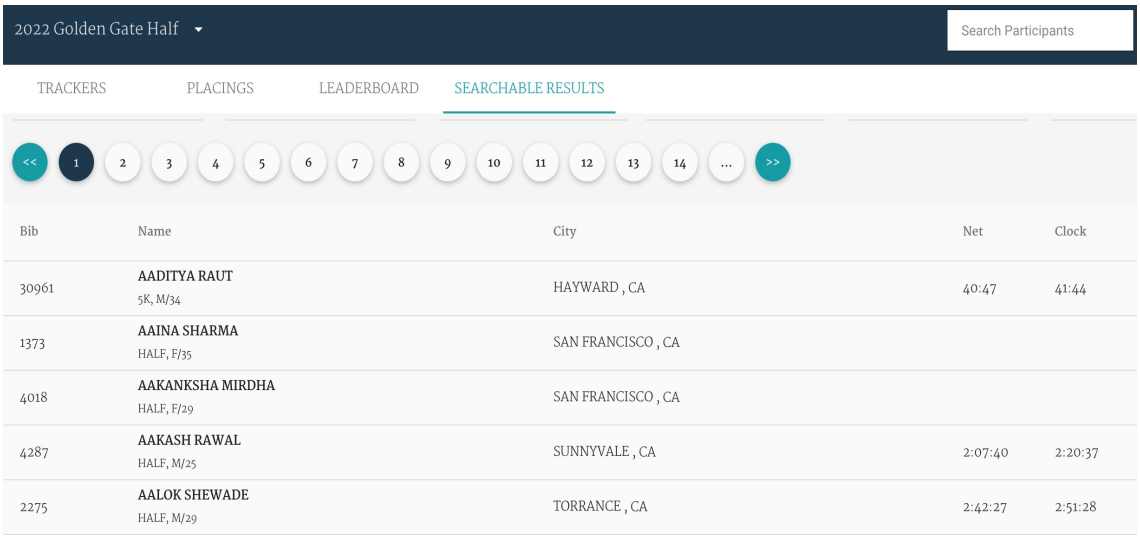
Text Analysis, Regular Expression, Data Cleaning

(Jing Li, Miami University)

1. This note provides one example of analyzing text data and another example of data cleaning with the tool of regular expression. For instance, the webpage below

   `http://results2.xacte.com/#/e/2459/searchable`

   contains information about runners in a competition:



Bib	Name	City	Net	Clock
30961	AADITYA RAUT 5K, M/34	HAYWARD , CA	40:47	41:44
1373	AAINA SHARMA HALF, F/35	SAN FRANCISCO , CA		
4018	AAKANKSHA MIRDHA HALF, F/29	SAN FRANCISCO , CA		
4287	AAKASH RAWAL HALF, M/25	SUNNYVALE , CA	2:07:40	2:20:37
2275	AALOK SHEWADE HALF, M/29	TORRANCE , CA	2:42:27	2:51:28

   We want to conduct a statistical analysis examining how the performance of a runner (Net, the second to the last column) is related to age, gender, etc. But the web does not offer a link to download data.

2. To scrape data from the webpage, I just highlight, copy, and paste the records for the first 21 runners into Excel, and save it as tab delimitated txt file.

3. R can read the txt file with function readLines. The first six observations are

```
> d = readLines("run.txt")
> head(d)
 [1] "30961\tAADITYA RAUT\t\"HAYWARD , CA\"\t40:47:00\t41:44:00"
 [2] "\t\"5K, M/34\"\t\t\t"
 [3] "1373\tAAINA SHARMA\t\"SAN FRANCISCO , CA\"\t\t"
 [4] "\t\"HALF, F/35\"\t\t\t"
```

```
[5] "4018\tAAKANKSHA MIRDHA\t\"SAN FRANCISCO , CA\"\t\t"
[6] "\t\"HALF, F/29\"\t\t\t"
```

Note that every runner has two entries (rows) in the data.

4. This data is not ready for R analysis. We need to remove the first troublemaker—double quotation mark shown in the middle of string.

```
> d1 = gsub("\"", "", d)
> head(d1)
[1] "30961\tAADITYA RAUT\tHAYWARD , CA\t40:47:00\t41:44:00"
[2] "\t5K, M/34\t\t\t"
[3] "1373\tAAINA SHARMA\tSAN FRANCISCO , CA\t\t"
[4] "\tHALF, F/35\t\t\t"
[5] "4018\tAAKANKSHA MIRDHA\tSAN FRANCISCO , CA\t\t"
[6] "\tHALF, F/29\t\t\t"
```

where gsub function replaces slash quotation in the middle of string with nothing.

5. Then we replace the second troublemaker—slash t with $< b >$, which effectively separates information as a new delimiter

```
> d2 = gsub("\t", "<b>", d1)
> head(d2)
[1] "30961<b>AADITYA RAUT<b>HAYWARD , CA<b>40:47:00<b>41:44:00"
[2] "<b>5K, M/34<b><b><b>"
[3] "1373<b>AAINA SHARMA<b>SAN FRANCISCO , CA<b><b>"
[4] "<b>HALF, F/35<b><b><b>"
[5] "4018<b>AAKANKSHA MIRDHA<b>SAN FRANCISCO , CA<b><b>"
[6] "<b>HALF, F/29<b><b><b>"
```

6. An efficient way to extract (parse) information from text is utilizing regular expression that describes the search pattern. A good starting point to learn it is checking out

```
https://bookdown.org/rdpeng/rprogdatascience/regular-expressions.html
https://www.youtube.com/watch?v=NvHjYOilOf8
https://www.youtube.com/watch?v=q8SzNKib5-4
```

7. In this case, the easiest information to extract is whether a runner runs 5K or HALF.

```
> r = regexpr("5K|HALF",d2)
> group = regmatches(d2,r)
> head(group)
[1] "5K"   "HALF" "HALF" "HALF" "HALF" "HALF"
```

   Google or use Chatgpt to learn more about the regexpr and regmatches functions. They are the key functions handelling regular expression.

8. The pattern for sex is that the value always appears after a comma and space, but in front of a slash

```
> r = regexpr(", [M|F]/",d2)
> m = regmatches(d2,r)
> head(m)
[1] ", M/" ", F/" ", F/" ", M/" ", M/" ", M/"
> sex = substr(m,nchar(m)-1,nchar(m)-1)
> head(sex)
[1] "M" "F" "F" "M" "M" "M"
```

   Note that we use function substr to extract part of a string based on the location.

9. The pattern for age is that they are two digits that always appear immediately after MF slash

```
> r = regexpr("[M|F]/[0-9][0-9]",d2)
> m = regmatches(d2,r)
> head(m)
[1] "M/34" "F/35" "F/29" "M/25" "M/29" "M/20"
> age = substr(m,nchar(m)-1,nchar(m))
> head(age)
[1] "34" "35" "29" "25" "29" "20"
```

10. The state are always two uppercase letters that appear after space, comma, and space

```
> r = regexpr(" , [A-Z][A-Z]",d2)
> m = regmatches(d2,r)
> head(m)
[1] " , CA" " , CA" " , CA" " , CA" " , CA" " , CA"
> state = substr(m,nchar(m)-1,nchar(m))
> head(state)
[1] "CA" "CA" "CA" "CA" "CA" "CA"
```

11. Extracting Net is trickier because they are missing (not available NA) for some runners, and the length is not fixed. Nevertheless, they show the pattern that two digits followed by : followed by two digits.

```
> index = 1:length(d2)
> inde = index[index%%2!=0]
> net = rep(0, length(d2)/2)
> r = regexpr("(.:)?[0-9][0-9]:[0-9][0-9]",d2)
> net = ifelse(r[inde]==-1,NA,net)
> net[!is.na(net)] = regmatches(d2,r)
> head(net)
[1] "40:47"   NA        NA        "2:07:40" "2:42:27" "2:14:39"
```

12. We may extract name in three steps

```
> r = regexpr("[0-9]?[0-9][0-9][0-9][0-9]<b>[A-Z -]+<b>",d2)
> m = regmatches(d2,r)
> head(m)
[1] "30961<b>AADITYA RAUT<b>"    "1373<b>AAINA SHARMA<b>"     "4018<b>AAKANKSHA MIR
[4] "4287<b>AAKASH RAWAL<b>"     "2275<b>AALOK SHEWADE<b>"    "5729<b>AARADHYA POUD
> r = regexpr("<b>[A-Z -]+<b>",m)
> m = regmatches(m,r)
> head(m)
[1] "<b>AADITYA RAUT<b>"     "<b>AAINA SHARMA<b>"     "<b>AAKANKSHA MIRDHA<b>" "<b>
[5] "<b>AALOK SHEWADE<b>"    "<b>AARADHYA POUDYAL<b>"
> name = gsub("<b>","",m)
> head(name)
[1] "AADITYA RAUT"     "AAINA SHARMA"     "AAKANKSHA MIRDHA" "AAKASH RAWAL"     "AA
```

```
[6] "AARADHYA POUDYAL"
```

13. Now we are ready to put all information in a data frame that R can analyze

```
> da = data.frame(name,age,sex,state,net)
> head(da)
              name age sex state     net
1     AADITYA RAUT  34   M    CA   40:47
2     AAINA SHARMA  35   F    CA    <NA>
3 AAKANKSHA MIRDHA  29   F    CA    <NA>
4     AAKASH RAWAL  25   M    CA 2:07:40
5    AALOK SHEWADE  29   M    CA 2:42:27
6 AARADHYA POUDYAL  20   M    CA 2:14:39
```

14. Statistical analysis now is doable. Just keep in mind that currently all variables are character. We need to coerce values to a different type if necessary.

```
> str(da)
'data.frame': 21 obs. of  5 variables:
 $ name : chr  "AADITYA RAUT" "AAINA SHARMA" "AAKANKSHA MIRDHA" "AAKASH RAWAL" ...
 $ age  : chr  "34" "35" "29" "25" ...
 $ sex  : chr  "M" "F" "F" "M" ...
 $ state: chr  "CA" "CA" "CA" "CA" ...
 $ net  : chr  "40:47" NA NA "2:07:40" ...
> mean(as.numeric(da$age))
[1] 36.95238
> table(factor(da$sex))
 F  M
 2 19
```

15. (Exercise): how to use regular expression to extract the recipients from the email address. For instance, how to extract "mr.A" and "jingli" from *mr.A@ohio.gov* and *jingli@miamioh.edu*. How to extract "gov" and "edu"?

16. (Exercise) Use regular expression to count how many words end with "ad" in the sentence "Dad had a bad day, so is sad."

Data Cleaning

1. Now we look at an example of data cleaning. The following webpage

   `https://en.wikipedia.org/wiki/Member_states_of_the_World_Trade_Organization`

   has a table:

State	Date of membership application	Status[43]
Algeria	3 June 1987	Inactive since 2014
Andorra	4 July 1997	Inactive since 1999
Azerbaijan	30 June 1997	Work in progress
Bahamas	10 May 2001	Inactive since 2019
Belarus	23 September 1993	Inactive since 2019
Bhutan	1 September 1999	Inactive since 2008
Bosnia and Herzegovina	11 May 1999	Work in progress
Comoros	22 February 2007	Strategic focus
Curaçao[44]	31 October 2019[45]	Activation
Equatorial Guinea	19 February 2007	Activation
Ethiopia	13 January 2003	Work in progress
Holy See	None[a]	Observer since 1997[46]
Iran	19 July 1996	Inactive since 2011
Iraq	30 September 2004	Reactivation

   I copy and paste the table into Excel, and save it as comma separated csv file (because the original table has spaces in all three columns, so I need comma as delimiter). I use R function read.csv to read data

```
> d = read.csv("wto.csv",header=F)
> names(d)=c("country","applicationdate","status")
> d
                  country      applicationdate                  status
1                 Algeria              3-Jun-87     Inactive since 2014
2                 Andorra              4-Jul-97     Inactive since 1999
3              Azerbaijan             30-Jun-97        Work in progress
4                 Bahamas             10-May-01     Inactive since 2019
5                 Belarus             23-Sep-93     Inactive since 2019
6                  Bhutan              1-Sep-99     Inactive since 2008
7  Bosnia and Herzegovina             11-May-99        Work in progress
```

8	Comoros	22-Feb-07	Strategic focus
9	Curaao[44]	31 October 2019[45]	Activation
10	Equatorial Guinea	19-Feb-07	Activation
11	Ethiopia	13-Jan-03	Work in progress
12	Holy See	None[a]	Observer since 1997[46]
13	Iran	19-Jul-96	Inactive since 2011
14	Iraq	30-Sep-04	Reactivation
15	Lebanon[b]	30-Jan-99	Reactivation
16	Libya	10-Jun-04	Inactive since 2004
17	So Tom and Prncipe	14-Jan-05	Inactive since 2005
18	Serbia	23-Dec-04	Inactive since 2013
19	Somalia	12 December 2015[47]	Activation
20	South Sudan	5 December 2017[48]	Inactive since 2019
21	Sudan	11-Oct-94	Work in progress
22	Syria[b]	10-Oct-01	Inactive since 2010
23	Timor-Leste	9 April 2015[47]	Strategic focus
24	Turkmenistan[c]	24 November 2021[50]	Activation
25	Uzbekistan	8-Dec-94	Work in progress

Each variable (column) is "messy", and needs cleaning.

2. For country (first column), we need to remove brackets for Curaao, Lebanon... with gsub function and regular expression

```
dc = d
dc$country = gsub("\\[(.*)\\]", "", d$country)
```

3. For applicationdate (second column), we need to remove brackets as well. Plus, we need to standardize the format as day-month-year

```
dc$applicationdate = gsub("\\[(.*)\\]", "", d$applicationdate)

fdate_con = function(x) {
xs = strsplit(x,"\\s+")
day = xs[[1]][1]
mon = xs[[1]][2]
```

```
mon = substr(mon,1,3)
year = xs[[1]][3]
year = substr(year,nchar(year)-1,nchar(year))
return(paste(day,"-",mon,"-",year,sep=""))
}

r = regexpr("[0-9]?[0-9] (.*)",dc$applicationdate)
dc$applicationdate[r!=-1]=sapply(dc$applicationdate[r!=-1],fdate_con)
```

4. For status, we need to remove brackets, extract the first part of string, and extract year if necessary

```
dc$status = gsub("\\[(.*)\\]", "", d$status)
dc$statusn = sapply(strsplit(d$status, "\\s+"),"[",1)

dc$since = NA
r = regexpr("since",dc$status)
f_pick = function(x) {
return(strsplit(x, "\\s+")[[1]][3])
}
dc$since[r!=-1]=sapply(dc$status[r!=-1],f_pick)
```

The clean data look like

```
> dc[,-3]
                   country applicationdate      statusn since
1                  Algeria        3-Jun-87     Inactive  2014
2                  Andorra        4-Jul-97     Inactive  1999
3               Azerbaijan       30-Jun-97         Work  <NA>
4                  Bahamas       10-May-01     Inactive  2019
5                  Belarus       23-Sep-93     Inactive  2019
6                   Bhutan        1-Sep-99     Inactive  2008
7  Bosnia and Herzegovina       11-May-99         Work  <NA>
8                  Comoros       22-Feb-07    Strategic  <NA>
9                   Curaao       31-Oct-19   Activation  <NA>
10      Equatorial Guinea       19-Feb-07   Activation  <NA>
```

11	Ethiopia	13-Jan-03	Work	<NA>
12	Holy See	None	Observer	1997
13	Iran	19-Jul-96	Inactive	2011
14	Iraq	30-Sep-04	Reactivation	<NA>
15	Lebanon	30-Jan-99	Reactivation	<NA>
16	Libya	10-Jun-04	Inactive	2004
17	So Tom and Prncipe	14-Jan-05	Inactive	2005
18	Serbia	23-Dec-04	Inactive	2013
19	Somalia	12-Dec-15	Activation	<NA>
20	South Sudan	5-Dec-17	Inactive	2019
21	Sudan	11-Oct-94	Work	<NA>
22	Syria	10-Oct-01	Inactive	2010
23	Timor-Leste	9-Apr-15	Strategic	<NA>
24	Turkmenistan	24-Nov-21	Activation	<NA>
25	Uzbekistan	8-Dec-94	Work	<NA>

5. If data is small, you may manually clean data record after record in Excel. Regular expression becomes a must if there are thousands of observations.