Answers to train data challenge

Zahra Arjmandi Lari, Tom Stafford

January 24, 2023

Although we've created an example with artificial data, we think tackling this problem will teach you a number of fundamental data science skills. These include:

- dealing with under-specified questions
- data imports and preprocessing ("data wrangling")
- filtering and slicing data
- generating summary statistics using different basis-units [maybe change it to "approaches" or sth else?]

If this last item doesn't make much sense to you, you may be exactly the sort of person who would benefit from working through this challenge.

[detailed work through of a possible analysis of this data, and how it illustrates the importance of understanding different basis units, to come here]

First importing the needed libraries. We need to install them in case they were not installed before (uncomment the commented lines if you get an error by running `library(package_name)`)

```
#install.packages("dplyr")
#install.packages("ggplot2")
#install.packages("lubridate")
library("dplyr")
library("ggplot2")
library(lubridate)
```

Then, import the data:

```
df <- read.csv('train_data.csv')
```

Take a look at first five rows of the data

```
head(df)
```

```
##       X train_id route arrival_time passenger_id       date
## 1  1649      t10     A        09:52      p787266 2022-11-24
## 2 17888      t10     A        12:02      p229570 2022-11-22
## 3 55150      t09     A        19:36      p467365 2022-11-18
## 4   319      t10     A        07:42      p995170 2022-11-24
## 5 15034      t02     E        18:16      p341943 2022-11-23
## 6 25525      t10     A        09:52      p284688 2022-11-21
```

And take a look at the data in each column:

```
str(df)
```

```
## 'data.frame':    60168 obs. of  6 variables:
##  $ X           : int  1649 17888 55150 319 15034 25525 36869 43313 174 29866 ...
##  $ train_id    : chr  "t10" "t10" "t09" "t10" ...
##  $ route       : chr  "A" "A" "A" "A" ...
```

```
##  $ arrival_time: chr  "09:52" "12:02" "19:36" "07:42" ...
##  $ passenger_id: chr  "p787266" "p229570" "p467365" "p995170" ...
##  $ date        : chr  "2022-11-24" "2022-11-22" "2022-11-18" "2022-11-24" ...
```

It seems most columns include categorical variable. Let's see what are their unique values:

```
unique(df$train_id)
```

```
##  [1] "t10" "t09" "t02" "t06" "t01" ""    "t07" "t08" "t03" "t05" "t04"
```

```
unique(df$route)
```

```
## [1] "A" "E" "B" "D" "C" ""
```

```
unique(df$arrival_time)
```

```
##  [1] "09:52" "12:02" "19:36" "07:42" "18:16" "16:41" "10:57" "22:33" "15:16"
## [10] "07:23" "22:51" "13:06" ""      "14:11" "16:21" "23:35" "23:42" "09:12"
## [19] "05:32" "16:27" "16:39" "05:28" "20:07" "20:41" "06:15" "18:31" "22:17"
## [28] "17:26" "08:47" "09:43" "20:04" "14:44" "06:37" "21:46" "06:56" "11:05"
## [37] "11:01" "05:34" "13:11" "12:49" "14:38" "21:53"
```

It seems like we have multiple trains which run in 5 routes and arrived to the destination at different time throughout the day. Plus, it seems there are some missing values in this data set. Let's see how many data points have missing values.

```
sum(is.na(df))
```

```
## [1] 0
```

It looks like we cannot handle the missing values like " " with `na` functions in r. But we can search for empty elements in the data set and replace them with NA.

```
df <- replace(df, df=='', NA)
sum(is.na(df))
```

```
## [1] 2989
```

The data set has ~3000 missing values. Let's see what percentage of the data would that be.

```
sum(is.na(df)) / nrow(df) * 100
```

```
## [1] 4.967757
```

Around 5%. Let's get rid of the rows with a missing value, then!

```
df = na.omit(df)
#check if everything is ok now
sum(is.na(df))
```

```
## [1] 0
```

Since we wanted to look at every single journey and none of the variables show us a specific journey on their own, we need o combine them:

```
journeys <- df %>%
  group_by(train_id, route, arrival_time, date) %>%
  summarise(passenger_count = n())
journeys
```

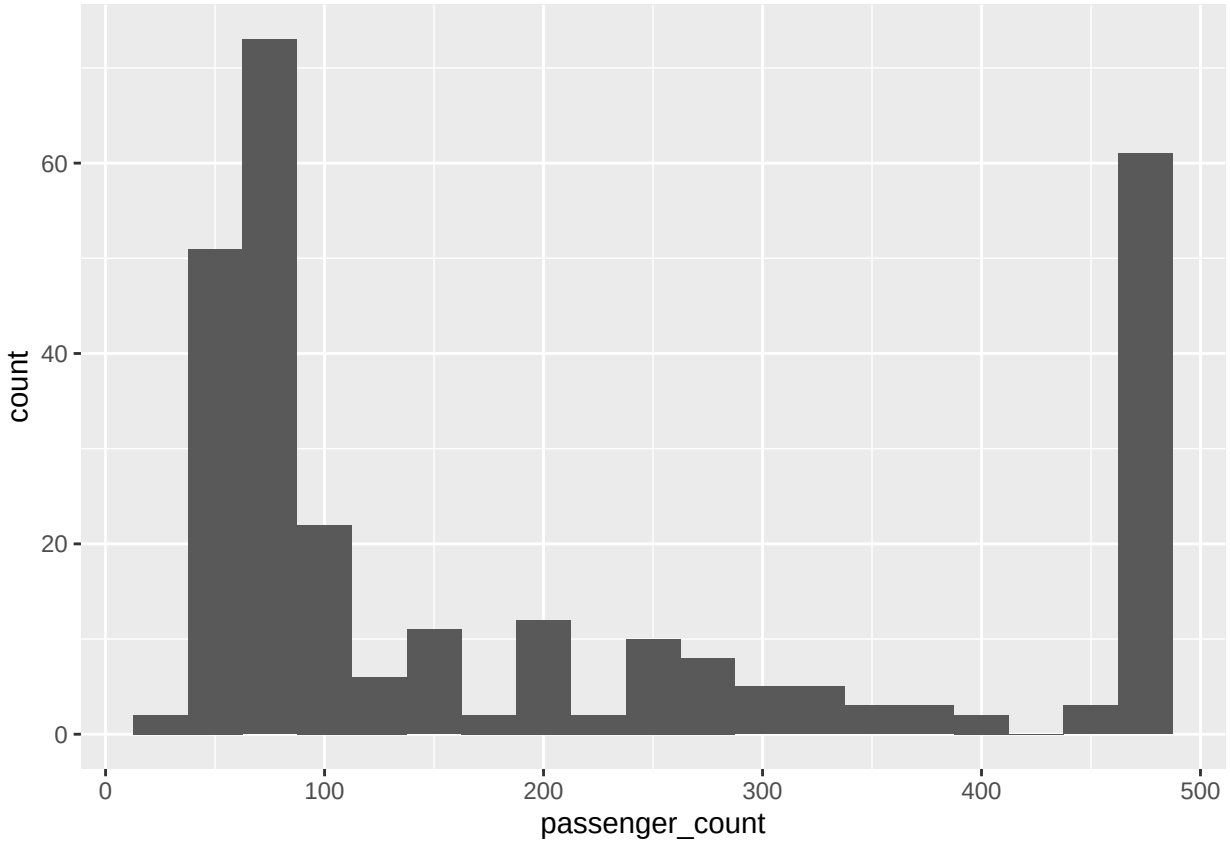
```
## # A tibble: 281 x 5
## # Groups:   train_id, route, arrival_time [41]
##    train_id route arrival_time date       passenger_count
```

```
##    <chr>    <chr> <chr>        <chr>                <int>
##  1 t01      D     06:56        2022-11-18              62
##  2 t01      D     06:56        2022-11-19             312
##  3 t01      D     06:56        2022-11-20             209
##  4 t01      D     06:56        2022-11-21              44
##  5 t01      D     06:56        2022-11-22              45
##  6 t01      D     06:56        2022-11-23              60
##  7 t01      D     06:56        2022-11-24             112
##  8 t01      D     22:33        2022-11-18              77
##  9 t01      D     22:33        2022-11-19             158
## 10 t01      D     22:33        2022-11-20             290
## # ... with 271 more rows
```

Each row in `journeys` is a unique combination of `trin_id`, `route`, `arrival_time` and `date`. `passenger_count` variable shows the count of that specific combination, which is the number of passengers in that journey.

Now that we have number of passengers per journey, we can take a look at its distribution.

```
ggplot(journeys, aes(x = passenger_count)) +
  geom_histogram(binwidth = 25)
```



There are a lot of trains which run near empty, and a lot of them which are full. Let's extract summary statistics:

```
summary(journeys$passenger_count)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    36.0    68.0    96.0   203.7   355.0   487.0
```

based on the number of passengers per journey, we can see that the average number of passenger per journey is 214. Given that the maximum number of passengers in a journey is 500 (the capacity of a train) and the median of the passenger count is 102, more that half of the trains are more than half empty. In fact we can check out how many journeys run with less than 250 passengers.

```
sum(journeys$passenger_count < 250)
```

```
## [1] 190
```

179 trains were less than half empty; what is the frequency of half empty trains?

```
sum(journeys$passenger_count < 250) / length(journeys$passenger_count)
```

```
## [1] 0.6761566
```

63% ! it seems that the argument of train companies are true about half train running half empty! But what about how passenger feel? Do passengers have a point when complaining about crowded trains?

Let's add a column to the original data frame `df`, showing the number of passenger in each journey.

```
df <- df %>%
  group_by(arrival_time, date) %>%
  mutate(passengerc_count = n())
head(df)
```

```
## # A tibble: 6 x 7
## # Groups:   arrival_time, date [6]
##       X train_id route arrival_time passenger_id date       passengerc_count
##   <int> <chr>    <chr> <chr>        <chr>        <chr>                 <int>
## 1  1649 t10      A     09:52        p787266      2022-11-24              469
## 2 17888 t10      A     12:02        p229570      2022-11-22               37
## 3 55150 t09      A     19:36        p467365      2022-11-18              476
## 4   319 t10      A     07:42        p995170      2022-11-24              469
## 5 15034 t02      E     18:16        p341943      2022-11-23              471
## 6 25525 t10      A     09:52        p284688      2022-11-21              481
```

And define a busy train to be 80% full.

```
max_capacity = 500
busy_train_percent = .8
```

To calculate the frequency of passengers commuting with a "busy" train, we can write:

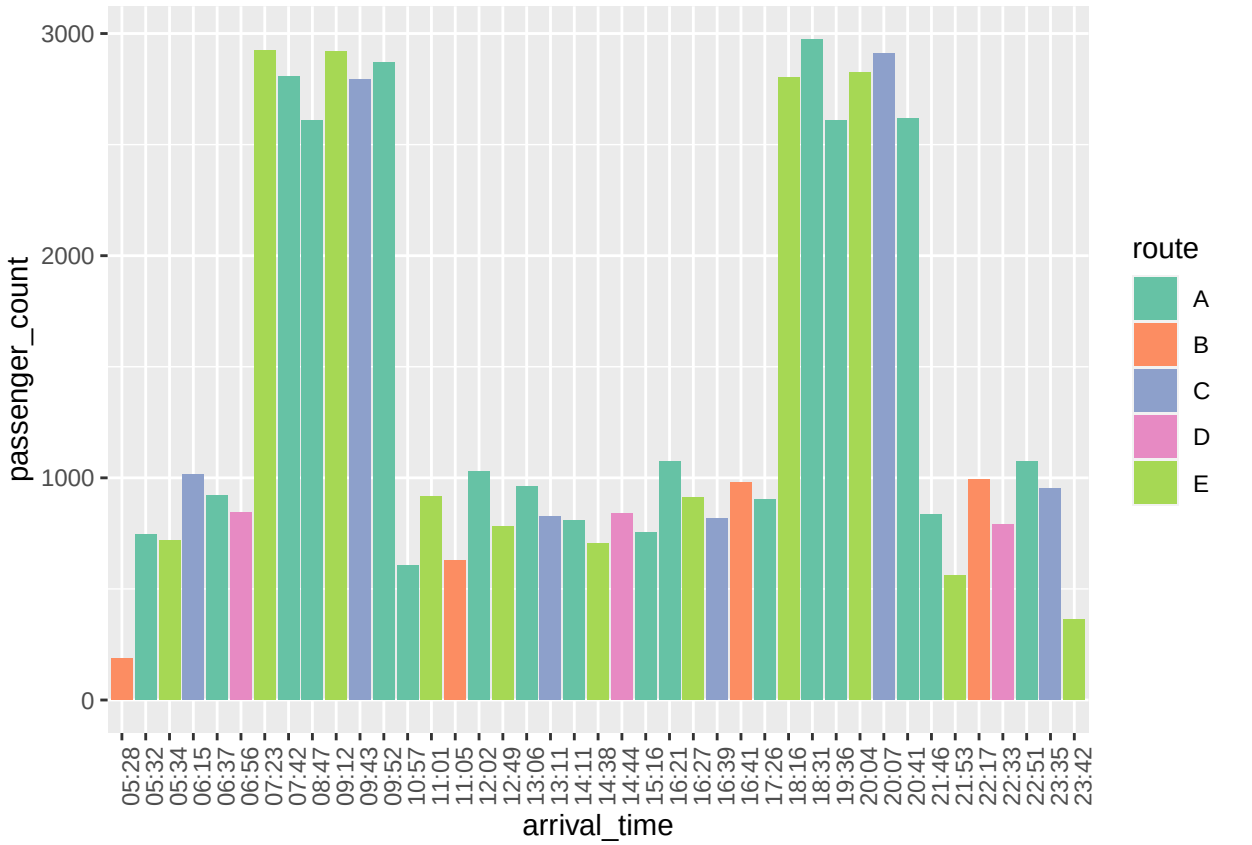
```
sum(df$passengerc_count > busy_train_percent * max_capacity) /
  length(df$passengerc_count)
```

```
## [1] 0.5452258
```

54% of passengers are commuting with busy trains! It seems the commuters also have a point!

We can also make a bar plot from crowdedness of the trains, based on the time they have arrived.

```
ggplot(journeys, aes(x = arrival_time, y = passenger_count, fill = route)) +
  geom_bar(stat = "identity") +
  theme(axis.text.x = element_text(angle = 90, vjust = 1, hjust=1)) +
  scale_fill_brewer(palette="Set2")
```



See that there are two pick hours during the day. One in the morning and one at night. That's why people feel they are commuting in busy trains. A lot of people commute during rush hours! Let's check which dates we have data for.

```
sort(unique(df$date))
```

```
## [1] "2022-11-18" "2022-11-19" "2022-11-20" "2022-11-21" "2022-11-22"
## [6] "2022-11-23" "2022-11-24"
```

We have 7 consequetive dates. But is the above pattern the same in the weekends and weekdays? Let's try to separate the data from weekends!

The function `wday()` gives us which day of the week a date is.

```
df$date[1]
```

```
## [1] "2022-11-24"
```

```
wday(df$date[1], label=TRUE, abbr=FALSE)
```

```
## [1] Thursday
## 7 Levels: Sunday < Monday < Tuesday < Wednesday < Thursday < ... < Saturday
```

We can also handle the days of the week numerically. It would make our lives easier in the code! It seems like the first day is Sunday, and the last day is Saturday. Let's check if that is true for a date which is Saturday.

```
day = "2022-11-19"
wday(day)
```

```
## [1] 7
```

Now, we need to make a column for weekdays in the `journeys` dataframe.

```
journeys$week_day = wday(journeys$date)
head(journeys)
```

```
## # A tibble: 6 x 6
## # Groups:   train_id, route, arrival_time [1]
##   train_id route arrival_time date       passenger_count week_day
##   <chr>    <chr> <chr>        <chr>                <int>    <dbl>
## 1 t01      D     06:56        2022-11-18              62        6
## 2 t01      D     06:56        2022-11-19             312        7
## 3 t01      D     06:56        2022-11-20             209        1
## 4 t01      D     06:56        2022-11-21              44        2
## 5 t01      D     06:56        2022-11-22              45        3
## 6 t01      D     06:56        2022-11-23              60        4
```

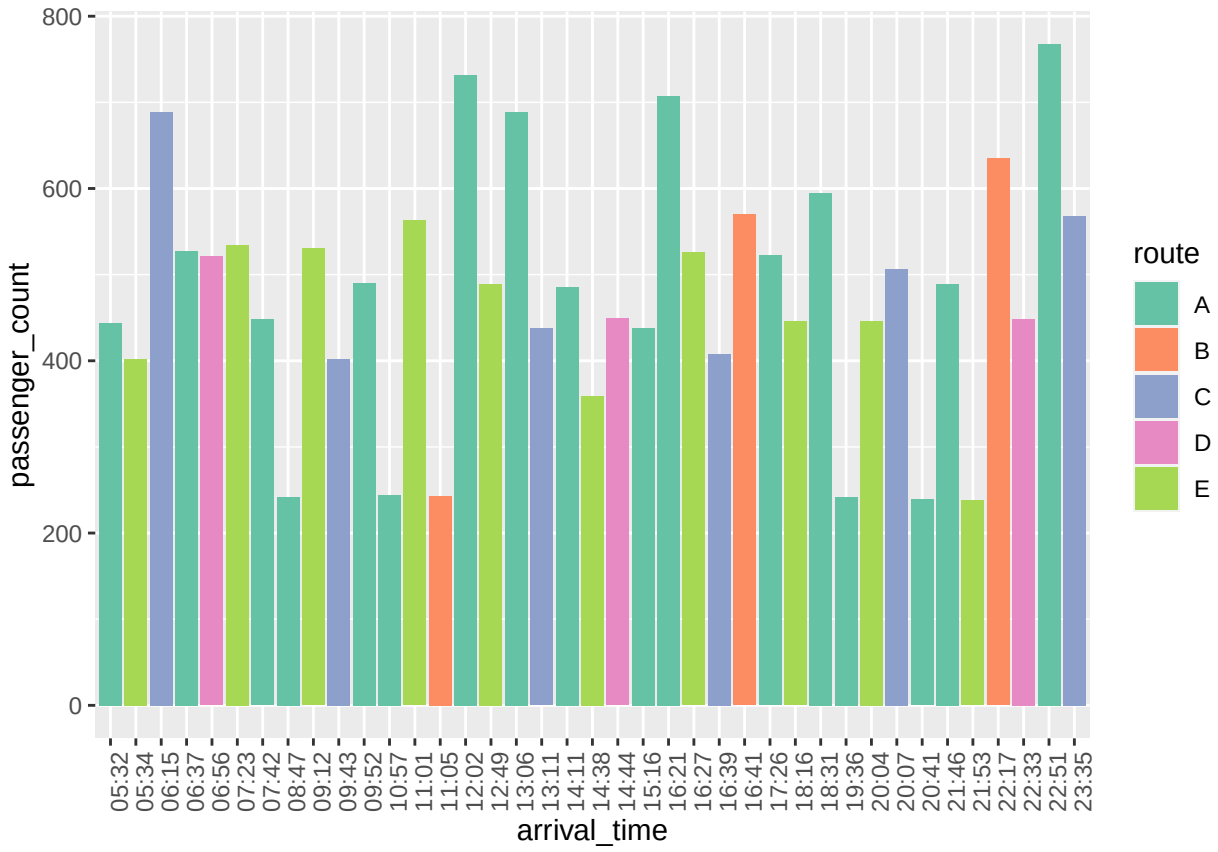
Now we filter the weekends by slicing through the `journeys` dataframe.

```
weekend_journeys = journeys[journeys$week_day == 1 | journeys$week_day == 7,]
head(weekend_journeys)
```

```
## # A tibble: 6 x 6
## # Groups:   train_id, route, arrival_time [3]
##   train_id route arrival_time date       passenger_count week_day
##   <chr>    <chr> <chr>        <chr>                <int>    <dbl>
## 1 t01      D     06:56        2022-11-19             312        7
## 2 t01      D     06:56        2022-11-20             209        1
## 3 t01      D     22:33        2022-11-19             158        7
## 4 t01      D     22:33        2022-11-20             290        1
## 5 t02      E     07:23        2022-11-19             162        7
## 6 t02      E     07:23        2022-11-20             372        1
```

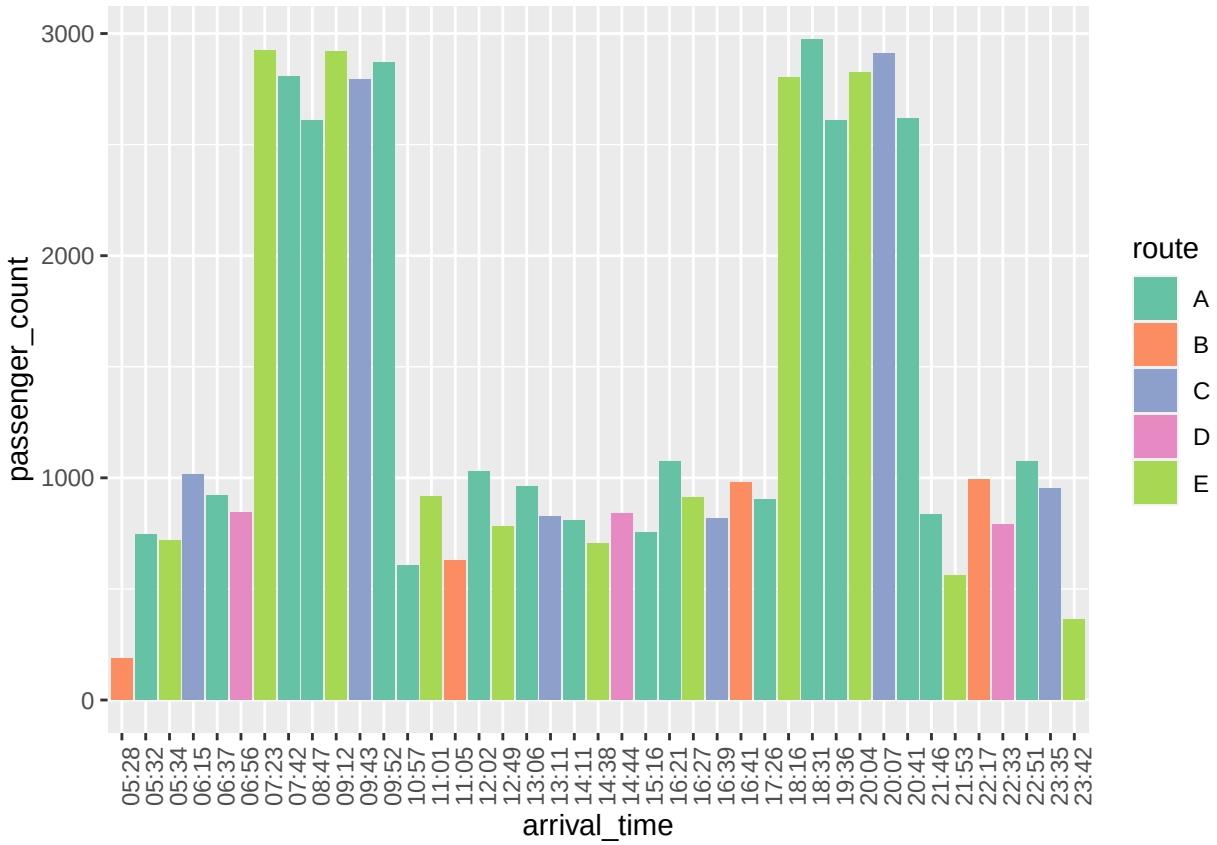
And make a bar plot for the weekend journeys.

```
ggplot(weekend_journeys, aes(x = arrival_time, y = passenger_count, fill = route)) +
  geom_bar(stat = "identity") +
  theme(axis.text.x = element_text(angle = 90, vjust = 1, hjust=1)) +
  scale_fill_brewer(palette="Set2")
```



No rush hour patterns! What about weekdays?

```
weekday_journeys = journeys[journeys$week_day != 1 | journeys$week_day != 7,]
ggplot(weekday_journeys, aes(x = arrival_time, y = passenger_count, fill = route)) +
  geom_bar(stat = "identity") +
  theme(axis.text.x = element_text(angle = 90, vjust = 1, hjust=1)) +
  scale_fill_brewer(palette="Set2")
```



The rush hour pattern is present!

Hope you have enjoyed! :)