Week4_Notes

March 5, 2024

1 Week 4 Notes - Getting and Cleaning Data

Final week of the course! Let's get it!

1.1 Editing text variables

Imagine we have imported a data frame, and upon perusing it, we realise that the column names are formatted inconsistently, with variable cases, numbers and other characters in them, perhaps spaces too, making them more difficult to work with than they need to be - what can we do?

[2]:
```
using DataFrames, CSV, DataFramesMeta, HTTP, Dates
```

1.1.1 mixed case formatting

In R we can use the **tolower()** function alongside the **names()** function to transform the column names to lower case - there is also likely an opposite **toupper()** function as well.

```
tolower(names(data))
```

In Julia we can directly rename the column names of the data frame to lowercase using the **rename()** function of DataFrames package

```
rename(lowercase, df)
```

And then some playing around with the functions directly

[6]:
```
sample_names = ["MusicData", "tommorrowTemp", "AttRibutes"]
```

[6]:
```
3-element Vector{String}:
 "MusicData"
 "tommorrowTemp"
 "AttRibutes"
```

[7]:
```
lowercase.(sample_names)
```

[7]:
```
3-element Vector{String}:
 "musicdata"
 "tommorrowtemp"
 "attributes"
```

1.1.2 Stripping, splitting characters

Say we have a variable which contains a dot followed by a number, and we're only interested in the string before the dot, we can strip and split these characters based on the character. In R;

```
splitNames = strsplit(names(cameraData, "\\.")
```

[17]:
```
sample_split = ["MusicData.3", "tommorrowTemp.3", "AttRibutes"]
```

[17]:
```
3-element Vector{String}:
 "MusicData.3"
 "tommorrowTemp.3"
 "AttRibutes"
```

Replace all digits and numbers with nothing - effectively deleting it.

[45]:
```
replace.(sample_split, r"\d" => "", "." => "")
```

[45]:
```
3-element Vector{String}:
 "MusicData"
 "tommorrowTemp"
 "AttRibutes"
```

This would be done with something like **gsub** in R;

```
gsub("_", "", testName)
```

Split the variable names on the dot character

[46]:
```
split_names = split.(sample_split, ".")
```

[46]:
```
3-element Vector{Vector{SubString{String}}}:
 ["MusicData", "3"]
 ["tommorrowTemp", "3"]
 ["AttRibutes"]
```

[50]:
```
split_names[1][1]
```

[50]:
```
"MusicData"
```

1.1.3 Finding values - grep and so on

R borrows directly from the GNU grep function, whereas Julia uses it's own **find**, **findall** etc. functions for this purpose - along with **in**, **all** etc..

Let's see R's;

```
grep("Alameda", data)
table(grepl("Alameda", data))
```

In julia we could use **findall()**;

```
findall("something", data)
```

1.2 Dates and Time

The best guide on this in julia is found at https://en.wikibooks.org/wiki/Introducing_Julia/Working_with_dates_ – very, very helpful

1.3 Quiz

1.4 1.

The American Community Survey distributes downloadable data about United States communities. Download the 2006 microdata survey about housing for the state of Idaho using download.file() from here:

https://d396qusza40orc.cloudfront.net/getdata%2Fdata%2Fss06hid.csv

and load the data into R. The code book, describing the variable names is here:

https://d396qusza40orc.cloudfront.net/getdata%2Fdata%2FPUMSDataDict06.pdf

Apply strsplit() to split all the names of the data frame on the characters "wgtp". What is the value of the 123 element of the resulting list?

[9]:
```
q1_csv = CSV.read(HTTP.get("https://d396qusza40orc.cloudfront.net/
↪getdata%2Fdata%2Fss06hid.csv").body, DataFrame)
```

[9]:

	RT	SERIALNO	DIVISION	PUMA	REGION	ST	ADJUST	WGTP	NP	TYPE	
	String1	Int64	Int64	Int64	Int64	Int64	Int64	Int64	Int64	Int64	
1	H	186	8	700	4	16	1015675	89	4	1	...
2	H	306	8	700	4	16	1015675	310	1	1	...
3	H	395	8	100	4	16	1015675	106	2	1	...
4	H	506	8	700	4	16	1015675	240	4	1	...
5	H	835	8	800	4	16	1015675	118	4	1	...
6	H	989	8	700	4	16	1015675	115	4	1	...
7	H	1861	8	700	4	16	1015675	0	1	2	...
8	H	2120	8	200	4	16	1015675	35	1	1	...
9	H	2278	8	400	4	16	1015675	47	2	1	...
10	H	2428	8	500	4	16	1015675	51	2	1	...
11	H	2677	8	400	4	16	1015675	114	2	1	...
12	H	2928	8	800	4	16	1015675	51	2	1	...
13	H	3283	8	400	4	16	1015675	67	3	1	...
14	H	3331	8	600	4	16	1015675	0	1	2	...
15	H	3341	8	500	4	16	1015675	92	1	1	...
16	H	3827	8	300	4	16	1015675	51	2	1	...
17	H	4331	8	800	4	16	1015675	71	3	1	...
18	H	4425	8	400	4	16	1015675	349	1	1	...
19	H	4457	8	900	4	16	1015675	270	4	1	...
20	H	4551	8	500	4	16	1015675	83	3	1	...
21	H	4609	8	600	4	16	1015675	78	1	1	...
22	H	4720	8	200	4	16	1015675	115	1	1	...
23	H	5191	8	600	4	16	1015675	314	2	1	...
24	H	5465	8	100	4	16	1015675	22	1	1	...
25	H	5590	8	100	4	16	1015675	73	2	1	...
26	H	5763	8	800	4	16	1015675	73	4	1	...
27	H	5890	8	800	4	16	1015675	39	1	1	...
28	H	6517	8	800	4	16	1015675	63	1	1	...
29	H	6581	8	300	4	16	1015675	88	1	1	...
30	H	6789	8	600	4	16	1015675	321	2	1	...
...	...	...	...	...	...	...	...	...	...	...	

[7]:
```
colnames = names(q1_csv)
```

[7]:
```
188-element Vector{String}:
 "RT"
 "SERIALNO"
 "DIVISION"
 "PUMA"
 "REGION"
 "ST"
 "ADJUST"
 "WGTP"
 "NP"
 "TYPE"
```

```
"ACR"
"AGS"
"BDS"

"wgtp69"
"wgtp70"
"wgtp71"
"wgtp72"
"wgtp73"
"wgtp74"
"wgtp75"
"wgtp76"
"wgtp77"
"wgtp78"
"wgtp79"
"wgtp80"
```

[15]:
```
split_colnames = split.(colnames, "wgtp")
```

[15]:
```
188-element Vector{Vector{SubString{String}}}:
 ["RT"]
 ["SERIALNO"]
 ["DIVISION"]
 ["PUMA"]
 ["REGION"]
 ["ST"]
 ["ADJUST"]
 ["WGTP"]
 ["NP"]
 ["TYPE"]
 ["ACR"]
 ["AGS"]
 ["BDS"]

 ["", "69"]
 ["", "70"]
 ["", "71"]
 ["", "72"]
 ["", "73"]
 ["", "74"]
 ["", "75"]
 ["", "76"]
 ["", "77"]
 ["", "78"]
 ["", "79"]
 ["", "80"]
```

[17]:
```
split_colnames[123]
```

[17]:
```
2-element Vector{SubString{String}}:
 ""
 "15"
```

The answer is “ ” “15”

1.5 2.

Load the Gross Domestic Product data for the 190 ranked countries in this data set:

https://d396qusza40orc.cloudfront.net/getdata%2Fdata%2FGDP.csv

Remove the commas from the GDP numbers in millions of dollars and average them. What is the average?

Original data sources:

http://data.worldbank.org/data-catalog/GDP-ranking-table

[22]:
```
gdp = download("https://d396qusza40orc.cloudfront.net/getdata%2Fdata%2FGDP.
↪csv", "gdp.csv")
```

[22]:
```
"gdp.csv"
```

[23]:
```
run(`head gdp.csv`)
```

```
,Gross domestic product 2012,,,,,,,
,,,,,,,,,
,,,,(millions of,,,,,
,Ranking,,Economy,US dollars),,,,,
,,,,,,,,,
USA,1,,United States," 16,244,600 ",,,,,
CHN,2,,China," 8,227,103 ",,,,,
JPN,3,,Japan," 5,959,718 ",,,,,
DEU,4,,Germany," 3,428,131 ",,,,,
FRA,5,,France," 2,612,878 ",,,,,
```

[23]:
```
Process(`head gdp.csv`, ProcessExited(0))
```

[31]:
```
gdp_raw = CSV.read("gdp.csv", DataFrame, skipto=6, header=4) ;␣
↪gdp_raw_nomissing = dropmissing(gdp_raw[:, [:1, :2, :4, :5]]) ; gdp_cleaned␣
↪= rename!(gdp_raw_nomissing, ["CountryCode", "Rank", "Country",␣
↪"US_dollars"])
```

[31]:

	CountryCode	Rank	Country	US_dollars
	String3	String	String31	String15
1	USA	1	United States	16,244,600
2	CHN	2	China	8,227,103
3	JPN	3	Japan	5,959,718
4	DEU	4	Germany	3,428,131
5	FRA	5	France	2,612,878
6	GBR	6	United Kingdom	2,471,784
7	BRA	7	Brazil	2,252,664
8	RUS	8	Russian Federation	2,014,775
9	ITA	9	Italy	2,014,670
10	IND	10	India	1,841,710
11	CAN	11	Canada	1,821,424
12	AUS	12	Australia	1,532,408
13	ESP	13	Spain	1,322,965
14	MEX	14	Mexico	1,178,126
15	KOR	15	Korea, Rep.	1,129,598
16	IDN	16	Indonesia	878,043
17	TUR	17	Turkey	789,257
18	NLD	18	Netherlands	770,555
19	SAU	19	Saudi Arabia	711,050
20	CHE	20	Switzerland	631,173
21	SWE	21	Sweden	523,806
22	IRN	22	Iran, Islamic Rep.	514,060
23	NOR	23	Norway	499,667
24	POL	24	Poland	489,795
25	BEL	25	Belgium	483,262
26	ARG	26	Argentina	475,502
27	AUT	27	Austria	394,708
28	ZAF	28	South Africa	384,313
29	VEN	29	Venezuela, RB	381,286
30	COL	30	Colombia	369,606
...	...	...	...	...

[32]:
```
gdp_cleaned.US_dollars .= replace.(gdp_cleaned.US_dollars, "," => "")
```

[32]:
```
190-element Vector{String}:
 " 16244600 "
 " 8227103 "
 " 5959718 "
 " 3428131 "
 " 2612878 "
 " 2471784 "
 " 2252664 "
 " 2014775 "
 " 2014670 "
 " 1841710 "
```

```
" 1821424 "
" 1532408 "
" 1322965 "

" 767 "
" 713 "
" 684 "
" 596 "
" 480 "
" 472 "
" 326 "
" 263 "
" 228 "
" 182 "
" 175 "
" 40 "
```

[33]:
```
gdp_cleaned.US_dollars = parse.(Int, gdp_cleaned.US_dollars)
```

[33]:
```
190-element Vector{Int64}:
 16244600
  8227103
  5959718
  3428131
  2612878
  2471784
  2252664
  2014775
  2014670
  1841710
  1821424
  1532408
  1322965

      767
      713
      684
      596
      480
      472
      326
      263
      228
      182
      175
       40
```

[34]:
```
using StatsBase
```

[35]:
```
mean(gdp_cleaned.US_dollars)
```

[35]:
```
377652.4210526316
```

The answer is 377652.4210526316

1.6 3.

Question 3 In the data set from Question 2 what is a regular expression that would allow you to count the number of countries whose name begins with "United"? Assume that the variable with the country names in it is named countryNames. How many countries begin with United?

[61]:
```
contains.(gdp_cleaned.Country, "United")
```

[61]:
```
190-element BitVector:
 1
 0
 0
 0
 0
 1
 0
 0
 0
 0
 0
 0
 0
 0

 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
```

[65]:
```
filter(x -> contains.(x.Country, "United"), gdp_cleaned)
```

[65]:

	CountryCode	Rank	Country	US_dollars
	String3	String	String31	Int64
1	USA	1	United States	16244600
2	GBR	6	United Kingdom	2471784
3	ARE	32	United Arab Emirates	348595

[42]:
```
occursin("United", "United States")
```

[42]:
```
true
```

[45]:
```
united_countries = occursin.("United", gdp_cleaned.Country)
```

[45]:
```
190-element BitVector:
 1
 0
 0
 0
 0
 1
 0
 0
 0
 0
 0
 0
 0

 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
```

[46]:
```
countmap(united_countries)
```

[46]:
```
Dict{Bool, Int64} with 2 entries:
  0 => 187
  1 => 3
```

1.7 4.

Load the Gross Domestic Product data for the 190 ranked countries in this data set:

https://d396qusza40orc.cloudfront.net/getdata%2Fdata%2FGDP.csv

Load the educational data from this data set:

https://d396qusza40orc.cloudfront.net/getdata%2Fdata%2FEDSTATS_Country.csv

Match the data based on the country shortcode. Of the countries for which the end of the fiscal year is available, how many end in June?

Same data as above, simply need to download the education data

[66]:
```
download("https://d396qusza40orc.cloudfront.net/
  ↪getdata%2Fdata%2FEDSTATS_Country.csv", "edu.csv")
```

[66]:
```
"edu.csv"
```

[68]:
```
edu_df = CSV.read("edu.csv", DataFrame)
```

[68]:

	CountryCode	Long Name	Income Group	Region
	String3	String	String31?	String31?
1	ABW	Aruba	High income: nonOECD	Latin America & Caribbean
2	ADO	Principality of Andorra	High income: nonOECD	Europe & Central Asia
3	AFG	Islamic State of Afghanistan	Low income	South Asia
4	AGO	People's Republic of Angola	Lower middle income	Sub-Saharan Africa
5	ALB	Republic of Albania	Upper middle income	Europe & Central Asia
6	ARE	United Arab Emirates	High income: nonOECD	Middle East & North Africa
7	ARG	Argentine Republic	Upper middle income	Latin America & Caribbean
8	ARM	Republic of Armenia	Lower middle income	Europe & Central Asia
9	ASM	American Samoa	Upper middle income	East Asia & Pacific
10	ATG	Antigua and Barbuda	Upper middle income	Latin America & Caribbean
11	AUS	Commonwealth of Australia	High income: OECD	East Asia & Pacific
12	AUT	Republic of Austria	High income: OECD	Europe & Central Asia
13	AZE	Republic of Azerbaijan	Upper middle income	Europe & Central Asia
14	BDI	Republic of Burundi	Low income	Sub-Saharan Africa
15	BEL	Kingdom of Belgium	High income: OECD	Europe & Central Asia
16	BEN	Republic of Benin	Low income	Sub-Saharan Africa
17	BFA	Burkina Faso	Low income	Sub-Saharan Africa
18	BGD	People's Republic of Bangladesh	Low income	South Asia
19	BGR	Republic of Bulgaria	Upper middle income	Europe & Central Asia
20	BHR	Kingdom of Bahrain	High income: nonOECD	Middle East & North Africa
21	BHS	Commonwealth of The Bahamas	High income: nonOECD	Latin America & Caribbean
22	BIH	Bosnia and Herzegovina	Upper middle income	Europe & Central Asia
23	BLR	Republic of Belarus	Upper middle income	Europe & Central Asia
24	BLZ	Belize	Lower middle income	Latin America & Caribbean
25	BMU	The Bermudas	High income: nonOECD	North America
26	BOL	Plurinational State of Bolivia	Lower middle income	Latin America & Caribbean
27	BRA	Federative Republic of Brazil	Upper middle income	Latin America & Caribbean
28	BRB	Barbados	High income: nonOECD	Latin America & Caribbean
29	BRN	Brunei Darussalam	High income: nonOECD	East Asia & Pacific
30	BTN	Kingdom of Bhutan	Lower middle income	South Asia
...	...	...	...	...

[70]:
```
edu_gdp = innerjoin(gdp_cleaned, edu_df, on="CountryCode")
```
[70]:

	CountryCode	Rank	Country	US_dollars	Long Name	
	String3	String	String31	Int64	String	
1	ABW	161	Aruba	2584	Aruba	...
2	AFG	105	Afghanistan	20497	Islamic State of Afghanistan	...
3	AGO	60	Angola	114147	People's Republic of Angola	...
4	ALB	125	Albania	12648	Republic of Albania	...
5	ARE	32	United Arab Emirates	348595	United Arab Emirates	...
6	ARG	26	Argentina	475502	Argentine Republic	...
7	ARM	133	Armenia	9951	Republic of Armenia	...
8	ATG	172	Antigua and Barbuda	1134	Antigua and Barbuda	...
9	AUS	12	Australia	1532408	Commonwealth of Australia	...
10	AUT	27	Austria	394708	Republic of Austria	...
11	AZE	68	Azerbaijan	66605	Republic of Azerbaijan	...
12	BDI	162	Burundi	2472	Republic of Burundi	...
13	BEL	25	Belgium	483262	Kingdom of Belgium	...
14	BEN	140	Benin	7557	Republic of Benin	...
15	BFA	128	Burkina Faso	10441	Burkina Faso	...
16	BGD	59	Bangladesh	116355	People's Republic of Bangladesh	...
17	BGR	76	Bulgaria	50972	Republic of Bulgaria	...
18	BHR	93	Bahrain	29044	Kingdom of Bahrain	...
19	BHS	138	Bahamas, The	8149	Commonwealth of The Bahamas	...
20	BIH	111	Bosnia and Herzegovina	17466	Bosnia and Herzegovina	...
21	BLR	69	Belarus	63267	Republic of Belarus	...
22	BLZ	169	Belize	1493	Belize	...
23	BMU	149	Bermuda	5474	The Bermudas	...
24	BOL	96	Bolivia	27035	Plurinational State of Bolivia	...
25	BRA	7	Brazil	2252664	Federative Republic of Brazil	...
26	BRB	153	Barbados	4225	Barbados	...
27	BRN	113	Brunei Darussalam	16954	Brunei Darussalam	...
28	BTN	167	Bhutan	1780	Kingdom of Bhutan	...
29	BWA	117	Botswana	14504	Republic of Botswana	...
30	CAF	165	Central African Republic	2184	Central African Republic	...
...	...	...	...	...	...	

[143]:
```
edu_gdp_special = filter(row -> any(!ismissing.(row.var"Special Notes")),edu_gdp)
```

[143]:

	CountryCode	Rank	Country	US_dollars	Long Nan
	String3	String	String31	Int64	String
1	AFG	105	Afghanistan	20497	Islamic State of A
2	ATG	172	Antigua and Barbuda	1134	Antigua and B
3	AUS	12	Australia	1532408	Commonwealth of
4	AUT	27	Austria	394708	Republic of A
5	BEL	25	Belgium	483262	Kingdom of B
6	BGD	59	Bangladesh	116355	People's Republic of
7	BHS	138	Bahamas, The	8149	Commonwealth of T
8	BLZ	169	Belize	1493	Belize
9	BMU	149	Bermuda	5474	The Bermu
10	BWA	117	Botswana	14504	Republic of Bo
11	CAN	11	Canada	1821424	Canada
12	CHN	2	China	8227103	People's Republic
13	CYP	102	Cyprus	22767	Republic of C
14	DEU	4	Germany	3428131	Federal Republic o
15	EGY	38	Egypt, Arab Rep.	262832	Arab Republic o
16	ESP	13	Spain	1322965	Kingdom of S
17	ETH	85	Ethiopia	41605	Federal Democratic Rep
18	FIN	43	Finland	247546	Republic of F
19	FRA	5	France	2612878	French Repu
20	FSM	185	Micronesia, Fed. Sts.	326	Federated States of
21	GMB	175	Gambia, The	917	Republic of The
22	GRC	42	Greece	249099	Hellenic Rep
23	GTM	77	Guatemala	50234	Republic of Gu
24	HKG	37	Hong Kong SAR, China	263259	Hong Kong Special Administrative Region
25	HRV	71	Croatia	59228	Republic of C
26	HTI	139	Haiti	7843	Republic of
27	IDN	16	Indonesia	878043	Republic of In
28	IND	10	India	1841710	Republic of
29	IRL	46	Ireland	210771	Ireland
30	IRN	22	Iran, Islamic Rep.	514060	Islamic Republi
...	...	...	...	...	...

[146]:
```
june_fiscal = filter(x -> contains.(x.var"Special Notes", "Fiscal year end: June"), edu_gdp_special)
```

[146]:

	CountryCode	Rank	Country	US_dollars	Long Name	
	String3	String	String31	Int64	String	
1	AUS	12	Australia	1532408	Commonwealth of Australia	...
2	BGD	59	Bangladesh	116355	People's Republic of Bangladesh	...
3	BWA	117	Botswana	14504	Republic of Botswana	...
4	EGY	38	Egypt, Arab Rep.	262832	Arab Republic of Egypt	...
5	GMB	175	Gambia, The	917	Republic of The Gambia	...
6	KEN	87	Kenya	40697	Republic of Kenya	...
7	KWT	56	Kuwait	160913	State of Kuwait	...
8	PAK	44	Pakistan	225143	Islamic Republic of Pakistan	...
9	PRI	61	Puerto Rico	101496	Puerto Rico	...
10	SLE	157	Sierra Leone	3796	Republic of Sierra Leone	...
11	SWE	21	Sweden	523806	Kingdom of Sweden	...
12	UGA	106	Uganda	19881	Republic of Uganda	...
13	ZWE	134	Zimbabwe	9802	Republic of Zimbabwe	...

[148]:
```
size(june_fiscal)
```

[148]:
```
(13, 34)
```

Answer is 13 rows - 13 countries have a fiscal year which ends on June 30 - what a roller coaster!!!!! Patience and clear headedness are needed

1.8 5.

You can use the quantmod (http://www.quantmod.com/) package to get historical stock prices for publicly traded companies on the NASDAQ and NYSE. Use the following code to download data on Amazon's stock price and get the times the data was sampled.

[ ]:

[ ]:

[ ]:

[ ]:

[ ]: