5 November 2014

TO: NOMADSS Project File
FROM: Al Cooper
SUBJECT: A new terrain-elevation variable for NOMADSS

The Source For Data

During the Shuttle Radar Topography Mission (SRTM) of 2000,[1][2] the altitude of the Earth's surface was mapped from 56S to 60N latitude with resolution of 3 arc-sec or about 90 m at the equator. For the US and territories, the resolution was 1 arc-sec or about 30 m. The data from this mission are archived at this web site: http://www.webgis.com/srtm3.html. The measurements can be download in individual files that span 1 degree by 1 degree. The format of these files leads to the need for some processing that is documented here. An improved dataset that is largely based on the SRTM database but has been edited extensively to fill in missing values is available here: http://www.viewfinderpanoramas.org/dem3.html. That database was constructed and is maintained by Jonathan de Ferranti BA, Lochmill Farm, Newburgh, Fife, KY14 6EX, United Kingdom. For the US, higher-resolution data are available at this web page: http://ned.usgs.gov/epqs/. This USGS website provides the ability for a user to supply a latitude-longitude position and receive a height measurement, so it could have been used for NOMADSS instead of the Ferranti site that was selected, but the higher resolution does not have an advantage for the present purpose and it was thought preferable to develop a tool that would work worldwide. However, for NOMADSS the USGS data are valuable for checking the results obtained, as discussed later in this memo.

The R code that downloaded these files is in the 'chunk' of this document called 'download-zip-files'. Initially, the range downloaded covered 24N to 45N latitude and -105W to -72W longitude. After unzipping, the data set was large, approaching 1 GB as saved for use in this routine, but the area covered was a significant part of the U.S. so this large database was needed to span that full area. The heights in the 3-arc-sec files are presented in 1201×1201 arrays where the edges duplicate the values in the adjacent arrays and the measurements cover $1^\circ \times 1^\circ$. The format is row-major, i.e., the 1201 values for the first west-to-east row are presented first, then the next row to the south, etc. Because R is inherently column-major, there are some aspects of indexing in the code used here that have indices reversed from what might have been expected. The unpacked individual 1-deg files have 2,884,802 bytes. In these archives, the missing-value flag is -32768.

The reference location for each 1-degree by 1-degree array is the name of the individual file (e.g., "N43W101.hgt" has a reference position of 43°N and 101°W at the center of the southwest-corner element of the array). The values give the height in meters above the WGS84/EGM96 geoid. The measurement uncertainty was about 9 m at 90% confidence[3] (Farr et al.~2007), but there are some biases. The SAR radar did not always penetrate fully through vegetation and so might reflect the top of the vegetation canopy or some level intermediate between the canopy and the surface, and the radar penetrated a few meters into snow and so measured a height between

[1] Farr, T.G., M. Kobrick, 2000, Shuttle Radar Topography Mission produces a wealth of data, Amer. Geophys. Union Eos, v. 81, p. 583-585.

[2] Farr, T. G., et al. (2007), The Shuttle Radar Topography Mission, Rev. Geophys., 45, RG2004, doi:10.1029/2005RG000183

[3] The standard uncertainty would be about 5 m.

the snow cover and the terrain (as measured in Feb. 2000). Also, there are some gaps, especially in mountainous areas, although those have mostly been filled in this Ferranti-edited dataset by reference to topographic maps and there are only very rare gaps for the CONUS.

Below, R code that downloads, unzips, and reads the data files is listed. The entire-Earth dataset would require about 25 GB to store, so the download should be limited in area to the region of the project. For NOMADSS, the project range covered a large portion of the SE U.S., so there was still a necessity to download and save a large amount of data, approaching 1 GB. The routine is set to do this the first time it is run and save the data in an archive that can be used on subsequent runs. Some files are missing because they cover only ocean areas; in this processing, those are interpreted as leading to zero values for the terrain.

Here the data files are saved in Rdata-format gzipped files suitable for loading via commands like "load(file='ZN40W101.gz')", which will retrieve the 'height' matrix for that lat/lon square. This is handled in the code-chunk 'download-zip-files' listed below.

```
# there must be a subdirectory named 'TerrainData'
setwd ("./TerrainData")     # Save the data in a subdirectory
###### next are the limits for the range to download
## each zip file contains 4 deg latitude x 6 deg longitude, for the source used
#     (http://www.viewfinderpanoramas.org/dem3.html)
#     Acknowledgement: Jonathan de Ferranti BA
#                      Lochmill Farm
#                      Newburgh, Fife, KY14 6EX, United Kingdom
## Identifier for individual files is lat/lon at SE corner
#  Identifier for zip files containing 4 x 6 individual files is [none/S]LetterNmbr where
## for US, e.g., NOMADSS, indices are as follows:
#      letter = LETTER[floor (lat/4) + 1]
#      Nmbr = 30 + floor (lon/6) + 1
#  from Janine: range is 24--45N and 105--72W for NOMADSS
# lt_s <- 48    # limits for DEEPWAVE
# lt_n <- 40
# lg_w <- 165
# lg_e <- 175
# range for NOMADSS: had to download G13 to L19
lt_s <- 24 # N
lt_n <- 45 # N
lg_w <- -106 # W -- used 106 to get exactly 105 to work
lg_e <-  -72 # W
###### loop through the needed files
for (lt in lt_s:lt_n) {    # latitude limits (note 'N' or 'S' in sprintf statement)
  ifelse (lt >= 0, NS <- 'N', NS <- 'S')
  for (lg in lg_w:lg_e) {   # longitude limits (note 'E' or 'W')
    ifelse (lg >= 0, EW <- 'E', EW <- 'W')
    sname <- sprintf("Z%s%d%s%03d.gz", NS, abs(lt), EW, abs(lg))
    dname <- sprintf ("%s%02d%s%03d.hgt", NS, abs(lt), EW, abs(lg)) # a sq. degree of data
    if (file.exists(sname)) {   # Skip if file is already present
```

```
      unlink (dname)
    } else {
# is it already there from a previous download?
      if (file.exists (dname)) {
        #                # 'swap' changes from big-endian to little-endian
        height <- readBin (dname, 'int', size=2, n=1201*1201, endian='swap')
        height [height == -32768] <- NA      # set NA for missing values
        dim (height) <- c(1201,1201)         # Make into a matrix
        save (height, file=sname, compress='gzip')
        unlink (dname) # delete the unzipped file
      } else {
# find the database file that contains this:
        lettr <- floor (lt %/% 4) + 1
        numbr  <- 30 + floor (lg %/% 6) + 1
        if (lt < 0) {
          zipFileName <- sprintf ("S%s%02d.zip", LETTERS[1-lettr], numbr)
        } else {
          zipFileName <- sprintf ("%s%02d.zip", LETTERS[lettr], numbr)
        }
      # sprintf(" needed zip file is %s", zipFileName)
      # if it's already present, skip download
        if (!file.exists(zipFileName)) {
          url <- sprintf("http://www.viewfinderpanoramas.org/dem3/%s", zipFileName)
          if (RCurl::url.exists (url, followlocation=FALSE)) { # there are false moved-URLs ...
            f = RCurl::CFILE(zipFileName, mode="wb")
            RCurl::curlPerform(url = url, writedata = f@ref)
            close(f)
            unzip (zipFileName, junkpaths=TRUE)
            unlink (zipFileName)
            system (sprintf("touch %s", zipFileName))
            ## The reason for the preceding statement is to prevent trying to
            ## download repeatedly in cases where the file is not present, for
            ## example because it is entirely over ocean.
          }
        }
      }
      if (file.exists(dname)) {
        height <- readBin (dname, 'int', size=2, n=1201*1201, endian='swap')
        height [height == -32768] <- NA      # set missing values to NA
        dim (height) <- c(1201,1201)         # Make into a matrix
        save (height, file=sname, compress='gzip')
        unlink (dname) # delete the unzipped file
      }
    }
  }
```

```
}
setwd("..")
```

The values in these data files are binary two-byte or 16-bit signed integers and are in big-endian format (most significant byte first) while our processing machines are mostly little-endian, so a byte-swapping conversion is necessary. This was readily performed by the R reading function 'readBin', as illustrated in the preceding code chunk. It was useful to construct a function that would return the terrain altitude for a given latitude and longitude, so that is shown in the next chunk. This function is also used in the validation step discussed later in this memo.

```
HeightOfTerrain <- function (.lat, .long) {
  lt <- as.integer (floor(.lat))
  lg <- as.integer (floor(.long))
  if (is.na(lt) || is.na(lg)) {return (NA)} # beware of bad input
  if (lt < 0) {
    NS <- "S"
    lt <- -lt
  } else {
    NS <- "N"
  }
  if (lg < 0) {
    EW <- "W"
    lg <- -lg
  } else {
    EW <- "E"
  }
  vname <- sprintf("Z%s%02d%s%03d", NS, lt, EW, lg)
  if (!exists(vname, .GlobalEnv)) {
    zfile <- sprintf("%s.gz", vname)
    if (file.exists(sprintf("./TerrainData/%s",zfile))) {
      load(file=sprintf("./TerrainData/%s.gz", vname))
      assign (vname, height, envir=.GlobalEnv)
    } else {
      return (NA)
    }
  }
  ix <- as.integer ((.long - floor (.long) + 1/2400) * 1200) + 1
  iy <- as.integer ((ceiling (.lat) - .lat + 1/2400) * 1200) + 1
  if (ceiling (.lat) == .lat) { # exact match fails; correct it
    iy <- 1201
  }
  hgt <- get(vname, envir=.GlobalEnv)[ix, iy]
  return (hgt)
}
```

Testing results against the USGS values

The USGS provides a service where a user can obtain the height of terrain at a specified latitude and longitude. The web site is http://ned.usgs.gov/epqs. The following code chunk generates random latitude and longitude coordinates covering the NOMADSS operational area, fetches the altitude from the USGS site, and compares it to the altitude obtained from the 'HeightOfTerrain' subroutine used in this program.

```
require(RCurl)
n <- 5000
HOT <- vector ("numeric", n)
USGS <- vector ("numeric", n)
ltt <- runif (n, lt_s, lt_n)
lgg <- runif (n, lg_w, lg_e)
for (i in 1:n) {
  HOT[i] <- HeightOfTerrain (ltt[i], lgg[i])
  url <- sprintf ("http://137.227.248.58/pqs.php?x=%f&y=%f&units=Meters&output=json", lgg[i], lt
  USGS[i] <- as.numeric (strsplit (strsplit (RCurl::getURL(url), 'Elevation\":')[[1]][2], ',.*')
}
meanDiff <- mean (HOT-USGS, na.rm=TRUE)
sdDiff <- sd (HOT-USGS, na.rm=TRUE)
print (sprintf ("mean difference: %f", meanDiff))
print (sprintf ("std dev: %f", sdDiff))
hist (HOT-USGS, breaks=50, xlim=c(-10,10))
```

The mean difference from this comparison was -0.7 m and the standard deviation in the difference was 3.4 m for 5000 generated points. Because these data sources have different resolution (about 3 arc-sec for SRTM, about 1/3 arc-sec for USGS), some small differences are expected, but these values are closer than would be expected from the uncertainty in the SRTM values discussed above so this is good evidence that the values used here are within expected tolerances for uncertainty. Figure 1 is a histogram of the differences for the 5000 randomly selected positions.

Adding a netCDF terrain-height variable

For a netCDF file, it is then possible to define new variables that represent the elevation of the terrain below the aircraft and also the height of the aircraft above the terrain. For example, here is code that does this:

```
fname <- sprintf("%s%s/%s%s.nc", DataDirectory (), Project, Project, Flight)
fnew <- sprintf("%s%s/%s%sWAC.nc", DataDirectory (), Project, Project, Flight)
# copy file to avoid changing original: note 'Z' in new file name
file.copy (fname, fnew, overwrite=TRUE)   #careful: will overwrite 'Z' file
```

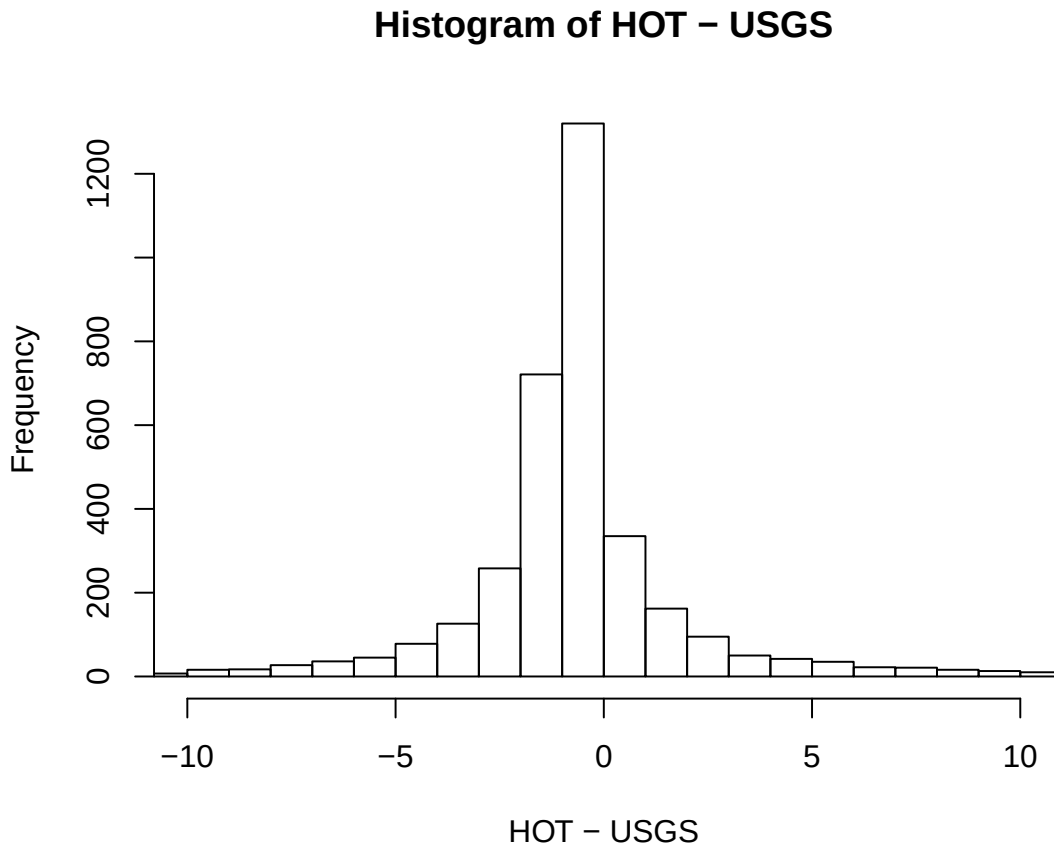

Figure 1: *Difference in altitude between that obtained from the USGS web site and that provided by the function HeightOfTerrain in this program, in meters, for the 5000 randomly generated points in the NOMADSS operations area.*

```
## [1] TRUE

# load data needed to calculate the new variables:
#   ... there is no GGALTB in NOMADSS files? (What is GGALTC?) Use GGALT
Data <- getNetCDF (fnew, c("LATC", "LONC", "GGALT"))
SFC <- vector ("numeric", length (Data$Time))
netCDFfile <- open.ncdf (fnew, write=TRUE)
# have to use a loop here because HeightOfTerrain looks
# up and loads needed files so is not suited to vector ops
for (i in 1:length (Data$Time)) {
  if (is.na (Data$LONC[i]) || is.na (Data$LATC[i])) {
    SFC[i] <- NA
  } else {
    SFC[i] <- HeightOfTerrain (Data$LATC[i], Data$LONC[i])
  }
}

# replace missing values with interpolated values for gaps up to 10 s in length:
SFC <- zoo::na.approx (SFC, maxgap=10, na.rm = FALSE)
SFC[is.na(SFC)] <- 0      # replace missing values with zero; mostly ocean pts
ALTG <- Data$GGALT - SFC
Data["SFC_SRTM"] <- SFC   # add new variable to data.frame
Data["ALTG_SRTM"] <- ALTG
SaveRData <- "NOMADSSterrain.Rdata.gz"
# comment one of these
save(Data, file=SaveRData, compress="gzip")
# load(file=SaveRData)
```

This example was for project NOMADSS and flight rf11. Note in the code that there is a step for interpolating to fill in short periods (up to 10 s) that otherwise would be missing values; that is the 'zoo::' command above. When there is no terrain but only ocean, the dataset did not include lat-long squares for those regions, so there are also missing-value regions over ocean that are not filled in. I think the remaining missing-value regions after interpolation to fill small gaps are mostly over ocean and could be replaced by zero, so for now I have done that to have better appearing plots. The flight track is shown in Fig. 2, and the altitude of the terrain below the aircraft is shown in Fig. 3 for that flight.

```
#SFC[is.na(SFC)] <- 0
#r <- setRange(Data$Time, 82400,85200)
Z <- plotWAC (Data$Time, SFC, ylab="Terrain Elevation [m]")
title (Flight)
```

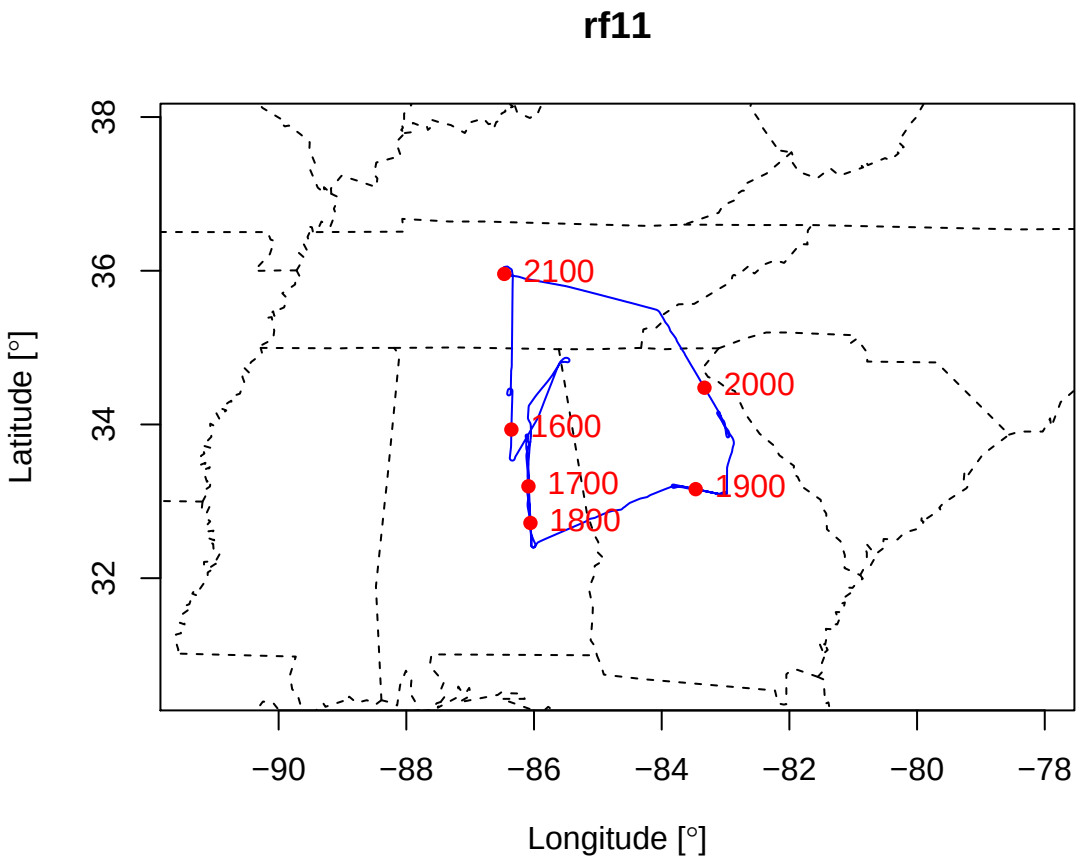

Figure 2: The flight track for NOMADSS flight rf11.

Considerations for routine implementation

The result of this processing is a new netCDF file that duplicates the original except for the addition of two variables, SFC_SRTM and ALTG_SRTM (respectively surface elevation and altitude above the ground). The netCDF file has an identifying 'Z' at the end of the name, in this case NOMADSSrf01Z.nc. (Beware: This program overwrites that file, if present, without warning.) If this is a desirable variable to include in production files, that could be done in two ways. The function used here, 'HeightOfTerrain (lat, long)', with appropriate communication tools can be called from C/C++ programs, so that may be the best way to get this variable into nimbus. Alternately, this program could be run on production files as a second-pass processor to add the variable as I have done here.

Acknowledgements:

Data obtained from http://www.viewfinderpanoramas.org/dem3.html, a database constructed

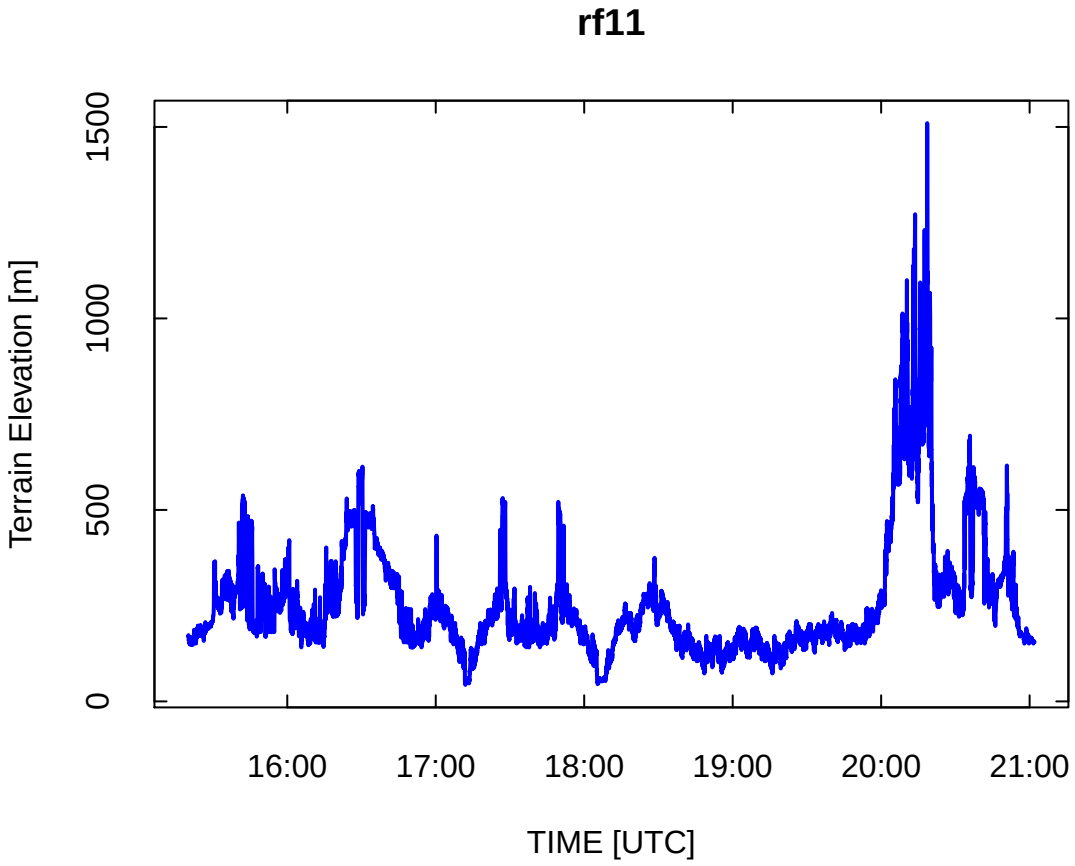

Figure 3: The elevation of the terrain below the position of the aircraft during NOMADSS Flight rf11.

by Jonathan de Ferranti BA, Lochmill Farm, Newburgh, Fife, KY14 6EX, United Kingdom. The analyses reported here were performed using R[4], with RStudio[5] and knitr[6].

– End of Memo –

[4]R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL http://www.R-project.org/

[5]RStudio: Integrated development environment for R (Version 0.98.879) [Computer software]. Boston, MA. Available from http://www.rstudio.org/ (2009)

[6]Xie, Y. (2013), knitr: A general-purpose package for dynamic report generation in R. R package version 1.3. Version 1.6 was used for this work. See also Xie, Y (2014), Dynamic documents with R and knitr, CRC Press, Chapman and Hall, 190 pp.

Reproducibility:

PROJECT:	HeightOfTerrain
ARCHIVE PACKAGE:	HeightOfTerrainNOMADSS.zip
CONTAINS:	attachment list below
PROGRAM:	HeightOfTerrainNOMADSS.Rnw
ORIGINAL DATA:	/scr/raf_data/NOMADSS/NOMADSSrf11.nc
GIT:	git@github.com:WilliamCooper/HeightOfTerrain.git

Attachments: HeightOfTerrainNOMADSS.Rnw
HeightOfTerrainNOMADSS.pdf
NOMADSSterrain.Rdata.gz
SessionInfo