Assignment 3

Isaac Davis

October 31, 2023

11.1

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: any string with an "a" in it

```
strings <- c( "Arya", "Jon", "Samwell", "Varys", "Jaime", "Podrick", "Aemon", "Tyrion" )
data.frame( string = strings ) %>%
  mutate( result = str_detect( string, 'a' ) )
```

```
##    string result
## 1    Arya   TRUE
## 2     Jon  FALSE
## 3 Samwell   TRUE
## 4   Varys   TRUE
## 5   Jaime   TRUE
## 6 Podrick  FALSE
## 7   Aemon  FALSE
## 8  Tyrion  FALSE
```

b) This regular expression matches: any string with an "ab" in it

```
# This regular expression matches:  Insert your answer here...
strings <- c( "Arya", "Jon-ab", "Samwell", "Varys-ab", "Jaime", "Podrick", "Aemon", "Tyrion" )
data.frame( string = strings ) %>%
  mutate( result = str_detect( string, 'ab' ) )
```

```
##     string result
## 1     Arya  FALSE
## 2   Jon-ab   TRUE
## 3  Samwell  FALSE
## 4 Varys-ab   TRUE
## 5    Jaime  FALSE
## 6  Podrick  FALSE
## 7    Aemon  FALSE
## 8   Tyrion  FALSE
```

c) This regular expression matches: any string with an "a" or a "b" in it

```
strings <- c( "Arya", "Jon", "Samwell", "Varys", "Jaime", "Podrick", "Aemon", "Tyrion" )
data.frame( string = strings ) %>%
  mutate( result = str_detect( string, '[ab]' ) )
```

```
##    string result
## 1    Arya   TRUE
## 2     Jon  FALSE
## 3 Samwell   TRUE
## 4   Varys   TRUE
## 5   Jaime   TRUE
## 6 Podrick  FALSE
## 7   Aemon  FALSE
## 8  Tyrion  FALSE
```

d) This regular expression matches: any string that starts with an "a" or "b"; note that it is case s

```
strings <- c( "arya", "Jon", "Samwell", "Varys", "Jaime", "Podrick", "aemon", "Tyrion" )
data.frame( string = strings ) %>%
  mutate( result = str_detect( string, '^[ab]' ) )
```

```
##    string result
## 1    arya   TRUE
## 2     Jon  FALSE
## 3 Samwell  FALSE
## 4   Varys  FALSE
## 5   Jaime  FALSE
## 6 Podrick  FALSE
## 7   aemon   TRUE
## 8  Tyrion  FALSE
```

e) This regular expression matches: any string that has one or more digits, any white space attached

```
strings <- c( "1 Arya", "2 Jon", "3 Samwell", "4 Varys", "5 Jaime", "6 Podrick", "7 Aemon", "8 Tyrion"
data.frame( string = strings ) %>%
  mutate( result = str_detect( string, '\\d+\\s[aA]' ) )
```

```
##      string result
## 1    1 Arya   TRUE
## 2     2 Jon  FALSE
## 3 3 Samwell  FALSE
## 4   4 Varys  FALSE
## 5   5 Jaime  FALSE
## 6 6 Podrick  FALSE
## 7   7 Aemon   TRUE
## 8  8 Tyrion  FALSE
```

f) This regular expression matches: any string with one or more integers, there may or may not be any

```
strings <- c( "1-Arya", "2 Jon", "3 Samwell", "4 Varys", "5 Jaime", "6 Podrick", "7 Aemon", "8 Tyrion"
data.frame( string = strings ) %>%
  mutate( result = str_detect( string, '\\d+\\s*[aA]' ) )
```

```
##      string result
## 1    1-Arya  FALSE
## 2     2 Jon  FALSE
## 3 3 Samwell  FALSE
## 4   4 Varys  FALSE
## 5   5 Jaime  FALSE
## 6 6 Podrick  FALSE
## 7   7 Aemon   TRUE
## 8  8 Tyrion  FALSE
```

g) This regular expression matches: literally any string; it means any string that does or does not ha

```
strings <- c( "Arya", "Jon", "Samwell", "Varys", "Jaime", "Podrick", "aemon", "Tyrion", "777", "333", "
data.frame( string = strings ) %>%
  mutate( result = str_detect( string, '.*' ) )
```

```
##     string result
## 1     Arya   TRUE
## 2      Jon   TRUE
## 3  Samwell   TRUE
## 4    Varys   TRUE
## 5    Jaime   TRUE
## 6  Podrick   TRUE
## 7    aemon   TRUE
## 8   Tyrion   TRUE
## 9      777   TRUE
## 10     333   TRUE
## 11           TRUE
```

h) This regular expression matches: any string that starts with two alphanumeric character followed by

```
strings <- c( "Arbarya", "Jbaron", "Sambarwell", "Vabarrys", "Jaime", "Podrick", "Aemon", "Tyrion" )
data.frame( string = strings ) %>%
  mutate( result = str_detect( string, '^\\w{2}bar' ) )
```

```
##       string result
## 1    Arbarya   TRUE
## 2     Jbaron  FALSE
## 3 Sambarwell  FALSE
## 4   Vabarrys   TRUE
## 5      Jaime  FALSE
## 6    Podrick  FALSE
## 7      Aemon  FALSE
## 8     Tyrion  FALSE
```

i) This regular expression matches: any string that is "foo" followed by a "." followed by "bar", or a s

```
strings <- c( "foo-bar", "foo.bar", "foobar", "aabar", "ccbar", "dddbar" )
data.frame( string = strings ) %>%
  mutate( result = str_detect( string, '(foo\\.bar)|(^\\w{2}bar)' ) )
```

```
##    string result
## 1 foo-bar  FALSE
## 2 foo.bar   TRUE
## 3  foobar  FALSE
## 4   aabar   TRUE
## 5   ccbar   TRUE
## 6  dddbar  FALSE
```

11.2

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`

```
 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
```

```
data <- data.frame(
  Site = str_extract( file.names, '(S\\d+)', group=1 ),
  Plot = str_extract( file.names, '(P\\d+)', group=1 ),
  Camera = str_extract( file.names, '(C\\d+)', group=1 ),
  Year = str_extract( file.names, '(20\\d{2})', group=1 ),
  Month = str_extract( file.names, '20\\d{2}(\\d{2})', group=1 ),
  Day = str_extract( file.names, '20\\d{2}\\d{2}(\\d{2})', group=1 ),
  Hour = str_extract( file.names, '20\\d{2}\\d{2}\\d{2}_(\\d{2})', group=1 ),
  Minute = str_extract( file.names, '20\\d{2}\\d{2}\\d{2}_\\d{2}(\\d{2})', group=1 ),
  Second = str_extract( file.names, '20\\d{2}\\d{2}\\d{2}_\\d{2}\\d{2}(\\d{2})', group=1 )
)
data
```

```
##   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
```

11.3

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.'
# method 1
words <- str_extract_all( Gettysburg, '[A-Za-z]+\\-*[a-z]+' )
sum <- 0
for( i in 1:length( words[[1]] ) )
```

```
{
  for( ii in 1:length( words[[1]][i] ) )
  {
    if( words[[1]][i] != '-' )
    {
      sum <- sum + 1
    }
  }
}
mean( str_length( words[[1]] ) )
```

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

```
# method 2
words <- str_c( words[[1]], collapse='' )
spaces = str_extract_all( Gettysburg, '\\s' )
spaces = spaces[[1]]
spaces = spaces[ spaces != '\n' ]
spaces <- str_c( spaces, collapse='' )
str_length( words ) / str_length( spaces )
```

```
## [1] 4.156364
```

12.1

1. Convert the following to date or date/time objects.
   a) September 13, 2010.
   b) Sept 13, 2010.
   c) Sep 13, 2010.
   d) S 13, 2010. Comment on the month abbreviation needs. if you use mdy with multiple inputs it works, but if you single it them out individually, the s breaks it because it doesnt know what s means, also, another note, if i were to put j in the multiple inputs, it would have to guess june or july because it wouldnt know

```
# a b c d
mdy( 'September 13, 2010', 'Sept 13, 2010', 'Sep 13, 2010', 'S 13, 2010' )
```

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

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

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

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

```
e) 07-Dec-1941.
f) 1-5-1998. Comment on why you might be wrong.
```

i dont know if 1 or 5 is the month, so i had to guess g) 21-5-1998. Comment on why you know you are correct. however 21 can only be the day

```
# e f g
dmy( '07-Dec-1941', '1-5-1998', '21-5-1998' )
```

```
## [1] "1941-12-07" "1998-05-01" "1998-05-21"
```

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

```
# h
ymd_hm( '2020-May-5 5:30 PM' )
```

```
## [1] "2020-05-05 17:30:00 UTC"
```

```
# i
ymd_hm( '2020-May-5 5:30 PM', tz='PST8PDT' )
```

```
## [1] "2020-05-05 17:30:00 PDT"
```

```
# j
ymd_hm( '2020-May-5 5:30 PM', tz='America/Puerto_Rico' )
```

```
## [1] "2020-05-05 17:30:00 AST"
```

12.2

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.
   b) Calculate your current age (in years). *Hint: Check your age is calculated correctly if your birthday was yesterday and if it were tomorrow!*
   c) Using your result in part (b), calculate the date of your next birthday.
   d) The number of *days* until your next birthday.
   e) The number of *months* and *days* until your next birthday.

```
# a
date = ymd( '01 01 01' )
date + years( 64 )
```

```
## [1] "2065-01-01"
```

```
# b
curDate = mdy( format( Sys.time(),  '%B %d, %Y' ) )
as.period( interval( date, curDate ) )
```

```
## [1] "22y 9m 30d 0H 0M 0S"
```

```
# d
nextDate = date + years( year( as.period( interval( date, mdy( format( Sys.time(),  '%B %d, %Y' ) ) ) ) )
nextDate
```

```
## [1] "2024-01-01"
```

```
# e
rangeDays = as.period( curDate %--% nextDate, unit='days' )
paste( "Days:", day( rangeDays ) )
```

```
## [1] "Days: 62"
```

```
# f
range = as.period( curDate %--% nextDate )
paste( paste( paste( "Months:", month( range ) ), "|" ), paste( "Days:", day( range ) ) )
```

```
## [1] "Months: 2 | Days: 1"
```

12.3

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?

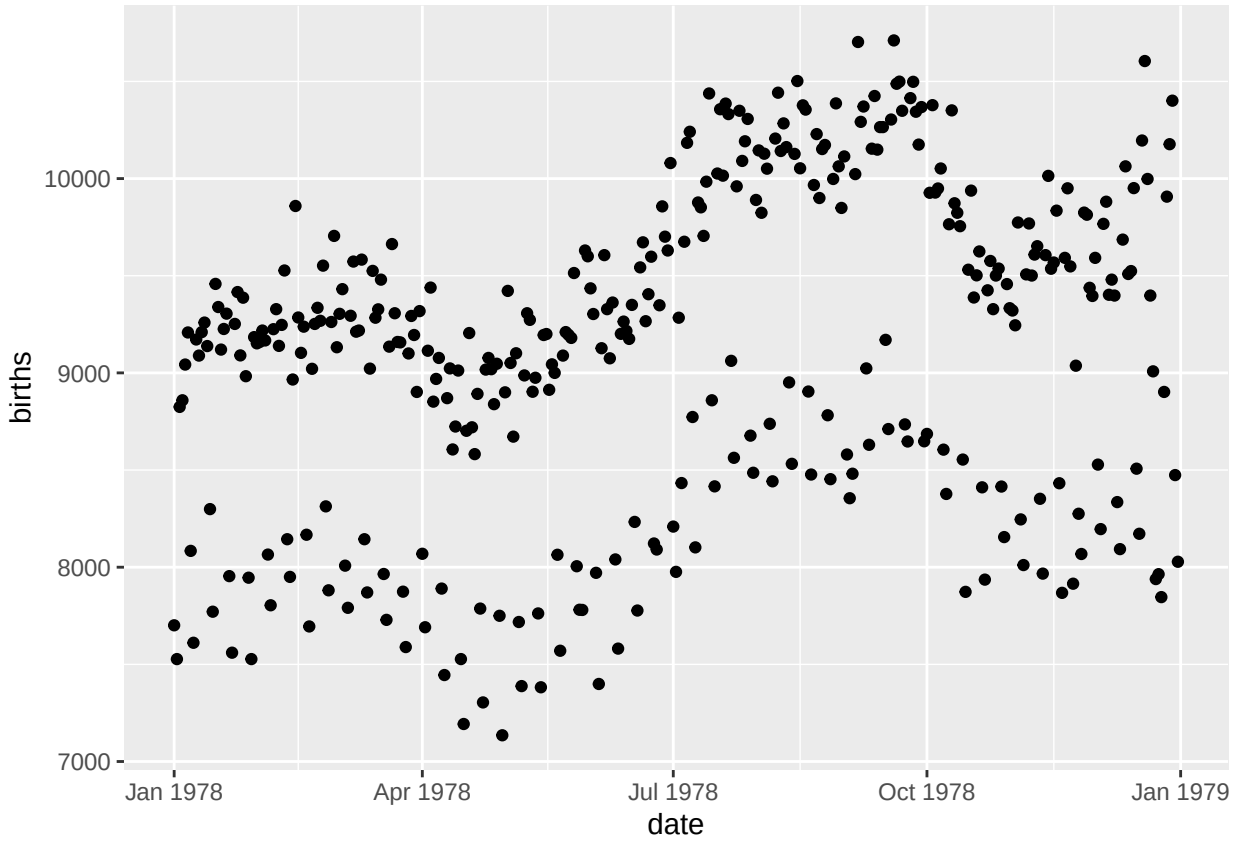
```
with_tz( with_tz( '2015-5-8 15:00', tzones = 'MST' ), "NZ" )
```

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

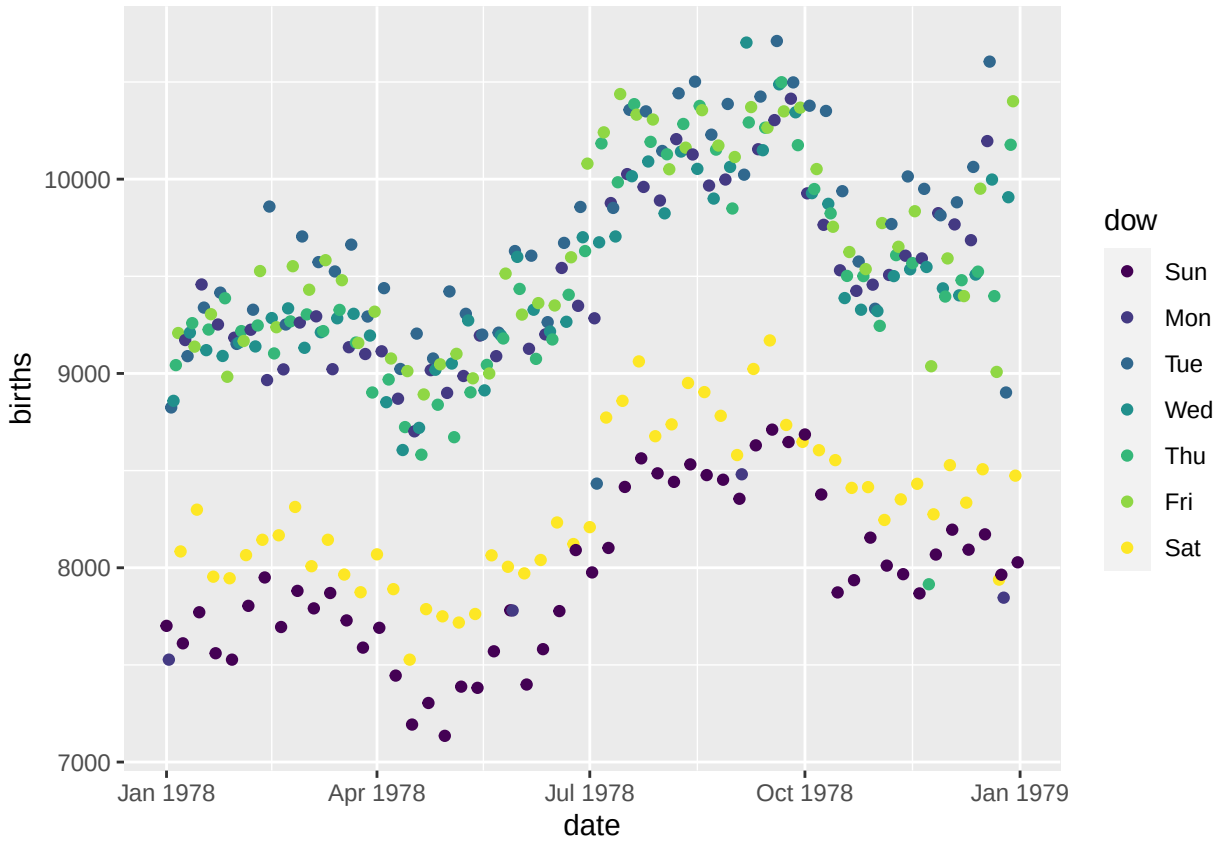
12.5

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

```
# a
data <- mosaicData::Births78 %>% select( date, births )
# b
ggplot( data=data, aes( x=date, y=births ) ) + geom_point()
```




```
# c
data <- data %>% mutate( dow = wday( date, label=TRUE ) )
# d
ggplot( data=data, aes( x=date, y=births ) ) + geom_point( aes( 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