AIRS data description

Part 1: monthly climatology

David W. Pierce
Ocean Research Consultants
david@david.pierce.name

15 March 2005

Contents

1 Introduction

The purpose of this note is twofold. First, it is to display some of the basic characteristics of the AIRS data. At the moment, the data has largely been unanalyzed, but it is hard to know how to analyze it in detail before one knows what the data “looks like.” The bulk of this note, then, simply presents plots of various aspects of the data, so that an understanding of its basic mean state and variability can be appreciated. In this first work, monthly and seasonal values are emphasized; the plots are generally presented without comment.

The second purpose is to demonstrate the use of the JPL netcdf operators (“jnc” operators) and other JPL deliverables. Processing started with the twice-daily level 3 gridded data in HDF-EOS format. This was converted to netCDF using the hdfeos-R package version 0.3, using the script in Appendix A. All subsequent data processing for the figures was done with version 0.5 of the jnc operators. Plotting was done with R, and a R plotting support library that is also included in the JPL deliverables. Therefore, at least conceptually, all the results presented here could have been done in the final JPL processing environment, whether web based, driven by a python script, or from the command line.

The data included covers the period September 2002 through January 2005. It should be pointed out that data aliasing issues are problematic with the satellite data, which are not daily averages but rather samples taken twice daily. The “ascending” data are taken at 1330 local time, while the “descending” data are taken at 0130 local time. As a result, there is basically a night sample and a day sample. Problems interpreting this data can arise if processes that are linked to the diurnal cycle fall outside of these two sample times, or, even worse, wander into and out of the sample times over the course of a year. For example, imagine a tropical region where it rains every afternoon, but in the early afternoon during winter and late afternoon during summer. This would incorrectly appear in the satellite record as dry summers and wet winters. No attempt is made to correct for this kind of problem in the basic analysis done here.

Another issue that complicates the interpretation of the figures is that the currently available level 3 gridded data does not seem to retain an explicit data retrieval quality flag, although it is present in the level 2 data. (Or, if it is included, it is done so implicitly, by virtue of no values being recorded for bad retrievals.) It will be seen in the figures that some values seem physically unrealistic; for example, the very warm daily surface high temperature values over land (Figure 1). It may be that these would be masked by the quality flag, were it present. The physical oddities will not be commented on further, pending investigation of whether the data quality flag was taken into account in the level 3 data and, if not, if it can be retrieved. Also, it should be kept in mind that part of the purpose of this project is to have the data processing operators be already available when the final released version of the data becomes available. As of this writing, the final data has not been released, so the data set used here should be considered preliminary, and subject to modification.

2 Surface Skin Temperature

Figure 1 shows the monthly surface skin temperature for the ascending orbit, while Figure 2 shows the same thing for the descending orbit. The difference between the two, which is related to the diurnal temperature cycle subject to the aliasing caveats noted in the introduction, is shown in Figure 3.

As one would expect, the diurnal range is far less over the ocean than over land, with dry land areas (such as the Sahara and central Asian deserts) showing the largest diurnal range. The increase in the diurnal range of ocean temperature in the summer hemisphere is also evident, and seems to be a noticably larger effect in the southern hemisphere than in the northern hemisphere.

Another interesting point is the enhanced diurnal variability in oceanic upwelling regions along the west coast of the continents. These regions also tend to be subject to persistent stratus decks, especially off the west coast of South America and central Africa. It is perhaps not immediately obvious why these regions should have a diurnal cycle appreciably stronger than other regions.

The standard deviation of the daily surface skin temperature anomalies are shown in Figure 4 for both the ascending branch of the orbit (upper panel) and the descending branch (lower panel). The greater day-to-day variability over continental interiors is, of course, very pronounced, except in the tropical regions.

3 Relative Humidity

The AIRS instrument can sense relative humidity in the vertical as well as the horizontal, which is a major advance in instrumentation. Figure 5 shows the relative humidity reported at level 1, the level nearest the surface. Some of the largest seasonal changes in humidity occur over India and Australia; in most other regions the seasonal differences

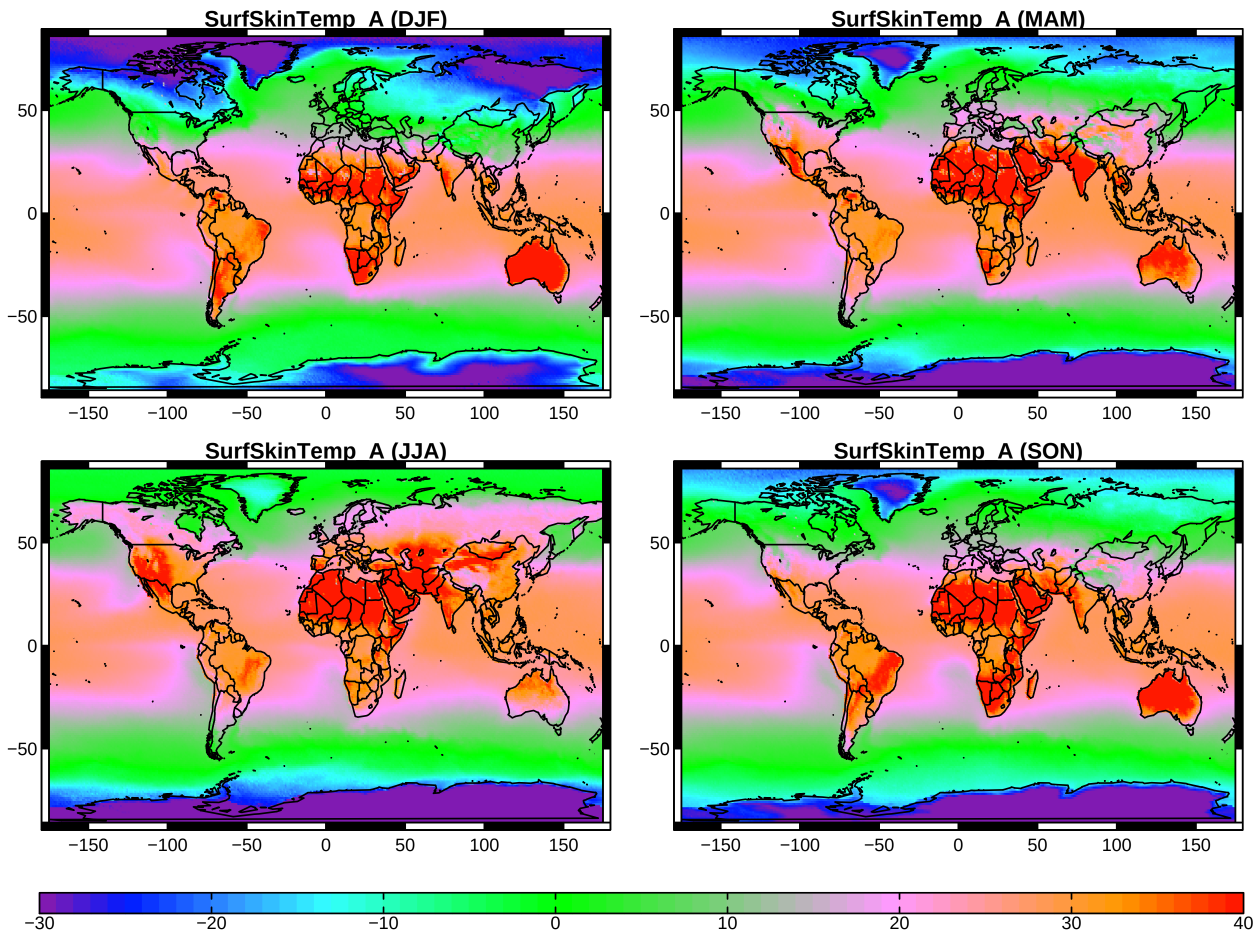

Figure 1: Surface skin temperature (degC) on the ascending branch of the orbit (1330 local time).

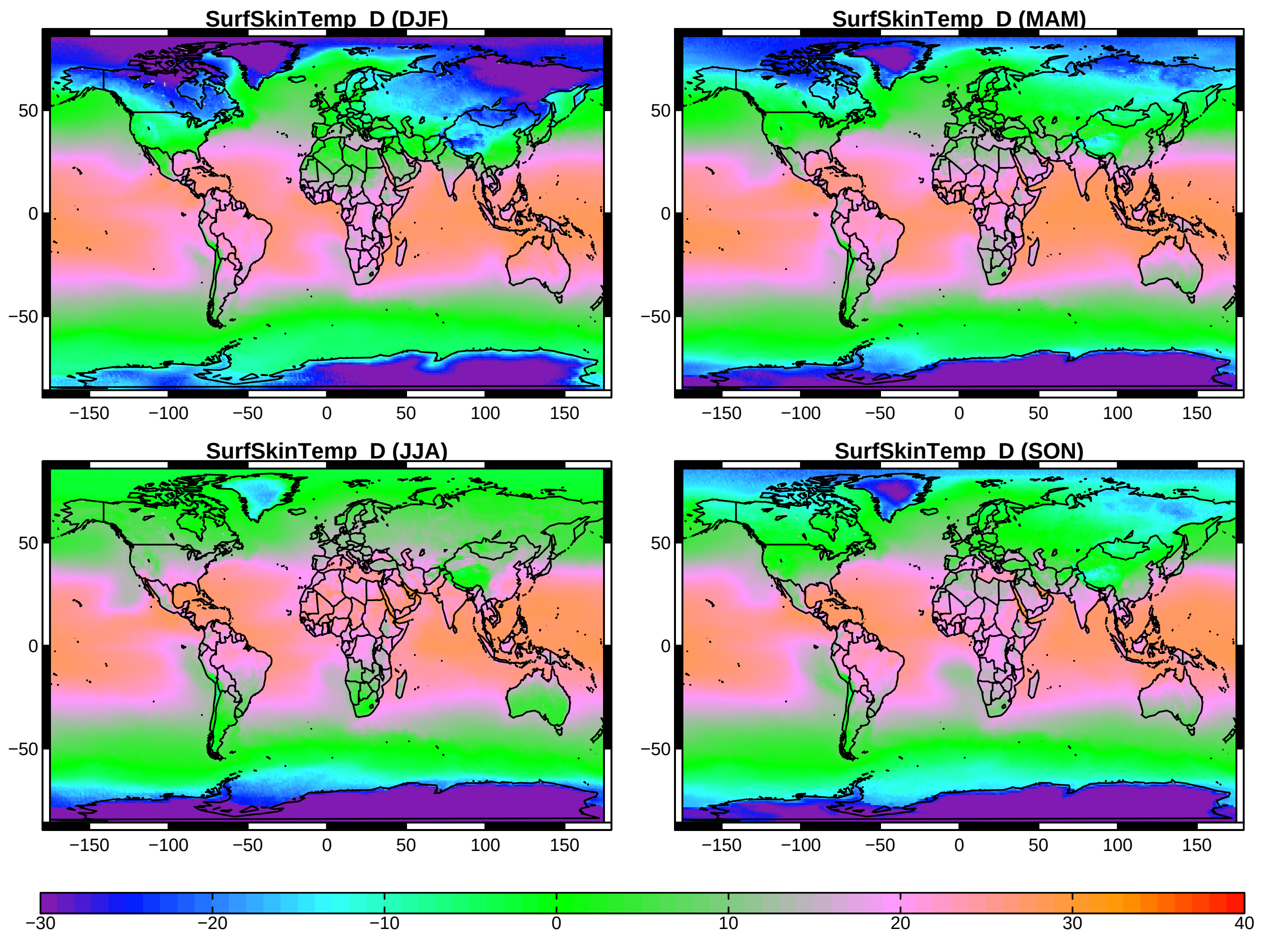

Figure 2: Surface skin temperature (degC) on the descending branch of the orbit (0130 local time).

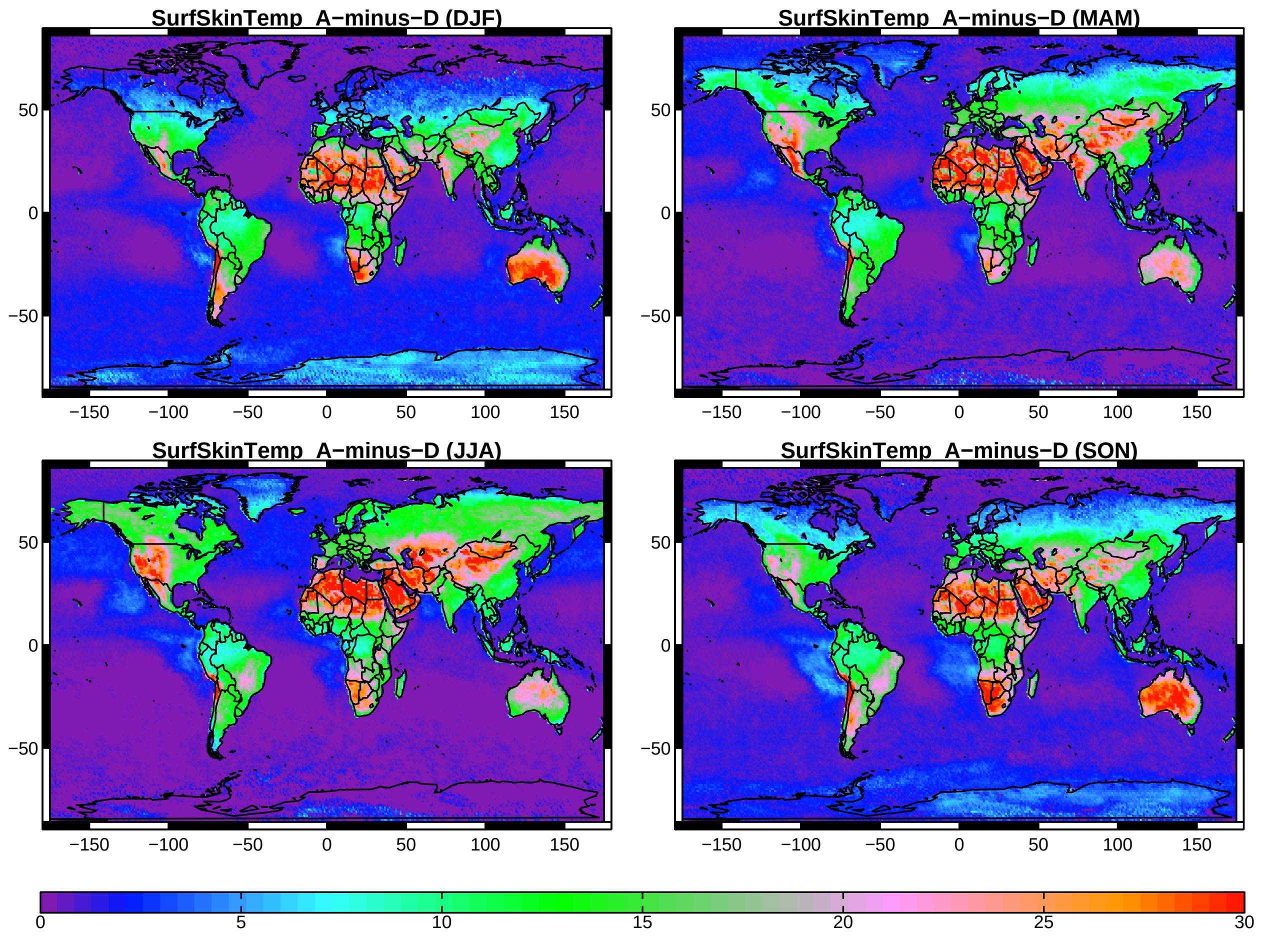
SurfSkinTemp A−minus−D (DJF)
SurfSkinTemp A−minus−D (MAM)
SurfSkinTemp A−minus−D (JJA)
SurfSkinTemp A−minus−D (SON)
50
0
−50
−150
−100
−50
0
50
100
150
0
5
10
15
20
25
30

Figure 3: Difference (degC) between surface skin temperature on the ascending branch and descending branch of the orbit.

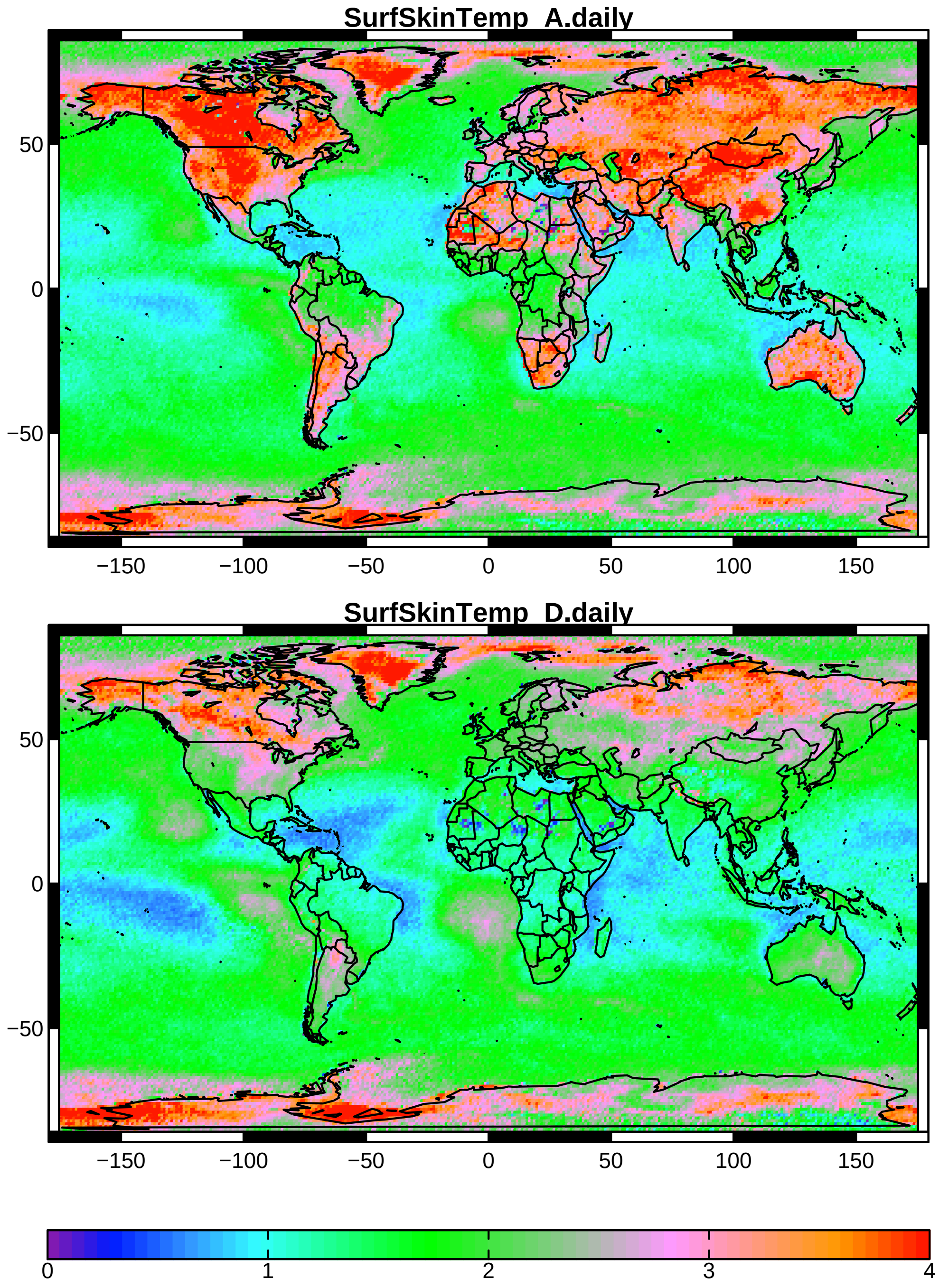

Figure 4: Standard deviation (degC) of the daily anomalies for surface skin temperature. Upper panel: the ascending branch of the orbit (1330 local time). Lower panel: the descending branch of the orbit (0130 local time).

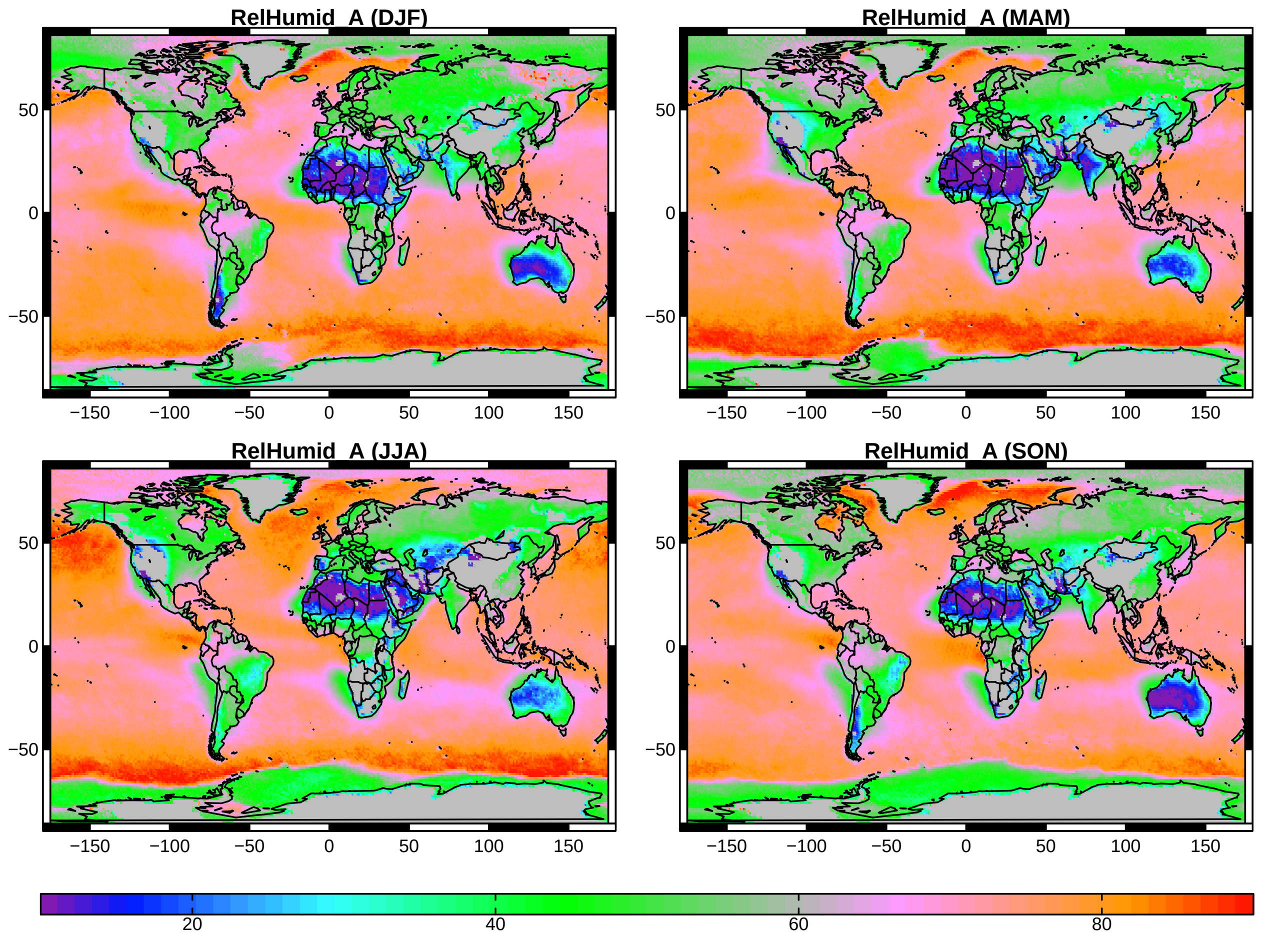

Figure 5: Seasonal average relative humidity (percent) at the nearest surface level, for the ascending branch of the orbit (1330 local time).

appear rather modest. The low humidity values over near-complete areal coverage of sea ice (for example, around Antarctica in the southern hemisphere winter) are notable, and suggest that if the data quality flags indicate the retrievals are valid in this region, the extend of sea ice would be easily obtainable. On the other hand, there is also a suggestion that the reported values may be skewed in the marginal ice zone, as they are higher there than anywhere else.

A similar plot, but at level 5, is shown in Figure 6. The very dry descending branches of the Hadley cell show up clearly at this level. This is especially true in the Southern hemisphere, where the band of dry air is near continuous at a latitude of about 20 S. In both hemispheres, the relative humidity values of this dry air are noticably lower in the winter than in the summer.

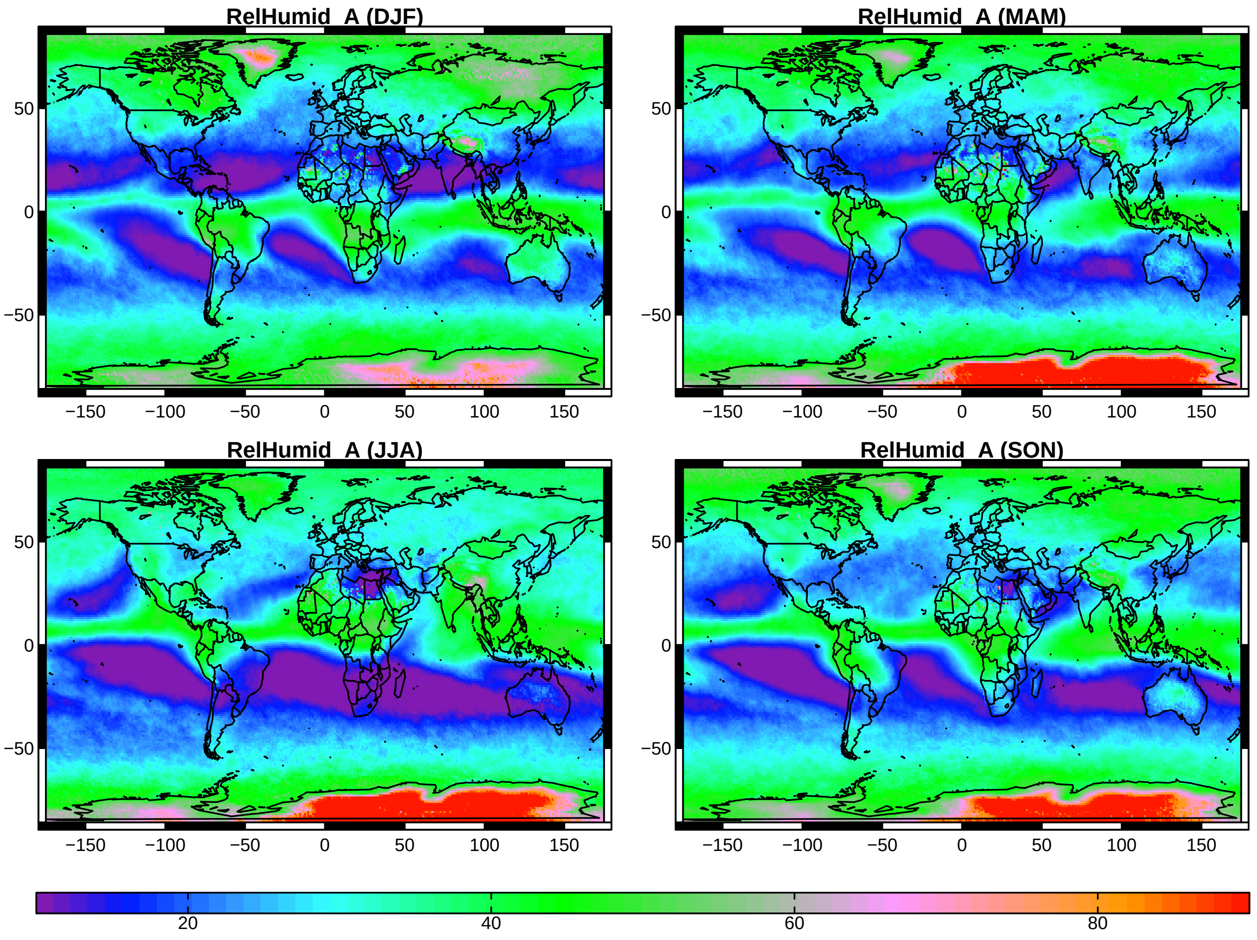

Figure 6: Seasonal average relative humidity (percent) at level 5, for the ascending branch of the orbit (1330 local time).

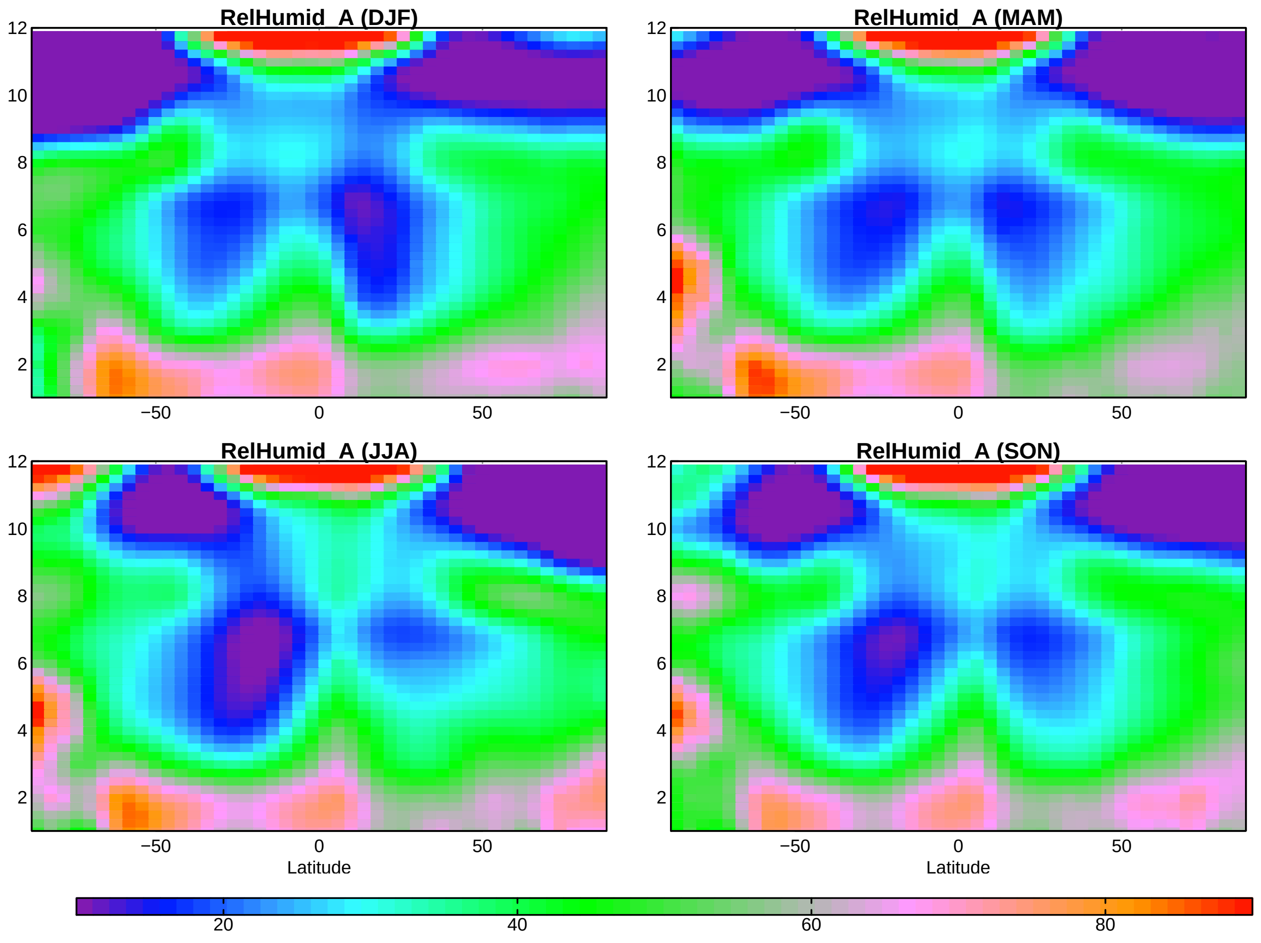

Figure 7: Zonal mean relative humidity (percent) by season as a function of latitude and vertical level, for the ascending branch of the orbit (1330 local time).

The zonal mean values as a function of latitude and level are shown in Figure 7. Again, in the dry air that is sinking at about 20 N, the relative humidity is appreciably lower in the winter than in the summer. Also, the overall humidity levels of this sinking air are rather lower in the Southern hemisphere than in the Northern. The standard deviation of the monthly values for the zonally averaged relative humidity are shown in Figure 8. The largest values are reported as being in the polar regions of both hemispheres, although they are at high altitudes in the Southern hemisphere and low altitudes in the Northern hemisphere. Low variability is seen near the surface in the tropics, and at altitude in the midlatitudes.

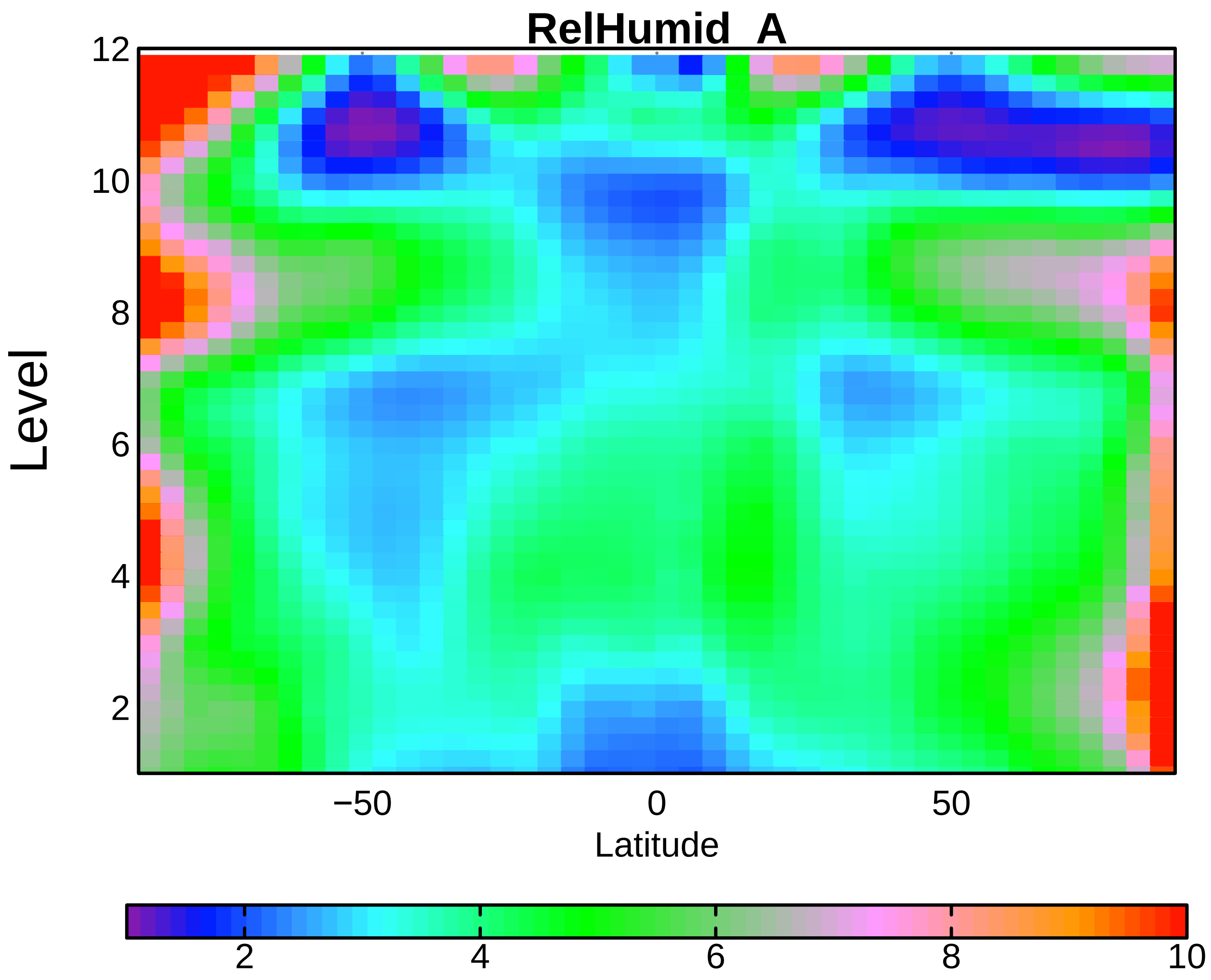

Figure 8: Standard deviation (percent) of the zonal mean relative humidity as a function of latitude and vertical level, for the ascending branch of the orbit (1330 local time).

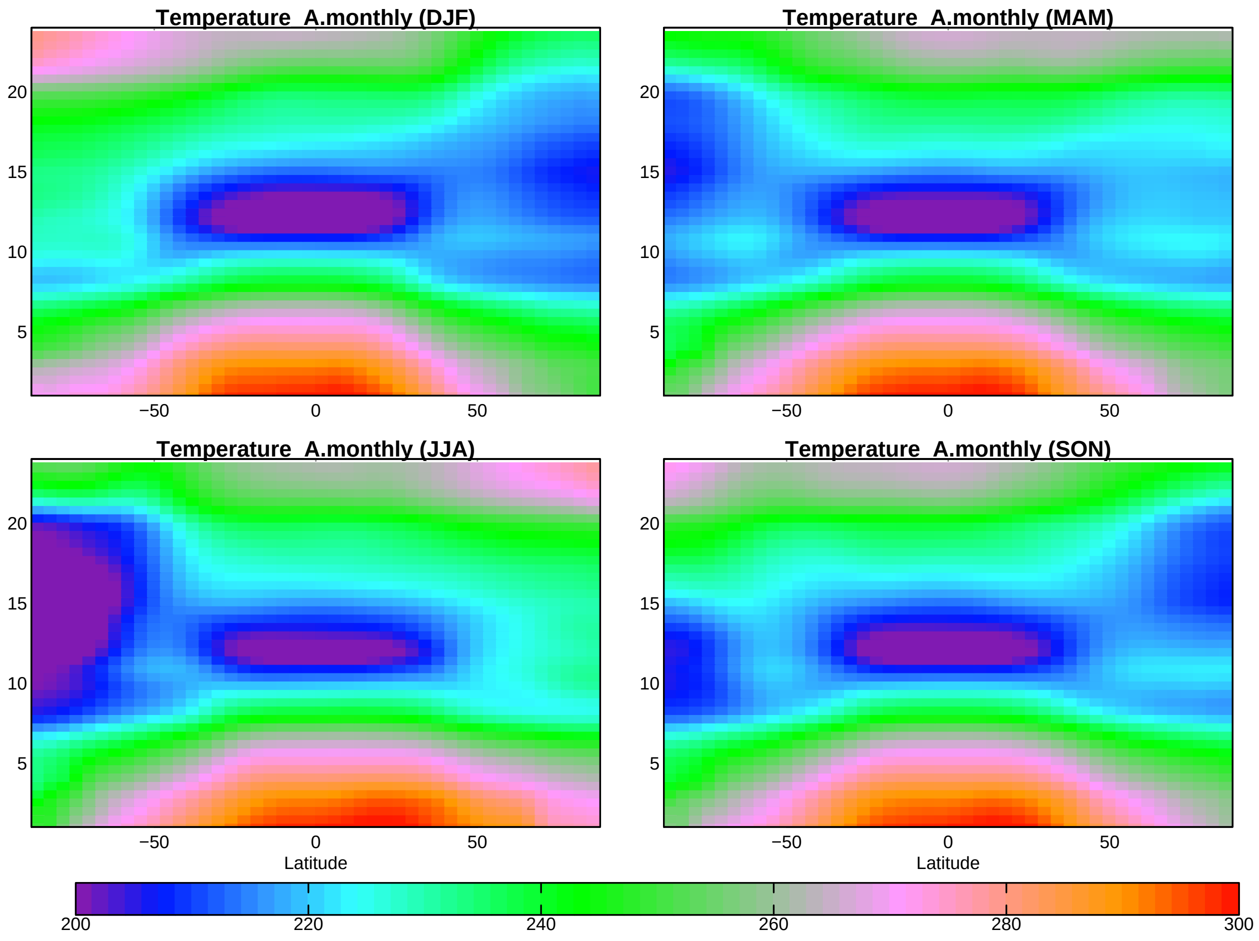

Figure 9: Zonal mean monthly temperature (degC) by season, as a function of latitude and vertical level, for the ascending branch of the orbit (1330 local time).

4 Three-Dimensional Temperature

In addition to the surface skin temperature noted above, AIRS samples atmospheric temperature on 24 vertical levels. Zonal mean averages of seasonally average temperature are shown in Figure 9, and the standard deviation of monthly anomalies in Figure 10. This of course shows the cooler atmospheric temperatures in the winter hemisphere – no surprise – and the very low temperatures in particular over Antarctica in the winter. The standard deviation reports a strong bias towards high values of variability over the Arctic ocean, which has near twice the standard deviation of values over Antarctica. Values are generally lowest in the tropics, especially near the surface.

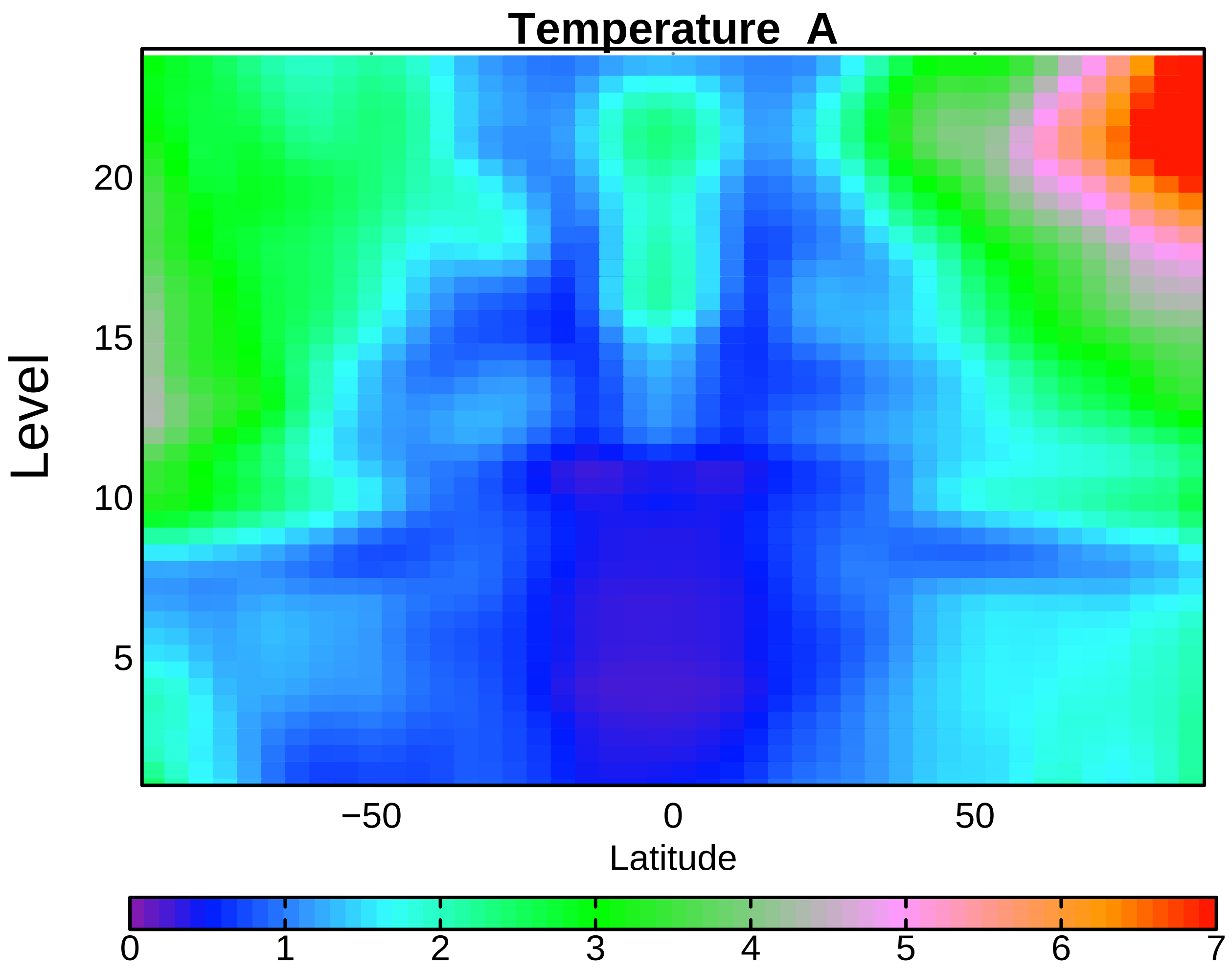

Figure 10: Standard deviation (deg-C) of the zonal mean monthly temperature anomaly as a function of latitude and vertical level, for the ascending branch of the orbit (1330 local time).

5 Outgoing Longwave Radiation (OLR)

The OLR field in the level 3 data appears to have a problem; all values are either zero or the fill value. This goes for both branches of the orbit (ascending and descending).

6 Summary

The purpose of this note has been to show a few climatological data fields from the AIRS level 3 data. It should be kept in mind that a primary motivation for this exercise has been to demonstrate the utility of the jnc scientific operators, which were used for all data processing (the plotting was done in R). As such, a preliminary version of the data was used, but this is not relevant to demonstrating the ability of the operators to handle the data processing requirements. The preliminary nature of the data does seem to be evident in a few of the reported fields, however. Also, the sampling and aliasing issues are significant, and need to be taken into account when the results are examined. This problem is intrinsic to the way the satellite takes its observations (once in the night, once in the day) and so will always be an issue, even in the final version of the data set.

From a purely data processing point of view, a few issues do come up. One is that the level 3 data, as delivered, does not appear to retain the data quality flag that is present in the level 2 data. It is not clear whether this is because the information has already been incorporated into the level 3 data by, for example, omitting from the level 3 processing all level 2 data that does not have a good quality flag. One argument against this is the fact that some of the fields seem to have implausible values in places where retrieval quality might be suspect, for example, in the marginal sea ice zone. Another issue is that the outgoing longwave radiation (OLR) data appears to be missing from the current level 3 product. Presumably, since this is not yet the final release of the data, these issues will be resolved before the final data product is released.

A An R script to convert HDF-EOS to netCDF

```
tunits  <- 'seconds since 1993-01-01 00:00'  # HDF-EOS standard (?)


#=========================================================================
convert_latlon_2d_1d <- function( lat2d, lon2d ) {

nd <- length(dim(lat2d))
if( nd != 2 )
stop(paste("expected input lat2d to have 2 dims; instead it has",nd))

nd <- length(dim(lon2d))
if( nd != 2 )
stop(paste("expected input lon2d to have 2 dims; instead it has",nd))

nx <- dim(lat2d)[1]
ny <- dim(lat2d)[2]
if( nx != dim(lon2d)[1] )
stop(paste("expected lat2d and lon2d to have same nx; but they don't",nx,dim(lon2d)[1]))
if( ny != dim(lon2d)[2] )
stop(paste("expected lat2d and lon2d to have same nx; but they don't",ny,dim(lon2d)[2]))

#---------------------------------------------------
# See if lat can be expressed as a repeated 1-d array
#---------------------------------------------------
for( iy in 1:ny )
for( ix in 2:nx )
if( lat2d[ix,iy] != lat2d[1,iy] )
stop("sorry, latitude cannot be expressed as a simple function of X -- reprogram!")

#-------------
# Same for lon
#-------------
for( ix in 1:nx )
for( iy in 2:ny )
if( lon2d[ix,iy] != lon2d[ix,1] )
stop("sorry, longitude cannot be expressed as a simple function of y -- reprogram!")

rv <- list()

rv$lat <- lat2d[1,]
rv$lon <- lon2d[,1]

return(rv)
}

#=========================================================================
parse_fname_for_date <- function( fname, year, month ) {

#------------------------------------------------------------------
# First find where the string "year.month." exists in the filename
#------------------------------------------------------------------
```

```
iy <- as.integer(year)
im <- as.integer(month)
strpat <- paste(mformatC("%04d",iy),".",
mformatC("%02d",im),".", sep='')
npat   <- nchar(strpat)
ipl    <- -1
for( i in 1:nchar(fname) ) {
if( strpat == substr(fname,i,i+npat-1))
ipl <- i
}
if( ipl < 0 )
stop(paste("did not find year/month pattern in filename!  Pattern=",
strpat,"  filename=",fname))

# example: '2003/01/AIRS.2003.01.01.L3.RetStd001.v4.0.8.0.L3_Test.T05025125354.hdf'
yr <- (substr(fname, ipl,   ipl+3))
mo <- (substr(fname, ipl+5, ipl+6))
dy <- (substr(fname, ipl+8, ipl+9))

date <- list()
class(date) <- 'utDate'
date$year   <- yr
date$month  <- mo
date$day    <- dy

#----------------------------------------------------------------------
# Should fix these to correspond to ascending or descending branches --
# get info from Amy or Eric
#----------------------------------------------------------------------
date$hour   <- 12
date$minute <- 0
date$second <- 0.

print(date)
tval <- utInvCalendar( date, tunits )

return(tval)
}

#=======================================================================
dat2nc <- function( data_dir, year, month, grid, field, units ) {

month_tag <- mformatC( '%02d', month )
dir <- paste(data_dir,"/", year,"/",month_tag,"/airx3std",sep='')
files <- getfilelist( paste(dir,"/*.hdf",sep="" ))

nx <- 360
ny <- 180

outfile <- paste(field,'.',year,'.',month_tag,'.nc',sep='')

got_latlon_ok <- FALSE
file_to_examine <- 1
```

```
while( ! got_latlon_ok ) {
#----------------
# Get lat and lon
#----------------
hdf   <- EosGDOpen(files[file_to_examine])
if( hdf$hdfid == -1 ) {
print(paste("when trying to get location, error opening file",files[file_to_examine]))
crash
}
lat2d <- EosGDreadfield(hdf,'location','Latitude')
lon2d <- EosGDreadfield(hdf,'location','Longitude')

if( (length(lat2d)>1) && (length(lon2d)>1) ) {
#-----------------------------------
# Convert 2-D lat/lon to 1-D, or fail
#-----------------------------------
rv  <- convert_latlon_2d_1d( lat2d, lon2d )
lat <- rv$lat
lon <- rv$lon

got_latlon_ok <- TRUE
}
else
file_to_examine <- file_to_examine + 1

#---------------------------------------------
# How many dims does the requested field have?
#---------------------------------------------
ff <- hdf$grid[[grid]]$field[[field]]
ndims <- ff$ndims

if( ndims > 2 ) {
if( ndims == 3 ) {
nz <- ff$dimlen[1]   # C ordering
zvals <- 1:nz  # ??? where do we get the real values??
}
else
stop(paste("uncaught case for # of dims in field:",ndims))
}

EosGDClose(hdf)
}

#-------------------
# Create output file
#-------------------
dimx <- dim.def.ncdf('Lon','degreesE',lon)
dimy <- dim.def.ncdf('Lat','degreesN',lat)
dimlist <- list(dimx,dimy)
if( ndims >= 3 ) {
dimz <- dim.def.ncdf('Lev','1',zvals)
dimlist[[3]] <- dimz
}
```

```
dimt <- dim.def.ncdf('Time',tunits,0,unlim=TRUE)
dimlist[[ndims+1]] <- dimt

mv <- 1.e30
mainvar <- var.def.ncdf( field, units, dimlist,
longname=paste("field",field,"from grid",grid),mv )

ncid <- create.ncdf(outfile, list(mainvar))

#--------------------------------------------------------
# Set up start and count to use.  Remember that 'ndims'
# indicates SPATIAL dims only; we actually have one more
# than that, since we add a time dim.
#--------------------------------------------------------
if( ndims == 2 ) {
start <- c(1,1,1)
count <- c(nx,ny,1)
}
else if( ndims == 3 ) {
start <- c(1,1,1,1)
count <- c(nx,ny,nz,1)
}
else
stop(paste("uncaught case for ndims=",ndims))

nf <- length(files)
for( i in 1:nf ) {
print(paste("working on file",files[i]))

#---------------------------
# Parse filename to get date
#---------------------------
timeval <- parse_fname_for_date( files[i], year, month )

hdf  <- EosGDOpen(files[i])
if( hdf$hdfid == -1 ) {
print(paste("Error opening file", files[i]))
crash
}
else
{
data <- EosGDreadfield( hdf, grid, field )
if( (length(data) == 1) && (is.na(data)))
print(paste("Error reading data from file", files[i]))
else
{
start[ndims+1] <- i
put.var.ncdf( ncid, mainvar, data,   start=start, count=count )
put.var.ncdf( ncid, dimt,   timeval, start=i,     count=1 )
}
EosGDClose(hdf)
}
}
```

```
close.ncdf(ncid)
}

#========================================================================
doallmonths <- function( y0, m0, y1, m1, data_dir, grid, field, units ) {

y <- y0
m <- m0
for( i in 1:999999 ) {
if( ((y == y1) && (m > m1))  ||  (y>y1))
return
dat2nc( data_dir, y, m, grid, field, units )
m <- m + 1
if( m == 13 ) {
m <- 1
y <- y + 1
}
}

}

#========================================================================

utInit()

data_dir <- '/genesis/data/airs/L3_thunder/thunder/AIRS_Data'

grid  <- 'ascending'
#grid  <- 'descending'

#field <- 'SurfSkinTemp_A'
#field <- 'SurfSkinTemp_D'
#field <- 'TotH2OVap_A'
field <- 'RelHumid_A'
units <- 'kg'

#-------------------------------------------------------------------------
# If invoked with 2 command line arguments, this treats them as year and
# month to process and does just that month.  Otherwise, it does all
# years and months we have.
#-------------------------------------------------------------------------
argv <- commandArgs()
argc <- length(argv)

{
if(argc > 2){
#-----------------------------------------------------------------------
# This is for running from the command line with 2 args, year and month
#-----------------------------------------------------------------------

if( argc != 4 ) {
stop(paste("takes 2 args: year and month to process.  # args found:",argc-2))
}
year  <- argv[3]
month <- argv[4]

print(paste("working on year=",year,"month=",month))
dat2nc( data_dir, year, month, grid, field, units )
}
else
{
y0 <- 2002
m0 <- 9
y1 <- 2005
m1 <- 1
doallmonths( y0, m0, y1, m1, data_dir, grid, field, units )
}
}
```