Show me the data!

Week07: Regular Expression

Big Data & Social Analysis R

Instructors: Chung-pei Pien

ZU1942001/266868001/Z23937001/ZM1941001

International College of INNOVATION
National Chengchi University
國立政治大學創新國際學院

CONTENTS

1 Plot and Loop

2 Regular Expression

3 Ukraine Conflict Twitter

Plot and Loop

05 Plot and Loop

Can loop function can help us to do data mining?

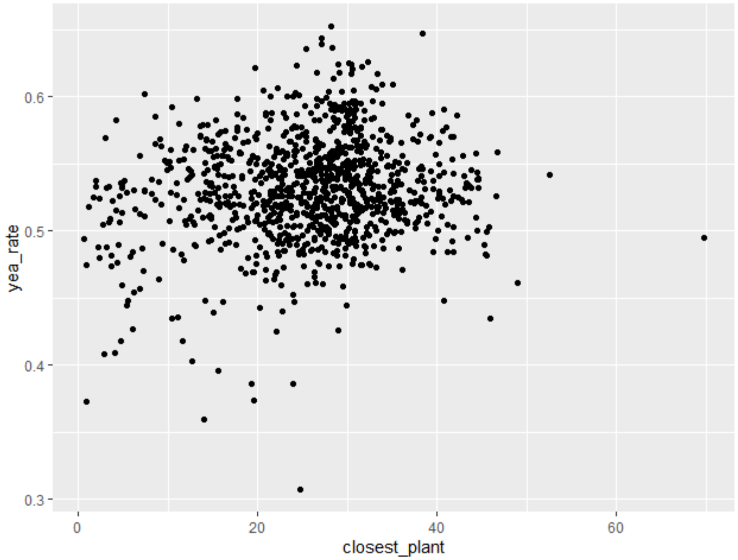
Yes! It is!

The relationship between a li's distance from nuclear power plant and ref_16_yea in 2018.

05 Plot and Loop

Let’s make a plot first.

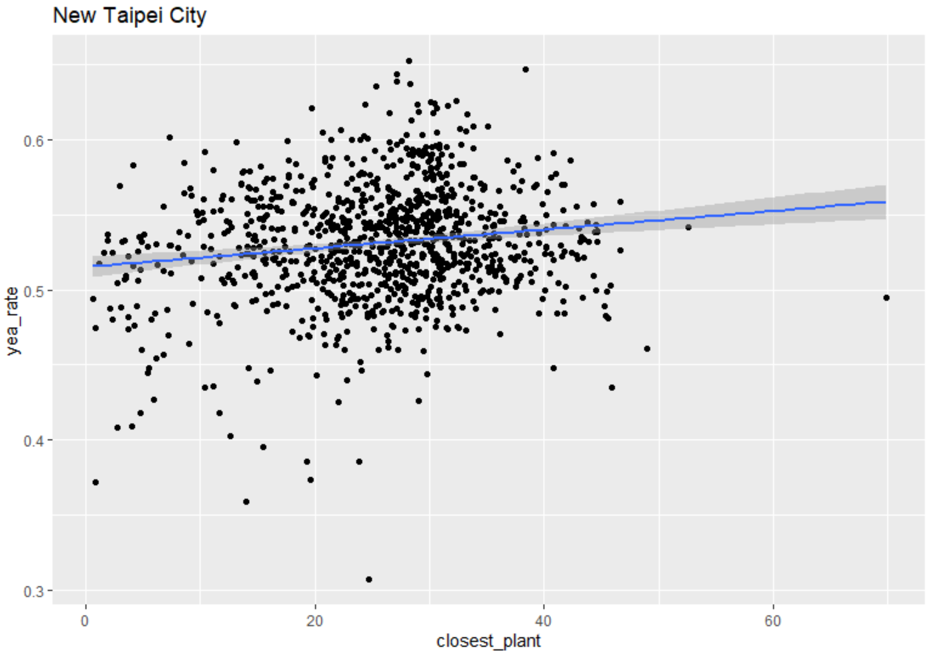
```
ntc_2018 <- ntc_2018 %>%
  mutate(yea_rate = rf_16_yea / rf_16_turnout)

ggplot(ntc_2018, aes(closest_plant, yea_rate)) +
  geom_point()
```



05 Plot and Loop

Add a regression line and a title:

```
ggplot(ntc_2018, aes(closest_plant, yea_rate)) +
  geom_point() +
  geom_smooth(method = lm) +
  ggtitle("New Taipei City")
```



05 Plot and Loop

How to save a ggplot plot?

```
ntc_plot <- ggplot(ntc_2018, aes(closest_plant, yea_rate)) +
  geom_point() +
  geom_smooth(method = lm) +
  ggtitle("New Taipei City")

ggsave(ntc_plot, file = "ntc_plot.png", width = 14, height = 10, units = "cm")
```

05

Plot and Loop

Product all districts' plots (x = closest_plant and y = yea_reat) to check individual district's results.

Could you use loop function to do this task?

05 Plot and Loop

If we use loop, what codes we should adjust in the loop?

```
ntc_plot <- ggplot(ntc_2018, aes(closest_plant, yea_rate)) +
  geom_point() +
  geom_smooth(method = lm) +
  ggtitle("New Taipei City")

ggsave(ntc_plot, file = "ntc_plot.png", width = 14, height =
10, units = "cm")
```

05 Plot and Loop

How to create single district's table and allow loop to read it?

```
dlist <- ------(ntc_2018$district)

ddf <- ntc_2018 %>%
  -----(-----   ---   dlist[1])
```

Fill tem out and paste the codes in Moodle (Practice 2)

05 Plot and Loop

Adjust the codes

```
dlist <- unique(ntc_2018$district)

ddf <- ntc_2018 %>%
  filter(district == dlist[1])

temp_plot~~ntc_plot~~ <- ggplot(ddf~~ntc_2018~~, aes(closest_plant, yea_rate)) +
  geom_point() +
  geom_smooth(method = lm) +
  ggtitle(dlist[1]~~"New Taipei City"~~)

ggsave(temp_plot~~ntc_plot~~, file = ~~"ntc_plot.png"~~paste0(dlist[1], ".png"),
          width = 14, height = 10, units = "cm")
```

05 Plot and Loop

Copy and paste the codes into the loop:

```
for (i in 1:29) {

  ddf <- ntc_2018 %>%
    filter(district == dlist[i~~1~~])

  temp_plot <- ggplot(ddf, aes(yea_rate, closest_plant)) +
    geom_point() +
    geom_smooth(method = lm) +
    ggtitle(dlist[i~~1~~])

  ggsave(temp_plot, file = paste0(dlist[i~~1~~], ".png"),
                width = 14, height = 10, units = "cm")
  }
```

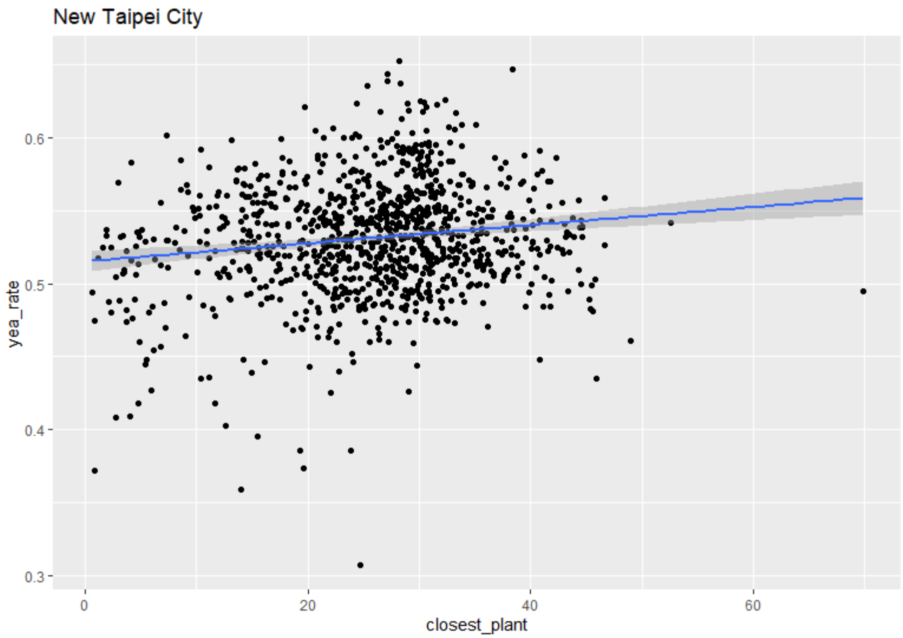
05 Plot and Loop

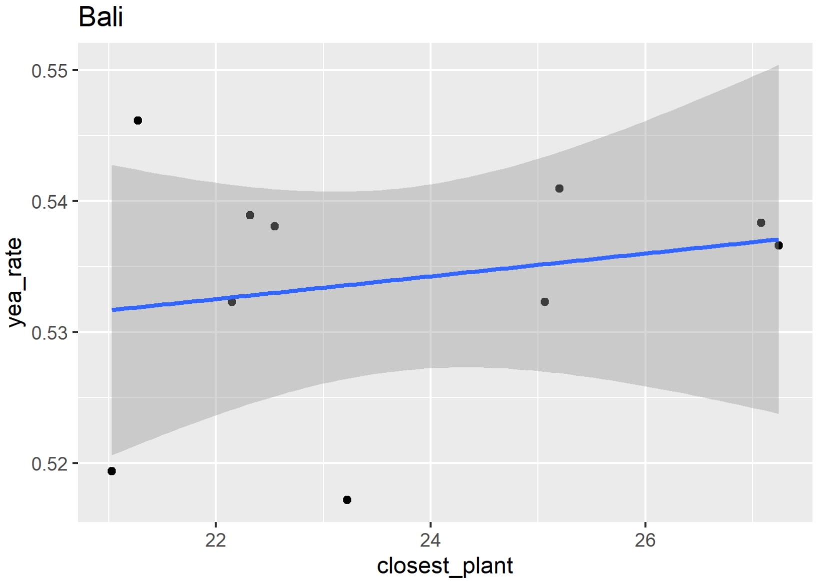
Save the plots into sub-directory:

```
for (i in 1:29) {

 ddf <- ntc_2018 %>%
  filter(district == dlist[i])

 temp_plot <- ggplot(ddf, aes(yea_rate, closest_plant)) +
  geom_point() +
  geom_smooth(method = lm) +
  ggtitle(dlist[i])

 ggsave(temp_plot, file = paste0("ntc_plot/", dlist[i], ".png"),
              width = 14, height = 10, units = "cm")
 }
```

05 Plot and Loop

The plots we produced are great but not perfect for data sciencists.

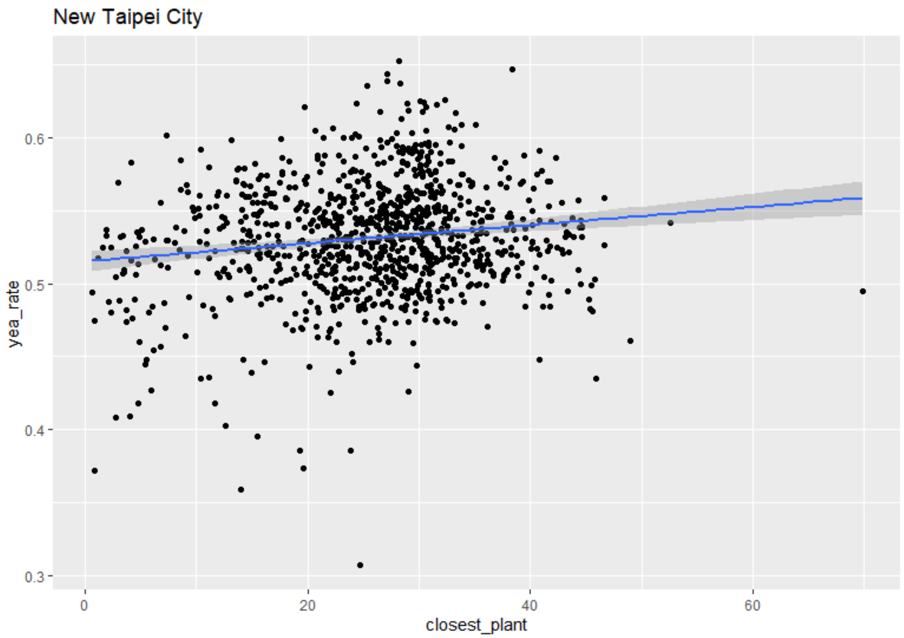


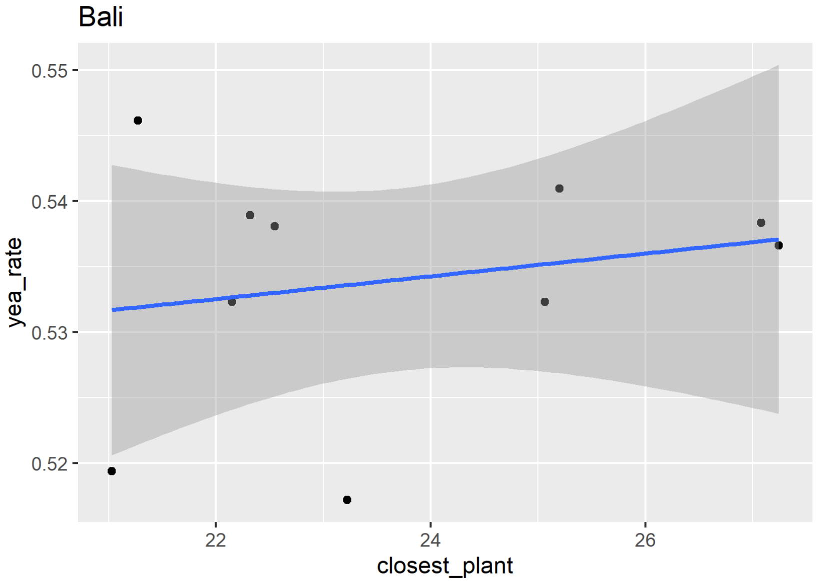


Why?

05 Plot and Loop

The plots we produced are great but not perfect for data sciencists.

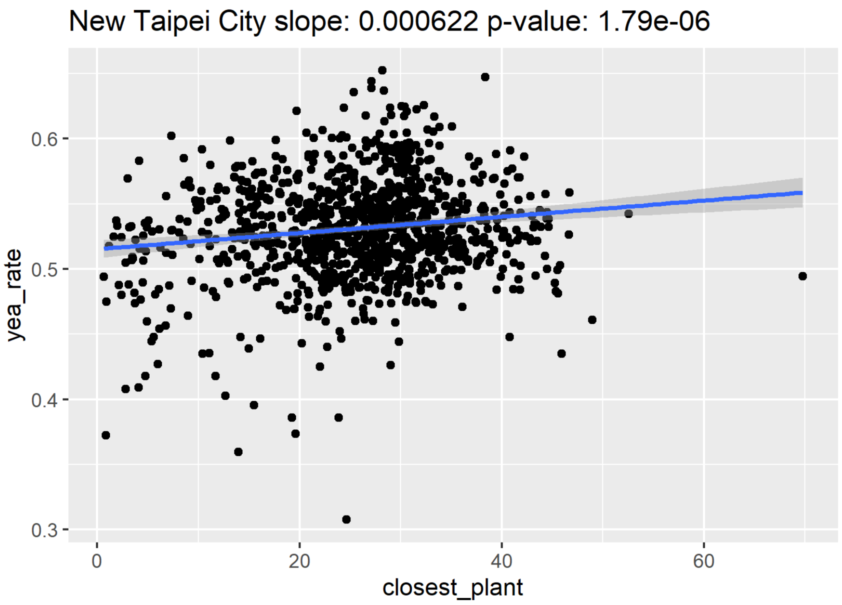


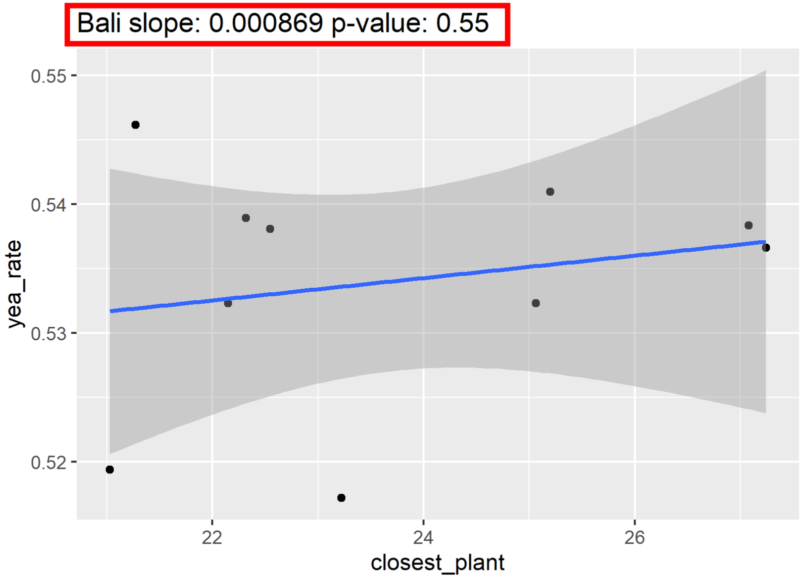


We don't know the districts' slope values and if the values' are significant.

05 Plot and Loop

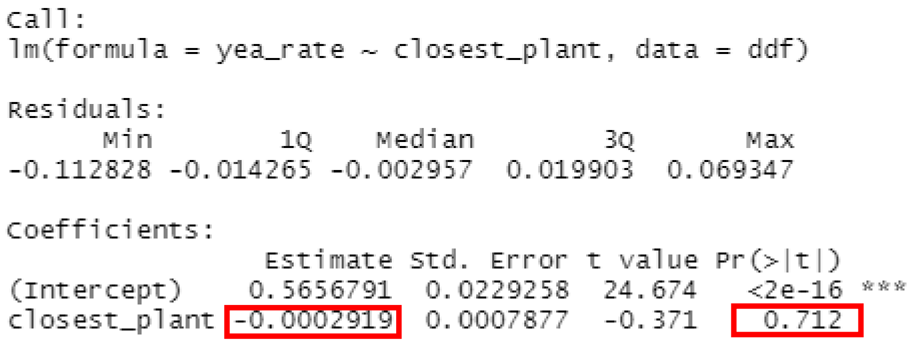
Bali’s slope of closest_plant is larger than the whole city, but it’s insignificant.


New Taipei City slope: 0.000622 p-value: 1.79e-06


Bali slope: 0.000869 p-value: 0.55

We don’t need to pay much attention into Bali.

05 Plot and Loop



```
Call:
lm(formula = yea_rate ~ closest_plant, data = ddf)

Residuals:
      Min        1Q    Median        3Q       Max 
-0.112828 -0.014265 -0.002957  0.019903  0.069347 

Coefficients:
                Estimate Std. Error t value Pr(>|t|)    
(Intercept)    0.5656791  0.0229258  24.674   <2e-16 ***
closest_plant -0.0002919  0.0007877  -0.371    0.712    
```

```
for (i in 1:29) {

  ddf <- ntc_2018 %>%
    filter(district == dlist[i])

  yea_close <- lm(yea_rate ~ closest_plant, data = ddf)
  #summary(yea_close)

  temp_plot <- ggplot(ddf, aes(closest_plant, yea_rate)) +
    geom_point() +
    geom_smooth(method = lm) +
    ggtitle(paste(dlist[i], "slope:", signif(yea_close$coefficients[2], 3), "p-value:", signif(summary(yea_close)$coef[2, 4], 3)))

  ggsave(temp_plot, file = paste0("ntc_plot/", dlist[i], ".png"), width = 14, height = 10, units = "cm")

}
```

05 Plot and Loop

Too many plots? Only produce plots that p-value is small than 0.1

```
for (i in 1:29) {

  ddf <- ntc_2018 %>%
    filter(district == dlist[i])
  yea_close <- lm(yea_rate ~ closest_plant, data = ddf)
  pv  <- signif(summary(yea_close)$coef[2, 4], 3)
if (pv <= 0.1) {

  temp_plot <- ggplot(ddf, aes(closest_plant, yea_rate)) +
    geom_point() +
    geom_smooth(method = lm) +
    ggtitle(paste(dlist[i], "slope:", signif(yea_close$coefficients[2], 3), "p-value:", pv))

  ggsave(temp_plot, file = paste0("ntc_plot_filter/", dlist[i], ".png"), width = 14, height = 10, units = "cm")
}
}
```

Regular Expression

02

Regular Expression

Introduction

What is regular expression?

Regular expressions are an encoded text system for matching sets of strings.

In general, we use the skills of regular expressions to find patterns of huge set of text dataset to deal with them.

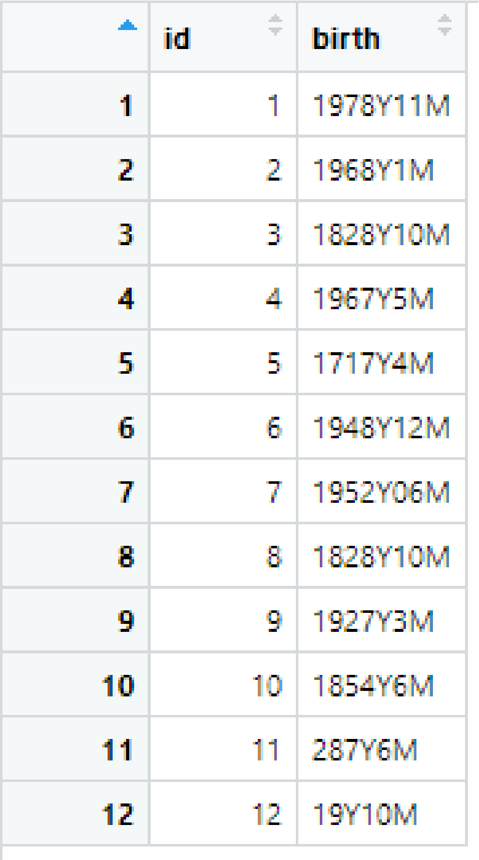
02

Regular Expression

Introduction

Run the codes: re1

How to separate birth into year and month?



	id	birth
1	1	1978Y11M
2	2	1968Y1M
3	3	1828Y10M
4	4	1967Y5M
5	5	1717Y4M
6	6	1948Y12M
7	7	1952Y06M
8	8	1828Y10M
9	9	1927Y3M
10	10	1854Y6M
11	11	287Y6M
12	12	19Y10M

02

Regular Expression

Introduction

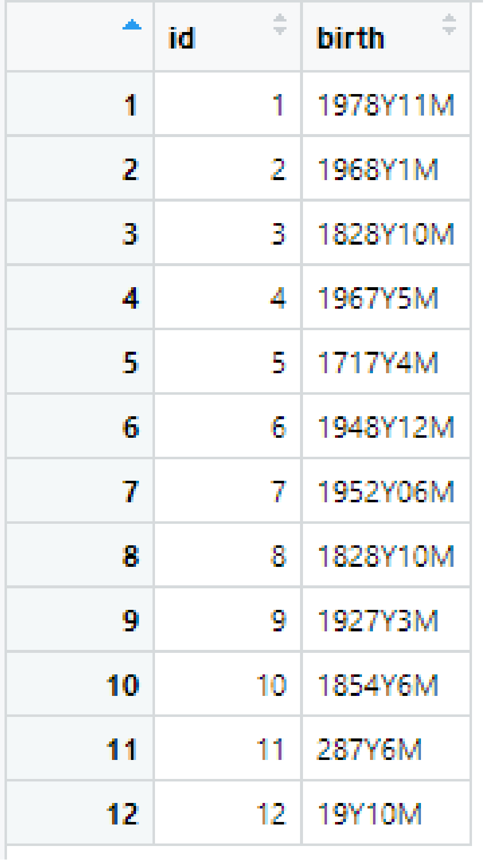
How to separate birth into year and month?

Y is your pattern

regexpr(pattern, x)

regexpr() produces two information:

The starting location
The length of the pattern

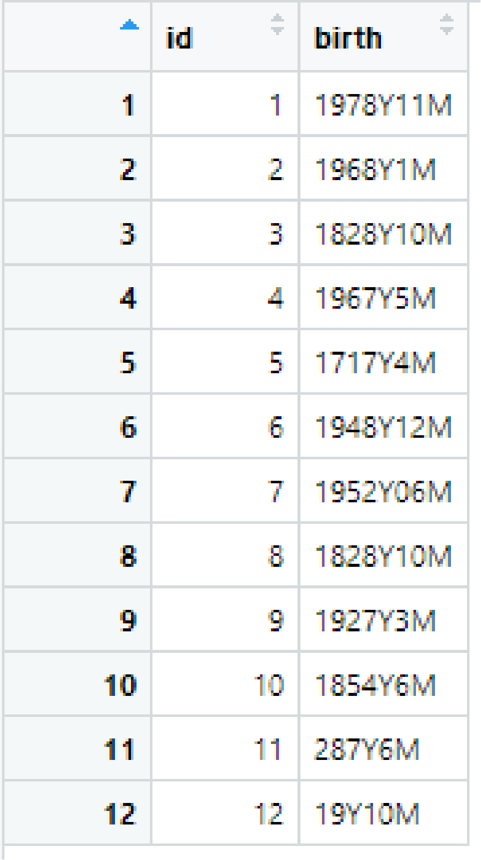


	id	birth
1	1	1978Y11M
2	2	1968Y1M
3	3	1828Y10M
4	4	1967Y5M
5	5	1717Y4M
6	6	1948Y12M
7	7	1952Y06M
8	8	1828Y10M
9	9	1927Y3M
10	10	1854Y6M
11	11	287Y6M
12	12	19Y10M

02 Regular Expression

Introduction

```
yp <- regexpr("Y", re1$birth)

re1$year <- substr(re1$birth, 1, yp - 1)

re1$month <- substr(re1$birth, yp + 1,
nchar(re1$birth) - 1)
```



	id	birth
1	1	1978Y11M
2	2	1968Y1M
3	3	1828Y10M
4	4	1967Y5M
5	5	1717Y4M
6	6	1948Y12M
7	7	1952Y06M
8	8	1828Y10M
9	9	1927Y3M
10	10	1854Y6M
11	11	287Y6M
12	12	19Y10M

02

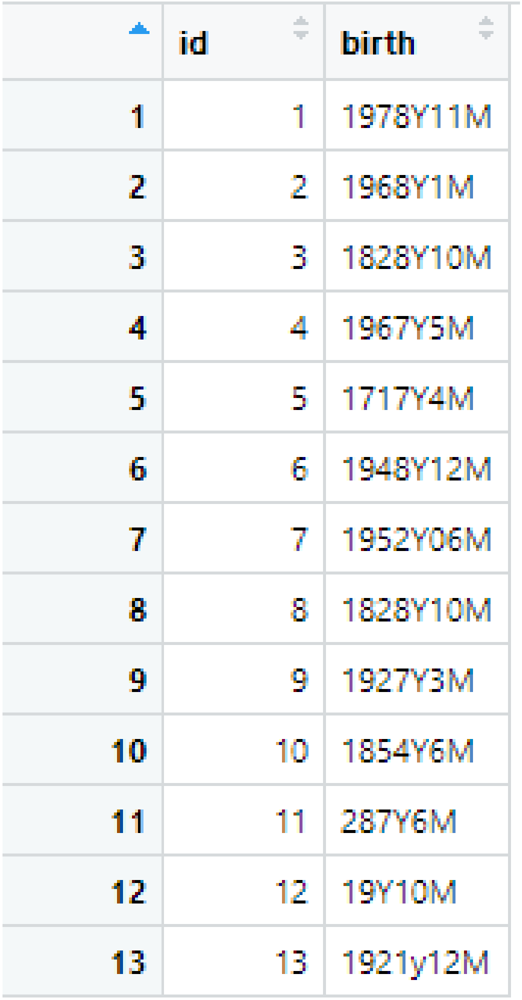
Regular Expression

Introduction

Typos: re2

It always happen!!!

```
yp <- regexpr("Y", re2$birth)
```



	id	birth
1	1	1978Y11M
2	2	1968Y1M
3	3	1828Y10M
4	4	1967Y5M
5	5	1717Y4M
6	6	1948Y12M
7	7	1952Y06M
8	8	1828Y10M
9	9	1927Y3M
10	10	1854Y6M
11	11	287Y6M
12	12	19Y10M
13	13	1921y12M

02

Regular Expression

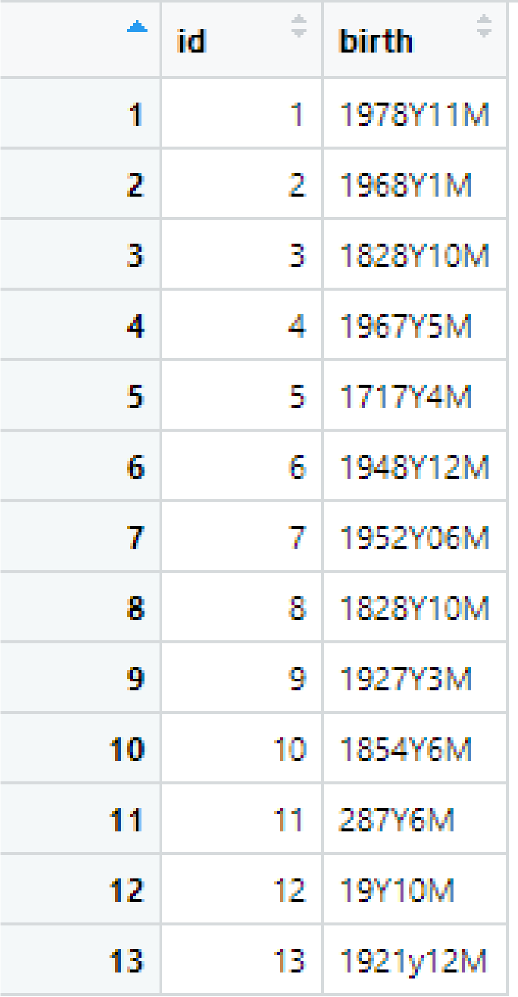
Introduction

Typos: re2

It always happen!!!

yp <- regexpr("Y", re2$birth)

regexpr cannot identify 13th id’s y.



	id	birth
1	1	1978Y11M
2	2	1968Y1M
3	3	1828Y10M
4	4	1967Y5M
5	5	1717Y4M
6	6	1948Y12M
7	7	1952Y06M
8	8	1828Y10M
9	9	1927Y3M
10	10	1854Y6M
11	11	287Y6M
12	12	19Y10M
13	13	1921y12M

02

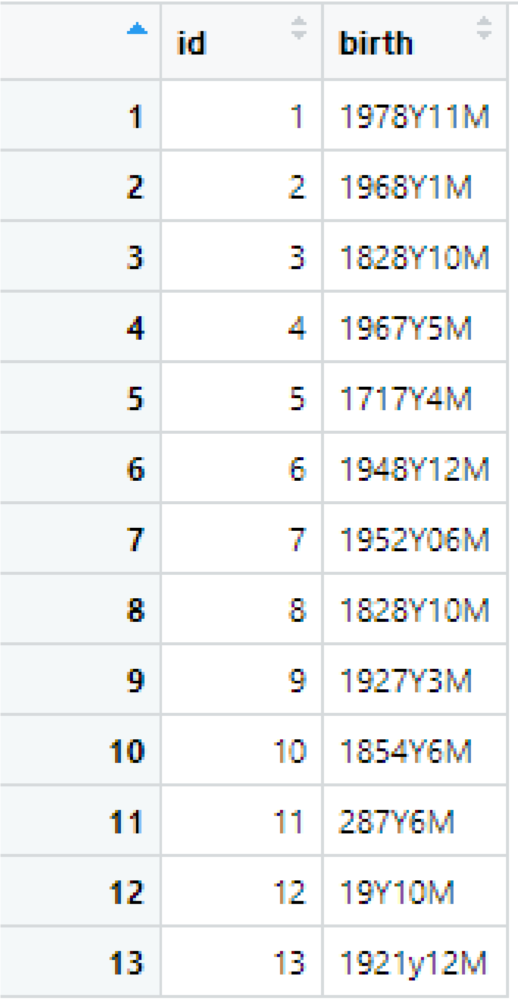
Regular Expression

Introduction

Two opinions to fix this problem:

1. Tell R the pattern is Y or y

```
yp <- regexpr("Y|y", re2$birth)

yp
```



	id	birth
1	1	1978Y11M
2	2	1968Y1M
3	3	1828Y10M
4	4	1967Y5M
5	5	1717Y4M
6	6	1948Y12M
7	7	1952Y06M
8	8	1828Y10M
9	9	1927Y3M
10	10	1854Y6M
11	11	287Y6M
12	12	19Y10M
13	13	1921y12M

02 Regular Expression

Introduction

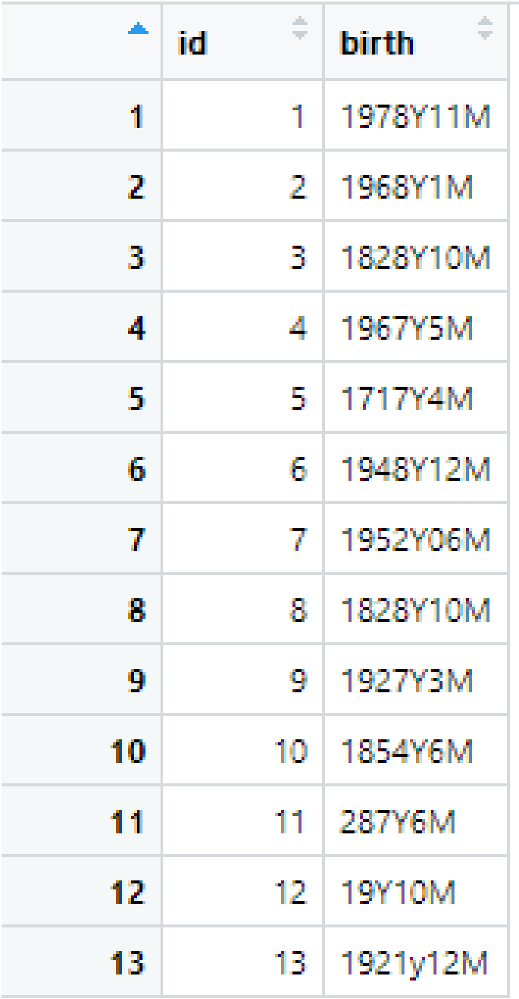
Two opinions to fix this problems:

2. Replace Y to y

gsub(pattern, replace, x)

gsub("y", "Y", re2$birth)

yp <- regexpr("Y", re2$birth)



	id	birth
1	1	1978Y11M
2	2	1968Y1M
3	3	1828Y10M
4	4	1967Y5M
5	5	1717Y4M
6	6	1948Y12M
7	7	1952Y06M
8	8	1828Y10M
9	9	1927Y3M
10	10	1854Y6M
11	11	287Y6M
12	12	19Y10M
13	13	1921y12M

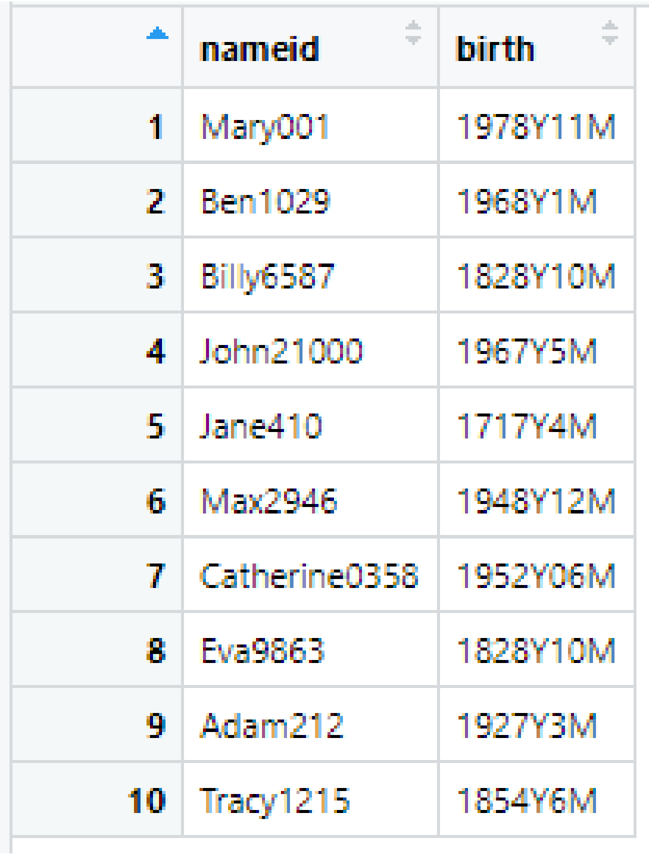
02 Regular Expression

Introduction

Name and ID: re3

Name and id are mixed up!

You don’t have a specific letter (Y) to refer.



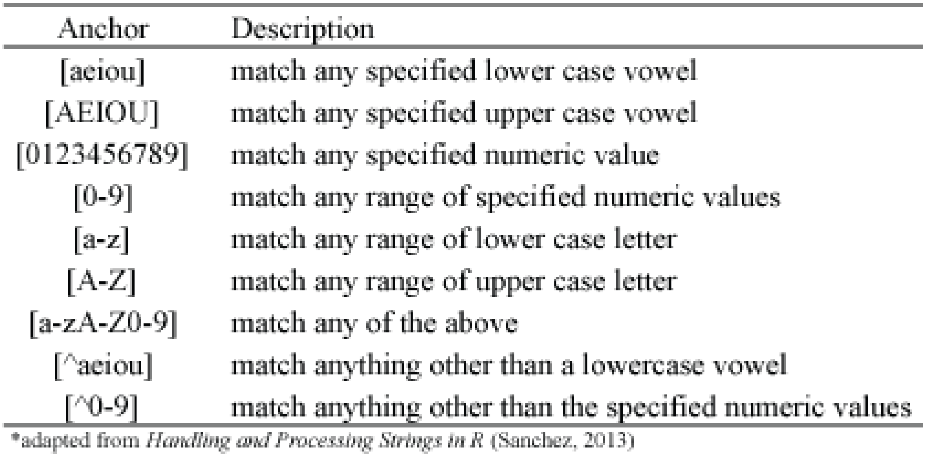
	nameid	birth
1	Mary001	1978Y11M
2	Ben1029	1968Y1M
3	Billy6587	1828Y10M
4	John21000	1967Y5M
5	Jane410	1717Y4M
6	Max2946	1948Y12M
7	Catherine0358	1952Y06M
8	Eva9863	1828Y10M
9	Adam212	1927Y3M
10	Tracy1215	1854Y6M

02

Regular Expression

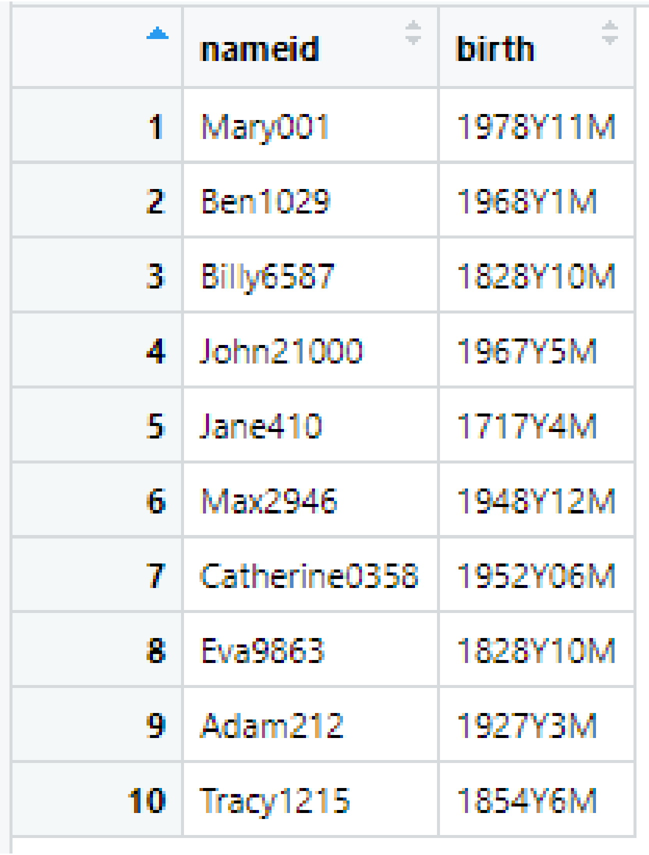
Introduction

To match one of several characters in a specified set we can enclose the characters of concern with square brackets [ ]



Anchor	Description
[aeiou]	match any specified lower case vowel
[AEIOU]	match any specified upper case vowel
[0123456789]	match any specified numeric value
[0-9]	match any range of specified numeric values
[a-z]	match any range of lower case letter
[A-Z]	match any range of upper case letter
[a-zA-Z0-9]	match any of the above
[^aeiou]	match anything other than a lowercase vowel
[^0-9]	match anything other than the specified numeric values

*adapted from *Handling and Processing Strings in R* (Sanchez, 2013)

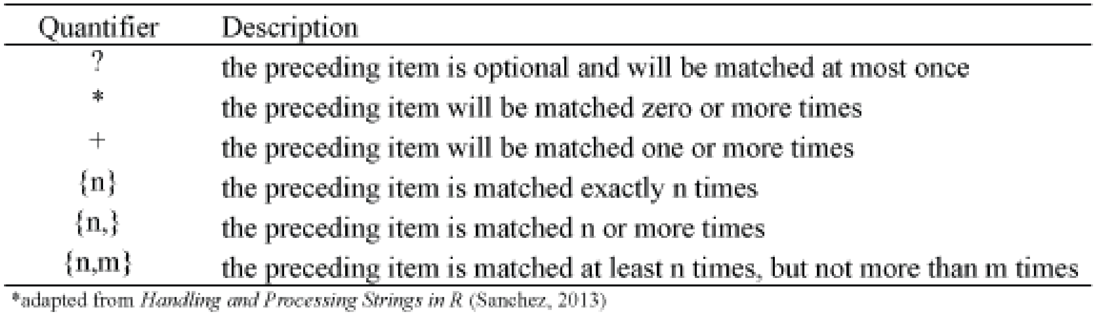


	nameid	birth
1	Mary001	1978Y11M
2	Ben1029	1968Y1M
3	Billy6587	1828Y10M
4	John21000	1967Y5M
5	Jane410	1717Y4M
6	Max2946	1948Y12M
7	Catherine0358	1952Y06M
8	Eva9863	1828Y10M
9	Adam212	1927Y3M
10	Tracy1215	1854Y6M

02 Regular Expression

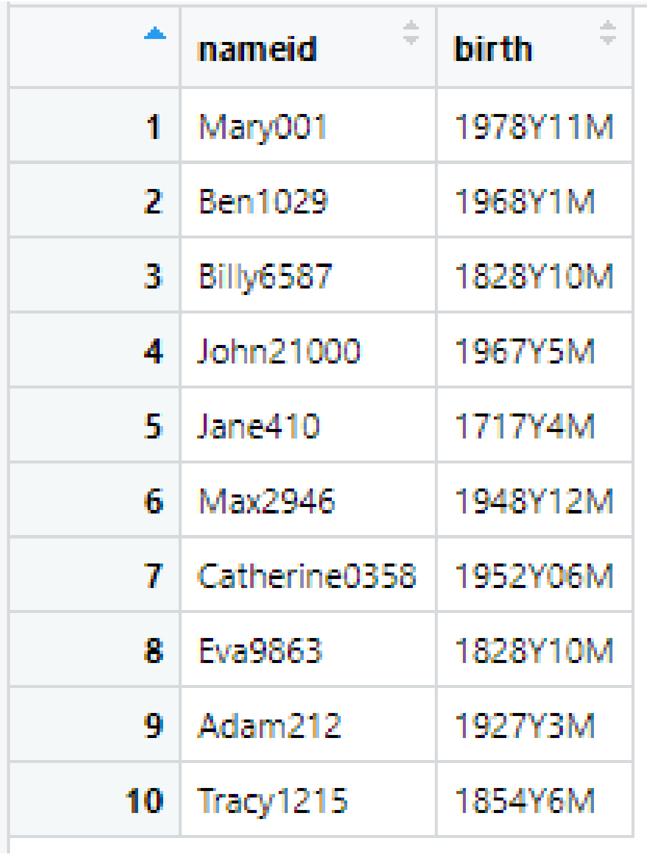
Introduction

```
np <- regexpr("[A-Za-z]*",
re3$nameid)

np
attr(np, "match.length")
```



Quantifier	Description
?	the preceding item is optional and will be matched at most once
*	the preceding item will be matched zero or more times
+	the preceding item will be matched one or more times
{n}	the preceding item is matched exactly n times
{n,}	the preceding item is matched n or more times
{n,m}	the preceding item is matched at least n times, but not more than m times

*adapted from *Handling and Processing Strings in R* (Sanchez, 2013)

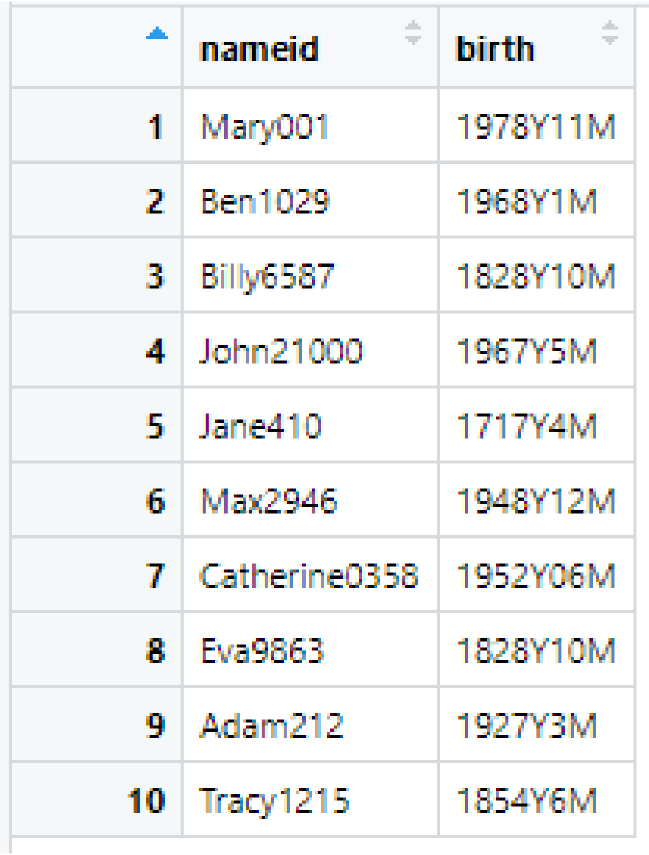


	nameid	birth
1	Mary001	1978Y11M
2	Ben1029	1968Y1M
3	Billy6587	1828Y10M
4	John21000	1967Y5M
5	Jane410	1717Y4M
6	Max2946	1948Y12M
7	Catherine0358	1952Y06M
8	Eva9863	1828Y10M
9	Adam212	1927Y3M
10	Tracy1215	1854Y6M

02 Regular Expression

Introduction

```
re3$name <- substr(re3$nameid,
np, attr(np, "match.length"))

re3$id <- substr(re3$nameid,
attr(np, "match.length") + 1,
nchar(re3$nameid))
```



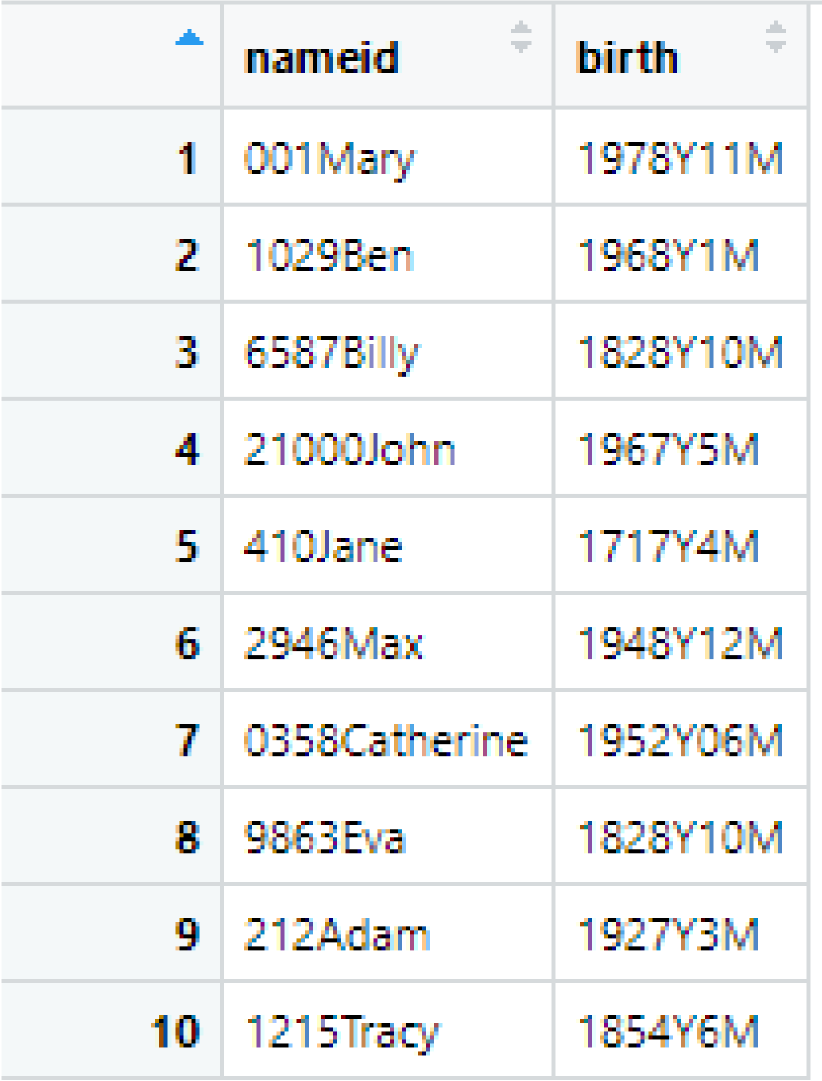
	nameid	birth
1	Mary001	1978Y11M
2	Ben1029	1968Y1M
3	Billy6587	1828Y10M
4	John21000	1967Y5M
5	Jane410	1717Y4M
6	Max2946	1948Y12M
7	Catherine0358	1952Y06M
8	Eva9863	1828Y10M
9	Adam212	1927Y3M
10	Tracy1215	1854Y6M

02 Regular Expression

Introduction

Practice1

Run ex_re3 dataframe codes. To separate name and id.



	nameid	birth
1	001Mary	1978Y11M
2	1029Ben	1968Y1M
3	6587Billy	1828Y10M
4	21000John	1967Y5M
5	410Jane	1717Y4M
6	2946Max	1948Y12M
7	0358Catherine	1952Y06M
8	9863Eva	1828Y10M
9	212Adam	1927Y3M
10	1215Tracy	1854Y6M

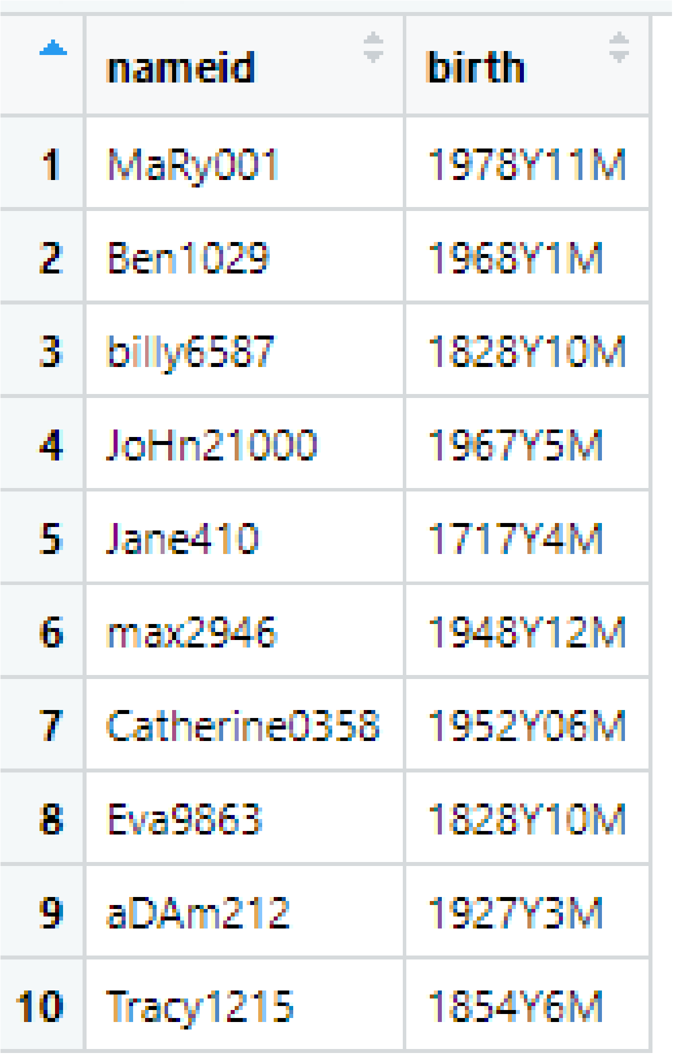
02

Regular Expression

Introduction

Typos II: re4

First letter: upper
Other letters: lower



	nameid	birth
1	MaRy001	1978Y11M
2	Ben1029	1968Y1M
3	billy6587	1828Y10M
4	JoHn21000	1967Y5M
5	Jane410	1717Y4M
6	max2946	1948Y12M
7	Catherine0358	1952Y06M
8	Eva9863	1828Y10M
9	aDAm212	1927Y3M
10	Tracy1215	1854Y6M

02

Regular Expression

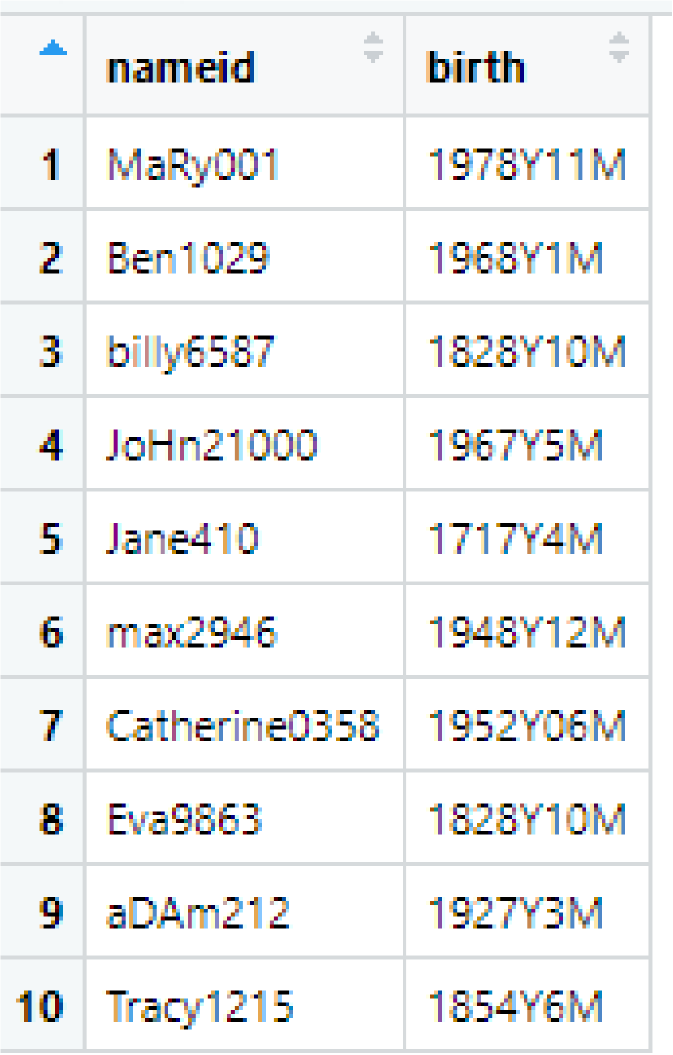
Introduction

Typos II: re4

First letter: upper
Other letters: lower

Use

1. toupper() for the first letter
2. tolower() for the other letters
3. Paste() them together



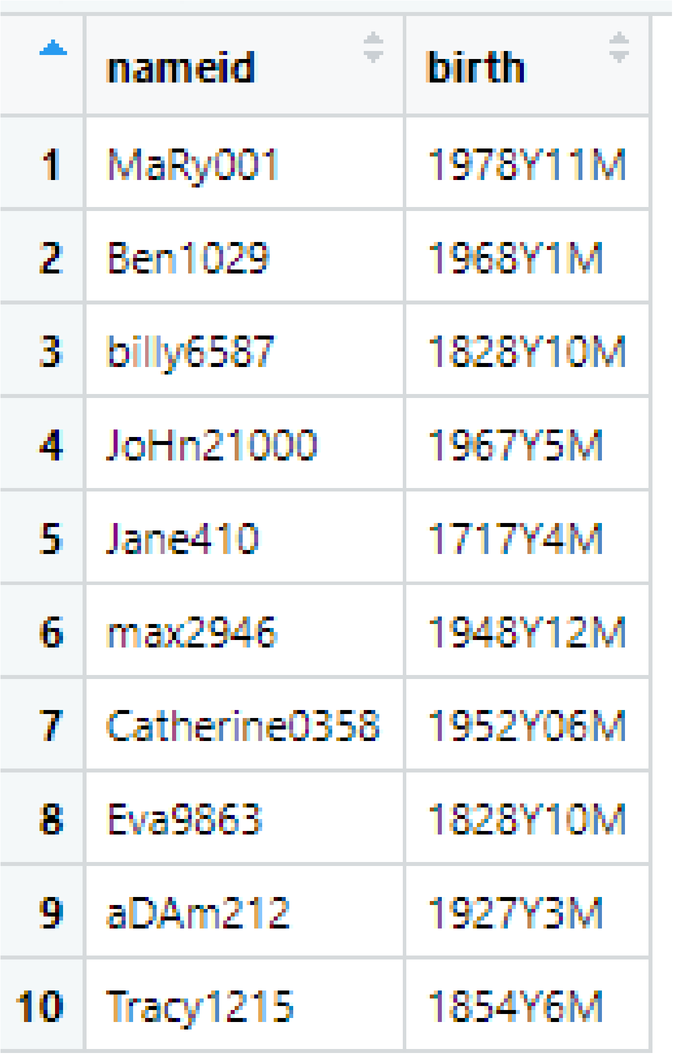
	nameid	birth
1	MaRy001	1978Y11M
2	Ben1029	1968Y1M
3	billy6587	1828Y10M
4	JoHn21000	1967Y5M
5	Jane410	1717Y4M
6	max2946	1948Y12M
7	Catherine0358	1952Y06M
8	Eva9863	1828Y10M
9	aDAm212	1927Y3M
10	Tracy1215	1854Y6M

02

Regular Expression

Introduction

```
np <- regexpr("[A-Za-z]*", re4$nameid)
np

re4$name <- substr(re4$nameid, np, attr(np,
"match.length"))

re4$id <- substr(re4$nameid, attr(np,
"match.length") + 1, nchar(re4$nameid))

re4$name_fix <-
paste0(toupper(substr(re4$name, 1, 1)),
tolower(substr(re4$name, 2,
nchar(re4$name))))
```



	nameid	birth
1	MaRy001	1978Y11M
2	Ben1029	1968Y1M
3	billy6587	1828Y10M
4	JoHn21000	1967Y5M
5	Jane410	1717Y4M
6	max2946	1948Y12M
7	Catherine0358	1952Y06M
8	Eva9863	1828Y10M
9	aDAm212	1927Y3M
10	Tracy1215	1854Y6M

02

Regular Expression

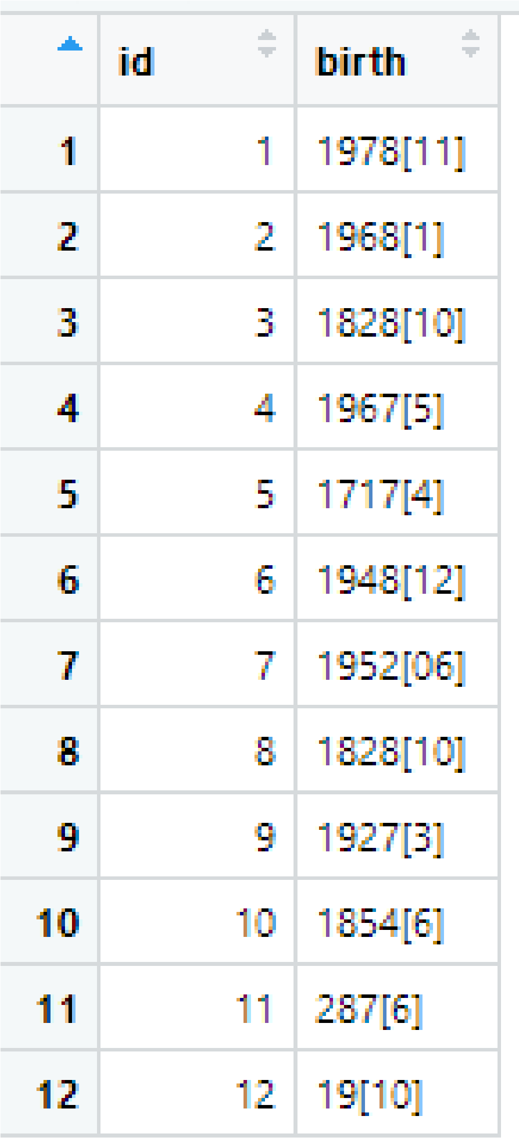
Introduction

Specific marks

Using [to identify location

yp <- regexpr("[", re5$birth)

yp <- regexpr("\\[", re5$birth)



	id	birth
1	1	1978[11]
2	2	1968[1]
3	3	1828[10]
4	4	1967[5]
5	5	1717[4]
6	6	1948[12]
7	7	1952[06]
8	8	1828[10]
9	9	1927[3]
10	10	1854[6]
11	11	287[6]
12	12	19[10]

02 Regular Expression

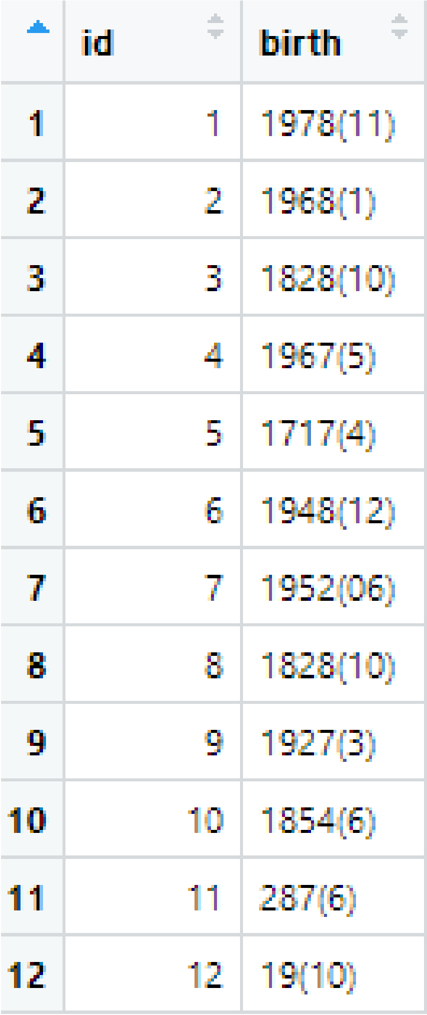
Introduction

Specific marks

Using (to identify location

Run ex_re5 dataframe codes

Practice 2



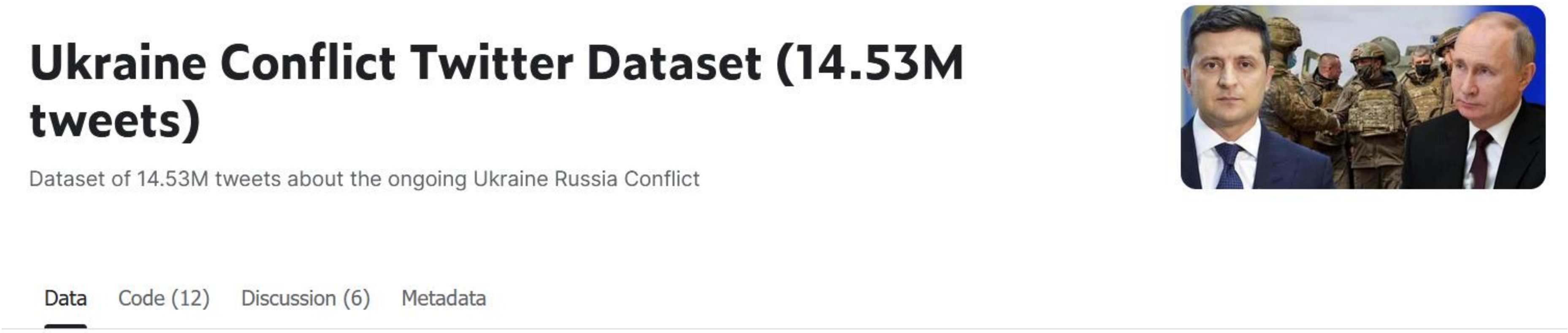
	id	birth
1	1	1978(11)
2	2	1968(1)
3	3	1828(10)
4	4	1967(5)
5	5	1717(4)
6	6	1948(12)
7	7	1952(06)
8	8	1828(10)
9	9	1927(3)
10	10	1854(6)
11	11	287(6)
12	12	19(10)

Ukraine Conflict Twitter

03

Ukraine Conflict Twitter

Introduction



Ukraine Conflict Twitter Dataset (14.53M tweets)

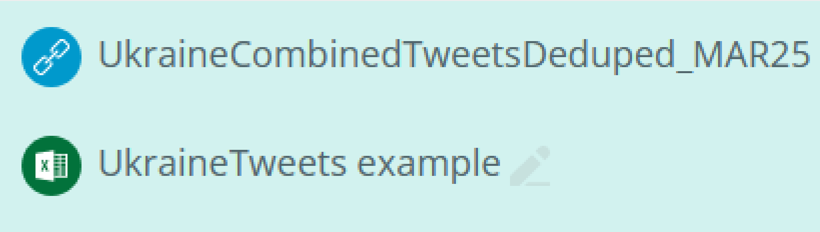
Dataset of 14.53M tweets about the ongoing Ukraine Russia Conflict

In Kaggle, you can download all Urkraine conflict tweets after March 25.

03

Ukraine Conflict Twitter

Introduction



UkraineCombinedTweetsDeduped_MAR25

UkraineTweets example

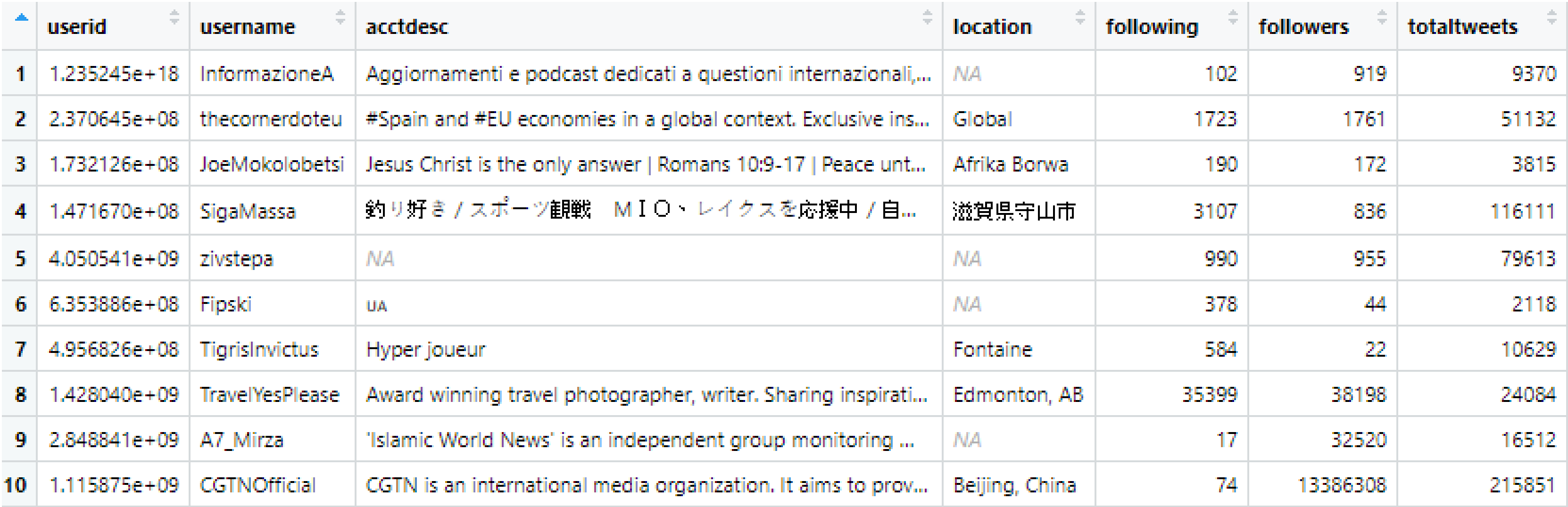
You can download March 25’s tweets and read the contend

I sliced 10 observations for you to do exercise.

```
ua_example <-
        read_xlsx("UkraineTweets_example.xlsx")
```

03 Ukraine Conflict Twitter

Introduction



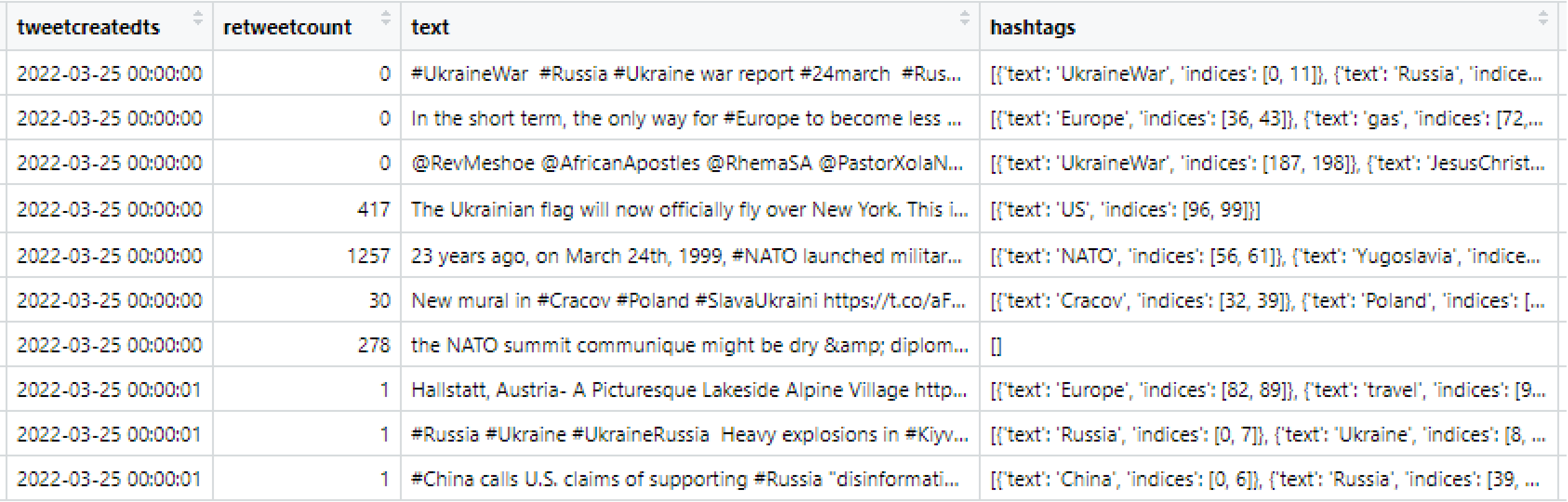
	userid	username	acctdesc	location	following	followers	totaltweets
1	1.235245e+18	InformazioneA	Aggiornamenti e podcast dedicati a questioni internazionali,...	NA	102	919	9370
2	2.370645e+08	thecornerdoteu	#Spain and #EU economies in a global context. Exclusive ins...	Global	1723	1761	51132
3	1.732126e+08	JoeMokolobetsi	Jesus Christ is the only answer | Romans 10:9-17 | Peace unt...	Afrika Borwa	190	172	3815
4	1.471670e+08	SigaMassa	釣り好き／スポーツ観戦　ＭＩＯ、レイクスを応援中／自...	滋賀県守山市	3107	836	116111
5	4.050541e+09	zivstepa	NA	NA	990	955	79613
6	6.353886e+08	Fipski	UA	NA	378	44	2118
7	4.956826e+08	TigrisInvictus	Hyper joueur	Fontaine	584	22	10629
8	1.428040e+09	TravelYesPlease	Award winning travel photographer, writer. Sharing inspirati...	Edmonton, AB	35399	38198	24084
9	2.848841e+09	A7_Mirza	'Islamic World News' is an independent group monitoring ...	NA	17	32520	16512
10	1.115875e+09	CGTNOfficial	CGTN is an international media organization. It aims to prov...	Beijing, China	74	13386308	215851

10 observations and 17 columns

03

Ukraine Conflict Twitter

Introduction



tweetcreatedts	retweetcount	text	hashtags
2022-03-25 00:00:00	0	#UkraineWar #Russia #Ukraine war report #24march #Rus...	[{'text': 'UkraineWar', 'indices': [0, 11]}, {'text': 'Russia', 'indice...
2022-03-25 00:00:00	0	In the short term, the only way for #Europe to become less ...	[{'text': 'Europe', 'indices': [36, 43]}, {'text': 'gas', 'indices': [72,...
2022-03-25 00:00:00	0	@RevMeshoe @AfricanApostles @RhemaSA @PastorXolaN...	[{'text': 'UkraineWar', 'indices': [187, 198]}, {'text': 'JesusChrist...
2022-03-25 00:00:00	417	The Ukrainian flag will now officially fly over New York. This i...	[{'text': 'US', 'indices': [96, 99]}]
2022-03-25 00:00:00	1257	23 years ago, on March 24th, 1999, #NATO launched militar...	[{'text': 'NATO', 'indices': [56, 61]}, {'text': 'Yugoslavia', 'indice...
2022-03-25 00:00:00	30	New mural in #Cracov #Poland #SlavaUkraini https://t.co/aF...	[{'text': 'Cracov', 'indices': [32, 39]}, {'text': 'Poland', 'indices': [...
2022-03-25 00:00:00	278	the NATO summit communique might be dry &amp; diplom...	[]
2022-03-25 00:00:01	1	Hallstatt, Austria- A Picturesque Lakeside Alpine Village http...	[{'text': 'Europe', 'indices': [82, 89]}, {'text': 'travel', 'indices': [9...
2022-03-25 00:00:01	1	#Russia #Ukraine #UkraineRussia Heavy explosions in #Kiyv...	[{'text': 'Russia', 'indices': [0, 7]}, {'text': 'Ukraine', 'indices': [8, ...
2022-03-25 00:00:01	1	#China calls U.S. claims of supporting #Russia "disinformati...	[{'text': 'China', 'indices': [0, 6]}, {'text': 'Russia', 'indices': [39, ...

Apply regular expression to get all hashtages in text column

Ukraine Conflict Twitter

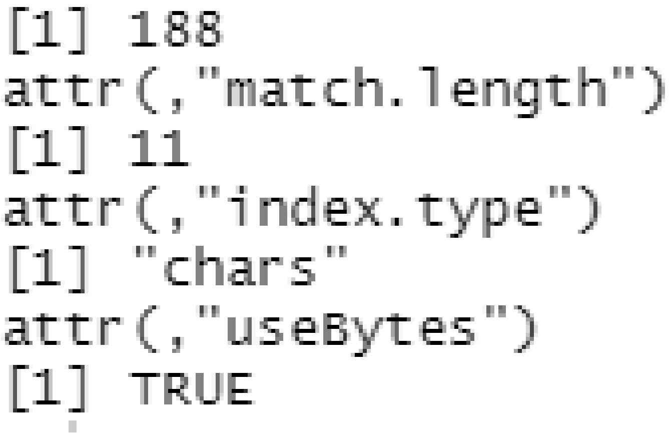
```
ua_example$text[3]

[1] "@RevMeshoe @AfricanApostles @RhemaSA
@PastorXolaNzo @Creflo_Dollar @BishopJakes @Benny_Hinn
@perrystonevoe @JosephPrince @Paula_White @Israel
@JustinWelby @MRCza @VP @BBC\n\nUnder cover of
#UkraineWar, the diabolical efficacy of COVID vaccines is playing
out in the UK\n\n#JesusChrist https://t.co/gPHib20XhI"
```

03

Ukraine Conflict Twitter

Introduction

```
ht <- regexpr("#[a-zA-Z0-9]+", ua_example$text[3])

ht
```



```
[1] 188
attr(,"match.length")
[1] 11
attr(,"index.type")
[1] "chars"
attr(,"useBytes")
[1] TRUE
```

03 Ukraine Conflict Twitter

Introduction

```
ht <- regexpr("#[a-zA-Z0-9]+", ua_example$text[3])

ht

substr(ua_example$text[3], ht,
                    ht + attr(ht, "match.length") - 1)

[1] "#UkraineWar"
```

03

Ukraine Conflict Twitter

Introduction

```
ua_example$text[3]

[1] "@RevMeshoe @AfricanApostles @RhemaSA
@PastorXolaNzo @Creflo_Dollar @BishopJakes @Benny_Hinn
@perrystonevoe @JosephPrince @Paula_White @Israel
@JustinWelby @MRCza @VP @BBC\n\nUnder cover of
#UkraineWar, the diabolical efficacy of COVID vaccines is playing
out in the UK\n\n#JesusChrist https://t.co/gPHib20XhI"
```

regexpr() only gives us the first pattern’s location and length.

gregexpr() can meet our requirement.

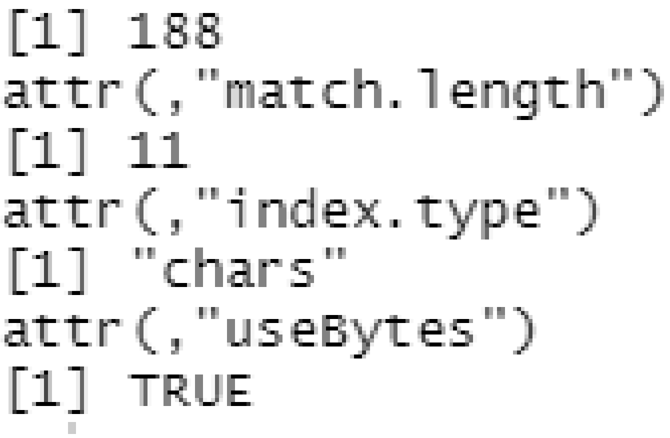
03

Ukraine Conflict Twitter

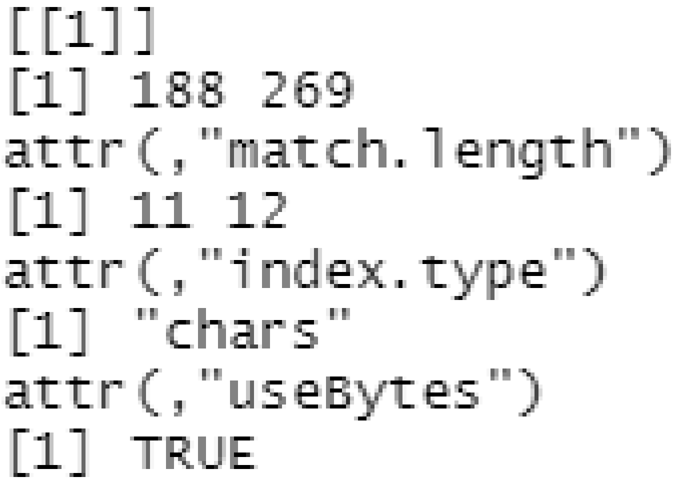
Introduction

ht <- gregexpr("#[a-zA-Z0-9]+", ua_example$text[3])

ht



```
[1] 188
attr(,"match.length")
[1] 11
attr(,"index.type")
[1] "chars"
attr(,"useBytes")
[1] TRUE
```

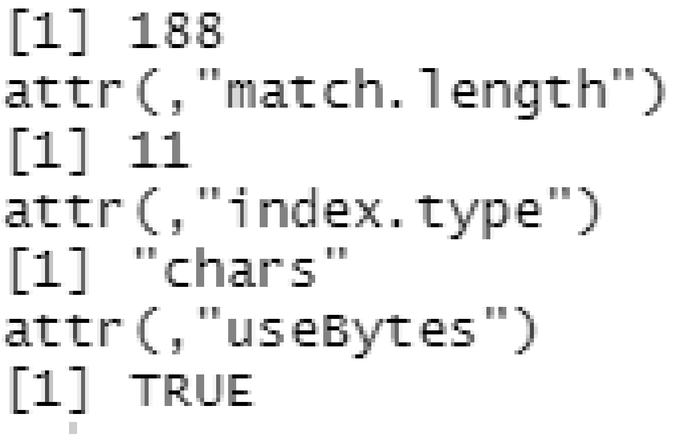


```
[[1]]
[1] 188 269
attr(,"match.length")
[1] 11 12
attr(,"index.type")
[1] "chars"
attr(,"useBytes")
[1] TRUE
```

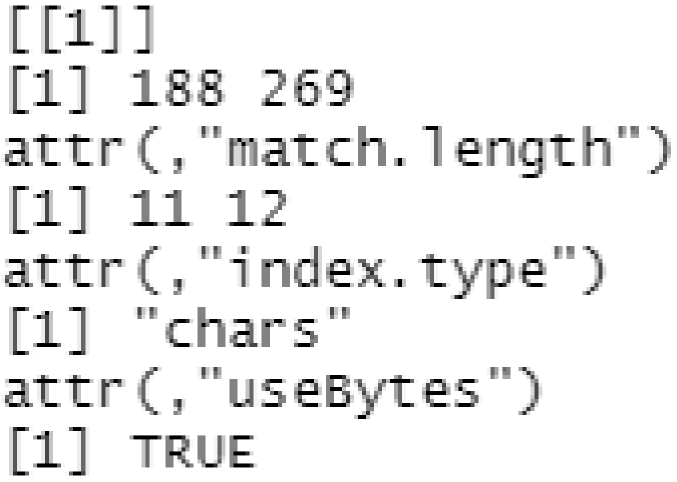
03

Ukraine Conflict Twitter

Introduction



```
[1] 188
attr(,"match.length")
[1] 11
attr(,"index.type")
[1] "chars"
attr(,"useBytes")
[1] TRUE
```



```
[[1]]
[1] 188 269
attr(,"match.length")
[1] 11 12
attr(,"index.type")
[1] "chars"
attr(,"useBytes")
[1] TRUE
```

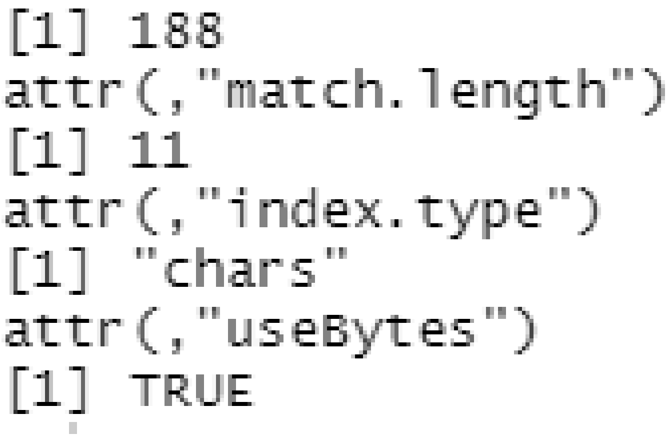
ht[[1]][1]

attr(ht[[1]], "match.length")[1]

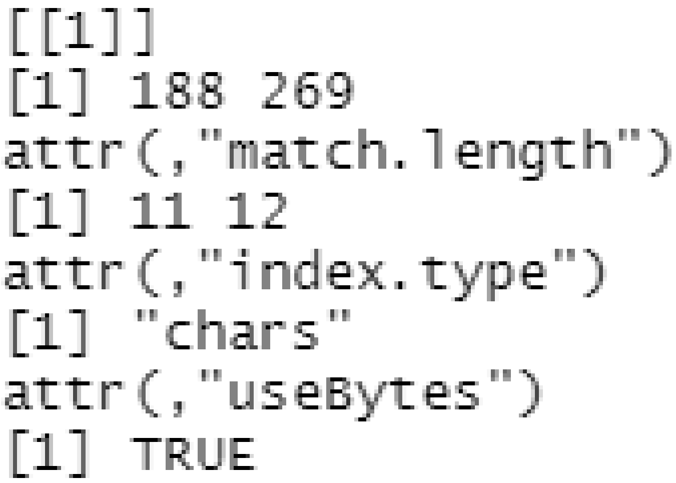
03

Ukraine Conflict Twitter

Introduction



```
[1] 188
attr(,"match.length")
[1] 11
attr(,"index.type")
[1] "chars"
attr(,"useBytes")
[1] TRUE
```



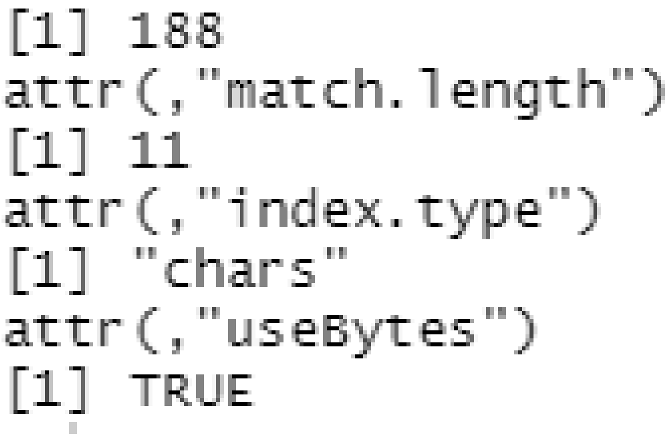
```
[[1]]
[1] 188 269
attr(,"match.length")
[1] 11 12
attr(,"index.type")
[1] "chars"
attr(,"useBytes")
[1] TRUE
```

```
substr(ua_example$text[3], ht[[1]][1],
         ht[[1]][1] + attr(ht[[1]], "match.length")[1])

substr(ua_example$text[3], ht[[1]][1],
         ht[[1]][1] + attr(ht[[1]], "match.length")[1] - 1)
```

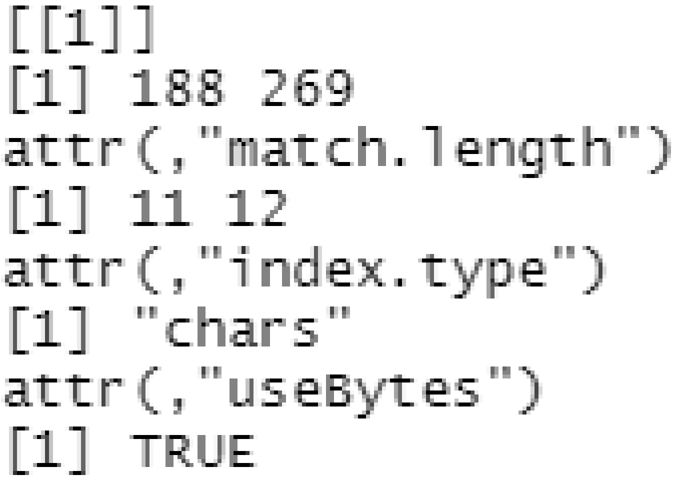
03

Ukraine Conflict Twitter

Introduction



```
[1] 188
attr(,"match.length")
[1] 11
attr(,"index.type")
[1] "chars"
attr(,"useBytes")
[1] TRUE
```



```
[[1]]
[1] 188 269
attr(,"match.length")
[1] 11 12
attr(,"index.type")
[1] "chars"
attr(,"useBytes")
[1] TRUE
```

substr(ua_example$text[3], ht[[1]][1,
ht[[1]][1] + attr(ht[[1]], "match.length")[1] - 1)

Give me the second hashtag (Practice)

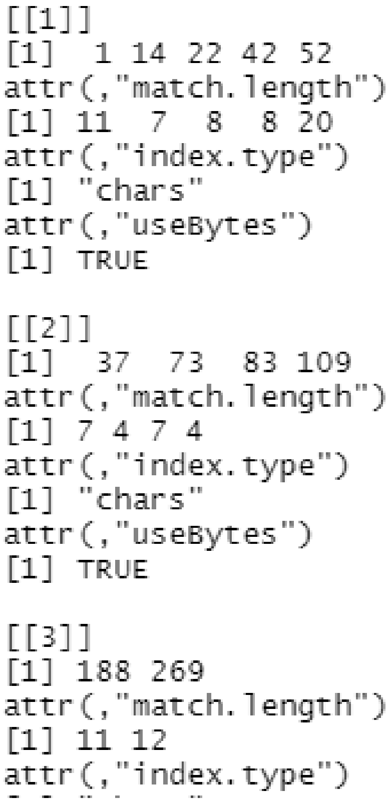
03

Ukraine Conflict Twitter

Introduction

ht <- gregexpr("#[a-zA-Z0-9]+", ua_example$text~~[3]~~)

ht



```
[[1]]
[1]  1 14 22 42 52
attr(,"match.length")
[1] 11  7  8  8 20
attr(,"index.type")
[1] "chars"
attr(,"useBytes")
[1] TRUE

[[2]]
[1]  37  73  83 109
attr(,"match.length")
[1] 7 4 7 4
attr(,"index.type")
[1] "chars"
attr(,"useBytes")
[1] TRUE

[[3]]
[1] 188 269
attr(,"match.length")
[1] 11 12
attr(,"index.type")
```

We can use loops to extract all hashtags in ua_example

03 Ukraine Conflict Twitter

Introduction

```
ht_df <- data.frame()

for (i in 1:length(ht)) {

  ht_temp <- ht[[i]]
  df_temp <- data.frame()

  for(x in 1:length(ht_temp)) {

    ht_content <- substr(ua_example$text[i], ht_temp[x],
                         ht_temp[x] + attr(ht_temp, "match.length")[[x]] - 1)
    temp <- data.frame(tweetid = ua_example$tweetid[i],
                         hashtag = ht_content)
      df_temp <- rbind(df_temp, temp)

  }

  ht_df <- rbind(ht_df, df_temp)

}
```

03

Ukraine Conflict Twitter

Introduction

```
ua_example$text[3]
```

```
[1] "@RevMeshoe @AfricanApostles @RhemaSA
@PastorXolaNzo @Creflo_Dollar @BishopJakes @Benny_Hinn
@perrystonevoe @JosephPrince @Paula_White @Israel
@JustinWelby @MRCza @VP @BBC\n\nUnder cover of
#UkraineWar, the diabolical efficacy of COVID vaccines is playing
out in the UK\n\n#JesusChrist https://t.co/gPHib20XhI"
```

Apply regular expression to get all hypelinks in text column