Coding task 12.03.21 - Looping over lists

Lists in R

- in lists you can sum up all other types of data in R
- e.g a list of data frames, a list of vectors, a list of lists
- the list items do not have to have the same dimensions
- very usefull if you have two similar data sets, that you want to treat the same way (e.g two conditions/samples)

```
library(purrr)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
# example data
data("beavers")
data("airquality")
data("mdeaths")
```

```
## Warning in data("mdeaths"): data set 'mdeaths' not found
```

```
# we have data from two beavers
# beaver1
# beaver2

# and put both together in a list
beavers <- list(beaver1, beaver2)

# you can give each item within a list a name
names(beavers)<- c("Beaver1", "Beaver2")

# you can access list items via [[]]
head(beavers[[1]])
```

```
##   day time  temp activ
## 1 346  840 36.33     0
## 2 346  850 36.34     0
## 3 346  900 36.35     0
## 4 346  910 36.42     0
## 5 346  920 36.55     0
## 6 346  930 36.69     0
```

```r
# or via the name
head(beavers$Beaver2)
```

```
##   day time  temp activ
## 1 307  930 36.58     0
## 2 307  940 36.73     0
## 3 307  950 36.93     0
## 4 307 1000 37.15     0
## 5 307 1010 37.23     0
## 6 307 1020 37.24     0
```

Task 1: Build a List containing the beaver list and the airquality data set and maybe a random word of you choice.

```r
l <- list(beavers, airquality, "Hello List")
names(l)<- c("A", "B", "C")
```

Loops in R

If you want to loop over lists it is of course helpfull is they have a similar layout. Here are three ways to loop over a list:

For Loops

The easiest loop in R is s for loop. (However, for loops are often the most coding intensive and slowest type of loops.) Here is an example on how to use a for loop on a data set

```r
# initialise a variable, where you want to save the results from the loop
mean_temp <- vector()

for(i in 1:length(beavers)) # here you specify the loop variable and how often the loop is run (from 1
  {
  mean_temp[i] <- mean(beavers[[i]]$temp)
}


# note: of course with two you could still calcualte it seperatly, but you might have 10 beavers :)
```

Apply, sapply, lapply

These are base R functionalities that are faster and less text intensive then for loops. Their differ in thei output type: - lapply return a list - sapply returns a vector - apply returns a dataframe

```r
# lapply, sapply
# takes as input a list and a function of what to do with each list element, here the function variable
mean_temp2_list <- lapply(beavers, function(x) mean(x$temp))
mean_temp2_vector <- sapply(beavers, function(x) mean(x$temp))

# apply cannot be used directly on the list but needs a data frame
# but you can loop over all columns at the same time
# you can specify 1 for apply this function rowwise or 2 for column wise
all_means_beaver1 <- apply(beaver1, 2, mean)

# you can combine laply an apply to get loop over all columns of all list menbers
all_mean_both_beavers <- lapply(beavers, apply,2, mean)
```

purrr::map

This is a tidyverse alternative to loop over lists.

```
library(purrr)

# map marks the function by the ~ sign and within the function the variable is always caled .x
map(beavers, ~mean(.x$temp))
beavers %>% map( ~mean(.x$temp))

# you can again change what kind of output you want to get
# a numeric vector
map_dbl(beavers, ~mean(.x$temp))

# a character vector
map_chr(beavers, ~mean(.x$temp))

# map allows you to use dplyr within each loop
library(dplyr) ## You made a nice task on dplyr if you are not familiar with it, it might be worth look
map(beavers, ~mutate(.x, hour = substring(as.character(time),1,1)))

# and to use pipes within the loop
map(beavers, ~mutate(.x, hour = substring(as.character(time),1,1)) %>%
     arrange(hour))
```

Try to solve these tasks with all three methods

Task 2 Calucalte the sd of the 2 columns of the 2 dataframes.

```
###################
# for loop
##################
sd_temp <- vector()

for(i in 1:length(beavers))
  {
  sd_temp[i] <- sd(beavers[[i]]$temp)
}

sd_temp
```

```
## [1] 0.1934217 0.4467889
```

```
################
# sapply
################
sd_temp2 <- lapply(beavers, function(x) sd(x$temp))
sd_temp2
```

```
## $Beaver1
## [1] 0.1934217
##
## $Beaver2
## [1] 0.4467889
```

```
##################
# map
```

```
#################
sd_temp3 <- map_dbl(beavers, ~sd(.x$temp))
sd_temp3
```

```
##   Beaver1   Beaver2
## 0.1934217 0.4467889
```

Task 2b Add up the time and the temperature.

```
###################
# for loop
#################
add <- vector()

for(i in 1:length(beavers))
  {
  add[i] <- beavers[[i]]$temp + beavers[[i]]$time
}
```

```
## Warning in add[i] <- beavers[[i]]$temp + beavers[[i]]$time: number of items to
## replace is not a multiple of replacement length

## Warning in add[i] <- beavers[[i]]$temp + beavers[[i]]$time: number of items to
## replace is not a multiple of replacement length
```

```
add
```

```
## [1] 876.33 966.58
```

```
#################
# sapply
#################
add2 <- lapply(beavers, function(x) x$temp = x$time)
add2
```

```
## $Beaver1
##   [1]  840  850  900  910  920  930  940  950 1000 1010 1020 1030 1040 1050 1100
##  [16] 1110 1120 1130 1140 1150 1200 1210 1220 1230 1240 1250 1300 1310 1320 1330
##  [31] 1340 1350 1400 1410 1420 1430 1440 1450 1500 1510 1520 1530 1540 1550 1600
##  [46] 1610 1620 1630 1640 1650 1700 1710 1720 1730 1740 1750 1800 1810 1820 1830
##  [61] 1840 1850 1900 1910 1920 1930 1940 1950 2000 2010 2020 2030 2040 2050 2100
##  [76] 2110 2120 2130 2140 2150 2200 2210 2230 2240 2250 2300 2310 2320 2330 2340
##  [91] 2350    0   10   20   30   40   50  100  110  120  130  140  150  200  210
## [106]  220  230  240  250  300  310  320  330  340
##
## $Beaver2
##   [1]  930  940  950 1000 1010 1020 1030 1040 1050 1100 1110 1120 1130 1140 1150
##  [16] 1200 1210 1220 1230 1240 1250 1300 1310 1320 1330 1340 1350 1400 1410 1420
##  [31] 1430 1440 1450 1500 1510 1520 1530 1540 1550 1600 1610 1620 1630 1640 1650
##  [46] 1700 1710 1720 1730 1740 1750 1800 1810 1820 1830 1840 1850 1900 1910 1920
##  [61] 1930 1940 1950 2000 2010 2020 2030 2040 2050 2100 2110 2120 2130 2140 2150
##  [76] 2200 2210 2220 2230 2240 2250 2300 2310 2320 2330 2340 2350    0   10   20
##  [91]   30   40   50  100  110  120  130  140  150  200
```

```
#################
# map
#################
add3 <- map(beavers, ~ (.x$temp + .x$time))
```

```
add3
```

```
## $Beaver1
##   [1]  876.33  886.34  936.35  946.42  956.55  966.69  976.71  986.75 1036.81
##  [10] 1046.88 1056.89 1066.91 1076.85 1086.89 1136.89 1146.67 1156.50 1166.74
##  [19] 1176.77 1186.76 1236.78 1246.82 1256.89 1266.99 1276.92 1286.99 1336.89
##  [28] 1346.94 1356.92 1366.97 1376.91 1386.79 1436.77 1446.69 1456.62 1466.54
##  [37] 1476.55 1486.67 1536.69 1546.62 1556.64 1566.59 1576.65 1586.75 1636.80
##  [46] 1646.81 1656.87 1666.87 1676.89 1686.94 1736.98 1746.95 1757.00 1767.07
##  [55] 1777.05 1787.00 1836.95 1847.00 1856.94 1866.88 1876.93 1886.98 1936.97
##  [64] 1946.85 1956.92 1966.99 1977.01 1987.10 2037.09 2047.02 2056.96 2066.84
##  [73] 2076.87 2086.85 2136.85 2146.87 2156.89 2166.86 2176.91 2187.53 2237.23
##  [82] 2247.20 2267.25 2277.20 2287.21 2337.24 2347.10 2357.20 2367.18 2376.93
##  [91] 2386.83   36.93   46.83   56.80   66.75   76.71   86.73  136.75  146.72
## [100]  156.76  166.70  176.82  186.88  236.94  246.79  256.78  266.80  276.82
## [109]  286.84  336.86  346.88  356.93  366.97  377.15
##
## $Beaver2
##   [1]  966.58  976.73  986.93 1037.15 1047.23 1057.24 1067.24 1076.90 1086.95
##  [10] 1136.89 1146.95 1157.00 1166.90 1176.99 1186.99 1237.01 1247.04 1257.04
##  [19] 1267.14 1277.07 1286.98 1337.01 1346.97 1356.97 1367.12 1377.13 1387.14
##  [28] 1437.15 1447.17 1457.12 1467.12 1477.17 1487.28 1537.28 1547.44 1557.51
##  [37] 1567.64 1577.51 1587.98 1638.02 1648.00 1658.24 1668.10 1678.24 1688.11
##  [46] 1738.02 1748.11 1758.01 1767.91 1777.96 1788.03 1838.17 1848.19 1858.18
##  [55] 1868.15 1878.04 1887.96 1937.84 1947.83 1957.84 1967.74 1977.76 1987.76
##  [64] 2037.64 2047.63 2058.06 2068.19 2078.35 2088.25 2137.86 2147.95 2157.95
##  [73] 2167.76 2177.60 2187.89 2237.86 2247.71 2257.78 2267.82 2277.76 2287.81
##  [82] 2337.84 2348.01 2358.10 2368.15 2377.92 2387.64   37.70   47.46   57.41
##  [91]   67.46   77.56   87.55  137.75  147.76  157.73  167.77  178.01  188.04
## [100]  238.07
```

```
add4 <- map(beavers, ~.x %>% mutate(., add = temp + time))
add4 %>% map(~head(.x))
```

```
## $Beaver1
##   day time  temp activ    add
## 1 346  840 36.33     0 876.33
## 2 346  850 36.34     0 886.34
## 3 346  900 36.35     0 936.35
## 4 346  910 36.42     0 946.42
## 5 346  920 36.55     0 956.55
## 6 346  930 36.69     0 966.69
##
## $Beaver2
##   day time  temp activ     add
## 1 307  930 36.58     0  966.58
## 2 307  940 36.73     0  976.73
## 3 307  950 36.93     0  986.93
## 4 307 1000 37.15     0 1037.15
## 5 307 1010 37.23     0 1047.23
## 6 307 1020 37.24     0 1057.24
```

Task 3 Calcluate the rowwise mean (athough its meaningless here).

```
##############
# for loops
############
rowmean <- list()

for(i in 1:length(beavers))  {
      rowmean[[i]] <- list()
  for(j in 1:nrow(beavers[[i]])){
  rowmean[[i]][[j]] <- mean(as.numeric(beavers[[i]][j,]))
}}

rowmean %>% map(~head(.x))
```

```
## [[1]]
## [[1]][[1]]
## [1] 305.5825
##
## [[1]][[2]]
## [1] 308.085
##
## [[1]][[3]]
## [1] 320.5875
##
## [[1]][[4]]
## [1] 323.105
##
## [[1]][[5]]
## [1] 325.6375
##
## [[1]][[6]]
## [1] 328.1725
##
##
## [[2]]
## [[2]][[1]]
## [1] 318.395
##
## [[2]][[2]]
## [1] 320.9325
##
## [[2]][[3]]
## [1] 323.4825
##
## [[2]][[4]]
## [1] 336.0375
##
## [[2]][[5]]
## [1] 338.5575
##
## [[2]][[6]]
## [1] 341.06
```

```
rowmean2 <- list()
for(i in 1:length(beavers)) {
```

```
  rowmean2[[i]] <- rowMeans(beavers[[i]])
}

rowmean2 %>% map(~head(.x))
```

```
## [[1]]
## [1] 305.5825 308.0850 320.5875 323.1050 325.6375 328.1725
##
## [[2]]
## [1] 318.3950 320.9325 323.4825 336.0375 338.5575 341.0600
```

```
###############
# apply
############
rowmean3 <-lapply(beavers, apply, 1, mean)
rowmean3 %>% map(~head(.x))
```

```
## $Beaver1
## [1] 305.5825 308.0850 320.5875 323.1050 325.6375 328.1725
##
## $Beaver2
## [1] 318.3950 320.9325 323.4825 336.0375 338.5575 341.0600
```

```
################
# map
##############

rowmean4 <- beavers %>% map(~mutate(.x, means = rowMeans(.)))
rowmean4 %>% map(~head(.x))
```

```
## $Beaver1
##   day time  temp activ    means
## 1 346  840 36.33     0 305.5825
## 2 346  850 36.34     0 308.0850
## 3 346  900 36.35     0 320.5875
## 4 346  910 36.42     0 323.1050
## 5 346  920 36.55     0 325.6375
## 6 346  930 36.69     0 328.1725
##
## $Beaver2
##   day time  temp activ    means
## 1 307  930 36.58     0 318.3950
## 2 307  940 36.73     0 320.9325
## 3 307  950 36.93     0 323.4825
## 4 307 1000 37.15     0 336.0375
## 5 307 1010 37.23     0 338.5575
## 6 307 1020 37.24     0 341.0600
```

```
# and a somewhat overcomplicated solution: pmap

rowmean5 <- beavers %>% map( ~.x%>%
  mutate(mean_all = pmap_dbl(., function(...) mean(c(...)))))

#  If we want to make use of more than 2 variables (or lists), we can use the pmap function. We need to

rowmean5 %>% map(~head(.x))
```

```
## $Beaver1
##   day time  temp activ mean_all
## 1 346  840 36.33     0 305.5825
## 2 346  850 36.34     0 308.0850
## 3 346  900 36.35     0 320.5875
## 4 346  910 36.42     0 323.1050
## 5 346  920 36.55     0 325.6375
## 6 346  930 36.69     0 328.1725
##
## $Beaver2
##   day time  temp activ mean_all
## 1 307  930 36.58     0 318.3950
## 2 307  940 36.73     0 320.9325
## 3 307  950 36.93     0 323.4825
## 4 307 1000 37.15     0 336.0375
## 5 307 1010 37.23     0 338.5575
## 6 307 1020 37.24     0 341.0600
```

For these tasks choose your favorite method

Task 4 Caluclate the mean temp hourwise (900-950,1000-1050 etc.)

```
# get hour
hourwise_mean_temp <- beaver1 %>% mutate(hour = substring(time,1,1))
head(hourwise_mean_temp)
```

```
##   day time  temp activ hour
## 1 346  840 36.33     0    8
## 2 346  850 36.34     0    8
## 3 346  900 36.35     0    9
## 4 346  910 36.42     0    9
## 5 346  920 36.55     0    9
## 6 346  930 36.69     0    9
```

```
# group by hour and clac mean
hourwise_mean_temp <- beaver1 %>% mutate(hour = substring(time,1,1)) %>%
  group_by(hour)%>%
  summarise(hour_temp <- mean(temp), .groups = "keep")
head(hourwise_mean_temp)
```

```
## # A tibble: 6 x 2
## # Groups:   hour [6]
##   hour  `hour_temp <- mean(temp)`
##   <chr>                     <dbl>
## 1 0                          36.9
## 2 1                          36.8
## 3 2                          37.0
## 4 3                          36.9
## 5 4                          36.7
## 6 5                          36.7
```

```
# with map loop
hourwise_mean_temp <- beavers %>% map(~.x %>% mutate(hour = substring(time,1,1)) %>%
  group_by(hour)%>%
  summarise(hour_temp = mean(temp), .groups = "keep"))
hourwise_mean_temp %>% map(~head(.x))
```

```
## $Beaver1
## # A tibble: 6 x 2
## # Groups:   hour [6]
##   hour  hour_temp
##   <chr>     <dbl>
## 1 0          36.9
## 2 1          36.8
## 3 2          37.0
## 4 3          36.9
## 5 4          36.7
## 6 5          36.7
##
## $Beaver2
## # A tibble: 6 x 2
## # Groups:   hour [6]
##   hour  hour_temp
##   <chr>     <dbl>
## 1 0          37.7
## 2 1          37.5
## 3 2          37.9
## 4 3          37.5
## 5 4          37.6
## 6 5          37.6
```

```
# with lapply
hourwise_mean_temp2 <- lapply(beavers, function(x) x %>% mutate(hour = substring(time,1,1)) %>%
  group_by(hour)%>%
  summarise(hour_temp = mean(temp), .groups = "keep") )

hourwise_mean_temp2 %>% map(~head(.x))
```

```
## $Beaver1
## # A tibble: 6 x 2
## # Groups:   hour [6]
##   hour  hour_temp
##   <chr>     <dbl>
## 1 0          36.9
## 2 1          36.8
## 3 2          37.0
## 4 3          36.9
## 5 4          36.7
## 6 5          36.7
##
## $Beaver2
## # A tibble: 6 x 2
## # Groups:   hour [6]
##   hour  hour_temp
##   <chr>     <dbl>
## 1 0          37.7
## 2 1          37.5
## 3 2          37.9
## 4 3          37.5
## 5 4          37.6
## 6 5          37.6
```

Task 5 Make a new list that contains the beaver data of both beavers only for even days

```
even_beavers <- beavers %>% map(~.x[.x$day %% 2 == 0,])

beavers %>% map(~table(.x$day))
```

```
## $Beaver1
##
## 346 347
##  91  23
##
## $Beaver2
##
## 307 308
##  87  13
```

```
even_beavers %>% map(~table(.x$day))
```

```
## $Beaver1
##
## 346
##  91
##
## $Beaver2
##
## 308
##  13
```

Task 6 Look at the starwars data set column films. This is a list in da data frame! Lets say you want to look at the distribution of skin color and haircolor for the different films. Can you make a list containing one element per film with all characters? How are the distributions filmwise? Make some nice plots (maybe in a loop).

```
library(ggplot2)
head(dplyr::starwars)
```

```
## # A tibble: 6 x 14
##   name  height  mass hair_color skin_color eye_color birth_year sex   gender
##   <chr>  <int> <dbl> <chr>      <chr>      <chr>          <dbl> <chr> <chr>
## 1 Luke~    172    77 blond      fair       blue            19   male  mascu~
## 2 C-3PO    167    75 <NA>       gold       yellow         112   none  mascu~
## 3 R2-D2     96    32 <NA>       white, bl~ red             33   none  mascu~
## 4 Dart~    202   136 none       white      yellow          41.9 male  mascu~
## 5 Leia~    150    49 brown      light      brown           19   fema~ femin~
## 6 Owen~    178   120 brown, gr~ light      blue            52   male  mascu~
## # ... with 5 more variables: homeworld <chr>, species <chr>, films <list>,
## #   vehicles <list>, starships <list>
```

```
# "cheating" with nice tidyverse functions
head(starwars$films)
```

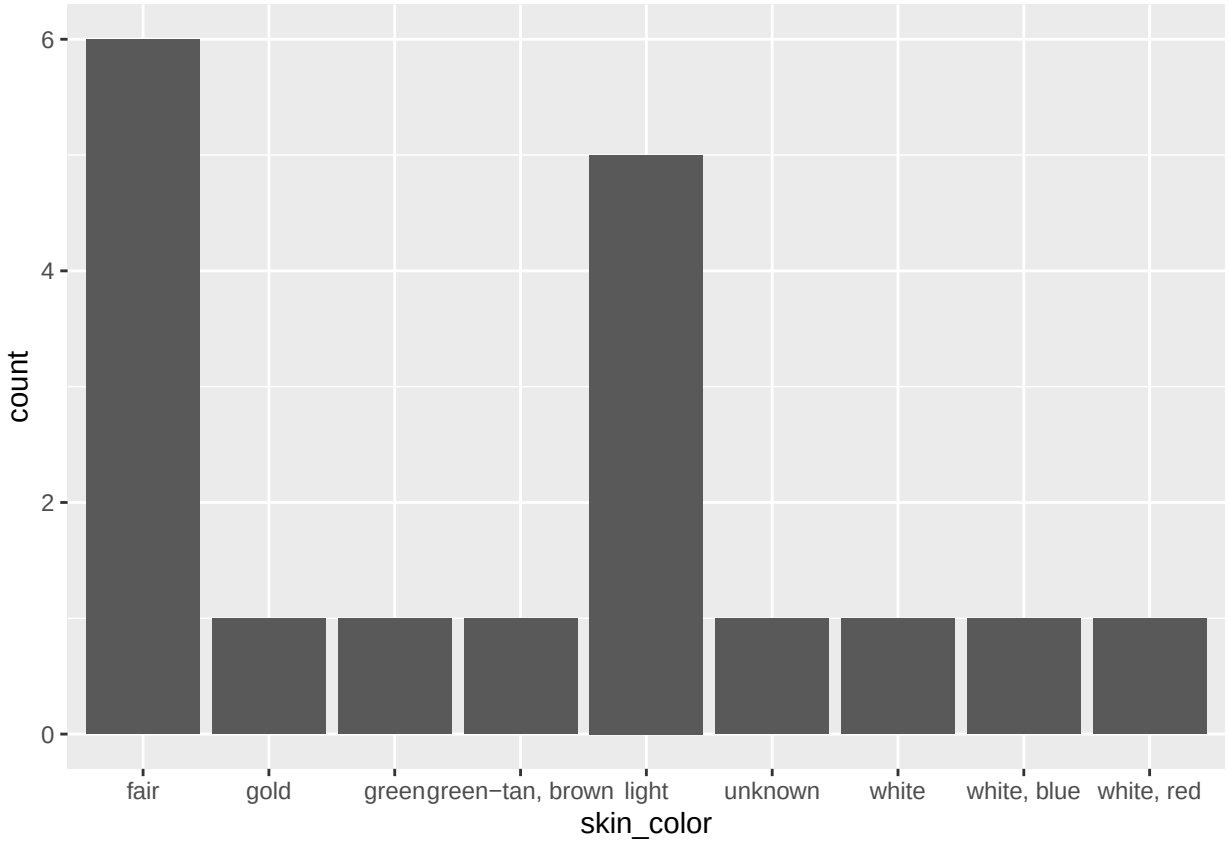
```
## [[1]]
## [1] "The Empire Strikes Back" "Revenge of the Sith"
## [3] "Return of the Jedi"      "A New Hope"
## [5] "The Force Awakens"
##
## [[2]]
## [1] "The Empire Strikes Back" "Attack of the Clones"
```

```
## [3] "The Phantom Menace"      "Revenge of the Sith"
## [5] "Return of the Jedi"      "A New Hope"
##
## [[3]]
## [1] "The Empire Strikes Back" "Attack of the Clones"
## [3] "The Phantom Menace"      "Revenge of the Sith"
## [5] "Return of the Jedi"      "A New Hope"
## [7] "The Force Awakens"
##
## [[4]]
## [1] "The Empire Strikes Back" "Revenge of the Sith"
## [3] "Return of the Jedi"      "A New Hope"
##
## [[5]]
## [1] "The Empire Strikes Back" "Revenge of the Sith"
## [3] "Return of the Jedi"      "A New Hope"
## [5] "The Force Awakens"
##
## [[6]]
## [1] "Attack of the Clones" "Revenge of the Sith"  "A New Hope"
```

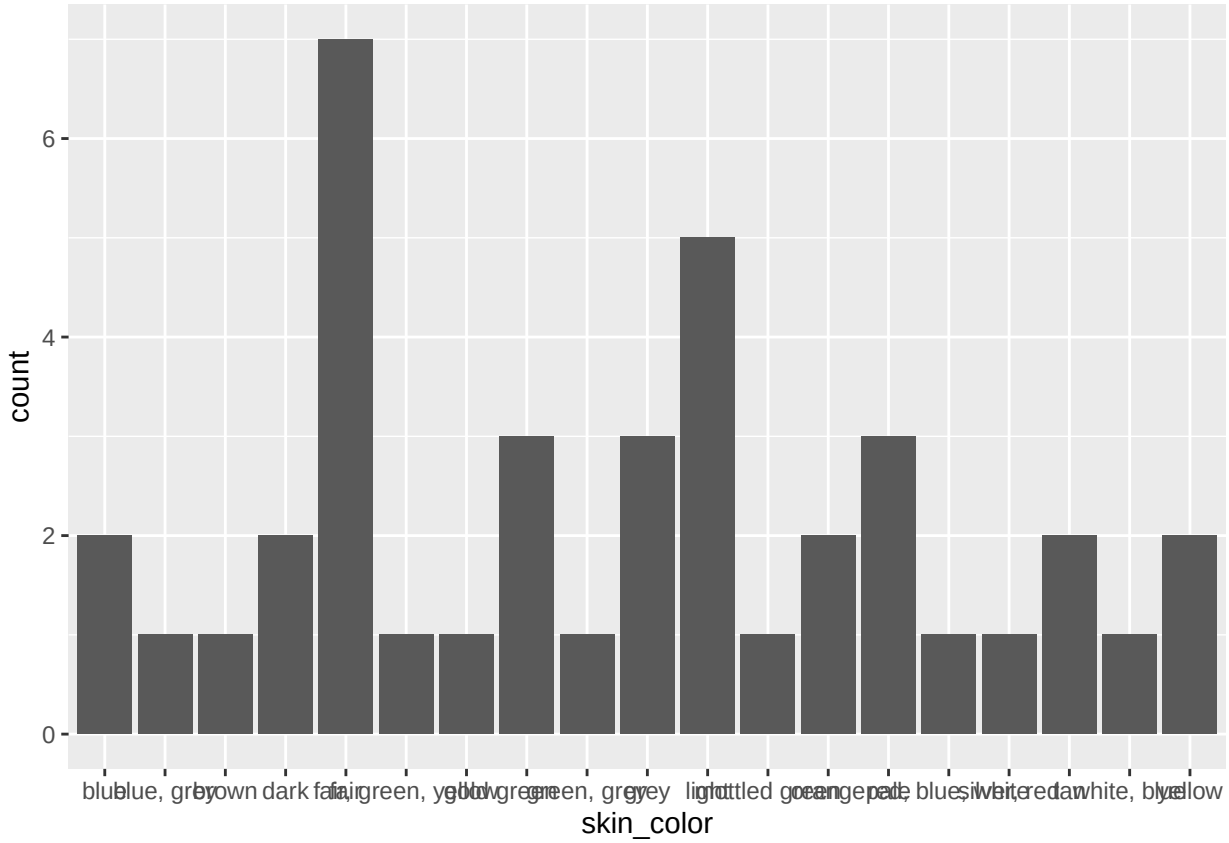
```
starwars_by_films <- starwars %>% tidyr::unnest(films) %>% split(., .$films)


map(starwars_by_films, ~ggplot(.x, aes(x=skin_color))+
      geom_bar())
```

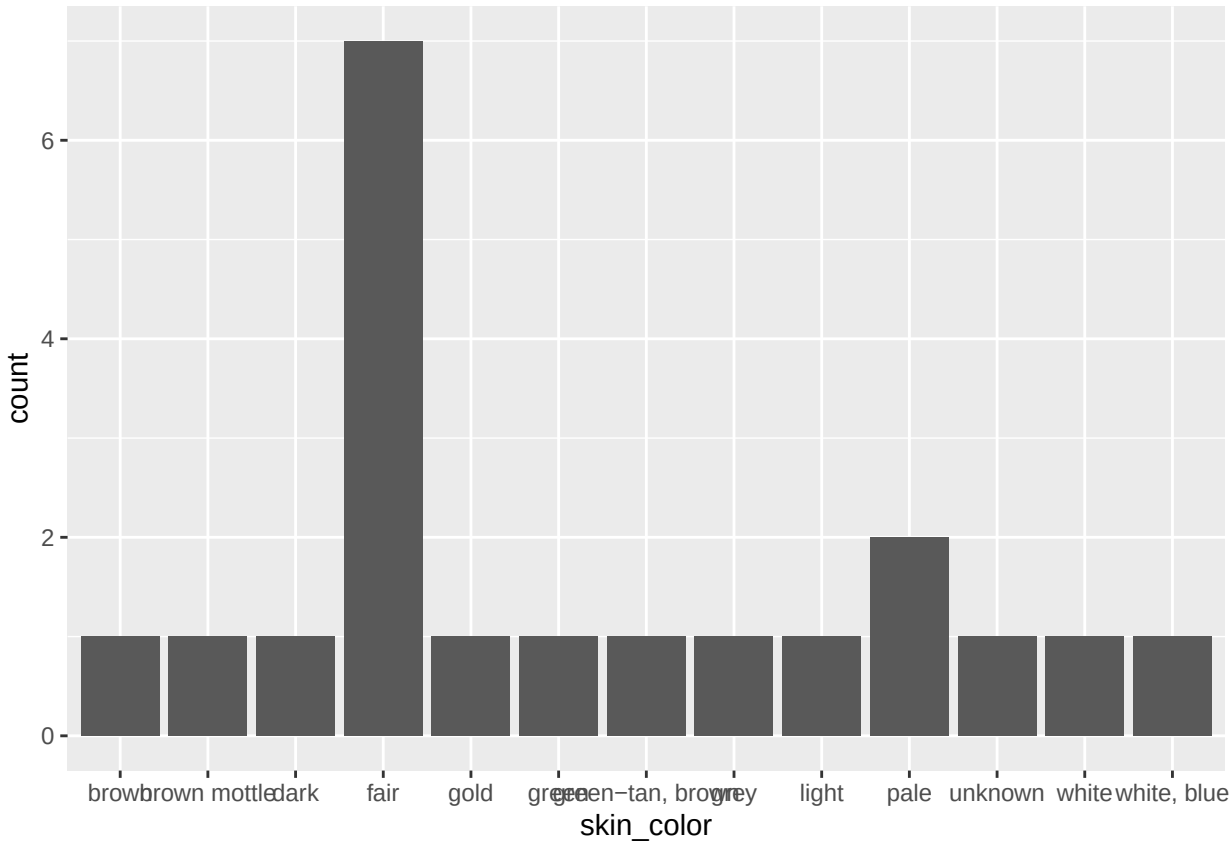
```
## $`A New Hope`
```



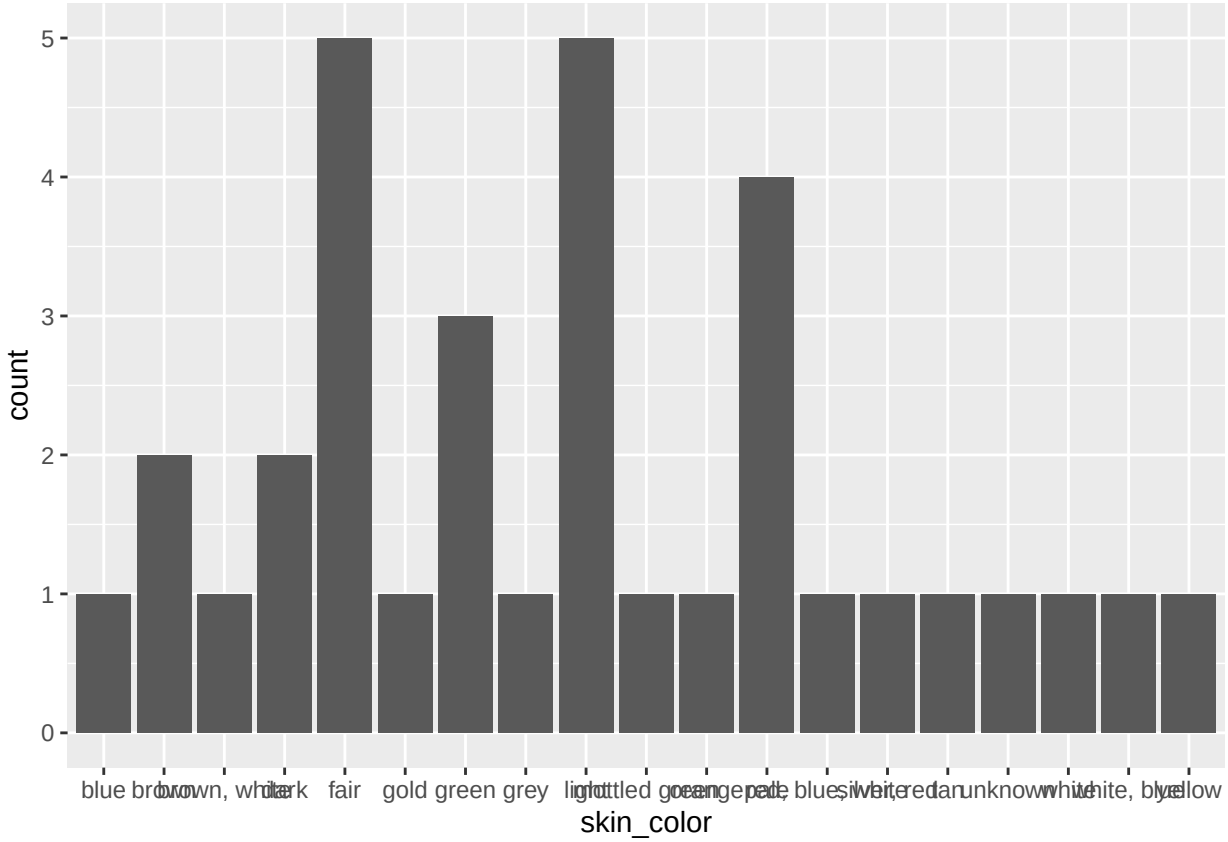
```
##
## $`Attack of the Clones`
```



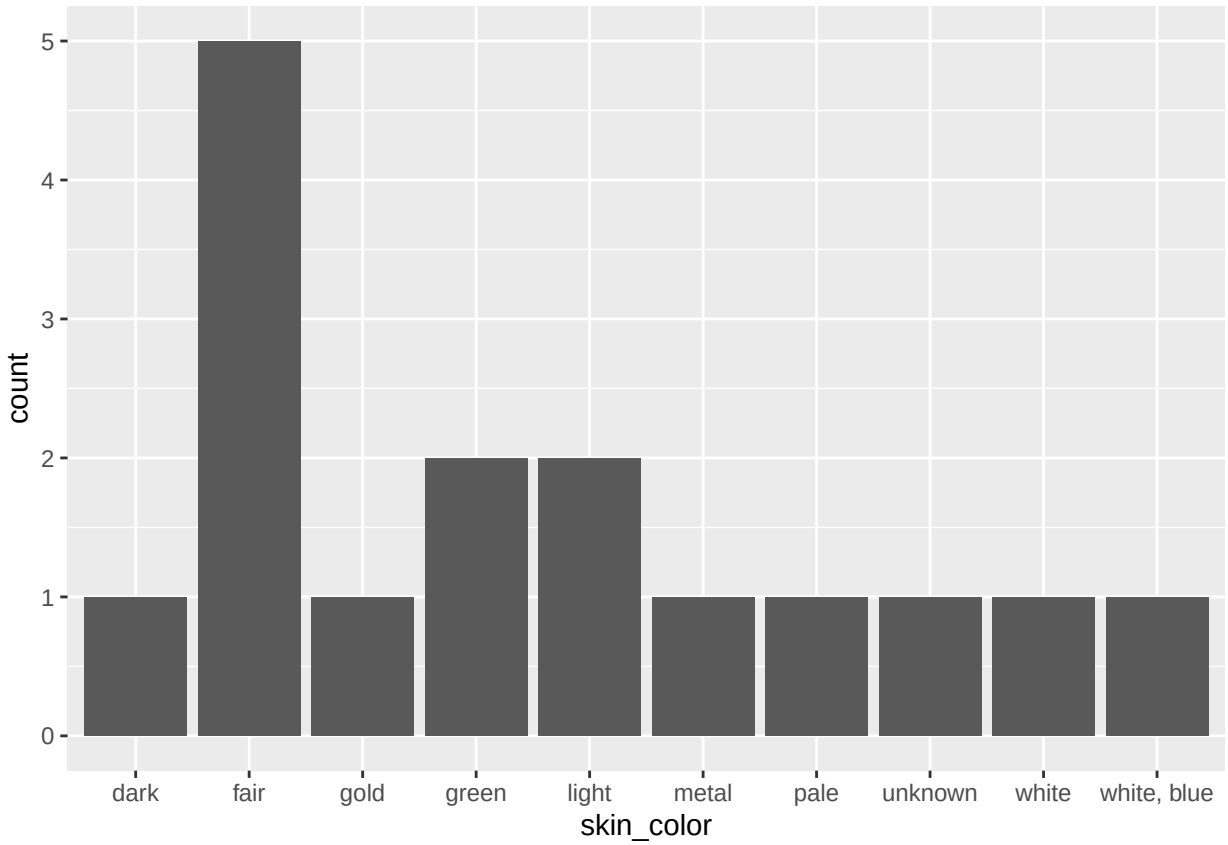

```
##
## $`Return of the Jedi`
```



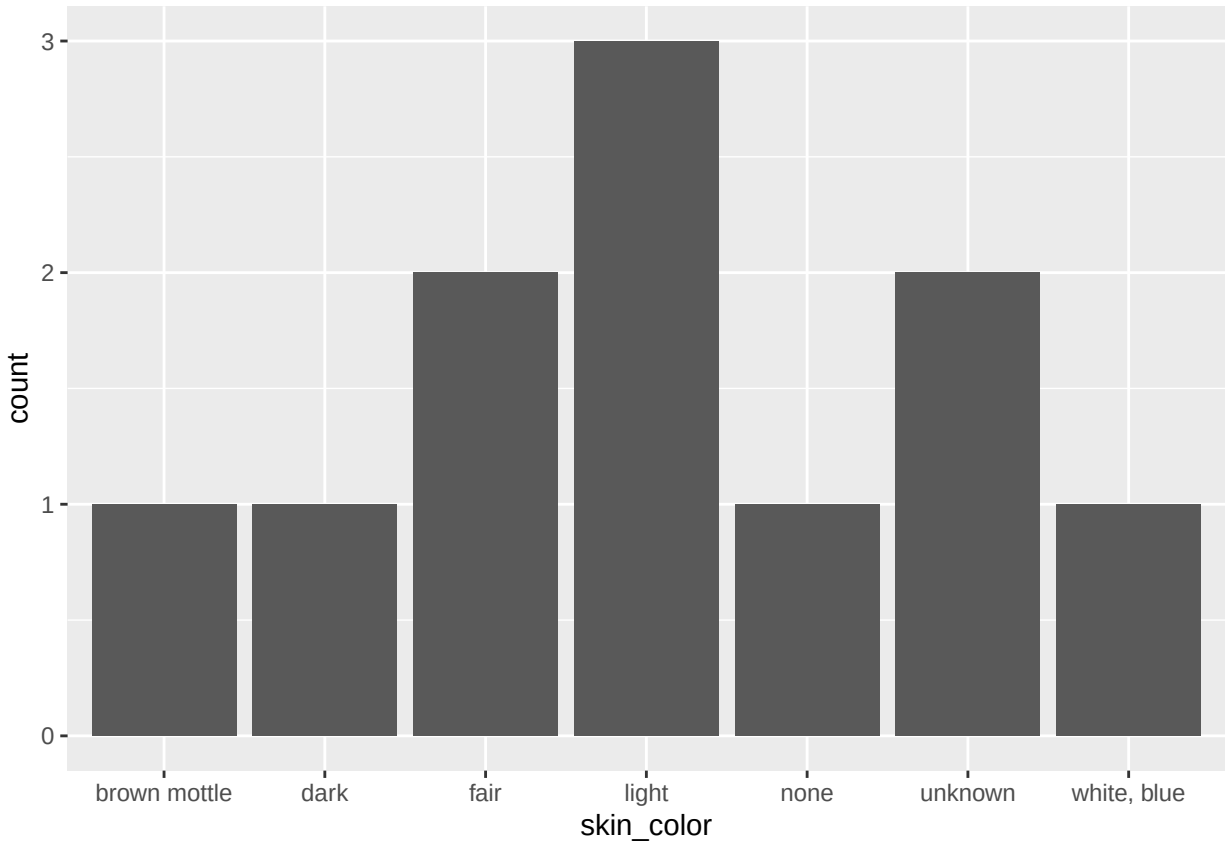

```
##
## $`Revenge of the Sith`
```



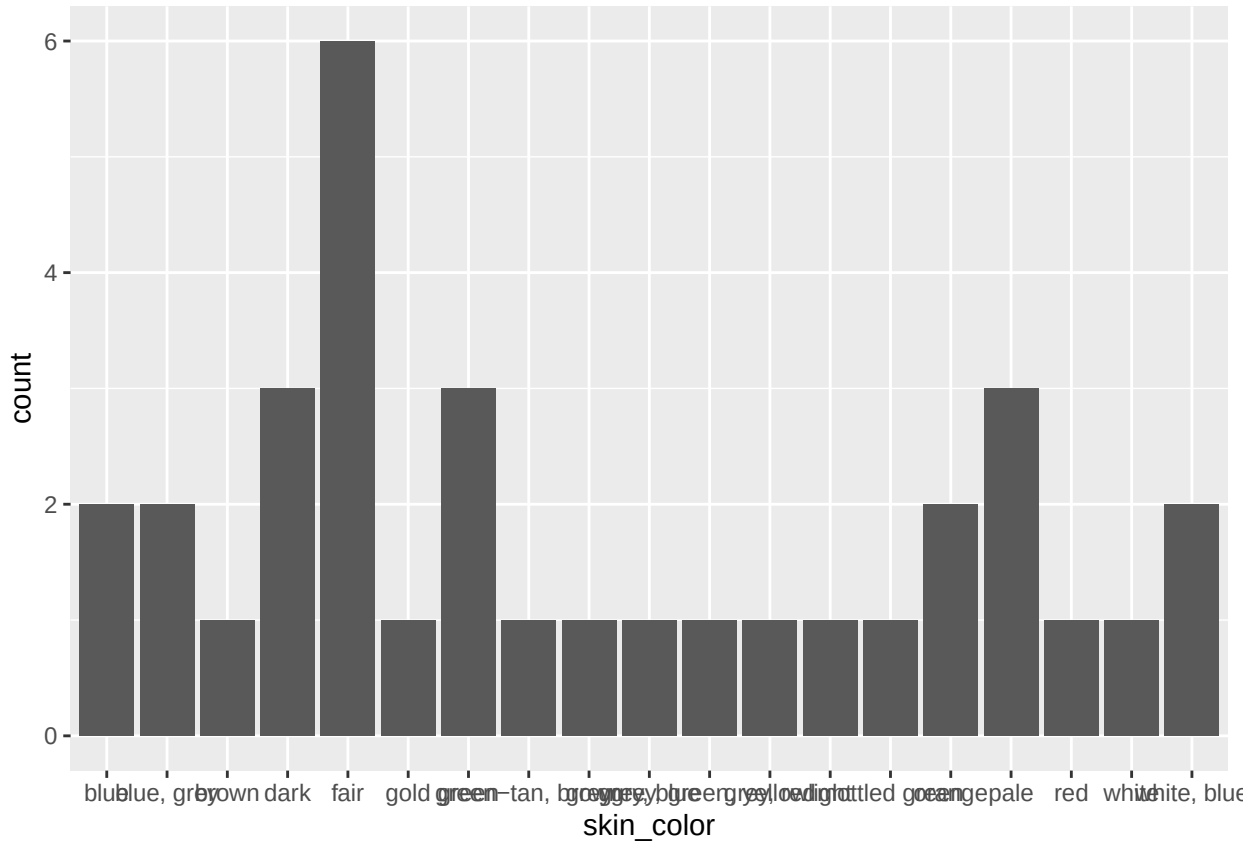

```
##
## $`The Empire Strikes Back`
```




```
##
## $`The Force Awakens`
```

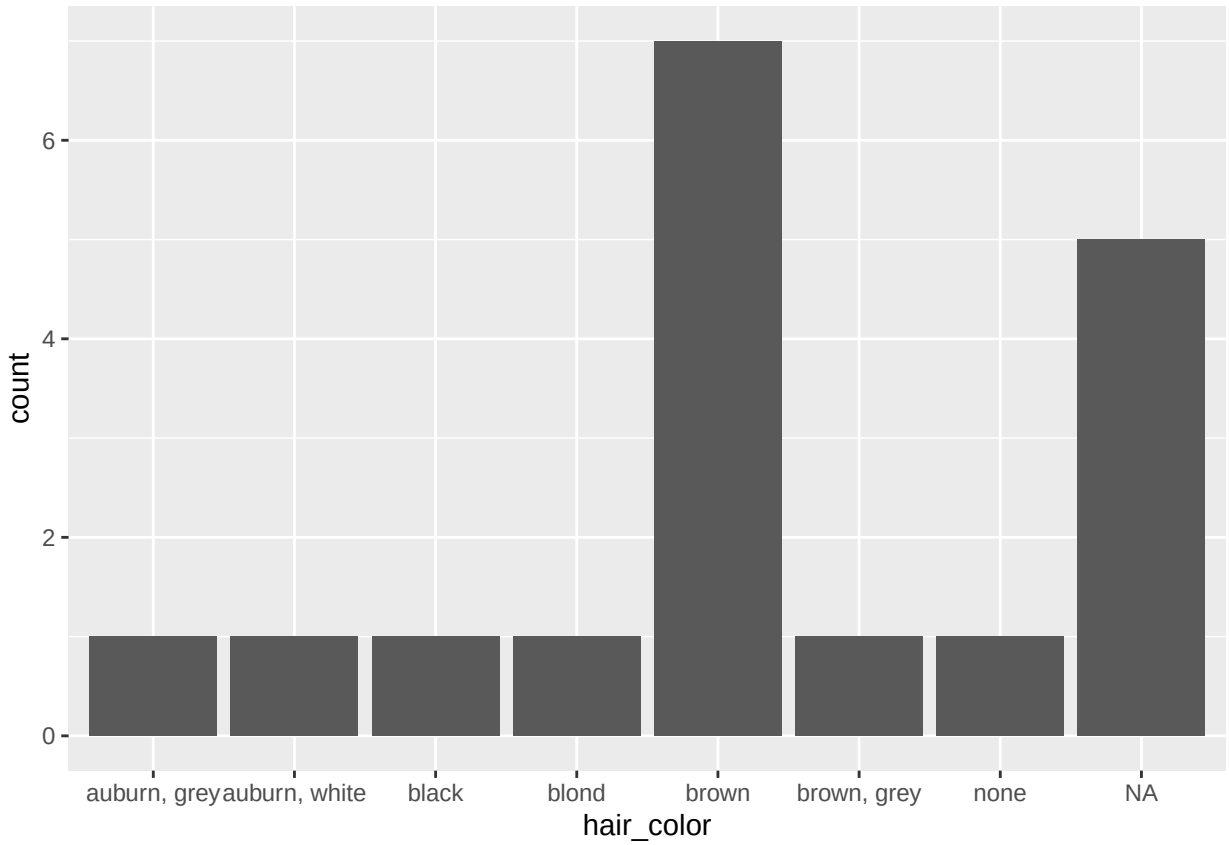



```
##
## $`The Phantom Menace`
```

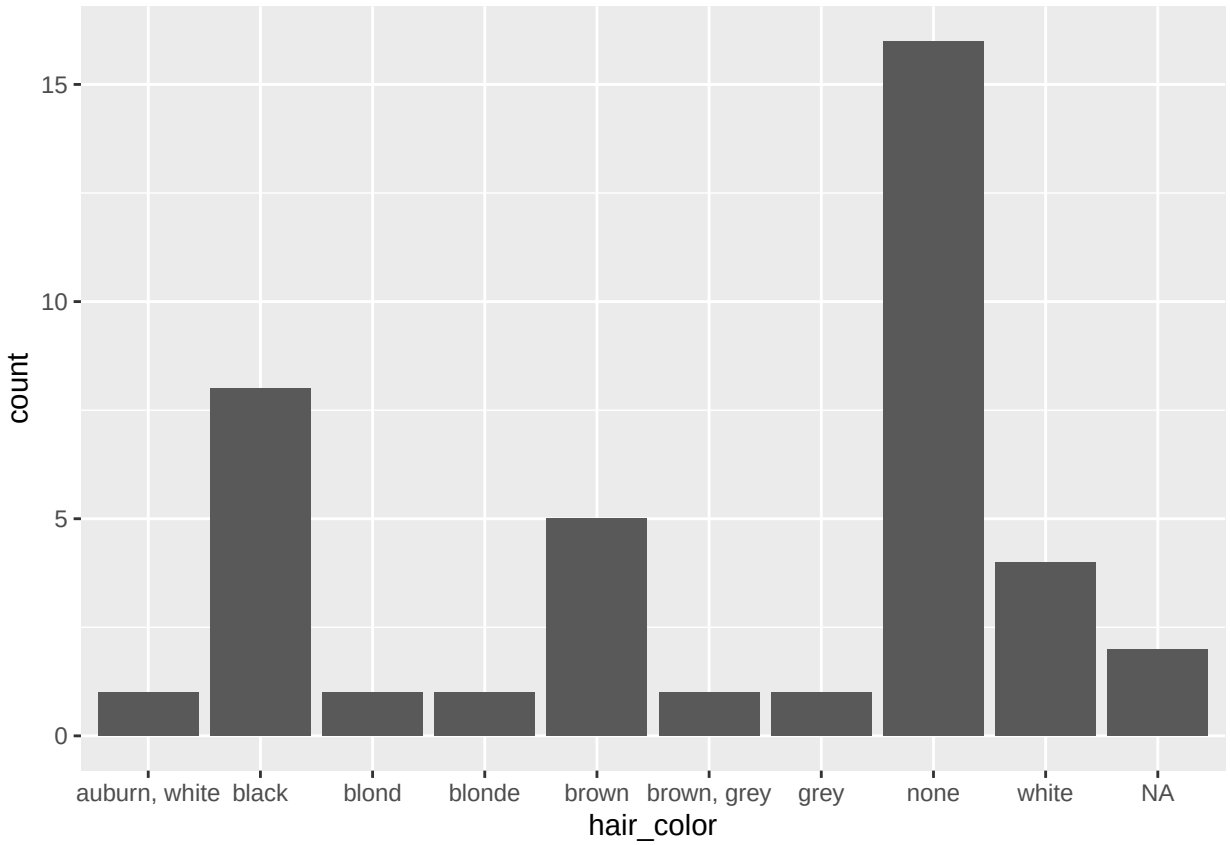



```
map(starwars_by_films, ~ggplot(.x, aes(x=hair_color))+
      geom_bar())
```

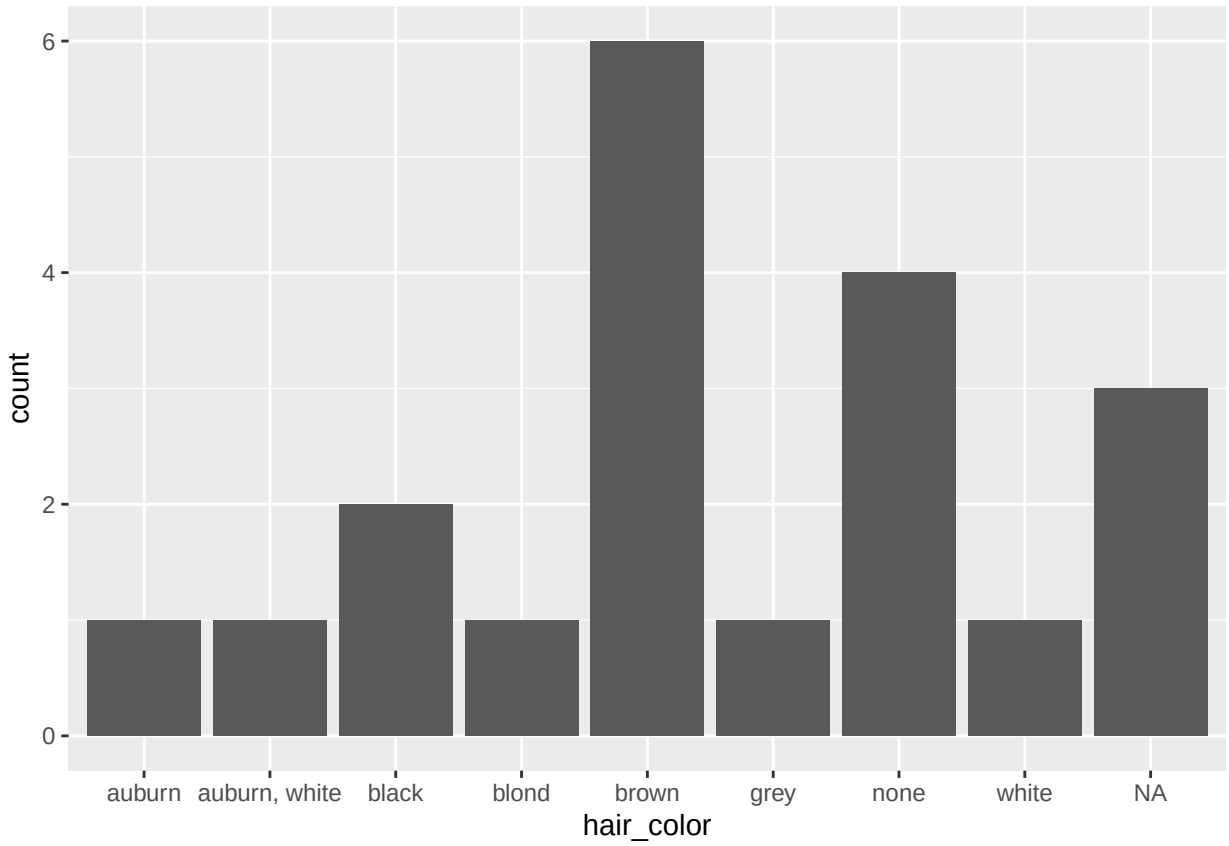
```
## $`A New Hope`
```



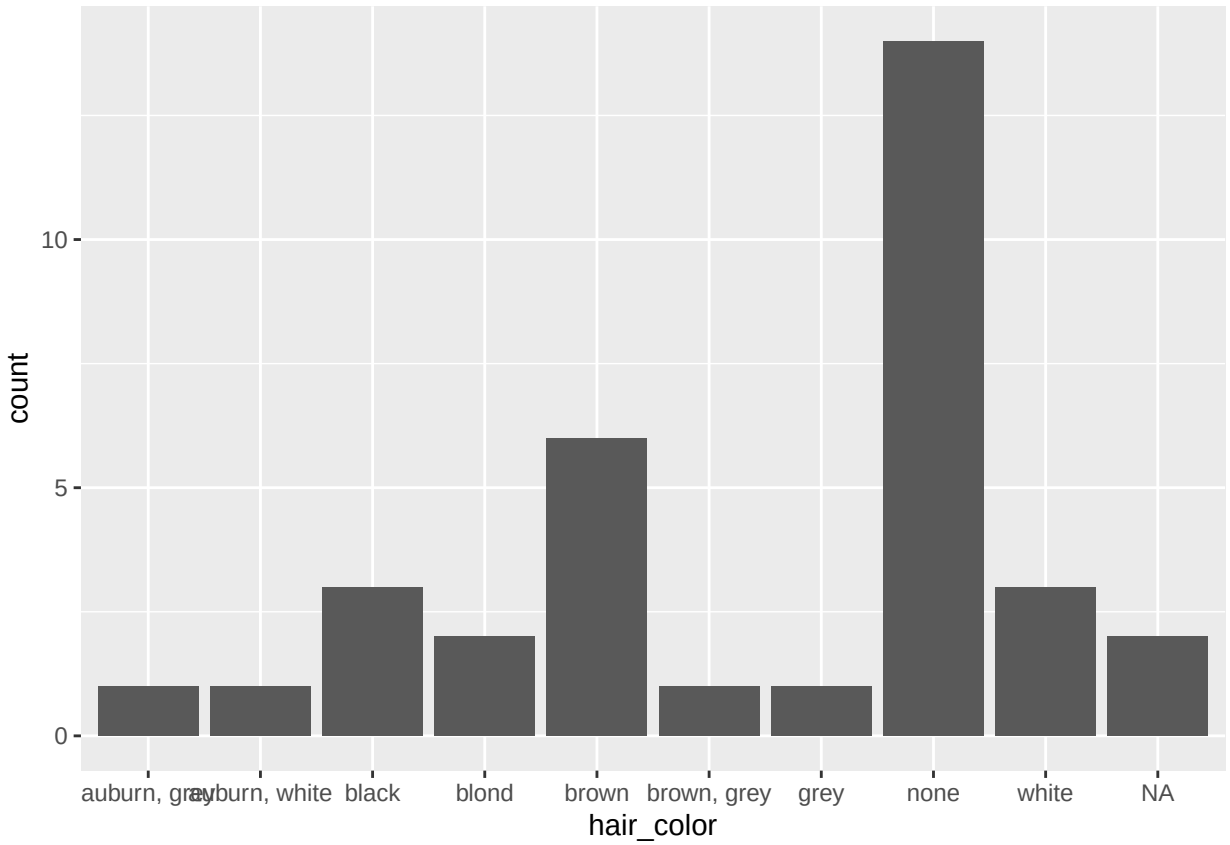

```
##
## $`Attack of the Clones`
```



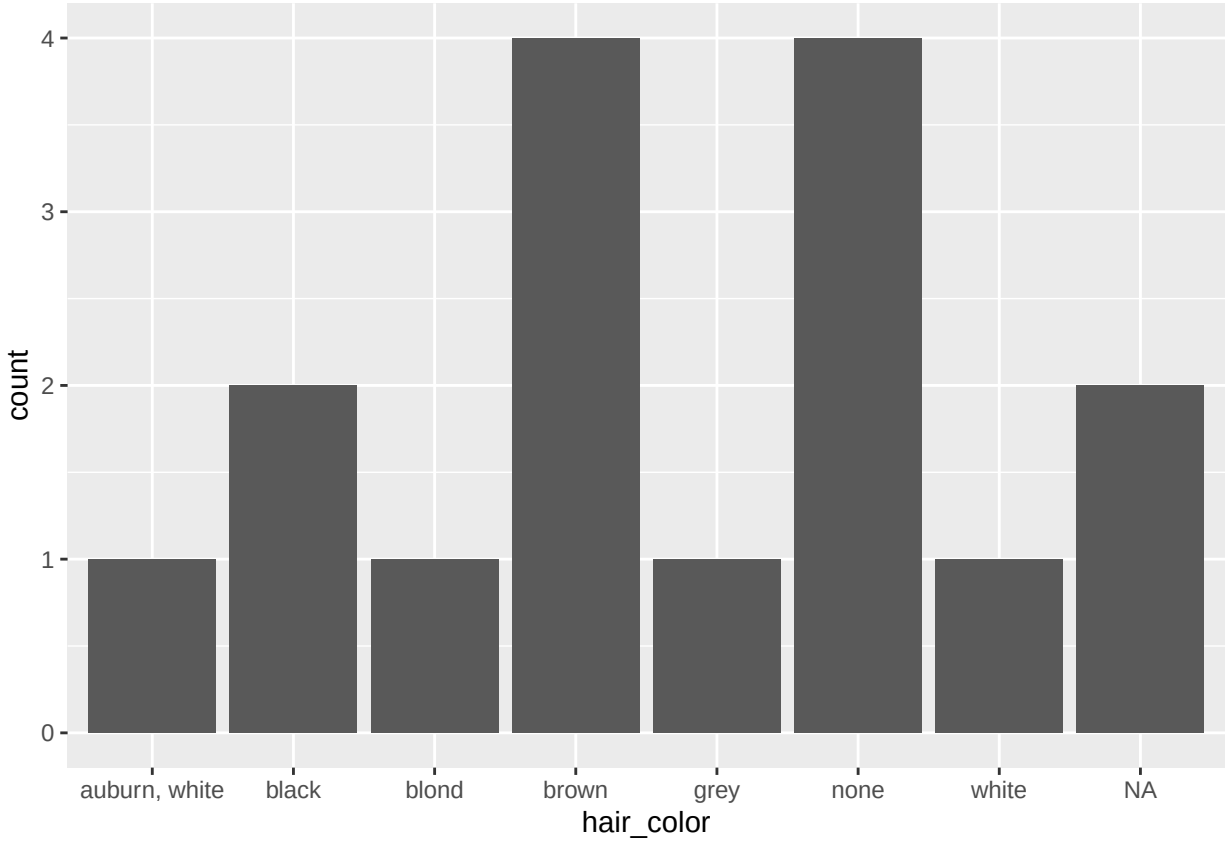

```
##
## $`Return of the Jedi`
```



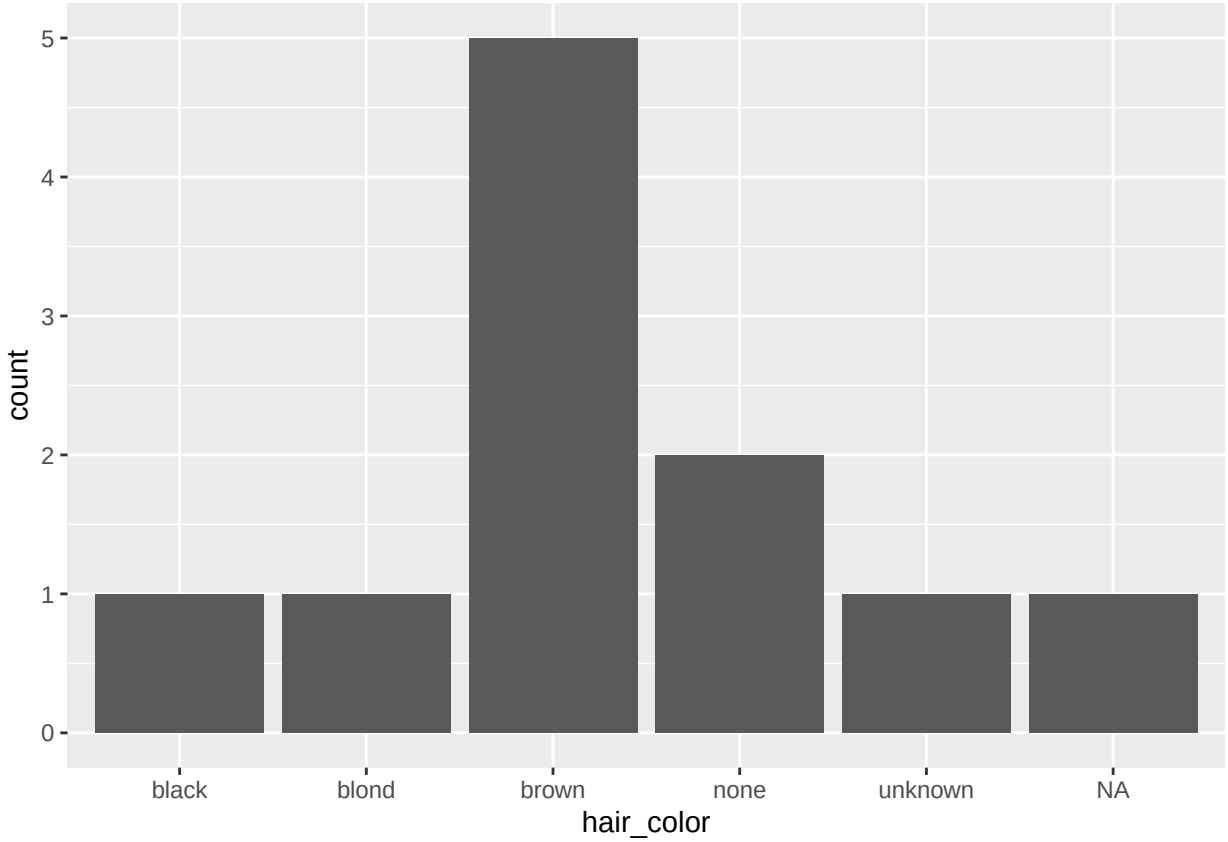

```
##
## $`Revenge of the Sith`
```



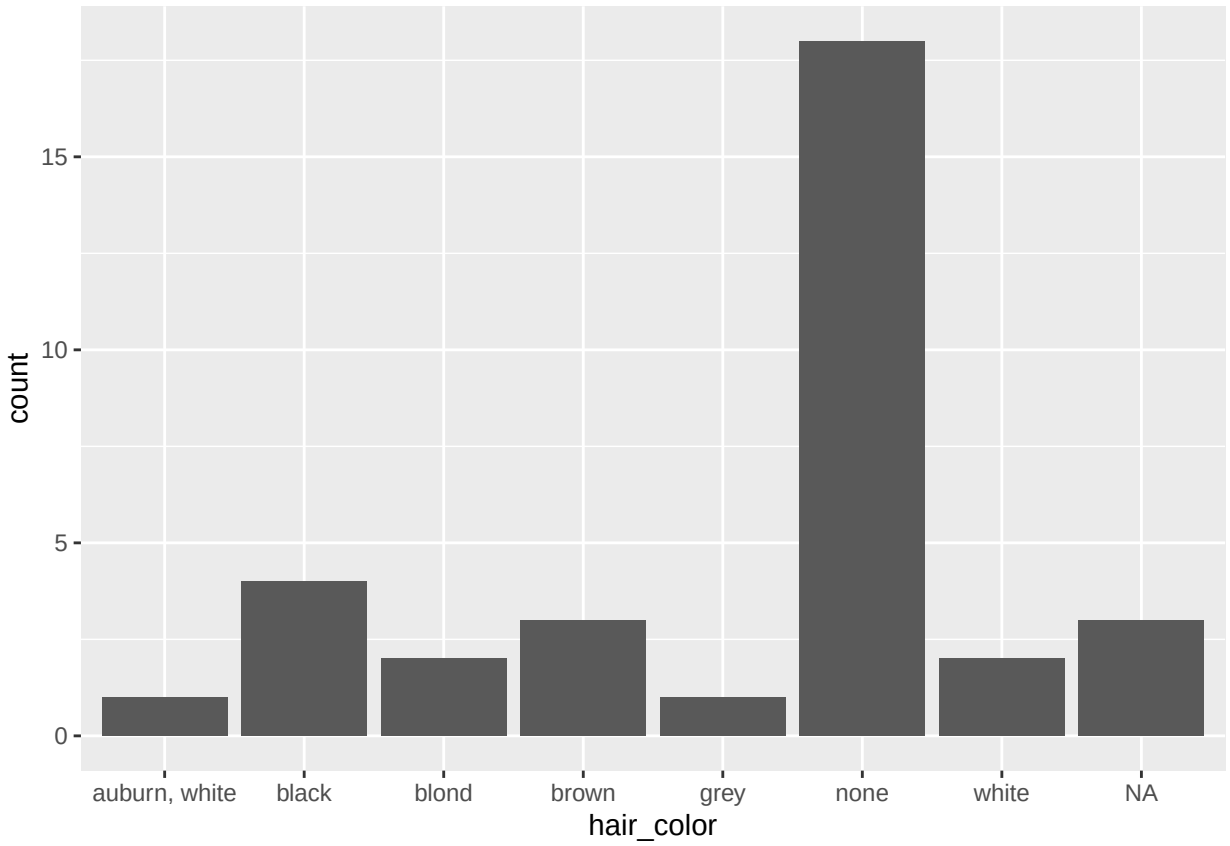

```
##
## $`The Empire Strikes Back`
```




```
##
## $`The Force Awakens`
```




```
##
## $`The Phantom Menace`
```



Benchmark the different options

Task 7 There is a function microbenchmark from the package microbenchmark, which allows you to compare the efficency of different functions. Take one of the tasks before and benchmark the three loop types. Which was the best solution?

```
################
# Microbenchmark Task 3
################

microbenchmark::microbenchmark(for(i in 1:length(beavers))  {
      rowmean[[i]] <- list()
  for(j in 1:nrow(beavers[[i]])){
  rowmean[[i]][[j]] <- mean(as.numeric(beavers[[i]][j,]))
}},

for(i in 1:length(beavers)) {rowmean2[[i]] <- rowMeans(beavers[[i]])},

lapply(beavers, apply, 1, mean),

beavers %>% map(~mutate(.x, means = rowMeans(.))),

beavers %>% map( ~.x%>%
  mutate(mean_all = pmap_dbl(., function(...) mean(c(...)))))

                              )
```

```
## Unit: microseconds
```

```
##
##  for (i in 1:length(beavers)) {     rowmean[[i]] <- list()     for (j in 1:nrow(beavers[[i]])) {
##
##
##
##                                                                                                  b
##        min         lq       mean    median         uq       max neval
##  10824.690 11843.9605 12980.5973 12363.141 13726.7340 26318.558   100
##   1891.910  2084.1280  2278.6686  2174.617  2355.8460  4763.329   100
##    767.509   823.0715   938.6075   878.407   997.1525  1559.901   100
##   2174.093  2383.1400  2695.8945  2584.532  2883.8405  3886.070   100
##   3101.271  3410.9475  3757.6711  3626.602  3992.1360  6201.925   100
```

```
####################
# Microbenchmark Task 4
####################

microbenchmark::microbenchmark(
beavers %>% map(~.x %>% mutate(hour = substring(time,1,1)) %>%
  group_by(hour)%>%
  summarise(hour_temp = mean(temp), .groups = "keep")),

lapply(beavers, function(x) x %>% mutate(hour = substring(time,1,1)) %>%
  group_by(hour)%>%
  summarise(hour_temp = mean(temp), .groups = "keep") ))
```

```
## Unit: milliseconds
##
##                 beavers %>% map(~.x %>% mutate(hour = substring(time, 1, 1)) %>%      group_by(hour)
##  lapply(beavers, function(x) x %>% mutate(hour = substring(time,      1, 1)) %>% group_by(hour) %>% s
##       min       lq      mean   median       uq      max neval
##  8.375459 8.868718 10.396485 9.289992 10.38458 84.26765   100
##  8.295350 8.786746  9.627065 9.203470 10.27234 13.15937   100
```