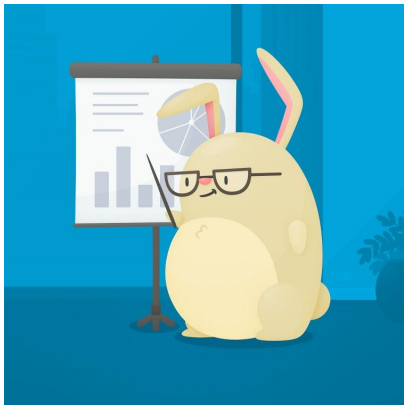
# Senior Data Scientist - Air

## Candidate Assignment



Thanks for your interest in a data science position at Hopper!  As the next step in your interview process we would like you to complete the 4 exercises below.  Please calibrate the depth of your answers such that you spend about 1 hour total on this work.

We use this homework to gauge how you solve a range of technical problems that require both lite coding and quantitative thinking.  You can use the language of your choice to complete these questions (though Hopper is most familiar with Python and R).  Feel free to use any resources you need to solve these questions, so long as you complete and present your own work.

Submit your answers in a separate document (Jupyter notebooks or RMarkdown are both great!) and make sure to give us any instructions needed for running the code sections. We look forward to seeing your work!

## Exercise 1 - Programming

Given the table of airports and locations (in latitude and longitude) below, write a function that takes an airport code as input and returns the airports listed from nearest to furthest from the input airport. Use only the basic libraries for the language of your choice (using sorting functions/methods provided by the standard library is definitely fine).

| Airport Code | Lat | Long |
|---|---|---|
| CDG | 49.0128 | 2.5500 |
| CHC | -43.4894 | 172.5320 |
| DYR | 64.7349 | 177.7410 |
| EWR | 40.6925 | -74.1687 |
| HNL | 21.3187 | -157.9220 |
| OME | 64.5122 | -165.4450 |
| ONU | -20.6500 | -178.7000 |
| PEK | 40.0801 | 116.5850 |

Airport data is ex1_table.csv

# Exercise 2 - Testing

*Note:* please use any of your favorite packages/libraries for this section of the homework

One of Hopper's innovative products is "Price Freeze" which allows users to freeze a price for a period of time before purchasing the ticket.

Suppose we are running a test comparing the current best model for pricing a Price Freeze (the Champion) and a new model we think might be better (the Challenger). We run the test showing these two different variants to our users but we realize there is an issue! The model is being shown to different numbers of people on different mobile devices (iOS and Android) and some of the users are also seeing a discount being offered. This makes the results of the test a bit hard to interpret. Given this table of information about our test:

| variant | device_type | discount | total_views | price_freezes |
|---|---|---|---|---|
| Challenger | android | FALSE | 6192 | 183 |
| Challenger | android | TRUE | 315 | 15 |
| Challenger | iOS | FALSE | 6718 | 330 |
| Challenger | iOS | TRUE | 1994 | 199 |
| Champion | android | FALSE | 1023 | 28 |
| Champion | android | TRUE | 48 | 2 |
| Champion | iOS | FALSE | 6704 | 265 |
| Champion | iOS | TRUE | 2006 | 155 |

Data is ex2_table.csv

Where

- variant describes which model was used
- device_type tells us which mobile OS that app is running on
- discount is whether or not the users in this group received a discount
- total_views is how many users saw the option to freeze
- price_freezes is how many users chose to price freeze

Answer the following questions about the experiment, make sure to show any code, math, or reasoning you have for choosing your answer.

1. What is the probability that the Challenger is the superior model?
2. Based on your answer to number 1, would you be comfortable deciding yes/no on whether or not to change models?
3. If we decide to switch exclusively to the Challenger model for our iOS users, do we have a reasonable chance at getting 500 prices freezes in the first 10,000 views? what about 600?

## Exercise 3 - Representation

*Note:* Don't worry about writing code in this section, you can just describe any transformations of the data you would perform. Your description should be clear enough that a data scientist reading this would know how to implement your solution if necessary.

Here is an example of a single user's searches in our app:

| user_id | trip_type | first_searched_dt | origin | destination | departure_date | return_date | stay |
|---|---|---|---|---|---|---|---|
| 03787 | round_trip | 2018-03-09 21:31:01 | city/WAS | airport/PUJ | 2018-08-01 | 2018-08-06 | 5 |
| 03787 | round_trip | 2018-03-09 21:32:33 | airport/DCA | airport/PUJ | 2018-08-01 | 2018-08-08 | 7 |
| 03787 | one_way | 2018-03-10 08:59:04 | city/WAS | airport/MIA | 2018-08-01 | NA | NA |
| 03787 | one_way | 2018-03-10 08:59:54 | city/WAS | airport/MIA | 2018-08-01 | NA | NA |
| 03787 | one_way | 2018-03-10 09:01:03 | city/WAS | airport/MIA | 2018-08-02 | NA | NA |
| 03787 | one_way | 2018-03-10 09:01:15 | city/WAS | airport/MIA | 2018-08-02 | NA | NA |
| 03787 | one_way | 2018-03-10 09:23:49 | airport/BWI | airport/LAX | 2018-06-29 | NA | NA |
| 03787 | round_trip | 2018-03-10 09:24:29 | airport/BWI | airport/LAX | 2018-06-29 | 2018-07-02 | 3 |
| 03787 | round_trip | 2018-03-10 09:27:44 | city/WAS | airport/LAX | 2018-06-29 | 2018-07-02 | 3 |
| 03787 | round_trip | 2018-03-10 09:28:41 | airport/DCA | airport/LAX | 2018-06-29 | 2018-07-02 | 3 |
| 03787 | one_way | 2018-03-14 09:12:55 | airport/DCA | airport/MIA | 2018-08-02 | NA | NA |
| 03787 | round_trip | 2018-03-14 09:13:34 | airport/MIA | airport/PUJ | 2018-08-02 | 2018-08-06 | 4 |
| 03787 | round_trip | 2018-03-19 12:35:35 | city/WAS | airport/LAX | 2018-06-29 | 2018-07-02 | 3 |
| 03787 | round_trip | 2018-03-21 10:53:39 | city/WAS | airport/FLL | 2018-08-02 | 2018-08-07 | 5 |
| 03787 | round_trip | 2018-03-23 18:26:09 | city/WAS | airport/PUJ | 2018-08-02 | 2018-08-06 | 4 |
| 03787 | round_trip | 2018-03-23 18:32:12 | city/WAS | city/ORL | 2018-09-14 | 2018-09-17 | 3 |
| 03787 | round_trip | 2018-03-23 18:32:46 | city/WAS | city/ORL | 2018-09-14 | 2018-09-17 | 3 |
| 03787 | round_trip | 2018-03-23 20:12:17 | city/WAS | airport/LAX | 2018-06-29 | 2018-07-02 | 3 |
| 03787 | round_trip | 2018-03-23 20:16:22 | airport/IAD | airport/PUJ | 2018-08-02 | 2018-08-06 | 4 |
| 03787 | round_trip | 2018-03-23 20:17:01 | airport/BWI | airport/PUJ | 2018-08-02 | 2018-08-06 | 4 |
| 03787 | round_trip | 2018-03-23 20:17:49 | airport/DCA | airport/PUJ | 2018-08-02 | 2018-08-06 | 4 |
| 03787 | round_trip | 2018-03-24 12:38:01 | airport/BWI | airport/BZE | 2018-10-04 | 2018-10-08 | 4 |
| 03787 | round_trip | 2018-03-25 13:37:46 | airport/RIC | airport/FLL | 2018-08-04 | 2018-08-11 | 7 |
| 03787 | round_trip | 2018-04-12 03:25:19 | airport/BWI | airport/PUJ | 2018-08-03 | 2018-08-07 | 4 |
| 03787 | round_trip | 2018-04-12 03:27:34 | airport/IAD | airport/PUJ | 2018-08-03 | 2018-08-07 | 4 |
| 03787 | round_trip | 2018-04-12 03:28:37 | airport/IAD | airport/PUJ | 2018-08-03 | 2018-08-07 | 4 |
| 03787 | round_trip | 2018-04-16 15:48:37 | airport/BWI | airport/ATL | 2018-08-03 | 2018-08-07 | 4 |
| 03787 | round_trip | 2018-04-16 15:49:23 | city/WAS | airport/ATL | 2018-08-02 | 2018-08-07 | 5 |
| 03787 | round_trip | 2018-04-19 13:04:17 | city/WAS | airport/LAX | 2018-06-28 | 2018-07-02 | 4 |

Data is ex3_table.csv

We want to create a mathematical model of a user's "trip" which can be described as a collection of searches. This requires us to represent this non-numeric data such that we can draw quantitative conclusions.

1. How would you transform this collection of searches into a numeric vector representing a

trip?

- Assume that we have hundreds of thousands of users and we want to represent all of their trips this way.
- We ideally want this to be a general representation we could use in multiple different modeling projects, but we definitely care about finding similar trips.

2. How, precisely, would you compare two trips to see how similar they are?
3. What information do you feel might be missing from data above that would be helpful in improving your representation?

## Exercise 4 - Experiments and Data Collection

An essential job of Hopper data science is coming up with new models for our products and testing to both see which models are better and to learn more about our products to help us better understand how to improve our models.

One of the core features of the Hopper app is that it advises users whether to buy a ticket now or wait for the price to go down and book later.  But what if our buy recommendation is wrong and the price in fact drops after the user books on Hopper?  To lower the pain when this happens Hopper has introduced “Price Drop” which refunds users a certain % of the fare difference if the price drops *after* they book.

If you were the data scientist in charge of this project what information would you want to track to decide whether this feature is successful? What would you track to determine how to improve this feature?