RavenR Introduction (v1.2)

Robert Chlumsky, Dr. James R. Craig

May 2018

Introduction

This short exercise is meant to introduce you through the RavenR package (version 1.2). The RavenR package is intended for pre- and post-processing of data files and information related to the use of the Raven Hydrologic Modelling Framework, and includes functions for:

- reading Raven output files, including Hydrographs, ReservoirStages, ExhaustiveMB, custom output, etc.
- processing input time series data files, including Environment Canada meteorological and Water Survey Canada hydrometric data
- processing and plotting Raven hydrograph diagnostics
- visualizing Raven output, including GIS support for subbasin/ HRU mappiung
- other Raven file processing, including RVH file support
- other utilities, including water year time series manipulation and Ostrich file processing

As you go through this exercise, don't just follow along blindly. Try to pay attention to what you are doing and how you are doing it. Note that the RavenR package is in active development, so feel free to pass along any feedback, errors, or suggestions to Robert Chlumsky (rchlumsk@uwaterloo.ca).

This exercise will use the internally stored data files, some of which are obtained from the Raven Tutorial files. If you are unfamiliar with the Raven Hydrologic Modelling Framework, it is recommended that you download the Raven executable and tutorials (see Raven Downloads) and go through the Raven tutorials prior to the RavenR tutorial. The GIS examples in this exercise are based on Raven Tutorial #2 for the Nith subwatershed.

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)
```

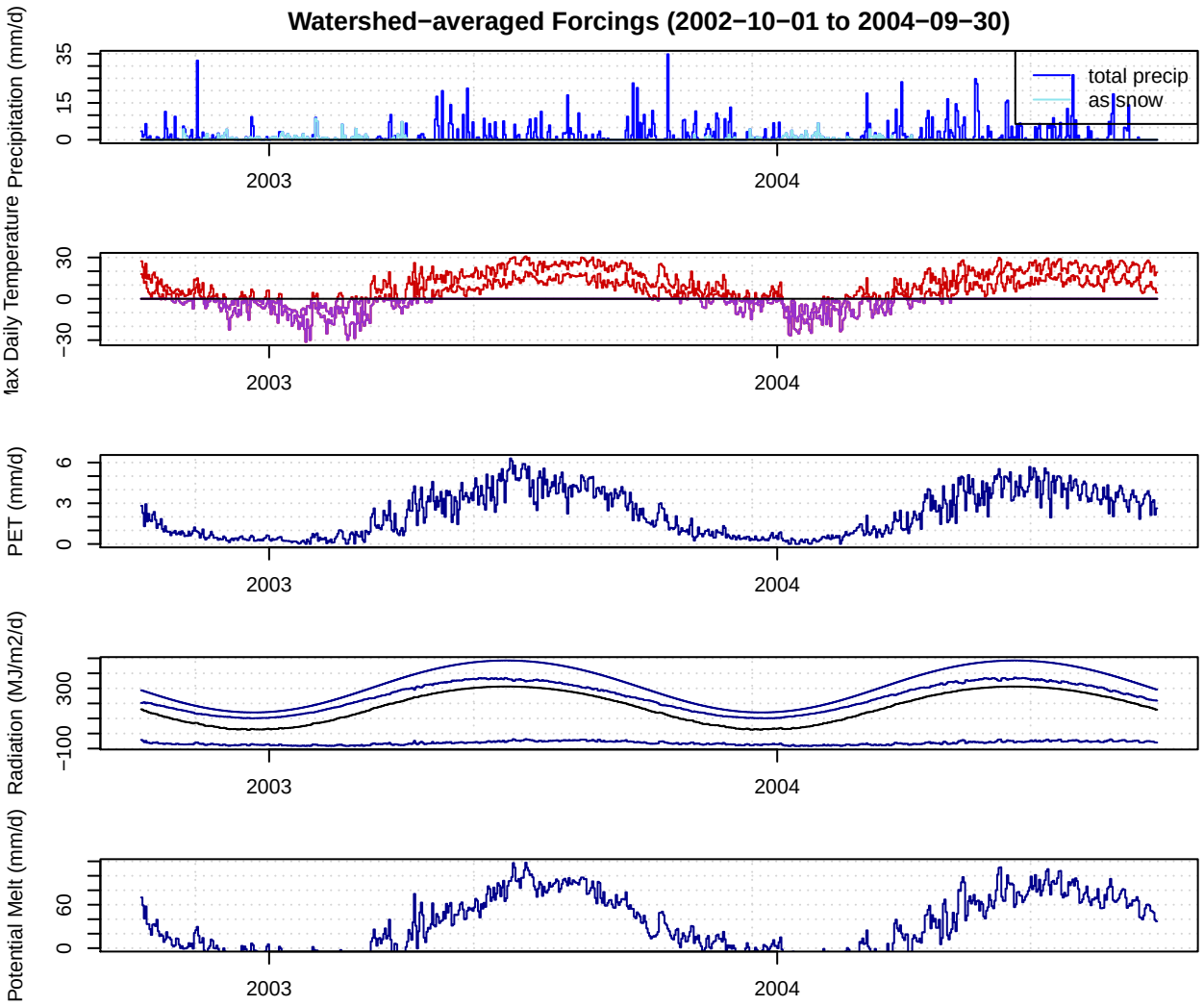
We will store the packaged forcing.data into an object called ff (and obtain just the subobject using the '$' operator), and then view the first few rows using the head function. We will show only the first few columns of the data for brevity.

```
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.

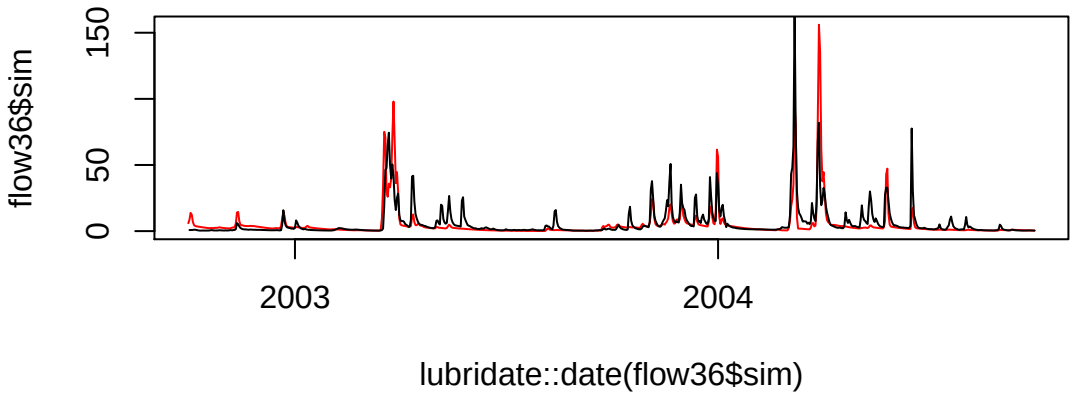
```
data(hydrograph.data)
hy <- hydrograph.data
head(hy$hyd)
```

```
##              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
```

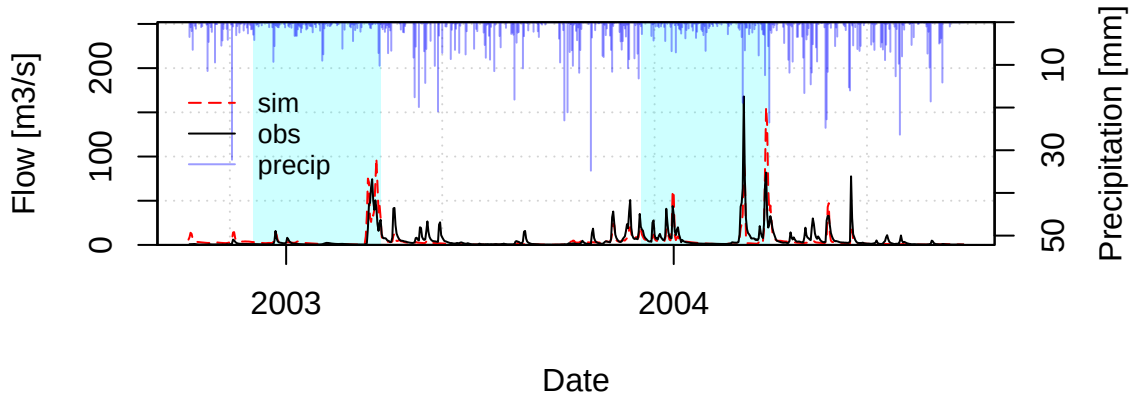
The hydrograph object flow3 now stores the simulated hydrograph (`flow36$sim`) and the observed hydrograph (`flow36$obs`), and the null subobject (`flow36$inflow`). The precip object stores the entire time series of watershed-averaged precip (`precip$sim`). We can plot the simulated and observed hydrograph 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 hydrograph plot function, which is part of the RavenR library.

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




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

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

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

Using the function documentation accessed by ? option, figure out how to plot:

1. a comparison of annual peak flows,
2. a bar plot of monthly volume bias in simulated and observed flows,
3. a 'spaghetti plot' of flows,
4. a residuals plot of simulated and observed flows.

Subbasin and HRU Mapping Functions

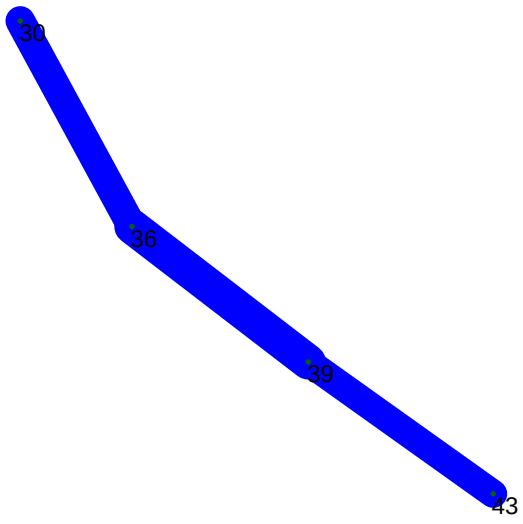
The new v1.2 of RavenR contains several functions for visualizing the watershed itself directly in R. For example, a network plot can be made, which uses the information in the .rvh file to map the subbasin network of the watershed. First let's read in the sample rvh file for the Nith subwatershed (from the Raven tutorials) from the RavenR package.

```
Nithrvh <- system.file("extdata","Nith.rvh", package="RavenR")
myrvh <- rvh.read(Nithrvh)
```

The object `myrvh` now contains the information from the .rvh file, plus some useful diagnostics. For example, try calling 1myrvh$SBtable‘ and find the dominant landuse of each subbasin.

We can use the SBtable information in `myrvh` to create a subbasinnetwork plot.

```
subbasinNetwork.plot(myrvh$SBtable,labeled=T)
```




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

The plot is fairly simple since we have four subbasins connected in a linear way, but try the function out with more complex basins for effect!

What about plotting from a GIS file? The `SBMap.plot` function plots custom data onto a shapefile for the given subbasins. Let's try this out with the Nith dataset stored in the RavenR raw data files:

```
Nith.shp <- system.file("extdata","Nith_shapefile_sample.shp", package="RavenR")
Nith.customdata <- custom.read(system.file("extdata","run1_PRECIP_Daily_Maximum_BySubbasin.csv", packag
SBMap.plot(Nith.shp,4,cust.data=Nith.customdata,plot.date="2002-11-10")
```

Creating a plot like this one for a single date is great, but why settle for showing one date at a time? You can also look at animating your plot and producing a GIF of all of the custom data within a given date range

using `?SBMap.animate`. Try using the data above to produce an animated plot to file.

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 (visit the Raven Downloads page if you have not yet done so).

```
# 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.

```
# 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 is also available, which contains a more comprehensive version of this exercise for the RavenR package. 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.