Midterm Practice

Aditya Singhania

3/1/2022

These are the packages that I will need for my solutions.

```
library(tidyverse)
library(ggrepel)
library(Hmisc)
```

```
## Warning: package 'Hmisc' was built under R version 4.1.3
```

1 Relation between tibbles, data frames, and ists

1.1 (a) Tibble

```
musicians_tbl <- tibble(
  name = c("Keith", "John", "Paul"),
  band = c("Stones", "Beatles", "Beatles"),
  instrument = c("guitar", "guitar", "bass")
)
```

1.2 (b) Data frame

```
musicians_dfr <- data.frame(
  name = c("Keith", "John", "Paul"),
  band = c("Stones", "Beatles", "Beatles"),
  instrument = c("guitar", "guitar", "bass")
)
```

1.3 (c) List

```
musicians_list <- list(
  name = c("Keith", "John", "Paul"),
  band = c("Stones", "Beatles", "Beatles"),
  instrument = c("guitar", "guitar", "bass")
)
```

1.4 (d) Applying functions to all

```
musicians <- Hmisc::llist(musicians_tbl, musicians_dfr, musicians_list)
sapply(musicians, function(x) {
  list(
    class = class(x),
    is_tib = is_tibble(x),
    is_df = is.data.frame(x),
    is_list = is.list(x)
  )
})
```

```
##         musicians_tbl musicians_dfr musicians_list
## class   character,3   "data.frame"  character,2
## is_tib  TRUE          FALSE         FALSE
## is_df   TRUE          TRUE          FALSE
## is_list TRUE          TRUE          TRUE
```

1.5 (e) Attributes

```
lapply(musicians, attributes)
```

```
## $musicians_tbl
## $musicians_tbl$row.names
## [1] 1 2 3
##
## $musicians_tbl$names
## [1] "name"       "band"       "instrument"
##
## $musicians_tbl$label
## [1] "musicians_tbl"
##
## $musicians_tbl$class
## [1] "tbl_df"     "tbl"        "data.frame"
##
##
## $musicians_dfr
## $musicians_dfr$names
## [1] "name"       "band"       "instrument"
##
## $musicians_dfr$row.names
## [1] 1 2 3
##
## $musicians_dfr$label
## [1] "musicians_dfr"
##
## $musicians_dfr$class
## [1] "data.frame"
##
##
```

```
## $musicians_list
## $musicians_list$names
## [1] "name"       "band"       "instrument"
##
## $musicians_list$label
## [1] "musicians_list"
##
## $musicians_list$class
## [1] "labelled" "list"
```

1.6 (f) Subsetting operations

```
lapply(musicians, attributes)
```

```
## $musicians_tbl
## $musicians_tbl$row.names
## [1] 1 2 3
##
## $musicians_tbl$names
## [1] "name"       "band"       "instrument"
##
## $musicians_tbl$label
## [1] "musicians_tbl"
##
## $musicians_tbl$class
## [1] "tbl_df"     "tbl"        "data.frame"
##
##
## $musicians_dfr
## $musicians_dfr$names
## [1] "name"       "band"       "instrument"
##
## $musicians_dfr$row.names
## [1] 1 2 3
##
## $musicians_dfr$label
## [1] "musicians_dfr"
##
## $musicians_dfr$class
## [1] "data.frame"
##
##
## $musicians_list
## $musicians_list$names
## [1] "name"       "band"       "instrument"
##
## $musicians_list$label
## [1] "musicians_list"
##
## $musicians_list$class
## [1] "labelled" "list"
```

1.7 (g) Remove attributes

```
remove_attr <- function(x) {
  attr(x, "class") <- NULL
  attr(x, "row.names") <- NULL
  # dropping appended "label" attribute
  attr(x,"label") <- NULL
  return(x)
}

musicians_cleaned <- lapply(musicians, remove_attr)

# to check attributes
# lapply(musicians_cleaned,attributes)

# Check whether identical

lapply(musicians_cleaned, function(x) identical(x,musicians_list))
```

```
## $musicians_tbl
## [1] TRUE
##
## $musicians_dfr
## [1] TRUE
##
## $musicians_list
## [1] TRUE
```

1.8 Map-family of functions

```
mtcars
```

```
##                      mpg cyl  disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4           21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4
## Mazda RX4 Wag       21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4
## Datsun 710          22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1
## Hornet 4 Drive      21.4   6 258.0 110 3.08 3.215 19.44  1  0    3    1
## Hornet Sportabout   18.7   8 360.0 175 3.15 3.440 17.02  0  0    3    2
## Valiant             18.1   6 225.0 105 2.76 3.460 20.22  1  0    3    1
## Duster 360          14.3   8 360.0 245 3.21 3.570 15.84  0  0    3    4
## Merc 240D           24.4   4 146.7  62 3.69 3.190 20.00  1  0    4    2
## Merc 230            22.8   4 140.8  95 3.92 3.150 22.90  1  0    4    2
## Merc 280            19.2   6 167.6 123 3.92 3.440 18.30  1  0    4    4
## Merc 280C           17.8   6 167.6 123 3.92 3.440 18.90  1  0    4    4
## Merc 450SE          16.4   8 275.8 180 3.07 4.070 17.40  0  0    3    3
## Merc 450SL          17.3   8 275.8 180 3.07 3.730 17.60  0  0    3    3
## Merc 450SLC         15.2   8 275.8 180 3.07 3.780 18.00  0  0    3    3
## Cadillac Fleetwood  10.4   8 472.0 205 2.93 5.250 17.98  0  0    3    4
## Lincoln Continental 10.4   8 460.0 215 3.00 5.424 17.82  0  0    3    4
## Chrysler Imperial   14.7   8 440.0 230 3.23 5.345 17.42  0  0    3    4
```

```
## Fiat 128            32.4   4  78.7  66 4.08 2.200 19.47  1  1    4    1
## Honda Civic         30.4   4  75.7  52 4.93 1.615 18.52  1  1    4    2
## Toyota Corolla      33.9   4  71.1  65 4.22 1.835 19.90  1  1    4    1
## Toyota Corona       21.5   4 120.1  97 3.70 2.465 20.01  1  0    3    1
## Dodge Challenger    15.5   8 318.0 150 2.76 3.520 16.87  0  0    3    2
## AMC Javelin         15.2   8 304.0 150 3.15 3.435 17.30  0  0    3    2
## Camaro Z28          13.3   8 350.0 245 3.73 3.840 15.41  0  0    3    4
## Pontiac Firebird    19.2   8 400.0 175 3.08 3.845 17.05  0  0    3    2
## Fiat X1-9           27.3   4  79.0  66 4.08 1.935 18.90  1  1    4    1
## Porsche 914-2       26.0   4 120.3  91 4.43 2.140 16.70  0  1    5    2
## Lotus Europa        30.4   4  95.1 113 3.77 1.513 16.90  1  1    5    2
## Ford Pantera L      15.8   8 351.0 264 4.22 3.170 14.50  0  1    5    4
## Ferrari Dino        19.7   6 145.0 175 3.62 2.770 15.50  0  1    5    6
## Maserati Bora       15.0   8 301.0 335 3.54 3.570 14.60  0  1    5    8
## Volvo 142E          21.4   4 121.0 109 4.11 2.780 18.60  1  1    4    2
```

1.8.1 (ai) Mean

```
map(mtcars, mean)
```

```
## $mpg
## [1] 20.09062
##
## $cyl
## [1] 6.1875
##
## $disp
## [1] 230.7219
##
## $hp
## [1] 146.6875
##
## $drat
## [1] 3.596563
##
## $wt
## [1] 3.21725
##
## $qsec
## [1] 17.84875
##
## $vs
## [1] 0.4375
##
## $am
## [1] 0.40625
##
## $gear
## [1] 3.6875
##
## $carb
## [1] 2.8125
```

1.8.2 (aii) Type

```
map(nycflights13::flights, typeof)
```

```
## $year
## [1] "integer"
##
## $month
## [1] "integer"
##
## $day
## [1] "integer"
##
## $dep_time
## [1] "integer"
##
## $sched_dep_time
## [1] "integer"
##
## $dep_delay
## [1] "double"
##
## $arr_time
## [1] "integer"
##
## $sched_arr_time
## [1] "integer"
##
## $arr_delay
## [1] "double"
##
## $carrier
## [1] "character"
##
## $flight
## [1] "integer"
##
## $tailnum
## [1] "character"
##
## $origin
## [1] "character"
##
## $dest
## [1] "character"
##
## $air_time
## [1] "double"
##
## $distance
## [1] "double"
##
## $hour
```

```
## [1] "double"
##
## $minute
## [1] "double"
##
## $time_hour
## [1] "double"
```

1.8.3 (aiii) Is factor

```
map(forcats::gss_cat, is.factor)
```

```
## $year
## [1] FALSE
##
## $marital
## [1] TRUE
##
## $age
## [1] FALSE
##
## $race
## [1] TRUE
##
## $rincome
## [1] TRUE
##
## $partyid
## [1] TRUE
##
## $relig
## [1] TRUE
##
## $denom
## [1] TRUE
##
## $tvhours
## [1] FALSE
```

1.8.4 (aiv) Unique values

```
map_int(iris, n_distinct)
```

```
## Sepal.Length  Sepal.Width Petal.Length  Petal.Width      Species
##           35           23           43           22            3
```

1.9 (b) pmap

```r
pmap(list(10, 1:10, 2:11), runif)
```

```
## [[1]]
##  [1] 1.240828 1.377730 1.592045 1.478084 1.024099 1.604649 1.494719 1.546647
##  [9] 1.002565 1.913438
##
## [[2]]
##  [1] 2.748966 2.310764 2.630719 2.851406 2.894808 2.655983 2.894873 2.718155
##  [9] 2.776824 2.202624
##
## [[3]]
##  [1] 3.386881 3.604231 3.701149 3.588391 3.664078 3.545919 3.563886 3.942101
##  [9] 3.189142 3.061040
##
## [[4]]
##  [1] 4.720833 4.777610 4.612681 4.950583 4.869083 4.098648 4.677376 4.743332
##  [9] 4.937774 4.443693
##
## [[5]]
##  [1] 5.249371 5.431763 5.220769 5.401037 5.569406 5.050133 5.728483 5.798772
##  [9] 5.297462 5.471958
##
## [[6]]
##  [1] 6.272390 6.814267 6.789839 6.259429 6.845640 6.141238 6.472810 6.469989
##  [9] 6.141488 6.771103
##
## [[7]]
##  [1] 7.976977 7.999241 7.958596 7.781584 7.576240 7.291003 7.463905 7.040751
##  [9] 7.889205 7.208050
##
## [[8]]
##  [1] 8.416139 8.967459 8.141009 8.858236 8.929999 8.027932 8.608841 8.177276
##  [9] 8.759444 8.466034
##
## [[9]]
##  [1] 9.619443 9.541165 9.112623 9.850683 9.289588 9.165385 9.367661 9.278727
##  [9] 9.321573 9.623586
##
## [[10]]
##  [1] 10.23233 10.53518 10.15884 10.59977 10.86870 10.07098 10.47787 10.26605
##  [9] 10.37970 10.82089
```

What does pmap(list(10, 1:10, 2:11), runif) do? Why?

Ans: `pmap()` allows us to specify a single list that will contain all the vectors that we want to supply to any command or function. In this case, pmap allows us to accommodate 3 vectors in a single list that will run with the runif() command, rather than using `runif()` and `map()` on each list 1) 10, 2) 1:10 and 3) 2:11 3 separate times.

2 source:

https://dcl-prog.stanford.edu/purrr-parallel.html#pmap

2.1 (c) `map_dfr beaver1, beaver2`

```
# Original function
map_dfr(
  list(beaver1, beaver2), function(dfr) {
    glm(activ ~ temp, data = dfr, family = binomial) |> pluck(coef)
  }
)
```

```
## # A tibble: 2 x 2
##   `(Intercept)`  temp
##           <dbl> <dbl>
## 1         -557.  15.0
## 2         -551.  14.7
```

```
# Formula
map_dfr(
  list(beaver1, beaver2), ~ {
    glm(activ ~ temp, data = ., family = binomial) |> pluck(coef)
  }
)
```

```
## # A tibble: 2 x 2
##   `(Intercept)`  temp
##           <dbl> <dbl>
## 1         -557.  15.0
## 2         -551.  14.7
```

Describe what this code chunk does.

Ans: 1. `map_dfr()` helps to bind the rows and columns that come from the two lists (beaver1 and beaver2) into a single dataframe.

2. Within the function, `glm()` calls for a generalized linear model to be fit to the dataframe.
3. (activ ~ temp) indicates that the line should be fitted based on the relationship between temperature on activity. There will be two linear models due to it being applied to 2 datasets.
4. (family = binomial) specifies that there is binomial data involved, we can see that for the column `activ`, the values are either 0 (indicating no activity) or 1 (indicating activity).
5. The coefficients of the two lines from beaver1 and beaver2 are retrieved, which includes the intercept in the 1st column and the gradient in the second column (temp).

https://purrr.tidyverse.org/reference/map.html https://data.princeton.edu/r/glms

3 Tibbles with list columns

```
data(starwars)
```

3.1 (a) Names of all list columns

```
starwars |>
  # Predicate functions must be wrapped in `where()`
  select(where(is.list)) |>
  names()
```

```
## [1] "films"     "vehicles"  "starships"
```

3.2 (b) Milenium Falcon

```
starwars |>
  select(name, starships) |>
  filter(map_lgl(starships, ~ "Millennium Falcon" %in% .)) |>
  select(name)
```

```
## # A tibble: 4 x 1
##   name
##   <chr>
## 1 Chewbacca
## 2 Han Solo
## 3 Lando Calrissian
## 4 Nien Nunb
```

```
# Alternatively, and perhaps, more succintly
starwars$name[map_lgl(starwars$starships, ~ "Millennium Falcon" %in% .)]
```

```
## [1] "Chewbacca"        "Han Solo"         "Lando Calrissian" "Nien Nunb"
```

3.3 (c) Unique Films

```
unique(unlist(starwars$films))
```

```
## [1] "The Empire Strikes Back" "Revenge of the Sith"
## [3] "Return of the Jedi"      "A New Hope"
## [5] "The Force Awakens"       "Attack of the Clones"
## [7] "The Phantom Menace"
```

3.4 (d) Feminine characters

```
starwars_fem_percent <- unnest_longer(starwars, films) |>
  filter(!is.na(gender)) |>
  group_by(films) |>
  summarise(
    n_male = sum(gender == "masculine"),
```

```
    n_female = sum(gender == "feminine"),
    fem_percent = (n_female / (n_male + n_female)) * 100
  )
```

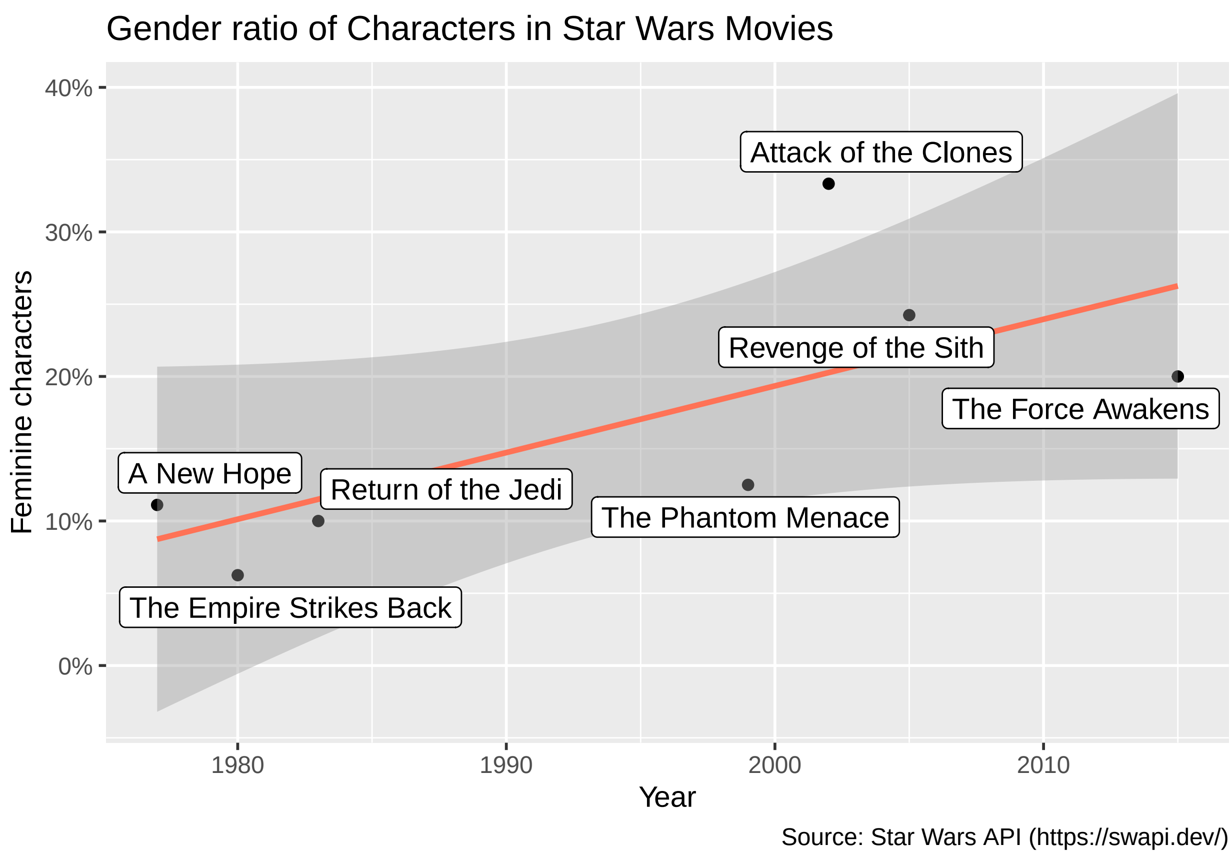
https://stackoverflow.com/questions/41803446/group-data-frame-by-elements-from-a-variable-containing-lists-of-elements

3.5 (e) Plot feminine percentage by move release date

```
release <- tribble(
  ~movies, ~year,
  "A New Hope", 1977,
  "The Empire Strikes Back", 1980,
  "Return of the Jedi", 1983,
  "The Phantom Menace", 1999,
  "Attack of the Clones", 2002,
  "Revenge of the Sith", 2005,
  "The Force Awakens", 2015
)

fem_by_year <-
  left_join(starwars_fem_percent, release, by = c("films" = "movies"))

ggplot(fem_by_year, aes(year, fem_percent, label = films)) +
  geom_point() +
  # fitting regression line
  stat_smooth(method = "lm", col = "coral1") +
  geom_label_repel() +
  labs(
    y = "Feminine characters",
    x = "Year",
    title = "Gender ratio of Characters in Star Wars Movies",
    caption = "Source: Star Wars API (https://swapi.dev/)"
  ) +
  # default scales::percent() multiplies its input value by 100, manually input scale value = 1
  scale_y_continuous(labels = scales::percent_format(scale = 1))
```



Source: https://thomasadventure.blog/posts/ggplot2-percentage-scale/

3.6 (f) Comments

Ans: There is a positive trend in the percentage of feminine characters appearing onscreen with each Star Wars Movie released through the years from 1977 (when a New Hope is released) to 2015 (The Force Awakens). We see one partciular outlier outside the general linear model which is from Attack of the Clones, where 33.3% of the characters were feminine characters.