Regular Expressions

```
library(stringi)
library(stringr)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

Regular expression also known as **Regex** is a tool for describing patterns in strings. The package associated with regex in R are stringr and stringi. Regex is also commonly used in other languages with perhaps slight syntatic changes.

stringr: character manipulation. *stringi*: character string processing facilities.

1) stringi package

`str_detect()` helps detect string patterns in a string. Ensure you have installed stringi and stringr packages.

```
city <- "Los Angeles"
str_detect(city,"os")
```

```
## [1] TRUE
```

```
cities <- c("Los Angeles", "New York", "Atlanta", "New Delhi")
str_detect(cities, "New")
```

```
## [1] FALSE  TRUE FALSE  TRUE
```

```
# case sensitive
str_detect(cities, "new")
```

```
## [1] FALSE FALSE FALSE FALSE
```

`str_extract()` helps extract a substring from a string and `str_locate()` helps with the start and end points of a pattern.

```r
str_extract(cities, "New")
```

```
## [1] NA    "New" NA    "New"
```

```r
str_locate(cities, "New")
```

```
##      start end
## [1,]    NA  NA
## [2,]     1   3
## [3,]    NA  NA
## [4,]     1   3
```

`str_sub()` returns a substring of a string given start and end position.

```r
# Atlanta has only 7 characters
# thus it returns "nta"
str_sub(cities, start = 5, end = 8)
```

```
## [1] "Ange" "York" "nta"  "Delh"
```

2) Regular Expressions

2.1) Anchors

Anchors help us assert the position, i.e., the beginning or end of the string.

Anchor	Description	Example
^	Matches any string starting with a substring	`^New`
$	Matches any string ending with a substring	`y$`
Exact Match ^$	Matches the string that starts and ends with substring	`^Hi There$`

2.2) Character Classes

Character classes match any characters given a class

Character Class	Match
`[aeiou]`	Matches vowels
`[0-9]`	Matches digits
`[a-z]`	Matches lower case letters
`[A-Z]`	Matches upper case letters
`[a-zA-Z]`	Matches both lower and upper case letters
`[a-g]`	Matches characters a through g

2.3) Quantifiers

Quantifiers quantify the number of instances of a character, group or character class. Your quantifier should be placed after the character/group/character class that is being quantified.

Quantifier	Description
`P*`	0 or more instances of P
`P+`	1 or more instances of P
`P?`	0 or 1 instance of P
`P{m}`	Exactly m instances of P
`P{m,}`	At least m instances of P
`P{m,n}`	Between m and n instances of P

2.4) Logical Operators

Operator	Usage	Example	Explanation
^	Not	`[^A-Za-z0-9]`	Identifies characters that are not alphanumeric
|	Or	(Apple|Orange)	Identifies cases that have Apple or Oranges or both

```r
str_extract(cities, regex("ew"))
```

```
## [1] NA   "ew" NA   "ew"
```

```r
str_extract(cities, regex("^New [a-zA-Z]*"))
```

```
## [1] NA          "New York"  NA          "New Delhi"
```

```r
banks = c("Bank of America", "Bank of the West", "Citibank",
          "TD Bank", "Bank of England", "People\'s United Bank")

#banks starting with "Bank"
str_detect(banks, regex("^Bank"))
```

```
## [1]  TRUE  TRUE FALSE FALSE  TRUE FALSE
```

```r
#banks ending with "Bank" (note: citibank is False as b is lowercase)
str_detect(banks, regex("Bank$"))
```

```
## [1] FALSE FALSE FALSE  TRUE FALSE  TRUE
```

```r
#bank called "TD Bank" (exact match)
str_detect(banks, regex("^TD Bank$"))
```

```
## [1] FALSE FALSE FALSE  TRUE FALSE FALSE
```

```
# same as above
str_detect(banks, "TD Bank")
```

```
## [1] FALSE FALSE FALSE  TRUE FALSE FALSE
```

```
#Note the whitespace after Z in the regular expression
str_extract(banks, regex("^Bank[a-zA-Z ]*"))
```

```
## [1] "Bank of America"  "Bank of the West" NA                 NA
## [5] "Bank of England"  NA
```

```
#Without the whitespace after Z, it will not give us the desired output
str_extract(banks, regex("^Bank[a-zA-Z]*"))
```

```
## [1] "Bank" "Bank" NA     NA     "Bank" NA
```

```
str_extract(banks,regex("^[a-zA-Z ]{1,9}$"))
```

```
## [1] NA         NA         "Citibank" "TD Bank"  NA         NA
```

```
print(banks)
```

```
## [1] "Bank of America"      "Bank of the West"     "Citibank"
## [4] "TD Bank"              "Bank of England"      "People’s United Bank"
```

```
#Detect all bank names containing Bank or bank
str_detect(banks, regex("Bank|bank"))
```

```
## [1] TRUE TRUE TRUE TRUE TRUE TRUE
```

```
#Detect all bank names containg special characters excluding whitespace
str_detect(banks, regex("[^A-z0-9 ]"))
```

```
## [1] FALSE FALSE FALSE FALSE FALSE  TRUE
```

```
#List the names of first 10 nba players
# with college name starting with "University"
nba = read.csv("nba2018-players.csv")

nba %>%
  filter(str_detect(college, regex("^University"))) %>% head(10)
```

```
##              player team position height weight age experience
## 1  Alfonzo McKinnie  TOR       SF     80    215  25          0
## 2      Delon Wright  TOR       PG     77    183  25          2
## 3     DeMar DeRozan  TOR       SG     79    221  28          8
## 4      Jakob Poeltl  TOR        C     84    248  22          1
## 5       Nigel Hayes  TOR       SF     80    254  23          0
## 6     Norman Powell  TOR       SG     76     215  24          2
## 7        Al Horford  BOS        C     82     245  31         10
## 8       Jabari Bird  BOS       SG     78     197  23          0
## 9      Jaylen Brown  BOS       SG     79     225  21          1
## 10     Kadeem Allen  BOS       PG     75     192  25          0
##                                  college   salary games minutes points3 points2
## 1      University of Wisconsin-Green Bay   815615    14      53       3       5
## 2                     University of Utah  1645200    69    1433      56     145
## 3      University of Southern California 27739975    80    2711      89     556
## 4                     University of Utah  2825640    82    1524       1     252
## 5                University of Wisconsin    92160     2       6       2       0
## 6  University of California, Los Angeles  1471382    70    1062      53      97
## 7                  University of Florida 27734405    72    2277      97     271
## 8               University of California        0    13     115       3      12
## 9               University of California  4956480    70    2152     121     252
## 10                 University of Arizona        0    18     107       0       6
##    points1 points rebounds assists steals blocks turnovers fouls
## 1        2     21        7       1      1      1         3     8
## 2       97    555      198     200     72     33        78    81
## 3      461   1840      315     417     85     22       175   151
## 4       60    567      393      57     39    100        85   212
## 5        0      6        0       0      0      0         1     0
## 6       32    385      119      89     37     16        66   111
## 7       94    927      530     339     43     78       132   138
## 8        6     39       19       8      3      1         8     7
## 9      150   1017      346     114     70     26       124   181
## 10       7     19       11      12      3      2         9    15
```

```
# List the names of colleges of nba players containing
# special characters (don’t consider whitespace and ,
# as special character here)
nba %>%
  filter(str_detect(college, regex("[^A-Za-z, ]"))) %>%
  select(college)
```

```
##                                               college
## 1                   University of Wisconsin-Green Bay
## 2  Indiana University-Purdue University Indianapolis
## 3                                Texas A&M University
## 4                  Saint Mary’s College of California
## 5                           Saint Joseph’s University
## 6                           Saint Joseph’s University
## 7                           Saint Joseph’s University
## 8                                Alabama - Huntsville
## 9                               St. John’s University
## 10        University of Illinois at Urbana-Champaign
## 11        University of Illinois at Urbana-Champaign
## 12                 Saint Mary’s College of California
## 13                               Texas A&M University
## 14                               Texas A&M University
## 15                              St. John’s University
## 16                               Texas A&M University
## 17                               Texas A&M University
```

3) Using Apply

Apply functions are useful when you want to apply a certain operation to all the rows of a list or dataframe. In the example below, we will only consider the case of a data frame. Note that this idea would be useful for your next lab. Here we apply `str_locate()` to locate area code from phone numbers. Using the output of `str_locate()`, we could then extract the area code.

```
phone_num = c(
  "401-501-1111",
  "(401)501-1111",
  "401 501 1111",
  "401-5011111",
  "+408-501-1111")

code_pos = lapply(
  phone_num,
  function(x) str_locate(x, pattern = regex(’[0-9]{3}[- ]*’)))

# Use substring() to extract the area code by sharing the start
# and end position with the function as arguments.
code_pos = do.call(rbind, code_pos)
substring(phone_num, code_pos[,1], code_pos[,2]-1)
```

```
## [1] "401" "401" "401" "401" "408"
```

Practice Problems

1. Print only those cities that start with `"New"` from `city`. Hint: Using `str_detect()`, we could create a mask (a vector of `TRUE` and `FALSE`) that could then be subsetted to get the city names starting with `"New"`.

```
cities[str_detect(cities, "New")]
```

```
## [1] "New York"  "New Delhi"
```

2. Using `str_match()` explained in cheatsheet, check if `city` has any matches for `Los Angeles`.

```
str_match(cities, "Los Angeles")
```

```
##      [,1]
## [1,] "Los Angeles"
## [2,] NA
## [3,] NA
## [4,] NA
```

3. Using `nba` dataframe, print names of Players containing `Marcus`

```
# your code
nba %>% filter(str_detect(player, regex("Marcus"))) %>% select(player)
```

```
##                player
## 1       Marcus Morris
## 2        Marcus Smart
## 3        Marcus Paige
## 4    DeMarcus Cousins
## 5   LaMarcus Aldridge
## 6 Marcus Georges-Hunt
```

4. Using `nba` dataframe, print names of Universities that contain `California` or `Los Angeles` in them.

```
# your code
nba %>% filter(str_detect(college, regex(’California|Los Angeles’))) %>% select(college)
```

```
##                                                                 college
## 1                                    University of Southern California
## 2                                University of California, Los Angeles
## 3                                             University of California
## 4                                             University of California
## 5  University of North Carolina, University of California, Los Angeles
## 6                                University of California, Los Angeles
## 7                                University of California, Los Angeles
## 8                                University of California, Los Angeles
## 9                                University of California, Los Angeles
## 10                                 California State University, Fresno
## 11                                  Saint Mary’s College of California
## 12                               University of California, Los Angeles
## 13                             California State University, Long Beach
## 14                                            University of California
## 15            California Polytechnic State University, San Luis Obispo
## 16                               University of California, Los Angeles
## 17                               University of California, Los Angeles
## 18                                   University of Southern California
## 19                                   University of Southern California
## 20                              California State University, Fullerton
## 21                               University of California, Los Angeles
## 22                                            University of California
## 23                               University of California, Los Angeles
## 24                               University of California, Los Angeles
## 25                                   University of Southern California
## 26                                 California State University, Fresno
## 27                               University of California, Los Angeles
## 28                               University of California, Los Angeles
## 29                               University of California, Los Angeles
## 30                                  Saint Mary’s College of California
## 31                                   University of Southern California
## 32                                            University of California
## 33                               University of California, Los Angeles
## 34 University of North Carolina, University of California, Los Angeles
## 35                                            University of California
## 36                             University of California, Santa Barbara
```

5. Consider the variable below `myVar`. Using str_detect and regular expression, detect those strings that contain at least one `z` and a maximum of three `z`. Your output should be `TRUE TRUE TRUE FALSE`

```r
myVar = c("bizarre", "bizzarre", "bizzzarrre", "bizzzzare")
```

```r
# your code
str_detect(myVar, regex("^[a-y]+z{1,3}[a-y]+"))
```

```
## [1]  TRUE  TRUE  TRUE FALSE
```

6. Consider the output of ls statement from command line. We have stored it in `file_names`, a vector containing file names. Detect file names that start with `STAT154` and have extension `.csv`
   Expected Output: `TRUE FALSE FALSE TRUE FALSE TRUE`

```r
file_names = c(
  "STAT154nba.csv",
  "test0102_STAT154.csv",
  "STAT154myfile.csv.tmp",
  "STAT154_lab.csv",
  "STAT154_HW.pdf",
  "STAT154_test0102.csv")
```

```r
# your code
str_detect(file_names, regex("^(STAT154)[a-zA-Z0-9_]+(.csv)$"))
```

```
## [1]  TRUE FALSE FALSE  TRUE FALSE  TRUE
```

7. For the variable `newVar`, identify strings that start with a number. Your output should be `TRUE FALSE TRUE TRUE FALSE`.

```r
newVar = c(
  "1 Student(s)",
  "None but 1 Student(s)",
  "5 Students",
  "120! Students",
  "Two Students")
```

```r
# your code
str_detect(newVar, regex("^[0-9]"))
```

```
## [1]  TRUE FALSE  TRUE  TRUE FALSE
```

8. Detect phone numbers from a given list. Note that these won't have international codes. Your ouput for given test case should be `TRUE TRUE TRUE TRUE FALSE FALSE` Hint: The last string in the vector has more than 10 digits.

```r
phone_num = c(
  "401-501-1111",
  "(401)501-1111",
  "401 501 1111",
  "4015011111",
  "+408-501-1111",
  "40850211111")
```

```
# your code
str_detect(phone_num, regex("([0-5()-[ ]]{6})([1]{4})"))
```

```
## [1]  TRUE  TRUE  TRUE  TRUE FALSE FALSE
```

9. In the solution you gave above, can you think of cases where your expression would fail to detect an incorrect phone number? If yes, how could you improve it?

```
# No code, but some text
#
# It would fail to detect an incorrect phone number
# if the character length is longer or shorter than your average 10 character numbers.
```

10. In column of a dataframe, we have stored strings that tell us if someone likes orange juice, apple juice etc. Extract the names of students who like orange juice or oranges. Expected Output: Annie Harry Hint: Use `str_detect` before using `str_extract()`

```
myVar = c(
  "Annie likes Orange juice", "Sonny likes Apple juice",
  "Katy dislikes Orange juice", "Harry likes Oranges",
  "Charlie likes Apple juice", "Margo like Orange Pie")

df = data.frame(myVar)
```

```
# your code
filtered <- df %>% filter(str_detect(myVar, regex(" likes Orange juice|( likes Oranges)$")))
str_extract(filtered$myVar, regex("^[A-Za-z]+"))
```

```
## [1] "Annie" "Harry"
```

11. Using the idea above, from the `nba` dataframe, pull the last names of all NBA players. Find the frequencies of all the last names. *Hint:* Use `str_locate()` and `substring()`. Eg: str_locate(nba$player, . . .). Alternatively, you may choose to try this with lapply() but this is slightly tricky.

```
# your code
head(str_sub(nba$player, str_locate(nba$player, pattern = regex(’[^ ]+$’))), 10)
```

```
##  [1] "McKinnie"    "Miles"       "Wright"      "DeRozan"     "VanVleet"
##  [6] "Poeltl"      "Valanciunas" "Lowry"       "Brown"       "Nogueira"
```

As the name suggests, it helps look around the string. `Look Arounds` indicate positions just like anchors, `$`, `^`, that we learnt in previous lab.

4) Look Aheads

The expression `A(?=B)` for look-ahead means “look for A, but match only if followed by B”. There may be any pattern instead of A and B.

4.1) Summary of Look-ahead

Type	Syntax	Description
Positive Look Ahead	`(?=pattern)`	Lookahead Asserts that what immediately follows the current position in the string is pattern
Negative Look Ahead	`(?!pattern)`	Negative Lookahead Asserts that what immediately follows the current position in the string is not pattern

```
myVar2 = "7 days of 10mm rainfall in Ohio and 2 nights of 15mm rainfall in NYC"

# extract all digits followed by "mm"
str_extract_all(myVar2, regex(’[0-9]+(?=mm)’))
```

```
## [[1]]
## [1] "10" "15"
```

```
# extract all digits not followed by "mm"
str_extract_all(myVar2, regex(’[0-9]+ (?!mm)’))
```

```
## [[1]]
## [1] "7 " "2 "
```

5) Look Behinds

Look Behind allows to match a pattern only if there’s something before it. This is contrary to lookahead which allows to assert for “what follows”. The expression `(?<=B)A` matches A, but only if there’s B before it.

5.1) Summary of Look-behind

Type	Syntax	Description
Positive Look Behind	`(?<=pattern)`	Lookbehind Asserts that what immediately precedes the current position in the string is pattern
Negative Look Behind	`(?<!pattern)`	Negative Lookbehind Asserts that what immediately precedes the current position in the string is not pattern

```
myVar3 = ’1 apple costs $1.50 in USA, 2 apples cost £3 elsewhere and 5€ in France’

# extract only prices
str_extract_all(myVar3, regex(’((?<=[$£]{1})[0-9.]+)|([0-9.]+(?=[€]{1}))’))
```

```
## [[1]]
## [1] "1.50" "3"    "5"
```

Prictice Problems

1. In these examples, lets try to extract the second university listed for every player in `sample_college` variable, a subset of the `nba` dataset we used earlier.

```
nba = read.csv("nba2018-players.csv")

sample_college = nba[c(1,4,14,18,37,51,56,245,254:256,274:276,291),’college’]
sample_college
```

```
##  [1] University of Wisconsin-Green Bay
##  [2] University of Southern California
##  [3] University of California, Los Angeles
##  [4] Northern Illinois University, Iowa State University
##  [5] University of Colorado, Northern Illinois University
##  [6] University of North Carolina, University of California, Los Angeles
##  [7] Duquesne University, University of Arizona
##  [8] University of Pittsburgh, University of Nevada, Las Vegas
##  [9] University of Nebraska, Syracuse University
## [10] Southeast Missouri State University
## [11] Drexel University, University of Louisville
## [12] California State University, Fullerton
## [13] University of North Carolina
## [14] Virginia Commonwealth University
## [15] University of Memphis, University of Kansas
## 150 Levels:  Alabama - Huntsville ... Xavier University
```

In case our input string is `University of Memphis, University of Kansas`, we should get output as `University of Kansas`, i.e., print the second university name listed.

```
str_extract(sample_college, regex("(?<=, ).*University.*"))
```

```
##  [1] NA
##  [2] NA
##  [3] NA
##  [4] "Iowa State University"
##  [5] "Northern Illinois University"
##  [6] "University of California, Los Angeles"
##  [7] "University of Arizona"
##  [8] "University of Nevada, Las Vegas"
##  [9] "Syracuse University"
## [10] NA
## [11] "University of Louisville"
## [12] NA
## [13] NA
## [14] NA
## [15] "University of Kansas"
```

2. Using the expressions developed so far, write a function that returns a dataframe with two columns, last name of player and the name of second university that we extracted in last questions. The input to the function would be a subset of nba players. You only need to populate the function definition.

Hint: On running the above chunk, your output should be (for first five lines):

	Last_name	Second_Univ
1	McKinnie	
2	DeRozan	
3	Powell	
4	Nader	Iowa State University
5	Silas	Northern Illinois University

```r
myfunc = function(df) {
  Last_name <- str_extract(df$player, regex("(?<= ).*"))
  Second_Univ <- str_extract(df$college, regex("(?<=, ).*University.*"))
  new_df <- data.frame(Last_name, Second_Univ)
  ### Write your code here and modify return statement
  return(head(new_df, 10))
}

myfunc(nba[c(1,4,14,18,37,51,56,245,254:256,274:276,291),])
```

```
##    Last_name                          Second_Univ
## 1   McKinnie                                 <NA>
## 2    DeRozan                                 <NA>
## 3     Powell                                 <NA>
## 4      Nader                Iowa State University
## 5      Silas         Northern Illinois University
## 6       Drew University of California, Los Angeles
## 7  McConnell                University of Arizona
## 8      Birch       University of Nevada, Las Vegas
## 9      White                  Syracuse University
## 10 Cleveland                                 <NA>
```

3. In webscrapped content, one could retrieve desired content by looking for tags. In this case we would like to retrieve `firstHeading`.

Example: `<h1 id="firstHeading" class="firstHeading" lang="en">University of California, Berkeley</h1>`

Here **University of Berkeley** is the content we need to retrieve, i.e. our `firstHeading`. First headings are always enclosed between

(a) `<h1 id="firstHeading" class="firstHeading" lang="en">`
(b) `</h1>`

Write code to extract all occurances of First Headings in variable `text`. Use positive look ahead and positive look behind to capture the content between (a) and (b) given above.

Your output should be:

```
[[1]]
[1] "Yoshua Bengio"             "Turing Award"
[3] "University of Manchester" "Chicken soup"
```

```
text = ’<h1 id="firstHeading" class="firstHeading" lang="en">University of California, Berkeley</h1><h1
```

```
str_extract_all(text, regex(’(?<="en">)[a-zA-Z ]*(?=</)’))
```

```
## [[1]]
## [1] "Yoshua Bengio"              "Turing Award"
## [3] "University of Manchester" "Chicken soup"
```

6) Some Metacharacters

Expression	Description
`\\d`	match any digit (same as `[0-9]`)
`\\s`	match any whitespace (space, tab)
`\\t`	match only tab
`\\b`	match a word boundary
`\\A`	match the beginning of input
`\\Z`	match the end of input

Metacharacters have a special meaning during pattern processing. Literals as we learnt in lectures are actual strings that we match. Lets look at some examples.

```
myVar5 = "7 days of 10mm rainfall in Ohio and 2 nights of 15mm rainfall in NYC"

str_extract_all(myVar5, regex(’\\d’))
```

```
## [[1]]
## [1] "7" "1" "0" "2" "1" "5"
```

```
str_extract_all(myVar5, regex(’\\d\\d’))
```

```
## [[1]]
## [1] "10" "15"
```

```
str_extract_all(myVar5, regex(’\\d+’))
```

```
## [[1]]
## [1] "7"  "10" "2"  "15"
```

```
str_extract_all(myVar5, regex(’\\w’))
```

```
## [[1]]
##  [1] "7" "d" "a" "y" "s" "o" "f" "1" "0" "m" "m" "r" "a" "i" "n" "f" "a" "l" "l"
## [20] "i" "n" "O" "h" "i" "o" "a" "n" "d" "2" "n" "i" "g" "h" "t" "s" "o" "f" "1"
## [39] "5" "m" "m" "r" "a" "i" "n" "f" "a" "l" "l" "i" "n" "N" "Y" "C"
```

```
str_extract_all(myVar5, regex(’\\w+’))
```

```
## [[1]]
##  [1] "7"        "days"     "of"       "10mm"     "rainfall" "in"
##  [7] "Ohio"     "and"      "2"        "nights"   "of"       "15mm"
## [13] "rainfall" "in"       "NYC"
```

Practice Problems

1. Write code to detect three digit area code The output for this case should be `TRUE TRUE FALSE FALSE`. Hint: Replace periods with solution `^[...]?\\d{...}[...]?$`

```
test_case = c(’401’, ’(401)’, ’4015’, ’+401’)

str_detect(test_case, regex(’^[(]?\\d{3}[)]?$’))
```

```
## [1]  TRUE  TRUE FALSE FALSE
```

2. To the previous expression make changes to detect first 6 numbers. Your output here should be TRUE TRUE TRUE TRUE FALSE FALSE

```
test_case2 = c(’401 501’, ’401-501’, ’(401)501’, ’401501’, ’+401501’, ’4011501’)

str_detect(test_case2, regex(’^[(]?\\d{3}[) -]?\\d{3}$’))
```

```
## [1]  TRUE  TRUE  TRUE  TRUE FALSE FALSE
```

3. Now consider the actual problem. Modify solution to previous question to find a pattern for detecting phone numbers. Your ouput for given test case should be TRUE TRUE TRUE TRUE FALSE FALSE

```
phone_num = c(
  "401-501-1111",
  "(401)501-1111",
  "401 501 1111",
  "4015011111",
  "+408-501-1111",
  "40850211111")

str_detect(phone_num, regex(’^[(]?\\d{3}[) -]?\\d{3}[- ]?\\d{4}$’))
```

```
## [1]  TRUE  TRUE  TRUE  TRUE FALSE FALSE
```