R Notebook

# Loading in packages  
  
library(readr)  
library(dplyr)

##   
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':  
##   
## filter, lag

## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

library(ggplot2)

1. The data set

The dataset we will use contains one week of data from a sample of players who played Candy Crush back in 2014. The data is also from a single episode, that is, a set of 15 levels. It has the following columns:

player\_id: a unique player id  
dt: the date  
level: the level number within the episode, 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.

The granularity of the dataset is player, date, and level. That is, there is a row for every player, day, and level recording the total number of attempts and how many of those resulted in a win.

Now, let’s load in the dataset and take a look at the first couple of rows.

# Reading in the data  
data <- read\_csv("datasets/candy\_crush.csv")

## Rows: 16865 Columns: 5  
## ── Column specification ────────────────────────────────────────────────────────  
## Delimiter: ","  
## chr (1): player\_id  
## dbl (3): level, num\_attempts, num\_success  
## date (1): dt  
##   
## ℹ Use `spec()` to retrieve the full column specification for this data.  
## ℹ Specify the column types or set `show\_col\_types = FALSE` to quiet this message.

# Printing out the first six rows  
head(data)

## # A tibble: 6 × 5  
## player\_id dt level num\_attempts num\_success  
## <chr> <date> <dbl> <dbl> <dbl>  
## 1 6dd5af4c7228fa353d505767143f5815 2014-01-04 4 3 1  
## 2 c7ec97c39349ab7e4d39b4f74062ec13 2014-01-01 8 4 1  
## 3 c7ec97c39349ab7e4d39b4f74062ec13 2014-01-05 12 6 0  
## 4 a32c5e9700ed356dc8dd5bb3230c5227 2014-01-03 11 1 1  
## 5 a32c5e9700ed356dc8dd5bb3230c5227 2014-01-07 15 6 0  
## 6 b94d403ac4edf639442f93eeffdc7d92 2014-01-01 8 8 1

1. Checking the data set

Now that we have loaded the dataset let’s count how many players we have in the sample and how many days worth of data we have.

# Count and display the number of unique players  
print("Number of players:")

## [1] "Number of players:"

data$player\_id %>% unique() %>% length()

## [1] 6814

# Display the date range of the data  
print("Period for which we have data:")

## [1] "Period for which we have data:"

range(data$dt)

## [1] "2014-01-01" "2014-01-07"

1. Computing level difficulty

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

A simple approach to model this probability is as a Bernoulli process

; as a binary outcome (you either win or lose) 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:

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 = 2 / 10 = 20%.

Now, let’s compute the difficulty pwin separately for each of the 15 levels.

# 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 level difficulty  
difficulty

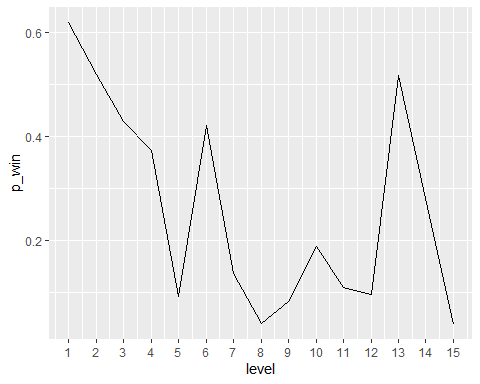
## # A tibble: 15 × 4  
## level attempts wins p\_win  
## <dbl> <dbl> <dbl> <dbl>  
## 1 1 1322 818 0.619   
## 2 2 1285 666 0.518   
## 3 3 1546 662 0.428   
## 4 4 1893 705 0.372   
## 5 5 6937 634 0.0914  
## 6 6 1591 668 0.420   
## 7 7 4526 614 0.136   
## 8 8 15816 641 0.0405  
## 9 9 8241 670 0.0813  
## 10 10 3282 617 0.188   
## 11 11 5575 603 0.108   
## 12 12 6868 659 0.0960  
## 13 13 1327 686 0.517   
## 14 14 2772 777 0.280   
## 15 15 30374 1157 0.0381

1. Plotting difficulty profile

Great! We now have the 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 difficulty.

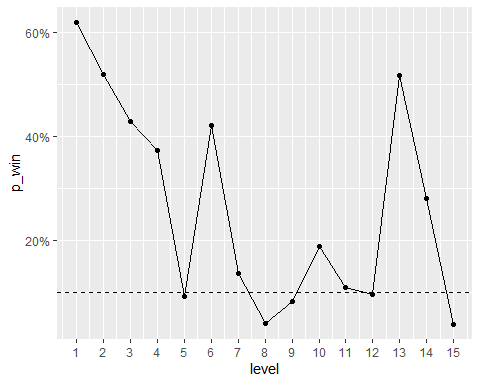
Now that we have the difficulty of the episode we should plot it. Let’s plot a line graph with the levels on the X-axis and the difficulty (pwin) on the Y-axis. We call this plot the difficulty profile of the episode.

# Plotting the level difficulty profile  
ggplot(difficulty, aes(x = level, y = p\_win)) +  
geom\_line() +  
scale\_x\_continuous(breaks = 1:15)

 6. Spotting hard levels

What constitutes a hard level is subjective. However, to keep things simple, we could define a threshold of difficulty, say 10%, and label levels with pwin < 10% as hard. It’s relatively easy to spot these hard levels on the plot, but we can make the plot more friendly by explicitly highlighting the hard levels.

# Adding points and a dashed line  
ggplot(difficulty, aes(x = level, y = p\_win)) +  
geom\_line() +  
scale\_x\_continuous(breaks = 1:15) +  
scale\_y\_continuous(label = scales::percent) +  
geom\_point() +  
geom\_hline(yintercept = 0.1, linetype = "dashed")

 7. Computing uncertainty

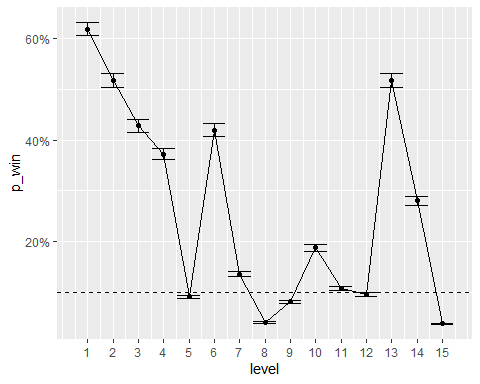
# Computing the standard error of p\_win for each level  
difficulty <- difficulty %>%  
 group\_by(level) %>%  
 mutate(error = sqrt(p\_win\*(1-p\_win)/attempts))  
  
difficulty

## # A tibble: 15 × 5  
## # Groups: level [15]  
## level attempts wins p\_win error  
## <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 1 1322 818 0.619 0.0134   
## 2 2 1285 666 0.518 0.0139   
## 3 3 1546 662 0.428 0.0126   
## 4 4 1893 705 0.372 0.0111   
## 5 5 6937 634 0.0914 0.00346  
## 6 6 1591 668 0.420 0.0124   
## 7 7 4526 614 0.136 0.00509  
## 8 8 15816 641 0.0405 0.00157  
## 9 9 8241 670 0.0813 0.00301  
## 10 10 3282 617 0.188 0.00682  
## 11 11 5575 603 0.108 0.00416  
## 12 12 6868 659 0.0960 0.00355  
## 13 13 1327 686 0.517 0.0137   
## 14 14 2772 777 0.280 0.00853  
## 15 15 30374 1157 0.0381 0.00110

1. Showing uncertainty

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. The upper limit and the lower limit of each error bar should then be pwin + σerror and pwin - σerror, respectively.

# Adding standard error bars  
ggplot(difficulty, aes(x = level, y = p\_win)) +  
geom\_line() +  
scale\_x\_continuous(breaks = 1:15) +  
scale\_y\_continuous(label = scales::percent) +  
geom\_point() +  
geom\_hline(yintercept = 0.1, linetype = "dashed") +  
geom\_errorbar(aes(ymin = p\_win - error, ymax = p\_win + error))

 9. A final metric

It looks like our difficulty estimates are pretty precise! Using this plot, a level designer can quickly spot where the hard levels are and also see if there seems to be too many hard 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!

# The probability of completing the episode without losing a single time  
p <- prod(difficulty$p\_win)  
  
# Printing it out  
p

## [1] 9.447141e-12

1. Should our level designer worry?

Given the probability we just calculated, should our level designer worry about that a lot of players might complete the episode in one attempt?

No, The probability p gets printed out using scientific notation. So a probability of 9.447-12 is the same as 0.000000000009447. That is, a really small probability.