Code fragments by metric

Patrick Eslick

September 25, 2017

Data download and preliminary calculations

This fragment of code gets data for each site before calculation of the metrics

```
#Load packages
library(dataRetrieval)
library(lubridate)

#Get daily mean flow for kill creek. The period of record is 2004-2018, counting only com
plete years
killFlow <- readNWISdv("06892360", "00060")[,c("Date", "X_00060_00003")]
names(killFlow) <- c("date", "discharge")
killFlow$year <- year(killFlow$date)

#Get daily mean flow for stateline. The period of record is 2004-2018, counting only comp
lete years
statelineFlow <- readNWISdv("06893390", "00060")[,c("Date", "X_00060_00003")]
names(statelineFlow) <- c("date", "discharge")
statelineFlow$year <- year(statelineFlow$date)

#Get data for the overland park site, because of the mid-1980s shift in distribution, only
get data from 1985 forward
overlandFlow <- readNWISdv("06893300", "00060", "1984-12-31")[,c("Date", "X_00060_00003")
]
names(overlandFlow) <- c("date", "discharge")
overlandFlow$year <- year(overlandFlow$date)

#Discard incomplete years of data - the preCheck only keeps complete calendar years of fl
ow data
preCheck <- function(dailyData) {
  dates <- dailyData$date
  #Find the starting and ending years
  firstYear <- year(min(dates))
  lastYear <- year(max(dates))
  years <- firstYear:lastYear
  firstDay <- as.Date(paste(firstYear, "-01-01", sep=""))
  lastDay <- as.Date(paste(lastYear, "-12-31", sep=""))
  #Generate a list of dates from the first of the first year to the last of the last year
  completeDates <- seq(from=firstDay, to=lastDay, by=1)
  completeYears <- year(completeDates)
  #Check if the dates in the dataframe are are in completeDates
  present <- completeDates %in% dates
  complete <- data.frame(date = completeDates, year=completeYears, present=present)
  isComplete <- vector()
  for(i in years) {
    if(all(complete[complete$year==i,"present"])) {
      isComplete[length(isComplete)+1] <- TRUE
```

```
    } else {
      isComplete[length(isComplete)+1] <- FALSE
    }
  }
  keepDates <- complete[complete$year %in% years[isComplete],"date"]
  dailyData <- dailyData[dailyData$date %in% keepDates,]
  return(dailyData)
}

killFlow <- preCheck(killFlow)
statelineFlow <- preCheck(statelineFlow)
overlandFlow <- preCheck(overlandFlow)

#Print the first five rows of each dataset to demonstrate what they look like
print(head(killFlow))
```

```
          date discharge year
281 2004-01-01       4.6 2004
282 2004-01-02       4.8 2004
283 2004-01-03       4.5 2004
284 2004-01-04       6.5 2004
285 2004-01-05       5.3 2004
286 2004-01-06       3.5 2004
```

```
print(head(statelineFlow))
```

```
          date discharge year
255 2004-01-01        39 2004
256 2004-01-02        38 2004
257 2004-01-03        36 2004
258 2004-01-04        74 2004
259 2004-01-05        39 2004
260 2004-01-06        30 2004
```

```
print(head(overlandFlow))
```

```
        date discharge year
2 1985-01-01        64 1985
3 1985-01-02        43 1985
4 1985-01-03        30 1985
5 1985-01-04        24 1985
6 1985-01-05        25 1985
7 1985-01-06        34 1985
```

This fragment of code uses data downloaded and more U.S. Geological Survey National Water Information System (NWIS) calls to do preliminary calculations used in subsequent analysis.

```
#Calulate P10, P50, and P90 for each site, based on the period of record (or subset, in t
he case of Overland Park site)
#Calculate kill creek
kill_P10P50P90 <- quantile(killFlow$discharge, c(0.10, 0.50, 0.90))
#Calculate state line
stateline_P10P50P90 <- quantile(statelineFlow$discharge, c(0.10, 0.50, 0.90))
#Calculate overland park
overland_P10P50P90 <- quantile(overlandFlow$discharge, c(0.10, 0.50, 0.90))

#Find the drainage area for each, define a function for getting drainage area from NWIS
```

```r
drainageArea <- function(site) {
  siteInfo <- readNWISsite(site)
  drainArea <- siteInfo$drain_area_va
  return(drainArea)
}
#Kill Creek
kill_drainage <- drainageArea("06892360")
#State Line
stateline_drainage <- drainageArea("06893390")
#Overland Park
overland_drainage <- drainageArea("06893300")
```

Code fragments by metric

Note: Some example output may vary from new output because of provisional discharge values that changed after this document was created.

P50 (MED_ANNUAL)

```r
#Calculate P50 for each site in the year 2004
kill_P50 <- median(killFlow$discharge[killFlow$year==2004])/kill_drainage
stateline_P50 <- median(statelineFlow$discharge[statelineFlow$year==2004])/stateline_drai
nage
overland_P50 <- median(overlandFlow$discharge[overlandFlow$year==2004])/overland_drainage

print(kill_P50)
```

```
[1] 0.1582397
```

```r
print(stateline_P50)
```

```
[1] 0.6233442
```

```r
print(overland_P50)
```

```
[1] 0.6390977
```

AVG_ANNUAL (MEAN_ANNUAL)

```r
#Calculate AVG_ANNUAL for each site in the year 2004
kill_AVG <- mean(killFlow$discharge[killFlow$year==2004])/kill_drainage
stateline_AVG <- mean(statelineFlow$discharge[statelineFlow$year==2004])/stateline_draina
ge
overland_AVG <- mean(overlandFlow$discharge[overlandFlow$year==2004])/overland_drainage

print(kill_AVG)
```

```
[1] 0.6175861
```

```r
print(stateline_AVG)
```

```
[1] 1.613759
```

```r
print(overland_AVG)
```

```
[1] 1.671648
```

CV_FLOW

```
#Calculate CV_FLOW for each site in the year 2004

#Define a function to calculate coefficient of variation
coefVar <- function(dailyData) {
  q <- dailyData$discharge
  cv <- (sd(q)/mean(q)) * 100
  return(cv)
}

kill_CV <- coefVar(killFlow[killFlow$year==2004,])
stateline_CV <- coefVar(statelineFlow[statelineFlow$year==2004,])
overland_CV <- coefVar(overlandFlow[overlandFlow$year==2004,])

print(kill_CV)
```

```
[1] 367.7348
```

```
print(stateline_CV)
```

```
[1] 289.137
```

```
print(overland_CV)
```

```
[1] 269.5491
```

AVG_JULY (MEAN_JULY)

This code fragment calculates one monthly average of daily mean flows to demonstrate how the function works

```
#Calculate AVG_JULY for each site in 2004

#Define a function
#Find the average discharge for a given month
avgMonth <- function(dailyData, month) {
  dailyData$month <- month(dailyData$date)
  avg <- mean(dailyData[dailyData$month==month,"discharge"])
  return(avg)
}

kill_JUL <- avgMonth(killFlow[killFlow$year==2004,], 7)/kill_drainage
stateline_JUL <- avgMonth(statelineFlow[statelineFlow$year==2004,], 7)/stateline_drainage
overland_JUL <- avgMonth(overlandFlow[overlandFlow$year==2004,], 7)/overland_drainage

print(kill_JUL)
```

```
[1] 2.046152
```

```
print(stateline_JUL)
```

```
[1] 2.483826
```

```
print(overland_JUL)
```

```
[1] 2.653044
```

MIN_1DAY and MAX_1DAY

```
#Calculate the 1 day minimum and 1 day maximum for each site in 2004

#Kill creek
kill_MIN1DAY <- min(killFlow$discharge[killFlow$year==2004])/kill_drainage
kill_MAX1DAY <- max(killFlow$discharge[killFlow$year==2004])/kill_drainage

#State Line
stateline_MIN1DAY <- min(statelineFlow$discharge[statelineFlow$year==2004])/stateline_dra
inage
stateline_MAX1DAY <- max(statelineFlow$discharge[statelineFlow$year==2004])/stateline_dra
inage

#Overland Park
overland_MIN1DAY <- min(overlandFlow$discharge[overlandFlow$year==2004])/overland_drainag
e
overland_MAX1DAY <- max(overlandFlow$discharge[overlandFlow$year==2004])/overland_drainag
e

print(c(kill_MIN1DAY, kill_MAX1DAY))
```

```
[1]  0.02059925 27.90262172
```

```
print(c(stateline_MIN1DAY, stateline_MAX1DAY))
```

```
[1]  0.2960885 66.2303257
```

```
print(c(overland_MIN1DAY, overland_MAX1DAY))
```

```
[1]  0.3684211 60.5263158
```

MIN_3DAY and MAX_3DAY

```
#Define a function for finding the rolling mean for X number of days
rollX <- function(dailyData, days) {
  #Initialize a vector to hold each rolling sum
  rolling_mean <- vector(mode="numeric")
  #For each set of days
  for(i in 1:(nrow(dailyData)-days)) {
    rolling_mean[length(rolling_mean) + 1] <- mean(dailyData[i:(i+(days-1)),"discharge"])
  }
```

```
  #Return the lowest mean
  return(rolling_mean)
}

#Find the rolling 3 day means for each site
kill_roll3 <- rollX(killFlow[killFlow$year==2004,], 3)
stateline_roll3 <- rollX(statelineFlow[statelineFlow$year==2004,], 3)
overland_roll3 <- rollX(overlandFlow[overlandFlow$year==2004,], 3)

#Find the minimum and maximum rolling 3 day mean for each
kill_MIN3DAY <- min(kill_roll3)/kill_drainage
kill_MAX3DAY <- max(kill_roll3)/kill_drainage

stateline_MIN3DAY <- min(stateline_roll3)/stateline_drainage
stateline_MAX3DAY <- max(stateline_roll3)/stateline_drainage

overland_MIN3DAY <- min(overland_roll3)/overland_drainage
overland_MAX3DAY <- max(overland_roll3)/overland_drainage

#Print the metrics
print(c(kill_MIN3DAY, kill_MAX3DAY))

[1]  0.02247191 11.66666667

print(c(stateline_MIN3DAY, stateline_MAX3DAY))

[1]  0.3324503 25.8843696

print(c(overland_MIN3DAY, overland_MAX3DAY))

[1]  0.387218 23.696742
```

MIN_7DAY and MAX_7DAY

```
#Define a function for finding the rolling mean for X number of days
rollX <- function(dailyData, days) {
  #Initialize a vector to hold each rolling sum
  rolling_mean <- vector(mode="numeric")
  #For each set of days
  for(i in 1:(nrow(dailyData)-days)) {
    rolling_mean[length(rolling_mean) + 1] <- mean(dailyData[i:(i+(days-1)),"discharge"])
  }
  #Return the lowest mean
  return(rolling_mean)
}

#Find the rolling 7 day means for each site
kill_roll7 <- rollX(killFlow[killFlow$year==2004,], 7)
stateline_roll7 <- rollX(statelineFlow[statelineFlow$year==2004,], 7)
overland_roll7 <- rollX(overlandFlow[overlandFlow$year==2004,], 7)

#Find the minimum and maximum rolling 7 day mean for each
```

```r
kill_MIN7DAY <- min(kill_roll7)/kill_drainage
kill_MAX7DAY <- max(kill_roll7)/kill_drainage

stateline_MIN7DAY <- min(stateline_roll7)/stateline_drainage
stateline_MAX7DAY <- max(stateline_roll7)/stateline_drainage

overland_MIN7DAY <- min(overland_roll7)/overland_drainage
overland_MAX7DAY <- max(overland_roll7)/overland_drainage

#Print the metrics
print(c(kill_MIN7DAY, kill_MAX7DAY))
```

```
[1] 0.02675227 5.20599251
```

```r
print(c(stateline_MIN7DAY, stateline_MAX7DAY))
```

```
[1]  0.3866961 11.8079209
```

```r
print(c(overland_MIN7DAY, overland_MAX7DAY))
```

```
[1]  0.4822771 11.0365199
```

MIN_30DAY

```r
#Define a function for finding the rolling mean for X number of days
rollX <- function(dailyData, days) {
  #Initialize a vector to hold each rolling sum
  rolling_mean <- vector(mode="numeric")
  #For each set of days
  for(i in 1:(nrow(dailyData)-days)) {
    rolling_mean[length(rolling_mean) + 1] <- mean(dailyData[i:(i+(days-1)),"discharge"])
  }
  #Return the lowest mean
  return(rolling_mean)
}

#Find the rolling 30 day means for each site
kill_roll30 <- rollX(killFlow[killFlow$year==2004,], 30)
stateline_roll30 <- rollX(statelineFlow[statelineFlow$year==2004,], 30)
overland_roll30 <- rollX(overlandFlow[overlandFlow$year==2004,], 30)

#Find the minimum and maximum rolling 30 day mean for each
kill_MIN30DAY <- min(kill_roll30)/kill_drainage
stateline_MIN30DAY <- min(stateline_roll30)/stateline_drainage
overland_MIN30DAY <- min(overland_roll30)/overland_drainage

#Print the metrics
print(kill_MIN30DAY)
```

```
[1] 0.05973783
```

```r
print(stateline_MIN30DAY)
```

```
[1] 0.6659394
```

```
print(overland_MIN30DAY)
```

```
[1] 0.7255639
```

MIN_90DAY

```
#Define a function for finding the rolling mean for X number of days
rollX <- function(dailyData, days) {
  #Initialize a vector to hold each rolling sum
  rolling_mean <- vector(mode="numeric")
  #For each set of days
  for(i in 1:(nrow(dailyData)-days)) {
    rolling_mean[length(rolling_mean) + 1] <- mean(dailyData[i:(i+(days-1)),"discharge"])
  }
  #Return the lowest mean
  return(rolling_mean)
}

#Find the rolling 90 day means for each site
kill_roll90 <- rollX(killFlow[killFlow$year==2004,], 90)
stateline_roll90 <- rollX(statelineFlow[statelineFlow$year==2004,], 90)
overland_roll90 <- rollX(overlandFlow[overlandFlow$year==2004,], 90)

#Find the minimum and maximum rolling 90 day mean for each
kill_MIN90DAY <- min(kill_roll90)/kill_drainage
stateline_MIN90DAY <- min(stateline_roll90)/stateline_drainage
overland_MIN90DAY <- min(overland_roll90)/overland_drainage

#Print the metrics
print(kill_MIN90DAY)
```

```
[1] 0.284434
```

```
print(stateline_MIN90DAY)
```

```
[1] 1.122695
```

```
print(overland_MIN90DAY)
```

```
[1] 1.185714
```

PUL_NO_P10, PUL_DUR_P10, PUL_MAG_P10

```
#Define a function outputting 3 statistics related to the p10
##Number of flow pulses less than the 10th percentile
##Average duration (in days) of flow pulses less than the 10th percentile
##Average magnitude of flow pulses less than the 10th percentile
p10Stats <- function(dailyData, p10) {
```

```
  inPulse <- dailyData$discharge < p10
  inPulse <- as.numeric(inPulse)
  diff <- inPulse[1]
  for(i in 2:length(inPulse)) {
    diff[i] <- inPulse[i] - inPulse[i-1]
  }
  nPulses <- length(diff[diff==1])
  magnitude <- mean(dailyData$discharge[as.logical(inPulse)])
  duration <- sum(inPulse)/nPulses
  output <- c(nPulses, magnitude, duration)
  return(output)

}

#For each site, calculate the set of p10 statistics, keep in mind the _P10P50P90 vector
ns the
#p10, p50 and p90 for each site
kill_p10output <- p10Stats(killFlow[killFlow$year==2004,], kill_P10P50P90[1])
stateline_p10output <- p10Stats(statelineFlow[statelineFlow$year==2004,], stateline_P10P5
0P90[1])
overland_p10output <- p10Stats(overlandFlow[overlandFlow$year==2004,], overland_P10P50P90
[1])

kill_PUL_NO_P10 <- kill_p10output[1]
kill_PUL_DUR_P10 <- kill_p10output[3]
kill_PUL_MAG_P10 <- kill_p10output[2]/kill_drainage

stateline_PUL_NO_P10 <- stateline_p10output[1]
stateline_PUL_DUR_P10 <- stateline_p10output[3]
stateline_PUL_MAG_P10 <- stateline_p10output[2]/stateline_drainage

overland_PUL_NO_P10 <- overland_p10output[1]
overland_PUL_DUR_P10 <- overland_p10output[3]
overland_PUL_MAG_P10 <- overland_p10output[2]/overland_drainage

print(c(kill_PUL_NO_P10, kill_PUL_DUR_P10, kill_PUL_MAG_P10))
```

```
[1] 2.00000000 2.00000000 0.02294007
```

```
print(c(stateline_PUL_NO_P10, stateline_PUL_DUR_P10, stateline_PUL_MAG_P10))
```

```
[1] 3.0000000 1.0000000 0.3090749
```

```
print(c(overland_PUL_NO_P10, overland_PUL_DUR_P10, overland_PUL_MAG_P10))
```

```
[1] 4.0000000 2.0000000 0.3909774
```

PUL_NO_P90, PUL_DUR_P90, PUL_MAG_P90

```
#Define a function outputting 3 statistics related to the p90
##Number of flow pulses greater than the 90th percentile
```

```r
##Average duration (in days) of flow pulses greater than the 90th percentile
##Average magnitude of flow pulses greater than the 90th percentile
p90Stats <- function(dailyData, p90) {

  inPulse <- dailyData$discharge > p90
  inPulse <- as.numeric(inPulse)
  diff <- inPulse[1]
  for(i in 2:length(inPulse)) {
    diff[i] <- inPulse[i] - inPulse[i-1]
  }
  nPulses <- length(diff[diff==1])
  magnitude <- mean(dailyData$discharge[as.logical(inPulse)])
  duration <- sum(inPulse)/nPulses
  output <- c(nPulses, magnitude, duration)
  return(output)

}

#For each site, calculate the set of p90 stats, keep in mind the _P10P50P90 vector contai
ns the
#p10, p50 and p90 for each site
kill_p90output <- p90Stats(killFlow[killFlow$year==2004,], kill_P10P50P90[3])
stateline_p90output <- p90Stats(statelineFlow[statelineFlow$year==2004,], stateline_P10P5
0P90[3])
overland_p90output <- p90Stats(overlandFlow[overlandFlow$year==2004,], overland_P10P50P90
[3])

kill_PUL_NO_P90 <- kill_p90output[1]
kill_PUL_DUR_P90 <- kill_p90output[3]
kill_PUL_MAG_P90 <- kill_p90output[2]/kill_drainage

stateline_PUL_NO_P90 <- stateline_p90output[1]
stateline_PUL_DUR_P90 <- stateline_p90output[3]
stateline_PUL_MAG_P90 <- stateline_p90output[2]/stateline_drainage

overland_PUL_NO_P90 <- overland_p90output[1]
overland_PUL_DUR_P90 <- overland_p90output[3]
overland_PUL_MAG_P90 <- overland_p90output[2]/overland_drainage

print(c(kill_PUL_NO_P90, kill_PUL_DUR_P90, kill_PUL_MAG_P90))

[1] 15.000000  2.066667  5.006403

print(c(stateline_PUL_NO_P90, stateline_PUL_DUR_P90, stateline_PUL_MAG_P90))

[1] 27.000000  1.222222 10.101955

print(c(overland_PUL_NO_P90, overland_PUL_DUR_P90, overland_PUL_MAG_P90))

[1] 27.000000  1.370370  9.240195
```

PEAK_NUM, PEAK_MAG

```r
#Define a function for calculating peak statistics

#Output number of peaks (above peak threshold - peakThresh)
peakStats <- function(dailyData, peakThresh) {
  inPeak <- dailyData$discharge > peakThresh
  inPeak <- as.numeric(inPeak)
  diff <- inPeak[1]
  for(i in 2:length(inPeak)) {
    diff[i] <- inPeak[i] - inPeak[i-1]
  }
  nPeaks <- length(diff[diff==1])
  magnitude <- mean(dailyData$discharge[as.logical(inPeak)])
  output <- c(nPeaks, magnitude)
  return(output)
}

#Calculate peak stats for each
kill_peakOut <- peakStats(killFlow[killFlow$year==2004,], peakThresh = (kill_P10P50P90[2]
* 3))
stateline_peakOut <- peakStats(statelineFlow[statelineFlow$year==2004,], peakThresh = (st
ateline_P10P50P90[2] * 3))
overland_peakOut <- peakStats(overlandFlow[overlandFlow$year==2004,], peakThresh = (overl
and_P10P50P90[2] * 3))

kill_PEAK_NUM <- kill_peakOut[1]
kill_PEAK_MAG <- kill_peakOut[2]/kill_drainage

stateline_PEAK_NUM <- stateline_peakOut[1]
stateline_PEAK_MAG <- stateline_peakOut[2]/stateline_drainage

overland_PEAK_NUM <- overland_peakOut[1]
overland_PEAK_MAG <- overland_peakOut[2]/overland_drainage

print(c(kill_PEAK_NUM, kill_PEAK_MAG))

[1] 22.000000  2.665152

print(c(stateline_PEAK_NUM, stateline_PEAK_MAG))

[1] 33.00000  6.88044

print(c(overland_PEAK_NUM, overland_PEAK_MAG))

[1] 31.000000  7.443835
```

RB_INDEX

```r
#Define a function for calculating Richards-Baker flashiness index
richardsBaker <- function(dailyData) {
  qq <- vector()
  for(i in 2:nrow(dailyData)) {
    qq[length(qq) + 1] <- abs(dailyData$discharge[i] - dailyData$discharge[i-1])
  }
```

```
  rb <- sum(qq)/sum(dailyData$discharge)
  return(rb)
}

kill_rb <- richardsBaker(killFlow[killFlow$year==2004,])
stateline_rb <- richardsBaker(statelineFlow[statelineFlow$year==2004,])
overland_rb <- richardsBaker(overlandFlow[overlandFlow$year==2004,])

print(kill_rb)
```

```
[1] 1.075622
```

```
print(stateline_rb)
```

```
[1] 1.027857
```

```
print(overland_rb)
```

```
[1] 0.9656272
```

TQMEAN

```
#Define a function for calculating TQMEAN
tqmean <- function(dailyData) {
  #Find the annual mean
  Qmean <- mean(dailyData$discharge)
#Find how many days exceed the mean
  nExcede <- sum(dailyData$discharge > Qmean)
#Find the fraction of days that exceed the mean
  tqmean <- nExcede/nrow(dailyData)
  return(tqmean)
}

kill_tq <- tqmean(killFlow[killFlow$year==2004,])
stateline_tq <- tqmean(statelineFlow[statelineFlow$year==2004,])
overland_tq <- tqmean(overlandFlow[overlandFlow$year==2004,])

print(kill_tq)
```

```
[1] 0.147541
```

```
print(stateline_tq)
```

```
[1] 0.1666667
```

```
print(overland_tq)
```

```
[1] 0.1721311
```

RISE_RATE, FALL_RATE

```r
#Define a function to calculate the rise and fall rate
riseFallRate <- function(dailyData) {
  change <- vector()
  for(i in 2:nrow(dailyData)) {
    change[length(change) + 1] <- dailyData$discharge[i] - dailyData$discharge[i-1]
  }
  rises <- change[change>0]
  falls <- abs(change[change<0])
  riseRate <- median(rises)
  fallRate <- median(falls)
  return(c(riseRate, fallRate))
}

kill_risefall <- riseFallRate(killFlow[killFlow$year==2004,]) / kill_drainage
stateline_risefall <- riseFallRate(statelineFlow[statelineFlow$year==2004,]) / stateline_
drainage
overland_risefall <- riseFallRate(overlandFlow[overlandFlow$year==2004,]) / overland_drai
nage

print(kill_risefall)

[1] 0.03745318 0.03183521

print(stateline_risefall)

[1] 0.5298426 0.1090852

print(overland_risefall)

[1] 0.4511278 0.1428571
```