

# an introduction to R for epidemiologists

categorical variables, time and regular expressions

Charles DiMaggio, PhD, MPH, PA-C

New York University Department of Surgery and Population Health  
NYU-Bellevue Division of Trauma and Surgical Critical Care  
550 First Avenue, New York, NY 10016

Spring 2015

- [http://www.columbia.edu/~cjd11/charles\\_dimaggio/DIRE/](http://www.columbia.edu/~cjd11/charles_dimaggio/DIRE/)
- [Charles.DiMaggio@nyumc.org](mailto:Charles.DiMaggio@nyumc.org)

# Outline

- 1 categorical variables
  - coding categorical variables
- 2 dates and times
- 3 regular expressions

# Outline

- 1 categorical variables
  - coding categorical variables
- 2 dates and times
- 3 regular expressions

# creating intervals

cut()

- want 7 age categories
  - less than 1, 1 to 4, 5 to 14, 15 to 24, 25 to 44, 45 to 64, older 65

```
ages<-sample(1:100,500, replace=T)
agecat<-cut(ages,breaks = c(0, 1, 5, 15, 25,
  45, 65, 100))
agecat[1:20]
table(agecat)
```

- creates *factor* with 7 levels
- notation (15, 25] interval "open" (parenthesis) on left ( $> 15$ ) and "closed" (bracket) on the right boundary ( $\leq 25$ )
- "right = FALSE" closed on the left and open on the right: [a, b) use

## add labels to intervals

"labels="

clarify the  $(a, b]$  interval bracket notation

```
ages<-sample(1:100,500, replace=T)
agelabs <- c("<1", "1-4", "5-14", "15-24", "25-44",
            "45-64", + "65+")
agecat<-cut(ages,breaks = c(0, 1, 5, 15, 25,
                            45, 65, 100), right=FALSE, labels=agelabs)
agecat[1:20]
table(agecat)
```

# assigning intervals

## indexing

create a categorical *character* vector

```
agecat2<-character(0)
agecat2[ages<1] <- "<1"
agecat2[ages>=1 & ages<5] <- "1-4"
agecat2[ages>=5 & ages<15] <- "5-14"
agecat2[ages>=15 & ages<25] <- "15-24"
agecat2[ages>=25 & ages<45] <- "25-44"
agecat2[ages>=45 & ages<65] <- "45-64"
agecat2[ages>=65] <- "65+"
table(agecat2)
```

# about factors

the good, the bad, and the ugly

- factors are default R categorical variables
  - integers with "names"
  - optimized for modeling functions (e.g. lmer) and graphics (ggplot)
  - preclude having to keep track of assigned levels (vs. data dictionaries)
- but can be a pain
  - behave weirdly
  - get in way of data manipulations (e.g. merges)
  - probably not necessary most of the time
- `read.table` *automatically* converts *all* strings to *factors*
  - make "stringsAsFactors=F" part of your vocabulary (should be default...)
  - `unclass()` returns integers (as.integer does not preserve levels)

# Outline

- 1 categorical variables
  - coding categorical variables
- 2 dates and times
- 3 regular expressions



## setting a referent category

relevel()

```
names <- c("White", "Black", "Latino", "Asian")
ethnic <- sample(names, 100, replace = T)
ethnic <- factor(ethnic, levels = names)
ethnic
levels(ethnic)
ethnic2 <- relevel(ethnic, ref = "Latino")
levels(ethnic)
unclass(ethnic)
unclass(ethnic2)
```

### factors as dummy variables

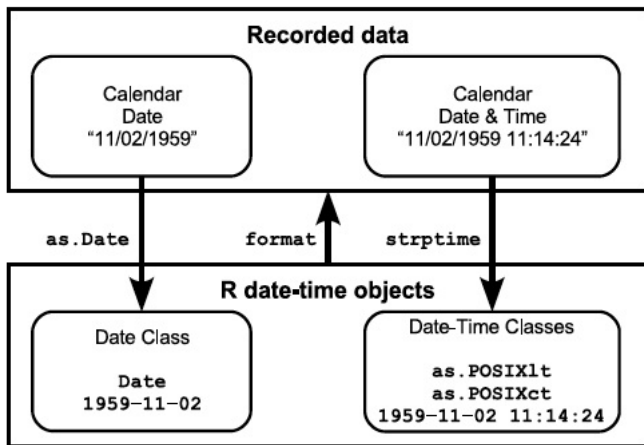
- generally create  $k - 1$  dichotomous variables, each coded 0 or 1
- in R, just create a single factor with the desired number of levels and set the reference level

# Outline

- 1 categorical variables
  - coding categorical variables
- 2 dates and times
- 3 regular expressions

# over view of date objects

from Tomas Aragon, "Applied Epidemiology Using R"



**Fig. 3.4.** Displayed are functions to convert calendar date and time data into R date-time classes (`as.Date`, `strptime`, `as.POSIXlt`, `as.POSIXct`), and the `format` function converts date-time objects into character dates, days, weeks, months, times, etc.

# from days to dates

as.Date()

- *yyyy-mm-dd*
- Julian dates - number of days since January 1, 1970
- specify the format of the input calendar date

```
myDays <- c("10/11/1945", "8/19/2003", "5/15/1964")
myDates<- as.Date(myDays, format = "%m/%d/%Y")
myDates
```
- looks like a character, but is class "date", mode "numeric"
- displayed in a standard format (yyyy-mm-dd)
- `help(strptime)` - to get conversion formats see .

# working with dates

## calculations

```
as.numeric(myDates)
#calculate age (as of today)
today <- Sys.Date()
today
age <- (today - myDates)/365.25
age
age2 <- trunc(as.numeric(age))
age2
```

# working with dates

## conversion formats

- `"%a"` Abbreviated weekday name.
- `"%A"` Full weekday name.
- `"%b"` Abbreviated month name.
- `"%B"` Full month name.
- `"%d"` Day of the month as decimal number (01-31)
- `"%j"` Day of year as decimal number (001-366).
- `"%m"` Month as decimal number (01-12).
- `"%U"` Week of the year as decimal number (00-53) Sunday day 1
- `"%w"` Weekday as decimal number (0-6, Sunday is 0).
- `"%W"` Week of the year as decimal number (00-53) Monday day 1
- `"%y"` Year without century (00-99). (not recommended)

# working with dates

## conversions

```
as.Date("December 8, 1989", format = "%B %d, %Y")
as.Date("12/8/1989", format = "%m/%d/%Y")
as.Date("12/8/89", format = "%m/%d/%y") #caution 2-digit year
as.Date("08Dec1989", format = "%d%b%Y")
as.Date("08Dec89", format = "%d%b%y")
#standard does not require format
as.Date("1989-12-08")
```

# working with dates

## extractions

extracting info from date objects: `weekdays()`, `months()`, `quarters()`, `julian()`

```
weekdays(myDates)
```

```
months(myDates)
```

```
quarters(myDates)
```

```
julian(myDates)
```

## date objects as integers

```
as.Date("2004-01-15") : as.Date("2004-01-23")
```

```
seq(as.Date("2004-01-15"), as.Date("2004-01-18"), by = 1)
```



# from dates to time

`strptime()`

- `strptime()` - accepts both dates and times (HH:MM:SS)

```
dayTime<-c("9/13/1996 5:55:00", "6/11/2002 22:45:00",  
           "6/28/2007 13:20:00")
```

```
dateTime<-strptime(dayTime, "%m/%d/%Y %H:%M:%S")
```

```
dayTime
```

```
dateTime
```

# POSIXlt objects

Portable Operating System Interface, legible time

- `strptime()` produces *named list*, class `POSIXlt`
- `POSIXlt` vector objects:
  - 'sec' (0-61), 'min' (0-59), 'hour' (0-23), 'mday' (1-31), 'mon' (0-11), 'year' (since 1900), 'wday' (0-6, starting on Sunday), 'yday' (0-365), and 'isdst' (DST flag, + if in force, 0 if not, - if unknown)

## working with elements of `POSIXlt` date time list

```
dateTime$min      dateTime$hour
dateTime$mon      dateTime$wday
```

- `POSIXct` - based on continuous time in seconds
- `as.POSIXct()/as.POSIXct()` to coerce

# date-time outputs

`format()`

```
decjan <- seq(as.Date("2003-12-15"), as.Date("2004-01-15"),  
by = 1)
```

```
disease.week <- format(decjan, "%U")
```

- `%U` for weeks starting on Sunday,
- `%W` for weeks starting on Monday

# working with time

## read in time data

- read in 1000 motor vehicle crash times (NYPD data)
- note, even if correct Julian, still need to specify format

```
daytime<-read.csv("~/daytime.csv", stringsAsFactors=F)
```

```
head(daytime)
daytime$x[1:20]
class(daytime$x)
```

```
dateTime<-strptime(daytime$x, "%Y-%M-%d %H:%M:%S")
class(dateTime)
```

# working with time

## extract specific times

- create logical vectors and index
- use weekdays(), months()
- format() for times

```
weekdays(dateTime)=="Sunday"  
months(dateTime)=="July"  
day<- !weekdays(dateTime)=="Sunday" &  
      !weekdays(dateTime)=="Saturday"  
month<-!months(dateTime)=="July" &  
       !months(dateTime)=="August"  
hours <- format(dateTime, "%H") #extract hours from dateTime  
sch.hours<-(hours>="07" & hours<="09" ) |  
            (hours>="02" & hours<="04")
```

# Outline

- 1 categorical variables
  - coding categorical variables
- 2 dates and times
- 3 regular expressions

# grep()

global regular expression print

- search for and return index location of character or string
- search and replace, e.g. code ICD-9 into fewer, mutually exclusive categories
- `grep("x", dataObject)`
- flexible (bewildering?) array of "meta" characters to specify search patterns
- google and StackOverflow are your friends...

# 5 kinds of searches

and some meta characters

- character - .
- class - group of characters [ ]
- concatenation - string together, e.g. a word
- repetition - ? (optional or once) \* (absent or any number) + (at least once)
- alternation - Or statement |



## some "metacharacters" (to control matches)

- `^` 2 possible uses, beginning of a string vs. NOT if inside character class brackets
- `$` - end of a string
- `.` any single character (except a newline)
- `?` occurring once, or not at all
- `+` at least once
- `*` any number of times
- `.*` any number of characters other than a newline
- `0-9, a-z, A-Z` shortcuts for ranges of numbers or letters
- `\` escape, returns literal for meta-characters

# match a single character

## metacharacters

```
vec1 <- c("x", "xa bc", "abc", "ax bc", "ab xc", "ab cx")
```

```
grep("x", vec1) # returns integer vector matches  
vec1[grep("x", vec1)] #index by position
```

```
grep("^x", vec1) # caret ^ matches beginning of line  
vec1[grep("^x", vec1)]
```

```
vec1[grep("x$", vec1)] # $ metacharacter for end of line
```

```
# front space for beginning of a word, but not beginning of a line  
vec1[grep(" x", vec1)]
```

```
# back space for end of a word, but not end of a line  
vec1[grep("x ", vec1)]
```

```
# period matches any single character, including a space.  
vec1[grep(".bc", vec1)]
```

# match any single character from among a list

enclose in brackets

- "[fhr]" matches f, h, or r
- combine with metacharacters for more specificity
- "^ [fhr]" f, h, or r at beginning of a line
 

```
vec2 <- c("fat", "bar", "rat", "elf", "mach", "hat")
vec2[grep("^ [fhr]", vec2)]
```
- ^ inside brackets (character class) is a "not" operator
- "^ [^ fhr]" any single character at the beginning of a line except f, h, or r
 

```
vec2[grep("^[^fhr]", vec2)]
```
- any first character, followed by any character except a, and followed by any character one or more times
 

```
vec2[grep("^.[^a].+", vec2)]
```

## ranges and predefined character classes

[0-9] single digit from 0 to 9

[A-Z] single letter from A to Z ([a-z] lower case)

[0-9A-Za-z] any single alphanumeric character

[:lower:] - lower-case letters ([a-z])

[:upper:] - upper-case letters ([A-Z])

[:alpha:] - alphabetic characters ([A-Za-z])

[:digit:] - digits ([0-9])

[:alnum:] - alphanumeric ([A-Za-z0-9])

[:punct:] - punctuation: ! " # \$ % & ' ( ) \* + , - . /  
: ; < = > ? @ [ \ ] ^ { | } ~

[:graph:] - graphical characters ([:alnum:][:punct:])

[:space:] - space characters: tab, newline, vertical tab,  
form feed, carriage return, and space

# concatenation

combining single characters to match patterns

- find the words "fat", "hat", or "rat"  

```
vec3 <- c("fat","bar","rat","fat boy","elf","mach","hat")  
vec3[grep("[fhr]at$", vec3)]
```
- find words that start with "c" or "t" followed by "a" then followed by "b" or "r"  

```
vec4 <- c("cab", "carat","tar","bar","tab","batboy","care")  
vec4[grep("[ct]a[br]", vec4)]
```
- just three-letter words  

```
vec4[grep("^...$", vec4)]
```
- any word with an "f" and a "t" separated by a single character  

```
vec5 <- c("fate","rat","fit","bat","futbol")  
vec5[grep("f.t", vec5)]
```

# repetition

specify times single character or match pattern repeated

- ? optional pattern, matched at most once, or not at all
- + matched one or more times
- \* any number (zero or more) times
- {n} exactly n times  
`vec4[grep("^[[[:alpha:]]]{3}$", vec4)]`
- {n,} n or more times
- {n, m} at least n times, but not more than m times
- match single, isolated words that start with "f" or "F", followed by one or more of any character, and ending with "t"

```
vec6 <- c("fat","fate","feat","bat","Fahrenheit","bat","foot")
vec6[grep("^[fF].+t$", vec6)]
```

# alternation ("OR")

infix operator | to match from among two or more regular expressions

- ICD-10 hepatitis B codes: B16, B160, B161, B162, B163, B164, B165, B167, B168, B169, B17, B170, B172, B178, B18, B180, B181, B188, B189

- grep ICD10 hepatitis B deaths:

```
dx1 <- c("B16", "B160", "B161", "B162",
        "B163", "B164", "B165", "B167", "B168",
        "B169", "B17", "B170", "B172", "B178",
        "B18", "B180", "B181", "B188", "B189")
```

```
grep("^B16[0-9]?|^B17[0,2,8]?|^B18[0,1,8,9]?$", dx)
```

- create an indicator

```
deathDat$hepB <- "No" #create new field set to no
get.hepB<-grep("^B16[0-9]?|^B17[0,2,8]?|^B18[0,1,8,9]?$",
              deathDat$icd10)
deatgDat$hepB[get.hepB] <- "Yes" # use index to assign yes
```

## other regular expression functions

explore on your own...

- `regexpr()` - similar to `grep`, returns integer vectors with detailed information for the first occurrence of a pattern match within text string elements of a character vector
- `gregexpr()` - similar to `regexpr` but returns a list with detailed information for the multiple occurrences of a pattern match within text string elements of a character vector
- `sub()` - searches and replaces the first occurrence of a pattern match within text string elements of a character vector
- `gsub()` - searches and replaces multiple occurrences of a pattern match within text string elements of a character vector