Assignment 3 Data Wrangling

Yaniv Bronshtein

02/15/2021

Import the libraries

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --

## v ggplot2 3.3.3     v purrr   0.3.4
## v tibble  3.0.5     v dplyr   1.0.3
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.0

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(Lahman)
library(lubridate)
```

```
##
## Attaching package: 'lubridate'

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

```
library(curl)
```

```
##
## Attaching package: 'curl'

## The following object is masked from 'package:readr':
##
##     parse_date
```

```
library(stringr)
```

(1) We would like to create a data frame like the babynames data frame, but for baseball players.

(1a) Use the Master data frame in the Lahman package to create a tibble with exactly the same variables as the babynames data frame, and ordered in the same way. You will need to use the summarize() function to get the counts of each name's use. For year, use the year of birth. For name, use the first name (variable nameFirst). The final table should look like this where prop is the proportion of names in a specific birthyear)

###————————————————

birthYear nameFirst n prop

###————————————————-

Call group by to group master by birthYear and nameFirst Then, pipe the result to summarise() which will return a tibble with birthYear nameFirst, and a column by the name n with the counts of each grouping Pipe the result to mutate() and create a calculated column prop, rounding the value to 3 digits after the decimal Call arrange() to sort by the birthYear ascending

```
Master_t <- as_tibble(Master)
name_by_year <- Master_t %>%
  group_by(birthYear, nameFirst) %>%
  summarise(n = n()) %>%
  mutate(prop = round(n / sum(n), digits = 3)) %>%
  arrange(birthYear)
```

```
## `summarise()` has grouped output by 'birthYear'. You can override using the `.groups` argument.
```

```
name_by_year
```

```
## # A tibble: 12,834 x 4
## # Groups:   birthYear [170]
##    birthYear nameFirst     n  prop
##        <int> <chr>     <int> <dbl>
##  1      1820 Alexander     1 1
##  2      1824 Henry         1 1
##  3      1832 Lew           1 0.333
##  4      1832 Nate          1 0.333
##  5      1832 William       1 0.333
##  6      1835 Harry         1 1
##  7      1836 Dickey        1 1
##  8      1837 Morgan        1 1
```

```
##  9      1838 Bill          1 0.333
## 10      1838 Dave          1 0.333
## # ... with 12,824 more rows
```

(1b) In the Master dataframe, let us check whether the variable birthYear is consistent with the year in birthDate.

Use a function in the lubridate package (discussed in Lecture 2) to extract the year from the birthDate. Call this variable birthYear2.

In how many cases does birthYear have an "NA" entry?

In how many cases does birthYear2 have an "NA" entry?

In how many cases do both have "NA" entries?

if you ignore all the cases with at least one "NA" entry (either in the birthYear or birthYear2 variable), do all the remaining cases match?

Create a temporary dataFrame

```
q1_t <- Master_t %>% select(playerID, birthYear, birthDate) %>%
  mutate(birthYear2 = year(birthDate))
```

In how many cases does birthYear have an "NA" entry

```
count_na_birthYear <- q1_t %>% select(birthYear) %>%
  is.na() %>% sum() %>% as.numeric()
count_na_birthYear
```

```
## [1] 115
```

In how many cases does birthYear2 have an "NA" entry?

```
count_na_birthYear2 <- q1_t %>% select(birthYear2) %>%
  is.na() %>% sum() %>% as.numeric()
count_na_birthYear2
```

```
## [1] 426
```

In how many cases do both have "NA" Entries

```
count_both_na <- q1_t %>%
  filter(is.na(birthYear), is.na(birthYear2)) %>% nrow() %>% as.numeric()
count_both_na
```

```
## [1] 115
```

If you ignore all cases with at least one "NA" entry,, do all the remaining Cases match? To answer this, first create a temporary data frame neither_na_t Use it to compute count_neither_na and count_matching_after_neither_na

```
neither_na_t <- q1_t%>%
  filter(!(is.na(birthYear) | is.na(birthYear2)))
count_neither_na <- neither_na_t %>% nrow() %>% as.numeric()
count_neither_na
```

```
## [1] 19452
```

```
count_matching_after_neither_na <- neither_na_t %>%
  filter(as.numeric(birthYear) == as.numeric(birthYear2)) %>% nrow() %>% as.numeric()
count_matching_after_neither_na
```

```
## [1] 19452
```

As you can see based on the results, the count is the same, meaning that all the remaining cases match

(1c) Create a data frame of players showing just the playerID, first name,

last name, given name, and career total

(meaning, summed over all years and all stints) of games

(that is, the G variable) according to the Fielding data frame.

Create a tibble to store Fielding data

```
fielding_t <- as_tibble(Fielding)
```

Create a variable FieldingSmall containing only playerID and G from fielding_t

```
FieldingSmall <- fielding_t %>% select(playerID, G)
```

Create a variable Master_small_t containing only playerID, nameFirst, nameLast and name given from Master_t

```
Master_small_t <- Master_t %>% select(playerID, nameFirst, nameLast, nameGiven)
```

Perform a left join between the two subset tables on the key playerID Then, pipe the result to group by playerID, nameFirst, nameLast, nameGiven Pipe the result after grouping to create a final tibble with added total games column

```
player_t <- FieldingSmall %>% left_join(Master_small_t, by = "playerID") %>%
  group_by(playerID, nameFirst, nameLast, nameGiven) %>% summarise(total_games = sum(G))
```

```
## `summarise()` has grouped output by 'playerID', 'nameFirst', 'nameLast'. You can override using the
```

```
player_t
```

```
## # A tibble: 19,491 x 5
## # Groups:   playerID, nameFirst, nameLast [19,491]
##    playerID  nameFirst nameLast    nameGiven        total_games
##    <chr>     <chr>     <chr>       <chr>                  <int>
##  1 aardsda01 David     Aardsma     David Allan              331
##  2 aaronha01 Hank      Aaron       Henry Louis             3020
##  3 aaronto01 Tommie    Aaron       Tommie Lee               387
##  4 aasedo01  Don       Aase        Donald William           448
##  5 abadan01  Andy      Abad        Fausto Andres              9
##  6 abadfe01  Fernando  Abad        Fernando Antonio         384
##  7 abadijo01 John      Abadie      John W.                   12
##  8 abbated01 Ed        Abbaticchio Edward James             830
##  9 abbeybe01 Bert      Abbey       Bert Wood                 79
## 10 abbeych01 Charlie   Abbey       Charles S.               452
## # ... with 19,481 more rows
```

(1d) Using mutate() and str_c(), add a variable to your data frame in (c) for full name by combining the first name and last name with a space between them.

```
player_t <- player_t %>% mutate(fullName =
                                     str_c(nameFirst, nameLast, sep = " "))
player_t
```

```
## # A tibble: 19,491 x 6
## # Groups:   playerID, nameFirst, nameLast [19,491]
##    playerID  nameFirst nameLast    nameGiven        total_games fullName
##    <chr>     <chr>     <chr>       <chr>                  <int> <chr>
##  1 aardsda01 David     Aardsma     David Allan              331 David Aardsma
##  2 aaronha01 Hank      Aaron       Henry Louis             3020 Hank Aaron
##  3 aaronto01 Tommie    Aaron       Tommie Lee               387 Tommie Aaron
##  4 aasedo01  Don       Aase        Donald William           448 Don Aase
##  5 abadan01  Andy      Abad        Fausto Andres              9 Andy Abad
##  6 abadfe01  Fernando  Abad        Fernando Antonio         384 Fernando Abad
##  7 abadijo01 John      Abadie      John W.                   12 John Abadie
##  8 abbated01 Ed        Abbaticchio Edward James             830 Ed Abbaticchio
##  9 abbeybe01 Bert      Abbey       Bert Wood                 79 Bert Abbey
## 10 abbeych01 Charlie   Abbey       Charles S.               452 Charlie Abbey
## # ... with 19,481 more rows
```

(1e) Use the data frames you've created to determine the 5 most popular first names in baseball among players who played at least 500 games.

Plot them over time with lines in a single plot.

Be sure to make the plot look nice by using a title and changing the

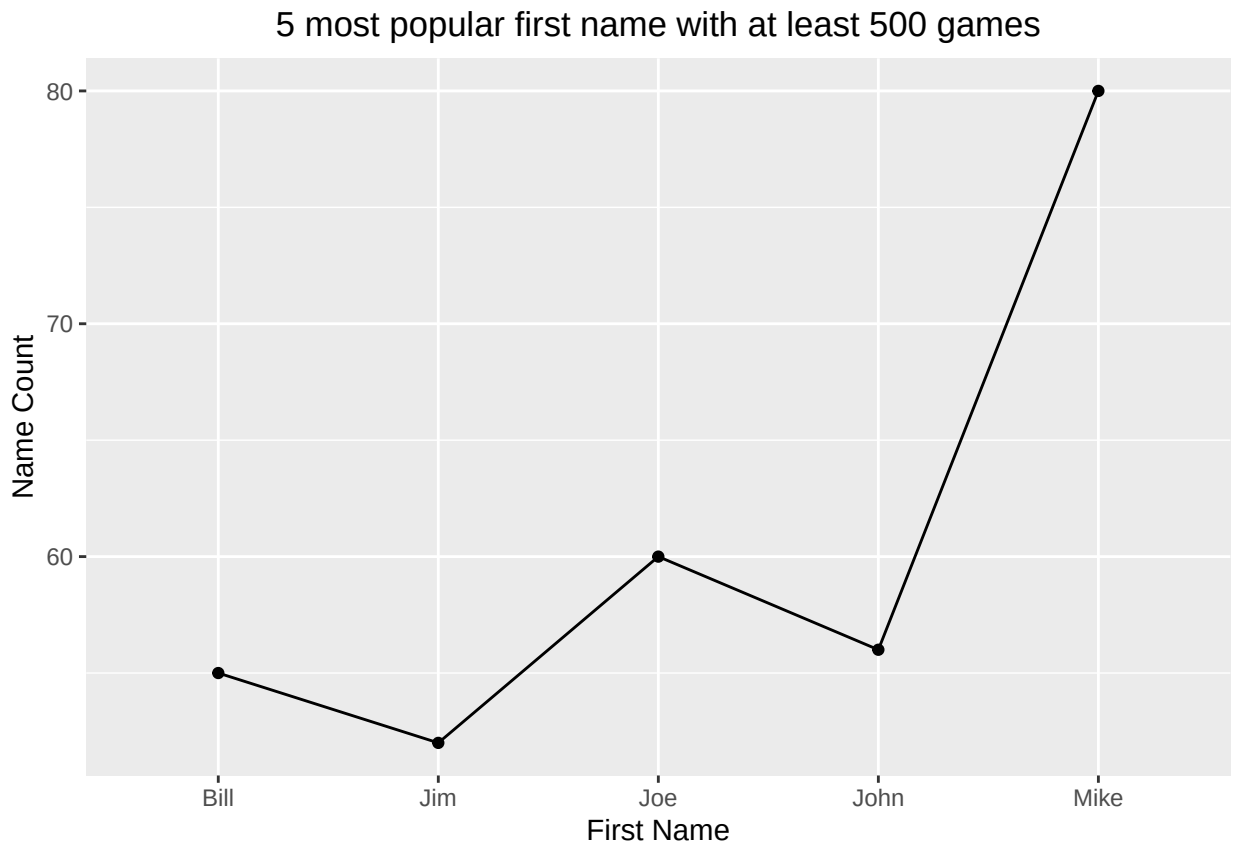
axis labels if necessary.

Compute the name_count by filtering player_t to only players over 500 games, piping the result to ungroup() in order to calculate the count of first names piping the result to arrange by n in descending, and finally using slice to split to top 5

```
name_count <- player_t %>% filter(total_games >= 500) %>%
  ungroup() %>%
  count(nameFirst) %>%
  arrange(desc(n)) %>%
  slice(1:5)
```

Generate the plot

```
ggplot(data = name_count,
       mapping = aes(x = nameFirst, y = n, group = 1)) +
  geom_line() +
  geom_point() +
  labs(
    title = "5 most popular first name with at least 500 games",
    x = "First Name",
    y = "Name Count"
  ) +
  theme(plot.title = element_text(hjust = 0.5))
```



2) Read the post at http://www.sumsar.net/blog/2016/09/whats-on-the-menu/ and follow the steps yourself.

(Please include the R code in the RMarkdown file up through the creation of the data frame d - a terrible name, by the way.)

Would you have plotted any of the graphs on the webpage differently?

The author of the blog post found some interesting, or at least amusing, things in the data. Replacing the boring "tea", "coffee", "apple" "banana", explore the rise or fall of more interesting things like noodles, tandoori, curry, sushi and kale in the menu.

```
menu_data_url <-
  "https://s3.amazonaws.com/menusdata.nypl.org/gzips/2016_09_16_07_00_30_data.tgz"
temp_dir <- tempdir()
curl_download(menu_data_url, file.path(temp_dir, "menu_data.tgz"))
untar(file.path(temp_dir, "menu_data.tgz"), exdir = temp_dir)
dish <- read_csv(file.path(temp_dir, "Dish.csv"))
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   id = col_double(),
##   name = col_character(),
##   description = col_logical(),
##   menus_appeared = col_double(),
##   times_appeared = col_double(),
##   first_appeared = col_double(),
##   last_appeared = col_double(),
##   lowest_price = col_double(),
##   highest_price = col_double()
## )
```

```
menu <- read_csv(file.path(temp_dir, "Menu.csv"))
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   .default = col_character(),
##   id = col_double(),
##   name = col_logical(),
##   keywords = col_logical(),
##   language = col_logical(),
##   date = col_date(format = ""),
##   location_type = col_logical(),
##   page_count = col_double(),
##   dish_count = col_double()
## )
## i Use `spec()` for the full column specifications.

## Warning: 3197 parsing failures.
##   row  col           expected               actual
##  4598 name 1/0/T/F/TRUE/FALSE The Modern           'C:\Users\Julia\AppData\Local\Temp\RtmpwnSSvd/Menu
## 11205 name 1/0/T/F/TRUE/FALSE Restaurant           'C:\Users\Julia\AppData\Local\Temp\RtmpwnSSvd/Menu
## 13185 name 1/0/T/F/TRUE/FALSE El Fuerte Del Palmar 'C:\Users\Julia\AppData\Local\Temp\RtmpwnSSvd/Menu
## 13325 name 1/0/T/F/TRUE/FALSE Archons of Colophone 'C:\Users\Julia\AppData\Local\Temp\RtmpwnSSvd/Menu
## 14353 name 1/0/T/F/TRUE/FALSE Chalfonte            'C:\Users\Julia\AppData\Local\Temp\RtmpwnSSvd/Menu
```

```
## ..... .... .................. ..... ..........
## See problems(...) for more details.
```

```
menu_item <- read_csv(file.path(temp_dir, "MenuItem.csv"))
```

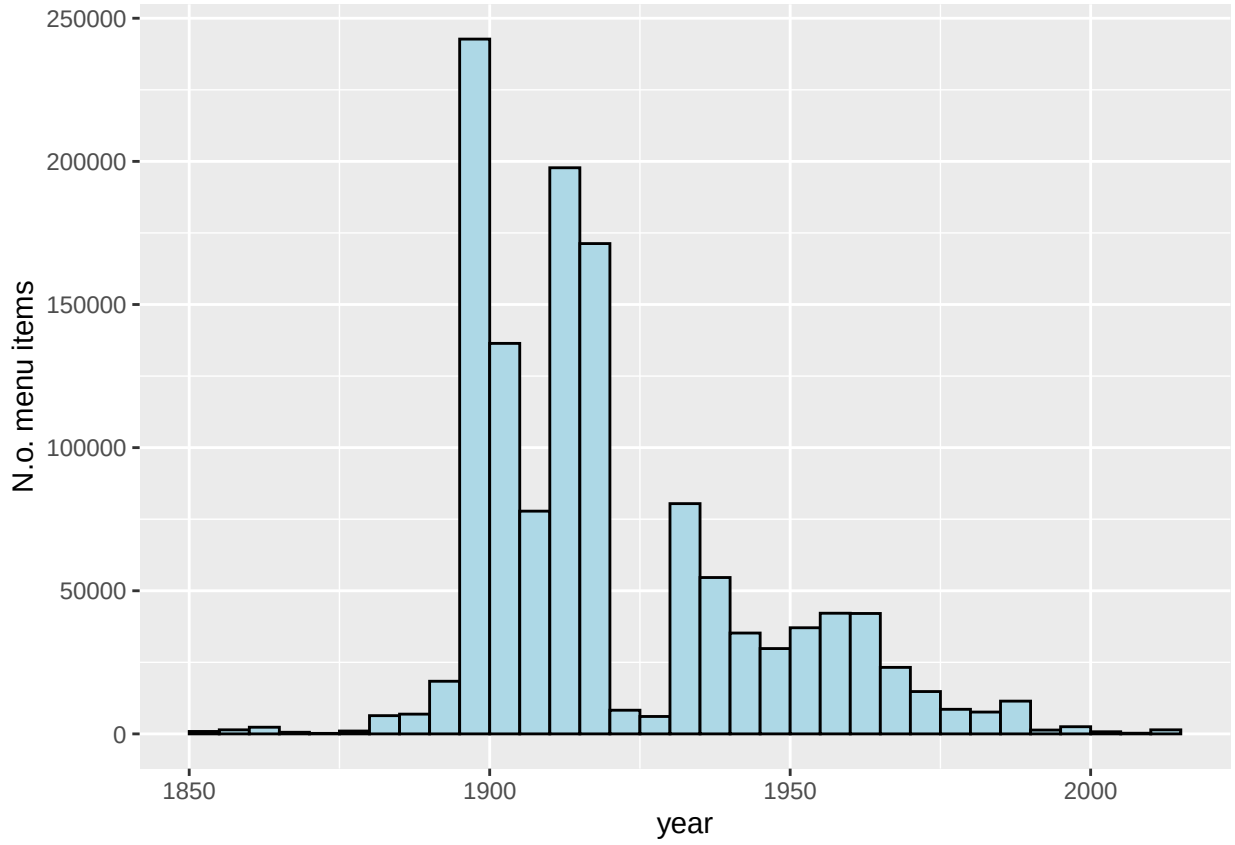
```
##
## -- Column specification --------------------------------------------------------
## cols(
##   id = col_double(),
##   menu_page_id = col_double(),
##   price = col_double(),
##   high_price = col_double(),
##   dish_id = col_double(),
##   created_at = col_character(),
##   updated_at = col_character(),
##   xpos = col_double(),
##   ypos = col_double()
## )
```

```
menu_page <- read_csv(file.path(temp_dir, "MenuPage.csv"))
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   id = col_double(),
##   menu_id = col_double(),
##   page_number = col_double(),
##   image_id = col_double(),
##   full_height = col_double(),
##   full_width = col_double(),
##   uuid = col_character()
## )

## Warning: 23 parsing failures.
##   row      col expected         actual                                                                  
## 13943 image_id a double ps_rbk_637     'C:\Users\Julia\AppData\Local\Temp\RtmpwnSSvd/MenuPage.csv'
## 13944 image_id a double ps_rbk_657     'C:\Users\Julia\AppData\Local\Temp\RtmpwnSSvd/MenuPage.csv'
## 13945 image_id a double ps_rbk_661     'C:\Users\Julia\AppData\Local\Temp\RtmpwnSSvd/MenuPage.csv'
## 13946 image_id a double psnypl_rbk_951 'C:\Users\Julia\AppData\Local\Temp\RtmpwnSSvd/MenuPage.csv'
## 13947 image_id a double psnypl_rbk_952 'C:\Users\Julia\AppData\Local\Temp\RtmpwnSSvd/MenuPage.csv'
## ..... ........ ........ ............. ..........
## See problems(...) for more details.
```

```
d <- menu_item %>% select( id, menu_page_id, dish_id, price) %>%
  left_join(dish %>% select(id, name) %>% rename(dish_name = name),
            by = c("dish_id" = "id")) %>%
  left_join(menu_page %>% select(id, menu_id),
            by = c("menu_page_id" = "id")) %>%
  left_join(menu %>% select(id, date, place, location),
            by = c("menu_id" = "id")) %>%
  mutate(year = lubridate::year(date)) %>%
  filter(!is.na(year)) %>%
```

```
  filter(year > 1800 & year <= 2016) %>%
  select(year, location, menu_id, dish_name, price, place)




ggplot(d, aes(year)) +
  geom_histogram(binwidth = 5, center = 1902.5, color = "black", fill = "lightblue") +
  scale_y_continuous("N.o. menu items")
```




```
d$decennium = floor(d$year / 10) * 10
#foods <- c("coffee", "tea", "pancake", "ice cream", "french frie",
#           "french peas", "apple", "banana", "strawberry")

foods <- c("noodle", "tandoori", "curry", "sushi", "kale")

food_over_time <- map_df(foods, function(food) {
  d %>%
    filter(year >= 1900 & year <= 1980) %>%
    group_by(decennium, menu_id) %>%
    summarise(contains_food =
                any(str_detect(dish_name, regex(paste0("\\b", food), ignore_case = TRUE)),
                    na.rm = TRUE)) %>%
    summarise(prop_food = mean(contains_food, na.rm = TRUE)) %>%
    mutate(food = food)
```

```
})
```

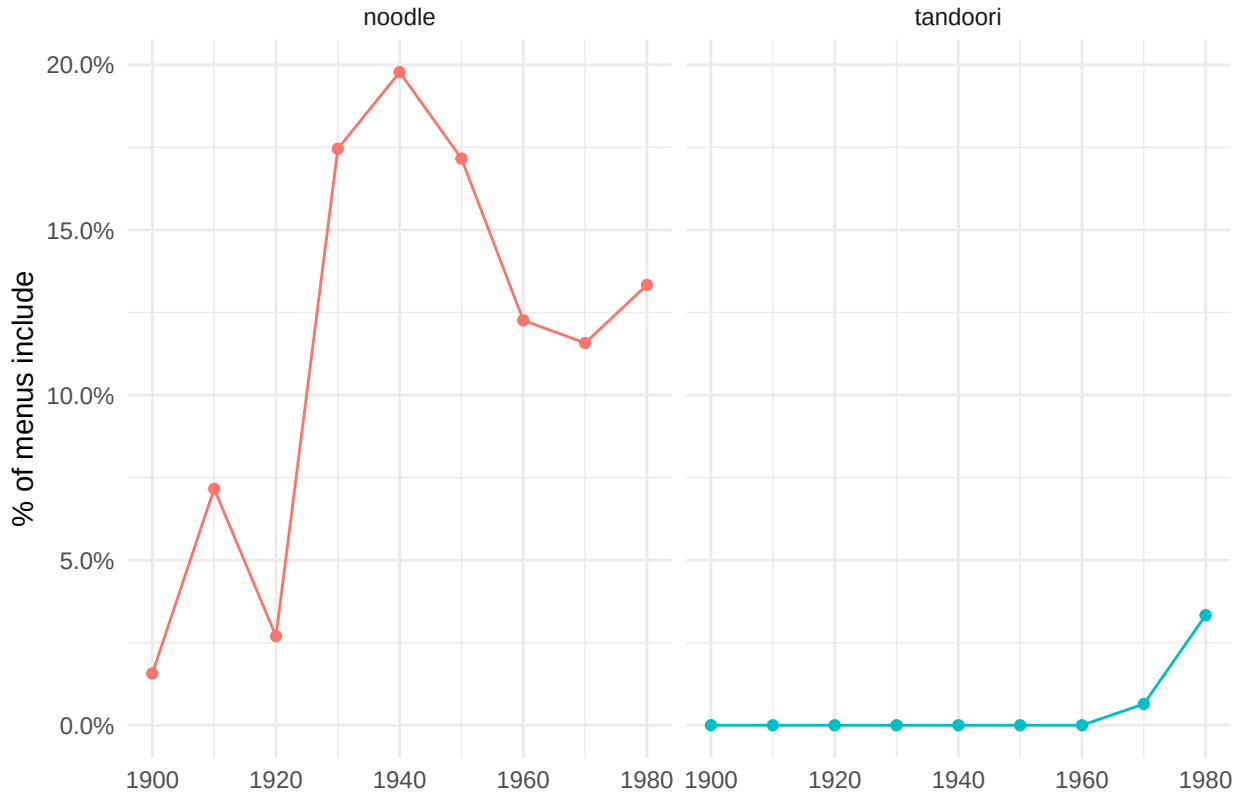
```
## `summarise()` has grouped output by 'decennium'. You can override using the `.groups` argument.
```

```
## `summarise()` has grouped output by 'decennium'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'decennium'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'decennium'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'decennium'. You can override using the `.groups` argument.
```

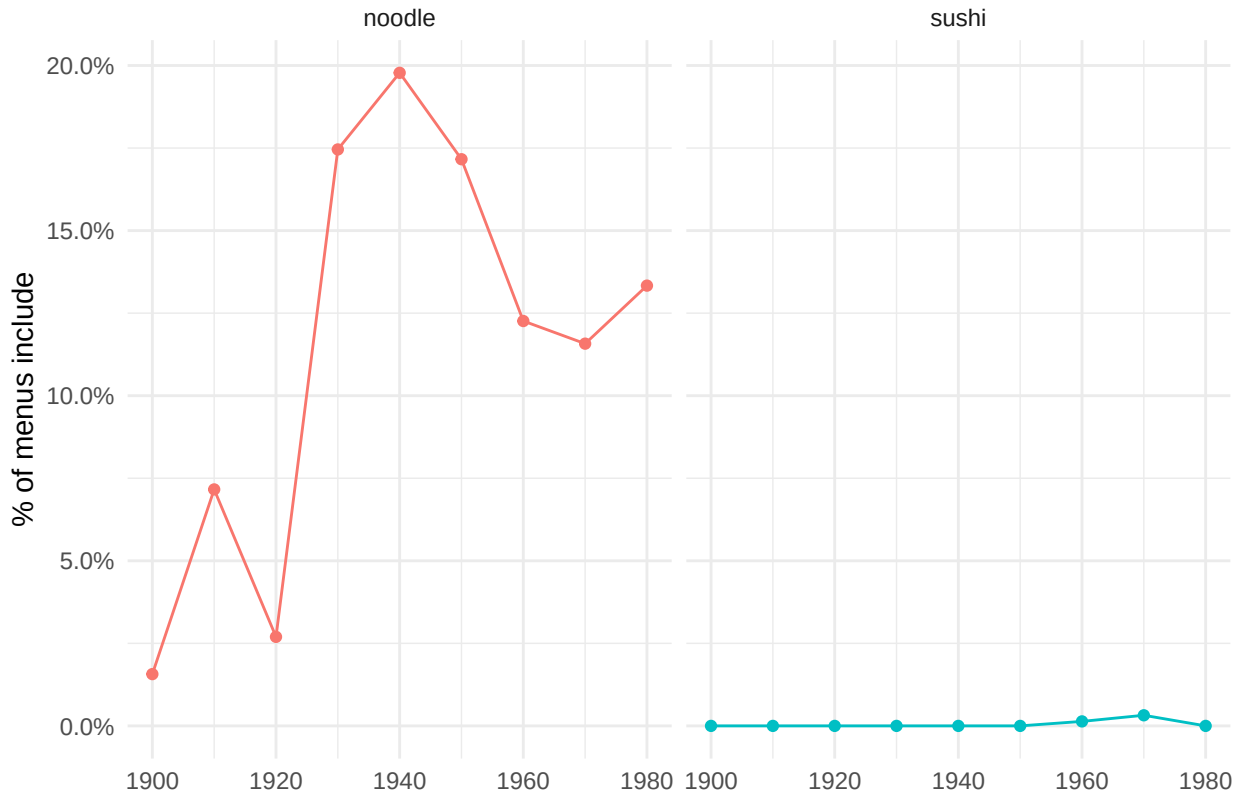
```
# A reusable list of ggplot2 directives to produce a lineplot
food_time_plot <- list(
  geom_line(),
  geom_point(),
  scale_y_continuous("% of menus include",labels = scales::percent,
                     limits = c(0, NA)),
  scale_x_continuous(""),
  facet_wrap(~ food),
  theme_minimal(),
  theme(legend.position = "none"))
```

Here is my own exploration of the various foods in the list noodles, tandoori, curry, sushi, and kale

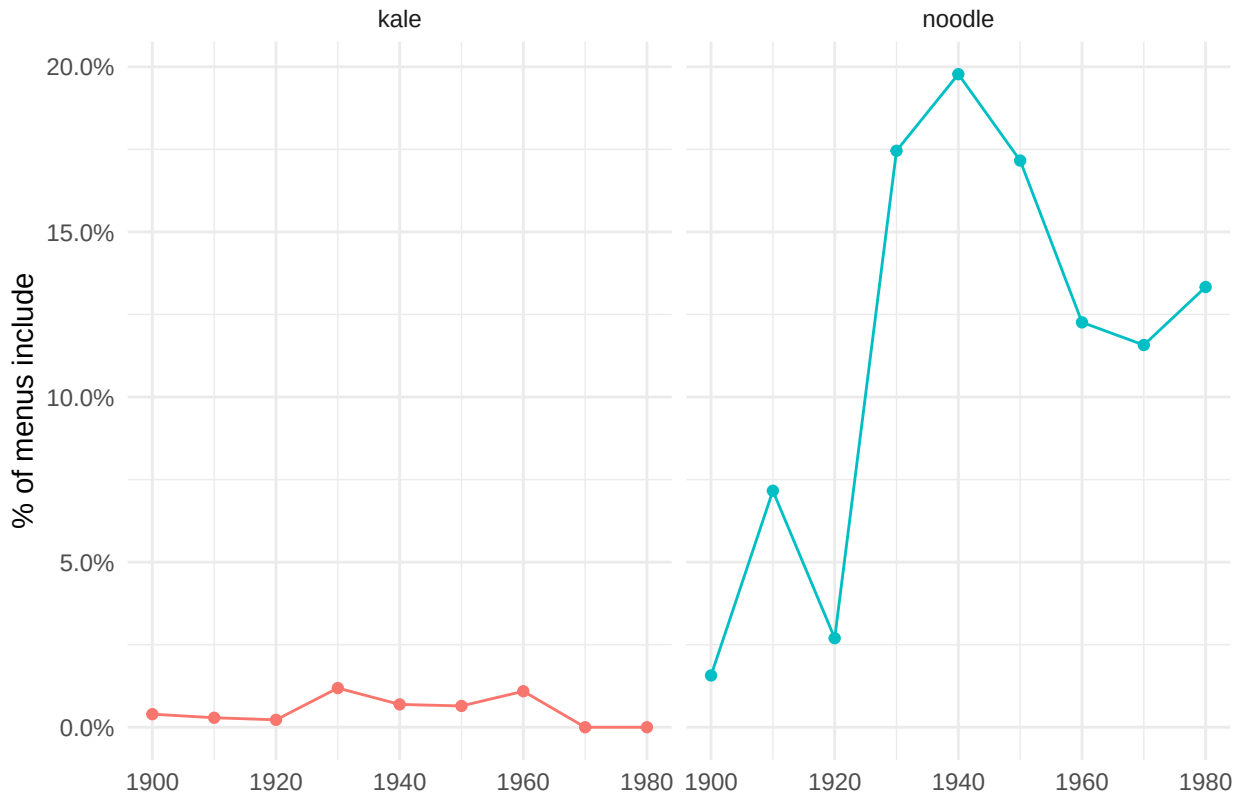
```
food_over_time %>% filter(food %in% c("noodle", "tandoori")) %>%
  ggplot(aes(decennium, prop_food, color = food)) + food_time_plot
```



```
food_over_time %>% filter(food %in% c("noodle", "sushi")) %>%
  ggplot(aes(decennium, prop_food, color = food)) + food_time_plot
```




```
food_over_time %>% filter(food %in% c("noodle", "kale")) %>%
  ggplot(aes(decennium, prop_food, color = food)) + food_time_plot
```



I would have plotted the above graphs as different colored lines on the same plot instead of each one on a different plot This first plot demonstrates that tandoori only started getting popular in the 1980's

the second plot demonstrates that noodes have generally been popular compared to sushi

In the last plot, it can be seen that Kale at it's peak comprised less than 2.5% of the data