**14.2.5**

1. The paste function combines two sections of text into one string, separated by a space. The paste0 function combines two sections of text into one string, separated by nothing.
2. In the sep argument you are adding in a unit to be inserted in between two strings. The collapse argument on the other hand takes a formed character vector and then forces the strings around a fixed unit, like a “,”
3. Clearly it’s impossible to extract the middle character of an even number of characters. You would just have to randomly select one of the two central variables.
4. str\_wrap: implements the Knuth-Plass paragraph wrapping algorithm. It wraps the text so that it all fits within certain dimensions. It’s handy with a massive string of text like extracted HTML of a webpage.
5. str\_trim(): gets rid of whitespace from start and end of a string. For example, str\_trim(“ abc “) will return “abc” rather than “ abc “. The opposite of this is str\_pad which will add whitespace on the side = “left”, “right”, or “both”
6. str\_c(“a”, “b”, “c”, sep = “,”, last = “, and”)

**14.3.1.1**

1. Explain why the following strings don’t match a \:
   1. “\”: this will just escape to the next character in the string. Like a “.” or any letter
   2. “\\”: this distills to just a backslash in regexps, which then escapes to the next character
   3. “\\\”: the only expression that matches \ is the “\\\\”. The triple backslash distills to a backslash in the regexps. Then it ends up escaping an escaped character so it doesn’t exactly match what the quadruple backslash does.
2. “”’\\\\”
3. This matches patterns that are characters followed by dots in a triple repeating sequence. As a string, it could be, “a.b.c.”

**14.3.2.1**

1. “\\$\\^\\$”
2. Create regular expressions that find all the words with:
   1. Starts with “y” 🡪 str\_view(words, “^y”, match = T)
   2. Ends with “x” 🡪 str\_view(words, “x$”, match = T)
   3. Exactly three letters long 🡪 str\_view(words, “^…$”, match = T)
   4. Has seven letters or more 🡪 str\_view(words, “^…….$”, match = T)

**14.3.3.1**

1. Create regular expressions to find all words that:
   1. str\_view(words, “^[aeiouy]”, match = T)
   2. str\_view(words, “^[^aeiouy]\*$”, match = T)
   3. str\_view(words, “[^e]ed$”, match = T)
   4. str\_view(words, “i(ng|se)$”, match = T)
2. str\_view(words, “([^c])|ei”, match = T)
3. str\_view(words, “^q”, match = T) 🡪 this shows all words starting with q and they all have u as the next character
4. str\_view(words, “our$”, match = T) 🡪 this will show words spelled with a u that don’t have them in American English, like colour, favour and labour
5. str\_view(“\\d\\d\\d(-|)\\d\\d\\d(-|)\\d\\d\\d\\d) --> ddd-ddd-dddd

**14.3.4.1**

1. ? 🡪 {1}, + 🡪 {1, }, \* 🡪 {0}
2. Describe in words what these regular expressions match:
   1. ^.\*$ 🡪 literally anything
   2. “\\{.+\\}” 🡪 1 or more characters
   3. dddd-dd-dd
   4. \\\\
3. Create regular expressions to find all words that:
   1. str\_view(words, “^[^aeiouy]{3}”, match = T)
   2. str\_view(words, “[aeiouy]{3, }”, match = T)
   3. str\_view(words, “([aeiouy][^aeiouy]){2,}”, match = T)
4. Crossword Puzzle

**14.4.2**

1. Solve it by using a single regexps and a combination of multiple calls
   1. All words that start or end with x
      1. str\_view(words, “^x|x$”, match = T)
      2. start = str\_detect(words, “^x”)  
         finish = str\_detect(words, “x$”)  
         words[start | finish]
   2. Find all words that start with a vowel and end with a consonant
      1. str\_view(words, “^[aeiouy][^aeiouy]$”, match = T)
      2. vowel = str\_detect(words, “^[aeiouy]”)  
         consonant = str\_detect(words, “[^aeiouy]$”  
         words[vowel & consonant]
   3. Are there any words that contain at least one of each different vowel?
      1. No there are no words that contain at least one of each different vowel. You can create 6 different str\_detect variables matching each vowel and then create a long “&” statement.
2. Process: I counted all the vowels in each word and counted the length of each word, then created a tibble with words, counts and lengths.
   1. There are many words that share the crown of highest number of vowels at 5.
   2. Highest proportion of vowels is “a, eye, and you”. The denominator is total letters in the word

**14.4.6.1**

1. y = “apples, pears, and bananas”  
   str\_split(y, “, and |,”)
2. Boundary accounts for any spaces and punctuation within the string of text. While “ “ just separates the by whitespace and not necessarily text or number
3. Splitting by (“ ”) splits the string into individual characters so it will create its own section even for punctuation and whitespace which is probably unhelpful.

**15.4.1**

1. Using the summary function, it appears that there is a large NA factor which must mean a lot of people choose not to answer that question in the Social Survey.   
     
   gss\_cat %>%

filter(!is.na(tvhours)) %>%

ggplot(aes(x = tvhours)) +

geom\_histogram(binwidth = 1)

1. glimpse(gss\_cat). There are 9 variables in the datset: year, marital, age, race, rincome, partyid, relig, denom and tvhours.
   1. Year 🡪 principled, very logical ordering
   2. Martial 🡪 principled because the categories are ordered by count
   3. Age 🡪 principled, logical ordering from youngest to oldest
   4. Race 🡪 principled, they are ordered by count
   5. Rincome 🡪 there are a lot of arbitrary placements of non-numerical values.
   6. Relig 🡪 completely arbitrary because there’s no ordering
   7. Partyid 🡪 arbitrary ordering
   8. Denom 🡪 arbitrary, no logical ordering with “No answer”, “Other”, “Don’t know”
   9. Tvhours 🡪 principled
2. It turns “Not Applicable” to a 1 rendering it pointless in the graph

**15.5.1**

1. gss\_cat %>%

mutate(partyid =

fct\_collapse(partyid,

other = c("No answer", "Don't know", "Other party"),

rep = c("Strong republican", "Not str republican"),

ind = c("Ind,near rep", "Independent", "Ind,near dem"),

dem = c("Not str democrat", "Strong democrat"))) %>%

count(year, partyid) %>%

group\_by(year)  
  
Based off the data, it appears that all traditional parties have lost support and more people have shifted towards an independent affiliation

1. I can’t seem to figure this problem out so it would be great if we can go over this in further detail on Thursday