MSc Business Analytics

Programming for Analytics MIS41110

Bingo Project - Group 9

UCD DUBLIN

UCD Michael Smurfit Graduate Business School

"Declaration of own work"

We declare that this material, which we now submit for assessment, is entirely our work and has not been taken from the work of others, save and to the extent that such work has been cited and acknowledged within the text of our work.

Team Member	Student Number	Code	User Manual	UML
Kevin Sheahen	22202667	25%	25%	25%
Andre Kurz	21201380	25%	25%	25%
Anish Joseph	22203566	25%	25%	25%
Conor Patterson	10354185	25%	25%	25%

Submission Date: the 29th of November, 2022

Table of Contents

ABOUT THE GAME

As Business Analysts, we were presented with developing a software application for an upcoming Bingo party. Delivering on the requirements of creating the desired number of bingo cards determined by the user and analytical predictions of how those cards will perform. Additionally, this program aims to be as efficiently and cleanly written as possible, keeping the user experience in mind.

OVERVIEW

At a high level, the code that the user interacts with can be broken into two main branches. A "Standard Bingo Game", which is the typical 5x5 bingo sheet with a single free space in the middle and numbers 1-75. The contrary is a "Dynamic Bingo Game", which will ask the user for more specifics such as number of columns, rows, free spaces/zeros in the matrix, and number range. These two paths will produce varying results in terms of data frame size, graphs and analytical interpretation.

TECHNICAL DOCUMENTATION

USER INPUT

Users have multiple choices in how they would like to interact with the bingo software. The UML Diagram at the end of this document provides a visual flow of the actions. Users can choose between the "Standard Bingo" and "Dynamic Bingo" game; the standard game will only ask for two criteria of the user, while the Dynamic Bingo game will ask for several user inputs regarding matrices and simulations.

ERROR HANDLING

Users should only encounter programmed errors and not any pythonic errors. For example, when entering a non-numeric into the number of matrices desired, the response is, "You have entered a none integer for number of cards. Re-enter an integer in this field". A user should not expect to see "ValueError: invalid literal for int() with base 10:". However, if this should happen, please file a complaint, and we will work to fix the problem.

VARIABLES

Throughout the program, there are numerous variables, more than what should be stated here. For readability, users should follow the prefix system, which is using a prefix to define the type of variable used. Below is a table of the prefixes and their corresponding meanings.

Prefix	**Meaning**
s (ex. sCARDS)	**String**
b	**Boolean**

d (ex. dBINGO_SHEET)	**Dictionary**
df (ex. df_BINGO_RESULTS)	**Dataframe**
a	**Array**
l (ex. lSELECTED)	**List**
k (ex. for k , v in columnnames.items())	**Keys within a dictionary**
v (ex. for k , v in columnnames.items())	**Values associated with a key in a dictionary**

STANDARD - BOARD CREATION

The "Standard Bingo" game route will take one input from the user: how many boards they would like. From this, a single while loop is performed, starting at 1 and looping until it reaches the desired board count + 1. During the loop, the board names are established and added to a dictionary. Following this, random numbers are chosen for each array row of the matrix following the specifications outlined in the assignment, concatenation of all separate arrays into one large array, and a zero is added to the middle of the bingo sheet to represent a free space. Lastly, we print the generated bingo sheet and repeat the process until we have reached the desired amount of bingo sheets.

DYNAMIC - BOARD CREATION

The "Dynamic Bingo" game differs in many ways from how the "Standard Bingo" game is created. Mainly because the freedom of customisation by the user is much greater. Users will input the Number of Boards, Columns, Rows, starting number range, ending number range and the number of zeros desired. From this, almost all the same steps as the standard process are replicated, but just with more flexibility with dynamic variables to handle the logic.

STANDARD - SIMULATION CREATION

The process for creating the standard simulation sequences is a simple 1 input process. The user inputs how many simulations they would like to have run per bingo sheet. From this, a while-loop is run and creates a dictionary holding the unique simulation sequences. These sequences range from 1 to 75.

DYNAMIC - SIMULATION CREATION

The dynamic process for creating the simulation sequences is basically the same as the standard process. The only difference is that the range of numbers in the sequence can differ from the standard 1 to 75.

DATAFRAME RESULTS

The results populating in the data frame contain information about how well each bingo card did in each simulation and what combination won at which turn. Dataframe results will vary depending on the actions of the users but will always have the same principal foundations. Regardless of a standard or dynamic game route, the data frame will have the bingo board number, simulation number, the first diagonal (left to right, also referred to as the primary), the second diagonal (right to left, referred to as the secondary diagonal), and the earliest win among the possible combinations. For a standard game, you will additionally have a column for winning rows and columns 1-5, while the dynamic game will be flexible to accompany the varying amount from game to game.

GUI – USER MANUAL

1. Install python3(path configure and download PIP)

2. Install the following modules/libraries in cmd prompt by following the example "py -m pip install matplotlib"

1. fpdf	2. numpy	3. Pillow	4. seaborn	5. dataframe_image
6. matplotlib	7. pandas	8. scipy	9. statistics	10. html2image

3. Run the program on Terminal or PowerShell or python IDE by running python main.py

4. The popup screen appears that welcomes you to the bingo game. A screen will appear where you have a dropdown to choose the game type.

5. Once a game type is selected, click next; if it's Standard, it takes you to a new window where you are expected to enter details about the game i.e. the number of simulations and number of cards.

6. If the game selected is dynamic, it takes you to another window where you are required to enter all values about the dynamic game, like the number of simulations, cards, rows, columns, zeros, start range and end range.

7. The system will show error messages if there is an error in the values entered.

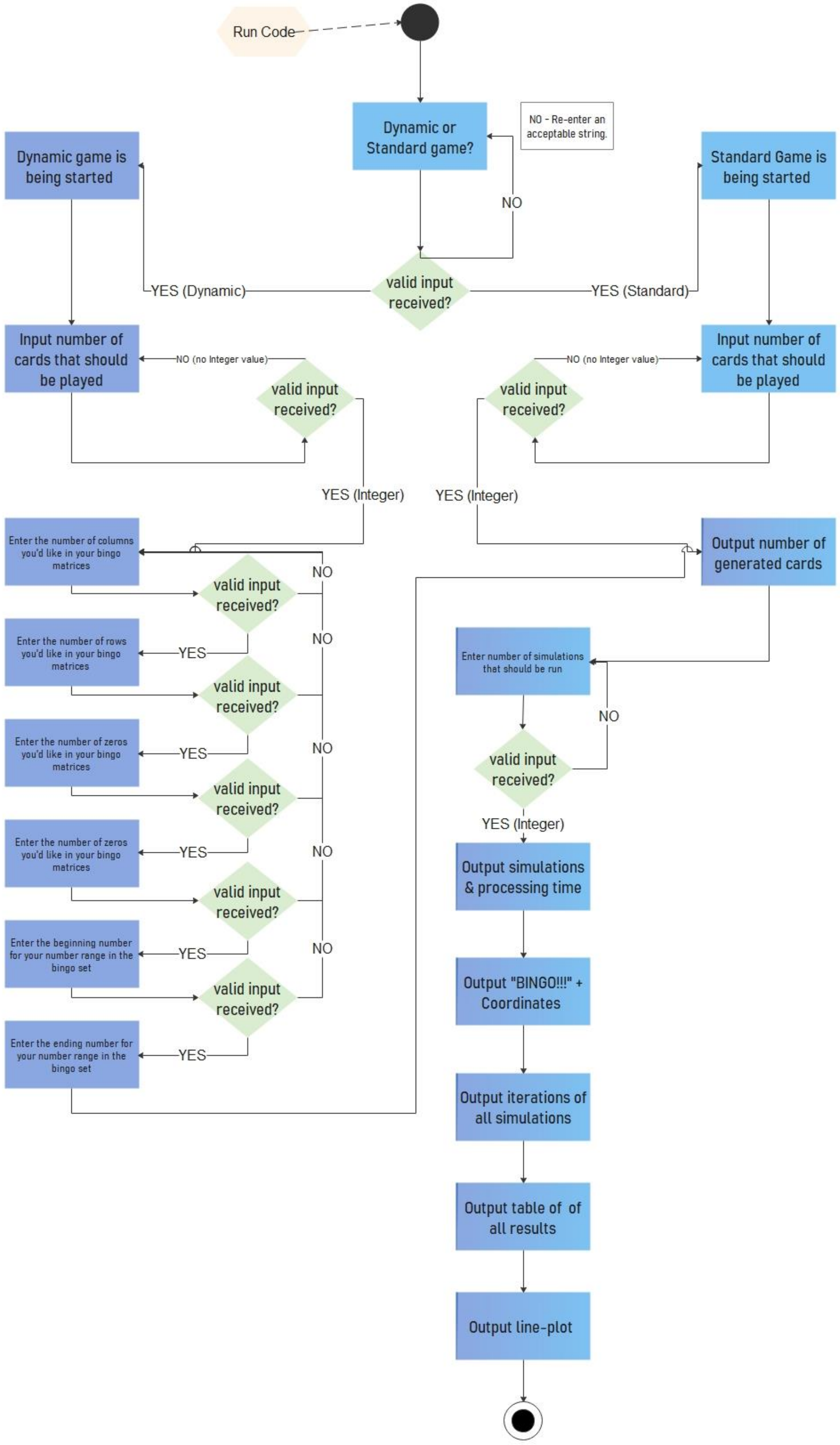
8. If there is no error on the values entered, the game will start running by calling card generation functions, simulation generation, and running simulations. This step may take some time, depending on the specifications of the computer running the programme.

9. The system will redirect to a new window that is divided into two. You can scroll using the mouse scroll button to view hidden window data.

10. The first window on the left shows the generated unique bingo cards that are automatically saved into the programme directory for later viewing as PDF’s.

11. On the right, i.e., other half of the screen, the system will plot graphs using matplotlib and display them depending on the generated results after running game.

BINGO SIMULATION UML ACTIVITY DIAGRAM


Run Code
Dynamic or Standard game?
NO - Re-enter an acceptable string.
NO
Dynamic game is being started
Standard Game is being started
YES (Dynamic)
valid input received?
YES (Standard)
Input number of cards that should be played
NO (no Integer value)
valid input received?
valid input received?
NO (no Integer value)
Input number of cards that should be played
YES (Integer)
YES (Integer)
Enter the number of columns you'd like in your bingo matrices
Output number of generated cards
NO
valid input received?
NO
Enter the number of rows you'd like in your bingo matrices
YES
Enter number of simulations that should be run
valid input received?
NO
Enter the number of zeros you'd like in your bingo matrices
YES
NO
valid input received?
valid input received?
YES (Integer)
Enter the number of zeros you'd like in your bingo matrices
YES
NO
Output simulations & processing time
valid input received?
Enter the beginning number for your number range in the bingo set
YES
NO
Output "BINGO!!!" + Coordinates
valid input received?
Enter the ending number for your number range in the bingo set
YES
Output iterations of all simulations
Output table of of all results
Output line-plot