In [1]:

```
library(tidyverse)
library(nycflights13)
```

```
── Attaching packages ─────────────────────────────────────── tidyverse 1.2.1 ──
✔ ggplot2 2.2.1     ✔ purrr   0.2.4
✔ tibble  1.4.1     ✔ dplyr   0.7.4
✔ tidyr   0.7.2     ✔ stringr 1.2.0
✔ readr   1.1.1     ✔ forcats 0.2.0
── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
✖ dplyr::filter() masks stats::filter()
✖ dplyr::lag()    masks stats::lag()
```

STATS 306

Problem set 3: data manipulation II

These questions again focus on the `flights` dataset. Each question is worth two points, for a total of 20.

Note: you do not need to use `install.packages()` in this notebook. You may assume that we have already installed all of the necessary packages when we run your code.

Problem 1

Recall that each airplane has a unique tail number given by `tailnum`. Find the tail number of the airplane which flew to the largest number of unique destinations from NYC airports in 2013. Store the string containing this tail number in a variable called `most_dests`.

In [2]:

```
### BEGIN SOLUTION
most_dests = flights %>% filter(!is.na(tailnum)) %>% distinct(tailnum, dest) %>%
             group_by(tailnum) %>% count %>% arrange(desc(n)) %>% ungroup %>%
             .[[1,1]]
print(most_dests)
### END SOLUTION
```

```
[1] "N11194"
```

In [3]:

```
stopifnot(exists("most_dests"))
### BEGIN HIDDEN TESTS
ans = c()
for (f in list(flights, filter(flights, origin %in% c("JFK", "LGA")))) { # technically it specified NYC
    ans = c(ans,
            f %>% filter(!is.na(tailnum)) %>% distinct(tailnum, dest) %>%
             group_by(tailnum) %>% count %>% arrange(desc(n)) %>% ungroup %>% mutate(rank=min_rank(-n)) %>%
             filter(rank==1) %>% .$tailnum)
}
print(ans)
stopifnot(most_dests %in% ans)
### END HIDDEN TESTS
```

```
[1] "N11194" "N520JB" "N531JB" "N606JB"
```

Problem 2

Define a new table `flights2` obtained by adding a new column dow equal to the day of the week on which that flight took place. For example, the first flight in `flights` took place on Tuesday, January 1, 2013, so the first entry of column dow should equal Tuesday.

In [4]:

```
### BEGIN SOLUTION
library(lubridate)
flights2 = mutate(flights, dow=lubridate::wday(time_hour, label=T, abbr=F))
### END SOLUTION
```

```
Attaching package: ‘lubridate’

The following object is masked from ‘package:base’:

    date
```

In [5]:

```
stopifnot(exists("flights2"))
### BEGIN HIDDEN TESTS
flights2_ans = mutate(flights,
                      dow=lubridate::wday(time_hour, label=T, abbr=F),
                      dow_abbr=lubridate::wday(time_hour, label=T, abbr=T)) %>%
               arrange(year, month, day)
f2a = as.character(arrange(flights2, year, month, day)$dow)
# Accept either full or abbreviated form, even though the question asked for full form.
stopifnot(
    all(f2a == as.character(flights2_ans$dow)) |
    all(f2a == as.character(flights2_ans$dow_abbr))
    )
### END HIDDEN TESTS
```

Problem 3

Define a new table `flights3` by adding a new column doy equal to the numerical day of the year in which that flight took place. For example, the final flight in `flights` took place during on December 31st, so since 2013 was not a leap year, the final entry of doy should equal 365.

In [6]:

```
### BEGIN SOLUTION
flights3 = mutate(flights, doy=lubridate::yday(time_hour))
### END SOLUTION
```

In [7]:

```
stopifnot(exists('flights3'))
### BEGIN HIDDEN TESTS
flights3_ans = mutate(flights, doy=lubridate::yday(time_hour))
stopifnot("doy" %in% colnames(flights3))
stopifnot(all(near(arrange(flights3, year, month, day)$doy,
                   arrange(flights3_ans, year, month, day)$doy)))
### END HIDDEN TESTS
```

Problem 4

Use your solution from problem 2 to define a new variable week such that week=1 for doy=`1,2,...,7`, week=2 for doy=`8,...,14`, etc. Store the resulting table in a variable called `flights4`.

In [8]:

```
### BEGIN SOLUTION
flights4 = mutate(flights3, week=as.integer((doy-1) %/% 7 + 1))
### END SOLUTION
```

In [9]:

```
stopifnot(exists("flights4"))
### BEGIN HIDDEN TESTS
flights4_ans = mutate(flights3, week=lubridate::week(time_hour))
stopifnot("week" %in% colnames(flights4))
stopifnot(all(near(flights4_ans$week, flights4$week)))
### END HIDDEN TESTS
```

Problem 5

Let a flight's "positive arrival delay" be defined as the larger of `arr_delay` and zero. We say a flight is *ridiculously late* if its arrival delay was more than ten times the average positive arrival delay for all flights in that week.

- Use your solution from problem 4 to calculate the number of ridiculously late flights in each week of the year.
- Also add in the total number of flights in the data set for each week.

Sort the resulting table in descending order of the number of ridiculously late flights and store it in a variable called `table5`. The table should have three columns, week, n, and `n_ridiculously_late`.

In [10]:

```
### BEGIN SOLUTION
table5 = group_by(flights4, week) %>% mutate(n=n()) %>% filter(!is.na(arr_delay)) %>%
    filter(arr_delay > 10 * mean(pmax(arr_delay, 0), na.rm=T)) %>%
    summarize(n_ridiculously_late=n()) %>%
    arrange(desc(n_ridiculously_late))
print(table5)
### END SOLUTION
```

```
# A tibble: 53 x 2
    week n_ridiculously_late
   <int>               <int>
 1    40                 186
 2    36                 183
 3    13                 173
 4    18                 162
 5    38                 161
 6    39                 150
 7    34                 142
 8    35                 130
 9    46                 130
10    37                 129
# ... with 43 more rows
```

In [11]:

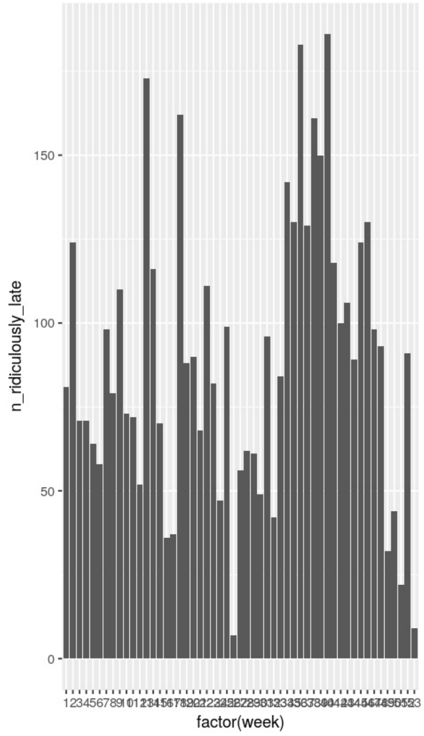
```
stopifnot(exists("table5"))
### BEGIN HIDDEN TESTS
table5_ans = group_by(flights4, week) %>% mutate(n=n()) %>%
    filter(arr_delay > 10 * mean(pmax(arr_delay, 0), na.rm=T)) %>%
    summarize(n=max(n), n_ridiculously_late=n()) %>%
    arrange(desc(n_ridiculously_late))
for (v in c('week', 'n_ridiculously_late')) {
    stopifnot(v %in% colnames(table5))
    if (! all(near(table5[[v]],
                   table5_ans[[v]]))) { stop(paste(v, "is not correct")) }
}
### END HIDDEN TESTS
```

Problem 6

Use your solution in part 5 to generate a bar plot, by week, of the number of ridiculously late flights each week. Give your plot an appropriate title and axis labels.

In [12]:

```
### BEGIN SOLUTION
ggplot(table5) + geom_col(aes(x=factor(week), y=n_ridiculously_late))
### END SOLUTION
```



Problem 7

Your plot in problem 6 should have a curious feature: in a couple of weeks there were far fewer ridiculously late flights than the rest.

- Investigate this further by determining the fraction of arrival delay times which were missing in each week.
- Additionally, rank each week by this fraction. The week with the highest fraction of missing arrival delay times should have rank one, second highest rank two, and so on.

Store the result in a variable called `table7`. Your table should have three columns: week, `p_miss_arr_delay` and `rank`.

In [13]:

```
### BEGIN SOLUTION
table7 = group_by(flights4, week)  %>%
    summarize(p_miss_arr_delay=mean(is.na(arr_delay))) %>%
    ungroup() %>% mutate(rank=min_rank(desc(p_miss_arr_delay)))

table7 = flights %>% mutate(week=lubridate::week(time_hour)) %>%
         group_by(week) %>%
         summarize(p_miss_arr_delay=sum(is.na(arr_delay)/n())) %>%
         mutate(rank=min_rank(desc(p_miss_arr_delay))) %>%
         select(week, p_miss_arr_delay, rank) %>% arrange(rank)
### END SOLUTION
```

In [14]:

```
stopifnot(exists("table7"))
### BEGIN HIDDEN TESTS
table7_ans = group_by(flights4, week)  %>%
    summarize(p_miss_arr_delay=mean(is.na(arr_delay))) %>%
    ungroup() %>% mutate(rank=min_rank(desc(p_miss_arr_delay))) %>%
    arrange(rank)
for (v in c('week', 'p_miss_arr_delay', 'rank')) {
    stopifnot(v %in% colnames(table7))
    t7a = arrange(table7, rank)
    if (! all(near(t7a[[v]],
                   table7_ans[[v]]))) { stop(paste(v, "is not correct")) }
}
### END HIDDEN TESTS
```

Problem 8

For the week with the highest fraction of missing arrival times in problem 6, generate a table `table8` which shows the total number of missing arrival delay values for each day of that week. Your table should have columns `year`, `month`, `day`, and `n_miss_arr_delay` and be sorted by year, month, and day.

In [15]:

```
### BEGIN SOLUTION
table8 = filter(flights4, week==filter(table7, rank==1)$week) %>% group_by(year, month, day) %>%
                summarize(n_miss_arr_delay=sum(is.na(arr_delay))) %>% arrange(year, month, day)
### END SOLUTION
```

In [16]:

```
stopifnot(exists("table8"))
### BEGIN HIDDEN TESTS
table8_ans = filter(flights4, week==filter(table7, rank==1)$week) %>% group_by(year, month, day) %>%
                summarize(n_miss_arr_delay=sum(is.na(arr_delay))) %>% arrange(year, month, day)
for (v in c('year', 'month', 'day', 'n_miss_arr_delay')) {
    stopifnot(v %in% colnames(table8))
    if (! all(near(table8[[v]],
                   table8_ans[[v]]))) { stop(paste(v, "is not correct")) }
}
### END HIDDEN TESTS
```

Problem 9

Two days in `table8` should stand out from the rest. To figure out what is going on, we will pull in some climate data using the package `rnoaa`. You do not need to install this package; I have run the command and saved the output for you.

In [17]:

```
load('nyc_weather.RData')
# this is the output of:
# library(rnoaa)
# # (daily weather data from Central Park)
# nyc_weather = as.tibble(meteo_pull_monitors("USW00094728",
#                                   date_min="2013-01-01",
#                                   date_max="2013-12-31"))
# nyc_weather = mutate(nyc_weather, year=year(date), month=month(date), day=day(date)) %>%
#               select(nyc_weather, year, month, day, prcp, snow, tmax)
# save(nyc_weather, file="nyc_weather.RData")
```

The table `nyc_weather` contains daily weather observations from Central Park in New York City. We need to merge this information in with the table you obtained in the previous exercise. Since we have not covered merges yet, the command to do so is provided for you:

In [18]:

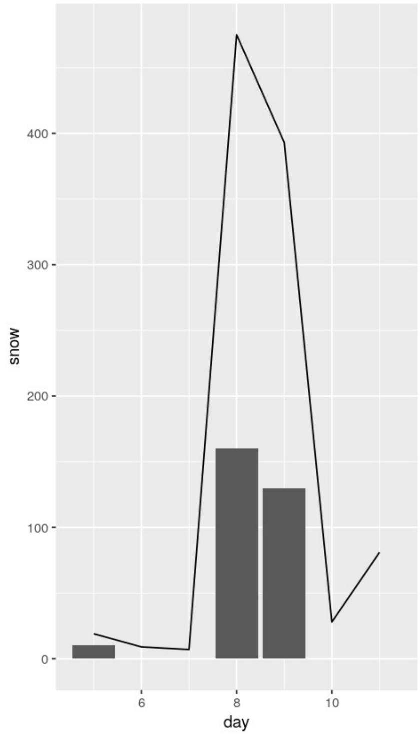
```
table9 = inner_join(table8, nyc_weather)
```

```
Joining, by = c("year", "month", "day")
```

Use `table9` to generate a bar plot of total snowfall for each day of the week in question. Add to this plot a line showing the total number of flights with missing arrival times for each day.

In [19]:

```
### BEGIN SOLUTION
ggplot(table9) +
       geom_col(aes(x=day, y=snow)) +
       geom_line(aes(x=day, y=n_miss_arr_delay))
### END SOLUTION
```



Problem 10

In your own words, summarize your findings from problems 8-10. What do NAs for arrival and departure delay likely represent in these data?

Your answer here...