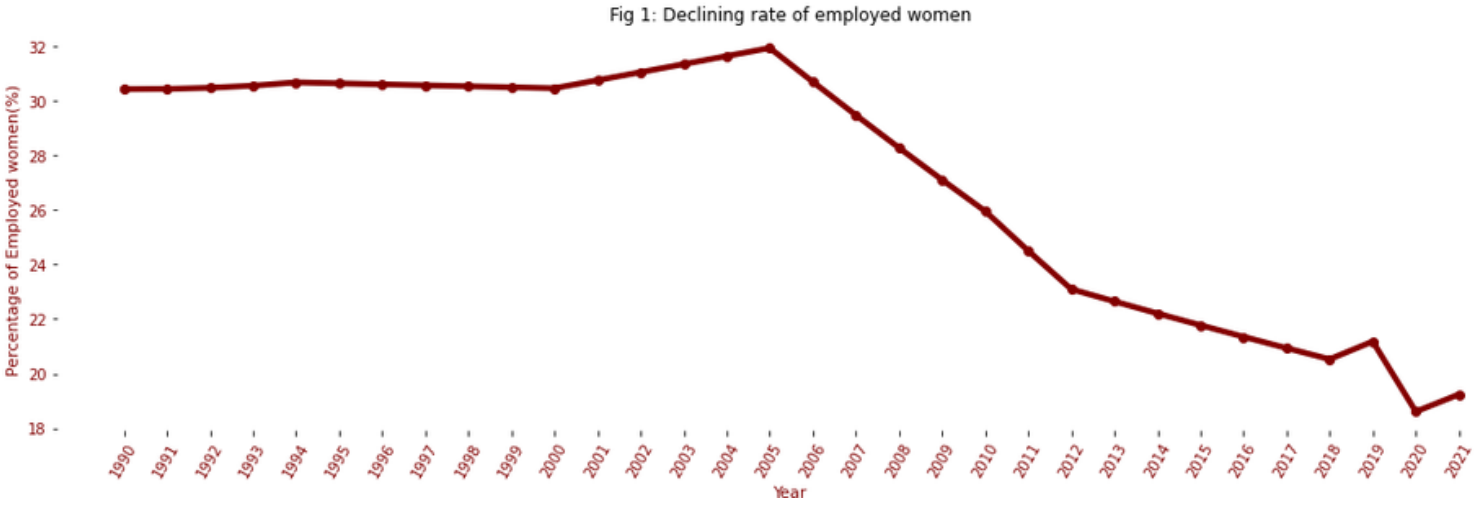
Education and Employment Rates of Indian Women

Introduction

The percentage of women over the age 15 employed in various sectors in India has been lower than 32 since 1990, which shows that a significant percentage of the population is not employed in regular jobs in any sector. Moreover the percentage of employed women has sharply **declined** in the last decade with just a minor uptick from 2020-21.



Analysis

Unemployment in women can be attributed to various reasons like opportunity, lack of access, lack of amenities to education or aspirations for better opportunities with higher education. Some of the predicted reasons are explored in the next sections.

1. Area type

One of the major factors for the low rate of participation by women is the availability of employment in their localities. This may depend on the type of location. In general, urban areas tend to have more opportunities than rural areas in all employment sectors. The below figure shows the comparison of unemployed women in rural and urban areas from recent years.

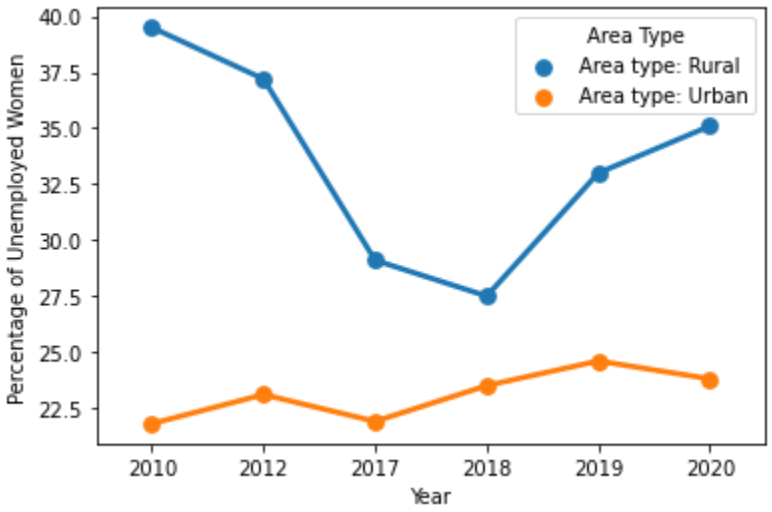


Fig 2: Unemployment rates of women in rural and urban areas

Spread of **awareness about education and employment** in rural areas may empower women in rural areas and help reduce the unemployment rates.

State wise analysis of unemployment in urban areas indicates that the number of unemployed women in urban areas per thousand have not seen a drastic change over the recent years, hinting that **opportunities** like those present in urban areas may help increase the overall employment rate for women.

The unemployment numbers in some states are, however drastically high compared to others. **New opportunities** in such areas would boost employment rates.
Below figure shows the State wise trend of unemployment in urban areas.

States like Karnataka and Tamil Nadu fare better than Lakshadweep and Chandigarh. Other states data have been dropped for better readability since the trend is similar.

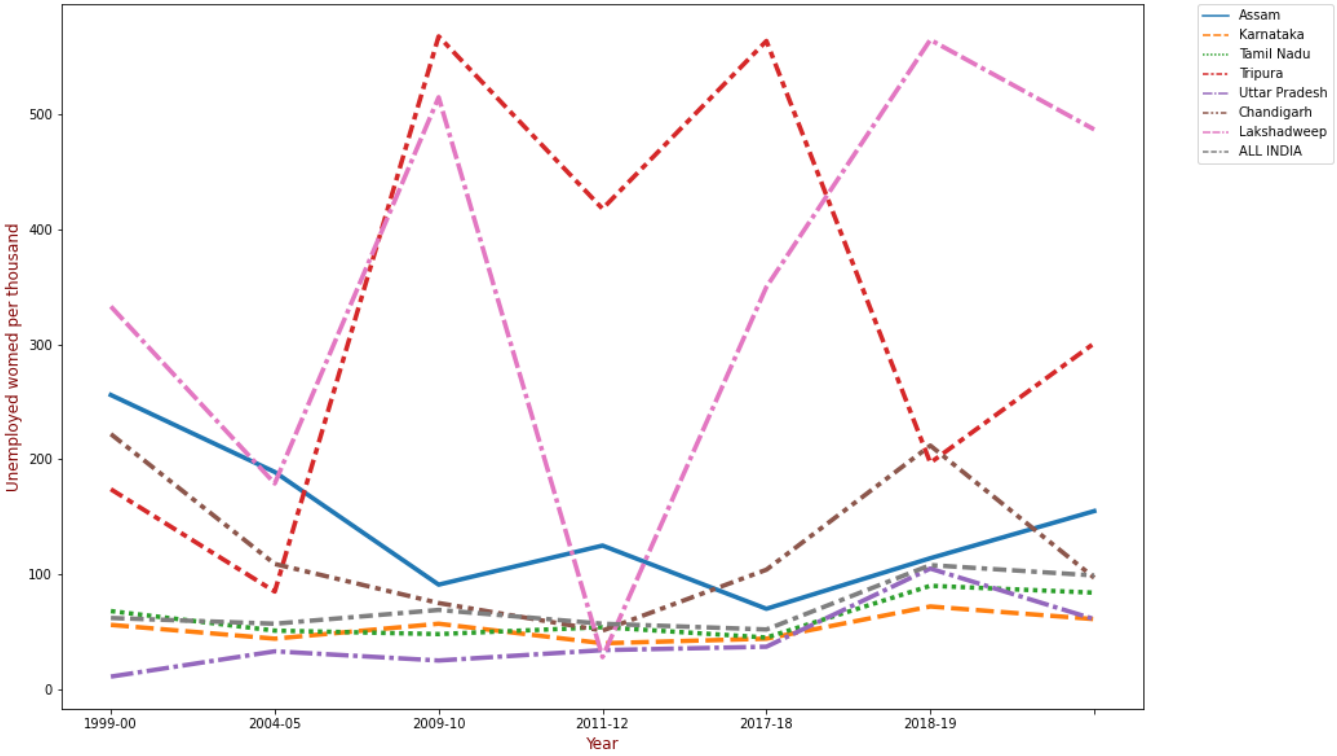


Fig 3: State wise trend of Unemployed women per thousand in all states

2. Higher Education

Another factor to consider is **education**. Below figure shows the Gender Parity Index (GPI) in higher education across various disciplines. The GPI is calculated as the quotient of the number of females by the number of males enrolled in all categories of education.

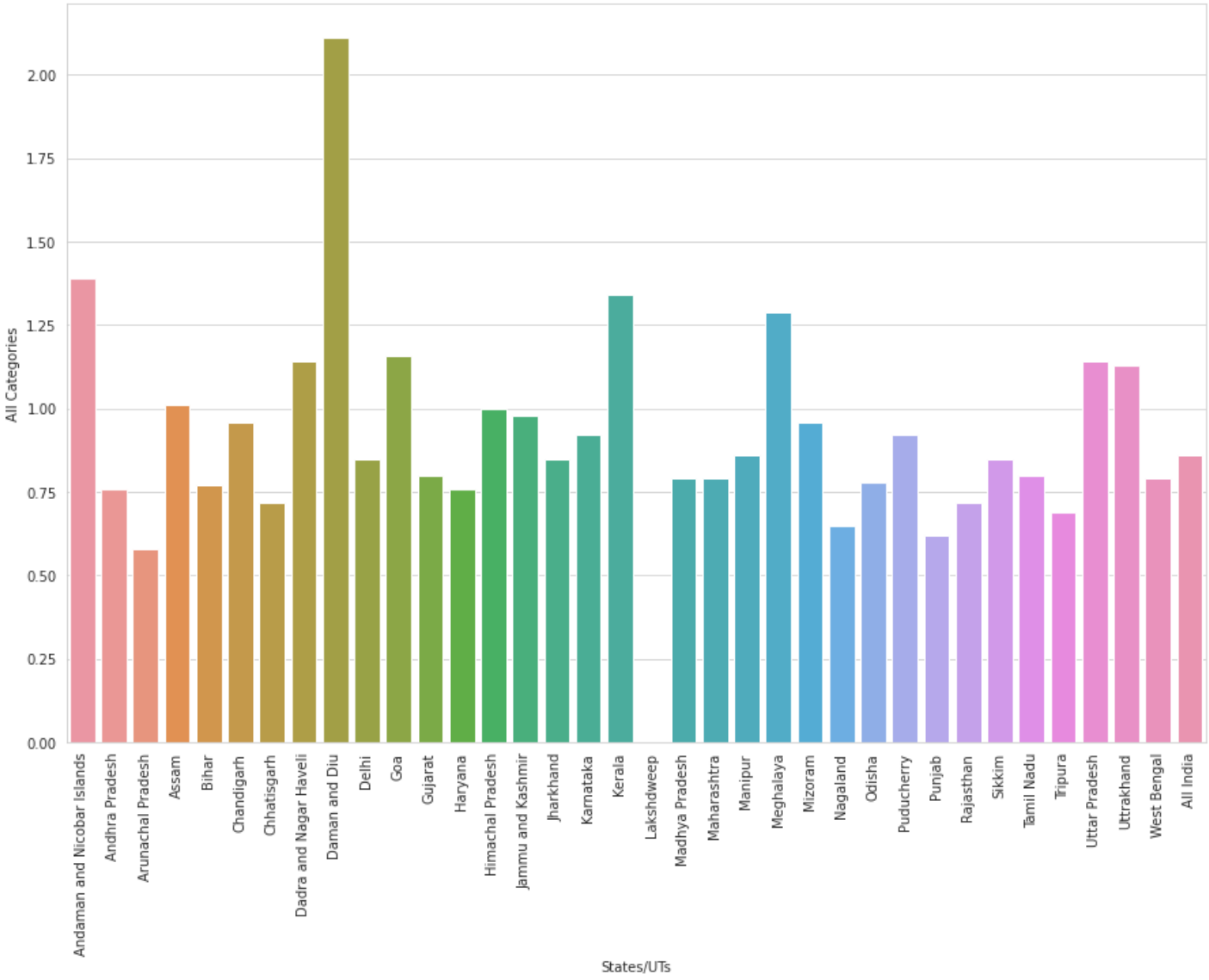


Fig 4: GPI rate in higher education across all states

The number of unemployed women per thousand in the year 2018-19 is comparatively lesser in states with higher GPIs(refer Fig 4 and Fig 5 for different states). The below figure shows the number of unemployed women per thousand in each state for the year 2018-19.

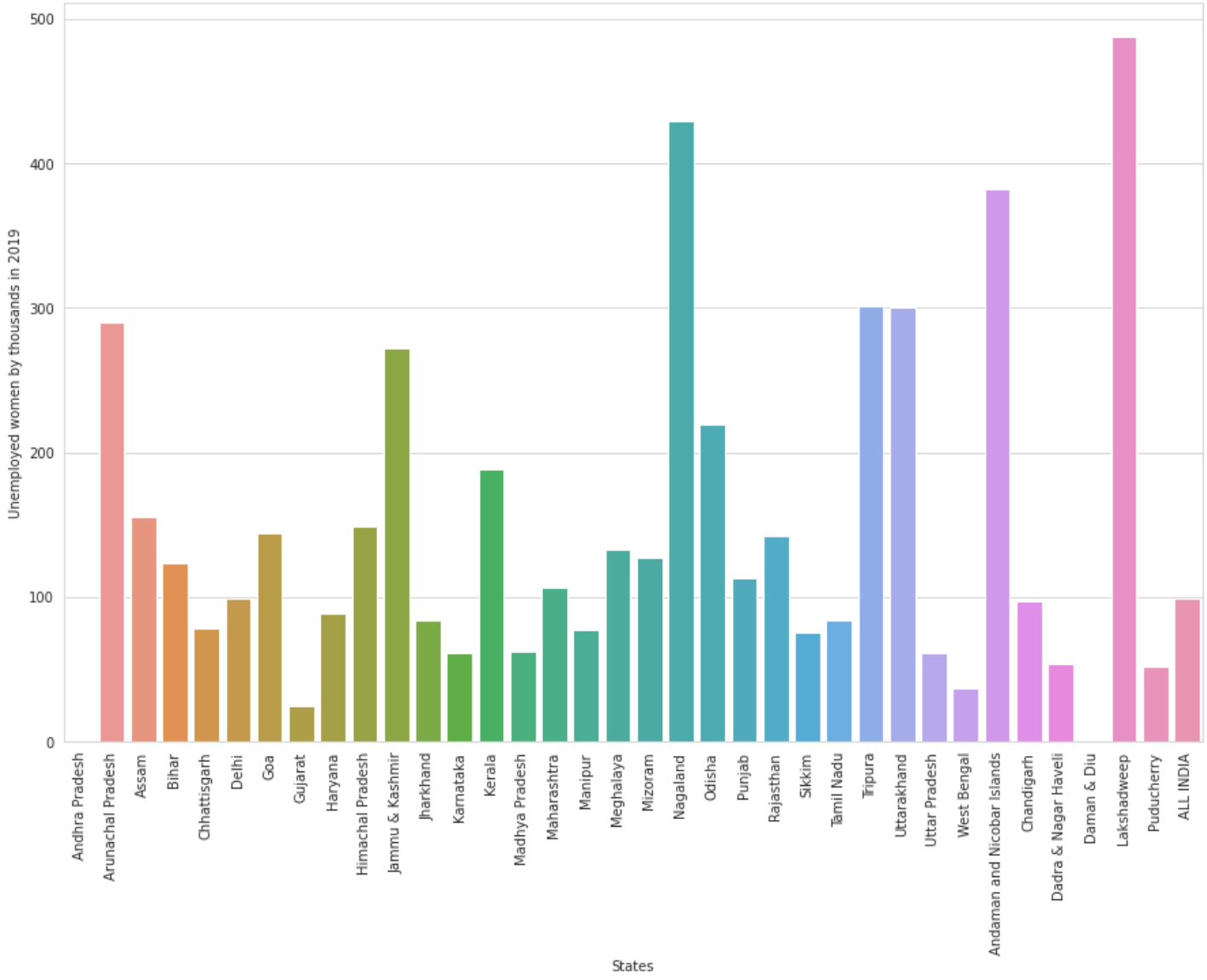


Fig 5: Number of unemployed women per thousand across all states

This indicates that girls should be **encouraged to pursue higher education** in order to increase the overall student GPI, which will lead to better employment opportunities.

3. Marriage

The next factor that affects employment is marriage and the share of household responsibilities. The below figures 5 and 6 show the percentage of married women and men across **various employment sectors** in india. It can be seen that there are significantly less married women in the workforce as opposed to married men. **Flexible schedules and other amenities** may encourage more married women to participate in the workforce.

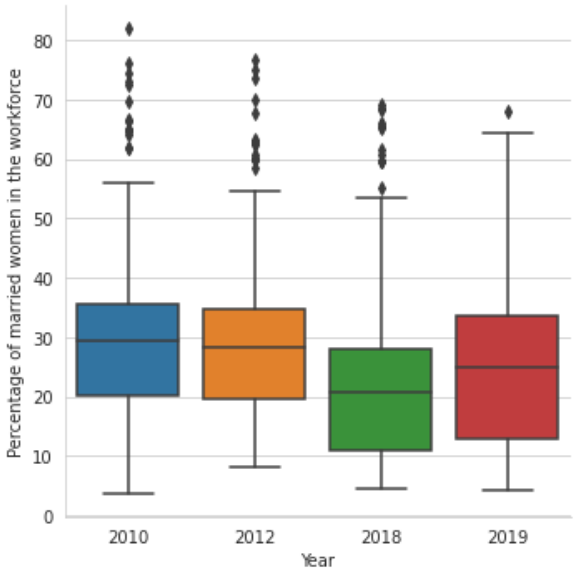


Fig 6: Percentage of Married Women in the workforce

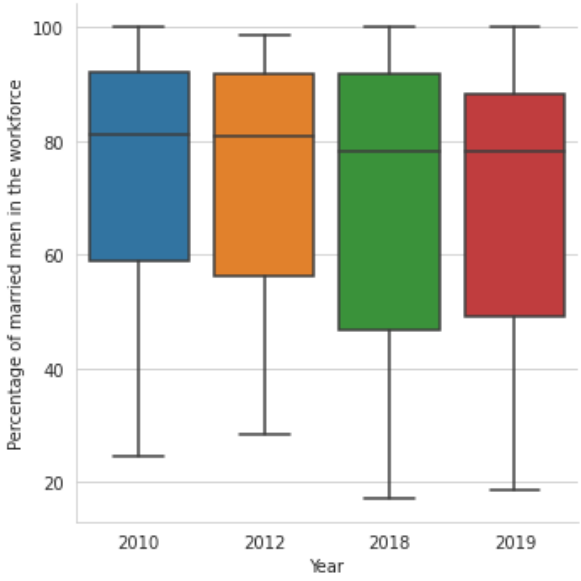


Fig 7: Percentage of Married Men in the workforce

4. Wage Gap

In most sectors, the wage gap between female and male employees is quite high. The below figure shows the percentage wage gap calculated annually in different sectors of employment. It can be observed that many sectors have a very significant percentage wage gap and it has increased from 2018 to 2020.
The wage gap should be significantly decreased to encourage women to aspire to participate in the various employment sectors.

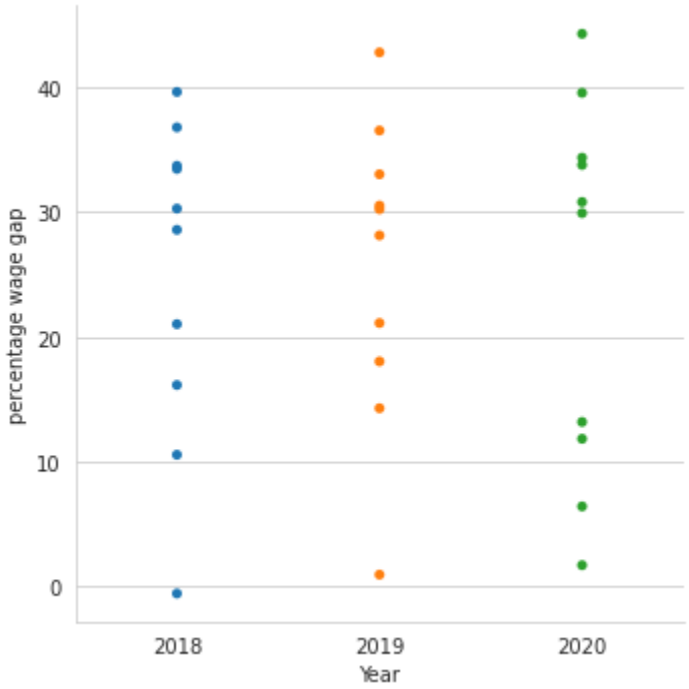


Fig 8: Percentage Wage gap in all employment sectors calculated for years 2018,19 and 20

Conclusion

Awareness about available opportunities, the need for employment as well as empowerment of women in all areas may lead to increased employment rates in the female population.

References:

1. Labor Force participation rate(% of female population ages 15+) *https://api.worldbank.org/v2/en/indicator/SL.TLF.CACT.FE.ZS?downloadformat=csv*

2. GPI in Higher education *https://data.gov.in/resource/gender-parity-index-gpi-higher-education-18-23-years-2010-11*

3. Pivot extracts from ILOSTAT for India *https://www.ilo.org/shinyapps/bulkexplorer24/?lang=en&segment=indicator&id=EAP_DWAP_SEX_AGE_MTS_RT_A* *https://www.ilo.org/shinyapps/bulkexplorer1/?lang=en&segment=indicator&id=EAR_GGAP_OCU_RT_A*

Float/Integer overflow/underflow errors

Overflow and underflow errors occur due to shortage of memory space allocated to store integer or floating point numbers. An Integer or floating point overflow occurs when the number that has to be stored is greater than the max number that can be stored in the memory space reserved for it. Underflow errors occur when the floating point result is smaller than the least floating point number that can be stored in the memory space reserved for it. They can lead to catastrophe if left unchecked.

The below are two interesting albeit less destructive examples of overflow error which are examined in the following pages:

1. PacMan Level 256 bug
2. Youtube views counter(Gangnam glitch)

Case Study 1: Pacman level 256 glitch

Introduction:

In the original arcade version of Pac-Man, players come across a garbled screen in level 256 instead of the regular maze, and find it impossible to clear this stage. This is an unintended result of integer overflow.

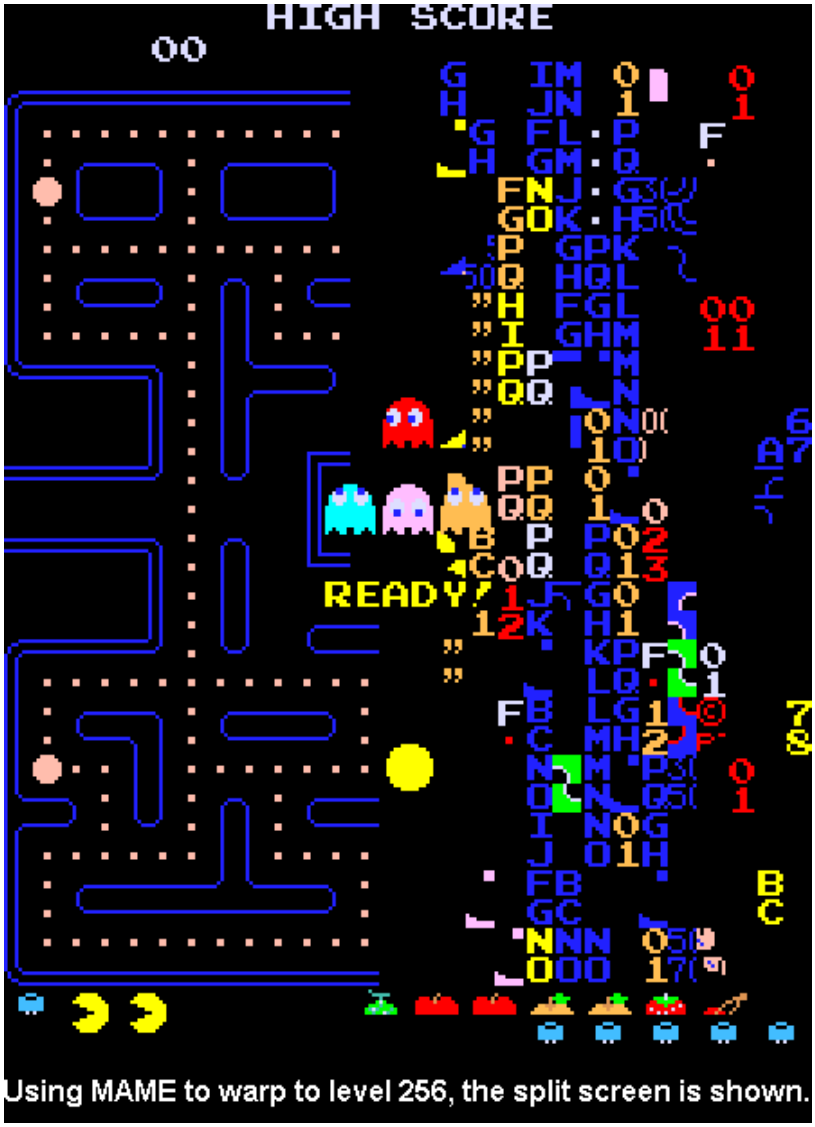


Fig 1: split screen on level 256

Root Cause:

The layout of the screen in each level of the game remains almost the same except for the level counter at the bottom left.

Fig 2: Example level counter

Each level is represented by an icon and the level counter in each level shows the current level icon and the past 6 level's icons **(7 icons in total)**.

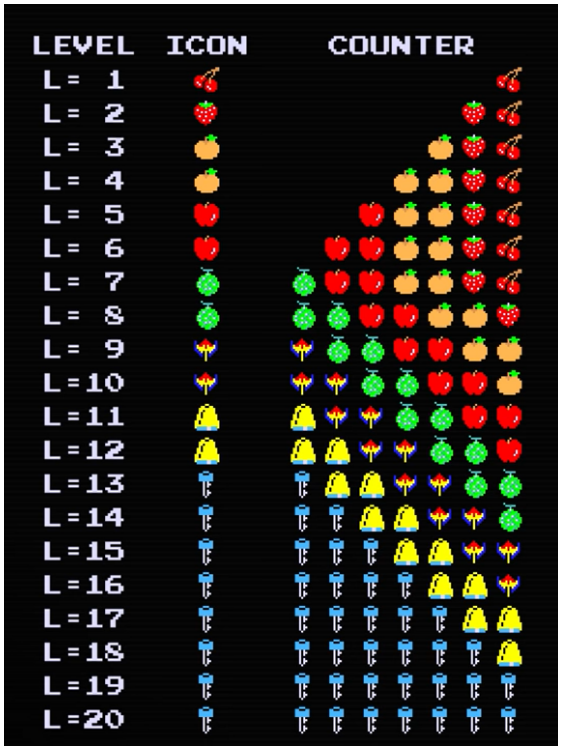


Fig 3: Level counter for all levels

The level counter icons along with coloring are maintained in a table 20 elements long.

Memory Location	Fruit Code	Color Code	Fruit
3B08	90	14	cherry
3B0A	94	0F	strawberry
3B0C	98	15	1st peach
3B0E	98	15	2nd peach
3B10	A0	14	1st apple
3B12	A0	14	2nd apple
3B14	A4	17	1st grape
3B16	A4	17	2nd grape
3B18	A8	09	1st galaxian
3B1A	A8	09	2nd galaxian
3B1C	9C	16	1st bell
3B1E	9C	16	2nd bell
3B20	AC	16	1st key
3B22	AC	16	2nd key
3B24	AC	16	3rd key
3B26	AC	16	4th key
3B28	AC	16	5th key
3B2A	AC	16	6th key
3B2C	AC	16	7th key
3B2E	AC	16	8th key

Fig 4: Table storing level icons to be drawn and color to be filled

The range of fruits to be drawn is dependent on the level number and there are three cases in the algorithm based on level number. In each case, two pointers are initialized.

The process starts by drawing and coloring the icon stored in pointer 1 and keeps incrementing this pointer until pointer 1 == pointer 2.
Case 1: If level <8: pointer 1= 1, pointer 2=current level,ie: **Range=[1,L]**
Here icons from level 1(cherry) to the current level icon are drawn and coloured accordingly and the remaining empty spaces are coloured black.
Case 2: if level <19:pointer 1= current level-6 , pointer 2=current level,ie, **Range=[L-6,L]**
The icons from 6 levels prior are drawn and coloured along with the current level icon.
Case 3: If level>=19:pointer 1= 13, pointer 2=19,ie,: **Range=[13,19]**
Here the icons from level 13(1st Key) to level 19(7th key) are drawn and colored.

Level numbers are stored in a **8-bit(1 byte)** memory space. This limits the maximum number that can be held to 255(11111111 in binary is 255 in integer format). If we try to add 1 and store the result in the same allocated space, all the bits from **0 to 8 get flipped to 0 and there is no space to store the extra '1' bit in the 9th position**. So the **level is now wrongly stored as 0 (00000000) for level 256**. The level drawing algorithm checks the level number and since it falls below 8, it starts the process for Case 1 and sets the starting pointer as level 1 and ending pointer as current level(0).
So the process starts by drawing fruit 1 and increments the pointer 1 by 1. Since pointer 2=current level=0, the pointer 1 keeps getting incremented by 1 **indefinitely**. However since the counter is also stored in a 8 bit memory location, this process stops after looping **255** times. As pointer1 overflows back to 0 at 256 and pointer 1 is now equal to pointer 2. Since the table for drawing icons are only 20 elements long, the icons that are drawn after the 20th level icon are no longer referencing the right memory address, instead accessing irrelevant information meant for other parts of the program. This causes other symbols to be drawn, giving the glitchy appearance to the screen.

Solution:

Many classic arcade games revert to the starting level after the final level is encountered. If an overflow check is performed on the level number, once the level 255 is cleared, instead of trying to store 255+1=256, it could be saved as level 1 in memory and control should flow normally from the start of the algorithm.This would mean that after level 255, the player then goes back to level 1 and continues from there on as usual.

Case 2: Gangnam glitch

Introduction

In early December 2014, the official YouTube accounted posted the below:

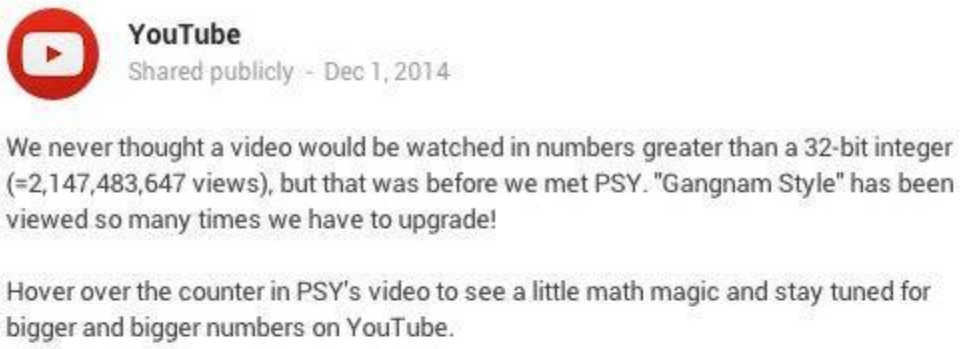


Fig 1: YouTubes's post

Here's a picture of YouTube's view counter showing the glitch:

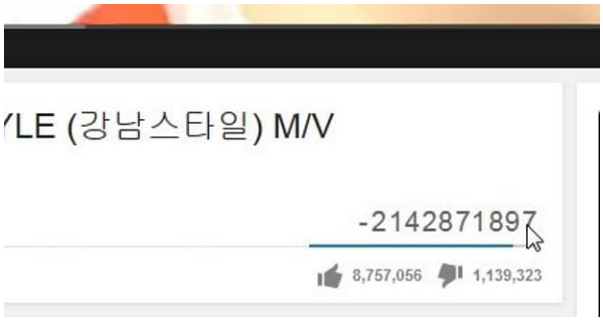


Fig 2: View counter glitch

Root Cause:

YouTube used a standard 32-bit signed integer to store the number of views for the videos. This may be because the computer used initially when they started coding and implementing the process was running a 32-bit operating system. Before the music video in question, it was thought two billion views per video was the upper limit. Any more views were thought to be unlikely.

In a signed integer representation, the leftmost bit is reserved to represent the sign of the number, where 1 represents a negative number.

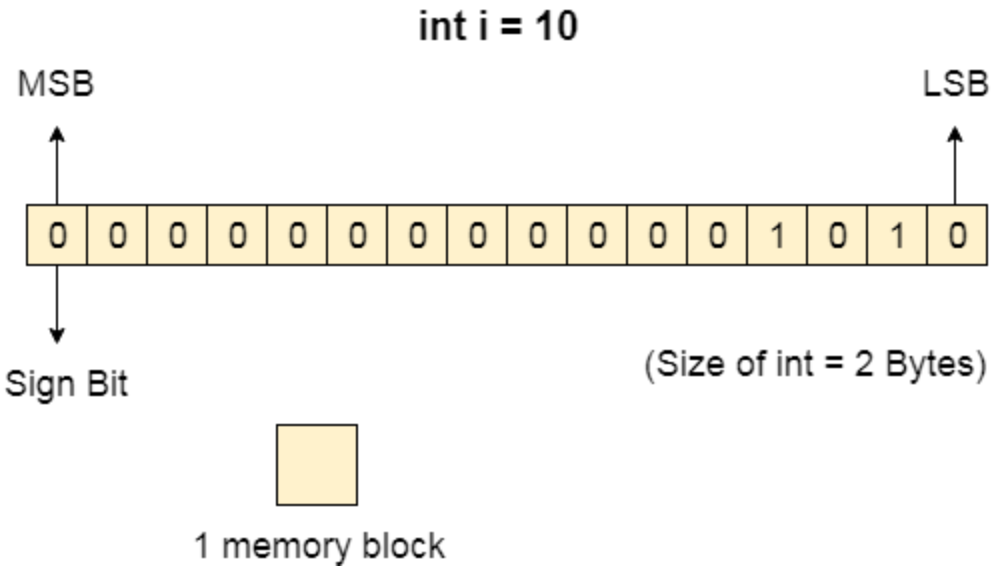


Fig 3: Representation of a signed integer

So the maximum positive number that can be represented in this system is 2^{31}.
This represents 2,147,483,647 in integer format (called INT_MAX in C).

In the music video when the process was trying to record the next view after INT_MAX, that is the 2,147,483,648th view, the bits from LSB to 31st bit gets set to 0 and the 32nd '1' bit gets stored in the Sign Bit, which causes the view count to flip over to negative numbers.

Solution:

The straightforward solution would be to use 32 bit unsigned integer storage instead of 32 bit signed integers to store the view count. However, Google's style guide explicitly warns against the usage of unsigned integers since they may lead to bugs in loops that are hard to spot.
Quote from style guide:
'You should not use the unsigned integer types such as uint32_t, unless there is a valid reason such as representing a bit pattern rather than a number, or you need defined overflow modulo 2^N. In particular, do not use unsigned types to say a number will never be negative. Instead, use assertions for this.'
So a signed 64 bit integer should be used to store the view count.