Homework 10: A Maze of Twisty Little Passages

36-350, Fall 2013

Due at 11:59 pm on Thursday, 21 November 2013

The "page-rank" scheme for ranking web pages says that a page is more important the sooner it is reached, on average, by a random walk over the whole web. This was the idea which launched Google. In this assignment, we will begin looking at the sub-problem of figuring out how long it takes a random walk to go between two arbitrary webpages.

Most problems will begin with a piece of supplied code. Some of these will contain small bugs. You will explain what the code is supposed to do; how it does it; and either how you know it works, or how to fix the bug, as appropriate.

General hints: `?grep, ?regexp, ?regmatches`, lectures 20–22.

All the provided functions are in the accompanying `http://www.stat.cmu.edu/~cshalizi/statcomp/13/hw/10/hw-10.R`.

1. (10)

   ```
   statshome <- paste(readLines("http://www.stat.cmu.edu/",warn=FALSE),
     collapse="")
   ```

2. (5)

   ```
   slurp <- function(url) {
     return(paste(readLines(url,warn=FALSE), collapse=" "))
   }
   ```

3. (15)

   ```
   raise.anchors <- function(text) {
     match.locations <- gregexpr('href=".+?"',text,useBytes=TRUE,ignore.case=TRUE)
     return(regmatches(text, match.locations)[[1]])
   }
   ```

4. (10)

   ```
   anchors.with.absolute.urls <- function(urls) {
     return(urls[grepl('"http://',urls)])
   }
   ```

5. (15)

```
extract.links <- function(text) {
  anchors <- raise.anchors(text)
  good.anchors <- anchors.with.absolute.urls(anchors)
  url.locations <- gregexpr('http://[^"]+?', good.anchors, ignore.case=TRUE)
  absolute.urls <- regmatches(good.anchors,url.locations)
  absolute.urls <- do.call(c,absolute.urls)
  return(absolute.urls)
}
```

Hint: Look at the result of `extract.links(statshome)`.

6. (5)

```
surf.from <- function(some.urls,home="http://crookedtimber.org/") {
  if (length(some.urls) == 0) {
    return(home)
  } else {
    return(sample(some.urls,size=1))
  }
}
```

7. (15)

```
surf.to <- function(origin,target,max.iter=20,verbose=TRUE,...) {
  url <- origin
  count <- 0
  while ( !(grepl(target, url, ignore.case=TRUE)) &&(count < max.iter)) {
    text <- slurp(url)
    outgoing.urls <- extract.links(text)
    url <- surf.from(outgoing.urls,...)
    if(verbose) { cat(paste("surfing to",url,"\n")) }
    count <- count+1
  }
  return(count)
}
```

8. (5) Run this several times, and describe what happens. (As usual, raw R output does not count as "describing what happens".)

```
surf.to(origin="http://crookedtimber.org/",target="powells.com")
```

9. (15)

```
surf.to.mk2 <- function(origin,target,max.iter=20,verbose=TRUE,...) {
  url <- origin
  last_url <- url
  count <- 0
  while ( !(grepl(target, url, ignore.case=TRUE)) &&(count < max.iter)) {
    text <- try(slurp(url))                    # Error handling
    if (class(text) == "try-error") {          # If slurp() yields an error
      url <- last_url
    } else {                                   # If slurp() worked OK
      last_url <- url
      outgoing.urls <- extract.links(text)
      url <- surf.from(outgoing.urls,...)
      if(verbose) { cat(paste("surfing to",url,"\n")) }
      count <- count+1
    }
  }
  return(count)
}
```

10. (5) Run this several times and describe what happens.

```
surf.to.mk2(origin="http://crookedtimber.org/",target="powells.com")
```