Stat344 Group Project

Group Leader: Wanxin Luo #33432808

Group member: Jun Jie Ou Yang #13971858

Nancy Liu #44411551

Yuan Cao #85579795

Harbor Zhang #54007349

Role & Contributions:

Coding: Yuan Cao/Harbor Zhang

Writing: Wanxin Luo/Nancy Liu/Harbor Zhang/Jun Jie Ou Yang

(a cover page image, https://upload.wikimedia.org/wikipedia/en/4/46/Pokemon_Go.png)

Introduction

Background: Pokemon Go is an augmented reality (AR) mobile game published in the year of 2016. By using the GPS in the mobile devices, users can locate, capture, train, and battle virtual creatures, called Pokemons, which appear as if they are in the real-life world. This game is an innovation that blends a fun smartphone game with actual, outdoor physical activities. As a result, it was one of the most profitable mobile apps in both the iOS and Android app stores and successfully attracted over 147 million monthly active users by May 2018.

Objectives: To determine and analyze the statistics of the pokemons inside the game "Pokemon Go". With this dataset we wish to determine and understand the whole statistics among the pokemons' attack damage, health, hit damage, etc of all the pokemons.

Why the problem is of interest: Considering that this game is super popular among modern teenagers and young adults, the purpose of our project is to provide them with an overall data analysis of the properties of the Pokemons, helping them to construct a better strategy when playing Pokemon Go.

In this report we will analyze the data by using four methods: Estimation of HP Mean with SRS, estimation of HP Mean with Stratified Sample, estimation of Proportion of Attack Power Above Average with SRS, and estimation of Proportion of Attack Power Above Average with Stratified Sample.

Methodology

Data Source and Acknowledgement

The dataset which contains information on all Pokemons was scraped from pokemondb.net and then sorted by Hamdalla F. Kareem on Kaggle.

Dataset Overview

There's a population of 1045 Pokemons in total, and each Pokemon represents a row in this dataset. We have both categorical variables (such as primary type, secondary type) and continuous variables (such as an attack, defense, HP, speed, etc). Besides, the dataset also includes the name of each Pokemon.

Data Analysis

(All the results are rounded to 3 decimal places)

Estimation of HP Mean with SRS

The analysis below is based on code in Appendix section 0.0.1

(Appendix[1],[2],[3]) We would like firstly to estimate the population mean of HP, also known as Hit Points or Health Points in the game, by applying the vanilla estimation method. We decide our simple random sample size to be 200. Then we use R code to generate our sample from the population and name our sample as pokemons.hp.s. The population mean of HP is 70.068.

$$\bar{y}_s = \frac{\sum_{i=1}^{n} y_i}{n} = \frac{\sum_{i=1}^{n} y_i}{200} = 69.525$$

(Appendix[4],[5]) By using the mean() function in R, we can calculate the sample mean of this SRS. The sample mean hp_bar_sample is 69.525.

$$SE(\bar{y}_s) = \sqrt{(1 - \frac{n}{N})\frac{S_s^2}{n}} = \sqrt{(1 - \frac{200}{1045})\frac{S_s^2}{200}} = 1.762$$

(Appendix [6])Since the population size is known in our case (population size = 1045), the Finite Population Correction Factor (FPC) is needed. The value of standard error hp_se.sample is 1.762.

(Appendix [7])Thus, the 95% confidence interval can be calculated by assuming the z-score is approximately equal to 1.96, samples are randomly selected from the population and the individuals in the sample are independent of each other.

$$[\bar{y}_s - 1.96 * SE(\bar{y}_s), \bar{y}_s + 1.96 * SE(\bar{y}_s)]$$

$$=[69.525-1.96*1.762397, 69.525+1.96*1.762397]$$

$$= [66.071, 72.980]$$

Knowing that the population mean of HP in this data frame is 70.068(Appendix[2]), we are 95% confident that the true population falls within the confidence interval[66.071, 72.980]. The 95% confidence interval successfully covers the population mean.

Estimation of HP Mean with Stratified Sample

The analysis below is based on code in Appendix section 0.0.2

(Appendix [8])We select the strata based on all categories of primary types because various categories have different properties and it is important to analyze which of them is more appropriate to use in different circumstances. We choose the sample size for each stratum with proportional allocation and set the sample size to be 200.

(Appendix [9])Then we estimate the population mean of HP, also known as Hit Points or Health Points in the game, by applying Stratified Sample Mean with proportional allocation.

$$\bar{y}_{str} = \sum_{h=1}^{H} (\frac{N_h}{N}\bar{y}_{sh})$$

By using the Appendix[9], [10] code in R, we can first calculate the sample mean of each stratum of STR. Then using the sum() function to get the estimate of the sample mean of this STR, which is 70.413.

$$SE(\bar{y}_{str}) = \sqrt{\sum_{h=1}^{H} (\frac{N_h^{\ 2}}{N})(1 - \frac{n_h}{N_h})\frac{S_{sh}^{\ 2}}{n_h}}$$

(Appendix [11]) Since the population size is known in our case (population size = 1045) and it is pretty large, the Finite Population Correction Factor (FPC) is needed. The value of the standard error of this STR is 1.462.

$$[\bar{y}_{str} - 1.96 * SE(\bar{y}_{str}), \bar{y}_{str} + 1.96 * SE(\bar{y}_{str})]$$

=[70.413-1.96*1.461583, 70.413+1.96*1.461583]

= [67.548, 73.278]

(Appendix [12]) Thus, the 95% confidence interval can be calculated by assuming the z-score is approximately equal to 1.96, samples are randomly selected from the population and the individuals in the sample are independent of each other.

According to the result above, we are 95% confident that the true population mean falls within the confidence interval [67.548, 73.278]. Since the population mean of HP in this data frame is 70.064, the 95% confidence interval successfully covers the population mean.

$$1 - \frac{SE(\bar{y}_{str})}{SE(\bar{y}_s)} = 1 - \frac{1.461583}{1.762397} = 0.171$$

According to the code in Appendix[13], if we use proportional allocation for stratified sampling instead of simple random sampling, the width of the 95% confidence interval is reduced by 17.1%.

Estimation of Proportion of Attack Power Above Average with SRS

The analysis below is based on code in Appendix section 0.0.4

(Appendix [14]) Now we want to estimate the proportion of pokemons whose attack power is above average attack value by utilizing the SRS. Then we use R to generate our sample from the population and name our sample as pokemons.s. The sample size here is n = 200, and the population size is N = 1045.

(Appendix [15], [16]) Based on the population information, we first calculate the true average attack value and the population proportion. The R code shows that the average attack power is 80.47 and the true proportion of pokemons whose attack power is above average is 0.449.

According to Appendix [17] and [18], In the vanilla estimation, we use the sample proportion generated from SRS to estimate the population proportion and its SE. Since the population size is known in our case (population size = 1045), the Finite Population Correction Factor (FPC) is needed. The sample proportion is 0.4 here, which is denoted as SRS.p. The SE of the proportion is 0.031, which is denoted as se.SRS.p.

$$\hat{p} = \frac{\sum_{i=1}^{n} p_i}{n} = \frac{\sum_{i=1}^{n} p_i}{200} = 0.4$$

$$SE(\hat{p}) = \sqrt{(1 - \frac{n}{N})\frac{p(1-p)}{n}} = \sqrt{(1 - \frac{200}{1045})\frac{p(1-p)}{200}} = 0.031$$

(Appendix [19]) Thus, the 95% confidence interval can be calculated by assuming the z-score is approximately equal to 1.96 because samples are randomly selected from the population and the individuals in the sample are independent of each other.

$$[\hat{p} - 1.96 * SE(\hat{p}), \hat{p} + 1.96 * SE(\hat{p})]$$

$$=[0.4\text{-}1.96*0.031115, 70.413+1.96*0.3115]$$

$$= [0.339,\ 0.461]$$

Therefore, we are 95% confident that the true population proportion falls within the confidence interval [0.339, 0.461]. Knowing that the population proportion of pokemons whose attack power is above average in this dataset is 0.449, we can conclude that the 95% confidence interval successfully covers the population proportion.

Estimation of Proportion of Attack Power Above Average with Stratified Sample

The analysis below is based on code in Appendix section 0.0.5

(Appendix [20]) Now we want to use the stratified sampling method to get the estimate of the targeted proportion. We select the strata based on the categories of primary types, on the grounds that various categories may correspond to different properties. Therefore, it is significant and useful to analyze which pokemons are more appropriate to use under different circumstances in the game. We choose the sample size for each stratum by proportional allocation and set the total sample size to be 200.

(Appendix[21], [22], [23]) The sample proportion of each stratum is denoted as p.sh. Then we use the sum() function to get the estimate of the proportion with the stratified sampling method and the SE for the proportion of the pokemons whose attack power is above average attack power. The estimate here is p.hat.str = 0.429 and its SE is 0.029.

$$\hat{p}_{str} = \sum_{h=1}^{H} (\frac{N_h}{N})\hat{p}_{sh} = 0.429$$

$$SE(\hat{p}_{str}) = \sqrt{\sum_{h=1}^{H} (\frac{N_h^{\ 2}}{N})(1 - \frac{n_h}{N_h})\frac{p_{sh}(1-p_{sh})}{n_{sh}}} = 0.029$$

(Appendix[24]) Then we calculate the 95% confidence interval of this estimate. The 95% confidence interval can be calculated by assuming the z-score is approximately equal to 1.96, because samples are randomly selected from the population and the individuals in the sample are independent of each other.

$$[\hat{p}_{str} - 1.96 * SE(\hat{p}_{str}), \hat{p}_{str} + 1.96 * SE(\hat{p}_{str})]$$

=[0.42916-1.96*0.02904, 0.42916+1.96*0.02904]

= [0.372, 0.486]

Therefore, we are 95% confident that the true population proportion falls within the confidence interval [0.372, 0.486]. Knowing that the population proportion of pokemons whose attack power is above average power in this dataset is 0.449, we can conclude that the 95% confidence interval successfully covers the population proportion.

(Appendix[25]) To sum up, we want to calculate how much reduction can be obtained by using stratification compared to SRS. There is a 6.76% reduction in 95% confidence interval width if we carry out a stratified sample using proportional allocation rather than SRS.

$$1 - \frac{SE(\hat{p}_{str})}{SE(\hat{p})} = 1 - \frac{0.02904}{0.03115} = 6.76\%$$

Conclusion and Discussion

Summary of the data analysis:

Method 1-SRS mean:
mean=69.525, se=1.762, 95%CI=[66.071,72.980]

Method 2-Stratified Sample Mean with proportional allocation:
mean=70.413, se=1462, 95%CI=[67.548,73.278]

Method 3-SRS proportion of attack over average:
proportion=0.449, se=0.031, 95%CI=[0.339,0.461]

Method4-Stratified Sample proportion with proportional allocation:
proportion=0.429, se=0.029, 95%CI=[0.372,0.486]

We choose the sample size to be 200 because it is large enough and the small sample size is not suitable for carrying out Stratified Sampling methods. Moreover, choosing 200 as the sample size is more convenient for comparing the difference between SRS and Stratified Sampling Method. We have originally chosen 100 as our Stratified Sample size, but apparently, it is not enough for selecting different groups. Since some groups only select one sample, the standard error can not be calculated. Our results can be generalized to larger populations since the data is our population, we can calculate that the true mean, true proportion and estimates are pretty similar.

Based on our conclusion from the parts before, it seems that according to the results there is a 17.7% reduction in the confidence interval width compared to SRS, therefore it means that the reduction is quite significant, which demonstrates that in this dataset it is a pretty good choice to use stratification. For the proportion part: The reduction is small, which means the between stratum variance is not big. Therefore, the stratification here may be not that helpful.

However, our project still has some limitations and needs improvement. The naming of pokemons has special rules, which are closely related to the cities, municipalities, and pokemons' primary types. Due to the fact that we are using R studio as the programming language, it is limited to comparing and analyzing numeric values and pokemons' names in the data set because we cannot compare and get the pattern towards pokemon's name etc. The only categorical variables we have are the primary and secondary types of pokemons. If we can categorize the data based on more geographical information and change the stratification method based on it, it is possible that the reduction of SE will become more significant.

Additionally, every row of the dataset represents one individual pokemon, and we cannot do the time series analysis on them to discover any trends. However, if we have the coding knowledge of data visualization, we can visualize the data by creating side-by-side boxplot of the HP value and attack power for pokemons with different primary types.

Citation

Kareem, Hamdalla F. “The World of Pokemons.” *Kaggle*, 29 Sept. 2021, https://www.kaggle.com/hamdallak/the-world-of-pokemons.

Appendix

[1]:
```
pokemons <- read.csv("pokemons dataset.csv") # population
head(pokemons,5)
```

A data.frame: 5 × 11

	Name <chr>	Name2 <chr>	Primary.Type <chr>	Secondary.type <chr>	Attack <int>	Defense <int>
1	Bulbasaur		GRASS	POISON	49	49
2	Ivysaur		GRASS	POISON	62	63
3	Venusaur		GRASS	POISON	82	83
4	Venusaur	Mega Venusaur	GRASS	POISON	100	123
5	Charmander		FIRE		52	43

[2]:
```
mean(pokemons$HP) # population mean of HP
```

70.0679425837321

[3]:
```
N <- length(pokemons$HP) #population size
n <- 200 #sample size
N
```

1045

0.0.1 SRS Mean

[4]:
```
set.seed(2)
SRS.index <- sample.int(N, n, replace = F)
pokemons.hp.s <- pokemons[SRS.index, ] #SRS sample
```

[5]:
```
#vanilla for hp_bar from the SRS sample
hp.bar.s <- mean(pokemons.hp.s$HP) #sample mean of HP
hp.bar.s
```

69.525

[6]:
```
#se of sample HP
hp.se.s <- sqrt((1 - n/N) * (var(pokemons.hp.s$HP)/n) )
hp.se.s
```

1.7623973563604

[7]:
```
#95% CI of sample HP mean
hp.CI <- c(hp.bar.s-1.96*hp.se.s,hp.bar.s+1.96*hp.se.s)
hp.CI
```

1. 66.0707011815336 2. 72.9792988184664

0.0.2 Stratification Mean with proportional allocation

[8]:
```
attach(pokemons)
N.h <- tapply(HP,Primary.Type, length)
type <- names(N.h)
n.h.prop <- (N.h/N)*n # h subpopulation sample propotion
n.h <- round(n.h.prop)
n.h #h subpopulation sample size
sum(n.h)
detach(pokemons)
```

BUG 16 **DARK** 9 **DRAGON** 8 **ELECTRIC** 12 **FAIRY** 4 **FIGHTING** 8 **FIRE** 12 **FLYING** 2 **GHOST** 8 **GRASS** 17 **GROUND** 8 **ICE** 7 **NORMAL** 22 **POISON** 8 **PSYCHIC** 15 **ROCK** 11 **STEEL** 7 **WATER** 26

200

[9]:
```
STR.pokemons.s <- NULL #Stratification sample
set.seed(2)
for(i in 1:length(type)){
  row.indes <- which(pokemons$Primary.Type==type[i])
  sample.index <- sample(row.indes,n.h[i],replace=F)
  STR.pokemons.s<-rbind(STR.pokemons.s,pokemons[sample.index,])
}
```

[10]:
```
#Stratification mean
hp.h.mean <- tapply(STR.pokemons.s$HP,STR.pokemons.s$Primary.Type,mean)
hp.bar.str <- sum(N.h/N*hp.h.mean)
hp.bar.str
```

70.4130782346621

[11]:
```
#Stratification se
var.hp.h <- tapply(STR.pokemons.s$HP,STR.pokemons.s$Primary.Type,var)
se.hp.h <- sqrt((1-n.h/N.h)*var.hp.h/n.h)
se.hp.bar.str<-sqrt(sum((N.h/N)^2*se.hp.h^2))
se.hp.bar.str
```

1.46158365680532

[12]:
```
#Stratification 95% CI
hp.str.CI <- c(hp.bar.str-1.96*se.hp.bar.str,hp.bar.str+1.96*se.hp.bar.str)
hp.str.CI
```

1. 67.5483742673237 2. 73.2777822020005

0.0.3 reduction in the 95% confidence interval

[13]:
```
reduction.hp <- 1-se.hp.bar.str/hp.se.s
reduction.hp <- round(reduction.hp,4)
paste(reduction.hp*100, "%", sep='')
```

'17.07%'

0.0.4 SRS proportion of attack over average

[14]:
```
set.seed(2)
SRS.index <- sample.int(N, n, replace = F)
pokemons.s <- pokemons[SRS.index, ] # SRS sample
```

[15]:
```
mean.attack <- mean(pokemons$Attack) #avergae attack
mean.attack
```

80.466985645933

[16]:
```
true.p <- nrow(pokemons[pokemons$Attack > mean.attack,])/N #true proportion
true.p
```

0.448803827751196

[17]:
```
#p : Probability that the attack power of Pokemon is greater than the average␣
 ↪attack power
#vanilla for p from the SRS sampl
SRS.p <- nrow(pokemons.s[pokemons.s$Attack > mean.attack,])/n
SRS.p
```

0.4

[18]:
```
#se of p
se.SRS.p <- sqrt((1-n/N)*SRS.p*(1-SRS.p)/n)
se.SRS.p
```

0.031150199489404

[19]:
```
#95% CI of p
SRS.p.CI <- c(SRS.p-1.96*se.SRS.p,SRS.p+1.96*se.SRS.p)
SRS.p.CI
```

1. 0.338945609000768 2. 0.461054390999232

0.0.5 Stratification proportion with proportional allocation

[20]:
```
attach(pokemons)
N.h <- tapply(HP,Primary.Type, length)
type <- names(N.h)
n.h.prop <- (N.h/N)*n #h subpopulation sample propotion
n.h <- round(n.h.prop)
n.h #h subpopulation sample size
sum(n.h)
detach(pokemons)
```

BUG 16 **DARK** 9 **DRAGON** 8 **ELECTRIC** 12 **FAIRY** 4 **FIGHTING** 8 **FIRE** 12 **FLYING** 2 **GHOST** 8 **GRASS** 17 **GROUND** 8 **ICE** 7 **NORMAL** 22 **POISON** 8 **PSYCHIC** 15 **ROCK** 11 **STEEL** 7 **WATER** 26

200

[21]:
```
STR.pokemons.s <- NULL #Stratification sample
p.sh <- NULL # h subpopulation sample proportion
set.seed(2)
for(i in 1:length(type)){
  sh <- NULL #sub population
  row.indes <- which(pokemons$Primary.Type==type[i])
  sample.index <- sample(row.indes,n.h[i],replace=F)
  sh <- rbind(sh,pokemons[sample.index,])
  p.sh <- c(p.sh,(nrow(sh[sh$Attack > mean.attack,])/n.h[i]))
  STR.pokemons.s<-rbind(STR.pokemons.s,pokemons[sample.index,])
}
p.sh
```

BUG 0.25 **DARK** 0.222222222222222 **DRAGON** 0.75 **ELECTRIC** 0.333333333333333 **FAIRY** 0.25 **FIGHTING** 0.875 **FIRE** 0.583333333333333 **FLYING** 1 **GHOST** 0.5 **GRASS** 0.294117647058824 **GROUND** 0.75 **ICE** 0.285714285714286 **NORMAL** 0.363636363636364 **POISON** 0.375 **PSYCHIC** 0.4 **ROCK** 0.636363636363636 **STEEL** 0.714285714285714 **WATER** 0.269230769230769

[22]:
```
#Stratification porpotion
p.hat.str <- sum(N.h/N*p.sh)
p.hat.str
```

0.42916682045165

[23]:
```
#Stratification se
se.p.hat.str <- sqrt(sum((N.h/N)^2*(p.sh*(1-p.sh)/n.h)*(1-n.h/N.h)))
se.p.hat.str
```

0.0290435351837081

[24]:
```
#95% CI
p.hat.str.CI <- c(p.hat.str-1.96*se.p.hat.str,p.hat.str+1.96*se.p.hat.str)
p.hat.str.CI
```

1. 0.372241491491582 2. 0.486092149411718

0.0.6 reduction in the 95% confidence interval

[25]:
```
reduction.p <- 1-se.p.hat.str/se.SRS.p
reduction.p <- round(reduction.p,4)
paste(reduction.p*100, "%", sep='')
```

'6.76%'