assignment 3

Jacob O

2023-10-17

Question 1

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: a

```
strings <- c('a', 'banana', 'orangutan', 'AAAAA', 'I want a mule', 'no, not the animal', 'you know what
data.frame( string = strings ) %>%
  mutate( result = str_detect(string, 'a') )
```

```
##                   string result
## 1                      a   TRUE
## 2                 banana   TRUE
## 3              orangutan   TRUE
## 4                  AAAAA  FALSE
## 5          I want a mule   TRUE
## 6     no, not the animal   TRUE
## 7  you know what i mean   TRUE
## 8            from moscow  FALSE
## 9                    bee  FALSE
## 10           more stuff  FALSE
## 11                 blAh  FALSE
```

b) This regular expression matches: 'ab'

```
strings <- c('b', 'a', 'christened', 'bayybeee', 'altruism', 'biscuit', 'now I want a biscoff', 'absent
data.frame( string = strings ) %>%
  mutate( result = str_detect(string, 'ab') )
```

```
##                   string result
## 1                      b  FALSE
## 2                      a  FALSE
## 3              christened  FALSE
## 4                bayybeee  FALSE
## 5                altruism  FALSE
## 6                 biscuit  FALSE
## 7  now I want a biscoff  FALSE
## 8                 absent   TRUE
## 9                ablative   TRUE
## 10                   cab   TRUE
```

c) This regular expression matches: a or b

```
strings <- c('b', 'a', 'christened', 'bayybeee', 'altruism', 'biscuit', 'now I want a biscoff', 'cortis
data.frame( string = strings ) %>%
  mutate( result = str_detect(string, '[ab]') )
```

```
##                   string result
## 1                      b   TRUE
## 2                      a   TRUE
## 3             christened  FALSE
## 4               bayybeee   TRUE
## 5               altruism   TRUE
## 6                biscuit   TRUE
## 7  now I want a biscoff   TRUE
## 8               cortisol  FALSE
## 9                 length  FALSE
## 10                ripper  FALSE
```

d) This regular expression matches: a or b at the beginning

```
strings <- c('cab', 'octal', 'obituary', 'apple', 'belch', 'beginning', 'actor')
data.frame( string = strings ) %>%
  mutate( result = str_detect(string, '^[ab]') )
```

```
##       string result
## 1        cab  FALSE
## 2      octal  FALSE
## 3   obituary  FALSE
## 4      apple   TRUE
## 5      belch   TRUE
## 6  beginning   TRUE
## 7      actor   TRUE
```

e) This regular expression matches: 1 or more digits, some whitespace, then an a or A

```
strings <- c('6639 ABQ', 'representin tha ABQ', 'my name is skyler white, yo', '1 a', '1a', '123 b', '1
data.frame( string = strings ) %>%
  mutate( result = str_detect(string, '\\d+\\s[aA]') )
```

```
##                        string result
## 1                    6639 ABQ   TRUE
## 2         representin tha ABQ  FALSE
## 3 my name is skyler white, yo  FALSE
## 4                         1 a   TRUE
## 5                          1a  FALSE
## 6                       123 b  FALSE
## 7                     1     a  FALSE
## 8        8675309 AAAAAAAAAAAA   TRUE
## 9               yeah thats it  FALSE
```

f) This regular expression matches: same as before but the whitespace can be multiple, or 0

```
strings <- c('6639 ABQ', 'representin tha ABQ', 'my name is skyler white, yo', '1 a', '1a', '123 b', '1
data.frame( string = strings ) %>%
  mutate( result = str_detect(string, '\\d+\\s*[aA]') )
```

```
##                        string result
## 1                    6639 ABQ   TRUE
## 2         representin tha ABQ  FALSE
```

```
## 3 my name is skyler white, yo  FALSE
## 4                          1 a   TRUE
## 5                           1a   TRUE
## 6                       123 b  FALSE
## 7        1     a, many spaces   TRUE
## 8        8675309 AAAAAAAAAAAA   TRUE
## 9               yeah thats it  FALSE
```

g) This regular expression matches: literally anything lmao

```
strings <- c('lighter',
'nut',
'location',
'sculpture',
'dealer',
'lamp',
'wolf',
'copy',
'flawed',
'12348127834')
data.frame( string = strings ) %>%
  mutate( result = str_detect(string, '.*') )
```

```
##         string result
## 1      lighter   TRUE
## 2          nut   TRUE
## 3     location   TRUE
## 4    sculpture   TRUE
## 5       dealer   TRUE
## 6         lamp   TRUE
## 7         wolf   TRUE
## 8         copy   TRUE
## 9       flawed   TRUE
## 10 12348127834   TRUE
```

h) This regular expression matches: beginning of string, 2 alphanumerics, then 'bar'

```
strings <- c('54bar', 'babar', '9Sbar', 'lalalal 54bar', '32 bar', '6h ba')
data.frame( string = strings ) %>%
  mutate( result = str_detect(string, '^\\w{2}bar') )
```

```
##          string result
## 1         54bar   TRUE
## 2         babar   TRUE
## 3         9Sbar   TRUE
## 4 lalalal 54bar  FALSE
## 5        32 bar  FALSE
## 6         6h ba  FALSE
```

i) This regular expression matches: 'foo.bar'. or the previous question thing

```
strings <- c('54bar', 'babar', '9Sbar', 'lalalal 54bar', '32 bar', '6h ba', 'foo.bar', 'foobar', 'foo%ba
data.frame( string = strings ) %>%
  mutate( result = str_detect(string, '(foo\\.bar)|(^\\w{2}bar)') )
```

```
##          string result
## 1         54bar   TRUE
## 2         babar   TRUE
```

```
## 3         9Sbar   TRUE
## 4 lalalal 54bar  FALSE
## 5        32 bar  FALSE
## 6         6h ba  FALSE
## 7       foo.bar   TRUE
## 8        foobar  FALSE
## 9       foo%bar  FALSE
```

Question 2

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`


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

```
extractorman <- '^([sS]\\d+)\\.([pP]\\d+)\\.([cC]\\d+)_(\\d{4})(\\d{2})(\\d{2})_(\\d{2})(\\d{2})(\\d{2}
names <- c("Site", "Plot", "Camera", "Year", "Month", "Day", "Hour", "Minute", "Second")
dfman <- NULL
for(i in 1:9) {
  dfman <- cbind(dfman, str_extract(file.names, extractorman, group = i))
}
colnames(dfman) <- names
dfman <- data.frame(dfman)
dfman
```

```
##   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

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*).

```
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.'

yahhhbebe <- str_extract_all(Gettysburg, "(\\w+)-*(\\w+)")
head(yahhhbebe[[1]])
```

```
## [1] "Four"  "score" "and"   "seven" "years" "ago"
```

```
mean(str_length(yahhhbebe[[1]]))
```

```
## [1] 4.329545
```

Question 1

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

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

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

b) Sept 13, 2010.

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

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

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

c) Sep 13, 2010.

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

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

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

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

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

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

It seems that you have to do either the full month name, or a 3 letter abbreviation e) 07-Dec-1941.

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

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

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

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

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

Day vs month is ambiguous. But that's only for europeans and I'm too busy being a free God-fearing, red-blooded, burger-grilling American. God bless America! g) 21-5-1998. Comment on why you know you are correct.

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

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

Day vs month is not ambiguous. there is no month 21. h) 2020-May-5 10:30 am

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

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

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

```
ymd_hm("2020-May-5 10:30 am", tz="PST8PDT")
```

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

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

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

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

Question 2

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.

```
mdy("March 15, 2002") + years(64)
```

```
## [1] "2066-03-15"
```

```
b) Calculate your current age (in years). _Hint: Check your age is calculated correctly if your birthda
```

```
ageYAAAAAHHHH <- year(as.period(mdy("Mar 15, 2002") %--% ymd(lubridate::today())  ))
ageYAAAAAHHHH
```

```
## [1] 21
```

^^^^ worlds most readable single line of code ^^^^ d) Using your result in part (b), calculate the date of your next birthday.

```
birthday<- mdy("Mar 15, 2002")
year(birthday) <- year(birthday) + ageYAAAAAHHHH + 1
birthday
```

```
## [1] "2024-03-15"
```

```
e) The number of _days_ until your next birthday.
```

```
birthday - ymd(lubridate::today())
```

```
## Time difference of 143 days
```

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

```
month(as.period(ymd(lubridate::today()) %--% birthday))
```

```
## [1] 4
```

```
day(as.period(ymd(lubridate::today()) %--% birthday))
```

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

Question 3

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?

```
with_tz(mdy_hm("May 8, 2015 3:00 pm", tz='US/Arizona'), tzone='NZ')
```

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

Question 5

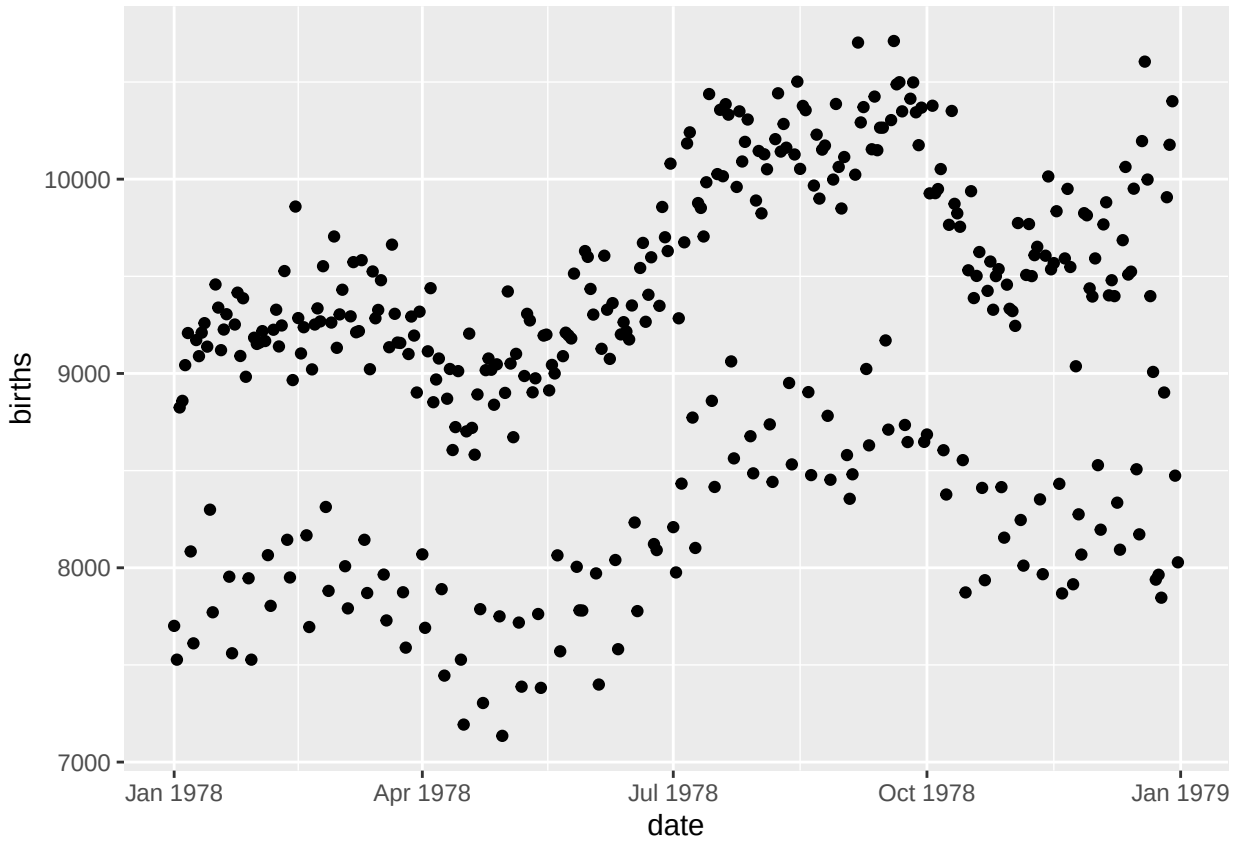
5. It turns out there is some interesting periodicity regarding the number of births on particular days of the year.
   a. Using the `mosaicData` package, load the data set `Births78` which records the number of children born on each day in the United States in 1978. Because this problem is intended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

```
yaboiiii <- mosaicData::Births78 %>%
  select(c("date", "births"))
head(yaboiiii)
```

```
##         date births
## 1 1978-01-01   7701
## 2 1978-01-02   7527
## 3 1978-01-03   8825
## 4 1978-01-04   8859
## 5 1978-01-05   9043
## 6 1978-01-06   9208
```

b. Graph the number of `births` vs the `date` with date on the x-axis. What stands out to you? Why do y

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



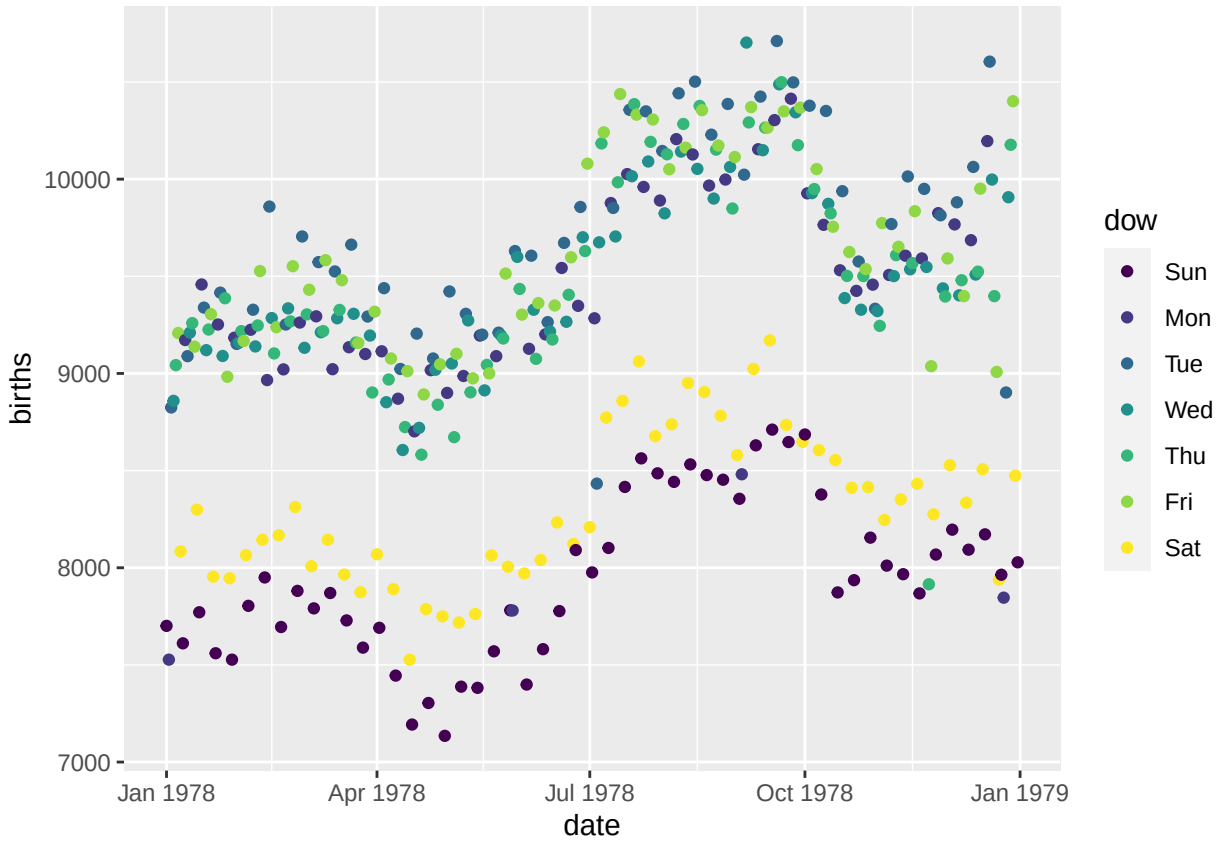
It seems there are two main groups, like its bimodal. I see no reason why it would be more popular to give birth on a certain day of the week, but judging by the next question, I guess that's what we're goin with. c. To test your assumption, we need to figure out the what day of the week each observation 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.

```
yaboiiii <- yaboiiii %>%
  dplyr::mutate(dow = wday(date, label = TRUE))
head(yaboiiii)
```

```
##         date births dow
## 1 1978-01-01   7701 Sun
## 2 1978-01-02   7527 Mon
## 3 1978-01-03   8825 Tue
## 4 1978-01-04   8859 Wed
## 5 1978-01-05   9043 Thu
## 6 1978-01-06   9208 Fri
```

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

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


births
10000
9000
8000
7000
Jan 1978
Apr 1978
Jul 1978
Oct 1978
Jan 1979
date
dow
Sun
Mon
Tue
Wed
Thu
Fri
Sat