Assignment 3

Nathan Underwood

2023-10-24

Chapter 11

Question 1

For the following regular expression, explain in words what it matches on. Then add test strings to demonstrate that it in fact does match on the pattern you claim it does. Make sure that your test set of strings has several examples that match as well as several that do not. *If you copy the Rmarkdown code for these exercises directly from my source pages, make sure to remove the `eval=FALSE` from the R-chunk headers.* a) This regular expression matches any string that contains the letter 'a' in it.

```
strings <- c("123", "abc", "this is a string", "a_b_c")
data.frame( string = strings ) %>%
  mutate( result = str_detect(string, 'a') )
```

```
##             string result
## 1              123  FALSE
## 2              abc   TRUE
## 3 this is a string   TRUE
## 4            a_b_c   TRUE
```

```
b) This regular expression matches any string that contains the letters 'a' and 'b' right
next to each other.
```

```
strings <- c("123", "abc", "this is a string", "a_b_c")
data.frame( string = strings ) %>%
  mutate( result = str_detect(string, 'ab') )
```

```
##             string result
## 1              123  FALSE
## 2              abc   TRUE
## 3 this is a string  FALSE
## 4            a_b_c  FALSE
```

```
c)  This regular expression matches any string that contains either an 'a', 'b', or 'c'.
```

```
strings <- c("123", "abc", "this is a string", "a_b_c")
data.frame( string = strings ) %>%
  mutate( result = str_detect(string, '[ab]') )
```

```
##             string result
## 1              123  FALSE
## 2              abc   TRUE
## 3 this is a string   TRUE
## 4            a_b_c   TRUE
```

```
d)  This regular expression matches any string that starts with an 'a', 'b', or 'c'.
```

```r
strings <- c("123", "abc", "this is a string", "a_b_c")
data.frame( string = strings ) %>%
  mutate( result = str_detect(string, '^[ab]') )
```

```
##             string result
## 1              123  FALSE
## 2              abc   TRUE
## 3 this is a string  FALSE
## 4            a_b_c   TRUE
```

e) This regular expression matches any string that has the following in this particular order: 1) at least one number; 2) exactly one white space; and 3) and an 'a' or an 'A'.

```r
strings <- c("11 a", "   aA1", "   1aA1", "1 aA", "1 1aA", "11 aA")
data.frame( string = strings ) %>%
  mutate( result = str_detect(string, '\\d+\\s[aA]') )
```

```
##    string result
## 1    11 a   TRUE
## 2     aA1  FALSE
## 3    1aA1  FALSE
## 4    1 aA   TRUE
## 5   1 1aA  FALSE
## 6   11 aA   TRUE
```

f) This regular expression matches any string that has the following in this particular order: 1) at least one number; and 2) and an 'a' or an 'A'. Optionally, there can be white spaces between the digits and the letters.

```r
strings <- c("11a", "   aA 1", "   1aA1", "1 aA", "1 1aA", "1 1")
data.frame( string = strings ) %>%
  mutate( result = str_detect(string, '\\d+\\s*[aA]') )
```

```
##    string result
## 1     11a   TRUE
## 2    aA 1  FALSE
## 3    1aA1   TRUE
## 4    1 aA   TRUE
## 5   1 1aA   TRUE
## 6     1 1  FALSE
```

g) This regular expression matches any string with any number of characters (including zero!).

```r
strings <- c('', "abcdefg", "1000", "aljsfhdg.2uig4")
data.frame( string = strings ) %>%
  mutate( result = str_detect(string, '.*') )
```

```
##           string result
## 1                  TRUE
## 2        abcdefg   TRUE
## 3           1000   TRUE
## 4 aljsfhdg.2uig4   TRUE
```

h) This regular expression matches any string that begins with two alphanumerical characters followed by the word "bar". This string must be five characters in length.

```r
strings <- c("1Abar", "1bar", "bar", "babar")
data.frame( string = strings ) %>%
```

```
  mutate( result = str_detect(string, '^\\w{2}bar') )
```

```
##   string result
## 1  1Abar   TRUE
## 2   1bar  FALSE
## 3    bar  FALSE
## 4  babar   TRUE
```

```
i) This regular expression matches any string that is either 1) a string from part H;
or 2) or the word "foobar" with a period stuck in the middle.
```

```
strings <- c("foo.bar", "foolbar", "fobar", "1bar", "bar", "babar")
data.frame( string = strings ) %>%
  mutate( result = str_detect(string, '(foo\\.bar)|(^\\w{2}bar)') )
```

```
##    string result
## 1 foo.bar   TRUE
## 2 foolbar  FALSE
## 3   fobar   TRUE
## 4    1bar  FALSE
## 5     bar  FALSE
## 6   babar   TRUE
```

Question 2

The following file names were used in a camera trap study. The S number represents the site, P is the plot within a site, C is the camera number within the plot, the first string of numbers is the YearMonthDay and the second string of numbers is the HourMinuteSecond.

```
file.names <- c( 'S123.P2.C10_20120621_213422.jpg',
                 'S10.P1.C1_20120622_050148.jpg',
                 'S187.P2.C2_20120702_023501.jpg')
```

```
Produce a data frame with columns corresponding to the `site`, `plot`, `camera`, `year`,
`month`, `day`, `hour`, `minute`, and `second` for these three file names. So we want
to produce code that will create the data frame:
```

```
file <-
  str_replace_all(file.names, pattern = '_', replacement = '.') %>%
  str_split_fixed(pattern = '\\.', n = 6)

Year <- str_sub(file[,4], start = 1, end = 4)
Month <- str_sub(file[,4], start = 5, end = 6)
Day <- str_sub(file[,4], start = 7, end = 8)
Hour <- str_sub(file[,5], start = 1, end = 2)
Minute <- str_sub(file[,5], start = 3, end = 4)
Second <- str_sub(file[,5], start = 5, end = 6)

file <- data.frame(file) %>%
  select(-X4) %>%
  select(-X5) %>%
  select(-X6) %>%
  mutate(Year = Year) %>%
  mutate(Month = Month) %>%
  mutate(Day = Day) %>%
  mutate(Hour = Hour) %>%
```

```r
  mutate(Minute = Minute) %>%
  mutate(Second = Second) %>%
  rename(Site = X1) %>%
  rename(Plot = X2) %>%
  rename(Camera = X3)

head(file)
```

```
##   Site Plot Camera Year Month Day Hour Minute Second
## 1 S123   P2    C10 2012    06  21   21     34     22
## 2  S10   P1     C1 2012    06  22   05     01     48
## 3 S187   P2     C2 2012    07  02   02     35     01
```

Question 3

The full text from Lincoln's Gettysburg Address is given below. Calculate the mean word length *Note: consider 'battle-field' as one word with 11 letters*).

```r
Gettysburg <- 'Four score and seven years ago our fathers brought forth on this
continent, a new nation, conceived in Liberty, and dedicated to the proposition
that all men are created equal.

Now we are engaged in a great civil war, testing whether that nation, or any
nation so conceived and so dedicated, can long endure. We are met on a great
battle-field of that war. We have come to dedicate a portion of that field, as
a final resting place for those who here gave their lives that that nation might
live. It is altogether fitting and proper that we should do this.

But, in a larger sense, we can not dedicate -- we can not consecrate -- we can
not hallow -- this ground. The brave men, living and dead, who struggled here,
have consecrated it, far above our poor power to add or detract. The world will
little note, nor long remember what we say here, but it can never forget what
they did here. It is for us the living, rather, to be dedicated here to the
unfinished work which they who fought here have thus far so nobly advanced. It
is rather for us to be here dedicated to the great task remaining before us --
that from these honored dead we take increased devotion to that cause for which
they gave the last full measure of devotion -- that we here highly resolve that
these dead shall not have died in vain -- that this nation, under God, shall
have a new birth of freedom -- and that government of the people, by the people,
for the people, shall not perish from the earth.'

GettysburgSplit <-
  Gettysburg %>%
  str_replace_all(pattern = "\\.|\\,|\\\n|-- ", replacement = '') %>%
  str_split(pattern = ' ')

str_length(GettysburgSplit)
```

```
## Warning in stri_length(string): argument is not an atomic vector; coercing
```

```
## [1] 2231
```

```r
meanLength <- (str_length(GettysburgSplit) - 1) / 269 / 2
```

```
## Warning in stri_length(string): argument is not an atomic vector; coercing
```

```r
meanLength
```

```
## [1] 4.144981
```

Chapter 12

Question 1

Convert the following to date or date/time objects. a) September 13, 2010.

```r
mdy("September 13, 2010")
```

```
## [1] "2010-09-13"
```

```
b) Sept 13, 2010.
```

```r
mdy("Sept 13, 2010")
```

```
## Warning: All formats failed to parse. No formats found.
```

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

```
c) Sep 13, 2010.
```

```r
mdy("Sep 13, 2010")
```

```
## [1] "2010-09-13"
```

```
d) S 13, 2010. Comment on the month abbreviation needs.
```

```r
mdy("S 13, 2010")
```

```
## Warning: All formats failed to parse. No formats found.
```

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

```
The `mdy()` function only recognizes certain abbreviations for the month. In this case, 'S'
is not an acceptable abbreviation for September.
e) 07-Dec-1941.
```

```r
dmy("07-Dec-1941")
```

```
## [1] "1941-12-07"
```

```
f) 1-5-1998. Comment on why you might be wrong.
```

```r
dmy("1-5-1998")
```

```
## [1] "1998-05-01"
```

```r
mdy("1-5-1998")
```

```
## [1] "1998-01-05"
```

```
It is hard to tell which of the previous formats is correct because both the month and day
values are numbers below 12. This means that the month could be the day, or the day could
be the month.
g) 21-5-1998. Comment on why you know you are correct.
```

```r
dmy("21-5-1998")
```

```
## [1] "1998-05-21"
```

This one is better than part F because the first entry is a number above 12. This means that the number has to represent days since there is not a 21st month.
h) 2020-May-5 10:30 am

```
ymd_hm("2020-May-5 10:30am")
```

```
## [1] "2020-05-05 10:30:00 UTC"
```

i) 2020-May-5 10:30 am PDT (ex Seattle)

```
ymd_hm("2020-May-5 10:30am", tz = "America/Vancouver")
```

```
## [1] "2020-05-05 10:30:00 PDT"
```

j) 2020-May-5 10:30 am AST (ex Puerto Rico)

```
ymd_hm("2020-May-5 10:30am", tz = "America/Puerto_Rico")
```

```
## [1] "2020-05-05 10:30:00 AST"
```

Question 2

Using just your date of birth (ex Sep 7, 1998) and today's date calculate the following. *Write your code in a manner that the code will work on any date after you were born.*:
a) Calculate the date of your 64th birthday.

```
birthdate <- dmy("17 Dec, 2002")

birthdate + years(64)
```

```
## [1] "2066-12-17"
```

b) Calculate your current age (in years).
Hint: Check your age is calculated correctly if your birthday was yesterday and if it
were tomorrow!

```
age <- interval(birthdate, today()) %>% as.period() %>%  year()

age
```

```
## [1] 20
```

c) Using your result in part (b), calculate the date of your next birthday.

```
nextBirthday <- birthdate + years(age + 1)

nextBirthday
```

```
## [1] "2023-12-17"
```

d) The number of _days_ until your next birthday.

```
(nextBirthday - today()) %>% as.period(unit = "days") %>% day()
```

```
## [1] 54
```

e) The number of _months_ and _days_ until your next birthday.

```
interval(today(), nextBirthday) %>% as.period()
```

```
## [1] "1m 23d 0H 0M 0S"
```

Question 3

Suppose you have arranged for a phone call to be at 3 pm on May 8, 2015 at Arizona time. However, the recipient will be in Auckland, NZ. What time will it be there?

```
dmy_h("8 May, 2015 3pm", tz = "US/Arizona") %>%
  with_tz("Pacific/Auckland")
```

```
## [1] "2015-05-09 10:00:00 NZST"
```

Question 5

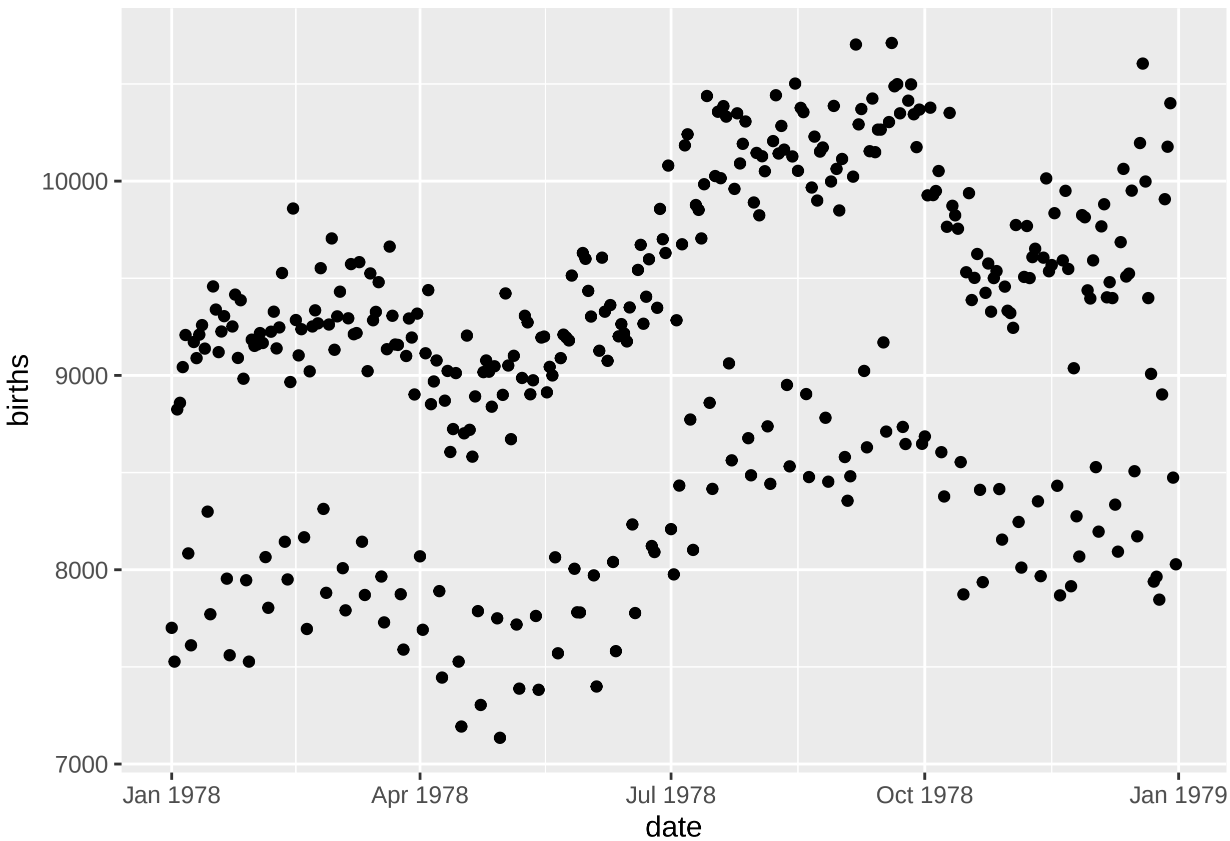
It turns out there is some interesting periodicity regarding the number of births on particular days of the year.
a. Using the `mosaicData` package, load the data set `Births78` which records the number of children born on each day in the United States in 1978. Because this problem is intended to show how to calculate the information using the `date`, remove all the columns *except* `date` and `births`.

```
data("Births78")

ModifiedBirths78 <- Births78 %>%
  select(date, births)
```

```
b. Graph the number of `births` vs the `date` with date on the x-axis.What stands out to
you? Why do you think we have this trend?
```

```
ggplot(ModifiedBirths78, aes(x = date, y = births)) +
  geom_point()
```



There are certain days of the week that have lower birthrates than others. Perhaps people don't want to work on certain days or patients aren't being accepted.

c. To test your assumption, we need to figure out the what day of the week each observation is. Use `dplyr::mutate` to add a new column named `dow` that is the day of the week (Monday, Tuesday, etc). This calculation will involve some function in the `lubridate` package and the `date` column.

```
ModifiedBirths78 <- ModifiedBirths78 %>%
  mutate(dow = wday(date, label = TRUE, abbr = FALSE))
```

```
d. Plot the data with the point color being determined by the day of the week variable.
```

```
ggplot(ModifiedBirths78, aes(x = date, y = births, color = dow)) +
  geom_point()
```