Session 4: R Packages

A sampler; also, 'Ranadu'

Al Cooper

RAF Sessions on R and RStudio

What is a package?

"Base" functions

- Most of what we have been reviewing is in the base package
  Always available, always loaded.
- Many functions, like plot (), are in other standard packages like 'graphics'
- Want to see everything available on CRAN?
  See this CRAN URL; better starting point is this URL

RStudio: see the 'Packages' button:

1. Most are inactive in the sense that they are not using memory or available. To use:
   - (a) check the box;
   - (b) include commands like "require(signal)" or "library(ggplot2)";
   - (c) beanplot::beanplot often useful
2. On barolo, many standard packages are installed. Set .Renviron appropriately (cf. Session 1) for Ranadu and others.

A few to packages to note:

Recently used:

1. ncdf4: basic netCDF functions

R input and response:

```r
## incorporated into Ranadu for
## netCDF access.  Example that
## uses ncdf4:
Data <- getNetCDF(filename)
```

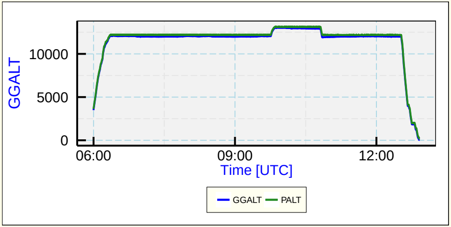
A few to packages to note:

Recently used:

1. ncdf4: basic netCDF functions
2. ggplot2 and ggthemes

R input and response:

```
## 'grammar of graphics' --
## high-quality plots. Used by
## Ranadu 'ggplotWAC()'.
```



A few to packages to note:

Recently used:

1. ncdf4: basic netCDF functions
2. ggplot2 and ggthemes
3. signal (includes filtering)

R input and response:

```
Provides filter functions
including Butterworth and
Savitzgy-Golay. Used in Ranadu.
```

A few to packages to note:

Recently used:

1. ncdf4: basic netCDF functions
2. ggplot2 and ggthemes
3. signal (includes filtering)
4. devtools: helpful constructing and downloading packages

R input and response:

```r
## example: get Ranadu from
## GitHub:
library(devtools)
install_github("WilliamCooper/Ranadu")
```

A few to packages to note:

Recently used:

1. ncdf4: basic netCDF functions
2. ggplot2 and ggthemes
3. signal (includes filtering)
4. devtools: helpful constructing and downloading packages
5. nleqslv: solve non-linear equations

R input and response:

```
## nleqslv::nleqslv() is used by
## several functions in the Ranadu
## package, including those for
## finding the LCL and CAPE and
## the dewpoint from the vapor
## pressure.
```

A few to packages to note:

Recently used:

1. ncdf4: basic netCDF functions
2. ggplot2 and ggthemes
3. signal (includes filtering)
4. devtools: helpful constructing and downloading packages
5. nleqslv: solve non-linear equations
6. knitr: intermix text and R code

R input and response:

```
## This is discussed later in
## connection with 'Reproducible
## Research'. The same program can
## contain the text and the
## processing code.
```

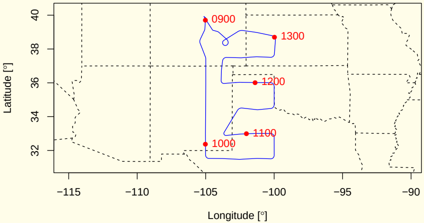
A few to packages to note:

Recently used:

1. ncdf4: basic netCDF functions
2. ggplot2 and ggthemes
3. signal (includes filtering)
4. devtools: helpful constructing and downloading packages
5. nleqslv: solve non-linear equations
6. knitr: intermix text and R code
7. maps

R input and response:

```
## provides a map background
fname = sprintf("%sMPEX/MPEXrf01.nc",
    DataDirectory())
plotTrack(getNetCDF(fname), .Spacing = 60)
```



A few to packages to note:

Recently used:

1. ncdf4: basic netCDF functions
2. ggplot2 and ggthemes
3. signal (includes filtering)
4. devtools: helpful constructing and downloading packages
5. nleqslv: solve non-linear equations
6. knitr: intermix text and R code
7. maps
8. shiny: interactive apps

R input and response:

```
## This tutorial is a shiny app.
## The construction of these apps
## requires the 'shiny' package.
## This is the topic of a later
## tab.
```

A few to packages to note:

Recently used:

1. ncdf4: basic netCDF functions
2. ggplot2 and ggthemes
3. signal (includes filtering)
4. devtools: helpful constructing and downloading packages
5. nleqslv: solve non-linear equations
6. knitr: intermix text and R code
7. maps
8. shiny: interactive apps
9. zoo::na.approx for interpolation

R input and response:

```
## short periods with missing
## values can be replaced by
## interpolation. This is used by
## the Ranadu::smoothInterp()
## routine.
```

'Ranadu'

Data-access functions:

Data <- getNetCDF ( ): loads data.frame with requested variables
V <- standardVariables ( ): defines a common set
DataDirectory ( ): "/scr/raf_data/" on barolo
i <- getIndex ( ): find index for a specified time
r <- setRange ( ): set a range of indices to a specified time interval
TellAbout (V): lists some characteristics of V

R code and response:

```
Project <- "DEEPWAVE"
Flight <- "rf15"
fname <- sprintf("%s%s/%s%s.nc", DataDirectory(), Project,
    Project, Flight)  # or fname <- '...'
Data <- getNetCDF(fname, standardVariables(c("GGALT", "PITCH")),
    Start = 40000, End = 53000, F = 15)  # loads data.frame
names(Data)  # shows variables in Data
 [1] "Time"   "ATX"    "DPXC"   "EWX"    "GGALT"  "LATC"   "LONC"   "MACHX"
 [9] "MR"     "PALT"   "PSXC"   "QCXC"   "TASX"   "WDC"    "WSC"    "WIC"
[17] "PITCH"  "RF"
```

'Ranadu'

R code and response:

```
TellAbout(Data)
[1] "Variable class is data.frame, length = 18, dim = "
[2] "5401"
[3] "18"
      Time                          ATX              DPXC
 Min.   :2014-07-03 04:00:00   Min.   :-55.67   Min.   :-63.12
 1st Qu.:2014-07-03 04:22:30   1st Qu.:-54.48   1st Qu.:-61.02
 Median :2014-07-03 04:45:00   Median :-31.59   Median :-50.41
 Mean   :2014-07-03 04:45:00   Mean   :-38.89   Mean   :-50.40
 3rd Qu.:2014-07-03 05:07:30   3rd Qu.:-30.35   3rd Qu.:-40.83
 Max.   :2014-07-03 05:30:00   Max.   :-12.03   Max.   :-20.51
      EWX              GGALT          LATC             LONC
 Min.   :0.01239   Min.   :2929   Min.   :-45.94   Min.   :170.7
 1st Qu.:0.01633   1st Qu.:5767   1st Qu.:-45.40   1st Qu.:171.7
 Median :0.06023   Median :5774   Median :-44.71   Median :172.4
 Mean   :0.10355   Mean   :6729   Mean   :-44.68   Mean   :172.4
 3rd Qu.:0.17339   3rd Qu.:8693   3rd Qu.:-43.88   3rd Qu.:173.3
 Max.   :1.20097   Max.   :8817   Max.   :-43.45   Max.   :173.8
     MACHX              MR               PALT           PSXC
 Min.   :0.4112   Min.   :0.01808   Min.   :3170   Min.   :295.7
```

More about getNetCDF ( ):

1. The first variable returned is “Time”. This is converted from the time variable used in netCDF files (seconds after a specified reference time) to 'POSIX'-format time that is understood by R.
   (a) Gives appropriate labels in plots vs time.
   (b) Includes date; no ambiguity if data.frames are merged.
   (c) Requires interpretation; not a simple index. This works: Data$ATX[Data$Time==as.POSIXct("2014-07-04 08:33:19", tz='UTC')]
   – but see 'getIndex', an easier way to reference one time
2. Handles high-rate files by returning 25 values per second in flat arrays. Where variables are lower rate, interpolation is used, Savitzky-Golay with 4th-order polynomials spanning 3 s centered on each 25-Hz point, so all variables are 25-Hz.
3. Data$RF is included to be able to merge resulting files and still identify data from individual flights: Data[RF==15,] gives only measurements from that flight.

(not-Ranadu) Ways of getting data into R: tables

read.table ()

- Easy way to read data in text spreadsheet form:
  export from Excel in CSV format;
  read.table with the same separator as the argument
- other options include 'header' and 'skip'
- The 'file' argument can also be a complete URL. This URL (modified to select the latest time) will download the current Denver sounding as a data.frame.

```
Names <- read.table(file = URL_UW, skip = 7, nrows = 1)
A <- read.table(file = URL_UW, skip = 13, nrows = 70,
    col.names = as.vector(t(Names)))  ## loads data.frame
head(A)  ## prints top of the data.frame
##    PRES HGHT TEMP  DWPT RELH MIXR DRCT SKNT  THTA  THTE  THTV
## 1 846.0 1625 -0.3 -17.3   26 1.17  185    5 286.2 289.9 286.4
## 2 842.0 1663  2.0 -17.0   23 1.21  198    5 289.0 292.8 289.2
## 3 837.0 1711  9.2 -18.8   12 1.04  215    4 297.1 300.5 297.3
## 4 825.1 1829 10.7 -20.0   10 0.95  255    3 299.8 303.0 300.0
## 5 819.0 1890 11.4 -20.6    9 0.91  245    3 301.3 304.3 301.4
## 6 794.7 2134  9.6 -20.9   10 0.92  205    3 301.9 305.0 302.1
```

(not-Ranadu) Ways of getting data into R: HTML pages

readHTMLTable(URL, ...)
Example: RTD schedule, route 228 southbound at the RAF hangar:

```
suppressMessages(require(XML))
Schedule <- readHTMLTable(U, which = 1, skip.rows = 1:5)
names(Schedule) <- c("Stop1", "2", "3", "4", "5", "6", "7",
    "(RAF)", "BPNR1", "BPNR2", "BPNR3", "BPNR4")
head(Schedule[, 8:12], 11)
   (RAF) BPNR1 BPNR2 BPNR3 BPNR4
1   842A    --  852A    --    --
2   915A    --  925A    --    --
3  1018A    --    --    -- 1028A
4  1118A    --    --    -- 1128A
5  1218P    --    --    -- 1228P
6   118P    --    --    --  128P
7   218P    --    --    --  228P
8   317P    --    --    --  327P
9   350P  400P    --    --    --
10  420P    --    --    --  430P
11  450P    --    --    --  500P
```

Ranadu Algorithm Functions (?Ranadu for full list)

Available in Ranadu: (learn more via ?Ranadu::xxx xxx=function)

MurphyKoop (DP, P)
DPfromE (E)
MixingRatio
PotentialTemperature
EquivalentPotentialTemperature
WetEquivalentPotentialTemperature
VirtualTemperature
VirtualPotentialTemperature
MachNumber
TrueAirspeed
PCorFunction
KingProbe
AdiabaticTandLWC
memCoef/memEval

AirTemperature
calcAttack
GV_AOAfromRadome
GV_YawFromRadome
ButterworthFilter
ComplementaryFilter
Gravity
PressureAltitude
RecoveryFactor
SpecificHeats
StandardConstant
CAPE/LCL
WindProcessor

Convenience and Special Functions:

Now available:

DataDirectory ( )
GetAttributes (V)
getIndex (Time, HHMMSS)
r <- setRange (Time, Start, End)
getRAFData ()
getStartEnd(Time)
ncsubset ( )
binStats ( )
TellAbout (V)
ValueOf ( )
ValueOfAll ( )

Special (available):

DemingFit ( )
AdiabaticTandLWC ( )
Ranadu shiny app

Plotting routines (available):

plotWAC ( )
ggplotWAC ()
lineWAC ( )
theme_WAC ( )
plotTrack ( )
skew-T based on Davies-Jones
pseudo-adiabatic lines
Paluch and Betts plots

Development projects:

size distributions: CDP etc.
2D image display
(both available in the shiny app)

MORE INFORMATION re Ranadu

Standard help functions:

?Ranadu::getNetCDF
?Ranadu::ggplotWAC
?Ranadu::Ranadu
etc

The manuals for Ranadu and the Ranadu Shiny App

1. See the manuals in the directory specified by the R function path.package('Ranadu').
2. See the versions on GitHub at this URL: "https://github.com/WilliamCooper/Ranadu/blob/master/inst/RanaduManual.pdf" and ".../RanaduShinyManual.pdf"

Examples using Ranadu

Study These as Guides

Function illustrated:

1. Load the library

R code and result:

```
library(Ranadu)
```

Examples using Ranadu

Study These as Guides

Function illustrated:

1. Load the library
2. Construct a data.frame from a netCDF file

R code and result:

```r
Project <- "DEEPWAVE"
Flight <- 16
fileName = sprintf("%s%s/%srf%02d.nc",
    DataDirectory(), Project, Project,
    Flight)
varNeeded <- c("ATHR1", "ATHR2", "ATRL")
Data <- getNetCDF(fileName, varNeeded)
names(Data)
## [1] "Time"  "ATHR1" "ATHR2" "ATRL"
## try also str(Data) to see that the
## original attributes of the variables
## are preserved.
```

Examples using Ranadu

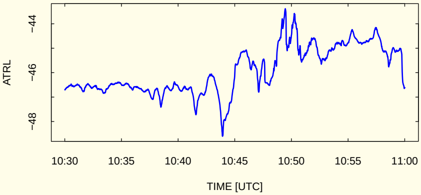
Study These as Guides

Function illustrated:

1. Load the library
2. Construct a data.frame from a netCDF file
3. Restrict the time range

R code and result:

```
DataR <- Data[setRange(Data, 103000,
    110000), ]  ## 10:30--11:00
plotWAC(DataR[, c("Time", "ATRL")])
```



Examples using Ranadu

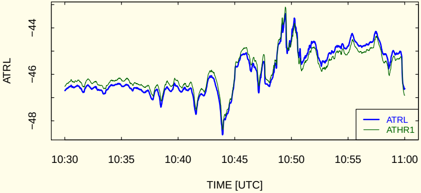
Study These as Guides

Function illustrated:

1. Load the library
2. Construct a data.frame from a netCDF file
3. Restrict the time range
4. Another simple plot

R code and result:

```
with(DataR, plotWAC(data.frame(Time,
    ATRL, ATHR1)))  ## another way to plot
```



Examples using Ranadu

Study These as Guides

Function illustrated:

1. Load the library
2. Construct a data.frame from a netCDF file
3. Restrict the time range
4. Another simple plot
5. Adding a new variable

R code and result:

```
## This is a reference value used for
## fitting expressions to represent
## angle-of-attack. This would be
## angle-of-attack if the vertical
## wind were zero.
Data$AOAREF <- Data$PITCH
  - (Data$GGVSPD/Data$TASX) * (180/pi)
```

Examples using Ranadu

Study These as Guides

Function illustrated:

1. Load the library
2. Construct a data.frame from a netCDF file
3. Restrict the time range
4. Another simple plot
5. Adding a new variable
6. Fitting

R code and result:

```
## data.frame Data previously built
## with PITCH, GGVSPD, TASX, ADIFR,
## QCF, AOAREF as needed here.
AOAfit <- lm(AOAREF ~ I(ADIFR/QCF),
    data = Data)
coefficients(AOAfit)
##  (Intercept) I(ADIFR/QCF)
##     4.339473    20.481498
summary(AOAfit)$sigma  ## residual error
## [1] 0.2959565
```

Examples using Ranadu

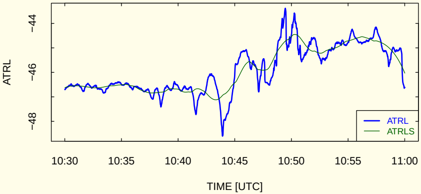
Study These as Guides

Function illustrated:

1. Load the library
2. Construct a data.frame from a netCDF file
3. Restrict the time range
4. Another simple plot
5. Adding a new variable
6. Fitting
7. Smoothing/interpolation

R code and result:

```r
## smooth, window length 300 s
DataR$ATRLS <- SmoothInterp(DataR$ATRL,
    .Length = 300)
plotVar <- c("Time", "ATRL", "ATRLS")
plotWAC(DataR[, plotVar])
```



Examples using Ranadu

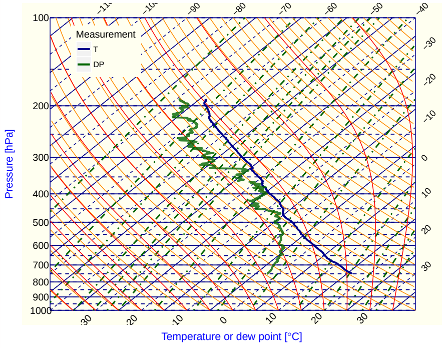
Study These as Guides

Function illustrated:

1. Load the library
2. Construct a data.frame from a netCDF file
3. Restrict the time range
4. Another simple plot
5. Adding a new variable
6. Fitting
7. Smoothing/interpolation
8. A skew-T sounding

R code and result:

```
## data from climb loaded into DS
with(DS, SkewTSounding(Pressure = PSXC,
    Temperature = ATX, DewPoint = DPXC))
```



Examples using Ranadu

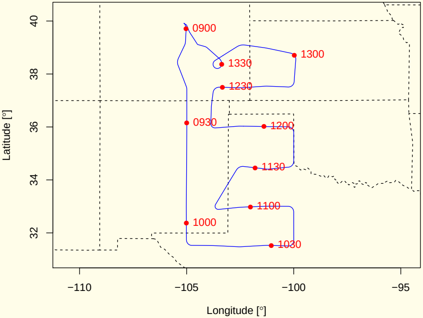
Study These as Guides

Function illustrated:

1. Load the library
2. Construct a data.frame from a netCDF file
3. Restrict the time range
4. Another simple plot
5. Adding a new variable
6. Fitting
7. Smoothing/interpolation
8. A skew-T sounding
9. A flight track

R code and result:

```
plotTrack(getNetCDF(fname), .Spacing = 30)
```



Examples using Ranadu

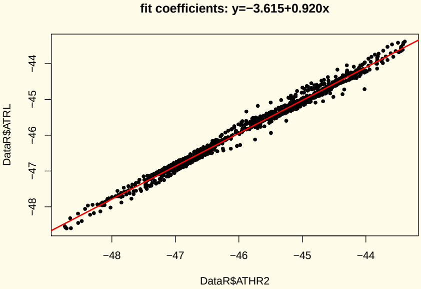
Study These as Guides

Function illustrated:

1. Load the library
2. Construct a data.frame from a netCDF file
3. Restrict the time range
4. Another simple plot
5. Adding a new variable
6. Fitting
7. Smoothing/interpolation
8. A skew-T sounding
9. A flight track
10. A Deming fit

R code and result:

```
## This fit minimizes the
## perpendicular distance from the
## fitted line to the measurements.
## Compare to lm() regression.
Dfit <- DemingFit(DataR$ATHR2, DataR$ATRL)
bestFit <- Dfit[1] + -50:-0 * Dfit[2]
plot(DataR$ATHR2, DataR$ATRL, pch = 20)
lines(-50:0, bestFit, lwd = 2, col = "red")
```



Examples using Ranadu

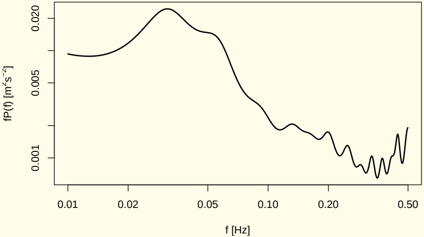
Study These as Guides

Function illustrated:

1. Load the library
2. Construct a data.frame from a netCDF file
3. Restrict the time range
4. Another simple plot
5. Adding a new variable
6. Fitting
7. Smoothing/interpolation
8. A skew-T sounding
9. A flight track
10. A Deming fit
11. A variance spectrum

R code and result:

```
## Using the MEM functions in Ranadu
Data <- getNetCDF(fname, "WIC", 110000,
    120000)  ## 1 h vertical wind
MEM <- memCoef(Data$WIC, .poles = 30)
frq <- 10^(seq(-2, log10(0.5), by = 0.001))
Pmem <- Mod(memEstimate(frq, MEM))^2
plot(frq, 2 * Pmem * frq, type = "l",
    log = "xy", lwd = 2, xlab = "f [Hz]",
    ylab = expression(paste("fP(f) [m"^"2",
        "s"^"-2", "]")))
```



Next Tab: More Detailed Information re Plotting