Problem Set 1

Data Analysis and Visualization using R

RStudy2020

Due by 2020년 10월 11일 일요일 5시 (한국시간)

Information & Instructions

출제자: 박상훈
데이터 파일
- R의 `nycflights13` 패키지가 제공하는 데이터를 사용.

제출
- 정해진 시간까지 정해진 문제에 대한 코드를 작성한 R 스크립트를 `sp23@email.sc.edu` 혹은 Dropbox의 `4_QnA or Discussion` 폴더에 업로드.
- 모든 코드는 다른 작업환경에서 열더라도 재생산가능하게 디렉토리 설정 등을 모두 고려한 결과물이어야 함.

Here starts the actual test

Part 1: Data loading and cleaning

문제 1

패키지의 설치와 라이브러리 이용 `nycflights13` 패키지를 설치하고, 해당 라이브러리에서 `flights` 데이터를 `df` 라는 이름으로 저장하라. 이후의 모든 문제에 대해서 결과를 별도의 데이터에 따로 저장하지 말고 그대로 작업하라. 즉, `new_df <- df %>% drop_na()`가 아니라 `df %>% drop_na()` 로만 작업하여 콘솔창에 그 결과를 바로 확인할 수 있도록 하라.

만약 부득이하게 별도의 데이터를 저장해야 할 경우, 해당 데이터는 `temp`로 명명하라.

```
install.packages("nycflights13")
```

```
## Error in contrib.url(repos, "source"): trying to use CRAN without setting a mirror
```

```
library(nycflights13)
df <- flights
```

문제 2

`dplyr`의 `filter()`

데이터에서 3월에 해당하는 경우만을 선택하라. `month` 변수는 숫자형으로 1은 1월, 2는 2월을 나타내는 형식을 가지고 있다. 결과를 별도의 객체로 저장하지말고 실행만 하는 코드를 제시하라.

```
library(tidyverse)
df %>% dplyr::filter(month == 3) %>% head()
```

```
## # A tibble: 6 x 19
##    year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
##   <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
## 1  2013     3     1        4           2159       125      318             56
## 2  2013     3     1       50           2358        52      526            438
## 3  2013     3     1      117           2245       152      223           2354
## 4  2013     3     1      454            500        -6      633            648
## 5  2013     3     1      505            515       -10      746            810
## 6  2013     3     1      521            530        -9      813            827
## # ... with 11 more variables: arr_delay <dbl>, carrier <chr>, flight <int>,
## #   tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>,
## #   hour <dbl>, minute <dbl>, time_hour <dttm>
```

기항지(origin airport)가 `JFK` 또는 `LGA`이며 동시에 겨울에 비행한 경우를 선택하라. 이때, 겨울은 11월부터 2월을 의미한다.

```
df %>% dplyr::filter((origin %in% c("JFK", "LGA")) &
                       (month %in% c(11, 12, 1, 2))) %>% head()
```

```
## # A tibble: 6 x 19
##    year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
##   <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
## 1  2013     1     1      533            529         4      850            830
## 2  2013     1     1      542            540         2      923            850
## 3  2013     1     1      544            545        -1     1004           1022
## 4  2013     1     1      554            600        -6      812            837
## 5  2013     1     1      557            600        -3      709            723
## 6  2013     1     1      557            600        -3      838            846
## # ... with 11 more variables: arr_delay <dbl>, carrier <chr>, flight <int>,
## #   tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>,
## #   hour <dbl>, minute <dbl>, time_hour <dttm>
```

문제 3

`dplyr`의 `select()`

변수의 제거 어차피 모든 연도는 2013년이므로 불필요한 연도 변수를 데이터에서 제거하라. 제거한 결과를 다시 `df`에 저장하지 말고, 실행하는 코드만을 제시하라.

```
df %>% dplyr::select(-year) %>% head()
```

```
## # A tibble: 6 x 18
##   month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
##   <int> <int>    <int>          <int>     <dbl>    <int>          <int>
## 1     1     1      517            515         2      830            819
## 2     1     1      533            529         4      850            830
## 3     1     1      542            540         2      923            850
```

```
## 4     1     1      544            545        -1     1004           1022
## 5     1     1      554            600        -6      812            837
## 6     1     1      554            558        -4      740            728
## # ... with 11 more variables: arr_delay <dbl>, carrier <chr>, flight <int>,
## #   tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>,
## #   hour <dbl>, minute <dbl>, time_hour <dttm>
```

변수의 순서 변경 변수 명에 언더스코어(_)를 포함한 경우를 제일 앞에 오도록 데이터의 변수 순서를 변경하라.

```
df %>% select(contains('_'), everything()) %>% head()
```

```
## # A tibble: 6 x 19
##   dep_time sched_dep_time dep_delay arr_time sched_arr_time arr_delay air_time
##      <int>          <int>     <dbl>    <int>          <int>     <dbl>    <dbl>
## 1      517            515         2      830            819        11      227
## 2      533            529         4      850            830        20      227
## 3      542            540         2      923            850        33      160
## 4      544            545        -1     1004           1022       -18      183
## 5      554            600        -6      812            837       -25      116
## 6      554            558        -4      740            728        12      150
## # ... with 12 more variables: time_hour <dttm>, year <int>, month <int>,
## #   day <int>, carrier <chr>, flight <int>, tailnum <chr>, origin <chr>,
## #   dest <chr>, distance <dbl>, hour <dbl>, minute <dbl>
```

변수의 부분 선택 dep_delay부터 tailnum 까지의 변수를 한 번에 선택하라.

```
df %>% select(dep_delay:tailnum) %>% head()
```

```
## # A tibble: 6 x 7
##   dep_delay arr_time sched_arr_time arr_delay carrier flight tailnum
##       <dbl>    <int>          <int>     <dbl> <chr>    <int> <chr>
## 1         2      830            819        11 UA        1545 N14228
## 2         4      850            830        20 UA        1714 N24211
## 3         2      923            850        33 AA        1141 N619AA
## 4        -1     1004           1022       -18 B6         725 N804JB
## 5        -6      812            837       -25 DL         461 N668DN
## 6        -4      740            728        12 UA        1696 N39463
```

변수의 유형별 선택 정수형(integer)인 변수들만 선택하라.

```
df %>% select_if(is.integer) %>% head()
```

```
## # A tibble: 6 x 8
##    year month   day dep_time sched_dep_time arr_time sched_arr_time flight
##   <int> <int> <int>    <int>          <int>    <int>          <int>  <int>
## 1  2013     1     1      517            515      830            819   1545
## 2  2013     1     1      533            529      850            830   1714
## 3  2013     1     1      542            540      923            850   1141
## 4  2013     1     1      544            545     1004           1022    725
## 5  2013     1     1      554            600      812            837    461
## 6  2013     1     1      554            558      740            728   1696
```

변수의 이름을 이용한 선택 변수명이 arr 또는 dep로 시작하는 경우만을 선택하라.

```
df %>% select(starts_with('arr'), starts_with('dep')) %>% head()
```

```
## # A tibble: 6 x 4
##   arr_time arr_delay dep_time dep_delay
##      <int>     <dbl>    <int>     <dbl>
## 1      830        11      517         2
## 2      850        20      533         4
## 3      923        33      542         2
## 4     1004       -18      544        -1
## 5      812       -25      554        -6
## 6      740        12      554        -4
```

문제 4

dplyr의 mutate()

변수 생성 total_delay라는 변수를 만들어라. 이 변수는 도착 시간과 출발 지연 시간을 더한 결과를 보여주는 변수여야 한다. 그리고 arr_time, dep_delay 보다는 앞에, 다른 변수들보다는 뒤에 위치하도록 순서를 변경하라 (dplyr::select 이용).

```
df %>% mutate(total_delay = arr_delay + dep_delay) %>%
  select(dep_delay, arr_delay, total_delay, everything()) %>% head()
```

```
## # A tibble: 6 x 20
##   dep_delay arr_delay total_delay  year month   day dep_time sched_dep_time
##       <dbl>     <dbl>       <dbl> <int> <int> <int>    <int>          <int>
## 1         2        11          13  2013     1     1      517            515
## 2         4        20          24  2013     1     1      533            529
## 3         2        33          35  2013     1     1      542            540
## 4        -1       -18         -19  2013     1     1      544            545
## 5        -6       -25         -31  2013     1     1      554            600
## 6        -4        12           8  2013     1     1      554            558
## # ... with 12 more variables: arr_time <int>, sched_arr_time <int>,
## #   carrier <chr>, flight <int>, tailnum <chr>, origin <chr>, dest <chr>,
## #   air_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>, time_hour <dttm>
```

변수의 조건 변경 모든 문자형 변수들을 요인형(factor)로 변경하고 바뀐 결과를 데이터의 구조를 보여주는 함수로 제시하라.

```
df %>% mutate_if(is.character, ~ as.factor(.)) %>% glimpse() %>% head()
```

```
## Rows: 336,776
## Columns: 19
## $ year           <int> 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013...
## $ month          <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1...
## $ day            <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1...
## $ dep_time       <int> 517, 533, 542, 544, 554, 554, 555, 557, 557, 558, 55...
## $ sched_dep_time <int> 515, 529, 540, 545, 600, 558, 600, 600, 600, 600, 60...
## $ dep_delay      <dbl> 2, 4, 2, -1, -6, -4, -5, -3, -3, -2, -2, -2, -2, -2,...
## $ arr_time       <int> 830, 850, 923, 1004, 812, 740, 913, 709, 838, 753, 8...
## $ sched_arr_time <int> 819, 830, 850, 1022, 837, 728, 854, 723, 846, 745, 8...
```

```
## $ arr_delay      <dbl> 11, 20, 33, -18, -25, 12, 19, -14, -8, 8, -2, -3, 7,...
## $ carrier        <fct> UA, UA, AA, B6, DL, UA, B6, EV, B6, AA, B6, B6, UA, ...
## $ flight         <int> 1545, 1714, 1141, 725, 461, 1696, 507, 5708, 79, 301...
## $ tailnum        <fct> N14228, N24211, N619AA, N804JB, N668DN, N39463, N516...
## $ origin         <fct> EWR, LGA, JFK, JFK, LGA, EWR, EWR, LGA, JFK, LGA, JF...
## $ dest           <fct> IAH, IAH, MIA, BQN, ATL, ORD, FLL, IAD, MCO, ORD, PB...
## $ air_time       <dbl> 227, 227, 160, 183, 116, 150, 158, 53, 140, 138, 149...
## $ distance       <dbl> 1400, 1416, 1089, 1576, 762, 719, 1065, 229, 944, 73...
## $ hour           <dbl> 5, 5, 5, 5, 6, 5, 6, 6, 6, 6, 6, 6, 6, 6, 6, 5, 6, 6...
## $ minute         <dbl> 15, 29, 40, 45, 0, 58, 0, 0, 0, 0, 0, 0, 0, 0, 0, 59...
## $ time_hour      <dttm> 2013-01-01 05:00:00, 2013-01-01 05:00:00, 2013-01-0...
```

```
## # A tibble: 6 x 19
##    year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
##   <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
## 1  2013     1     1      517            515         2      830            819
## 2  2013     1     1      533            529         4      850            830
## 3  2013     1     1      542            540         2      923            850
## 4  2013     1     1      544            545        -1     1004           1022
## 5  2013     1     1      554            600        -6      812            837
## 6  2013     1     1      554            558        -4      740            728
## # ... with 11 more variables: arr_delay <dbl>, carrier <fct>, flight <int>,
## #   tailnum <fct>, origin <fct>, dest <fct>, air_time <dbl>, distance <dbl>,
## #   hour <dbl>, minute <dbl>, time_hour <dttm>
```

변수의 조건 생성 `distnace_bins`라는 변수를 만들어라. 이 변수는 거리(`distance`)가 1000보다 짧으면 "short", 1000과 2000 사이면 "medium", 그리고 2000보다 크면 "long"이라는 값을 가져야 한다. `if_else()` 또는 `case_when()`을 이용해 조건을 부여할 수 있다.

```
df %>%
  mutate(
    distance_bins = case_when(
      distance < 1000 ~ 'short',
      distance >= 1000 & distance <= 2000 ~ 'medium',
      distance > 2000 ~ 'long',
      T ~ NA_character_
    )
  ) %>% head()
```

```
## # A tibble: 6 x 20
##    year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
##   <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
## 1  2013     1     1      517            515         2      830            819
## 2  2013     1     1      533            529         4      850            830
## 3  2013     1     1      542            540         2      923            850
## 4  2013     1     1      544            545        -1     1004           1022
## 5  2013     1     1      554            600        -6      812            837
## 6  2013     1     1      554            558        -4      740            728
## # ... with 12 more variables: arr_delay <dbl>, carrier <chr>, flight <int>,
## #   tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>,
## #   hour <dbl>, minute <dbl>, time_hour <dttm>, distance_bins <chr>
```

보너스 문제

아래의 힌트를 참고하여 `distance_bins`를 순위를 가진 요인형 변수로 만들어라.

```
## case_when()으로 만든 df의 x 함수가 문자형일 때 아래와 같이 요인형으로 만들
## 수 있다.
## 방법 1
df %>% mutate(
  x = case_when(
    condition1 ~ "result1",
    condition2 ~ "result2",
    condition3 ~ "result3",
    TRUE ~ NA_character_
  ) %>% parse_factor(., levels = c("results1", "results2", "results3"),
                     ordered = T, include_na = F)
)

## 방법 2
df$x <- factor(c("results1", "results2", "results3"))
as.integer(df$x) # 이 경우 알파벳 순서대로 순위가 매겨진다.

## 방법 3
df$x <- factor(c("results1", "results2", "results3"),
               levels = c("results1", "results2", "results3"))
as.integer(df$x)
## 이 경우 results1 < results2 < results3 로 순위가 매겨진다.
```

만들어진 요인형 변수의 순위를 확인하라. `levels()` 함수를 이용하라.

```
temp <- df %>%
  mutate(
    distance_bins = case_when(
      distance < 1000 ~ 'short',
      distance >= 1000 & distance <= 2000 ~ 'medium',
      distance > 2000 ~ 'long',
      T ~ NA_character_
    ),
    distance_bins = parse_factor(
      distance_bins, levels = c('short', 'medium', 'long'), ordered = T,
      include_na = F
    )
  )
levels(temp$distance_bins)
```

```
## [1] "short"  "medium" "long"
```

문제 5

dplyr의 count()

빈도 계산 데이터셋에서 매일 이륙한 비행기의 수를 계산하라.

```
df %>% count(month, day) %>% head()
```

```
## # A tibble: 6 x 3
##   month   day     n
##   <int> <int> <int>
```

```
## 1     1     1   842
## 2     1     2   943
## 3     1     3   914
## 4     1     4   915
## 5     1     5   720
## 6     1     6   832
```

요약통계치 계산 매 달 각 기항지의 평균 출발지연 시간을 계산하라.

```
df %>%
  group_by(origin, month) %>%
  summarise(mean_dep_delay = mean(dep_delay, na.rm = T)) %>%
  print(n = 50) %>% head()
```

```
## # A tibble: 36 x 3
## # Groups:   origin [3]
##    origin month mean_dep_delay
##    <chr>  <int>          <dbl>
##  1 EWR        1          14.9
##  2 EWR        2          13.1
##  3 EWR        3          18.1
##  4 EWR        4          17.4
##  5 EWR        5          15.4
##  6 EWR        6          22.5
##  7 EWR        7          22.0
##  8 EWR        8          13.5
##  9 EWR        9           7.29
## 10 EWR       10           8.64
## 11 EWR       11           6.72
## 12 EWR       12          21.0
## 13 JFK        1           8.62
## 14 JFK        2          11.8
## 15 JFK        3          10.7
## 16 JFK        4          12.2
## 17 JFK        5          12.5
## 18 JFK        6          20.5
## 19 JFK        7          23.8
## 20 JFK        8          12.9
## 21 JFK        9           6.64
## 22 JFK       10           4.59
## 23 JFK       11           4.68
## 24 JFK       12          14.8
## 25 LGA        1           5.64
## 26 LGA        2           6.96
## 27 LGA        3          10.2
## 28 LGA        4          11.5
## 29 LGA        5          10.6
## 30 LGA        6          19.3
## 31 LGA        7          19.0
## 32 LGA        8          11.2
## 33 LGA        9           6.21
## 34 LGA       10           5.31
## 35 LGA       11           4.77
## 36 LGA       12          13.6
```

```
## # A tibble: 6 x 3
## # Groups:   origin [1]
##   origin month mean_dep_delay
##   <chr>  <int>          <dbl>
## 1 EWR        1           14.9
## 2 EWR        2           13.1
## 3 EWR        3           18.1
## 4 EWR        4           17.4
## 5 EWR        5           15.4
## 6 EWR        6           22.5
```

dplyr의 mutate()와 group_by(), summarise() 함수 겨울(12월-2월), 봄(3월-5월), 여름(6월-8월), 가을(9월-11월)에 해당하는 계절 변수를 만들어라. 그리고 각 계절별 기항지의 평균 출발지연 시간과 도착지연 시간을 계산하라. 이때, 만들어지는 결과를 summary라는 이름으로 저장하고, 티블 혹은 데이타프레임의 형식을 갖도록 저장하라.

```
df %>%
  mutate(
    season = case_when(
      month %in% c(1, 2, 12) ~ 'winter',
      month %in% c(3, 4, 5) ~ 'spring',
      month %in% c(6, 7, 8) ~ 'summer',
      month %in% c(9, 10, 11) ~ 'fall'
    ),
    season = parse_factor(
      season, levels = c('winter', 'spring', 'summer', 'fall'),
      include_na = F
    )
  ) %>%
  group_by(origin, season) %>%
  summarize_at(vars(dep_delay, arr_delay), list(mean, median), na.rm = T) %>%
  head()
```

```
## # A tibble: 6 x 6
## # Groups:   origin [2]
##   origin season dep_delay_fn1 arr_delay_fn1 dep_delay_fn2 arr_delay_fn2
##   <chr>  <fct>          <dbl>         <dbl>         <dbl>         <dbl>
## 1 EWR    winter         16.4         13.9               0             1
## 2 EWR    spring         16.9         10.0               0            -3
## 3 EWR    summer         19.3         13.0               0            -3
## 4 EWR    fall            7.57        -0.406            -2            -8
## 5 JFK    winter         11.7          6.19             -1            -4
## 6 JFK    spring         11.8          3.87             -2            -7
```

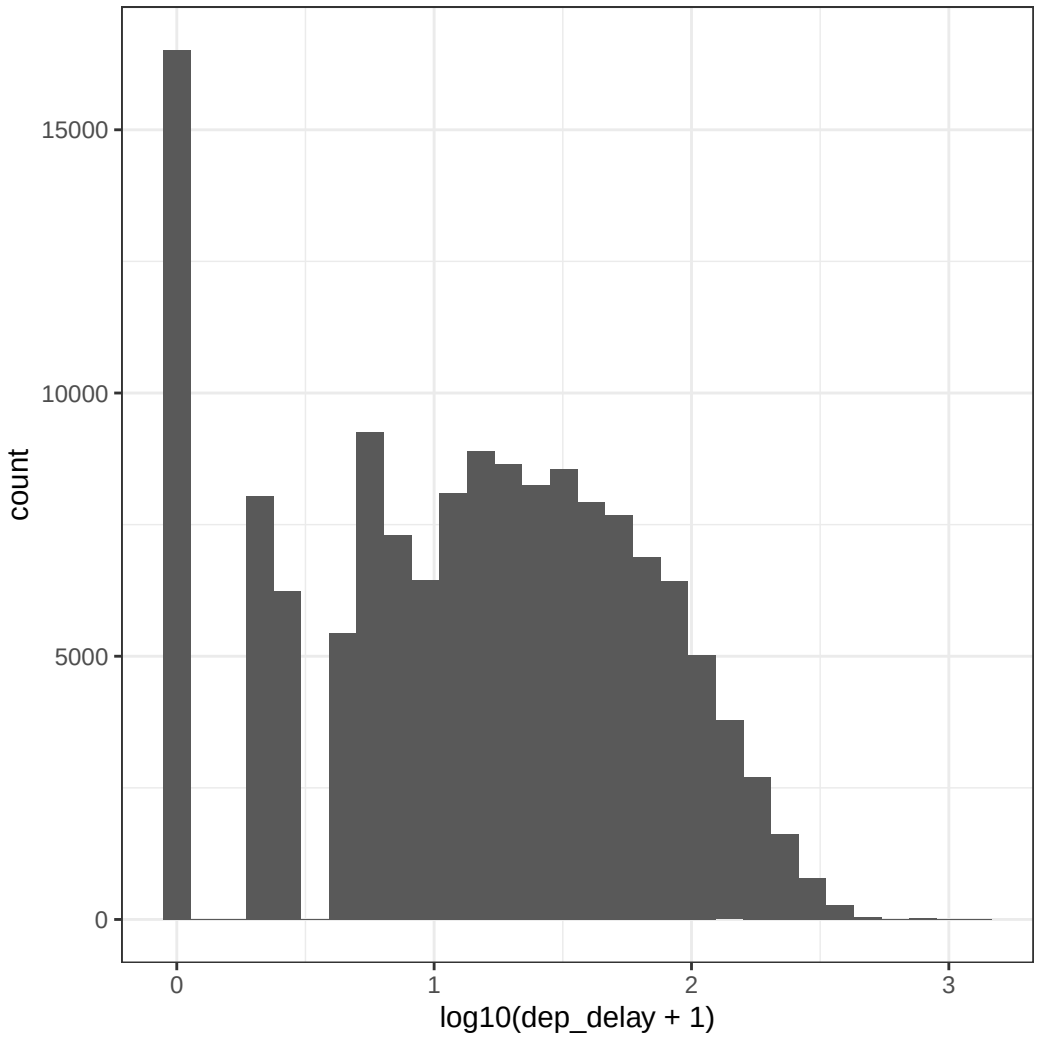
Part 2: Data Visualization

문제 6: ggplot2()

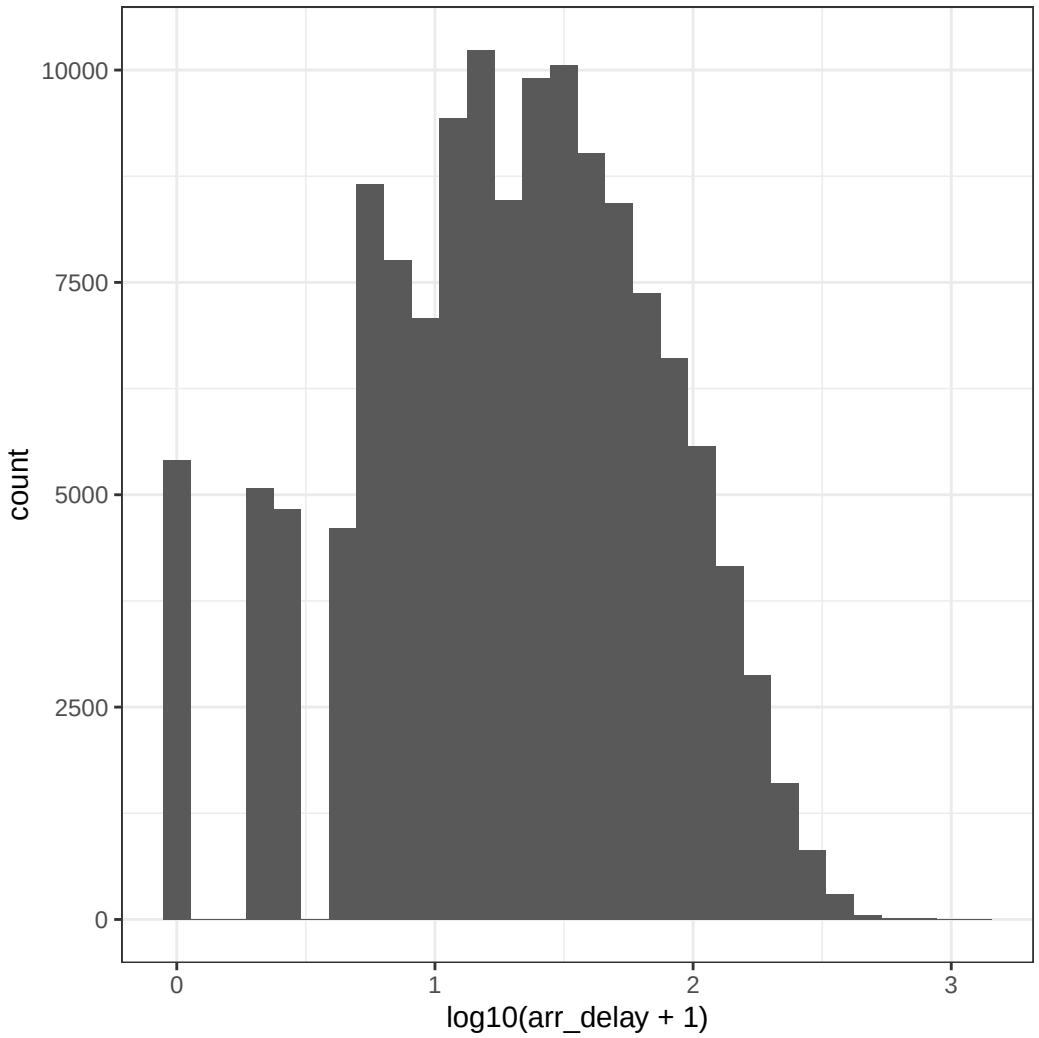
아래의 요구에 따라 플롯을 작성하라. 단, 모든 플롯은 적절한 축제목과 표제목을 갖추어야 한다.

dep_delay와 arr_delay의 분포를 가장 잘 보여줄 수 있는 플롯을 제시하라.

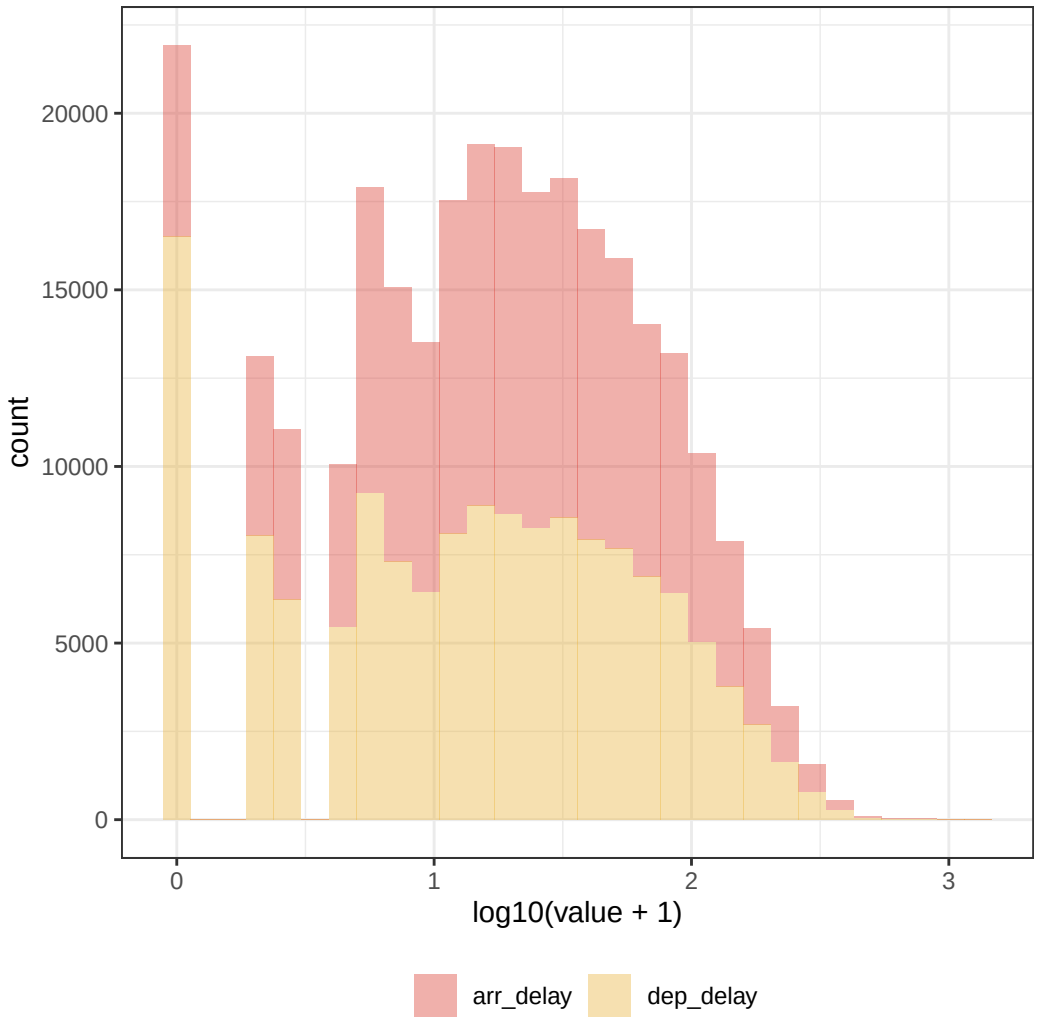
```
df %>%
  ggplot(aes(log10(dep_delay+1))) + geom_histogram() + theme_bw()
```




```
df %>%
  ggplot(aes(log10(arr_delay+1))) + geom_histogram() + theme_bw()
```

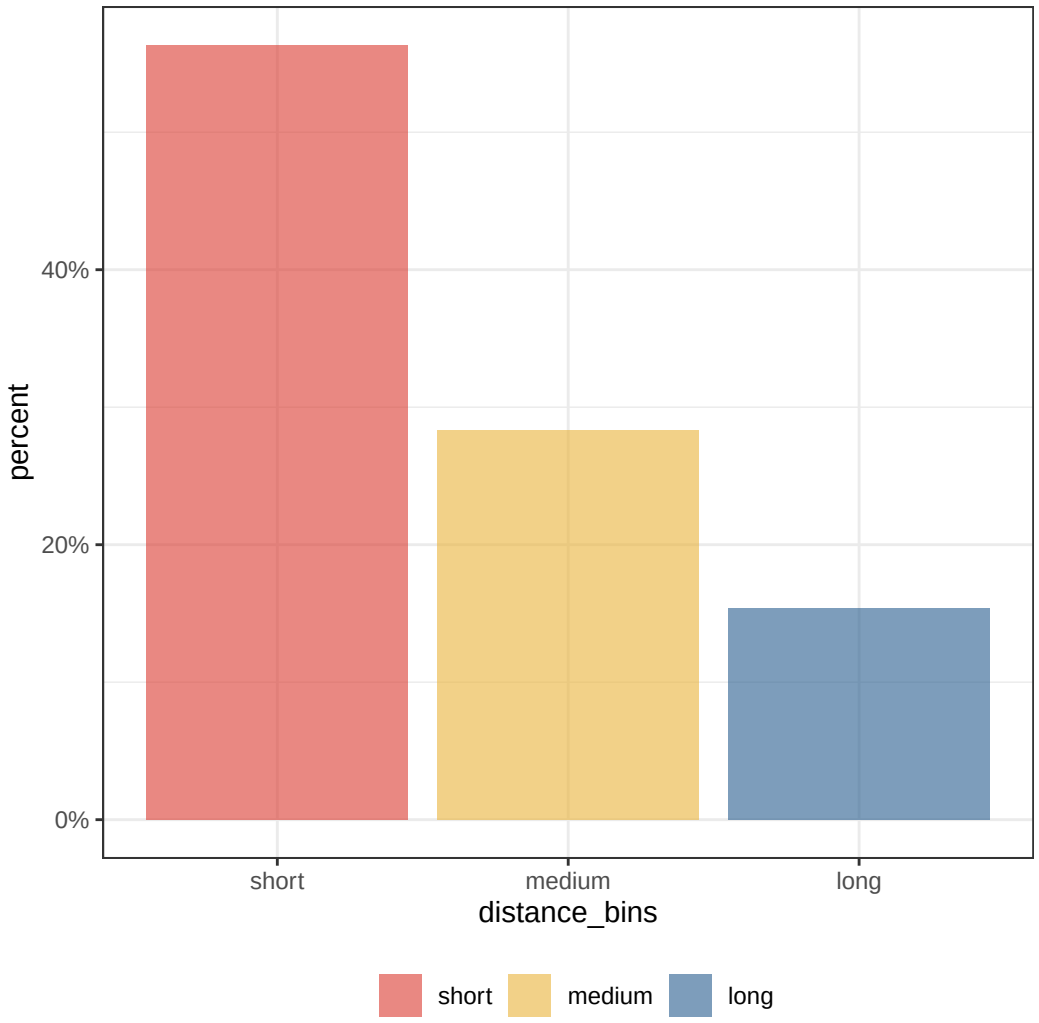



```
df %>% dplyr::select(dep_delay, arr_delay) %>% tidyr::gather(type, value) %>%
  ggplot(aes(log10(value+1), fill = as.factor(type))) +
  geom_histogram(alpha = 0.4) +
  scale_fill_manual(values = futurevisions::futurevisions("mars")) +
  theme_bw() + theme(legend.title = element_blank(),
                     legend.position = "bottom")
```



앞서 만든 distnace_bins 변수를 가장 잘 보여줄 수 있는 방식으로 플롯을 작성하라.

```
temp %>% group_by(distance_bins) %>% count() %>%
  mutate(percent = n / nrow(temp)) %>%
  ggplot(aes(x = distance_bins, y = percent, fill = distance_bins)) +
  geom_bar(stat = "identity", alpha = 0.6) +
  scale_y_continuous(labels=scales::percent) +
  scale_fill_manual(values = futurevisions::futurevisions("mars")) +
  theme_bw() + theme(legend.title = element_blank(),
                     legend.position = "bottom")
```



각 기항지의 계절별 평균 출발 지연 시간과 도착 지연 시간을 가장 잘 보여줄 수 있는 방식으로 플롯을 작성하라.

```
df %>%
  mutate(
    season = case_when(
      month %in% c(1, 2, 12) ~ 'winter',
      month %in% c(3, 4, 5) ~ 'spring',
      month %in% c(6, 7, 8) ~ 'summer',
      month %in% c(9, 10, 11) ~ 'fall'
      ),
    season = parse_factor(
      season, levels = c('winter', 'spring', 'summer', 'fall'),
      include_na = F
      )
    ) %>%
  group_by(origin, season) %>%
  summarize_at(vars(dep_delay, arr_delay), list(mean), na.rm = T) %>%
  tidyr::gather(delay, time, -origin, -season) %>%
  mutate(group = paste(origin, season, sep = " ")) %>%
  ggplot(aes(x = delay, fill = delay, y = time)) +
  geom_col(alpha = 0.6) +
```

```
scale_fill_manual(values = futurevisions::futurevisions("mars")) +
facet_wrap(~group) +
labs(x = "") + theme_bw() +
theme(axis.text.x = element_blank(),
      axis.ticks.x = element_blank(),
      legend.title = element_blank(),
                   legend.position = "bottom")
```