Introduction

One of the well-known games and one of the first freemium games to generate over USD 1 Billion, Candy Crush Saga is a mobile game developed by King, also part of Activision and Blizzard company that's played by millions of people across the world. Candy Crush has more than 3000 levels and is added new levels every week. In 2020 it had 8900 levels, And with that many levels, the game developer needs to release a level difficulty just right. If it's too easy, the game will be boring, and if it's too hard, the players become frustrated and quit playing. By Analyzing the data, we can recommend whether the developer needs to worry about the game too easy and hard.

The data set

The dataset we use is the sample of players who play Candy Crush back in 2014(https://www.kaggle.com/kingabzpro/candy-crush), and only from one episode. It has the following columns:

- **player_id**: a unique player id
- **dt:** the date
- **level**: The level number from the episode(range from 1 to 15)
- **num_attempts**: Number of level attempts for the player on that level and date.
- **num_success:** Number of level attempts that resulted in a success/win for the player on that level and date.

Now let's load the dataset and look at the first of a couple of rows.

```{r}
data <- read.csv("candy_crush.csv")

# Change the format to date format
data$dt <- as.Date(data$dt, format="%Y-%m-%d")

head(data)
```

Now that we have loaded the dataset, let's count how many players we have in the sample and how many day's worths of data we have.

```
# Count and display the number of unique players
print("Number of players:")
length(unique(data$player_id))

# Display the date range of the data
print("Data period:")
range(data$dt)
```

```
[1] "Number of players:"
6814
[1] "Period for which we have data:."
"2014-01-01" "2014-01-07"
```

Empirical Analysis

Within each Candy Crush episode, there is a mix of more accessible and more challenging levels. Luck and individual skill make the number of attempts required to pass a level different from player to player. The assumption is that challenging levels require more attempts on average than easier ones. The harder a level is, *the lower* the probability of passing that level in a single attempt is.

A simple approach to model this probability is as a Bernoulli process. Bernoulli process is a finite or infinite sequence of binary random variables, so it is a discrete-time stochastic process that takes only two values, canonically 0 and 1. A binary outcome (you either win or lose) is characterized by a single parameter *pwin*: the probability of winning the level in a single attempt. This probability can be estimated for each level as:

$$p_{win} = \frac{\Sigma wins}{\Sigma attempts}$$

For example, let's say a level has been played 10 times and 2 of those attempts ended up in a victory. Then the probability of winning in a single attempt would be *pwin* = 4 / 10 = 40%. Now, let's compute the difficulty *pwin* separately for each of the 15 levels.

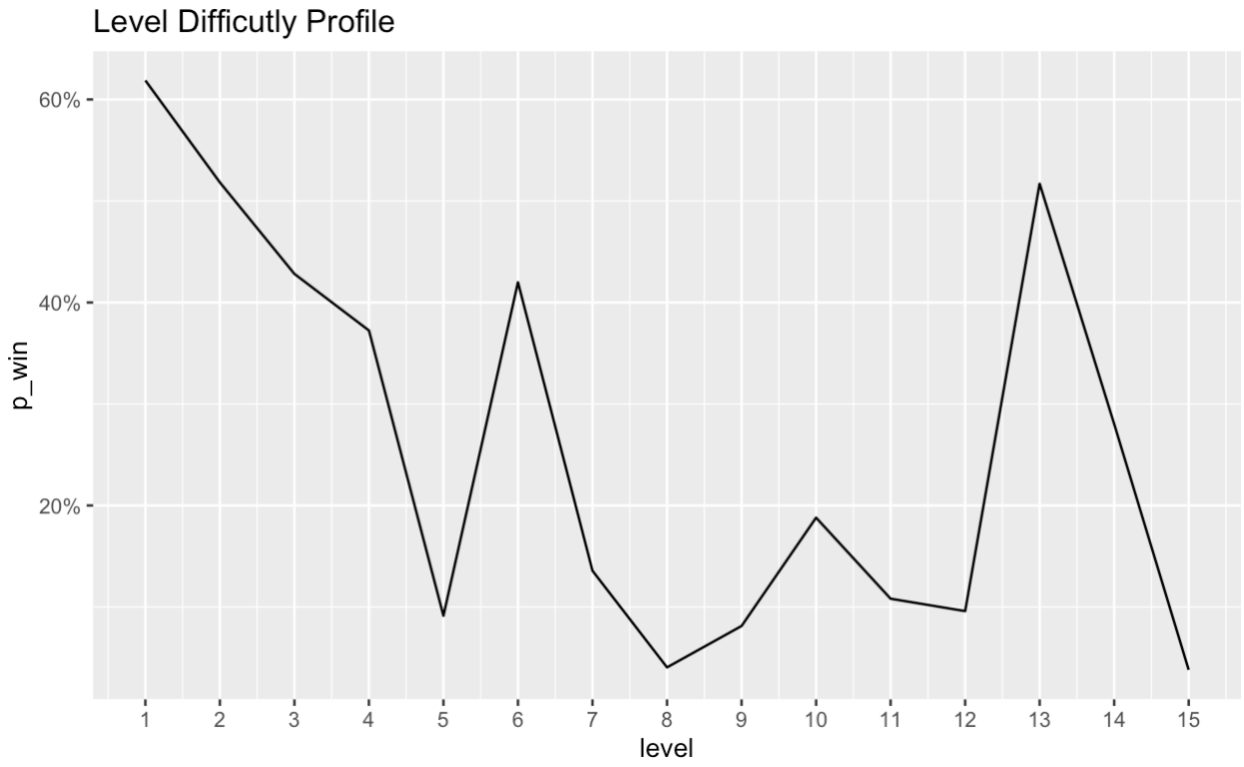
```{r}
# Calculating level difficulty
difficulty <- data %>%
  group_by(level) %>%
  summarise(attempts = sum(num_attempts), wins = sum(num_success)) %>%
  mutate(p_win = wins / attempts)

# Printing out the calculated difficulty
difficulty
```

level <int>	attempts <int>	wins <int>	p_win <dbl>
11	5575	603	0.10816143
12	6868	659	0.09595224
13	1327	686	0.51695554
14	2772	777	0.28030303
15	30374	1157	0.03809179

Now we have difficulty for all the 15 levels in the episode. Keep in mind that, as we measure difficulty as the probability to pass a level in a single attempt, a *lower* value (a smaller probability of winning the level) implies a *higher* level of difficulty. Now that we have difficulty with the episode, we should plot it. Let's plot a line graph with the levels on the X-axis and the difficulty (*pwin*) on the Y-axis. We call this plot the *difficulty profile* of the episode.

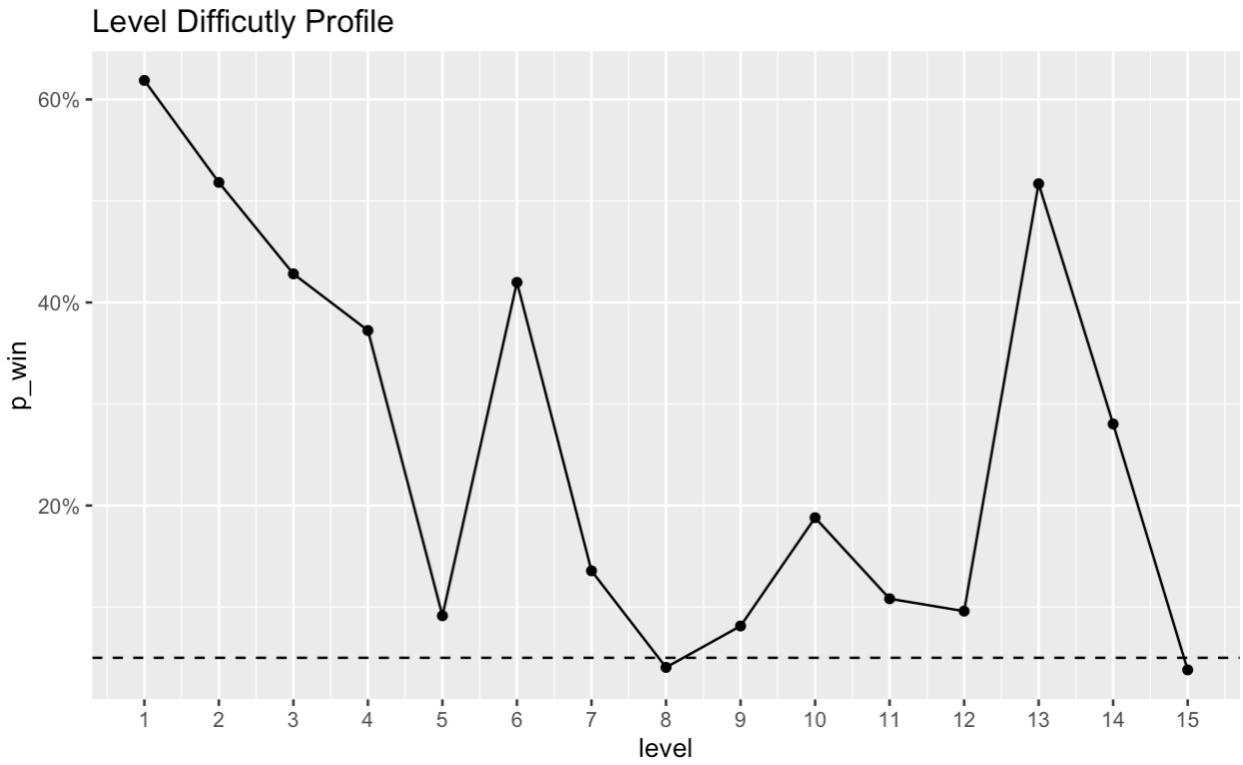
```{r}
# Plotting the level difficulty profile
difficulty %>%
ggplot(aes(x=level, y = p_win)) +
geom_line() +
scale_x_continuous(breaks=1:15) +
scale_y_continuous(label = scales::percent) +
  labs(title = "Level Difficutly Profile")
```



What constitutes a *hard* level is subjective. However, to keep things simple, we could define a difficulty threshold, say 5%, and label levels with *pwin* < 5% as *hard*. It's relatively easy to spot these challenging levels on the plot, but we can make the plot more friendly by explicitly highlighting the challenging levels.

```{r}
# Adding points and a dashed line
difficulty %>%
```

```
ggplot(aes(x=level, y = p_win)) +
geom_line() +geom_point() +
scale_x_continuous(breaks=1:15) +
scale_y_continuous(label = scales::percent)+
geom_hline(yintercept=0.05, linetype="dashed")+
  labs(title = "Level Difficutly Profile")
```



As Data Scientists, we should always report some measure of the uncertainty of any provided numbers. Maybe tomorrow, another sample will give us slightly different values for the difficulties? Here we will simply use the Standard Error(The standard error (SE) of a statistic (usually an estimate of a parameter) is the standard deviation of its sampling distribution or an estimate of that standard deviation):

$$\sigma_{error} \approx \frac{\sigma_{sample}}{\sqrt{n}}$$

Here *n* is the number of data points, and *σsample* is the sample standard deviation. For a Bernoulli process, the sample standard deviation is:

$$\sigma_{sample} = \sqrt{P_{win}(1 - P_{win})}$$

Therefore, we can calculate the standard error like this:

$$\sigma_{error} \approx \sqrt{\frac{P_{win}(1-P_{win})}{n}}$$

We already have all we need in the difficulty data frame! Every level has been played *n* number of times and we have their difficulty *pwin*. Now, let's calculate the standard error for each level.

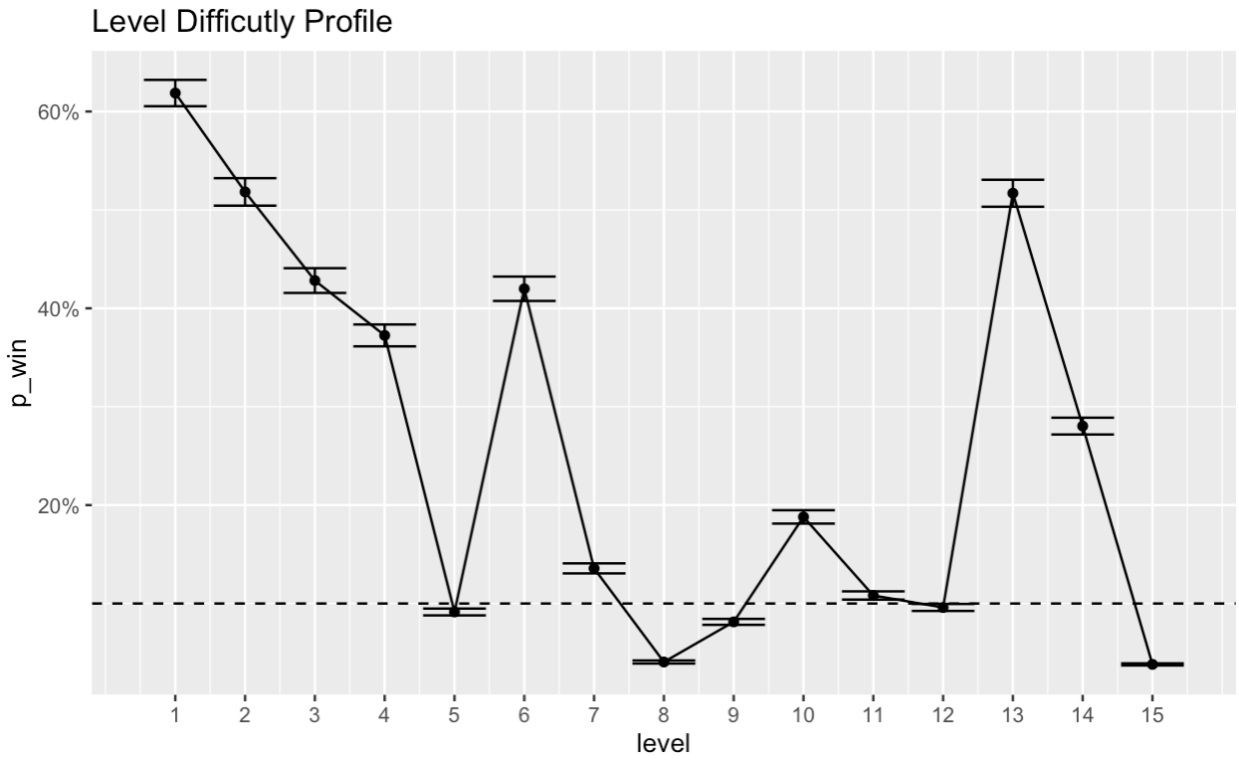
```{r}
# Computing the standard error of p_win for each level
difficulty <- difficulty %>%
group_by(level) %>%
mutate(error = sqrt(p_win * (1 - p_win) / attempts))

difficulty
```

level <int>	attempts <int>	wins <int>	p_win <dbl>	error <dbl>
1	1322	818	0.61875946	0.013358101
2	1285	666	0.51828794	0.013938876
3	1546	662	0.42820181	0.012584643
4	1893	705	0.37242472	0.011111607
5	6937	634	0.09139397	0.003459878
6	1591	668	0.41986172	0.012373251
7	4526	614	0.13566063	0.005089930
8	15816	641	0.04052858	0.001568008
9	8241	670	0.08130081	0.003010538
10	3282	617	0.18799512	0.006819983

Now that we have a measure of uncertainty for each levels' difficulty estimate, let's use *error bars* to show this uncertainty in the plot. We will set the length of the error bars to one standard error. Each error bar's upper limit and lower limit should be *pwin* + *σerror* and *pwin* - *σerror*, respectively.

```{r}
# Adding standard error bars
difficulty %>%
ggplot(aes(x=level, y = p_win)) +
geom_line() +geom_point() +
scale_x_continuous(breaks=1:15) +
scale_y_continuous(label = scales::percent)+
geom_hline(yintercept=0.1, linetype="dashed")+
geom_errorbar(aes(ymin=p_win - error, ymax=p_win + error)) +
  labs(title = "Level Difficutly Profile")
```



Conclusion

It looks like our difficulty estimates are pretty precise! Using this plot, a level designer can quickly spot the challenging levels and see if there seem to be too many challenging levels in the episode. One question a level designer might ask is: "How likely is it that a player will complete the episode without losing a single time?" Let's calculate this using the estimated level difficulties!

```{r}
# The probability of completing the episode without losing a single time
p <- prod(difficulty$p_win)

# Printing it out
p
```

```
[1] 9.447141e-12
```

Given the probability we just calculated, Our level designer **shouldn't worry** that a lot of players will complete the episode in one attempt because the likelihood of players will complete all of the episodes in one shot is 0.00000000000944714 or 1 in 105852141495.

Sources:
https://www.businessofapps.com/data/candy-crush-statistics/
https://www.kaggle.com/kingabzpro/candy-crush
https://en.wikipedia.org/wiki/Bernoulli_process
https://en.wikipedia.org/wiki/Standard_error