RavenR Tutorial

Robert Chlumsky, Dr. James R. Craig

2017-11-16

As you go through this tutorial, don't just follow along blindly. Try to pay attention to what you are doing and how you are doing it.

This exercise will use the Nith River modelled output available from within the RavenR package, thus the functions to read in data from csv files are not required. However, it is recommended that you download the Nith river model files, and try to both run the model and read in the output files. The Nith river model can be downloaded from the Raven Tutorial #2.

Installing RavenR

The RavenR package can be installed in a number of ways. The RavenR package can be installed either from a source tarball file, or from the GitHub repository directly. To install direct from the repository, load the ('devtools') library and call the following command.

```
library(devtools)
install_github("rchlumsk/RavenR")
```

Start a new Rstudio session by opening RStudio. Load the RavenR library from the console and view its contents with the following commands:

```
library(RavenR)
ls("package:RavenR") # view all functions in RavenR
```

You can look at what any one of these functions does by typing out the name of the function beginning with a question mark, which will show the help information at the right of the RStudio environment.

```
?flow.scatterplot
```

Now you are ready to start using RavenR to directly visualize and manipulate model output. The sample data set from the RavenR package can be loaded in using the data function, e.g.,

```
data(forcing.data)
```

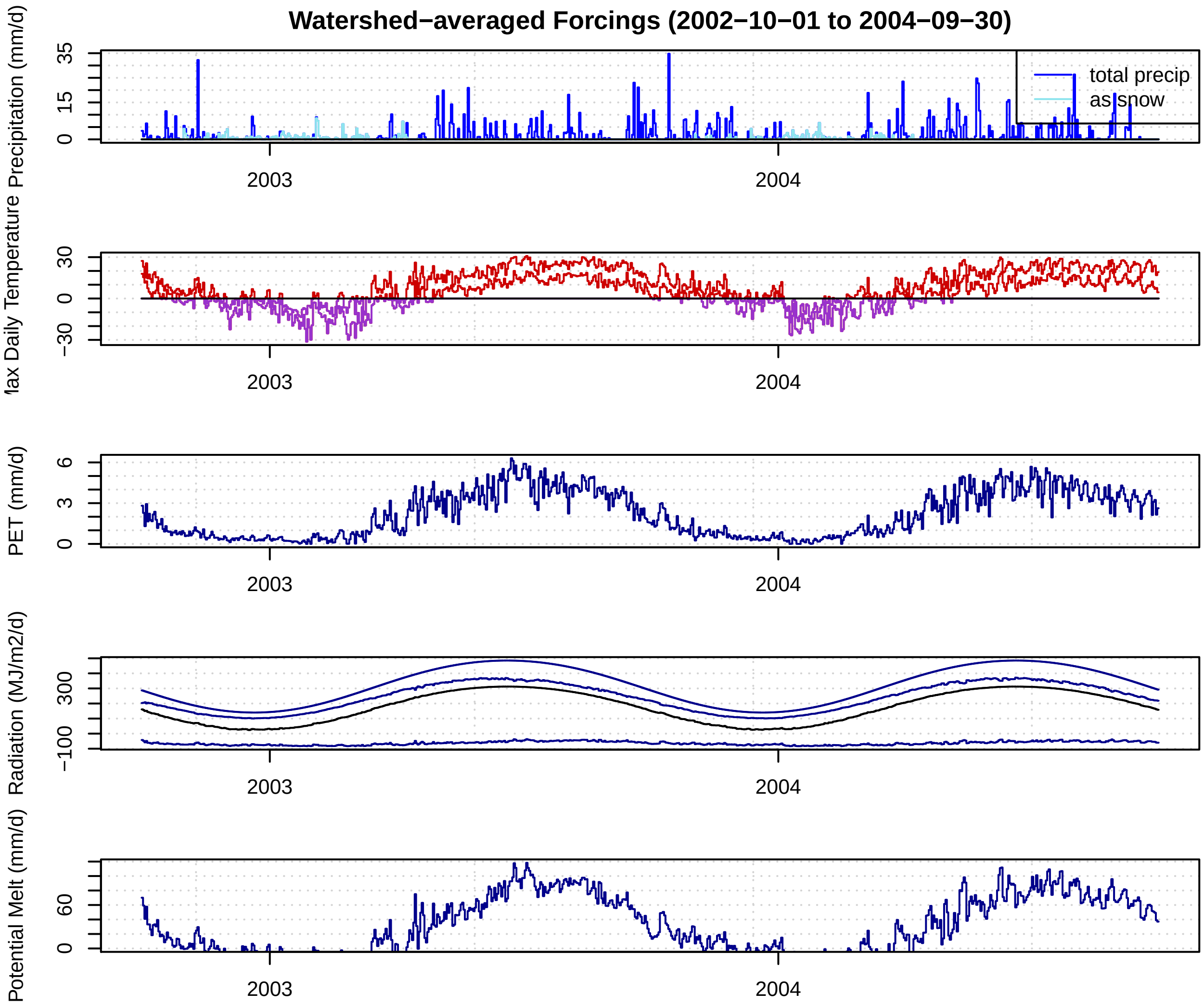
We will store the packaged forcing.data into an object called ff (and obtain just the subobject using the '$' operator), and then view the first few rows using the head function. We will show only the first few columns of the data for brevity.

```
ff <- forcing.data$forcings
head(ff[,1:6])
```

```
##            day_angle     rain snow    temp temp_daily_min temp_daily_max
## 2002-10-01   4.70809 3.468690    0 22.5956       17.92510        27.2662
## 2002-10-02   4.70809 3.468690    0 22.5956       17.92510        27.2662
## 2002-10-03   4.72530 1.189180    0 19.2076       15.40780        23.0075
## 2002-10-04   4.74251 2.083260    0 13.3714       11.49870        15.2440
## 2002-10-05   4.75973 6.474310    0 19.0304       12.50970        25.5510
## 2002-10-06   4.77694 0.125591    0 11.0186        7.43466        14.6024
```

Now we can plot the forcing data using the forcings.plot function. This creates an output of the five main forcings from the data set.

```
forcings.plot(ff)
```



This is typically a reasonable reality check on the model forcings. We can similarly access the hydrograph fit. Here the hydrograph sample data is set to the 'hy' object (normally read in from the Hydrographs.csv file using the hyd.read function). The flows from a specific subbasin can be extracted using the hyd.extract function, which is done here for subbasin 36. The precipitation can be extracted similarly.

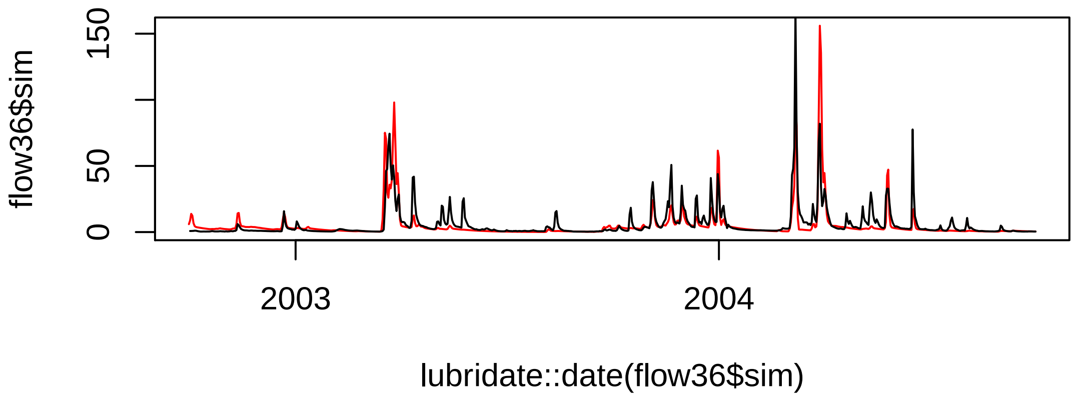
```
data(hydrograph.data)
hy <- hydrograph.data
head(hy$hyd)
```

```
##                precip    Sub36 Sub36_obs   Sub43 Sub43_obs
## 2002-10-01         NA  5.96354        NA 11.2505        NA
## 2002-10-02 3.468690  8.62464     0.801 13.3816      3.07
## 2002-10-03 1.189180 13.79200     0.828 16.6012      2.99
## 2002-10-04 2.083260 12.38190     0.860 17.4037      3.06
## 2002-10-05 6.474310  6.72838     0.903 18.7587      2.93
## 2002-10-06 0.125591  4.49263     1.040 16.3449      3.15
```

```
flow36 <- hyd.extract("Sub36",hy)
precip <- hyd.extract("precip",hy)$sim
```

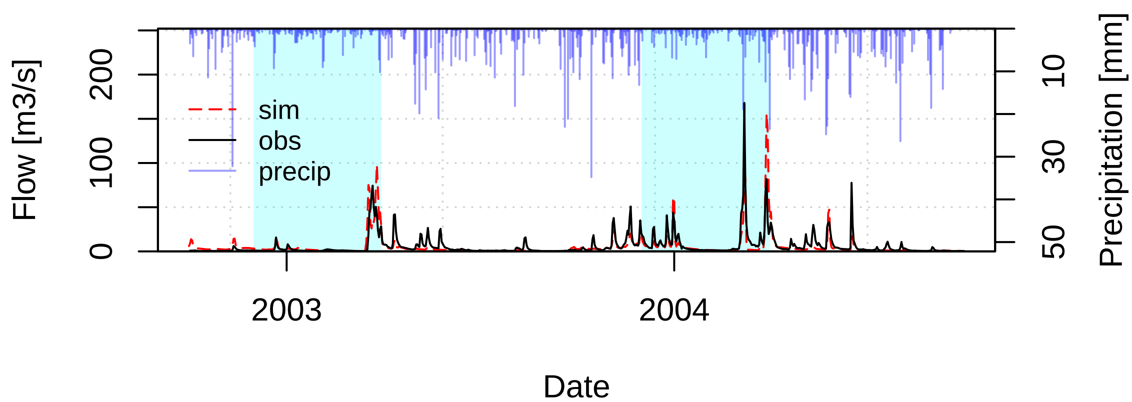
The hydrograph object flow3 now stores the simulated hydrograph (`flow36$sim`) and the observed hydrograph (`flow36$obs`), and the null subobject (`flow36$inflow`). The precip object stores the entire time series of watershed-averaged precip (`precip$sim`). We can plot the simulated and observed hydrograph with the simple commands:

```
plot(lubridate::date(flow36$sim),flow36$sim,col='red',type='l')
lines(lubridate::date(flow36$obs),flow36$obs,col='black')
```



Or using the special hydrograph plot function, which is part of the RavenR library.

```
hyd.plot(sim=flow36$sim, obs=flow36$obs, precip=precip, range.mult=1.5)
```




```
## [1] TRUE
```

The RavenR library can be explored to see what other functions are available in the package.

```
ls("package:RavenR")
```

Using the ?help option (where help is the name of a RavenR command), figure out how to plot:

1. a comparison of annual peak flows, and
2. the mean and median annual observed flow using the barplot() function (hint: use the apply.wyearly function to calculate annual mean and median)

Building a model workflow script

Now we will build a simple script which will provide a bunch of visualizations that we can use to look at the Nith river model each time we run it. This can be made as complex as you want.

Start with a new script. From RStudio, go to the main menu. `Choose File -> New File -> R Script`. Populate the script with the following. You can find the Nith model files in the Raven Tutorials.

```r
# Load the RavenR sample data
# =====================================================
indir <- "C:/temp/Nith/"
outdir <- "C:/temp/Nith/output/"
fileprefix <- "Nith"

if (dir.exists(outdir)==FALSE) {
  dir.create(outdir)
}

setwd(outdir)

# RUN RAVEN
# =====================================================
# writes complete command prompt command
# > Raven.exe [filename] -o [outputdir]
RavenCMD <-paste(indir,"Raven.exe ",indir,fileprefix," -o ",outdir,sep="")
system(RavenCMD) # this runs raven from the command prompt
```

Once the model is run, we can read in the output (or use the package data) and save some of the plots to file.

```r
# GENERATE OUTPUT PLOTS
# =====================================================
# use the package data, or read in the model output files

# ff<-forcings.read("ForcingFunctions.csv")
pdf("forcings.pdf") # create a pdf file to direct plot to
forcings.plot(ff$forcings)
dev.off() #finishes writing plot to .pdf file

data(watershed.data)
mywshd <- watershed.data$watershed.storage
#mywshd <- RavenR::watershed.read("WatershedStorage.csv")$watershed.storage
png("snowpack.png") # create a png file to direct plot to
plot(mywshd$snow)
dev.off() #finishes writing plot to .png file
```

Modify the script

Modify the above script to generate png image plots of monthly subbasin-averaged PET in Subbasin 43 using the :CustomOutput option (you will have to add a :CustomOutput command to the Raven input rvi file). You will also want to use the RavenR `custom.read()` and `customoutput.plot()` commands.

More exercises

This short exercise is meant to serve as a brief introduction to the RavenR package. The complete RavenR Tutorial can be found on the Raven downloads page. If you have any comments, suggestions or bug reports, please email the authors of the package or feel free to let us know on the Raven forum.