Exercise 3: Tools for big data

The objective of today's exercise is to provide a quick introduction to some common tools for dealing with big data. For each tool we are just using the most basic syntax and you are encouraged to go back and read the help for each at a later date. This exercise also focuses on "general purpose" tools. There are a multitude of R libraries available for accessing specific data sources and web services. A quick summary of some of these is available at http://cran.r-project.org/web/views/WebTechnologies.html. In addition, a Google search on many of the tools and topics covered in Chapters 3 and 4 will provide a lot of additional info on big data tools outside of R.

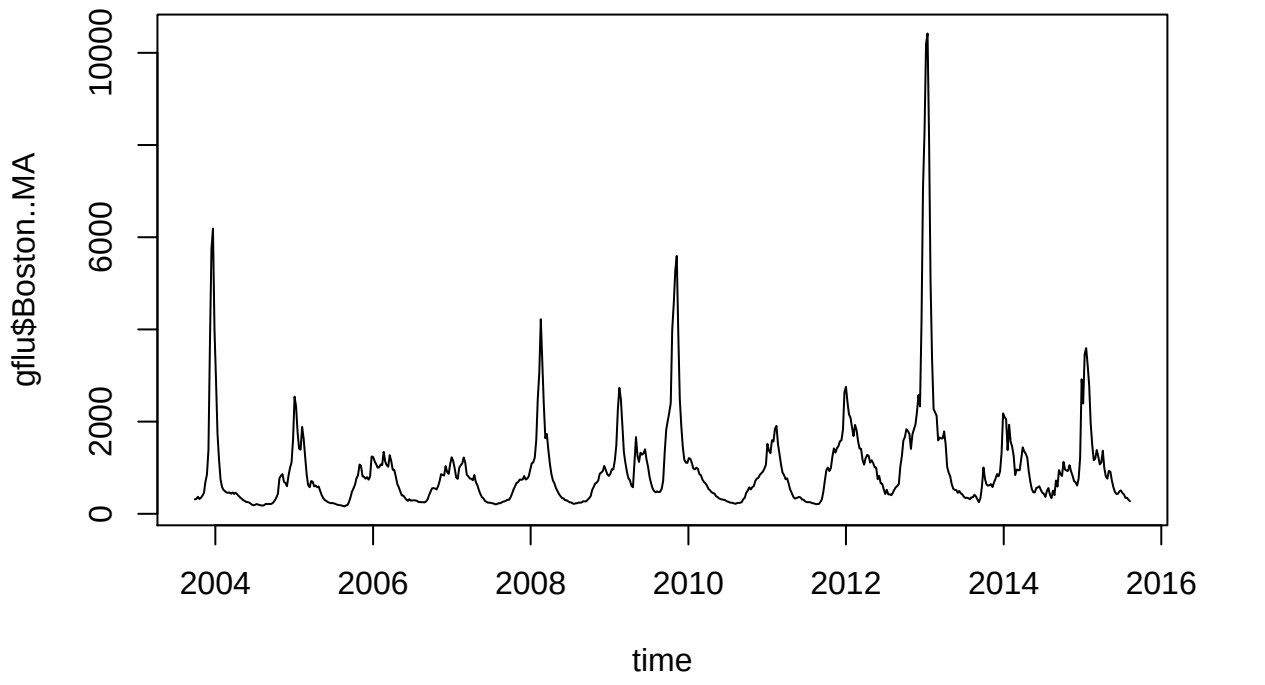
Note: The code in this exercise will download data off the web dynamically, which can take some time, so try to "knit" infrequently.

```
## Loading required package: RCurl
## Loading required package: bitops
## Loading required package: XML
## Loading required package: ncdf4
## Loading required package: devtools
## Loading required package: MODISTools
```

Pulling data directly off the web

In the previous exercises we loaded data into R using functions like read.csv. However, it is also possible to read data into R directly off the web by passing a web address to the file name. For smaller files that are quick to load this approach can ensure that the script is always operating with the most up-to-date version of a data file.

```
gflu = read.csv("http://www.google.org/flutrends/about/data/flu/us/data.txt",skip=11)
time = as.Date(gflu$Date)
plot(time,gflu$Boston..MA,type='l')
```



That said, for publication purposes it is usually important to save the data that you used for an analysis, and that the date of access is recorded (and version number if available), as some datasets are subject to frequent revision.

In this example, the file in question has an extensive header, which we skip during the load of the data, but as with any dataset, this metadata is important to read before using the data.

```
Google Flu Trends - United States
Copyright 2013 Google Inc.

Exported data may be used for any purpose, subject to the Google Terms of Service (http://www.google.co
If you choose to use the data, please attribute it to Google as follows: "Data Source: Google Flu Trend

Each week begins on the Sunday (Pacific Time) indicated for the row.
Data for the current week will be updated each day until Saturday (Pacific Time).
Note: To open these files in a spreadsheet application, we recommend you save each text file as a CSV s
For more information, please visit http://www.google.org/flutrends
```

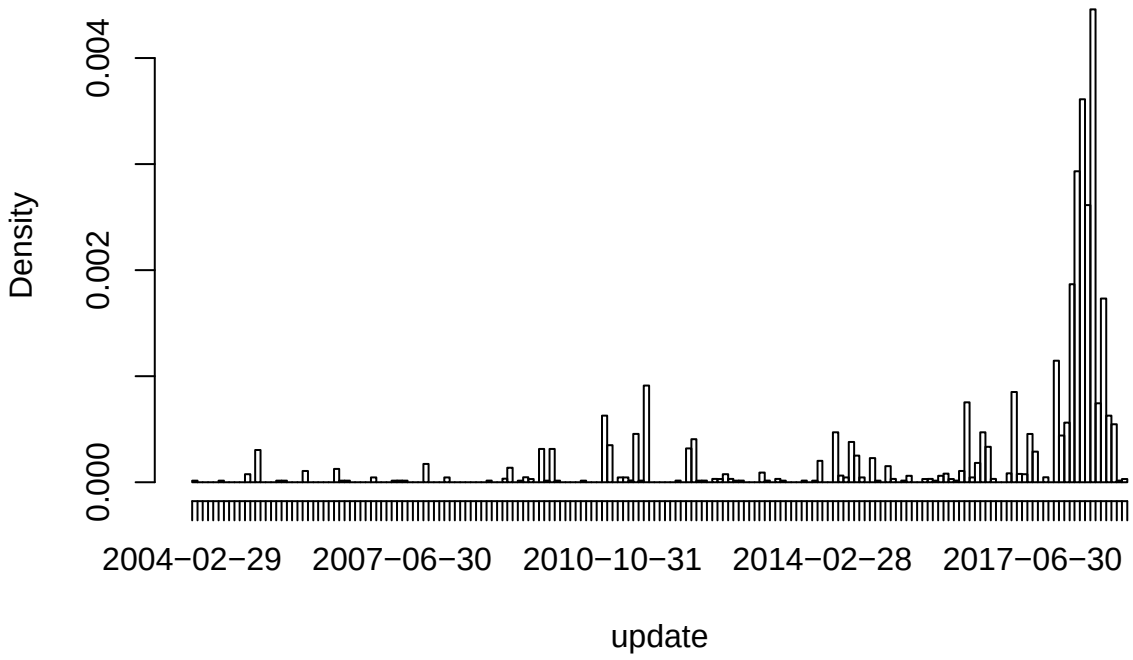
Question 1:

Using the US Forest Service's Forest Inventory and Analysis (FIA) data set, plot the rank vs log(abundance) curve for tree seedling counts from Rhode Island. Data is available at https://apps.fs.usda.gov/fia/datamart/CSV/RI_SEEDLING.csv and the relevant columns are TREECOUNT (raw seedling counts) and SPCD (species codes). Hints: tapply, sum, na.rm=TRUE, sort, decreasing=TRUE, log='y'

Web Scraping

Often the data that we want to use from the web has been formatted in HTML for human-readability rather than in tab- or comma-delimited files for inport and export. The process of extracting data from webpages has been dubbed **scraping**. For these sorts of more complex problems we can use the RCurl library to grab HTML or XML structured content directly off the web, and then use the XML library to parse the markup into more standard R data objects. In the example below we grab data on the status of all the files that make up the FIA in order to look for files that have been updated after a certain date.

```
nu <- function(x){as.numeric(as.character(x))}  ## simple function to convert data to numeric

fia_html <- getURL("https://apps.fs.usda.gov/fia/datamart/CSV/datamart_csv.html")  ## grab raw html
fia_table = readHTMLTable(fia_html)[[3]]    ## We're interested in the 3rd table on this webpage
update = as.Date(fia_table[,"Last Modified Date"])
hist(update,"months")                       ## Plot a histogram of update times
```




```
recent <- fia_table[which(update > "2018/01/01"),]
```

Question 2: Create a sorted table of how many FLUXNET eddy-covariance towers are in each country according to the website at http://fluxnet.fluxdata.org/sites/site-list-and-pages/. Hint: use substring to extract the country code from the overall FLUXNET ID code.

grep, system, RegExp

`grep` is a handy little *command prompt* function that returns lines from a file that match a search string. I continue to use this 'old school' utility on a daily basis to help manage code and data because this simple little search continues to be able to perform actions that elude newer search software:

- `grep` looks within files, but is able to search across file and recursively within a directory structure. I use this constantly to follow variables or functions through complex code. For example, if I wanted to find uses of the term *fia* in my current directory and all subdirectories I could type

```
grep -ir "fia" .
```

here the -i means ignore case when searching, the -r means to search recursively through subdirectories, and the `.` means to start from the current directory. Used in this way grep can help you quickly find your way through new and complex code, iteratively hopping through the code from one search to another. It is also extremely helpful in debugging, when a program returns a cryptic error message and you want to find *where* in the code that message was generated.

- `grep` returns the full lines/rows that match a search, allowing one to quickly and easily subset large datasets into smaller files and/or merge subsets across different files.
- `grep` supports **Regular Expressions**, both within the search itself and in the set of filenames searched. For example, if we wanted to find all lines that contain 'fia', in all the `.Rmd` files in the current directory we could type

```
grep -ir 'fia' *.Rmd
```

where the * means 'match zero or more occurances of any character', in this case preceeding .Rmd (the zero part means this would match a file just named .Rmd). If I just wanted to find instances where `fia` is at the start of the line I could use the ^ to indicate the beginning of the line

```
grep -ir '^fia' *.Rmd
```

If I instead wanted just instances where `fia` is followed immediately by another letter I could use [a-z] to match just letters in the English alphabet.

```
grep -ir 'fia[a-z]' *.Rmd
```

or I could be more specific an just look for specific letters, e.g. fia[fds] would match fiaf, fiad, and fias. A full description of regular expressions is beyond the scope of this tutorial, and RegExp statements for matching complex patterns can quickly become cryptic, so following up on this further is left to the reader.

There are often times when working in R that one needs to run another command, script, or piece of software that is external to R. If I'm working in an R script want the operating system to run a command I can do this with the `system` command

```
system('grep -ir "fia" *.Rmd')
```

Furthermore, often we want to capture the output of that command directly into R, which we can do using the `intern` flag:

```
fia.lines = system('grep -ir "fia" *.Rmd',intern=TRUE)
fia.lines[1:3]
```

```
## [1] "Exercise_03_BigData.Rmd:Using the US Forest Service's Forest Inventory and Analysis (FIA) data s
## [2] "Exercise_03_BigData.Rmd:Often the data that we want to use from the web has been formatted in HT
## [3] "Exercise_03_BigData.Rmd:fia_html <- getURL(\"https://apps.fs.usda.gov/fia/datamart/CSV/datamart_
```

Finally, it is also worth mentioning that R has its own, internal, version of grep that can be useful for searching and subsetting data and which also supports RegExp. Unlike the command-line version of grep, this function returns the row numbers matching the search string. In the example below we use the function readLines to read unstructured text in as vector of strings, one corresponding to each row in a file. It also demonstrates the function `sub`, which is related to grep but which substitutes the matching string rather than just finding it.

```
myCode = readLines("Exercise_03_BigData.Rmd")  ## read unstructured text
x = grep("RI",myCode)    ## returns the line numbers that include the string 'RI'
myCode[x]
```

```
## [1] "Using the US Forest Service's Forest Inventory and Analysis (FIA) data set, plot the rank vs log
## [2] "x = grep(\"RI\",myCode)    ## returns the line numbers that include the string 'RI'"
## [3] "sub(\"RI\",\"VT\",myCode[x])  ## substitute FIRST: VT for RI"
## [4] "gsub(\"RI\",\"VT\",myCode[x])  ## substitute ALL: VT for RI"
```

```
sub("RI","VT",myCode[x])  ## substitute FIRST: VT for RI
```

```
## [1] "Using the US Forest Service's Forest Inventory and Analysis (FIA) data set, plot the rank vs log
## [2] "x = grep(\"VT\",myCode)    ## returns the line numbers that include the string 'RI'"
## [3] "sub(\"VT\",\"VT\",myCode[x])  ## substitute FIRST: VT for RI"
## [4] "gsub(\"VT\",\"VT\",myCode[x])  ## substitute ALL: VT for RI"
```

```
gsub("RI","VT",myCode[x])  ## substitute ALL: VT for RI
```

```
## [1] "Using the US Forest Service's Forest Inventory and Analysis (FIA) data set, plot the rank vs log
## [2] "x = grep(\"VT\",myCode)    ## returns the line numbers that include the string 'VT'"
## [3] "sub(\"VT\",\"VT\",myCode[x])  ## substitute FIRST: VT for VT"
## [4] "gsub(\"VT\",\"VT\",myCode[x])  ## substitute ALL: VT for VT"
```

Question 3: Within the object myCode, find all the lines that begin with the comment character, #.

netCDF, wget

In this section I want to introduce another command-line utility, wget, which can be used to pull files and content off the web, and to demonstrate how netCDF can be used in R. For this example we will be using data from the WLEF eddy-covariance tower located in northern Wisconsin. Unlike most flux towers, WLEF is a "tall-tower" – it's actually a 440m TV antenna – which means that it integrates over a much larger footprint than most towers. Indeed, the tower is instrumented at multiple heights. First, let's use wget to grab the data off the web. A few notes: 1) wget could be used from command line rather than as a system command; 2) if you don't have wget installed, use your web browser

```
#system("wget http://flux.aos.wisc.edu/data/cheas/wlef/netcdf/US-PFa-WLEF-TallTowerClea#n-2012-L0-vFeb2
# I could not make this line working, so I manually donwloaded the netCDF data.
```

Next, lets open the file and look at what it contains

```
## open the netCDF file
wlef = nc_open("US-PFa-WLEF-TallTowerClean-2012-L0-vFeb2013.nc")
print(wlef)    ## metadata
```

```
## File US-PFa-WLEF-TallTowerClean-2012-L0-vFeb2013.nc (NC_FORMAT_CLASSIC):
##
##      47 variables (excluding dimension variables):
##         char Flag_lvl[level1]
##             units: N/A
##             long_name: Types of quality control
##         float M_lvl[level2]
##             units: meters
##             long_name: Heights for measurements
##         float CO2_lvl[level3]
##             units: meters
##             long_name: Heights for CO2 concentration measurements
##         int MONTH[time]
##             description: Calendar month
##             missing_value: -999
##             resolution: 1
##             valid_delta: 1
##             valid_max: 12
##             valid_min: 1
##             units: Calendar month
##             long_name: Calendar month
##             valid_range: 1, 12
##         int DOY[time]
##             description: Calendar Day
##             missing_value: -999
##             resolution: 1
##             valid_delta: 1
##             valid_max: 366
##             valid_min: 1
##             units: Calendar Day
##             long_name: Calendar Day
##             valid_range: 1, 366
##         float LST[time]
##             description: Local Standard Time
```

```
##             missing_value: -999
##             resolution: 0.100000001490116
##             valid_delta: 0.100000001490116
##             valid_max: 24
##             valid_min: 0
##             units: hour
##             long_name: Local Standard Time
##             valid_range: 0, 24
##         double FoD[time]
##             description: Fractional Julian day
##             missing_value: -999
##             resolution: 0.100000001490116
##             valid_delta: 10
##             valid_max: 367
##             valid_min: 0
##             units: Fractional Julian day
##             long_name: Fractional Julian day
##             valid_range: 0.0, 367.0
##         double FoY[time]
##             description: Fraction of Year
##             missing_value: -999
##             resolution: 0.100000001490116
##             valid_delta: 0.100000001490116
##             valid_max: 3000
##             valid_min: 0
##             units: Fraction of Year
##             long_name: Fraction of Year
##             valid_range: 0.0, 3000.0
##         double NEE_co2p[time]
##             description: Preferred NEE of CO2
##             long_name: Preferred NEE of CO2
##             units: umol/m2/s
##             valid_min: 0
##             valid_max: 100
##             missing_value: -999
##         double LE[time]
##             description: Preferred latent heat flux
##             missing_value: -999
##             resolution: 0.100000001490116
##             valid_max: 75
##             valid_min: -75
##             units: W/m2
##             long_name: Latent heat flux
##         double H[time]
##             description: Preferred Sensible heat flux
##             missing_value: -999
##             resolution: 0.100000001490116
##             valid_max: 75
##             valid_min: -75
##             units: W/m2
##             long_name: Sensible  heat  flux
##         double ustar_p[time]
##             description: Preferred ustar
##             long_name: ustar
```

```
##             units: m/s
##             valid_min: 0
##             valid_max: 100
##             missing_value: -999
##         double NEE_ch4[time]
##             description: NEE of CH4 at 122 m (WPL-corrected)
##             long_name: NEE of CH4 at 122 m (WPL-corrected)
##             units: umol/m2/s
##             valid_min: 0
##             valid_max: 100
##             missing_value: -999
##         double PAR[time]
##             description: Incoming photosynthetic active radiation
##             missing_value: -999
##             resolution: 0.100000001490116
##             valid_delta: 10
##             maximum: 100
##             minimum: -999
##             units: umol/m2/s
##             long_name: Photosynthetic active radiation
##             valid_range: -999
##              valid_range: 100
##         double Ta[time]
##             description: Preferred air temperature at 30 m
##             missing_value: -999
##             resolution: 0.100000001490116
##             valid_delta: 10
##             valid_max: 100
##             valid_min: -100
##             units: degrees C
##             long_name: Air temperature
##             valid_range: -100.0, 100.0
##         double WMR_p[time]
##             description: Preferred water vapor mixing ratio at 30 m
##             missing_value: -999
##             resolution: 0.100000001490116
##             valid_delta: 10
##             maximum: 100
##             minimum: -999
##             units: g/kg
##             long_name: Water vapor mixing ratio
##             valid_range: -999
##              valid_range: 100
##         double VPD[time]
##             description: Preferred vapor pressure deficit at 30 m
##             missing_value: -999
##             resolution: 0.100000001490116
##             valid_delta: 10
##             maximum: 100
##             minimum: -999
##             units: Pa
##             long_name: Vapor pressure deficit
##             valid_range: -999
##              valid_range: 100
```

```
##         double WS_p[time]
##             description: Preferred wind speed at 30m
##             long_name: Wind speed at 30m
##             units: m/s
##             valid_min: 0
##             valid_max: 100
##             missing_value: -999
##         double WD_p[time]
##             description: Preferred wind direction at 30m
##             long_name: Wind direction
##             units: degrees
##             valid_min: 0
##             valid_max: 100
##             missing_value: -999
##         double PREC[time]
##             description: Precipitation
##             missing_value: -999
##             resolution: 0.100000001490116
##             valid_delta: 10
##             maximum: 100
##             minimum: -999
##             units: mm
##             long_name: Precipitation
##             valid_range: -999
##              valid_range: 100
##         double SWC[time]
##             description: Near surface soil moisture by volume
##             missing_value: -999
##             resolution: 0.100000001490116
##             valid_delta: 10
##             maximum: 100
##             minimum: -999
##             units: percent
##             long_name: Soil moisture by volume
##             valid_range: -999
##              valid_range: 100
##         double Pa[time]
##             description: Surface air pressure, not sea-level corrected
##             missing_value: -999
##             resolution: 0.100000001490116
##             valid_delta: 10
##             maximum: 100
##             minimum: -999
##             units: kPa
##             long_name: Surface air pressure
##             valid_range: -999
##              valid_range: 100
##         int QcFc[level1,time]
##             description: Quality indicator for NEE_co2, LE, H, ustar, and NEE_ch4
##             long_name: Indicator of quality of flux measurements
##             units: unitless
##             valid_min: -1
##             valid_max: 4
##             missing_value: -999
```

```
##         double NEE_f[time]
##             description: Gap filled NEE_CO2
##             long_name: Net ecosystem exchange of CO2
##             units: umol/m2/s
##             valid_min: 0
##             valid_max: 100
##             missing_value: -999
##         double Reco[time]
##             description: Ecosystem respiration
##             long_name: Ecosystem respiration
##             units: umol/m2/s
##             valid_min: 0
##             valid_max: 100
##             missing_value: -999
##         double GPP[time]
##             description: Gross primary production
##             long_name: Gross primary production
##             units: umol/m2/s
##             valid_min: 0
##             valid_max: 100
##             missing_value: -999
##         double NEE_m[time]
##             description: Modeled NEE
##             long_name: Modeled NEE
##             units: umol/m2/s
##             valid_min: 0
##             valid_max: 100
##             missing_value: -999
##         double Sc[level2,time]
##             description: Storage of CO2 at 30 m
##             long_name: Storage of CO2
##             units: umol/m2/s
##             valid_min: 0
##             valid_max: 100
##             missing_value: -999
##         double Fc[level2,time]
##             description: Turbulent flux of CO2 at 30 m
##             long_name: Turbulent flux of CO2 at 30 m
##             units: umol/m2/s
##             valid_min: 0
##             valid_max: 100
##             missing_value: -999
##         double NEE_co2[level2,time]
##             description: Net ecosystem exchange of CO2 at 30 m
##             long_name: Net ecosystem exchange of CO2
##             units: umol/m2/s
##             valid_min: 0
##             valid_max: 100
##             missing_value: -999
##         double Sh2o[level2,time]
##             description: Storage of H2O at 30 m
##             long_name: Storage of H2O
##             units: W/m2
##             valid_min: 0
```

```
##             valid_max: 100
##             missing_value: -999
##         double Fh2o[level2,time]
##             description: Turbulent flux of H2O at 30 m
##             long_name: Turbulent flux of H2O at 30 m
##             units: W/m2
##             valid_min: 0
##             valid_max: 100
##             missing_value: -999
##         double NEE_h2o[level2,time]
##             description: Net ecosystem exchange of H2O
##             long_name: Net ecosystem exchange of H2O
##             units: W/m2
##             valid_min: 0
##             valid_max: 100
##             missing_value: -999
##         double St[level2,time]
##             description: Storage of heat
##             long_name: Storage of heat
##             units: W/m2
##             valid_min: 0
##             valid_max: 100
##             missing_value: -999
##         double Ft[level2,time]
##             description: Turbulent flux of heat
##             long_name: Turbulent flux of heat
##             units: W/m2
##             valid_min: 0
##             valid_max: 100
##             missing_value: -999
##         double NEE_heat[level2,time]
##             description: Net ecosystem exchange of heat
##             long_name: Net ecosystem exchange of heat
##             units: W/m2
##             valid_min: 0
##             valid_max: 100
##             missing_value: -999
##         double ustar[level2,time]
##             description: Friction velocity
##             long_name: Friction velocity
##             units: m/s
##             valid_min: 0
##             valid_max: 100
##             missing_value: -999
##         double Sch4[time]
##             description: Storage of CH4 at 122 m
##             long_name: Storage of CH4 at 122 m
##             units: umol/m2/s
##             valid_min: 0
##             valid_max: 100
##             missing_value: -999
##         double Fch4[time]
##             description: Turbulent flux of CH4 at 122 m
##             long_name: Turbulent flux of CH4 at 122 m
```

```
##             units: umol/m2/s
##             valid_min: 0
##             valid_max: 100
##             missing_value: -999
##         double CO2[level3,time]
##             description: CO2 concentration at different heights
##             long_name: CO2 concentration
##             units: ppm
##             valid_min: 0
##             valid_max: 100
##             missing_value: -999
##         double CH4[level2,time]
##             description: CH4 concentration at different heights
##             long_name: CH4 concentration
##             units: W/m2
##             valid_min: 0
##             valid_max: 100
##             missing_value: -999
##         double TA[level2,time]
##             description: Air temperature at different heights
##             long_name: Air temperature
##             units: degrees C
##             valid_min: 0
##             valid_max: 100
##             missing_value: -999
##         double WMR[level2,time]
##             description: Water vapor mixing ratio at different heights
##             long_name: Water vapor mixing ratio
##             units: g/kg
##             valid_min: 0
##             valid_max: 100
##             missing_value: -999
##         double WS[level2,time]
##             description: Wind speed at different heights
##             long_name: Wind speed
##             units: m/s
##             valid_min: 0
##             valid_max: 100
##             missing_value: -999
##         double WD[level2,time]
##             description: Wind direction at different heights
##             long_name: Wind direction
##             units: degrees
##             valid_min: 0
##             valid_max: 100
##             missing_value: -999
##         int base_time[]
##             origin: Epoch
##             string: Second
##             long_name: Seconds since 1998-1-1 0:00:00.0  00:00 UTC
##             units: sec since 1970-1-1 00:00:00.0 UTC
##         double time_offset[]
##             missing_value: -999
##             units: seconds since 1970-1-1 0:00:00 0:00
```

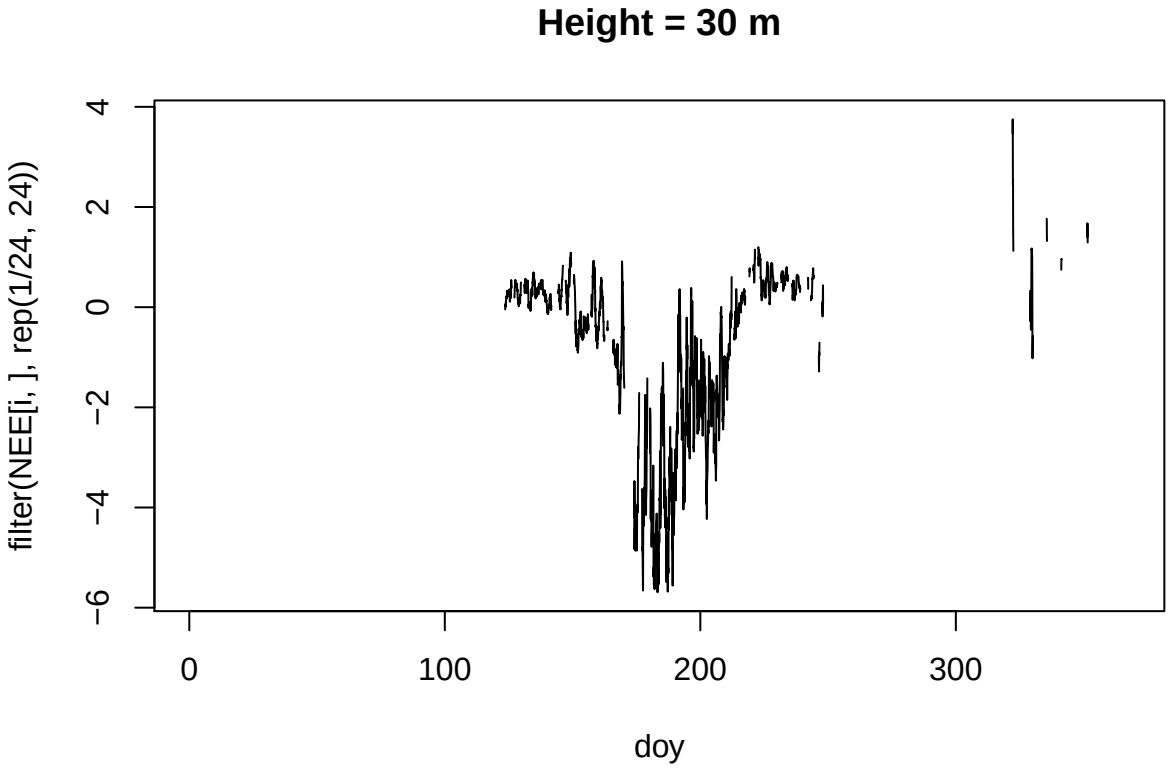
```
##             long_name: Time offset from base_time
##
##      5 dimensions:
##         time  Size:8784   *** is unlimited ***
##             long_name: Local Standard Time
##             standard_name: time
##             axis: T
##             actual_range: 0
##              actual_range: 8760
##             units: hours since 2012-1-1 00:00:00.00  UTC-6
##         record  Size:8784
##         level1  Size:5
##         level2  Size:3
##         level3  Size:6
##
##     5 global attributes:
##         Conventions: CF-1.3
##         history: File created by Dong Hua for flux tower data process
##         institution: CCR, WISC
##         source: http://flux.aos.wisc.edu/data/cheas/wlef/filled/current/
##         title: ChEAS flux tower data at WLEF
```

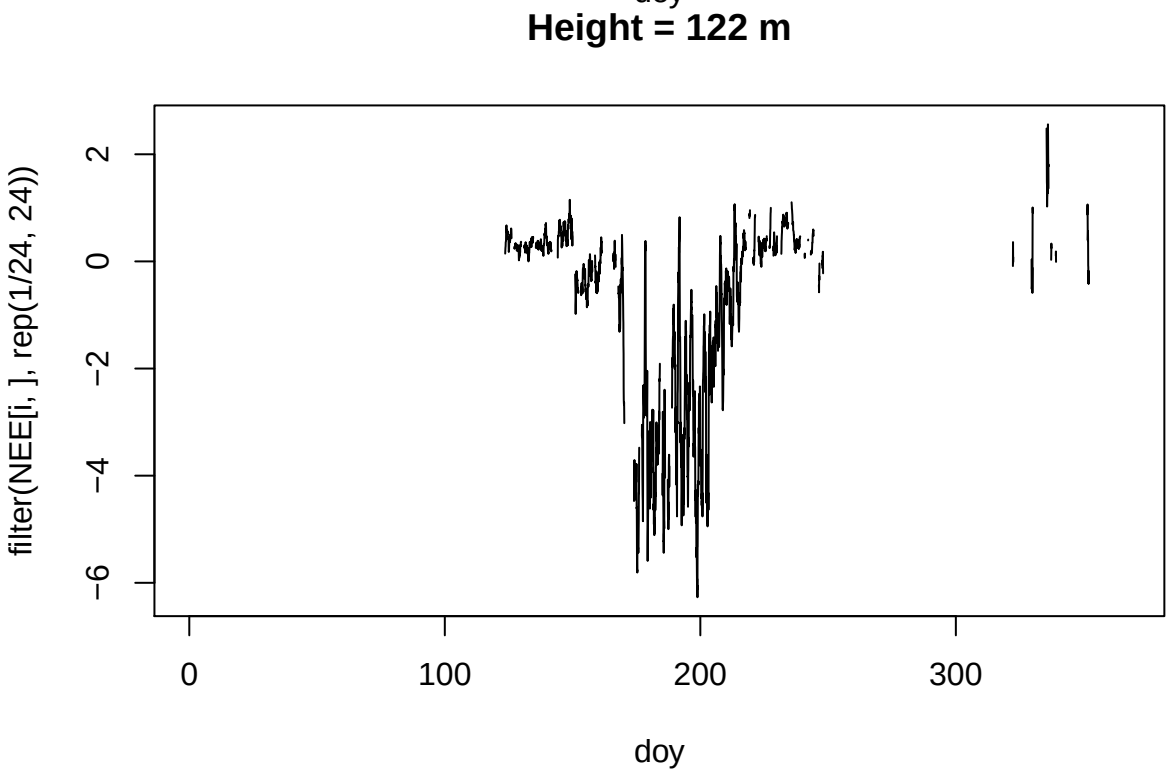
To start, lets look at the CO2 flux data, NEE_co2, which we see is stored in a matrix that has dimensions of [level2,time], where here level2 refers to the different measurements heights. If we want to grab this data and the vectors describing the dimensions we can do this as:

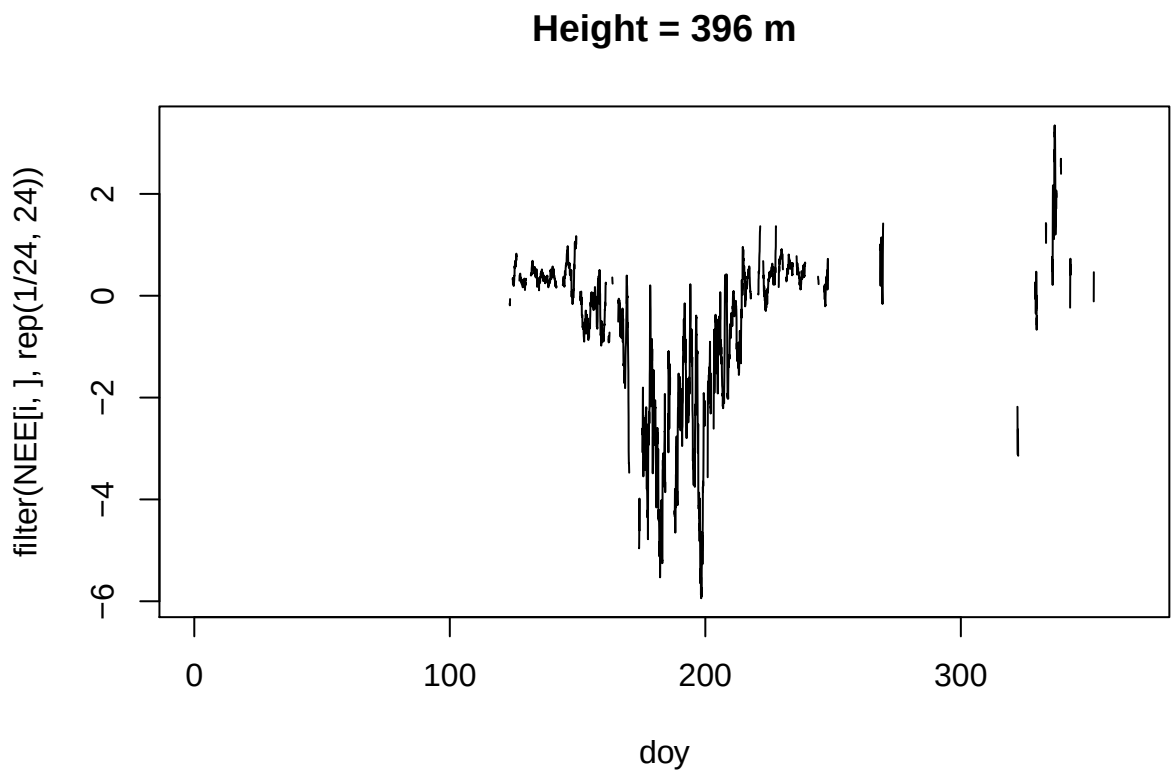
```
NEE = ncvar_get(wlef,"NEE_co2")    ## NEE data

## matrix dimensions
height = ncvar_get(wlef,"M_lvl")
doy = ncvar_get(wlef,"time")  # day of year

## close file connection
nc_close(wlef)
```

Finally, we can plot the data at the different heights. Since this flux data is recorded hourly the raw data is a bit of a cloud, therefore we use the function `filter` to impose a 24 hour moving window, which is indicated in the function as a vector of 24 weights, each given an equal weight of 1/24.

```
## print fluxes at 3 different heights
for(i in 1:3){
plot(doy,filter(NEE[i,],rep(1/24,24)),type='l',main=paste("Height =",height[i],"m"))
}
```

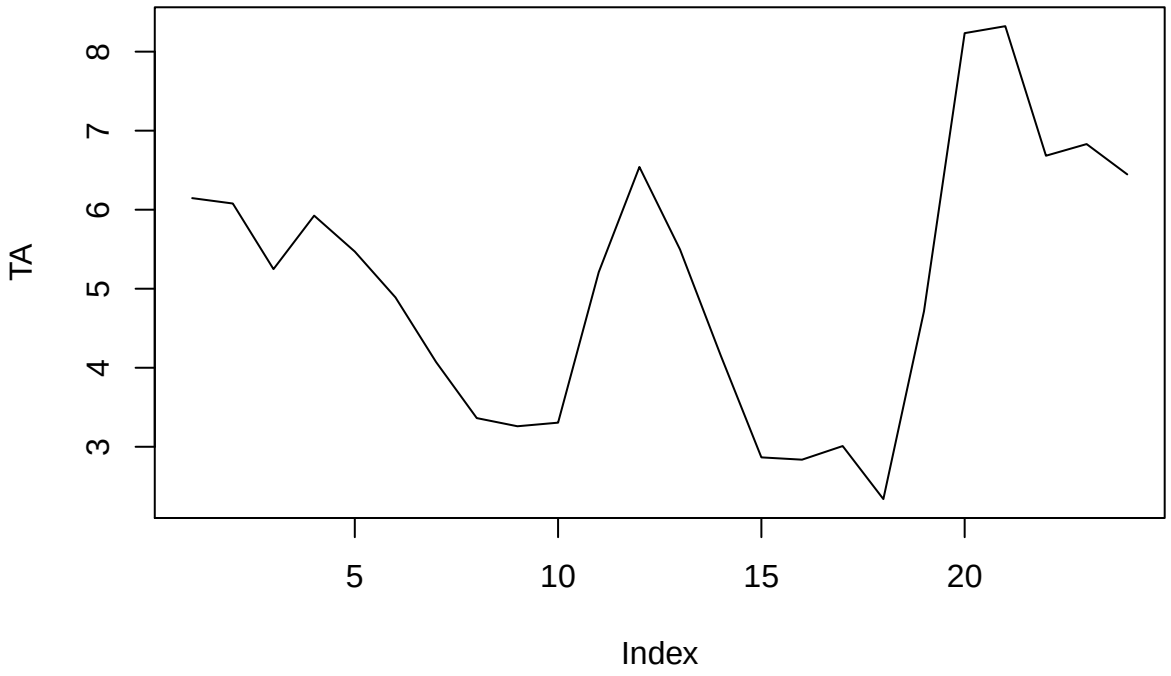

Height = 30 m
filter(NEE[i,], rep(1/24, 24))
4
2
0
−2
−4
−6
0
100
200
300
doy


Height = 122 m
filter(NEE[i,], rep(1/24, 24))
2
0
−2
−4
−6
0
100
200
300
doy



Alternative, if I just wanted to get a subset of air temperature data (e.g. 24 hours of data from the top height for the 220th day of the year)

```
start = which(doy > 220)[1]
wlef = nc_open("US-PFa-WLEF-TallTowerClean-2012-L0-vFeb2013.nc")
TA = ncvar_get(wlef,"TA",c(3,start),c(1,24))
plot(TA,type = 'l')
```



```
nc_close(wlef)
```

Question 4:

Similar to how we can point read.csv to the URL of a text file, you can open and manipulate netCDF files on remote servers if those servers support THREDDS/OpenDAP. Furthermore, these utilities let you grab just the part of the file that you need rather than the file in it's entirety. Using this approach, download and plot the air temperature data for Boston for 2004 that's located on the ORNL DAAC server `http://thredds.daac.ornl.gov/thredds/dodsC/ornldaac/1220/mstmip_driver_global_hd_climate_tair_2004_v1.n` The underlying file is quite large so make sure to grab just the subset you need. To do so you'll need to first grab the lat, lon, and time variables to find *which* grid cell to grab for lat and lon and how many values to grab from time (i.e. *length*).

Using APIs

In addition to data that is directly downloadable, and that which is scraped, there are a number of places on the web where data is available though interactive, code-based webservices called Application Programming Interfaces (APIs). In this example we will access the NASA MODIS API, using a pre-existing R package called MODISTools, as a demonstration of one of the many dataset-specific R packages.

First, we'll query the MODIS server to see what data products are available and what variables (bands) are available within one of those data products. More details about each data product (its definition, calculation, units, and missing data string) is available at https://lpdaac.usgs.gov/products/modis_products_table

```
MODISTools::mt_products()
```

```
##     product
## 1    Daymet
## 2   MCD12Q1
## 3   MCD12Q2
## 4  MCD15A2H
## 5  MCD15A3H
## 6   MCD19A3
## 7   MOD09A1
## 8   MOD11A2
## 9   MOD13Q1
## 10 MOD15A2H
## 11  MOD16A2
## 12 MOD17A2H
## 13 MOD17A3H
## 14  MYD09A1
## 15  MYD11A2
## 16  MYD13Q1
## 17 MYD15A2H
## 18  MYD16A2
## 19 MYD17A2H
## 20 MYD17A3H
## 21  MYD21A2
## 22  VNP09A1
## 23  VNP09H1
## 24  VNP13A1
## 25 VNP15A2H
##                                                                                 description
## 1             Daily Surface Weather Data (Daymet) on a 1-km Grid for North America, Version 3
## 2                       MODIS/Terra+Aqua Land Cover Type (LC) Yearly L3 Global 500 m SIN Grid
```

```
## 3                  MODIS/Terra+Aqua Land Cover Dynamics (LCD) Yearly L3 Global 500 m SIN Grid
## 4            MODIS/Terra+Aqua Leaf Area Index/FPAR (LAI/FPAR)  8-Day L4 Global 500 m SIN Grid
## 5             MODIS/Terra+Aqua Leaf Area Index/FPAR (LAI/FPAR) 4-Day L4 Global 500 m SIN Grid
## 6                MODIS/Terra+Aqua BRDF Model Parameters (MAIAC) 8-Day L3 Global 1 km SIN Grid
## 7                       MODIS/Terra Surface Reflectance (SREF) 8-Day L3 Global 500m SIN Grid
## 8     MODIS/Terra Land Surface Temperature and Emissivity (LST) 8-Day L3 Global 1 km SIN Grid
## 9                    MODIS/Terra Vegetation Indices (NDVI/EVI) 16-Day L3 Global 250m SIN Grid
## 10                MODIS/Terra Leaf Area Index/FPAR (LAI/FPAR) 8-Day L4 Global 500 m SIN Grid
## 11                   MODIS/Terra Net Evapotranspiration (ET) 8-Day L4 Global 500 m SIN Grid
## 12               MODIS/Terra Gross Primary Productivity (GPP) 8-Day L4 Global 500 m SIN Grid
## 13                  MODIS/Terra Net Primary Production (NPP) Yearly L4 Global 500 m SIN Grid
## 14                       MODIS/Aqua Surface Reflectance (SREF) 8-Day L3 Global 500m SIN Grid
## 15     MODIS/Aqua Land Surface Temperature and Emissivity (LST) 8-Day L3 Global 1 km SIN Grid
## 16                    MODIS/Aqua Vegetation Indices (NDVI/EVI) 16-Day L3 Global 250m SIN Grid
## 17                  MODIS/Aqua Leaf Area Index/FPAR (LAI/FPAR) 8-Day L4 Global 500 m SIN Grid
## 18                      MODIS/Aqua Net Evapotranspiration (ET) 8-Day L4 Global 500 m SIN Grid
## 19                MODIS/Aqua Gross Primary Productivity (GPP) 8-Day L4 Global 500 m SIN Grid
## 20                   MODIS/Aqua Net Primary Production (NPP) Yearly L4 Global 500 m SIN Grid
## 21 MODIS/Aqua Land Surface Temperature/3-Band Emissivity (LSTE) 8-Day L3 Global 1 km SIN Grid
## 22                       VIIRS/S-NPP Surface Reflectance (SREF) 8-Day L3 Global 1 km SIN Grid
## 23                       VIIRS/S-NPP Surface Reflectance (SREF) 8-Day L3 Global 500m SIN Grid
## 24                  VIIRS/S-NPP Vegetation Indices (NDVI/EVI) 16-Day L3 Global 500m SIN Grid
## 25                VIIRS/S-NPP Leaf Area Index/FPAR (LAI/FPAR) 8-Day L4 Global 500 m SIN Grid
##    frequency resolution_meters
## 1      Daily              1000
## 2     Yearly               500
## 3     Yearly               500
## 4      8-Day               500
## 5      4-Day               500
## 6      8-Day              1000
## 7      8-Day               500
## 8      8-Day              1000
## 9     16-Day               250
## 10     8-Day               500
## 11     8-Day               500
## 12     8-Day               500
## 13    Yearly               500
## 14     8-Day               500
## 15     8-Day              1000
## 16    16-Day               250
## 17     8-Day               500
## 18     8-Day               500
## 19     8-Day               500
## 20    Yearly               500
## 21     8-Day              1000
## 22     8-Day              1000
## 23     8-Day               500
## 24    16-Day               500
## 25     8-Day               500
```

```
MODISTools::mt_bands(product="MOD13Q1")
```

```
##                                    band
## 1                250m_16_days_VI_Quality
## 2           250m_16_days_blue_reflectance
```

```
## 3          250m_16_days_pixel_reliability
## 4            250m_16_days_NIR_reflectance
## 5            250m_16_days_MIR_reflectance
## 6                       250m_16_days_NDVI
## 7            250m_16_days_red_reflectance
## 8  250m_16_days_composite_day_of_the_year
## 9     250m_16_days_relative_azimuth_angle
## 10          250m_16_days_sun_zenith_angle
## 11                        250m_16_days_EVI
## 12         250m_16_days_view_zenith_angle
##                            description                  units
## 1                VI quality indicators              bit field
## 2           Surface Reflectance Band 3             reflectance
## 3      Quality reliability of VI pixel                   rank
## 4           Surface Reflectance Band 2             reflectance
## 5           Surface Reflectance Band 7             reflectance
## 6                  16 day NDVI average  NDVI ratio - No units
## 7           Surface Reflectance Band 1             reflectance
## 8                Day of year VI pixel Julian day of the year
## 9  Relative azimuth angle of VI pixel                 degrees
## 10       Sun zenith angle of VI pixel                 degrees
## 11                 16 day EVI average   EVI ratio - No units
## 12      View zenith angle of VI Pixel                 degrees
##       valid_range fill_value
## 1      0 to 65534         -1
## 2      0 to 10000      -1000
## 3          0 to 3         -1
## 4      0 to 10000      -1000
## 5      0 to 10000      -1000
## 6  -2000 to 10000      -3000
## 7      0 to 10000      -1000
## 8        1 to 366         -1
## 9   -3600 to 3600      -4000
## 10  -9000 to 9000     -10000
## 11 -2000 to 10000      -3000
## 12  -9000 to 9000     -10000
```

Next, lets grab the data for a specific band (EVI) within a specific product (MOD13Q1). We'll focus on the location of the WLEF flux tower and look at the same year as we did with the flux data (2012). The argument Size defines the dimensions of the box grabbed in terms of distance (in kilometers) outward from the center. Note that in practice we would also want to query the QAQC data for this variable, `250m_16_days_VI_Quality`, as well and use it to screen the data.

```
WC_file = "MODIS.WillowCreek.RData"
if(file.exists(WC_file)){
  load(WC_file)
} else {
  subset <- MODISTools::mt_subset(product = "MOD13Q1",
                                  band = "250m_16_days_EVI",
                                  lat=46.0827,
                                  lon=-89.9792,
                                  start="2012-01-01",
                                  end="2012-12-31",
                                  km_lr = 1,
                                  km_ab = 1,
```

```
                                  site_name = "WillowCreek")
  save(subset,file=WC_file)
}
subset$header
```

```
## $xllcorner
## [1] "-6940887.76"
##
## $yllcorner
## [1] "5123080.40"
##
## $cellsize
## [1] 231.6564
##
## $nrows
## [1] 9
##
## $ncols
## [1] 9
##
## $band
## [1] "250m_16_days_EVI"
##
## $units
## [1] "EVI ratio - No units"
##
## $scale
## [1] "0.0001"
##
## $latitude
## [1] 46.0827
##
## $longitude
## [1] -89.9792
##
## $site
## [1] "WillowCreek"
##
## $product
## [1] "MOD13Q1"
##
## $start
## [1] "2012-01-01"
##
## $end
## [1] "2012-12-31"
##
## $complete
## [1] TRUE
```

```
head(subset$data)
```

```
##   modis_date calendar_date             band   tile     proc_date pixel
## 1   A2012001    2012-01-01 250m_16_days_EVI h11v04 2015236172231     1
## 2   A2012017    2012-01-17 250m_16_days_EVI h11v04 2015237090200     1
```

```
## 3   A2012033    2012-02-02 250m_16_days_EVI h11v04 2015237195531     1
## 4   A2012049    2012-02-18 250m_16_days_EVI h11v04 2015238132930     1
## 5   A2012065    2012-03-05 250m_16_days_EVI h11v04 2015238152206     1
## 6   A2012081    2012-03-21 250m_16_days_EVI h11v04 2015239230118     1
##   data
## 1 1854
## 2  978
## 3 1657
## 4 2043
## 5  990
## 6 2582
```

Here we extracted a 250m data products and looked +/ 1km in both directions, which gives us a 9x9 area and thus 81 pixels.

```
unique(subset$data$pixel)
```

```
##  [1] "1"  "2"  "3"  "4"  "5"  "6"  "7"  "8"  "9"  "10" "11" "12" "13" "14"
## [15] "15" "16" "17" "18" "19" "20" "21" "22" "23" "24" "25" "26" "27" "28"
## [29] "29" "30" "31" "32" "33" "34" "35" "36" "37" "38" "39" "40" "41" "42"
## [43] "43" "44" "45" "46" "47" "48" "49" "50" "51" "52" "53" "54" "55" "56"
## [57] "57" "58" "59" "60" "61" "62" "63" "64" "65" "66" "67" "68" "69" "70"
## [71] "71" "72" "73" "74" "75" "76" "77" "78" "79" "80" "81"
```

For this example lets average over the spatial data and just generate a time-series of EVI.

```
## average EVI spatially & use 'scale' to set units
EVI = tapply(subset$data$data, subset$data$calendar_date, mean,na.rm=TRUE) * as.numeric(subset$header$s
time = as.Date(names(EVI))
```

Question 5: Plot EVI versus time and compare to the CO2 flux observations.

cron

The last topic I wanted to touch on isn't for data processing per se, but is handy for scheduling the automatic execution of tasks, and thus is frequently used in dynamic big data problems where new data is arriving on a regular basis and analyses need to be updated. An obvious example in the context of this course would be a forecast that would be updated on a daily or weekly basis. [note: like grep, cron is a *nix utility, so will run on linux, unix, and Mac OS, but not Windows].

cron jobs are specified in the cron table using the function `crontab` with takes the arguements -l to list the current contents or -e to edit the contents. The file contains a header component that allows us to specify information such as the shell used (SHELL=), path variables (PATH=), who to email job status updates (MAILTO=), and the directory to start from (HOME=), each on a separate line. Below the header is the table of the cron jobs themselves. A cron job consists of two components, the scheduling information and the command/script/program to be run. Lets take a look at a simple cron table

```
MAILTO=dietze@bu.edu
55 */2 * * * /home/scratch/dietze_lab/NOMADS/get_sref.sh
```

The last part of this is the easiest to explain – we're starting a script called get_sref from the NOMADS folder. NOMADS is the NOAA met server and SREF is one of their weather forecast products, so it should come as no surprise that this script is grabbing the numerical weather forecast. The first part of the script is more cryptic, but the five values given correspond to:

```
minute  This controls what minute of the hour the command will run on,
     and is between '0' and '59'
hour    This controls what hour the command will run on, and is specified in
```

```
         the 24 hour clock, values must be between 0 and 23 (0 is midnight)
dom This is the Day of Month, that you want the command run on, e.g. to
     run a command on the 19th of each month, the dom would be 19.
month   This is the month a specified command will run on, it may be specified
     numerically (0-12), or as the name of the month (e.g. May)
dow This is the Day of Week that you want a command to be run on, it can
     also be numeric (0-7) or as the name of the day (e.g. sun).
```

Values that are not specified explicitly are filled in with a *. Also, it is possible to specify lists (e.g. 0,6,12,18) or to specify a repeat frequency using a /. Thus the above example is set to run every other hour (/2) at 55 min past the hour.

Question #6:

Imagine you are working with the full FIA database and want to ensure that the data you are using is always up to date. However, the total size of the database is large, the USFS server is slow, and you don't want to completely delete and reinstall the database every day when only a small percentage of the data changes in any update.

- Write out the pseudocode/outline for how to keep the files up to date
- Write out what the cron table would look like to schedule this job (assume the update only needs to be done weekly)