King's Super Math Saga experiment

As a part of your application for Data Science roles at King, we would like to invite you to complete the following practical assignment. You can spend as much or as little time on it as you wish, although usually we see an investment of ca. 3-6 hours to complete. This does not need to be in one sitting, but please note that we need you to send us back your solution within one week from reception.

Send us a short report (max 3 pages) with your findings and supporting plots, and provide the code used for the analysis. Markdown or notebook is the preferred report format. All completed assignments are reviewed by our Data Scientists. Please be ready to eventually discuss it in an interview.

Problem statement

Last year, we released Super Math Saga, a mathematical quiz mobile game. The format is the same as other Saga games - players need to beat levels and progress through a map. At every level, players are faced with a math question they need to answer correctly to pass to the next level. The game is Free to Play meaning that players can download and play the game for free but can optionally buy hints on the levels.

Some time ago we ran an experiment (A/B Test) on the game to improve it's performance. The experiment offered two different game experiences that we call A and B, **group A** being the control group where the experience is kept as is, and **group B** being the experiment group that is exposed to the new experience. We set the assignment process to distribute players among the two groups: 80% to group A (control) and 20% to group B (test). The experiment ran from 2017-05-04 to 2017-05-22.

The question is: how would we determine which of the experiences make Super Math Saga a better game?

Tools

We recommend to do the analysis in R or Python but you are free to use your preferred tools, the important part is readable and reproducable code. If you are using a local environment, you can access and query the data using one of the BigQuery libraries (Python, R, Other) with the provided project ID. If you don't have a local setup, you can set up a Datalab instance (Jupyter Notebook) in the Google Cloud Project.

Two tables are provided in the **abtest** database in Google BigQuery:

The **assignment** table contains players assigned to the A/B test and attributes related to each player.

- playerid: Unique numeric identifier for each player
- abtest_group: The group the player was assigned to (A or B)
- assignment_date: The date when the player was assigned to the test
- install_date: The date when the player installed the game
- conversion_date: The date when the player made their first purchase

The **activity** table contains player activity for each day a player was active.

- playerid: Unique numeric identifier for each player
- activity_date: The date of activity
- purchases: Number of purchases made this day
- gameends: Number of gamerounds played this day

Instructions

1. Make sure you can access the provided Google Cloud Project. You can browse the data through the GUI by going to the *BigQuery* tab.
2. Set up the required BigQuery library above for your preferred language, *or* spin up a Datalab instance if you prefer the browser experience.
3. Do your thing! Based on your analysis, which recommendation would you provide to the game team?
4. Send your report, code and plots back to us.

Example topics for the analysis

- Which metrics would you consider important to look at in this case? What can you say about them in this A/B test?
- Look at daily installs and assignments over the duration of the A/B test. Do you expect to see any difference between groups?
- Compute the average number of gamerounds per player for each group. Can you confidently state which group has higher engagement?
- Do different types of players react differently to the treatment?
- What other aspects do you think would be valuable to consider for analysing an A/B test?
- What type of change to the game do you think was tested here?

Notes

- Make sure to disable *Legacy SQL* when running queries, it is BigQuery's SQL dialect and can give different results from what you are expecting from standard SQL. This can be done in following ways:
  - **In queries**: Prefix your queries with `#standardSQL`
  - **Using libraries**: Set `use_legacy_sql=false` or similar flag when using the libraries
  - **Using BigQuery GUI**: Uncheck `Use Legacy SQL` under Options
- If you happen to encounter problems with the libraries, it is OK to run your queries through the BigQuery GUI, save the results as csv and import it in your code. Just remember to include the underlying queries in your code for reproducability.
- For python, if you want to connect to bigquery the setup we have found works best is:
  1. **Install Cloud SDK**: MacOS: https://cloud.google.com/sdk/docs/quickstart-macos Windows: https://cloud.google.com/sdk/docs/quickstart-windows Linux: https://cloud.google.com/sdk/docs/quickstart-linux
  2. Run `gcloud init`
  3. Run `gcloud auth application-default login`
  4. **Install python client library**

For any arising questions or issues, please contact datasciencetechtest@king.com.