Strings: Extra Practice (Part 3)

You can download this .qmd file from here. Just hit the Download Raw File button.

```
library(tidyverse)
library(rvest)
library(httr)
```

On Your Own - Extra practice with strings and regular expressions

1. Describe the equivalents of ?, +, * in {m,n} form.

- The equivalent of ? is {,1}
- The equivalent of + is {1,}
- The equivalent of * is {0,}

2. Describe, in words, what the expression "(.)(.)\2\1" will match, and provide a word or expression as an example.

- This will match any words or expressions where two letters are repeated but in opposite order.
- Example: abba

3. Produce an R string which the regular expression represented by "\..\..\.." matches. In other words, find a string `y` below that produces a TRUE in `str_detect`.

```
y <-(".a.b.c")

str_detect(y, "\\..\\..\\..")
```

```
[1] TRUE
```

4. Solve with `str_subset()`, using the words from `stringr::words`:

- Find all words that start or end with x.

```
str_subset(words, "^x|x$")
```

```
[1] "box" "sex" "six" "tax"
```

- Find all words that start with a vowel and end with a consonant.

```
str_subset(words, "^[aeiou](.*)([^aeiou])$")
```

```
  [1] "about"       "accept"      "account"     "across"      "act"
  [6] "actual"      "add"         "address"     "admit"       "affect"
 [11] "afford"      "after"       "afternoon"   "again"       "against"
 [16] "agent"       "air"         "all"         "allow"       "almost"
 [21] "along"       "already"     "alright"     "although"    "always"
 [26] "amount"      "and"         "another"     "answer"      "any"
 [31] "apart"       "apparent"    "appear"      "apply"       "appoint"
 [36] "approach"    "arm"         "around"      "art"         "as"
 [41] "ask"         "at"          "attend"      "authority"   "away"
 [46] "awful"       "each"        "early"       "east"        "easy"
 [51] "eat"         "economy"     "effect"      "egg"         "eight"
 [56] "either"      "elect"       "electric"    "eleven"      "employ"
 [61] "end"         "english"     "enjoy"       "enough"      "enter"
 [66] "environment" "equal"       "especial"    "even"        "evening"
 [71] "ever"        "every"       "exact"       "except"      "exist"
 [76] "expect"      "explain"     "express"     "identify"    "if"
 [81] "important"   "in"          "indeed"      "individual"  "industry"
 [86] "inform"      "instead"     "interest"    "invest"      "it"
 [91] "item"        "obvious"     "occasion"    "odd"         "of"
 [96] "off"         "offer"       "often"       "okay"        "old"
[101] "on"          "only"        "open"        "opportunity" "or"
[106] "order"       "original"    "other"       "ought"       "out"
[111] "over"        "own"         "under"       "understand"  "union"
[116] "unit"        "university"  "unless"      "until"       "up"
[121] "upon"        "usual"
```

- Find all words that start and end with the same letter

```
str_subset(words, "^(.)(.*)(\\1)$")
```

```
 [1] "america"    "area"       "dad"        "dead"       "depend"
```

```
 [6] "educate"    "else"       "encourage"  "engine"     "europe"
[11] "evidence"   "example"    "excuse"     "exercise"   "expense"
[16] "experience" "eye"        "health"     "high"       "knock"
[21] "level"      "local"      "nation"     "non"        "rather"
[26] "refer"      "remember"   "serious"    "stairs"     "test"
[31] "tonight"    "transport"  "treat"      "trust"      "window"
[36] "yesterday"
```

5. What words in `stringr::words` have the highest number of vowels? What words have the highest proportion of vowels? (Hint: what is the denominator?) Figure this out using the tidyverse and piping, starting with `as_tibble(words) |>`.

```
as_tibble(words) |>
  mutate(n_vowels = str_count(words, "[aeiou]")) |>
  arrange(desc(n_vowels))
```

```
# A tibble: 980 x 2
   value       n_vowels
   <chr>          <int>
 1 appropriate        5
 2 associate          5
 3 available          5
 4 colleague          5
 5 encourage          5
 6 experience         5
 7 individual         5
 8 television         5
 9 absolute           4
10 achieve            4
# i 970 more rows
```

- The words wiht the highest number of vowels are “appropriate”, “associate”, “available”, “colleague”, “encourage”, “experience”, “individual”, and “television”, all at 5 vowels.

```
as_tibble(words) |>
  mutate(n_vowels = str_count(words, "[aeiou]"),
         n_letters = str_count(words,"."),
         prop_vowels = n_vowels/n_letters) |>
  arrange(desc(prop_vowels))
```

```
# A tibble: 980 x 4
```

```
   value  n_vowels n_letters prop_vowels
   <chr>     <int>     <int>       <dbl>
 1 a             1         1       1
 2 area          3         4       0.75
 3 idea          3         4       0.75
 4 age           2         3       0.667
 5 ago           2         3       0.667
 6 air           2         3       0.667
 7 die           2         3       0.667
 8 due           2         3       0.667
 9 eat           2         3       0.667
10 europe        4         6       0.667
# i 970 more rows
```

- The words with the highest proportion of vowels are "a" with 100% and "area" and "idea" with 75%.

6. From the Harvard sentences data, use `str_extract` to produce a tibble with 3 columns: the sentence, the first word in the sentence, and the first word ending in "ed" (NA if there isn't one).

```
as_tibble(sentences) |>
  mutate(first_word = str_extract(sentences, "\\b[^ ]+\\b"),
         ed_ending = str_extract(sentences, "\\b[^ ]+ed\\b"))
```

```
# A tibble: 720 x 3
   value                                          first_word ed_ending
   <chr>                                          <chr>      <chr>
 1 The birch canoe slid on the smooth planks.     The        <NA>
 2 Glue the sheet to the dark blue background.    Glue       <NA>
 3 It's easy to tell the depth of a well.         It's       <NA>
 4 These days a chicken leg is a rare dish.       These      <NA>
 5 Rice is often served in round bowls.           Rice       served
 6 The juice of lemons makes fine punch.          The        <NA>
 7 The box was thrown beside the parked truck.    The        parked
 8 The hogs were fed chopped corn and garbage.    The        fed
 9 Four hours of steady work faced us.            Four       faced
10 A large size in stockings is hard to sell.     A          <NA>
# i 710 more rows
```

7. Find and output all contractions (words with apostrophes) in the Harvard sentences, assuming no sentence has multiple contractions.

```
str_subset(sentences, "\\b[^ ]+\\'[a-z]\\b")
```

```
 [1] "It's easy to tell the depth of a well."
 [2] "The soft cushion broke the man's fall."
 [3] "Open the crate but don't break the glass."
 [4] "Add the store's account to the last cent."
 [5] "The beam dropped down on the workman's head."
 [6] "Let's all join as we sing the last chorus."
 [7] "The copper bowl shone in the sun's rays."
 [8] "A child's wit saved the day for us."
 [9] "A ripe plum is fit for a king's palate."
[10] "It's a dense crowd in two distinct ways."
[11] "We don't get much money but we have fun."
[12] "Ripe pears are fit for a queen's table."
[13] "Cheap clothes are flashy but don't last."
[14] "The facts don't always show who is right."
[15] "Pack the kits and don't forget the salt."
[16] "We don't like to admit our small faults."
[17] "Dig deep in the earth for pirate's gold."
[18] "She saw a cat in the neighbor's house."
```

- (there are a few instances here where the word is not necessarily a contraction, but has "'s" because it is a possessive.)

8. *Carefully* explain what the code below does, both line by line and in general terms.

```
str_replace_all(words, "^([A-Za-z])(.*)([a-z])$", "\\3\\2\\1")
```

```
 [1] "a"          "ebla"       "tboua"      "ebsoluta"   "tccepa"
 [6] "tccouna"    "echieva"    "scrosa"     "tca"        "ectiva"
[11] "lctuaa"     "dda"        "sddresa"    "tdmia"      "edvertisa"
[16] "tffeca"     "dffora"     "rftea"      "nfternooa"  "ngaia"
[21] "tgainsa"    "ega"        "tgena"      "oga"        "egrea"
[26] "ria"        "lla"        "wlloa"      "tlmosa"     "glona"
[31] "ylreada"    "tlrigha"    "olsa"       "hlthouga"   "slwaya"
[36] "america"    "tmouna"     "dna"        "rnothea"    "rnswea"
[41] "yna"        "tpara"      "tpparena"   "rppeaa"     "yppla"
[46] "tppoina"    "hpproaca"   "eppropriata" "area"      "ergua"
[51] "mra"        "drouna"     "erranga"    "tra"        "sa"
[56] "ksa"        "essociata"  "essuma"     "ta"         "dttena"
[61] "yuthorita"  "evailabla"  "ewara"      "ywaa"       "lwfua"
```

```
 [66] "yabb"        "kacb"        "dab"         "gab"         "ealancb"
 [71] "lalb"        "kanb"        "rab"         "easb"        "sasib"
 [76] "eb"          "reab"        "teab"        "yeautb"      "eecausb"
 [81] "eecomb"      "deb"         "eeforb"      "negib"       "dehinb"
 [86] "eelievb"     "tenefib"     "tesb"        "teb"         "netweeb"
 [91] "gib"         "lilb"        "hirtb"       "tib"         "klacb"
 [96] "elokb"       "dloob"       "wlob"        "elub"        "doarb"
[101] "toab"        "yodb"        "koob"        "hotb"        "rotheb"
[106] "eottlb"      "mottob"      "xob"         "yob"         "kreab"
[111] "frieb"       "trillianb"   "grinb"       "nritaib"     "rrotheb"
[116] "tudgeb"      "duilb"       "sub"         "susinesb"    "yusb"
[121] "tub"         "yub"         "yb"          "eakc"        "lalc"
[126] "nac"         "rac"         "darc"        "earc"        "yarrc"
[131] "easc"        "tac"         "hatcc"       "eausc"       "tenc"
[136] "eentrc"      "nertaic"     "rhaic"       "nhairmac"    "ehancc"
[141] "ehangc"      "phac"        "rharactec"   "ehargc"      "pheac"
[146] "khecc"       "dhilc"       "ehoicc"      "ehoosc"      "thrisC"
[151] "shristmaC"   "hhurcc"      "yitc"        "mlaic"       "slasc"
[156] "nleac"       "rleac"       "tlienc"      "klocc"       "elosc"
[161] "slosec"      "elothc"      "bluc"        "eoffec"      "dolc"
[166] "eolleaguc"   "tollecc"     "eollegc"     "rolouc"      "eomc"
[171] "tommenc"     "tommic"      "eommittec"   "nommoc"      "yommunitc"
[176] "yompanc"     "eomparc"     "eompletc"    "eomputc"     "noncerc"
[181] "nonditioc"   "ronfec"      "ronsidec"    "tonsulc"     "tontacc"
[186] "eontinuc"    "tontracc"    "lontroc"     "eonversc"    "kooc"
[191] "yopc"        "rornec"      "torrecc"     "tosc"        "doulc"
[196] "louncic"     "tounc"       "yountrc"     "yountc"      "eouplc"
[201] "eoursc"      "tourc"       "rovec"       "ereatc"      "srosc"
[206] "puc"         "turrenc"     "tuc"         "dad"         "ranged"
[211] "eatd"        "yad"         "dead"        "lead"        "read"
[216] "eebatd"      "eecidd"      "necisiod"    "peed"        "eefinitd"
[221] "eegred"      "tepartmend"  "depend"      "eescribd"    "nesigd"
[226] "letaid"      "pevelod"     "eid"         "eifferencd"  "tifficuld"
[231] "rinned"      "tirecd"      "siscusd"     "tistricd"    "eividd"
[236] "od"          "roctod"      "tocumend"    "god"         "rood"
[241] "eoubld"      "toubd"       "nowd"        "wrad"        "sresd"
[246] "krind"       "erivd"       "prod"        "yrd"         "eud"
[251] "gurind"      "hace"        "yarle"       "tase"        "yase"
[256] "tae"         "yconome"     "educate"     "tffece"      "gge"
[261] "tighe"       "rithee"      "tlece"       "clectrie"    "nlevee"
[266] "else"        "ymploe"      "encourage"   "dne"         "engine"
[271] "hnglise"     "ynjoe"       "hnouge"      "rntee"       "tnvironmene"
[276] "lquae"       "lspeciae"    "europe"      "nvee"        "gvenine"
```

```
[281] "rvee"        "yvere"       "evidence"    "txace"       "example"
[286] "txcepe"      "excuse"      "exercise"    "txise"       "txpece"
[291] "expense"     "experience"  "nxplaie"     "sxprese"     "axtre"
[296] "eye"         "eacf"        "tacf"        "raif"        "lalf"
[301] "yamilf"      "raf"         "marf"        "tasf"        "rathef"
[306] "ravouf"      "deef"        "leef"        "wef"         "dielf"
[311] "tighf"       "eigurf"      "eilf"        "lilf"        "milf"
[316] "linaf"       "einancf"     "dinf"        "einf"        "hinisf"
[321] "eirf"        "tirsf"       "hisf"        "tif"         "eivf"
[326] "tlaf"        "rloof"       "ylf"         "wollof"      "doof"
[331] "toof"        "rof"         "eorcf"       "torgef"      "morf"
[336] "eortunf"     "dorwarf"     "rouf"        "erancf"      "eref"
[341] "yridaf"      "drienf"      "mrof"        "tronf"       "lulf"
[346] "nuf"         "nunctiof"    "dunf"        "rurthef"     "euturf"
[351] "eamg"        "nardeg"      "sag"         "lenerag"     "yermang"
[356] "teg"         "lirg"        "eivg"        "slasg"       "og"
[361] "dog"         "doog"        "eoodbyg"     "noverg"      "drang"
[366] "trang"       "treag"       "nreeg"       "droung"      "proug"
[371] "wrog"        "suesg"       "yug"         "raih"        "falh"
[376] "lalh"        "danh"        "ganh"        "nappeh"      "yapph"
[381] "darh"        "eath"        "eavh"        "eh"          "deah"
[386] "health"      "reah"        "tearh"       "teah"        "yeavh"
[391] "lelh"        "pelh"        "eerh"        "high"        "yistorh"
[396] "tih"         "dolh"        "yolidah"     "eomh"        "tonesh"
[401] "eoph"        "eorsh"       "lospitah"    "toh"         "rouh"
[406] "eoush"       "woh"         "roweveh"     "oullh"       "dundreh"
[411] "dusbanh"     "adei"        "ydentifi"    "fi"          "emagini"
[416] "tmportani"   "emprovi"     "ni"          "encludi"     "encomi"
[421] "encreasi"    "dndeei"      "lndividuai"  "yndustri"    "mnfori"
[426] "ensidi"      "dnsteai"     "ensuri"      "tnteresi"    "onti"
[431] "entroduci"   "tnvesi"      "envolvi"     "essui"       "ti"
[436] "mtei"        "sesuj"       "boj"         "noij"        "eudgj"
[441] "pumj"        "tusj"        "peek"        "yek"         "dik"
[446] "lilk"        "dink"        "gink"        "nitchek"     "knock"
[451] "wnok"        "raboul"      "dal"         "yadl"        "danl"
[456] "eanguagl"    "eargl"       "tasl"        "eatl"        "haugl"
[461] "wal"         "yal"         "deal"        "nearl"       "eeavl"
[466] "tefl"        "gel"         "sesl"        "tel"         "rettel"
[471] "level"       "eil"         "eifl"        "tighl"       "eikl"
[476] "yikell"      "timil"       "einl"        "kinl"        "tisl"
[481] "nistel"      "eittll"      "eivl"        "doal"        "local"
[486] "kocl"        "nondol"      "gonl"        "kool"        "dorl"
[491] "eosl"        "tol"         "eovl"        "wol"         "kucl"
```

```
[496] "huncl"       "eachinm"     "naim"        "rajom"       "eakm"
[501] "nam"         "eanagm"      "yanm"        "karm"        "tarkem"
[506] "yarrm"       "hatcm"       "rattem"      "yam"         "eaybm"
[511] "neam"        "geaninm"     "eeasurm"     "teem"        "rembem"
[516] "nentiom"     "eiddlm"      "tighm"       "eilm"        "kilm"
[521] "nilliom"     "dinm"        "rinistem"    "sinum"       "einutm"
[526] "sism"        "ristem"      "tomenm"      "yondam"      "yonem"
[531] "hontm"       "eorm"        "gorninm"     "tosm"        "rothem"
[536] "notiom"      "eovm"        "srm"         "hucm"        "cusim"
[541] "tusm"        "eamn"        "nation"      "eaturn"      "rean"
[546] "yecessarn"   "deen"        "reven"       "wen"         "sewn"
[551] "texn"        "eicn"        "tighn"       "einn"        "on"
[556] "non"         "eonn"        "lorman"      "hortn"       "ton"
[561] "eotn"        "eoticn"      "won"         "rumben"      "sbviouo"
[566] "nccasioo"    "ddo"         "fo"          "ffo"         "rffeo"
[571] "effico"      "nfteo"       "ykao"        "dlo"         "no"
[576] "enco"        "eno"         "ynlo"        "npeo"        "eperato"
[581] "ypportunito" "epposo"      "ro"          "rrdeo"       "erganizo"
[586] "lriginao"    "rtheo"       "etherwiso"   "tugho"       "tuo"
[591] "rveo"        "nwo"         "kacp"        "eagp"        "tainp"
[596] "raip"        "rapep"       "haragrapp"   "nardop"      "tarenp"
[601] "karp"        "tarp"        "rarticulap"  "yartp"       "sasp"
[606] "tasp"        "yap"         "eencp"       "nensiop"     "eeoplp"
[611] "rep"         "tercenp"     "terfecp"     "serhapp"     "deriop"
[616] "nersop"      "hhotograpp"  "kicp"        "eicturp"     "eiecp"
[621] "elacp"       "nlap"        "ylap"        "eleasp"      "slup"
[626] "toinp"       "eolicp"      "yolicp"      "colitip"     "roop"
[631] "nositiop"    "eositivp"    "eossiblp"    "tosp"        "dounp"
[636] "rowep"       "eractisp"    "ereparp"     "tresenp"     "sresp"
[641] "eressurp"    "eresump"     "yrettp"      "srevioup"    "ericp"
[646] "trinp"       "erivatp"     "erobablp"    "mroblep"     "droceep"
[651] "srocesp"     "eroducp"     "troducp"     "erogrammp"   "trojecp"
[656] "rropep"      "eroposp"     "trotecp"     "erovidp"     "cublip"
[661] "lulp"        "eurposp"     "husp"        "tup"         "yualitq"
[666] "ruarteq"     "nuestioq"    "kuicq"       "duiq"        "tuieq"
[671] "euitq"       "oadir"       "lair"        "eaisr"       "eangr"
[676] "eatr"        "rather"      "dear"        "yeadr"       "lear"
[681] "eealisr"     "yeallr"      "neasor"      "eeceivr"     "tecenr"
[686] "neckor"      "eecognizr"   "decommenr"   "decorr"      "der"
[691] "eeducr"      "refer"       "degarr"      "negior"      "nelatior"
[696] "remember"    "teporr"      "tepresenr"   "eequirr"     "hesearcr"
[701] "eesourcr"    "tespecr"     "eesponsiblr" "tesr"        "tesulr"
[706] "neturr"      "dir"         "tighr"       "ginr"        "eisr"
```

```
[711] "doar"         "eolr"         "lolr"         "moor"         "dounr"
[716] "eulr"         "nur"          "eafs"         "eals"         "eams"
[721] "yaturdas"     "eavs"         "yas"          "echems"       "lchoos"
[726] "eciencs"      "ecors"        "dcotlans"     "teas"         "decons"
[731] "yecretars"    "nectios"      "eecurs"       "ees"          "mees"
[736] "fels"         "lels"         "dens"         "eenss"        "eeparats"
[741] "serious"      "eervs"        "eervics"      "tes"          "eettls"
[746] "neves"        "xes"          "lhals"        "ehars"        "ehs"
[751] "thees"        "ehos"         "thoos"        "phos"         "thors"
[756] "dhouls"       "whos"         "thus"         "kics"         "eids"
[761] "nigs"         "rimilas"      "eimpls"       "eincs"        "gins"
[766] "eingls"       "ris"          "ristes"       "tis"          "eits"
[771] "eituats"      "xis"          "eizs"         "plees"        "tlighs"
[776] "wlos"         "lmals"        "emoks"        "os"           "locias"
[781] "yociets"      "eoms"         "nos"          "noos"         "yorrs"
[786] "tors"         "douns"        "houts"        "epacs"        "kpeas"
[791] "lpecias"      "cpecifis"     "dpees"        "lpels"        "dpens"
[796] "equars"       "ftafs"        "etags"        "stairs"       "dtans"
[801] "dtandars"     "ttars"        "etats"        "ntatios"      "ytas"
[806] "ptes"         "ktics"        "ltils"        "ptos"         "ytors"
[811] "ttraighs"     "ytrategs"     "ttrees"       "etriks"       "gtrons"
[816] "etructurs"    "ttudens"      "ytuds"        "ftufs"        "dtupis"
[821] "tubjecs"      "duccees"      "hucs"         "nuddes"       "tuggess"
[826] "tuis"         "rummes"       "nus"          "yundas"       "yuppls"
[831] "tuppors"      "eupposs"      "eurs"         "eurpriss"     "hwitcs"
[836] "mystes"       "eablt"        "eakt"         "kalt"         "eapt"
[841] "xat"          "aet"          "heact"        "meat"         "eelephont"
[846] "nelevisiot"   "lelt"         "net"          "dent"         "mert"
[851] "eerriblt"     "test"         "nhat"         "khant"        "eht"
[856] "nhet"         "ehert"        "eherefort"    "yhet"         "ghint"
[861] "khint"        "nhirteet"     "yhirtt"       "shit"         "uhot"
[866] "hhougt"       "dhousant"     "ehret"        "hhrougt"      "whrot"
[871] "yhursdat"     "eit"          "eimt"         "ot"           "yodat"
[876] "rogethet"     "womorrot"     "tonight"      "oot"          "pot"
[881] "lotat"        "houct"        "dowart"       "nowt"         "eradt"
[886] "craffit"      "nrait"        "transport"    "lravet"       "treat"
[891] "eret"         "eroublt"      "erut"         "trust"        "yrt"
[896] "yuesdat"      "nurt"         "ewelvt"       "ywentt"       "owt"
[901] "eypt"         "rndeu"        "dnderstanu"   "nniou"        "tniu"
[906] "enitu"        "yniversitu"   "snlesu"       "lntiu"        "pu"
[911] "npou"         "esu"          "lsuau"        "ealuv"        "sariouv"
[916] "yerv"         "oidev"        "wiev"         "eillagv"      "tisiv"
[921] "eotv"         "eagw"         "taiw"         "kalw"         "lalw"
```

```
[926] "tanw"        "raw"         "marw"        "hasw"        "eastw"
[931] "hatcw"       "ratew"       "yaw"         "ew"          "reaw"
[936] "yednesdaw"   "eew"         "keew"        "heigw"       "eelcomw"
[941] "lelw"        "tesw"        "thaw"        "nhew"        "eherw"
[946] "rhethew"     "hhicw"       "ehilw"       "ehitw"       "ohw"
[951] "eholw"       "yhw"         "eidw"        "eifw"        "lilw"
[956] "niw"         "dinw"        "window"      "hisw"        "hitw"
[961] "nithiw"      "tithouw"     "nomaw"       "rondew"      "doow"
[966] "dorw"        "korw"        "dorlw"       "yorrw"       "eorsw"
[971] "hortw"       "doulw"       "eritw"       "gronw"       "reay"
[976] "sey"         "yesterday"   "tey"         "uoy"         "gouny"
```

```
temp <- str_replace_all(words, "^([A-Za-z])(.*)([a-z])$", "\\3\\2\\1")
as_tibble(words) |>
  semi_join(as_tibble(temp)) |>
  print(n = Inf)
```

```
Joining with `by = join_by(value)`
```

```
# A tibble: 45 x 1
   value
   <chr>
 1 a
 2 america
 3 area
 4 dad
 5 dead
 6 deal
 7 dear
 8 depend
 9 dog
10 educate
11 else
12 encourage
13 engine
14 europe
15 evidence
16 example
17 excuse
18 exercise
19 expense
```

```
20 experience
21 eye
22 god
23 health
24 high
25 knock
26 lead
27 level
28 local
29 nation
30 no
31 non
32 on
33 rather
34 read
35 refer
36 remember
37 serious
38 stairs
39 test
40 tonight
41 transport
42 treat
43 trust
44 window
45 yesterday
```

- Line one creates a new string, 'temp', which uses the string "words" and swaps the last and first letter.
  - In general, string_replace_all allows you to identify a certain part of a string and replace it with something else. In general, backtracing allows you to chunk elements together through parenthese and repeat one or more of those chunks in a certain order.
- Line two turns a string into a tibble. In this particular, instance, it is turning the string "words" into a tibble.
- Line three turns the temp string we made into a tibble and semi-joins the temp tibble to the words tibble. A semi-join here will return all of the rows in "words" that have a match in "temp"
- Line 4 prints all of the rows in the tibble.

Coco and Rotten Tomatoes

We will check out the Rotten Tomatoes page for the 2017 movie Coco, scrape information from that page (we'll get into web scraping in a few weeks!), clean it up into a usable format, and answer some questions using strings and regular expressions.

```
# used to work
# coco <- read_html("https://www.rottentomatoes.com/m/coco_2017")

robotstxt::paths_allowed("https://www.rottentomatoes.com/m/coco_2017")
```

```
www.rottentomatoes.com
```

```
[1] TRUE
```

```
library(polite)
coco <- "https://www.rottentomatoes.com/m/coco_2017" |>
  bow() |>
  scrape()

top_reviews <-
  "https://www.rottentomatoes.com/m/coco_2017/reviews?type=top_critics" |>
  bow() |>
  scrape()
top_reviews <- html_nodes(top_reviews, ".review-text")
top_reviews <- html_text(top_reviews)

user_reviews <-
  "https://www.rottentomatoes.com/m/coco_2017/reviews?type=user" |>
  bow() |>
  scrape()
user_reviews <- html_nodes(user_reviews, ".js-review-text")
user_reviews <- html_text(user_reviews)
```

9. `top_reviews` is a character vector containing the 20 most recent critic reviews (along with some other junk) for Coco, while `user_reviews` is a character vector with the 10 most recent user reviews.

a) Explain how the code below helps clean up both `user_reviews` and `top_reviews` before we start using them.

```
user_reviews <- str_trim(user_reviews)
top_reviews <- str_trim(top_reviews)
```

- str_trim removes the whitespace from the start and the end of the string and also turns any multiple consecutive instances of spaces within a text into a singular space. Using this code for the user reviews and the top reviews cleans them up by enforing normal spacing which will make it easier to find patterns and se regular expressions.

b) Print out the critic reviews where the reviewer mentions "emotion" or "cry". Think about various forms ("cried", "emotional", etc.) You may want to turn reviews to all lower case before searching for matches.

```
str_subset(str_to_lower(top_reviews), "emotion|cry|cried|sad|emotional|tears")
```

```
[1] "a wonderful return to form for pixar, who again deliver the emotional and creative goods
[2] "at worst it suggests that the brains trust at pixar, after 22 years of peerless output
[3] "funny, irreverent and eye-popping. it will also make you want to cry at least once but p
```

c) In critic reviews, replace all instances where "Pixar" is used with its full name: "Pixar Animation Studios".

```
str_replace(top_reviews, "Pixar", "Pixar Animation Studios")
```

```
 [1] "A fine addition to the Pixar Animation Studios legacy… a very sweet film about family,
 [2] "An unexpectedly brilliant and dynamic story about lineage, connection, and self-discove
 [3] "In a country with an ever increasing Hispanic and Mexican population, a film like Coco
 [4] "I don't think there's any question that Coco is really great."
 [5] "Several times I found myself sobbing without knowing exactly why only to realize why th
 [6] "A wonderful return to form for Pixar Animation Studios, who again deliver the emotional
 [7] "The film has a galloping rhythm, and the animation is scrupulous and ravishing, from it
 [8] "On paper, the mythology scans as complicated and dark, but in the capable hands of Acad
 [9] "Pixar Animation Studios's latest project is a glittering return to non-franchise form a
[10] "Its victorious denouement offers everyone a different way to think about what it means
[11] "At worst it suggests that the brains trust at Pixar Animation Studios, after 22 years o
[12] "Funny, irreverent and eye-popping. It will also make you want to cry at least once but
[13] "This is a charming and very memorable film."
[14] "Despite the fact that it's so well told and really beautifully directed, it doesn't hav
[15] "... Coco is another triumph for Pixar Animation Studios..."
[16] "Funny and heart-tugging with some knockout tunes, the movie glows with warmth. And how
[17] "Not top-tier Pixar Animation Studios. But decent enough."
[18] "Pixar Animation Studios has raised the animation bar again, with its most musical - and
```

```
[19] "While the animation is Pixar Animation Studios perfect, I don't think the story grips t
[20] "Every plot point and thematic implication slots into place, but the pleasures of Coco a
```

d) Find out how many times each user uses "I" in their review. Remember that it could be used as upper or lower case, at the beginning, middle, or end of a sentence, etc.

```
str_count(str_to_lower(user_reviews), "\\bi \\b")
```

```
 [1] 0 0 0 3 0 0 3 1 4 1 0 0 0 1 0 0 1 0 0 0
```

e) Do critics or users have more complex reviews, as measured by average number of commas used? Be sure your code weeds out commas used in numbers, such as "12,345".

```
str_count(str_to_lower(user_reviews), "\\b, ") |>
  mean()
```

```
[1] 2.15
```

```
str_count(str_to_lower(top_reviews), "\\b, ") |>
  mean()
```

```
[1] 1.3
```

- As measured by average number of commas used, users have more complex reviews with a mean of 2.15 commas used compared to a mean of 1.3 commas used for critics.