write(paste(YYYY, MM,DD, format(Sys.time(),tz="GMT","%Y,%m,%d"),sep=","),file="MER-
PFP.icartt",append=T)
write("0",file="MER-PFP.icartt",append=T) ## interval =0 , not regular apcing
write("UTC_Start, s",file="MER-PFP.icartt",append=T)
write(NV.PFP ,file="MER-PFP.icartt",append=T)
write(Scale.Factors.PFP ,file="MER-PFP.icartt",append=T)
write(Missing.Values.PFP ,file="MER-PFP.icartt",append=T)
write(Var.names.units.PFP ,file="MER-PFP.icartt",ncol=1,append=T)
write(length(Special.comments.PFP),file="MER-PFP.icartt",append=T)
write(Special.comments.PFP,file="MER-PFP.icartt",ncol=1,append=T)
write(N.Normal.Comments.PFP,file="MER-PFP.icartt",append=T)
write(Normal.Comments.PFP,file="MER-PFP.icartt",ncol=1,append=T)
write.table(Mer.PFP,file="MER-PFP.icartt", append=T,row.names=F,col.names=F,sep=",",na="-
99999",quote=F)
#
```

```
if(file.exists("rev_curr_PFP.txt") ) {          ## case REV is specified
if(lpublic){
system(paste("cp MER-PFP.icartt MER-PFP_DC8_",flt,"_R",REV+1,".ict",sep=""))
} else{
system(paste("cp MER-PFP.icartt MER-PFP-restricted_DC8_",flt,"_R",REV+1,".ict",sep=""))
}
}
}
##End: merge for PFP
#################################################################################################
####
# ************************
### Start Merge for SAGA-AERO
file.SAGA.AERO=system(paste("ls -1 SAGA-AERO_DC8_",flt,"_R?.ict",sep=""),intern=T)
if(length(file.SAGA.AERO)>0) {    ##proceed with SAGA-AERO
f.SAGA.AERO=file.SAGA.AERO[length(file.SAGA.AERO)] ## take the last revision
Dum.0 = read.1001(f.SAGA.AERO )
Dum=Dum.0$dat
#               use the time stamp for start of the interval, except for averaging measurements
l.ok=Dum[, 1 ]> Mer.1s[1,1] & Dum[,1 ]< Mer.1s[nrow(Mer.1s),1]
Dum=Dum[l.ok,]
##fix the colnames
## SAGA-AERO's do have a mid time
#UTC_Mid=(Dum[,1] + Dum[,2])/2
#Dum=cbind(Dum[,1:2],UTC_Mid,Dum[,3:ncol(Dum)] )
nom.cols=sapply(str_split(colnames(Dum),pattern="_"),function(x)return(x[1]))
colnames(Dum)[4:ncol(Dum)]=paste(nom.cols[4:ncol(Dum)],"SAGA.AERO",sep="_")
colnames(Dum)[1:3]=c("UTC_Start_SAGA.AERO","UTC_Stop_SAGA.AERO","UTC_Mid_SAGA.AERO")
#
LLOD_FLAG=as.numeric(unlist(str_split(system(paste("grep
LLOD_FLAG",f.SAGA.AERO),intern=T),":"))[2])
ULOD_FLAG=as.numeric(unlist(str_split(system(paste("grep
ULOD_FLAG",f.SAGA.AERO),intern=T),":"))[2])
#changed 20191027
#-# 20191027 Dum[Dum == LLOD_FLAG | Dum == ULOD_FLAG] =NA
Dum[Dum == LLOD_FLAG] = -8888 ;  Dum[Dum == ULOD_FLAG] = 77777   #-# 2019-10-27
#
bkk=c(t(Dum[,1:2]))
 brk=bkk
 brk[duplicated(bkk)]=bkk[duplicated(bkk)]+1

#sga.cuts=as.numeric( cut(Mer.1s[,1],breaks=brk ,labels=as.character(1:(length(brk) - 1))))
#sga.cuts[sga.cuts%%2==0] = NA
sga.cuts=rep(0,nrow(Mer.1s))
for(i in 1:nrow(Dum)){
  sga.cuts[Mer.1s[,"UTC_Start"]>=Dum[i,1] & Mer.1s[,"UTC_Start"]<=Dum[i,2] ] =i
                }
AA=as.matrix ( aggregate(Mer.1s,by=list(sga.cuts),mean,na.rm=T) )
```

```
AA=AA[AA[,1]!=0,]
prf=( tapply(Mer.1s[sga.cuts!=0,"prof.no"],list(sga.cuts[sga.cuts!=0]),max,na.rm=T) )
AA[,"prof.no"]=prf
## correct longitude for dateline trap
X.LONG=tapply(Mer.1s[,"G_LONG"],list(sga.cuts),FUN=funny     )
AA[,"G_LONG"]=X.LONG[as.numeric(names(X.LONG))>0]
## correct longitude for dateline trap
## cloudindicator mapping
if(is.finite(match("cloudindicator",colnames(Mer.1s)))) {
X.cloud=tapply(Mer.1s[sga.cuts!=0,"cloudindicator"],list(sga.cuts[sga.cuts!=0]),FUN=cloud.index     )
AA[,"cloudindicator"]=X.cloud
}

lU=grep(pattern="UTC",colnames(AA))
lU=lU[2:length(lU)]
lkk=rep(T,ncol(AA))
lkk[lU]=F
AB=AA[,lkk]
colnames(AB)[2]="UTC_Mean_1s"
Mer.SAGA.AERO = cbind(Dum[,1:3],AB[,2:ncol(AB)],Dum[,4:ncol(Dum)])
#
Special.comments.SAGA.AERO=c(Special.comments.1s,paste(f.SAGA.AERO
,paste(Dum.0$date.s,collapse=","),Dum.0$REV,sep=",") )
# Params.SAGA.AERO=c(Params.1s, colnames(Dum)[4:ncol(Dum)] )
# Units.SAGA.AERO=c(Units.1s,Dum.0$VUNITS[4:ncol(Dum)] )
Params.SAGA.AERO=colnames(Mer.SAGA.AERO)[2:ncol(Mer.SAGA.AERO)]
Units.SAGA.AERO=c("s","s","s",Units.1s[lkk[3:length(lkk)] ],Dum.0$VUNITS[3:(ncol(Dum)-1) ] )
## do the averaging for each flask


###########   start write SAGA-AERO merge file
Normal.Comments.SAGA.AERO=readLines(paste(base.dir,"RPROGS/Normal_Comments.txt",sep="") )
if(file.exists("rev_curr_SAGA.AERO.txt") ) {          ## case REV is specified
REV=scan("rev_curr_SAGA.AERO.txt")
N.C=Normal.Comments.SAGA.AERO[1:(length(Normal.Comments.SAGA.AERO) -1 ) ]
N.C.addon=paste("REVISION: R",REV+1,sep="")
if(REV>=0) { for(i in 1:(REV+1)) { N.C.addon= c( N.C.addon , paste("R",i,": updated", sep="")) } }  ##
creating next rev
Normal.Comments.SAGA.AERO=c(N.C,N.C.addon)
}
Normal.Comments.SAGA.AERO=c(Normal.Comments.SAGA.AERO,paste("UTC_Start",paste(Params.SAG
A.AERO,collapse=","),sep=","))
N.Normal.Comments.SAGA.AERO=length(Normal.Comments.SAGA.AERO)  ##add line of parameters
NV.SAGA.AERO=length(Params.SAGA.AERO )
Scale.Factors.SAGA.AERO=paste(rep(1,NV.SAGA.AERO),collapse=",")
Missing.Values.SAGA.AERO=paste(rep(-99999,NV.SAGA.AERO),collapse=",")
Var.names.units.SAGA.AERO=paste(Params.SAGA.AERO,Units.SAGA.AERO,sep=", ")
```

```
NLINES.SAGA.AERO=14+NV.SAGA.AERO+length(Special.comments.SAGA.AERO)+(N.Normal.Comments.S
AGA.AERO)

write(paste(NLINES.SAGA.AERO,"1001",sep=", "),file="MER-SAGA.AERO.icartt")
write(c("Wofsy,S", "Harvard_University", "merge of 1s ATom DC-8 data with SAGA.AERO flask samples",
"ATom", "1,1"),
ncol=1,file="MER-SAGA.AERO.icartt",append=T)
write(paste(YYYY, MM,DD, format(Sys.time(),tz="GMT","%Y,%m,%d"),sep=","),file="MER-
SAGA.AERO.icartt",append=T)
write("0",file="MER-SAGA.AERO.icartt",append=T) ## interval =0 , not regular apcing
write("UTC_Start, s",file="MER-SAGA.AERO.icartt",append=T)
write(NV.SAGA.AERO ,file="MER-SAGA.AERO.icartt",append=T)
write(Scale.Factors.SAGA.AERO ,file="MER-SAGA.AERO.icartt",append=T)
write(Missing.Values.SAGA.AERO ,file="MER-SAGA.AERO.icartt",append=T)
write(Var.names.units.SAGA.AERO ,file="MER-SAGA.AERO.icartt",ncol=1,append=T)
write(length(Special.comments.SAGA.AERO),file="MER-SAGA.AERO.icartt",append=T)
write(Special.comments.SAGA.AERO,file="MER-SAGA.AERO.icartt",ncol=1,append=T)
write(N.Normal.Comments.SAGA.AERO,file="MER-SAGA.AERO.icartt",append=T)
write(Normal.Comments.SAGA.AERO,file="MER-SAGA.AERO.icartt",ncol=1,append=T)
write.table(Mer.SAGA.AERO,file="MER-SAGA.AERO.icartt",
append=T,row.names=F,col.names=F,sep=",",na="-99999",quote=F)
#
if(file.exists("rev_curr_SAGA.AERO.txt") ) {          ## case REV is specified
if(lpublic){
system(paste("cp MER-SAGA.AERO.icartt MER-SAGA-AERO_DC8_",flt,"_R",REV+1,".ict",sep=""))
}else{
system(paste("cp MER-SAGA.AERO.icartt MER-SAGA-AERO-
restricted_DC8_",flt,"_R",REV+1,".ict",sep=""))
}
}
}
##End: merge for SAGA-AERO
##############################################################################################
####
### end Merge for SAGA-AERO

##Start: merge for MED
##############################################################################################
####

file.MED=system(paste("ls -1 MEDUSA_DC8_",flt,"_R?.ict",sep=""),intern=T)
if(length(file.MED)>0) {    ##proceed with MED
f.MED=file.MED[length(file.MED)] ## take the last revision
Dum.x = merge.medusa.atom(flt,path.to.flights="..") ## this function does the full merge with filling
kernel
Dum=Dum.x[,2:ncol(Dum.x)]
Dum.0 = read.1001(f.MED ) ## we read the file to get ICARTT info
#               use the time stamp for start of the interval, except for averaging measurements
```

```
########################  SKIP ALL the MERGING here !
l.ok=Dum[, 1 ]> Mer.1s[1,1] & Dum[,1 ]< Mer.1s[nrow(Mer.1s),1]
Dum=Dum[l.ok,]
##fix the colnames
## MED's do have a mid time
colnames(Dum)[1:4]=c("UTC_Start","UTC_Stop_MED","UTC_Mid_MED","UTC_Mean_MED")
#
LLOD_FLAG=as.numeric(unlist(str_split(system(paste("grep LLOD_FLAG",f.MED),intern=T),":"))[2])
ULOD_FLAG=as.numeric(unlist(str_split(system(paste("grep ULOD_FLAG",f.MED),intern=T),":"))[2])
#changed 20191027
#-# 20191027 Dum[Dum == LLOD_FLAG | Dum == ULOD_FLAG] =NA
Dum[Dum == LLOD_FLAG] = -8888 ;  Dum[Dum == ULOD_FLAG] = 77777   #-# 2019-10-27
#

prof.no=rep(0,nrow(Dum))
med.cuts=rep(0,nrow(Dum))
for(i in 1:nrow(Dum)){
  prof.no[i]=max(Mer.1s[Mer.1s[,"UTC_Start"]>=Dum[i,1] & Mer.1s[,"UTC_Start"]<=Dum[i,2]
,"prof.no"])
  med.cuts[Mer.1s[,"UTC_Start"]>=Dum[i,1] & Mer.1s[,"UTC_Start"]<=Dum[i,2] ] =i
                }
AA=cbind(Dum,prof.no)
# dateline trap fix
X.LONG=tapply(Mer.1s[,"G_LONG"],list(med.cuts),FUN=funny     )
AA[,"G_LONG"]=X.LONG[as.numeric(names(X.LONG))>0]
## cloudindicator mapping
## not for MED--not merged
X.cloud=tapply(Mer.1s[med.cuts!=0&is.finite(med.cuts),"cloudindicator"],list(med.cuts[med.cuts!=0&is.
finite(med.cuts)]),FUN=cloud.index     )
## not for MED--not merged AA[,"cloudindicator"]=X.cloud

#Mer.MED = cbind(Dum[,1:3],AB[,2:ncol(AB)],Dum[,4:ncol(Dum)])
Mer.MED = AA
#
##                                        v      v                   v
Special.comments.MED=c(Special.comments.1s,paste(f.MED
,paste(Dum.0$date.s,collapse=","),Dum.0$REV,sep=",") )
Params.MED=colnames(Mer.MED)[2:ncol(Mer.MED)]
#Units.MED=c("s","s","s",Dum.0$VUNITS[3:(ncol(Dum)-1) ] )
Units.MED=c("s","s","s",Dum.0$VUNITS[4:(length(Dum.0$VUNITS))
],"ppm","s","hPa","K","K","m/s","deg","deg","m","hPa","hPa","ppm","permil","ppm","ppm","ppm","pp
b",

"ppm","ppm","ppb","ppb","ppb","ppm","ppb","ppb","ppt","km","None","None",rep("None",26),"ppm",
"None")
names(Units.MED)=Params.MED
## do the averaging for each flask
```

```
###########   start write MED merge file
Normal.Comments.MED=readLines(paste(base.dir,"RPROGS/Normal_Comments.txt", sep="") )
if(file.exists("rev_curr_MED.txt") ) {          ## case REV is specified
REV=scan("rev_curr_MED.txt")
N.C=Normal.Comments.MED[1:(length(Normal.Comments.MED) -1 ) ]
N.C.addon=paste("REVISION: R",REV+1,sep="")
if(REV>=0) { for(i in 1:(REV+1)) { N.C.addon= c( N.C.addon , paste("R",i,": updated", sep="")) } }  ##
creating next rev
Normal.Comments.MED=c(N.C,N.C.addon)
}
Normal.Comments.MED=c(Normal.Comments.MED,paste("UTC_Start",paste(Params.MED,collapse=","),
sep=","))
N.Normal.Comments.MED=length(Normal.Comments.MED)  ##add line of parameters
NV.MED=length(Params.MED )
Scale.Factors.MED=paste(rep(1,NV.MED),collapse=",")
Missing.Values.MED=paste(rep(-99999,NV.MED),collapse=",")
Var.names.units.MED=paste(Params.MED,Units.MED,sep=", ")
NLINES.MED=14+NV.MED+length(Special.comments.MED)+(N.Normal.Comments.MED)

write(paste(NLINES.MED,"1001",sep=", "),file="MER-MED.icartt")
write(c("Wofsy,S", "Harvard_University", "merge of 1s ATom DC-8 data with MED flask samples",
"ATom", "1,1"),
ncol=1,file="MER-MED.icartt",append=T)
write(paste(YYYY, MM,DD, format(Sys.time(),tz="GMT","%Y,%m,%d"),sep=","),file="MER-
MED.icartt",append=T)
write("0",file="MER-MED.icartt",append=T) ## interval =0 , not regular apcing
write("UTC_Start, s",file="MER-MED.icartt",append=T)
write(NV.MED ,file="MER-MED.icartt",append=T)
write(Scale.Factors.MED ,file="MER-MED.icartt",append=T)
write(Missing.Values.MED ,file="MER-MED.icartt",append=T)
write(Var.names.units.MED ,file="MER-MED.icartt",ncol=1,append=T)
write(length(Special.comments.MED),file="MER-MED.icartt",append=T)
write(Special.comments.MED,file="MER-MED.icartt",ncol=1,append=T)
write(N.Normal.Comments.MED,file="MER-MED.icartt",append=T)
write(Normal.Comments.MED,file="MER-MED.icartt",ncol=1,append=T)
write.table(Mer.MED,file="MER-MED.icartt", append=T,row.names=F,col.names=F,sep=",",na="-
99999",quote=F)
#
if(file.exists("rev_curr_MED.txt") ) {          ## case REV is specified
if(lpublic){
system(paste("cp MER-MED.icartt MER-MED_DC8_",flt,"_R",REV+1,".ict",sep=""))
}else{
system(paste("cp MER-MED.icartt MER-MED-restricted_DC8_",flt,"_R",REV+1,".ict",sep=""))
}

}
save(Mer.MED,file=paste("Mer.MED_",flt,".RData",sep=""))
```

```
}
##End: merge for MED
#################################################################################################
####
##
##Start: merge for TOGA
#################################################################################################
####
##merge for TOGA
##merge for TOGA
file.TOGA=system(paste("ls -1 TOGA_DC8_",flt,"_R?.ict",sep=""),intern=T)
if(length(file.TOGA)>0) {    ##proceed with TOGA
f.TOGA=file.TOGA[length(file.TOGA)] ## take the last revision
Dum.0 = read.1001(f.TOGA )
Dum=Dum.0$dat
#               use the time stamp for start of the interval, except for averaging measurements
l.ok=Dum[, 1 ]> Mer.1s[1,1] & Dum[,1 ]< Mer.1s[nrow(Mer.1s),1]
Dum=Dum[l.ok,]
##fix the colnames which have "_ppt" at the end.  THIS SECTION TO CHANGE IF TOGA FILES CHANGE
#nom.cols=matrix(unlist(str_split(colnames(Dum),pattern="_")),ncol=2,byrow=T)
#colnames(Dum)[4:ncol(Dum)]=paste(nom.cols[4:ncol(Dum),1],"TOGA",sep="_")
colnames(Dum)[1:2]=c("UTC_Start_TOGA","UTC_Stop_TOGA")
#
LLOD_FLAG=as.numeric(unlist(str_split(system(paste("grep LLOD_FLAG",f.TOGA),intern=T),":"))[2])
ULOD_FLAG=as.numeric(unlist(str_split(system(paste("grep ULOD_FLAG",f.TOGA),intern=T),":"))[2])
#changed 20191027
#-# 20191027 Dum[Dum == LLOD_FLAG | Dum == ULOD_FLAG] =NA
Dum[Dum == LLOD_FLAG] = -8888 ;  Dum[Dum == ULOD_FLAG] = 77777   #-# 2019-10-27
#
toga.cuts=rep(0,nrow(Mer.1s))
for(i in 1:nrow(Dum)){
  toga.cuts[Mer.1s[,"UTC_Start"]>=Dum[i,"UTC_Start_TOGA"] &
Mer.1s[,"UTC_Start"]<=Dum[i,"UTC_Stop_TOGA"]]=i
                    }
#toga.cuts=as.numeric( cut(Mer.1s[,1],breaks=c(t(Dum[,1:2])) ,labels=as.character(1:(2*nrow(Dum) -
1))) )
#toga.cuts[toga.cuts%%2==0] = NA
AA=as.matrix ( aggregate(Mer.1s,by=list(toga.cuts),mean,na.rm=T) )
AA=AA[AA[,1]!=0,]
prf=( tapply(Mer.1s[toga.cuts!=0,"prof.no"],list(toga.cuts[toga.cuts!=0]),max,na.rm=T) )
AA[,"prof.no"]=prf
## correct longitude for dateline trap
X.LONG=tapply(Mer.1s[,"G_LONG"],list(toga.cuts),FUN=funny     )
AA[,"G_LONG"]=X.LONG[as.numeric(names(X.LONG))>0]
## correct longitude for dateline trap
## cloudindicator mapping
if(is.finite(match("cloudindicator",colnames(Mer.1s)))) {
X.cloud=tapply(Mer.1s[toga.cuts!=0,"cloudindicator"],list(toga.cuts[toga.cuts!=0]),FUN=cloud.index     )
```

```
AA[,"cloudindicator"]=X.cloud
}

lU=grep(pattern="UTC",colnames(AA))
lU=lU[2:length(lU)]
lkk=rep(T,ncol(AA))
lkk[lU]=F
AB=AA[,lkk]
colnames(AB)[2]="UTC_Mean_1s"
Mer.TOGA = cbind(Dum[,1:3],AB[,2:ncol(AB)],Dum[,4:ncol(Dum)])
#
Special.comments.TOGA=c(Special.comments.1s,paste(f.TOGA
,paste(Dum.0$date.s,collapse=","),Dum.0$REV,sep=",") )
# Params.TOGA=c(Params.1s, colnames(Dum)[4:ncol(Dum)] )
# Units.TOGA=c(Units.1s,Dum.0$VUNITS[4:ncol(Dum)] )
Params.TOGA=colnames(Mer.TOGA)[2:ncol(Mer.TOGA)]
Units.TOGA=c("s","s","s",Units.1s[lkk[3:length(lkk)] ],Dum.0$VUNITS[3:(ncol(Dum)-1) ] )
## do the averaging for each flask


###########   start write TOGA merge file
Normal.Comments.TOGA=readLines(paste(base.dir,"RPROGS/Normal_Comments.txt",sep="") )
if(file.exists("rev_curr_TOGA.txt") ) {          ## case REV is specified
REV=scan("rev_curr_TOGA.txt")
N.C=Normal.Comments.TOGA[1:(length(Normal.Comments.TOGA) -1 ) ]
N.C.addon=paste("REVISION: R",REV+1,sep="")
if(REV>=0) { for(i in 1:(REV+1)) { N.C.addon= c( N.C.addon , paste("R",i,": updated", sep="")) } }  ##
creating next rev
Normal.Comments.TOGA=c(N.C,N.C.addon)
}
Normal.Comments.TOGA=c(Normal.Comments.TOGA,paste("UTC_Start",paste(Params.TOGA,collapse="
,"),sep=","))
N.Normal.Comments.TOGA=length(Normal.Comments.TOGA)  ##add line of parameters
NV.TOGA=length(Params.TOGA )
Scale.Factors.TOGA=paste(rep(1,NV.TOGA),collapse=",")
Missing.Values.TOGA=paste(rep(-99999,NV.TOGA),collapse=",")
Var.names.units.TOGA=paste(Params.TOGA,Units.TOGA,sep=", ")
NLINES.TOGA=14+NV.TOGA+length(Special.comments.TOGA)+(N.Normal.Comments.TOGA)

write(paste(NLINES.TOGA,"1001",sep=", "),file="MER-TOGA.icartt")
write(c("Wofsy,S", "Harvard_University", "merge of 1s ATom DC-8 data with TOGA flask samples",
"ATom", "1,1"),
ncol=1,file="MER-TOGA.icartt",append=T)
write(paste(YYYY, MM,DD, format(Sys.time(),tz="GMT","%Y,%m,%d"),sep=","),file="MER-
TOGA.icartt",append=T)
write("0",file="MER-TOGA.icartt",append=T) ## interval =0 , not regular apcing
write("UTC_Start, s",file="MER-TOGA.icartt",append=T)
write(NV.TOGA ,file="MER-TOGA.icartt",append=T)
```

```
write(Scale.Factors.TOGA ,file="MER-TOGA.icartt",append=T)
write(Missing.Values.TOGA ,file="MER-TOGA.icartt",append=T)
write(Var.names.units.TOGA ,file="MER-TOGA.icartt",ncol=1,append=T)
write(length(Special.comments.TOGA),file="MER-TOGA.icartt",append=T)
write(Special.comments.TOGA,file="MER-TOGA.icartt",ncol=1,append=T)
write(N.Normal.Comments.TOGA,file="MER-TOGA.icartt",append=T)
write(Normal.Comments.TOGA,file="MER-TOGA.icartt",ncol=1,append=T)
write.table(Mer.TOGA,file="MER-TOGA.icartt", append=T,row.names=F,col.names=F,sep=",",na="-
99999",quote=F)
#
if(file.exists("rev_curr_TOGA.txt") ) {          ## case REV is specified
#
if(lpublic){
system(paste("cp MER-TOGA.icartt MER-TOGA_DC8_",flt,"_R",REV+1,".ict",sep=""))
}else{
system(paste("cp MER-TOGA.icartt MER-TOGA-restricted_DC8_",flt,"_R",REV+1,".ict",sep=""))
}
#
}

}
##End: merge for TOGA
##############################################################################################
####
## end of the perge script
```