Assignment Three

Calder Evans

20 October 2023

Chapter Eleven

Exercise One

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.

a) This regular expression matches: Detects whether a string contains an "a"

```
strings <- c("a", "b", "aa", "bb")
data.frame( string = strings ) %>%
  mutate( result = str_detect(string, 'a') )
```

```
##   string result
## 1      a   TRUE
## 2      b  FALSE
## 3     aa   TRUE
## 4     bb  FALSE
```

b) This regular expression matches: Detects whether a string contains "ab", it has to be in that order and that case.

```
strings <- c("ab", "a", "b", "cab", "z", "ba", "AB")
data.frame( string = strings ) %>%
  mutate( result = str_detect(string, 'ab') )
```

```
##   string result
## 1     ab   TRUE
## 2      a  FALSE
## 3      b  FALSE
## 4    cab   TRUE
## 5      z  FALSE
## 6     ba  FALSE
## 7     AB  FALSE
```

c) This regular expression matches: Detects whether a string contains an "a" or "b", it just needs to contain one to return true, case does matter.

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

```
##   string result
## 1     ab   TRUE
## 2      a   TRUE
## 3      b   TRUE
## 4    cab   TRUE
## 5      z  FALSE
## 6     ba   TRUE
## 7     AB  FALSE
```

d) This regular expression matches: Detects whether a string contains an "a" or "b", it can only contain those characters, case matters.

```
strings <- c("ab", "a", "b", "cab", "z", "ba", "AB", "bababa", "cababcabab")
data.frame( string = strings ) %>%
  mutate( result = str_detect(string, '^[ab]') )
```

```
##       string result
## 1         ab   TRUE
## 2          a   TRUE
## 3          b   TRUE
## 4        cab  FALSE
## 5          z  FALSE
## 6         ba   TRUE
## 7         AB  FALSE
## 8     bababa   TRUE
## 9 cababcabab  FALSE
```

e) This regular expression matches: Detects whether a string has multiple digits followed by a blank space, followed by an "a" or an "A". It can only have one space, and the digit part cannot contain non-digits.

```
strings <- c("78 a", "77 A", "77d  A", "77 ab")
data.frame( string = strings ) %>%
  mutate( result = str_detect(string, '\\d+\\s[aA]') )
```

```
##   string result
## 1   78 a   TRUE
## 2   77 A   TRUE
## 3 77d  A  FALSE
## 4  77 ab   TRUE
```

f) This regular expression matches: This is the same as the previous part except now it can contain more than one blank spaces.

```
strings <- c("78 a", "77 A", "77   A", "77 ab")
data.frame( string = strings ) %>%
  mutate( result = str_detect(string, '\\d+\\s*[aA]') )
```

```
##   string result
## 1   78 a   TRUE
## 2   77 A   TRUE
## 3 77   A   TRUE
## 4  77 ab   TRUE
```

g) This regular expression matches: Literally returns true for everything. ".*" looks for zero or more of anything, which is everything.

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

```
##   string result
## 1          TRUE
## 2      a   TRUE
## 3     ab   TRUE
## 4    123   TRUE
```

h) This regular expression matches: Detects whether a string contains exactly 2 characters followed by 'bar'.

```
strings <- c("7", "ab", "abcbar", "zybar")
data.frame( string = strings ) %>%
  mutate( result = str_detect(string, '^\\w{2}bar') )
```

```
##   string result
## 1      7  FALSE
## 2     ab  FALSE
## 3 abcbar  FALSE
## 4  zybar   TRUE
```

i) This regular expression matches: Detects whether a string contains either exactly "foo.bar" or any string that starts with 2 characters followed by "bar"

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

```
##         string result
## 1 fobardfghfdh   TRUE
## 2      foo.bar   TRUE
```

Exercise Two

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')
file.names.r <- str_replace_all(file.names, pattern = "_", replacement = "\\.")
file.names.replaced <- str_split_fixed(file.names.r, pattern = "\\.", n=6)

    camera.strap.study <- data.frame(
      Site = file.names.replaced[,1],
      Plot = file.names.replaced[,2],
      Camera = file.names.replaced[,3],
      Year = str_sub(file.names.replaced[,4], start=1, end=4),
      Month = str_sub(file.names.replaced[,4], start=5, end=6),
      Day = str_sub(file.names.replaced[,4], start=7, end=8),
      Hour = str_sub(file.names.replaced[,5], start=1, end=2),
      Minute = str_sub(file.names.replaced[,5], start=3, end=4),
      Second = str_sub(file.names.replaced[,5], start=5, end=6)
                                    )
    camera.strap.study
```

```
##   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
```

Exercise Three

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.'

    Gettysburg.space.gone <- str_replace_all(Gettysburg, pattern = "\\-\\-|\\.|\\,", replacement = "")
    Gettysburg.split <- str_split(Gettysburg.space.gone, pattern = "\\s+")
    Gettysburg.word.mean.length <- mean(str_length(Gettysburg.split[[1]]))
      Gettysburg.word.mean.length
```

```
## [1] 4.243542
```

Chapter Twelve

Exercise One

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

```
date.object.a <- mdy("September 13, 2010")
  date.object.a
```

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

b) Sept 13, 2010.

```
date.object.b <- mdy("Sept 13, 2010")
```

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

```
  date.object.b
```

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

(b) Wrong, "Sept" must not be an acceptable abbreviation.

c) Sep 13, 2010.

```
date.object.c <- mdy("Sep 13, 2010")
  date.object.c
```

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

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

```
date.object.d <- mdy("S 13, 2010")
```

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

```
  date.object.d
```

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

(d) Wrong, "S" must not be an acceptable abbreviation.

e) 07-Dec-1941.

```
date.object.e <- dmy("07-Dec-1941")
  date.object.e
```

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

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

```
date.object.f.america <- mdy("1-5-1998.")
date.object.f.restofworld <- dmy("1-5-1998.")
  date.object.f.america
```

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

```
  date.object.f.restofworld
```

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

(f) Ambiguous since in different parts of the world the order of month, day year is different.

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

```
date.object.g <- dmy("21-5-1998")
  date.object.g
```

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

```
(g) This one has to be correct because there isn't a 21st month.
```

h) 2020-May-5 10:30 am

```
date.object.h <- ymd_hm("2020-May-5 10:30")
  date.object.h
```

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

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

```
date.object.i <- ymd_hm("2020-May-5 10:30", tz = "US/Pacific")
  date.object.i
```

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

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

```
date.object.j <- ymd_hm("2020-May-5 10:30", tz = "America/Puerto_Rico")
  date.object.j
```

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

Exercise Two

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.

```
birthday <- dmy("11 August 2003")
sixtyforth.birthday <- birthday + years(64)
  sixtyforth.birthday
```

```
## [1] "2067-08-11"
```

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

```
current_age <- (as.period(interval(birthday, today())))
current_age_years <- year(current_age)
  current_age_years
```

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

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

```
next_birthday <- birthday + years((current_age_years + 1))
  next_birthday
```

```
## [1] "2024-08-11"
```

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

```
days_till_next_birthday <- as.period(interval(today(), next_birthday))
  days_till_next_birthday
```

```
## [1] "9m 16d 0H 0M 0S"
```

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

```
    days_till_next_birthday_months <- month(as.period(days_till_next_birthday, unit = "months"))
      days_till_next_birthday_months
```

```
## [1] 9
```

```
    days_till_next_birthday_days <- day(as.period(days_till_next_birthday, unit = "days"))
      days_till_next_birthday_days
```

```
## [1] 289
```

Exercise Three

It is some what surprising that there exists a `dmonths()` function. –> a) Using today's date, add 1 month using `dmonths(1)` and `months()`. Are there any differences?

```
    today() + dmonths(1)
```

```
## [1] "2023-11-25 10:30:00 UTC"
```

```
    today() + months(1)
```

```
## [1] "2023-11-26"
```

```
(a) The "dmonths" function includes the time and the time zone, but it also
    has the date as the 25th of November instead of the 26th of November
    like the "months" function.
```

b) Consider `dmonths(1)`. What does this represent and speculate on how the authors chose to define this, because it isn't just 30 days divided by 7 days in a week.

```
(b) The "dmonths" function must be some derivation from the "dyears"
    function, but it makes a lot less sense than the other functions
    because the amount of days in a month is different every month. It is
    not just adding thirty to the date, I checked. I cannot decipher exactly
    what it does but I'm assuming the authors included it for calculating
    leapyears and leapseconds when you do not want to add an entire year.
```

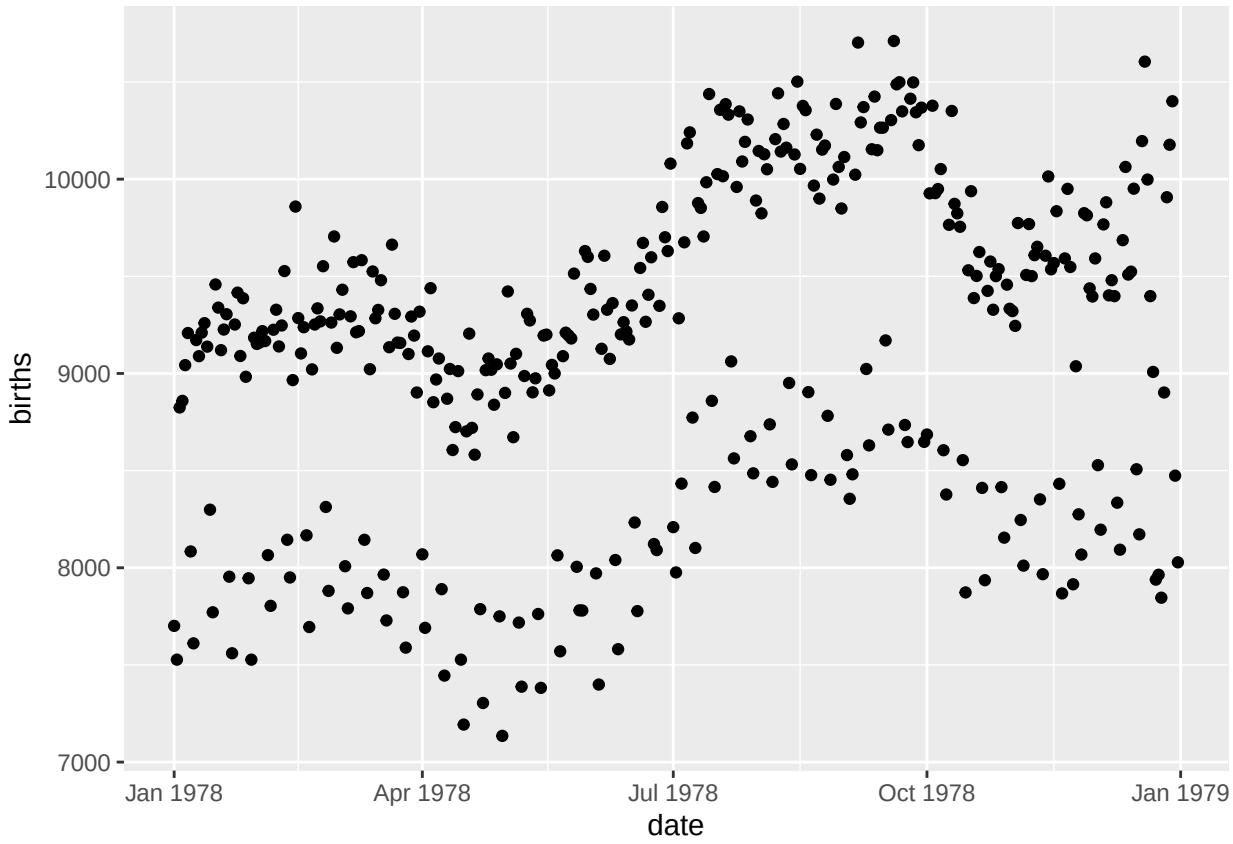
Exercise Five

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`.

```
    Births78.date.births <- 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( data = Births78.date.births, aes(x=date, y=births)) + geom_point()
```



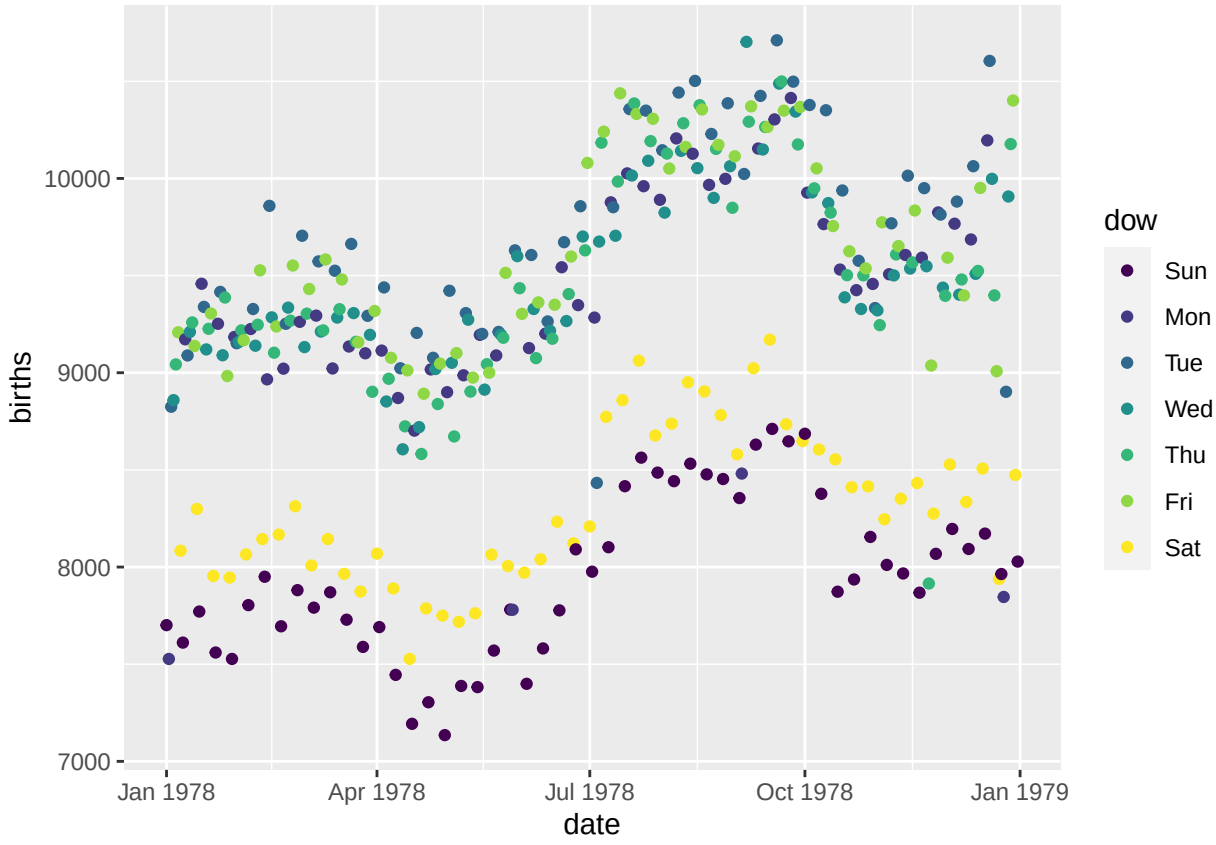
(b) There is two levels of data with a gap in between them. This probably means that certain days of the week are more common to give birth on.

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.

```
Births78.date.births.dow <- Births78.date.births %>%
  mutate(dow = wday(date, label = TRUE))
```

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

```
ggplot( data = Births78.date.births.dow, aes(x=date, y=births)) +
  geom_point( aes(color=dow))
```




```
(d) Doctors don't want to work on the weekend!!!
```