STA445_Assignment3

Nicole Sylvester

2023-10-17

Chapter 11 Exercises

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: the pattern 'a' occurs in the strings

```
strings <- c("hello", "goodbye", "apple")
data.frame( string = strings ) %>%
  mutate( result = str_detect(string, 'a') )
```

```
##    string result
## 1   hello  FALSE
## 2 goodbye  FALSE
## 3   apple   TRUE
```

b) This regular expression matches: the pattern 'ab' occurs in the strings

```
strings <- c("ab", "baba", "ba")
data.frame( string = strings ) %>%
  mutate( result = str_detect(string, 'ab') )
```

```
##   string result
## 1     ab   TRUE
## 2   baba   TRUE
## 3     ba  FALSE
```

c) This regular expression matches: any string that has a or b in the string

```
strings <- c("ab", "[ab]", "ba", "apple", "hello")
data.frame( string = strings ) %>%
  mutate( result = str_detect(string, '[ab]') )
```

```
##   string result
## 1     ab   TRUE
## 2   [ab]   TRUE
## 3     ba   TRUE
## 4  apple   TRUE
## 5  hello  FALSE
```

   d) This regular expression matches: strings that have a or b at the beginning of the string

```
strings <- c("b", "ab", "a", "helloab" )
data.frame( string = strings ) %>%
  mutate( result = str_detect(string, '^[ab]') )
```

```
##    string result
## 1       b   TRUE
## 2      ab   TRUE
## 3       a   TRUE
## 4 helloab  FALSE
```

e) This regular expression matches: strings that start with a digit, followed by a white space, followed by an 'a' or 'A' then any other digits.

```
strings <- c("1 a", "1a", "1 d", "2 Ab")
data.frame( string = strings ) %>%
  mutate( result = str_detect(string, '\\d+\\s[aA]') )
```

```
##   string result
## 1    1 a   TRUE
## 2     1a  FALSE
## 3    1 d  FALSE
## 4   2 Ab   TRUE
```

f) This regular expression matches: any string that starts with a digit, followed by zero or more spaces, followed by "a" or "A"

```
strings <- c("5      ab", "5AB", "AB")
data.frame( string = strings ) %>%
mutate( result = str_detect(string, '\\d+\\s*[aA]') )
```

```
##      string result
## 1 5      ab   TRUE
## 2       5AB   TRUE
## 3        AB  FALSE
```

g) This regular expression matches: strings that contain zero or more digits and/or numbers.

```
strings <- c("", " ", "3", "d ")
data.frame( string = strings ) %>%
  mutate( result = str_detect(string, '.*') )
```

```
##   string result
## 1          TRUE
## 2          TRUE
## 3      3   TRUE
## 4     d    TRUE
```

h) This regular expression matches: Any String that starts with 2 repetitions of a alphanumeric charcter.

```
strings <- c("aabar", "22bar", "22barabc", "23bar")
data.frame( string = strings ) %>%
    mutate( result = str_detect(string, '^\\w{2}bar') )
```

```
##     string result
## 1    aabar   TRUE
## 2    22bar   TRUE
## 3 22barabc   TRUE
## 4    23bar   TRUE
```

i) This regular expression matches: any string that starts with the exact string foo.bar OR a string that starts with two alphanumeric

characters followed by bar

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

```
##    string result
## 1   fobar   TRUE
## 2   aabar   TRUE
## 3 foo.bar   TRUE
## 4  foobar  FALSE
```

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:

   ```
   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
   ```

```
df <- data.frame(
  file_info = file.names) %>%
  cbind(str_split_fixed(.$file_info, pattern='[\\.\\_]', n=6)) %>%
  rename(Site= `1`, Plot = `2`, Camera= `3`, Date = `4`, Time = `5` ) %>%
  mutate(
    Year = str_sub(Date, 1, 4),
    Month = str_sub(Date, 5, 6),
    Day = str_sub(Date, 7, 8)
  ) %>%
  select(-Date) %>% #remove date
   mutate(
   Hour = str_sub(Time, 1, 2),
   Minute = str_sub(Time, 3, 4),
   Second = str_sub(Time, 5, 6)
  ) %>%
  select(-Time) %>% #remove time
  select(-`6`) %>% #remove jpg
  select(-file_info) #remove file name

# Print the data frame
df
```

```
##   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
```

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.'
```

```
text <- c(Gettysburg)

df<-data.frame(string = text) %>%
  mutate(result = str_split(string, "[\\,\\.\\s\\-{2}]+"))

wordMean <- str_length(df$result[[1]])
numStrings <- length(df$result[[1]])
wordAvg <- sum(wordMean)/numStrings

wordAvg
```

```
## [1] 4.208791
```

Chapter 12 Exercises

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('Sep 13, 2010.')
```

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

This does not work for Sept, only Sep

c) Sep 13, 2010.

```
mdy('Sep 13, 2010.')
```

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

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

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

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

This does not work with only S. September must be abbreviated with 'Sep'

e) 07-Dec-1941.

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

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

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

```
dmy('1-5-1998.')
```

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

This might be wrong because we don't know if 1 or 5 is the month or day.

g) 21-5-1998. Comment on why you know you are correct.

```
dmy('21-5-1998.')
```

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

This is correct because 21 cannot be a month number, so it must be a day. That means 5 must be the month.

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='US/Pacific')
```

```
## [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"
```

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.

```
dob <- mdy("May 15, 2002")

birthday64 <- dob + years(64)
birthday64
```

```
## [1] "2066-05-15"
```

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

```
dob <- mdy("May 15, 2002")
today <- Sys.Date()

difference <-interval(dob, today)
```

```
currentAge <-as.numeric(as.duration(difference), 'years')
currentAge
```

```
## [1] 21.44285
```

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

```
nextYear <- currentAge + 1
birthDate <- as.Date("2002-05-15")

nextAge <- as.numeric(format(birthDate, "%Y")) + nextYear

nextBirthday <- make_date(year=nextAge, month = 5, day = 15)

nextBirthday
```

```
## [1] "2024-05-15"
```

e) The number of _days_ until your next birthday.

```
days <- as.numeric(nextBirthday - Sys.Date())
days
```

```
## [1] 204
```

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

```
months <- floor(as.numeric(days)/30)
days <- days - months *30
cat("Months:", months, "Days:", days)
```

```
## Months: 6 Days: 24
```

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?

```
AZTime <- ymd_hms("2015-05-08 15:00:00", tz = "America/Phoenix")

aucklandTime <- with_tz(AZTime, tz = "Pacific/Auckland")

aucklandTime
```

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

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`.

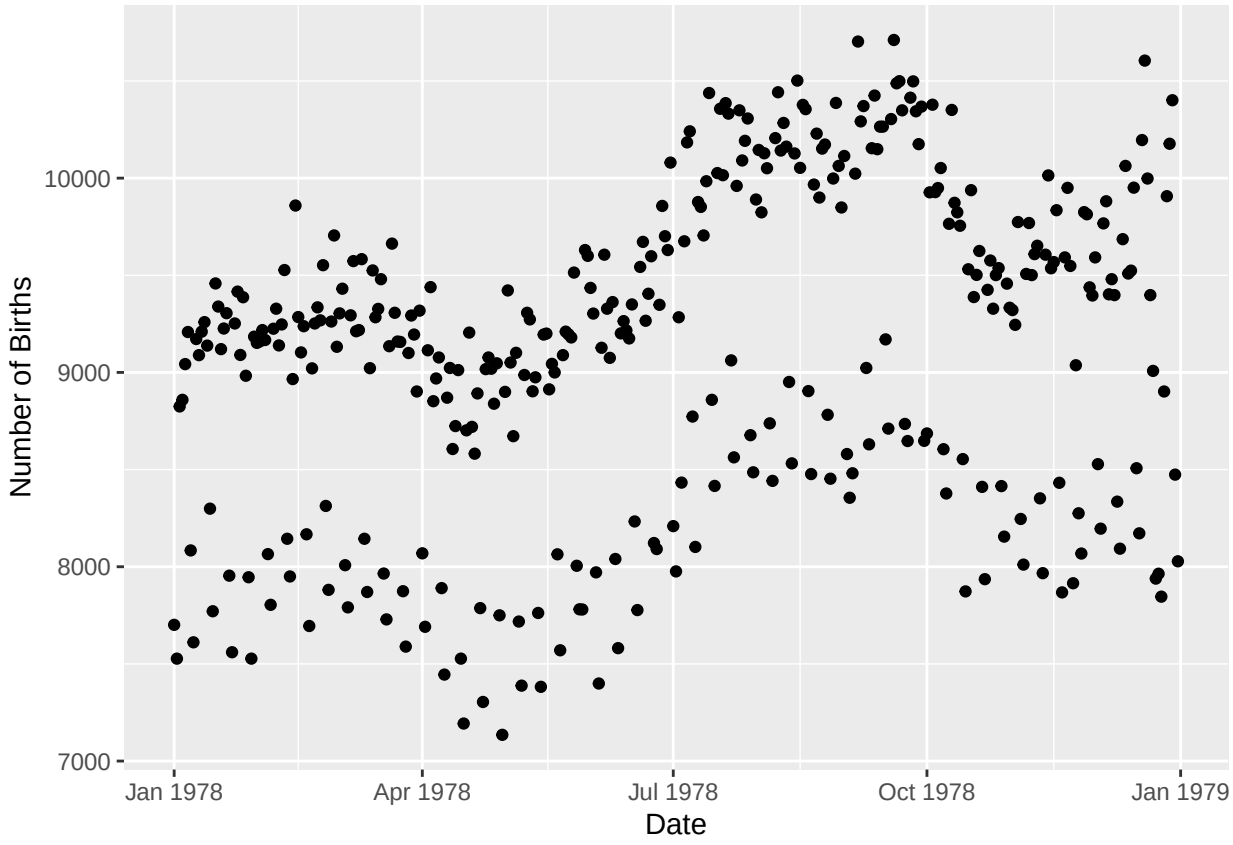
```
data("Births78")
birthsData <- Births78[, c("date", "births")]
head(birthsData)
```

```
##         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(data = birthsData, aes(x = date, y = births)) +
  geom_point() +
  labs(x = "Date", y = "Number of Births")
```



The number of births rise and fall based on dates most likely because of holidays, and seasonal trends.

c. To test your assumption, we need to figure out the what day of the week each observation is. Use `dpl

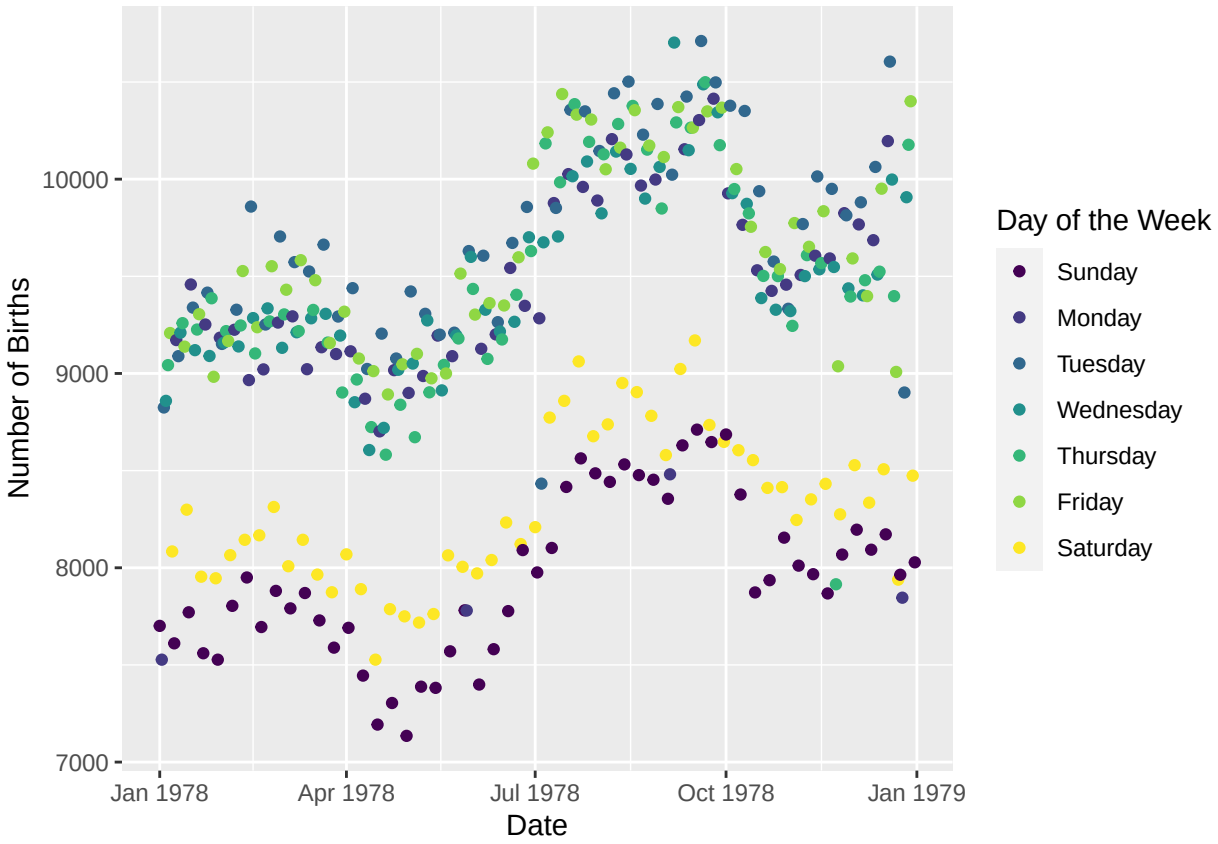
```
birthsData <- birthsData %>%
  mutate(dow = wday(date, label = TRUE, abbr = FALSE))

# View the updated data frame
head(birthsData)
```

```
##         date births       dow
## 1 1978-01-01   7701    Sunday
## 2 1978-01-02   7527    Monday
## 3 1978-01-03   8825   Tuesday
## 4 1978-01-04   8859 Wednesday
## 5 1978-01-05   9043  Thursday
## 6 1978-01-06   9208    Friday
```

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

```
ggplot(data = birthsData, aes(x = date, y = births, color = dow)) +
  geom_point() +
  labs(x = "Date", y = "Number of Births", color = "Day of the Week")
```


Number of Births
10000
9000
8000
7000
Jan 1978
Apr 1978
Jul 1978
Oct 1978
Jan 1979
Date
Day of the Week
Sunday
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday