MFE R Programming Workshop

Week 4

Brett Dunn and Mahyar Kargar

Fall 2017

Introduction

Questions

Any questions before we start?

Overview of Week 4

- Strings
- Dates
- Lab

Strings

Strings

- A string is a sequence of characters.
- In R, a sting falls in the character class.

```r
mystring <- "Hello"
str(mystring)
```

```
##  chr "Hello"
```

- Character vectors are created like numeric vectors.

```r
myvec <- c("Hello", "Goodbye")
str(myvec)
```

```
##  chr [1:2] "Hello" "Goodbye"
```

Manipulating Strings

- `R` provides many functions to manipulate strings.
  - `grep()`: Searches for a substring, like the Linux command of the same name.
  - `nchar()`: Finds the length of a string.
  - `paste()` and `paste0()`: Assembles a string from parts.
  - `sprintf()`: Assembles a string from parts.
  - `substr()`: Extracts a substring.
  - `strsplit()`: Splits a string into substrings.
- Hadley Wickham's `stringr` package provides additional functions for using regular expressions and examining text data.

grep()

- The call grep(pattern,x) searches for a specified substring pattern in a vector x of strings.
- If x has n elements—that is, it contains n strings—then grep(pattern,x) will return a vector of length up to n. -Each element of this vector will be the index in x at which a match of pattern as a substring of x was found.

```
grep("Pole",c("Equator","North Pole","South Pole"))
```

```
## [1] 2 3
```

nchar()

- The call nchar(x) finds the length of a string x.

```
nchar("South Pole")
```

```
## [1] 10
```

paste()

- The call `paste(...)` concatenates several strings, returning the result in one long string.

```r
paste("North","and","South","Poles")
```

```
## [1] "North and South Poles"
```

```r
paste("North","Pole",sep="")
```

```
## [1] "NorthPole"
```

```r
# paste0 is same as sep="" (more efficient)
paste0("North","Pole") == paste("North","Pole",sep="")
```

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

sprintf()

- The call sprintf(...) assembles a string from parts in a formatted manner.
- Similar to the C function printf.

```r
i <- 8
sprintf("the square of %d is %d",i,i^2)
```

```
## [1] "the square of 8 is 64"
```

substr()

- The call substr(x,start,stop) returns the substring in the given character position range start:stop in the given string x.

```
substring("Equator",3,5)
```

```
## [1] "uat"
```

strsplit()

- The call strsplit(x,split) splits a string x into an R list of substrings based on another string split in x.

```
strsplit("10-05-2017",split="-")
```

```
## [[1]]
## [1] "10"   "05"   "2017"
```

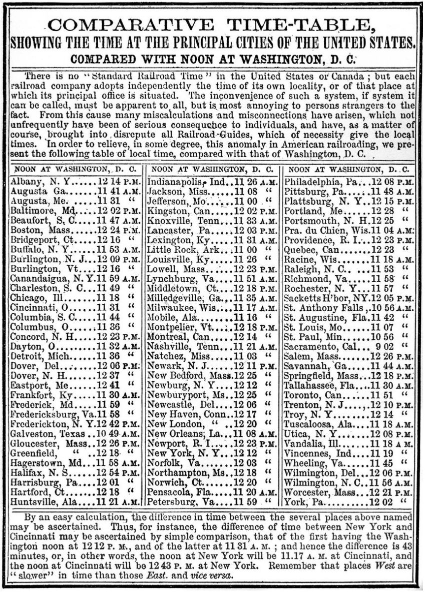
Example: Creating File Names

- Suppose we want to create five files, `q1.pdf` through `q5.pdf`, consisting of histograms of 100 random N(0,i2) variates. We could execute the following code:

```r
for (i in 1:5) {
  fname <- paste("q",i,".pdf")
  pdf(fname)
  hist(rnorm(100,sd=i))
  dev.off()
}
```

Dates

Why do we need date/time classes?



COMPARATIVE TIME-TABLE,

SHOWING THE TIME AT THE PRINCIPAL CITIES OF THE UNITED STATES.

COMPARED WITH NOON AT WASHINGTON, D. C.

There is no "Standard Railroad Time" in the United States or Canada; but each railroad company adopts independently the time of its own locality, or of that place at which its principal office is situated. The inconvenience of such a system, if system it can be called, must be apparent to all, but is most annoying to persons strangers to the fact. From this cause many miscalculations and misconnections have arisen, which not unfrequently have been of serious consequcnce to individuals, and have, as a matter of course, brought into disrcpute all Railroad-Guides, which of necessity give the local times. In order to relieve, in some degree, this anomaly in American railroading, we present the following table of local time, compared with that of Washington, D. C.

NOON AT WASHINGTON, D. C.		NOON AT WASHINGTON, D. C.		NOON AT WASHINGTON, D. C.	
Albany, N. Y	12 14 P.M.	Indianapolis, Ind.	11 26 A.M.	Philadelphia, Pa.	12 08 P.M.
Augusta Ga.	11 41 A.M.	Jackson, Miss.	11 08 "	Pittsburg, Pa.	11 48 A.M.
Augusta, Me.	11 31 "	Jefferson, Mo.	11 00 "	Plattsburg, N. Y.	12 15 P.M.
Baltimore, Md.	12 02 P.M.	Kingston, Can.	12 02 P.M.	Portland, Me.	12 28 "
Beaufort, S. C.	11 47 A.M.	Knoxville, Tenn.	11 33 A.M.	Portsmouth, N. H.	12 25 "
Boston, Mass.	12 24 P.M.	Lancaster, Pa.	12 03 P.M.	Pra. du Chien, Wis.	11 04 A.M.
Bridgeport, Ct.	12 16 "	Lexington, Ky.	11 31 A.M.	Providence, R. I.	12 23 P.M.
Buffalo, N. Y.	11 53 A.M.	Little Rock, Ark.	11 00 "	Quebec, Can.	12 23 "
Burlington, N. J.	12 09 P.M.	Louisville, Ky.	11 26 "	Racine, Wis.	11 18 A.M.
Burlington, Vt.	12 16 "	Lowell, Mass.	12 23 P.M.	Raleigh, N. C.	11 53 "
Canandaigua, N. Y.	11 59 A.M.	Lynchburg, Va.	11 51 A.M.	Richmond, Va.	11 58 "
Charleston, S. C.	11 49 "	Middletown, Ct.	12 18 P.M.	Rochester, N. Y.	11 57 "
Chicago, Ill.	11 18 "	Milledgeville, Ga.	11 35 A.M.	Sacketts H'bor, NY.	12 05 P.M.
Cincinnati, O.	11 31 "	Milwaukee, Wis.	11 17 A.M.	St. Anthony Falls,	10 56 A.M.
Columbia, S. C.	11 44 "	Mobile, Ala.	11 16 "	St. Augustine, Fla.	11 42 "
Columbus, O.	11 36 "	Montpelier, Vt.	12 18 P.M.	St. Louis, Mo.	11 07 "
Concord, N. H.	12 23 P.M.	Montreal, Can.	12 14 "	St. Paul, Min.	10 56 "
Dayton, O.	11 32 A.M.	Nashville, Tenn.	11 21 A.M.	Sacramento, Cal.	9 02 "
Detroit, Mich.	11 36 "	Natchez, Miss.	11 03 "	Salem, Mass.	12 26 P.M.
Dover, Del.	12 06 P.M.	Newark, N. J.	12 11 P.M.	Savannah, Ga.	11 44 A.M.
Dover, N. H.	12 37 "	New Bedford, Mass.	12 25 "	Springfield, Mass.	12 18 P.M.
Eastport, Me.	12 41 "	Newburg, N. Y.	12 12 "	Tallahassee, Fla.	11 30 A.M.
Frankfort, Ky.	11 30 A.M.	Newburyport, Ms.	12 25 "	Toronto, Can.	11 51 "
Frederick, Md.	11 59 "	Newcastle, Del.	12 06 "	Trenton, N. J.	12 10 P.M.
Fredericksburg, Va.	11 58 "	New Haven, Conn.	12 17 "	Troy, N. Y.	12 14 "
Frederickton, N. Y.	12 42 P.M.	New London, "	12 20 "	Tuscaloosa, Ala.	11 18 A.M.
Galveston, Texas	10 49 A.M.	New Orleans, La.	11 08 A.M.	Utica, N. Y.	12 08 P.M.
Gloucester, Mass.	12 26 P.M.	Newport, R. I.	12 23 P.M.	Vandalia, Ill.	11 18 A M.
Greenfield, "	12 18 "	New York, N. Y.	12 12 "	Vincennes, Ind.	11 19 "
Hagerstown, Md.	11 58 A.M.	Norfolk, Va.	12 03 "	Wheeling, Va.	11 45 "
Halifax, N. S.	12 54 P.M.	Northampton, Ms.	12 18 "	Wilmington, Del.	12 06 P.M.
Harrisburg, Pa.	12 01 "	Norwich, Ct.	12 20 "	Wilmington, N. C.	11 56 A.M.
Hartford, Ct.	12 18 "	Pensacola, Fla.	11 20 A.M.	Worcester, Mass.	12 21 P.M.
Huntsville, Ala.	11 21 A.M.	Petersburg, Va.	11 59 "	York, Pa.	12 02 "

By an easy calculation, the difference in time between the several places above named may be ascertained. Thus, for instance, the difference of time between New York and Cincinnati may be ascertained by simple comparison, that of the first having the Washington noon at 12 12 P. M., and of the latter at 11 31 A. M.; and hence the difference is 43 minutes, or, in other words, the noon at New York will be 11.17 A. M. at Cincinnati, and the noon at Cincinnati will be 12 43 P. M. at New York. Remember that places *West* are "slower" in time than those *East.* and *vice versa.*

Date Classes in R

- `Date` is in `yyyy-mm-dd` format and represents the number of days since Jamuary 1, 1970
- `POSIXct` represents the (signed) number of seconds since Jamuary 1, 1970 (in the UTC time zone) as a numeric vector.
- `POSIXlt` is a named list of vectors representing `sec`, `min`, `hour`, `mday`, `mon`, `year`, time zone par maters, and a few other items.

```
x <- Sys.time()  # clock time as a POSIXct object
x; as.numeric(x)
```

```
## [1] "2017-10-23 14:05:35 PDT"
```

```
## [1] 1508792735
```

Creating Dates

- Typically, dates come into R as character strings.
- By default, R assumes the string is in the format `yyyy-mm-dd` or `yyyy-mm-dd`

```r
mychar <- "2017-10-05"
mydate <- as.Date(mychar)
str(mydate)
```

```
##  Date[1:1], format: "2017-10-05"
```

Date Formats

- R can parse many other types of date formats.
- See ?strptime for details.

```
mychar <- "October 5th, 2017"
mydate <- as.Date(mychar, format = "%B %eth, %Y")
str(mydate)
```

```
##  Date[1:1], format: "2017-10-05"
```

Extract Parts of a Date Object

```
mydate <- as.Date("2017-10-05")
weekdays(mydate)
```

```
## [1] "Thursday"
```

```
months(mydate)
```

```
## [1] "October"
```

```
quarters(mydate)
```

```
## [1] "Q4"
```

Generate Regular Sequences of Dates

```
## first days of years
seq(as.Date("2007/1/1"), as.Date("2010/1/1"), "years")
```

```
## [1] "2007-01-01" "2008-01-01" "2009-01-01" "2010-01-01"
```

```
## by month
seq(as.Date("2000/1/1"), by = "month", length.out = 4)
```

```
## [1] "2000-01-01" "2000-02-01" "2000-03-01" "2000-04-01"
```

```
## quarters
seq(as.Date("2000/1/1"), as.Date("2001/1/1"), by = "quarter
```

```
## [1] "2000-01-01" "2000-04-01" "2000-07-01" "2000-10-01"
```

Time Intervals / Differences

- Function difftime calculates a difference of two date/time objects and returns an object of class “difftime” with an attribute indicating the units.

```
time1 <- as.Date("2017-10-05")
time2 <- as.Date("2008-07-08")
time1 - time2
```

```
## Time difference of 3376 days
```

```
difftime(time1, time2, units = "weeks")
```

```
## Time difference of 482.2857 weeks
```

Dates in Microsoft Excel

- Microsoft Excel stores dates as the number of days since Decemeber 31, 1899.
- However, Excel also incorrectly assumes that the year 1900 is a leap year to allow for compatability with Lotus 1-2-3.
- Therefore, for dates after 1901, set the origin to Decemeber 30, 1899 to convert an Excel date to an R date.

```r
as.Date(43013, origin = "1899-12-30")
```

```
## [1] "2017-10-05"
```

Lubridate

Lubridate

- Lubridate is an R package that makes it easier to work with dates and times.
- Lubridate was created by Garrett Grolemund and Hadley Wickham.

```r
# install.packages("lubridate")
library(lubridate)
```

```
##
## Attaching package: 'lubridate'

## The following object is masked from 'package:base':
##
##     date
```

Parse a date

- Lubridate accepts lots of formats

```
ymd("20110604")
```

```
## [1] "2011-06-04"
```

```
mdy("06-04-2011")
```

```
## [1] "2011-06-04"
```

```
dmy("04/06/2011")
```

```
## [1] "2011-06-04"
```

Parse a date and time

```
ymd_hms("2011-06-04 12:00:00", tz = "Pacific/Auckland")
```

```
## [1] "2011-06-04 12:00:00 NZST"
```

Extraction

```r
arrive <- ymd_hms("2011-06-04 12:00:00")
second(arrive)
```

```
## [1] 0
```

```r
second(arrive) <- 25
arrive
```

```
## [1] "2011-06-04 12:00:25 UTC"
```

Intervals

```r
arrive <- ymd_hms("2011-06-04 12:00:00")
leave <- ymd_hms("2011-08-10 14:00:00")
interval(arrive, leave)
```

```
## [1] 2011-06-04 12:00:00 UTC--2011-08-10 14:00:00 UTC
```

Arithmetic

```
mydate <- ymd("20130130")
mydate + days(2)
```

```
## [1] "2013-02-01"
```

```
mydate + months(5)
```

```
## [1] "2013-06-30"
```

Arithmetic

```
mydate <- ymd("20130130")
mydate + days(1:5)
```

```
## [1] "2013-01-31" "2013-02-01" "2013-02-02" "2013-02-03"
```

End of (next) month

```
jan31 <- ymd("2013-01-31")
jan31 + months(1)
```

```
## [1] NA
```

```
ceiling_date(jan31, "month") - days(1)
```

```
## [1] "2013-01-31"
```

```
floor_date(jan31, "month") + months(2) - days(1)
```

```
## [1] "2013-02-28"
```

Lab 2

Let’s work on Lab 2.