R - Practice 03

Sangkon Han(sangkon@pusan.ac.kr)

2023-09-11

Contents

Data Wrangle: strings - factors (stringr & forcats)

Strings inside tidyverse

```
# Create a string
s1 <- "Double quotes string"
s2 <- 'Single quotes string'
s3 <- "Double quotes outside, 'single quotes inside' a string"
s4 <- 'Single quotes outside, "double quotes inside" a string'
# Not possible!
# s5 <- "Double quotes inside "double" quotes"
# s6 <- 'Single quotes inside 'single' quotes'
# s7 <-  "not "working"
# s8 <-  'not 'working'
# s9 <- "Missing closing quote
```

```
# Create a vector of strings
vec <- c("a", "b", "c")
# Character vector inside a tibble
df <- tibble(letters = vec)
```

```
# How to escape a character ~ Regular expressions

## Literal single or double quotes
```

```
"\""   # escape a special charatcer with back slash - \
```

```
## [1] "\""
```

```
'\''
```

```
## [1] "'"
```

```
## New line
"\n"
```

```
## [1] "\n"
```

```
## Tabulator
"\t"
```

```
## [1] "\t"
```

```
##   Unicode non-english characters
"\u03B1"
```

```
## [1] "α"
```

```
## See raw content of the string (omiited escape characters and outside quotes)
s <- "string"
writeLines(s)
```

```
## string
```

```
s <- "\""
writeLines(s)
```

```
## "
```

```
s <- "line 1 \nline 2"
writeLines(s)
```

```
## line 1
## line 2
```

Strings matching

```
# load strings dataset
load("./data/strings.RData")
```

```
# str_detect() - Detect s pattern
# similar base R: grepl()
# Find a fruit containing letter "a" (anywhere in word)
fruit %>% head()
```

```
## [1] "apple"       "apricot"     "avocado"     "banana"      "bell pepper"
## [6] "bilberry"
```

```
ind <- str_detect(string = fruit, pattern = "a") # returns TRUE / FALSE
fruit[ind] %>% head()
```

```
## [1] "apple"        "apricot"      "avocado"      "banana"       "blackberry"
## [6] "blackcurrant"
```

```
# grepl
fruit[grepl(pattern = "a", x = fruit)] %>% head()
```

```
## [1] "apple"        "apricot"      "avocado"      "banana"       "blackberry"
## [6] "blackcurrant"
```

```
# Find a fruit not containing any letter "a" !

# we use negation
fruit[str_detect(fruit, "a", negate = T)] %>% head()
```

```
## [1] "bell pepper"  "bilberry"     "blueberry"    "boysenberry"  "cherry"
## [6] "chili pepper"
```

```
fruit[!str_detect(fruit, "a")] %>% head()
```

```
## [1] "bell pepper"  "bilberry"     "blueberry"    "boysenberry"  "cherry"
## [6] "chili pepper"
```

```
# Inside tibble add flag if fruit contains letter "a" or if it doesn't contain letter "a"
fruit.df %>%
  mutate(flag = case_when(str_detect(fruit, pattern = "a") ~ "contains 'a'", T ~ "does not contain 'a'")
  head()
```

```
## # A tibble: 6 x 2
##   fruit       flag
##   <chr>       <chr>
## 1 apple       contains 'a'
## 2 apricot     contains 'a'
## 3 avocado     contains 'a'
## 4 banana      contains 'a'
## 5 bell pepper does not contain 'a'
## 6 bilberry    does not contain 'a'
```

```
# Find fruits starting or ending with letter "a"
ind.start.a <- str_detect(fruit, pattern = "^a")
fruit[ind.start.a] %>% head()
```

```
## [1] "apple"   "apricot" "avocado"
```

```
ind.end.a <- str_detect(fruit, pattern = "a$")
fruit[ind.end.a] %>% head()
```

```
## [1] "banana"    "cherimoya" "feijoa"    "guava"     "papaya"    "satsuma"
```

```
# str_which() - Detect s pattern return index position
# similar base R: grep()
#   Find a fruit containing letter "a" (anywhere in word)
ind <- str_which(string = fruit, pattern = "a") # returns index position
fruit[ind] %>% head()
```

```
## [1] "apple"        "apricot"      "avocado"      "banana"       "blackberry"
## [6] "blackcurrant"
```

```
fruit[grep(pattern = "a", x = fruit)] %>% head()
```

```
## [1] "apple"        "apricot"      "avocado"      "banana"       "blackberry"
## [6] "blackcurrant"
```

```
# str_count() - Count number of pattern matches in string

# Add count of letter "a" in each fruit (use table)
fruit.df1 <- fruit.df %>%
```

```
  mutate(`count a` = str_count(fruit, pattern = "a"))

# Show counts of letter "a" in fruits
fruit.df1 %>%
  count(`count a`)
```

```
## # A tibble: 4 x 2
##   `count a`     n
##       <int> <int>
## 1         0    30
## 2         1    37
## 3         2    11
## 4         3     2
```

```
# Show fruit with 3 "a" letters
fruit.df1 %>%
  filter(`count a` == 3) %>%
  head()
```

```
## # A tibble: 2 x 2
##   fruit  `count a`
##   <chr>      <int>
## 1 banana         3
## 2 papaya         3
```

```
# str_locate() / str_locate_all() - Locate position(s) of pattern match in string

# Locate position of first letter "a" in each fruit (matrix is returned)
str_locate(fruit, pattern = "a") %>% head()
```

```
##      start end
## [1,]     1   1
## [2,]     1   1
## [3,]     1   1
## [4,]     2   2
## [5,]    NA  NA
## [6,]    NA  NA
```

```
fruit.df1 <- str_locate(fruit, pattern = "a") %>%
  as_tibble() %>% # convert matrix of positions to tibble
  mutate(fruit = fruit) %>% # add fruit name column
  select(fruit, start, end) # re-arrange columns

# Locate position of all letters "a" in each fruit (list is returned)
str_locate_all(fruit, pattern = "a") %>% head()
```

```
## [[1]]
##      start end
## [1,]     1   1
##
## [[2]]
##      start end
## [1,]     1   1
##
## [[3]]
##      start end
```

```
## [1,]     1   1
## [2,]     5   5
##
## [[4]]
##      start end
## [1,]     2   2
## [2,]     4   4
## [3,]     6   6
##
## [[5]]
##      start end
##
## [[6]]
##      start end
```

Strings subsetting

```
# str_sub() - Extract part of s string
# similar base R: substr()

# Extract first 3 letters of a fruit
str_sub(string = fruit, start = 1, end = 3) %>% head()
```

```
## [1] "app" "apr" "avo" "ban" "bel" "bil"
```

```
substr(x = fruit, start = 1, stop = 3) %>% head()
```

```
## [1] "app" "apr" "avo" "ban" "bel" "bil"
```

```
# Extract first letter of common word and count word frequency by first word letter
words.df %>%
  mutate(`first letter` = str_sub(word, 1, 1)) %>% # extract first letter
  count(`first letter`) %>% # count frequencies
  arrange(desc(n)) %>% # sort from high to low frequency
  head()
```

```
## # A tibble: 6 x 2
##   `first letter`     n
##   <chr>          <int>
## 1 s                334
## 2 c                297
## 3 p                246
## 4 a                214
## 5 t                174
## 6 e                164
```

```
# Extract middle part of the word
str_sub(fruit, start = 3, end = 5) %>% head() # from 3rd to 5th letter
```

```
## [1] "ple" "ric" "oca" "nan" "ll " "lbe"
```

```
# Extract last letter / last 3 letters (use negative counters - for counting backward)
str_sub(fruit, start = -1, end = -1) %>% head() # last letter
```

```
## [1] "e" "t" "o" "a" "r" "y"
```

```r
str_sub(fruit, start = -3, end = -1) %>% head() # last 3 letters
```

```
## [1] "ple" "cot" "ado" "ana" "per" "rry"
```

```r
# str_subset() - Return only strings that match pattern
# Return fruit containing letter "c"
str_subset(string = fruit, pattern = "c") %>% head()
```

```
## [1] "apricot"      "avocado"      "blackberry"   "blackcurrant" "canary melon"
## [6] "cantaloupe"
```

```r
# Return fruit starting with letter "c"
str_subset(string = fruit, pattern = "^c") %>% head()
```

```
## [1] "canary melon" "cantaloupe"   "cherimoya"    "cherry"       "chili pepper"
## [6] "clementine"
```

```r
# str_extract() / str_extract_all() - Return first or every pattern match
# Return fruit containing "a" first occurence
str_extract(string = fruit, pattern = "a") %>% head()  # vector is returned
```

```
## [1] "a" "a" "a" "a" NA  NA
```

```r
# Return fruit containing "a" all occurences
str_extract_all(string = fruit, pattern = "a") %>% head() #  list is returned
```

```
## [[1]]
## [1] "a"
##
## [[2]]
## [1] "a"
##
## [[3]]
## [1] "a" "a"
##
## [[4]]
## [1] "a" "a" "a"
##
## [[5]]
## character(0)
##
## [[6]]
## character(0)
```

```r
# str_match() / str_match_all() - Return first or every pattern match (as a matrx)
# Return fruit containing "a" first occurence
str_match(string = fruit, pattern = "a") %>% head()  # matrix is returned
```

```
##      [,1]
## [1,] "a"
## [2,] "a"
## [3,] "a"
## [4,] "a"
## [5,] NA
## [6,] NA
```

```r
# Return fruit containing "a" all occurences
str_match_all(string = fruit, pattern = "a") %>% head() #  matrix inside list is returned
```

```
## [[1]]
##      [,1]
## [1,] "a"
##
## [[2]]
##      [,1]
## [1,] "a"
##
## [[3]]
##      [,1]
## [1,] "a"
## [2,] "a"
##
## [[4]]
##      [,1]
## [1,] "a"
## [2,] "a"
## [3,] "a"
##
## [[5]]
##      [,1]
##
## [[6]]
##      [,1]
```

String lengths

```
# str_length() - Width of a string
# similar base R: nchar()
str_length("word")
```

```
## [1] 4
```

```
nchar("word")
```

```
## [1] 4
```

```
# Find all fruits with length 10 or more characters
fruit[str_length(fruit) >= 10] %>% head()
```

```
## [1] "bell pepper"  "blackberry"   "blackcurrant" "blood orange" "boysenberry"
## [6] "breadfruit"
```

```
# str_pad() - String padding
# Pad fruit names with symbol "X" to get a string with width = 20
str_pad(string = fruit, width = 20, side = "left", pad = "X") %>% head()  # left side padding
```

```
## [1] "XXXXXXXXXXXXXXXapple" "XXXXXXXXXXXXXapricot" "XXXXXXXXXXXXXavocado"
## [4] "XXXXXXXXXXXXXXbanana" "XXXXXXXXXbell pepper" "XXXXXXXXXXXXbilberry"
```

```
str_pad(string = fruit, width = 20, side = "right", pad = "X") %>% head() # right side padding
```

```
## [1] "appleXXXXXXXXXXXXXXXX" "apricotXXXXXXXXXXXXXX" "avocadoXXXXXXXXXXXXXX"
## [4] "bananaXXXXXXXXXXXXXXX" "bell pepperXXXXXXXXX" "bilberryXXXXXXXXXXXXX"
```

```
str_pad(string = fruit, width = 20, side = "both", pad = "X") %>% head()  # both side padding
```

```
## [1] "XXXXXXXappleXXXXXXXX" "XXXXXXapricotXXXXXXX" "XXXXXXavocadoXXXXXXX"
```

```
## [4] "XXXXXXXbananaXXXXXXX" "XXXXbell pepperXXXXX" "XXXXXXbilberryXXXXXX"
```

```
# Where padding is very useful in practice (ID numbers)
set.seed(123)
id.numbers <- sample(x = 1:1000, size = 25, replace = F) # generate some ID numbers
id.numbers %>% head()
```

```
## [1] 415 463 179 526 195 938
```

```
str_pad(id.numbers, width = 5, side = "left", pad = "0") %>% head() # add leading zeros
```

```
## [1] "00415" "00463" "00179" "00526" "00195" "00938"
```

```
# str_trunc() - String truncating
# Truncate fruit names with symbol "..." to get a string with width = 5
str_trunc(string = fruit, width = 5, side = "left", ellipsis = "...") %>% head()   # left side truncati
```

```
## [1] "apple" "...ot" "...do" "...na" "...er" "...ry"
```

```
str_trunc(string = fruit, width = 5, side = "right", ellipsis = "...") %>% head()  # right side truncat
```

```
## [1] "apple" "ap..." "av..." "ba..." "be..." "bi..."
```

```
str_trunc(string = fruit, width = 5, side = "center", ellipsis = "...") %>% head() # center side trunca
```

```
## [1] "apple" "a...t" "a...o" "b...a" "b...r" "b...y"
```

```
# str_trim() - Trim whitespaces
# Create a string with white spaces
whitespace <- c("nospaces",
                " leftspace",
                "      leftspaces",
                "rightspace ",
                "rightspaces     ",
                " bothspace ",
                "   bothspaces   ",
                "middle space",
                " mix space ")
whitespace %>% head()
```

```
## [1] "nospaces"         " leftspace"       "      leftspaces" "rightspace "
## [5] "rightspaces     " " bothspace "
```

```
# Trim left white space(s)
whitespace.trim.left <- str_trim(string = whitespace, side = "left")
whitespace %>% head()
```

```
## [1] "nospaces"         " leftspace"       "      leftspaces" "rightspace "
## [5] "rightspaces     " " bothspace "
```

```
whitespace.trim.left %>% head()
```

```
## [1] "nospaces"         "leftspace"        "leftspaces"       "rightspace "
## [5] "rightspaces     " "bothspace "
```

```
# Trim right white space(s)
whitespace.trim.right <- str_trim(string = whitespace, side = "right")
whitespace %>% head()
```

```
## [1] "nospaces"         " leftspace"       "      leftspaces" "rightspace "
## [5] "rightspaces     " " bothspace "
```

```r
whitespace.trim.right %>% head()
```

```
## [1] "nospaces"         " leftspace"       "      leftspaces" "rightspace"
## [5] "rightspaces"      " bothspace"
```

```r
# Trim both side white space(s)
whitespace.trim.both <- str_trim(string = whitespace, side = "both")
whitespace %>% head()
```

```
## [1] "nospaces"         " leftspace"       "      leftspaces" "rightspace "
## [5] "rightspaces     " " bothspace "
```

```r
whitespace.trim.both %>% head()
```

```
## [1] "nospaces"    "leftspace"   "leftspaces"  "rightspace"  "rightspaces"
## [6] "bothspace"
```

Strings mutating

```r
# str_sub() - Replace a part of given string
# Replace first 3 letters of each fruit with string "FRU"
fruit %>% head()
```

```
## [1] "apple"       "apricot"     "avocado"     "banana"      "bell pepper"
## [6] "bilberry"
```

```r
fruit.sub <- fruit
fruit.sub %>% head()
```

```
## [1] "apple"       "apricot"     "avocado"     "banana"      "bell pepper"
## [6] "bilberry"
```

```r
str_sub(fruit.sub, start = 1, end = 3) <- "FRU"
fruit.sub %>% head()
```

```
## [1] "FRUle"       "FRUicot"     "FRUcado"     "FRUana"      "FRUl pepper"
## [6] "FRUberry"
```

```r
# str_replace() - Replace the first matched pattern in a string
# Replace first occurence of letter "a" with "A" in each fruit
str_replace(string = fruit, pattern = "a", replacement = "A") %>% head()
```

```
## [1] "Apple"       "Apricot"     "Avocado"     "bAnana"      "bell pepper"
## [6] "bilberry"
```

```r
# str_replace_all() - Replace all matched patterns in a string
# Replace all occurences of letter "a" with "A" in each fruit
str_replace_all(string = fruit, pattern = "a", replacement = "A") %>% head()
```

```
## [1] "Apple"       "Apricot"     "AvocAdo"     "bAnAnA"      "bell pepper"
## [6] "bilberry"
```

```r
# str_to_lower() - Convert string to lower case
string.upper <- "THIS IS A STRING"
string.lower <- str_to_lower(string = string.upper)
string.lower %>% head()
```

```
## [1] "this is a string"
```

```r
# str_to_upper() - Convert string to upper case
string.upper <- str_to_lower(string = string.lower)
string.upper %>% head()
```

```
## [1] "this is a string"
```

```r
# str_to_title() - Convert string to "upper"title" case
string.title <- str_to_title(string = string.lower)
string.title %>% head()
```

```
## [1] "This Is A String"
```

Joining and splitting strings

```r
# str_c() - Join multiple strings into a single string
# Let's split vector "fruit" into 4 equal in size smaller vectors
fruit1 <- fruit[1:20]
fruit2 <- fruit[21:40]
fruit3 <- fruit[41:60]
fruit4 <- fruit[61:80]

# Create one vector of strings using all 4 smaller vectors
str_c(fruit1, fruit2, fruit3, fruit4, sep = "-") %>% head()
```

```
## [1] "apple-cranberry-jujube-physalis"
## [2] "apricot-cucumber-kiwi fruit-pineapple"
## [3] "avocado-currant-kumquat-plum"
## [4] "banana-damson-lemon-pomegranate"
## [5] "bell pepper-date-lime-pomelo"
## [6] "bilberry-dragonfruit-loquat-purple mangosteen"
```

```r
# Create vector of alphabet letters: one lower and one upper case
letters %>% head()
```

```
## [1] "a" "b" "c" "d" "e" "f"
```

```r
Letters %>% head()
```

```
## [1] "A" "B" "C" "D" "E" "F"
```

```r
str_c(letters, Letters) %>% head()
```

```
## [1] "aA" "bB" "cC" "dD" "eE" "fF"
```

```r
# str_c() - Colapse a vector of strings into  single string
# Collapse a vector of letters into a single string containing all letters
str_c(letters, collapse = "") %>% head()
```

```
## [1] "abcdefghijklmnopqrstuvwxyz"
```

```r
str_c(letters, collapse = " ") %>% head()
```

```
## [1] "a b c d e f g h i j k l m n o p q r s t u v w x y z"
```

```r
# str_dup() - Repeat a string multiple times
# Repeat one string 5 times
str_dup(string = "string", times = 5) %>% head()
```

```
## [1] "stringstringstringstringstring"
```

```r
# Repeat a vector of strings 2 times
str_dup(string = fruit1, times = 2) %>% head()
```

```
## [1] "appleapple"             "apricotapricot"         "avocadoavocado"
## [4] "bananabanana"           "bell pepperbell pepper" "bilberrybilberry"
```

```r
# str_split_fixed() - Split a vector of strings into a matrix of substrings base on pattern
# Split fruit by " " white space
str_split_fixed(string = fruit, pattern = " ", n = 2) %>% head() # n - number of pieces to return!
```

```
##      [,1]       [,2]
## [1,] "apple"    ""
## [2,] "apricot"  ""
## [3,] "avocado"  ""
## [4,] "banana"   ""
## [5,] "bell"     "pepper"
## [6,] "bilberry" ""
```

```r
# Split first 5 sentences by " " white space - increase n
str_split_fixed(sentences[1:5], pattern = " ", n = 10) %>% head()
```

```
##      [,1]    [,2]    [,3]    [,4]      [,5]  [,6]    [,7]     [,8]
## [1,] "The"   "birch" "canoe" "slid"    "on"  "the"   "smooth" "planks."
## [2,] "Glue"  "the"   "sheet" "to"      "the" "dark"  "blue"   "background."
## [3,] "It's"  "easy"  "to"    "tell"    "the" "depth" "of"     "a"
## [4,] "These" "days"  "a"     "chicken" "leg" "is"    "a"      "rare"
## [5,] "Rice"  "is"    "often" "served"  "in"  "round" "bowls." ""
##      [,9]    [,10]
## [1,] ""      ""
## [2,] ""      ""
## [3,] "well." ""
## [4,] "dish." ""
## [5,] ""      ""
```

```r
# str_split() - Split a vector of strings into a list / matrix of substrings base on pattern
# Split first 5 sentences by " " white space
str_split(sentences[1:5], pattern = " ") %>% head() # return a list
```

```
## [[1]]
## [1] "The"     "birch"   "canoe"   "slid"    "on"      "the"     "smooth"
## [8] "planks."
##
## [[2]]
## [1] "Glue"        "the"         "sheet"       "to"          "the"
## [6] "dark"        "blue"        "background."
##
## [[3]]
## [1] "It's"  "easy"  "to"    "tell"  "the"   "depth" "of"    "a"     "well."
##
## [[4]]
## [1] "These"   "days"    "a"       "chicken" "leg"     "is"      "a"
## [8] "rare"    "dish."
##
## [[5]]
## [1] "Rice"   "is"     "often"  "served" "in"     "round"  "bowls."
```

```r
str_split(sentences[1:5], pattern = " ", simplify = T) %>% head() # return a matrix
```

```
##      [,1]    [,2]    [,3]    [,4]      [,5]  [,6]    [,7]     [,8]
## [1,] "The"   "birch" "canoe" "slid"    "on"  "the"   "smooth" "planks."
## [2,] "Glue"  "the"   "sheet" "to"      "the" "dark"  "blue"   "background."
## [3,] "It's"  "easy"  "to"    "tell"    "the" "depth" "of"     "a"
## [4,] "These" "days"  "a"     "chicken" "leg" "is"    "a"      "rare"
## [5,] "Rice"  "is"    "often" "served"  "in"  "round" "bowls." ""
##      [,9]
## [1,] ""
## [2,] ""
## [3,] "well."
## [4,] "dish."
## [5,] ""
```

```r
# str_glue() - Glue/merge together string and expression
# Merge string and evaluated mathematical symbol
str_glue("What is the value of sqrt(2), it is {sqrt(2)}.") %>% head()
```

```
## What is the value of sqrt(2), it is 1.4142135623731.
```

```r
# Merge fixed string and assigned string to a variable
name <- "Marko"
str_glue("Hi my name is {name}")
```

```
## Hi my name is Marko
```

```r
# str_glue_data() - Use data.frame / list or environment to create strings from string and expression
# Merge string and values from a data.frame
mtcars %>% head()
```

```
##                   mpg cyl disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
## Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
## Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
## Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
## Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
## Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1
```

```r
str_glue_data(mtcars, "The car {rownames(mtcars)}: {hp} horsepower, {cyl} number of cylinders and consu
```

```
## The car Mazda RX4: 110 horsepower, 6 number of cylinders and consumption 21 miles per gallon
## The car Mazda RX4 Wag: 110 horsepower, 6 number of cylinders and consumption 21 miles per gallon
## The car Datsun 710: 93 horsepower, 4 number of cylinders and consumption 22.8 miles per gallon
## The car Hornet 4 Drive: 110 horsepower, 6 number of cylinders and consumption 21.4 miles per gallon
## The car Hornet Sportabout: 175 horsepower, 8 number of cylinders and consumption 18.7 miles per gall
## The car Valiant: 105 horsepower, 6 number of cylinders and consumption 18.1 miles per gallon
```

Other string helper functions

```r
# str_order() - Return a vector of indexes after character vector is sorted.
# Let's first shuffle fruits (to get random order)
set.seed(42)
fruit.shuf <- sample(x = fruit, size = length(fruit), replace = F)
fruit.shuf %>% head()
```

```
## [1] "mango"      "pomelo"     "date"       "satsuma"    "clementine"
```

```
## [6] "watermelon"
```

```
# Now get order index and use to sort shuffled fruits
str_order(x = fruit.shuf) # get index
```

```
##  [1] 59 48 13 35 20 46 61 33 65 56 70 67 63 75 31 69 60  5 68 11 52 32 44  8  3
## [26] 12 15 23 79 29 18 78 26 22 42 16 10 76 47 24 14 27 30 66 19 40  7 57  1 73
## [51] 77 80 43 58 53 50 55 38 41 49 39 51 71 54  2 72 62 25 45 21  9 17 28  4 36
## [76] 37 74 64 34  6
```

```
fruit.shuf[str_order(x = fruit.shuf)] %>% head() # use it for sort
```

```
## [1] "apple"       "apricot"     "avocado"     "banana"      "bell pepper"
## [6] "bilberry"
```

```
# str_sort() - Sort character vector
# Let' sort shuffled fruits
str_sort(x = fruit.shuf) %>% head()
```

```
## [1] "apple"       "apricot"     "avocado"     "banana"      "bell pepper"
## [6] "bilberry"
```

```
str_sort(x = fruit.shuf, decreasing = T) %>% head()
```

```
## [1] "watermelon" "ugli fruit" "tangerine"  "tamarillo"  "strawberry"
## [6] "star fruit"
```

```
# Sorting numbers stored as strings!
set.seed(567)
numbers.s <- sample(1:250, size = 20, replace = F) # generate some numbers
numbers.s <- as.character(numbers.s) # conver numbers to charatcer
numbers.s %>% head()
```

```
## [1] "141" "28"  "69"  "199" "215" "46"
```

```
str_sort(numbers.s) # not sorted as numbers but as strings
```

```
##  [1] "106" "141" "166" "182" "185" "199" "20"  "21"  "211" "215" "28"  "29"
## [13] "46"  "47"  "54"  "62"  "69"  "76"  "82"  "95"
```

```
str_sort(numbers.s, numeric = T) # sorted as numbers
```

```
##  [1] "20"  "21"  "28"  "29"  "46"  "47"  "54"  "62"  "69"  "76"  "82"  "95"
## [13] "106" "141" "166" "182" "185" "199" "211" "215"
```

```
# str_view() / str_view_all() - Useful HTML rendering function
# very useful in the context of regular expressions
# (given context will be shown later)
# View first match
str_view(string = fruit, pattern = "a")  %>% head() # displays all
```

```
## [1] | <a>pple
## [2] | <a>pricot
## [3] | <a>voc<a>do
## [4] | b<a>n<a>n<a>
## [7] | bl<a>ckberry
## [8] | bl<a>ckcurr<a>nt
```

```
str_view(string = fruit, pattern = "a", match = T) %>% head() # display only matched
```

```
## [1] | <a>pple
```

```
## [2] | <a>pricot
## [3] | <a>voc<a>do
## [4] | b<a>n<a>n<a>
## [7] | bl<a>ckberry
## [8] | bl<a>ckcurr<a>nt
```

```
str_view(string = fruit, pattern = "^a", match = T) %>% head()
```

```
## [1] | <a>pple
## [2] | <a>pricot
## [3] | <a>vocado
```

```
# View all matches
str_view_all(string = fruit, pattern = "a", match = T) %>% head()
```

```
## [1] | <a>pple
## [2] | <a>pricot
## [3] | <a>voc<a>do
## [4] | b<a>n<a>n<a>
## [7] | bl<a>ckberry
## [8] | bl<a>ckcurr<a>nt
```

Regular expressions (regex)

```
# Get list of some special characters
# ?"'"

# Escaping paradox
string <- c("string", "word", "letter", "word.letter", "character/letter")

# Match "tr"
str_view(string, "tr")
```

```
## [1] | s<tr>ing
```

```
# Match ".t." - any character before t and any character after t
str_view(string, ".t.")
```

```
## [1] | <str>ing
## [3] | l<ett>er
## [4] | word.l<ett>er
## [5] | chara<cte>r/l<ett>er
```

```
str_view_all(string, ".t.")
```

```
## [1] | <str>ing
## [2] | word
## [3] | l<ett>er
## [4] | word.l<ett>er
## [5] | chara<cte>r/l<ett>er
```

```
# Match "." as a dot not as a metacharacter meaning:
# 1) wrong way: since . is interpreted as metacharater ~ any charatcer
# str_view(string, ".")

# 2) wrong way (single backslash \ ): escaping is applied on . but \ is not escaped!
# str_view(string, "\.")
```

```r
# 3) correct way (double backslash \\ ): escaping is applied on . and \ !
# str_view(string, "\\.")

# Match "\" as a backslash character not as a metacharacter meaning:
# writeLines("\\") # \ must be escaped when written as a string
# str_view("\\", "\\\\") # double escaping is applied int he pattern ~ four \ in total at the end!
```

Regex: Special characters & Classes

```r
# Digits VS non-digits
string <- c(letters, "123", "1-5-6", "598642")
string %>% head()
```

```
## [1] "a" "b" "c" "d" "e" "f"
```

```r
# Find strings with digits
str_subset(string, "\\d") %>% head()
```

```
## [1] "123"    "1-5-6"  "598642"
```

```r
str_view_all(string, "\\d", match = T) %>% head()
```

```
## [27] | <1><2><3>
## [28] | <1>-<5>-<6>
## [29] | <5><9><8><6><4><2>
```

```r
# Find strings without digits
str_subset(string, "\\D") %>% head()
```

```
## [1] "a" "b" "c" "d" "e" "f"
```

```r
str_view_all(string, "\\D", match = T) %>% head()
```

```
## [1] | <a>
## [2] | <b>
## [3] | <c>
## [4] | <d>
## [5] | <e>
## [6] | <f>
```

```r
# Strings with pattern "digit-digit-digit"
str_subset(string, "\\d-\\d-\\d") %>% head()
```

```
## [1] "1-5-6"
```

```r
str_view_all(string, "\\d-\\d-\\d", match = T) %>% head()
```

```
## [28] | <1-5-6>
```

```r
# Locate whitespace(s)
set.seed(42)
string <- c(sample(sentences, 5),
            sample(fruit, 5),
            sample(words, 5),
            "This is \nnewline",
            "String with a tab \t")
string %>% head()
```

```
## [1] "Paint the sockets in the wall dull green."
## [2] "Fill the ink jar with sticky glue."
## [3] "He broke a new shoelace that day."
## [4] "The walled town was seized without a fight."
## [5] "Jazz and swing fans like fast music."
## [6] "clementine"
```

```
writeLines(string) %>% head()
```

```
## Paint the sockets in the wall dull green.
## Fill the ink jar with sticky glue.
## He broke a new shoelace that day.
## The walled town was seized without a fight.
## Jazz and swing fans like fast music.
## clementine
## mango
## lychee
## damson
## redcurrant
## burn
## meaning
## around
## term
## snow
## This is
## newline
## String with a tab
```

```
## NULL
```

```
str_subset(string, "\\s") %>% head() # only strings with white spaces
```

```
## [1] "Paint the sockets in the wall dull green."
## [2] "Fill the ink jar with sticky glue."
## [3] "He broke a new shoelace that day."
## [4] "The walled town was seized without a fight."
## [5] "Jazz and swing fans like fast music."
## [6] "This is \nnewline"
```

```
str_view_all(string, "\\s") %>% head()
```

```
## [1] | Paint< >the< >sockets< >in< >the< >wall< >dull< >green.
## [2] | Fill< >the< >ink< >jar< >with< >sticky< >glue.
## [3] | He< >broke< >a< >new< >shoelace< >that< >day.
## [4] | The< >walled< >town< >was< >seized< >without< >a< >fight.
## [5] | Jazz< >and< >swing< >fans< >like< >fast< >music.
## [6] | clementine
```

```
# Locate string with new lines or tabs
str_subset(string, "\\n") %>% head() # only strings with new lines
```

```
## [1] "This is \nnewline"
```

```
str_subset(string, "\\t") %>% head() # only strings with tabs
```

```
## [1] "String with a tab \t"
```

```r
# Different classes
string <- c("123abc", "abc", "123", ".,?", "ABC", "\nABC", "\tabc")
string %>% head()
```

```
## [1] "123abc" "abc"    "123"    ".,?"    "ABC"    "\nABC"
```

```r
# Strings with digits
str_subset(string, "[:digit:]") %>% head()
```

```
## [1] "123abc" "123"
```

```r
str_view_all(string, "[:digit:]", match = T) %>% head()
```

```
## [1] | <1><2><3>abc
## [3] | <1><2><3>
```

```r
# Strings with letters
str_subset(string, "[:alpha:]") %>% head()
```

```
## [1] "123abc" "abc"    "ABC"    "\nABC"  "\tabc"
```

```r
str_view_all(string, "[:alpha:]", match = T) %>% head()
```

```
## [1] | 123<a><b><c>
## [2] | <a><b><c>
## [5] | <A><B><C>
## [6] |
##     | <A><B><C>
## [7] | {\t}<a><b><c>
```

```r
# Strings with upper / lower case letters
str_subset(string, "[:lower:]") %>% head()
```

```
## [1] "123abc" "abc"    "\tabc"
```

```r
str_view_all(string, "[:lower:]", match = T) %>% head()
```

```
## [1] | 123<a><b><c>
## [2] | <a><b><c>
## [7] | {\t}<a><b><c>
```

```r
str_subset(string, "[:upper:]") %>% head()
```

```
## [1] "ABC"   "\nABC"
```

```r
str_view_all(string, "[:upper:]", match = T) %>% head()
```

```
## [5] | <A><B><C>
## [6] |
##     | <A><B><C>
```

```r
# Strings with letters or numbers
str_subset(string, "[:alnum:]") %>% head()
```

```
## [1] "123abc" "abc"    "123"    "ABC"    "\nABC"  "\tabc"
```

```r
str_view_all(string, "[:alnum:]", match = T) %>% head()
```

```
## [1] | <1><2><3><a><b><c>
## [2] | <a><b><c>
## [3] | <1><2><3>
```

```
## [5] | <A><B><C>
## [6] |
##     | <A><B><C>
## [7] | {\t}<a><b><c>
```

```
# Strings with punctuation
str_subset(string, "[:punct:]") %>% head()
```

```
## [1] ".,?"
```

```
str_view_all(string, "[:punct:]", match = T) %>% head()
```

```
## [4] | <.><,><?>
```

```
# Strings with letters, numbers or punctuation
str_subset(string, "[:graph:]") %>% head()
```

```
## [1] "123abc" "abc"    "123"    ".,?"    "ABC"    "\nABC"
```

```
str_view_all(string, "[:graph:]", match = T) %>% head()
```

```
## [1] | <1><2><3><a><b><c>
## [2] | <a><b><c>
## [3] | <1><2><3>
## [4] | <.><,><?>
## [5] | <A><B><C>
## [6] |
##     | <A><B><C>
```

```
# Strings with space characters
str_subset(string, "[:blank:]") %>% head()
```

```
## [1] "\tabc"
```

```
str_view_all(string, "[:blank:]", match = T) %>% head()
```

```
## [7] | <{\t}>abc
```

Regex: Alternates, anchors & groups

```
# Anchors
# Find a word starting with letter "a"
str_subset(words, "^a") %>% head()
```

```
## [1] "a"        "abandon"  "ability"  "able"     "abortion" "about"
```

```
str_view_all(words, "^a", match = T) %>% head()
```

```
## [1] | <a>
## [2] | <a>bandon
## [3] | <a>bility
## [4] | <a>ble
## [5] | <a>bortion
## [6] | <a>bout
```

```
# Find a word ending with letter "a"
str_subset(words, "a$") %>% head()
```

```
## [1] "a"        "agenda"   "area"     "camera"   "criteria" "data"
```

```r
str_view_all(words, "a$", match = T) %>% head()
```

```
##   [1] | <a>
##  [77] | agend<a>
## [158] | are<a>
## [373] | camer<a>
## [650] | criteri<a>
## [680] | dat<a>
```

```r
# Find exact word using ^....$
str_subset(words, "^actor$") %>% head()
```

```
## [1] "actor"
```

```r
str_view_all(words, "^actor$", match = T) %>% head()
```

```
## [35] | <actor>
```

```r
str_subset(fruit, "^lemon$") %>% head()
```

```
## [1] "lemon"
```

```r
str_view_all(fruit, "^lemon$", match = T) %>% head()
```

```
## [44] | <lemon>
```

```r
# Alternates
# Find words that starts with "af" or "ag"
str_subset(words, "^af|^ag") %>% head()
```

```
## [1] "affair"    "affect"    "afford"    "afraid"    "after"     "afternoon"
```

```r
str_view_all(words, "^af|^ag", match = T) %>% head()
```

```
## [65] | <af>fair
## [66] | <af>fect
## [67] | <af>ford
## [68] | <af>raid
## [71] | <af>ter
## [72] | <af>ternoon
```

```r
# Find words containing letters "x" or "y" or "z"
str_subset(words, "[xyz]") %>% head()
```

```
## [1] "ability"    "absolutely" "accompany"  "activity"   "actually"
## [6] "agency"
```

```r
# Find words not containing letters from "a" to "x"
str_subset(words %>% str_to_lower(), "[^[a-y]]") %>% head()
```

```
## [1] "african-american" "amazing"          "analyze"          "characterize"
## [5] "citizen"          "crazy"
```

```r
str_view_all(words %>% str_to_lower(), "[^[a-y]]", match = T) %>% head()
```

```
##  [70] | african<->american
## [111] | ama<z>ing
## [117] | analy<z>e
## [431] | characteri<z>e
## [460] | citi<z>en
## [639] | cra<z>y
```

```r
# Find all country names beginning with letter "A" or "E"
str_subset(countries, "^A|^E") %>% head()
```

```
## [1] "Afghanistan"    "Albania"        "Algeria"        "American Samoa"
## [5] "Andorra"        "Angola"
```

```r
# Find all country names ending with letter "a" or "e"
str_subset(countries, "a$|e$") %>% head()
```

```
## [1] "Albania"        "Algeria"        "American Samoa" "Andorra"
## [5] "Angola"         "Anguilla"
```

```r
# Groups
# Find all sentences that include words: "the", "a" or "an"
str_subset(sentences, "(\\sthe\\s|\\sa\\s|\\san\\s)") %>% head()
```

```
## [1] "The birch canoe slid on the smooth planks."
## [2] "Glue the sheet to the dark blue background."
## [3] "It's easy to tell the depth of a well."
## [4] "These days a chicken leg is a rare dish."
## [5] "The box was thrown beside the parked truck."
## [6] "The boy was there when the sun rose."
```

```r
str_view_all(sentences, "(\\sthe\\s|\\sa\\s|\\san\\s)", match = T) %>% head()
```

```
##  [1] | The birch canoe slid on< the >smooth planks.
##  [2] | Glue< the >sheet to< the >dark blue background.
##  [3] | It's easy to tell< the >depth of< a >well.
##  [4] | These days< a >chicken leg is< a >rare dish.
##  [7] | The box was thrown beside< the >parked truck.
## [11] | The boy was there when< the >sun rose.
```

```r
# Find words with repeated pair of letters (two letters must be repeated): use back references
str_subset(words, "(..)\\1") %>% head()   # \1 is a group reference 1st group, double backslash ~ escap
```

```
## [1] "competition" "competitive" "crisis"      "dining"      "remaining"
## [6] "remember"
```

```r
str_view_all(words, "(..)\\1", match = T) %>% head()
```

```
##  [526] | compe<titi>on
##  [527] | compe<titi>ve
##  [649] | cr<isis>
##  [763] | d<inin>g
## [2207] | rema<inin>g
## [2209] | r<emem>ber
```

```r
str_subset(fruit, "(..)\\1") %>% head()
```

```
## [1] "banana"      "coconut"     "cucumber"    "jujube"      "papaya"
## [6] "salal berry"
```

```r
str_view_all(fruit, "(..)\\1", match = T) %>% head()
```

```
##  [4] | b<anan>a
## [20] | <coco>nut
## [22] | <cucu>mber
## [41] | <juju>be
## [56] | <papa>ya
```

```
## [73] | s<alal> berry
```

```
# Mor ethan one greoup in back reference
string <- c("abc", "abcabc", "ababcc", "abababccc")
string %>% head()
```

```
## [1] "abc"       "abcabc"    "ababcc"    "abababccc"
```

```
str_view_all(string, "(a)(b)", match = T)             # ab
```

```
## [1] | <ab>c
## [2] | <ab>c<ab>c
## [3] | <ab><ab>cc
## [4] | <ab><ab><ab>ccc
```

```
str_view_all(string, "(a)(b)\\1", match = T)          # aba
```

```
## [3] | <aba>bcc
## [4] | <aba>babccc
```

```
str_view_all(string, "(a)(b)\\1\\2", match = T)       # abab
```

```
## [3] | <abab>cc
## [4] | <abab>abccc
```

```
str_view_all(string, "(a)(b)\\1\\2\\1\\2", match = T) # ababab
```

```
## [4] | <ababab>ccc
```

Regex: Look arounds & quantifiers

```
# Look arounds
# Find a word where letter "w" is followed by letter "a"
str_subset(words, "w(?=a)") %>% head()
```

```
## [1] "always"    "anyway"    "award"     "aware"     "awareness" "away"
```

```
str_view_all(words, "w(?=a)", match = T) %>% head()
```

```
## [109] | al<w>ays
## [136] | any<w>ay
## [217] | a<w>ard
## [218] | a<w>are
## [219] | a<w>areness
## [220] | a<w>ay
```

```
# Find a word where letter "w" is not followed by letter "a"
str_subset(words, "w(?!a)") %>% head()
```

```
## [1] "acknowledge" "allow"       "answer"      "anywhere"    "awful"
## [6] "below"
```

```
str_view_all(words, "w(?!a)", match = T) %>% head()
```

```
##  [27] | ackno<w>ledge
##  [99] | allo<w>
## [128] | ans<w>er
## [137] | any<w>here
## [221] | a<w>ful
## [270] | belo<w>
```

```r
# Find a word where letter "a" is preceded by letter "w"
str_subset(words, "(?<=w)a") %>% head()
```

```
## [1] "always"    "anyway"    "award"     "aware"     "awareness" "away"
```

```r
str_view_all(words, "(?<=w)a", match = T) %>% head()
```

```
## [109] | alw<a>ys
## [136] | anyw<a>y
## [217] | aw<a>rd
## [218] | aw<a>re
## [219] | aw<a>reness
## [220] | aw<a>y
```

```r
# Find a word where letter "a" is not preceded by letter "w"
str_subset(words, "(?<!w)a") %>% head()
```

```
## [1] "a"        "abandon"  "ability"  "able"     "abortion" "about"
```

```r
str_view_all(words, "(?<!w)a", match = T) %>% head()
```

```
## [1] | <a>
## [2] | <a>b<a>ndon
## [3] | <a>bility
## [4] | <a>ble
## [5] | <a>bortion
## [6] | <a>bout
```

```r
# Quantifiers
string <- " .A.AA.AAA.AAAA"

# zero or one "A"
str_view_all(string, "A?") %>% head()
```

```
## [1] | <> <>.<A><>.<A><A><>.<A><A><A><>.<A><A><A><A><>
```

```r
# zero or more "A"
str_view_all(string, "A*") %>% head()
```

```
## [1] | <> <>.<A><>.<AA><>.<AAA><>.<AAAA><>
```

```r
# one or more "A"
str_view_all(string, "A+") %>% head()
```

```
## [1] |  .<A>.<AA>.<AAA>.<AAAA>
```

```r
# exactly 2 "A"
str_view_all(string, "A{2}") %>% head()
```

```
## [1] |  .A.<AA>.<AA>A.<AA><AA>
```

```r
# 2 or more "A"
str_view_all(string, "A{2,}") %>% head()
```

```
## [1] |  .A.<AA>.<AAA>.<AAAA>
```

```r
# between 2 and 3 "A"
str_view_all(string, "A{2,3}") %>% head()
```

```
## [1] |  .A.<AA>.<AAA>.<AAA>A
```

```r
# Exercise with sentences
# count the number of words in each sentence
# first remove all punctuation and convert all to lower case
#   then count the number of words and show results
sentences.df1 <- sentences.df %>%
  mutate(sentence = str_remove_all(sentence, "[:punct:]"), # remove punctuation
         sentence = str_to_lower(sentence)) %>%  # convert to lower case
  mutate(`nr words` = str_count(string = sentence, "\\s+") + 1) # counts number of spaces between words

sentences.df1 %>% count(`nr words`) # show frequencies
```

```
## # A tibble: 8 x 2
##   `nr words`     n
##        <dbl> <int>
## 1          5    10
## 2          6    56
## 3          7   188
## 4          8   245
## 5          9   150
## 6         10    54
## 7         11    15
## 8         12     2
```

```r
# Countries with more than 3 words in a country name
countries.df %>%
  mutate(`nr words` = str_count(string = country, "\\s+") + 1) %>%
  filter(`nr words` > 3) %>%
  head()
```

```
## # A tibble: 6 x 2
##   country                           `nr words`
##   <chr>                                  <dbl>
## 1 British Indian Ocean Territory             4
## 2 Heard & McDonald Islands                   4
## 3 Hong Kong SAR China                        4
## 4 Micronesia (Federated States of)           4
## 5 St. Kitts & Nevis                          4
## 6 Saint Martin (French part)                 4
```

Factors ~ forcats

```r
# First lets' create a factor variables
df <- mpg %>%
  mutate_at(.vars = c("manufacturer", "model", "trans", "class"), .funs = as_factor)
str(df) %>% head()
```

```
## tibble [234 x 11] (S3: tbl_df/tbl/data.frame)
##  $ manufacturer: Factor w/ 15 levels "audi","chevrolet",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ model       : Factor w/ 38 levels "a4","a4 quattro",..: 1 1 1 1 1 1 1 2 2 2 ...
##  $ displ       : num [1:234] 1.8 1.8 2 2 2.8 2.8 3.1 1.8 1.8 2 ...
##  $ year        : int [1:234] 1999 1999 2008 2008 1999 1999 2008 1999 1999 2008 ...
##  $ cyl         : int [1:234] 4 4 4 4 6 6 6 4 4 4 ...
##  $ trans       : Factor w/ 10 levels "auto(l5)","manual(m5)",..: 1 2 3 4 1 2 4 2 1 3 ...
##  $ drv         : chr [1:234] "f" "f" "f" "f" ...
##  $ cty         : int [1:234] 18 21 20 21 16 18 18 18 16 20 ...
```

```
##  $ hwy          : int [1:234] 29 29 31 30 26 26 27 26 25 28 ...
##  $ fl           : chr [1:234] "p" "p" "p" "p" ...
##  $ class        : Factor w/ 7 levels "compact","midsize",..: 1 1 1 1 1 1 1 1 1 1 ...
```

```
## NULL
```

```
# Check factor levels
df$manufacturer %>% levels() %>% head()
```

```
## [1] "audi"       "chevrolet" "dodge"     "ford"      "honda"     "hyundai"
```

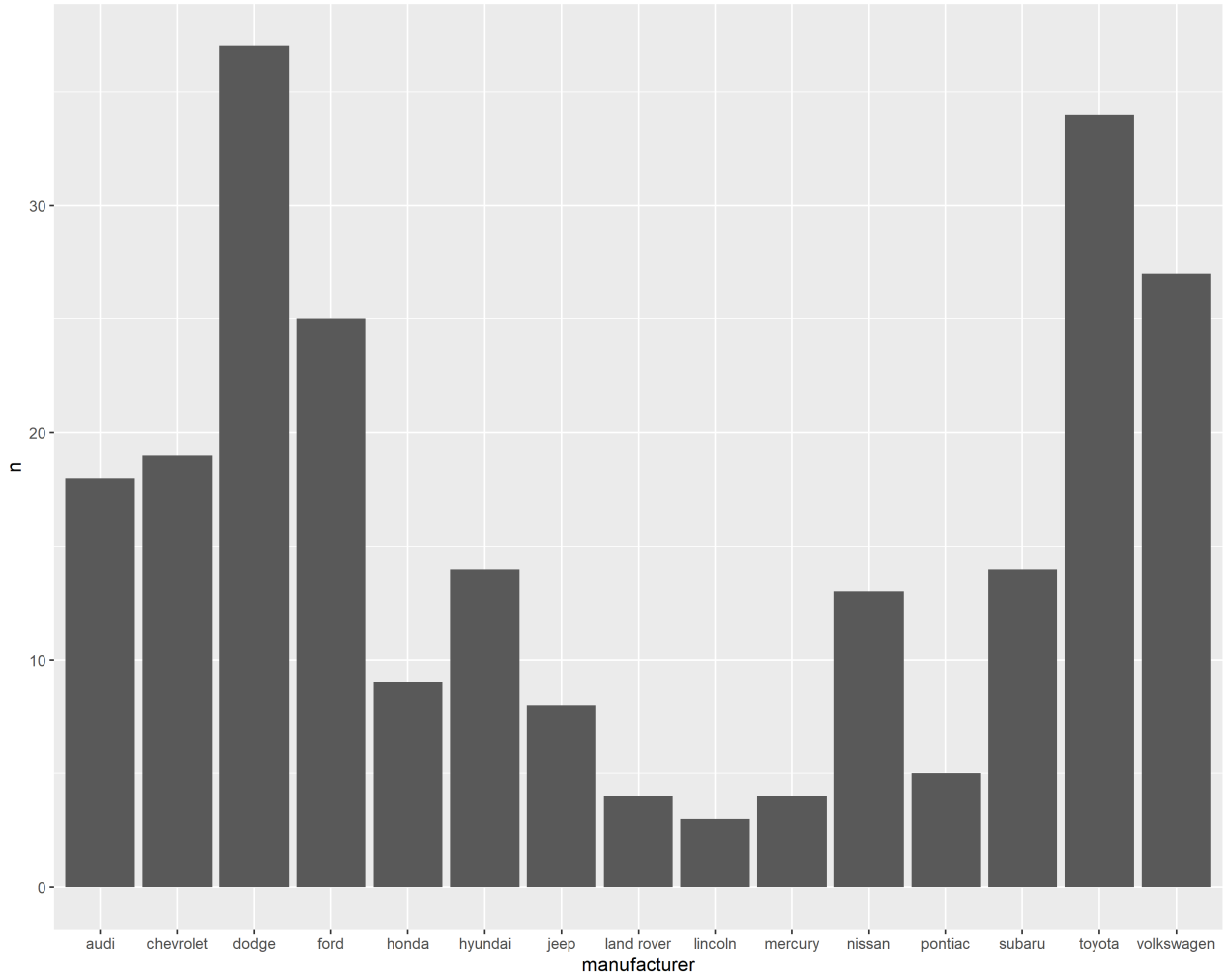
```
# fct_count() - Count factor values
# Check car manufacturer (frequencies count)
df %>% .$manufacturer %>%  fct_count() %>% head()
```

```
## # A tibble: 6 x 2
##   f              n
##   <fct>      <int>
## 1 audi          18
## 2 chevrolet     19
## 3 dodge         37
## 4 ford          25
## 5 honda          9
## 6 hyundai       14
```

```
df %>% count(manufacturer) %>% head()
```

```
## # A tibble: 6 x 2
##   manufacturer     n
##   <fct>        <int>
## 1 audi            18
## 2 chevrolet       19
## 3 dodge           37
## 4 ford            25
## 5 honda            9
## 6 hyundai         14
```

```
# Let's visualize frequencies (ggplot2 section is coming later!)
df %>%
  count(manufacturer) %>%
  ggplot(aes(x = manufacturer,
             y = n)) +
  geom_col()
```




```
# fct_unique() - Extract unique levels
# Car manufacturer unique levels
df %>% .$manufacturer %>% fct_unique() %>% head()
```

```
## [1] audi       chevrolet dodge      ford       honda      hyundai
## 15 Levels: audi chevrolet dodge ford honda hyundai jeep land rover ... volkswagen
```

```
df %>% .$manufacturer %>% fct_unique() %>% as.character() %>% head()
```

```
## [1] "audi"       "chevrolet" "dodge"      "ford"       "honda"      "hyundai"
```

Factors combine and order levels

```
# fct_c() - Combine factors
# First lets split cars into 2 data frames
manufacturers <- df %>% .$manufacturer %>% fct_unique() %>% as.character() # unique manufacturers

df1 <- df %>%  # first subset
  filter(manufacturer %in% manufacturers[1:8])

df2 <- df %>%  # second subset
  filter(manufacturer %in% manufacturers[9:15])
```

```r
# Extract only factor vectors
f1 <- df1 %>% pull(manufacturer)
f2 <- df2 %>% pull(manufacturer)

# Combine factors
c(f1, f2) %>% head()    # with classical vector bind we lose factor level labels!
```

```
## [1] audi audi audi audi audi audi
## 15 Levels: audi chevrolet dodge ford honda hyundai jeep land rover ... volkswagen
```

```r
fct_c(f1,f2) %>% head() # This way levels are preserved!
```

```
## [1] audi audi audi audi audi audi
## 15 Levels: audi chevrolet dodge ford honda hyundai jeep land rover ... volkswagen
```

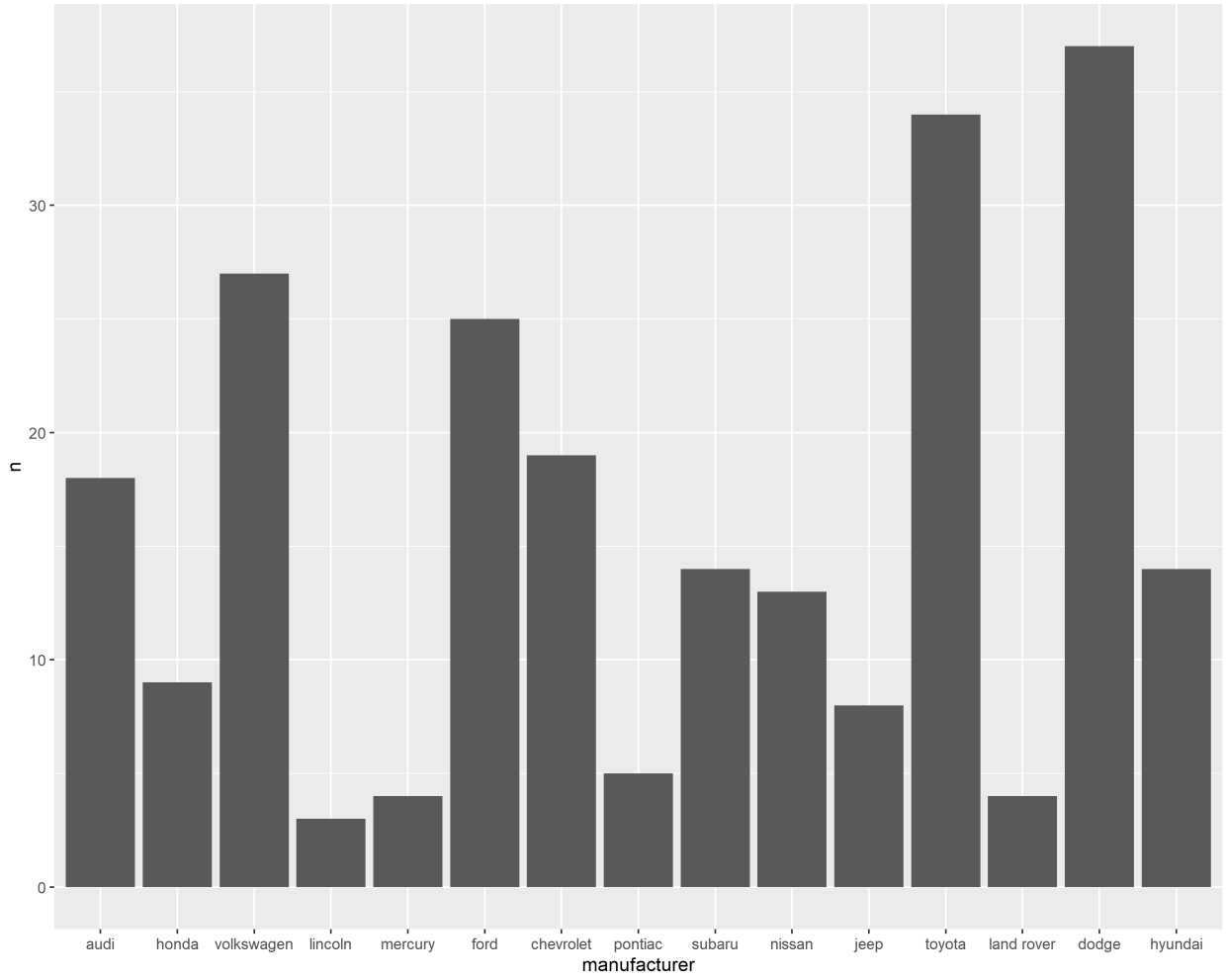
```r
# fct_relevel() - Manually reorder levels
# Lets randomly shuffle levels - manufacturers
set.seed(42)

manufacturers.rnd <- sample(manufacturers, size = length(manufacturers), replace = F)

# Now count frequencies &  create another bar plot with manually reordered levels
df %>%
  mutate(manufacturer = fct_relevel(manufacturer, manufacturers.rnd)) %>%
  count(manufacturer) %>%
  head()
```

```
## # A tibble: 6 x 2
##   manufacturer     n
##   <fct>        <int>
## 1 audi            18
## 2 honda            9
## 3 volkswagen      27
## 4 lincoln          3
## 5 mercury          4
## 6 ford            25
```

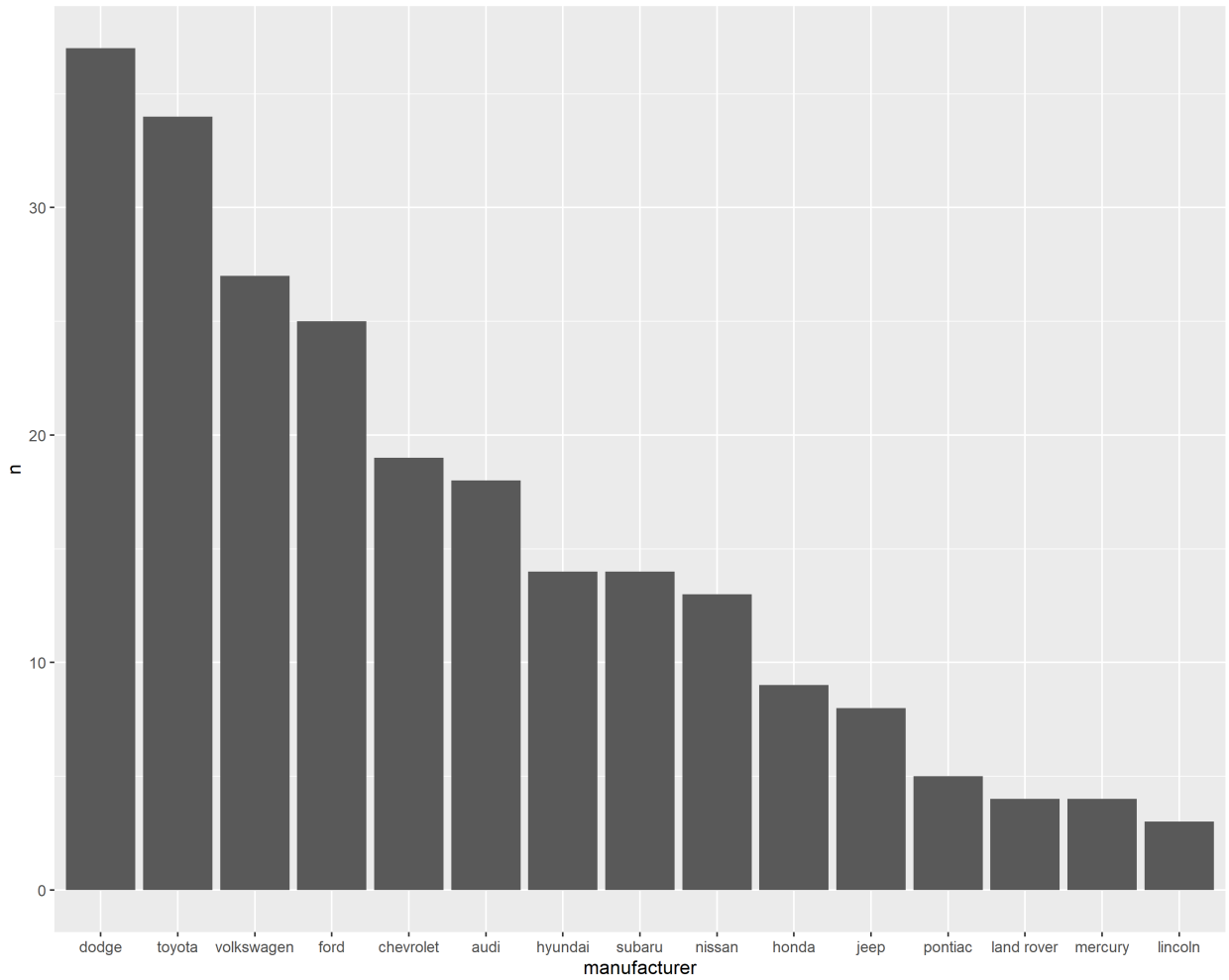
```r
df %>%
  mutate(manufacturer = fct_relevel(manufacturer, manufacturers.rnd)) %>%
  count(manufacturer) %>%
  ggplot(aes(x = manufacturer,
             y = n)) +
  geom_col()
```




```
# fct_infreq() - Levels by frequency
# Order manufacturers based on car count
df %>%
  mutate(manufacturer = fct_infreq(manufacturer)) %>%
  count(manufacturer) %>%
  head()
```

```
## # A tibble: 6 x 2
##   manufacturer     n
##   <fct>        <int>
## 1 dodge           37
## 2 toyota          34
## 3 volkswagen      27
## 4 ford            25
## 5 chevrolet       19
## 6 audi            18
```

```
df %>%
  mutate(manufacturer = fct_infreq(manufacturer)) %>%
  count(manufacturer) %>%
  ggplot(aes(x = manufacturer,
             y = n)) +
  geom_col()
```

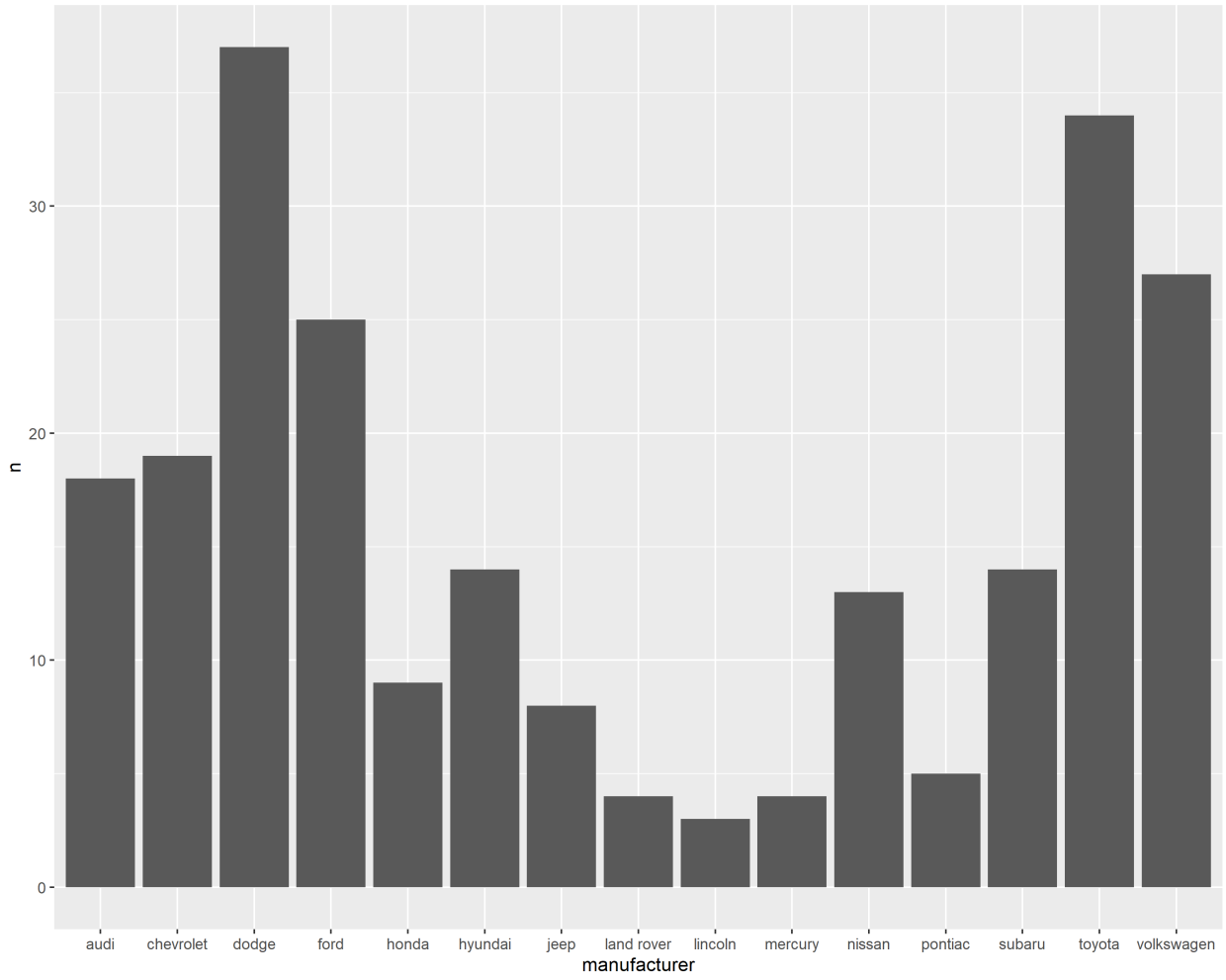



```
# fct_inorder() - Levels by order of appearance
# Order manufacturers based how they appear in the data
df %>%
  mutate(manufacturer = fct_inorder(manufacturer)) %>%
  count(manufacturer) %>%
  head()
```

```
## # A tibble: 6 x 2
##   manufacturer     n
##   <fct>        <int>
## 1 audi            18
## 2 chevrolet       19
## 3 dodge           37
## 4 ford            25
## 5 honda            9
## 6 hyundai         14
```

```
df %>%
  mutate(manufacturer = fct_inorder(manufacturer)) %>%
  count(manufacturer) %>%
  ggplot(aes(x = manufacturer,
```

```
            y = n)) +
  geom_col()
```

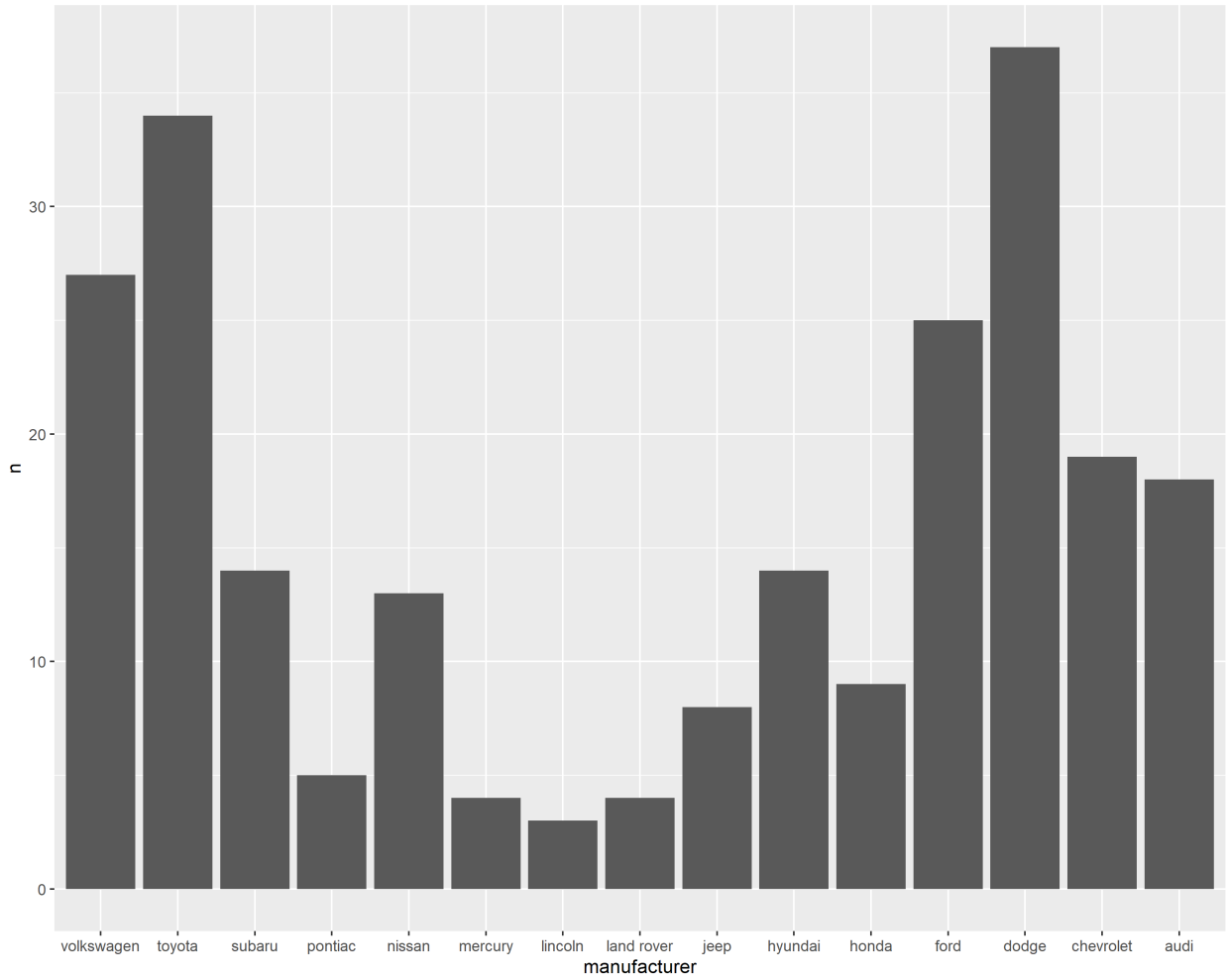



```
# fct_rev() - Reverse level of order
# Order manufacturers based on reverse appereance in the data
df %>%
  mutate(manufacturer = fct_rev(manufacturer)) %>%
  count(manufacturer) %>%
  head()
```

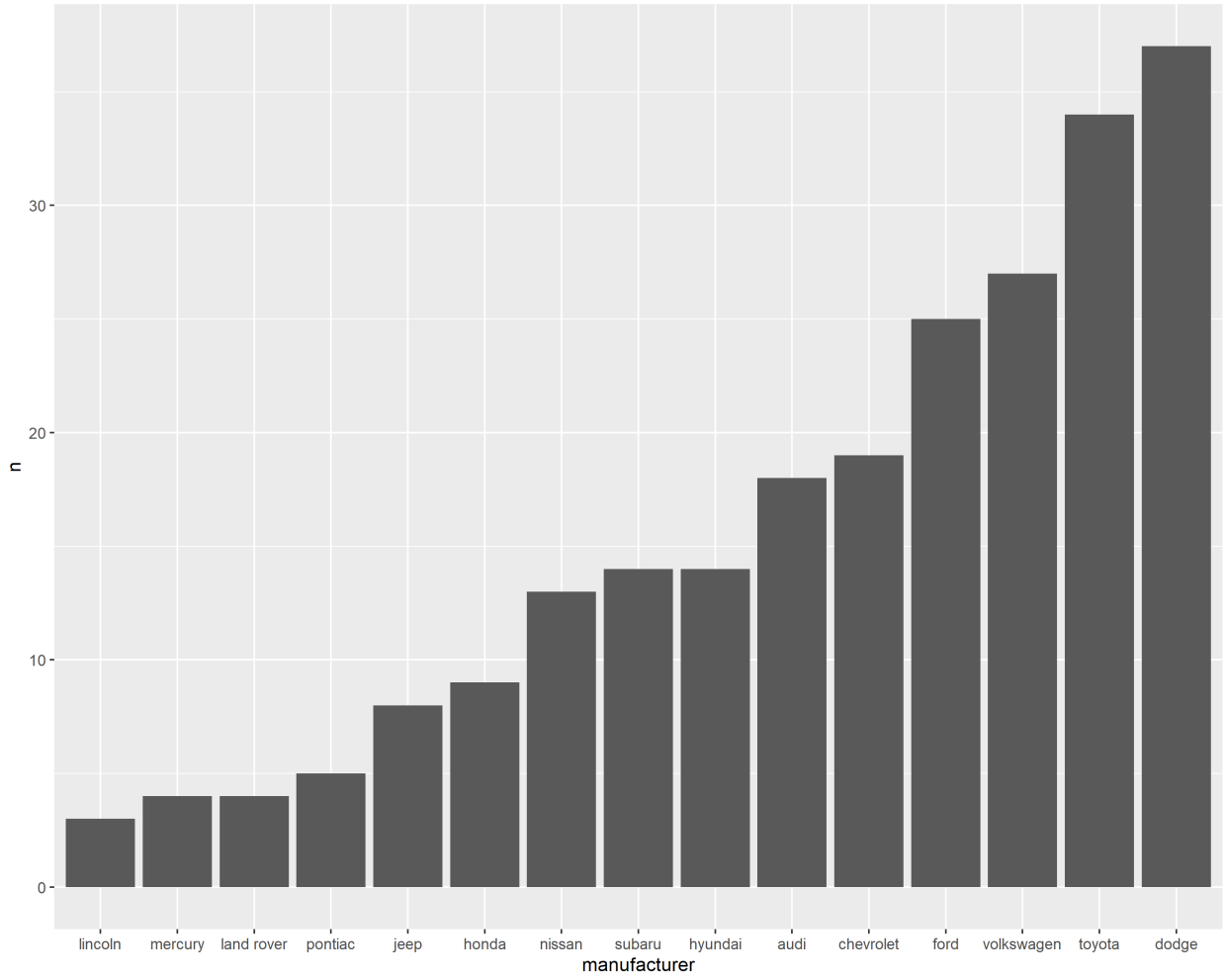
```
## # A tibble: 6 x 2
##   manufacturer     n
##   <fct>        <int>
## 1 volkswagen      27
## 2 toyota          34
## 3 subaru          14
## 4 pontiac          5
## 5 nissan          13
## 6 mercury          4
```

```
df %>%
  mutate(manufacturer = fct_rev(manufacturer)) %>%
```

```
  count(manufacturer) %>%
  ggplot(aes(x = manufacturer,
             y = n)) +
  geom_col()
```




```
# To get reverse frequency count: fct_infreq + fct_rev
df %>%
  mutate(manufacturer = fct_infreq(manufacturer),
         manufacturer = fct_rev(manufacturer)) %>%
  count(manufacturer) %>%
  ggplot(aes(x = manufacturer,
             y = n)) +
  geom_col()
```



Factors change levels value and add / drop levels

```
# fct_recode() - Manually change levels
# First lets pull levels and add county of origin
df %>% pull(manufacturer) %>% fct_count()
```

```
## # A tibble: 15 x 2
##    f              n
##    <fct>      <int>
##  1 audi          18
##  2 chevrolet     19
##  3 dodge         37
##  4 ford          25
##  5 honda          9
##  6 hyundai       14
##  7 jeep           8
##  8 land rover     4
##  9 lincoln        3
## 10 mercury        4
## 11 nissan        13
## 12 pontiac        5
```

```
## 13 subaru         14
## 14 toyota         34
## 15 volkswagen     27
```

```
levels.country <- tribble( # table of: company & country of origin
  ~company,    ~country,
   "audi",           "Germany",
   "chevrolet",      "USA",
   "dodge",          "USA",
   "ford",           "USA",
   "honda",          "Japan",
   "hyundai",        "South Korea",
   "jeep",           "USA",
   "land rover",     "England",
   "lincoln",        "USA",
   "mercury",        "USA",
   "nissan",        "Japan",
   "pontiac",        "USA",
   "subaru",         "Japan",
   "toyota",         "Japan",
   "volkswagen",     "Germany")

# Prepare pairs for re-coding factor levels
levels.country %>%
  mutate(recode = str_c(country, " = ", "'", company, "'", sep = "")) %>%
  pull(recode) %>%
  str_c(., collapse = ", ")
```

```
## [1] "Germany = 'audi', USA = 'chevrolet', USA = 'dodge', USA = 'ford', Japan = 'honda', South Korea =
```

```
# Now re-code factor levels: companies -> countries
df.recode <- df %>%
  mutate(manufacturer = fct_recode(manufacturer,
                                   Germany = 'audi',
                                   USA = 'chevrolet',
                                   USA = 'dodge',
                                   USA = 'ford',
                                   Japan = 'honda',
                                   `South Korea` = 'hyundai',
                                   USA = 'jeep',
                                   England = 'land rover',
                                   USA = 'lincoln',
                                   USA = 'mercury',
                                   Japan = 'nissan',
                                   USA = 'pontiac',
                                   Japan = 'subaru',
                                   Japan = 'toyota',
                                   Germany = 'volkswagen'))

# Check recoded factor levels
df.recode %>%
  count(manufacturer)
```

```
## # A tibble: 5 x 2
##   manufacturer     n
##   <fct>        <int>
```

```
## 1 Germany          45
## 2 USA             101
## 3 Japan            70
## 4 South Korea      14
## 5 England           4
```

```
# fct_collapse() - Collapse levels into manually defined groups
# Let's keep only USA manufacturers, others are collapsed
non.us.manufacturers <- levels.country %>% filter(country != "USA") %>% pull(company) # vector of non-U

df.collapse <- df %>%
  mutate(manufacturer = fct_collapse(manufacturer, `non US` = non.us.manufacturers))

# Check collapsed factor levels
df.collapse %>%
  count(manufacturer)
```

```
## # A tibble: 8 x 2
##   manufacturer     n
##   <fct>        <int>
## 1 non US         133
## 2 chevrolet       19
## 3 dodge           37
## 4 ford            25
## 5 jeep             8
## 6 lincoln          3
## 7 mercury          4
## 8 pontiac          5
```

```
# fct_other() - Replace levels with other
# All non-US companies -> Other
df.other <- df %>%
  mutate(manufacturer = fct_other(manufacturer, drop = non.us.manufacturers))

# Check other factor levels
df.other %>%
  count(manufacturer)
```

```
## # A tibble: 8 x 2
##   manufacturer     n
##   <fct>        <int>
## 1 chevrolet       19
## 2 dodge           37
## 3 ford            25
## 4 jeep             8
## 5 lincoln          3
## 6 mercury          4
## 7 pontiac          5
## 8 Other          133
```

```
# fct_drop() - Drop factor levels
# Drop other level
# - first filter out rows with "Other"
df.drop <- df.other %>%
  filter(manufacturer != "Other")
```

```r
# Check levels - "Other" still present!
df.drop %>% pull(manufacturer) %>%  fct_unique()
```

```
## [1] chevrolet dodge     ford      jeep      lincoln   mercury   pontiac
## [8] Other
## Levels: chevrolet dodge ford jeep lincoln mercury pontiac Other
```

```r
# Now drop "Other" level with no more data
df.drop <- df.drop %>%
  mutate(manufacturer = fct_drop(manufacturer))

# Check levels - "Other" removed!
df.drop %>% pull(manufacturer) %>%  fct_unique()
```

```
## [1] chevrolet dodge     ford      jeep      lincoln   mercury   pontiac
## Levels: chevrolet dodge ford jeep lincoln mercury pontiac
```

```r
# fct_expand() - Add additional levels to factor
# Lets add some additional manufacturers
df.expand <- df %>%
  mutate(manufacturer = fct_expand(manufacturer, c("Ferrari", "Lamborghini")))

# Check levels - "New levels added
df.expand %>% pull(manufacturer) %>% fct_unique()
```

```
##  [1] audi        chevrolet   dodge       ford        honda       hyundai
##  [7] jeep        land rover  lincoln     mercury     nissan      pontiac
## [13] subaru      toyota      volkswagen  Ferrari     Lamborghini
## 17 Levels: audi chevrolet dodge ford honda hyundai jeep land rover ... Lamborghini
```

```r
df.expand %>% pull(manufacturer) %>% levels()
```

```
##  [1] "audi"        "chevrolet"   "dodge"       "ford"        "honda"
##  [6] "hyundai"     "jeep"        "land rover"  "lincoln"     "mercury"
## [11] "nissan"      "pontiac"     "subaru"      "toyota"      "volkswagen"
## [16] "Ferrari"     "Lamborghini"
```

```r
# But at the moment there aren't any cars from "Ferrari" or "Lamborghini"
# - just levels are prepared in advance
df.expand %>% filter(manufacturer %in% c("Ferrari", "Lamborghini"))
```

```
## # A tibble: 0 x 11
## # i 11 variables: manufacturer <fct>, model <fct>, displ <dbl>, year <int>,
## #   cyl <int>, trans <fct>, drv <chr>, cty <int>, hwy <int>, fl <chr>,
## #   class <fct>
```