Computing Summary Statistics for Daily Data

Dave Lorenz

April 18, 2014

These examples demonstrate how to compute selected summary statistics for daily streamflow data. The examples can easily be extended to other statistics or data types.

```
> # Load the USGSwsGraphs package
> library(USGSwsBase)
> # Retrieve streamflow data for the Raccoon River at Van Meter, IA
> # for the 30-year period beginning 1980-10-01.
> # Use the renCol function to rename the straflow column to Flow
> RRVM <- renCol(readNWIS("05484500", begin.date="1980-10-01",
+                         end.date="2010-09-30"))
> # Print the first and last few rows of the data
> head(RRVM)

  agency_cd  site_no   datetime Flow Flow_cd
1      USGS 05484500 1980-10-01  278       A
2      USGS 05484500 1980-10-02  258       A
3      USGS 05484500 1980-10-03  235       A
4      USGS 05484500 1980-10-04  225       A
5      USGS 05484500 1980-10-05  221       A
6      USGS 05484500 1980-10-06  216       A

> tail(RRVM)

      agency_cd  site_no   datetime Flow Flow_cd
10952      USGS 05484500 2010-09-25 4180       A
10953      USGS 05484500 2010-09-26 6150       A
10954      USGS 05484500 2010-09-27 7060       A
10955      USGS 05484500 2010-09-28 6960       A
10956      USGS 05484500 2010-09-29 6240       A
10957      USGS 05484500 2010-09-30 5380       A

> # Check for missing values
> with(RRVM, screenData(datetime, Flow, year = "water"))

No missing data between 1980-10-01 and 2010-09-30
```

1 Computing Daily Mean Values

The simplest and most straightforward way to compute summary statistics from arbitrarily grouped data is to use the `tapply` function. At its simplest, it requires only three arguments—`X`, the data to summarize; `INDEX`, the grouping data; and `FUN`, the summary statistic function.

The `USGSwsBase` package contains the `baseDay` function that can be used to group data by day, so that all data for each day, including February 29, can be summarized. The output can be arranged so that the sequence represents the calendar-, water- or climate-year; beginning January 1, October 1, or April 1.

The following script demonstrates how to use the `tapply` and `baseDay` functions to compute the daily mean streamflow for the previously retrieved data. It uses the `with` function to facilitate referring to columns in the dataset.

```
> # There are no missing values, so only need the basic
> # 3 arguments for tapply
> RRVM.daily <- with(RRVM, tapply(Flow,
+     baseDay(datetime, numeric=FALSE, year="water"), mean))
> # Print the first and last few values of the output
> head(RRVM.daily)

  Oct 01   Oct 02   Oct 03   Oct 04   Oct 05   Oct 06
804.3000 777.7000 792.7667 884.4667 976.8333 984.5000

> tail(RRVM.daily)

   Sep 25    Sep 26    Sep 27    Sep 28    Sep 29    Sep 30
1059.5667 1093.9333 1097.7333 1067.0000 1060.5667  988.2333
```

The output from `tapply` is an array. Because the output from this example is an array of one dimension, it is printed in the form of a named vector. Had the summary statistic function been `quantile`, for example, the output would have been a list.

The `tapply` function is very powerful and easy to use. But there are times when we want the output in the form of a dataset rather than a vector or array. In those cases, the `aggregate` function is a better alternative. The `aggregate` function has several usage options. The script below demonstrates how to build a formula to compute the same statistics that we computed in the previous script. Early versions of `aggregate` required the output of the summary statistic function to be a scalar, but that is no longer a limitation.

```
> # There are no missing values
> RRVM.dailyDF <- aggregate(Flow ~
+     baseDay(datetime, numeric=FALSE, year="water"),
+     data=RRVM, FUN=mean)
> # Print the first and last few values of the output
> head(RRVM.dailyDF)
```

```
  baseDay(datetime, numeric = FALSE, year = "water")     Flow
1                                              Oct 01 804.3000
2                                              Oct 02 777.7000
3                                              Oct 03 792.7667
4                                              Oct 04 884.4667
5                                              Oct 05 976.8333
6                                              Oct 06 984.5000

> tail(RRVM.dailyDF)

    baseDay(datetime, numeric = FALSE, year = "water")      Flow
361                                              Sep 25 1059.5667
362                                              Sep 26 1093.9333
363                                              Sep 27 1097.7333
364                                              Sep 28 1067.0000
365                                              Sep 29 1060.5667
366                                              Sep 30  988.2333

> # Change the name of the grouping column
> names(RRVM.dailyDF)[1] <- "Day"
```

Note that the grouping column, now called Day, is a factor. There are times when it would be better to be a simple character. It can easily be converted by the executing the expression: `RRVM.dailyDF$Day <- as.character(RRVM.dailyDF$Day)`.

2 Computing Annual Mean Values

The example above can easily be expanded to any grouping that the user desires. This example computes annual means by water year. The `waterYear` function in `USGSwsBase` is used to group the data by water year.

```
> # There are no missing values
> RRVM.wyDF <- aggregate(Flow ~
+     waterYear(datetime),
+     data=RRVM, FUN=mean)
> # Change the name of the grouping column
> names(RRVM.wyDF)[1] <- "WaterYear"
> # Print the first few values of the output
> head(RRVM.wyDF)

  WaterYear      Flow
1      1981  373.6986
2      1982 2198.0192
3      1983 4572.0767
4      1984 4005.3962
5      1985 1009.1425
6      1986 2659.5178
```

Other grouping functions include `year` (calendar year) in `lubridate`, `month` (month) in `lubridate`, `seasons` (user-defined seasons) in `USGSwsBase`. Refer to the documentation for each of these function for a description of the arguments.

3 Computing Year and Month Mean Values

Aggregation can also be done by multiple grouping variables. This example computes the mean streamflow for each month by year. This example uses the `year` and the `month` functions because the output is sorted by groups. The sequence of the groups in the call is important—the sorting is done in the order specified in the formula. For this example, the data are sorted by month and then by year, which in his case, keeps the order correct; grouping by water year would misplace October, November and December. For a calendar year table, the months are in the correct order.

```
> # There are no missing values
> RRVM.my <- aggregate(Flow ~ month(datetime, label=TRUE) + year(datetime),
+     data=RRVM, FUN=mean)
> # Rename columns 1 and 2
> names(RRVM.my)[1:2] <- c("Month", "Year")
> # Print the first few values of the output
> head(RRVM.my)

  Month Year     Flow
1   Oct 1980 226.0323
2   Nov 1980 201.0333
3   Dec 1980 137.0000
4   Jan 1981 102.0000
5   Feb 1981 256.9643
6   Mar 1981 166.1613
```

The output dataset may be used as is, or it could be restructured to a table of monthly values for each water year. To do that, a column of water year must be added and the order months must be set to correspond to those in the water year. The following script uses the `%in%` function to recode the water year from the year.

```
> # Create new object, compute the water year and round Flow
> RRVM.myTbl <- transform(RRVM.my,
+     WY=ifelse(Month %in% c("Oct", "Nov", "Dec"), Year + 1, Year),
+     Flow=signif(Flow, 3))
> # Reorder months
> RRVM.myTbl$Month <- factor(RRVM.myTbl$Month,
+     levels=c("Oct", "Nov", "Dec", "Jan", "Feb", "Mar", "Apr", "May",
+              "Jun", "Jul", "Aug", "Sep"))
> # Restructure the dataset, overwrite the new object
> RRVM.myTbl <- group2row(RRVM.myTbl, "WY", "Month", "Flow")
> # Print the first few values of the output, set width for Vignette
> options(width=70)
> head(RRVM.myTbl)

    WY Oct.Flow Nov.Flow Dec.Flow Jan.Flow Feb.Flow Mar.Flow Apr.Flow
1 1981      226      201      137      102      257      166      207
```

```
2 1982      332      409      509      221     2180     4010     2380
3 1983     2450     2250     3090     2520     4970     8320    10600
4 1984     1230     3060     1760      865     5440     3260     9100
5 1985      558      776     1130     1390     1310     1280     1820
6 1986      436      268      247      278      474     3530     3230
  May.Flow Jun.Flow Jul.Flow Aug.Flow Sep.Flow
1      360      550     1170      727      364
2     6650     3960     3860      913      913
3     6300     6090     6920      853      551
4     9260    11000     2530      663      274
5     1900     1070      483      192      223
6     7240     4470     6730     2590     2200
```

Note that this example used the `group2row` function in `USGSwsBase`. The `reshape` function in `stats` and `stack` and `unstack` functions in `utils` are other functions that will restructure data.