



# USE OF VBA AND PYTHON IN ANALYTICAL TOOLS

**Mike Giannotti | Data Architect**  
**Ashlen Grote | Data Architect**

May 13, 2025



# Agenda

---

## Welcome

## Introduction to Excel VBA and Python

### Excel VBA Regression Techniques

- Performing Regression Using Excel VBA
- Excel Data Visualization

### Python Regression Techniques

- Performing Regression Using Python
- Python Data Visualization

### Combining VBA and Python

- Performing Regression Using Excel VBA and Python
- General Process and Use Cases

## Q&A





# Presenters

## Ashlen Grote

### College

B.S. Computer Engineering 2023

### Job Experience

Data Architect specializing in API tool development

Experience at GE Aerospace and Keysight

### Expertise

Designing and developing custom API-driven tools to support Cost Engineering and data architecture solutions



## Mike Giannotti

### College

B.S., M.S. Electrical Engineering 2008

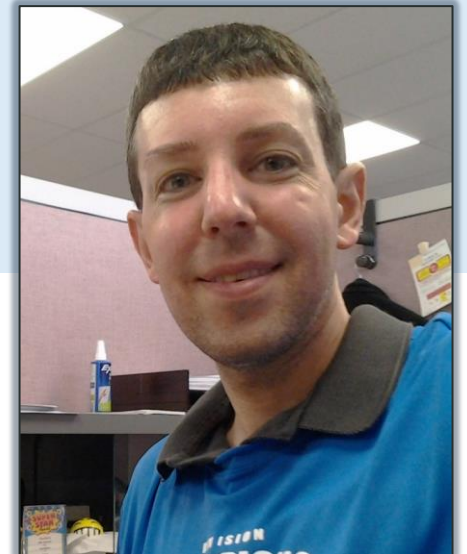
A.S. Computer Science, Mathematics, Physics

### Job Experience

10 years experience in Data Analysis, and Electrical design. Military radios, space telescopes, imaging satellites.

### Expertise

Designing parametric models for estimating labor costs.



# Excel VBA and Python Comparison



## VBA

- **Best for workflows mixing manual steps and automation**
- **Excel Automation:** Repetitive tasks within Excel
- **Interactive User Forms:** Buttons and controls for user input
- **Tight Excel Integration:** Charts and pivot tables that update dynamically
- **Lightweight APIs:** Small data exchanges with APIs
- **Quick Prototyping:** Rapid creation of small-scale tools



## PYTHON

- **Best for Data-Driven Analysis**
- **Heavy Data Processing:** Handles large datasets efficiently
- **Advanced Analytics:** Libraries like Pandas and NumPy for complex calculations
- **API Integration:** Retrieves and sends data at scale
- **Custom GUIs:** Create tailored interfaces with PyQt5
- **Scalable Solutions:** Long-term, robust applications



*VBA and Python can be integrated to leverage the strengths of both tools for more powerful and versatile solutions*



# Performing Regression Using Excel VBA





**Visual Basic for Applications (VBA) is a Microsoft programming language that lets you design software to automate Excel tasks.**

## **Advantages:**

- Save time by automating repetitive tasks.
- Eliminate human error.



# Perform Regression With VBA

## Excel Regression Manually

Regression

Input

Input Y Range: \$H\$4:\$H\$214

Input X Range: \$C\$4:\$G\$214

Labels  Constant is Zero

Confidence Level: 95 %

Output options

Output Range: \$J\$4

New Worksheet Ply:

New Workbook

Residuals

Residuals  Residual Plots

Standardized Residuals  Line Fit Plots

Normal Probability

Normal Probability Plots

OK Cancel Help

## Excel Regression VBA Software

```
Application.Run "ATPVBAEN.XLAM!Regress", _
    Y Input Range..... ActiveSheet.Range("$H$4:$H$214"), _
    X Input Range..... ActiveSheet.Range("$C$4:$G$214"), _
    Constant is Zero..... False, _
    Labels..... True, _
    Confidence Level..... 95, _
    Output Options..... ActiveSheet.Range("$J$4"), _
    Residuals..... True, _
    Standardized Residuals.... False, _
    Residual Plots..... False, _
    Line Fit Plots..... False, _
    Normal Probability Plots.... False
```



# Automate Regression By Using VBA

**Regression Tool**

**Start** **Delete Tabs**

X Input Range (Independent Variables)  
Get Highlighted Range

Y Input Range (Dependent Variable)  
Get Highlighted Range

Y Intercept = 0

VIF

Correlation

Put Dataset Below: Note All independent variables and dependent variables must be in consecutive columns.

**Start Regression**



# VBA Output: Regression Worksheet

**VBA allows you to add extra features**

REGRESSION SUMMARY OUTPUT								
<i>Regression Statistics</i>		<u>REGRESSION EQUATION</u>						
Multiple R	0.95377581	Runs = -559.997079813703 + (0.632785530889507)(Singles) + (0.705947336376347)(Doubles) + (1.26372148467694)(Triples) + (1.49074135216999)(Home Runs) + (0.346563388102523)(Walks + Hit by Pitcher)						
R Square	0.9096883							
Adjusted R Square	0.90747478							
Standard Error	24.4322331							
Observations	210							
ANOVA								
	df	SS	MS	F	Significance F			
Regression	5	1226605.957	245321.191	410.968694	2.099E-104			
Residual	204	121774.5386	596.934013					
Total	209	1348380.495						
	<i>Coefficients</i>	<i>Standard Error</i>	<i>t Stat</i>	<i>P-value</i>	<i>Lower 95%</i>	<i>Upper 95%</i>	<i>Lower 95.0%</i>	<i>Upper 95.0%</i>
Intercept	-559.99708	35.52184489	-15.7648647	3.8121E-37	-630.034112	-489.960048	-630.034112	-489.960048
Singles	0.63278553	0.030209403	20.9466412	9.7657E-53	0.57322283	0.69234823	0.57322283	0.69234823
Doubles	0.70594734	0.067573663	10.4470782	9.7354E-21	0.57271499	0.83917968	0.57271499	0.83917968
Triples	1.26372148	0.200532184	6.30183872	1.7822E-09	0.86834002	1.65910295	0.86834002	1.65910295
Home Runs	1.49074135	0.060847935	24.4994567	1.1048E-62	1.37076986	1.61071285	1.37076986	1.61071285
Walks + Hit by Pitcher	0.34656339	0.025508664	13.5861051	2.2959E-30	0.29626895	0.39685782	0.29626895	0.39685782
RESIDUAL OUTPUT								
	<i>Observation</i>	<i>Actual Runs</i>	<i>Predicted Runs</i>	<i>Residuals</i>				
	1	864	909.5427592	-45.5427592				
	2	794	799.0320991	-5.03209915				

Min and Max Runs  $y = x$

574 500

984.6675138 1000



# VBA Output: Regression Worksheet

## Important Results

1. Adjusted R Square
2. Coefficients
3. P-values
4. Residuals
5. Regression Equation

SUMMARY OUTPUT								
<i>Regression Statistics</i>								
Multiple R	0.953775811							
R Square	0.909688297							
Adjusted R Square	0.907474775							
Standard Error	24.43223307							
Observations	210							
<i>ANOVA</i>								
	<i>df</i>	<i>SS</i>	<i>MS</i>	<i>F</i>	<i>Significance F</i>			
Regression	5	1226605.957	245321.1913	410.9686936	2.0992E-104			
Residual	204	121774.5386	596.9340126					
Total	209	1348380.495						
	<i>Coefficients</i>	<i>Standard Error</i>	<i>t Stat</i>	<i>P-value</i>	<i>Lower 95%</i>	<i>Upper 95%</i>	<i>Lower 95.0%</i>	<i>Upper 95.0%</i>
Intercept	-559.9970798	35.52184489	-15.76486473	3.81209E-37	-630.0341121	-489.9600475	-630.0341121	-489.9600475
Singles	0.632785531	0.030209403	20.94664121	9.76565E-53	0.573222832	0.69234823	0.573222832	0.69234823
Doubles	0.705947336	0.067573663	10.44707819	9.73539E-21	0.572714989	0.839179684	0.572714989	0.839179684
Triples	1.263721485	0.200532184	6.301838725	1.78218E-09	0.86834002	1.65910295	0.86834002	1.65910295
Home Runs	1.490741352	0.060847935	24.49945673	1.10475E-62	1.370769858	1.610712846	1.370769858	1.610712846
Walks + Hit by Pitcher	0.346563388	0.025508664	13.58610506	2.29591E-30	0.296268953	0.396857823	0.296268953	0.396857823
<div style="border: 1px solid black; padding: 5px; display: inline-block;"> <p>Dependent Variable: <math>Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2</math></p> <p>Independent Variables: <math>X_1, X_2</math></p> <p>Y intercept: <math>\beta_0</math></p> <p>Coefficients: <math>\beta_1, \beta_2</math></p> </div>								
RESIDUAL OUTPUT								
	<i>Observation</i>	<i>Predicted Runs</i>	<i>Residuals</i>					
	1	909.5427592	-45.54275916					
	2	799.0320991	-5.032099146					
	3	803.9731688	-11.97316875					
	4	915.0553052	62.94469483					
	5	963.4255338	-13.42553378					

**Adjusted R Square - is a model accuracy measure.**  
**Range: 0 to 1**  
**(greater value indicates better predictive ability)**

**P-Value**  
**A low p-value (< 0.05)**  
**indicates the**  
**variable is likely to be a**  
**meaningful addition to**  
**the model.**

**Residuals**  
**Residual = Actual - Predicted**  
**(Residual value equal to 0 is desired)**

## Regression Equation

$$\text{Runs} = -559.997 + (0.63)(\text{Singles}) + (0.71)(\text{Doubles}) + (1.26)(\text{Triples}) + (1.49)(\text{Home Runs}) + (0.35)(\text{Walks + Hit by Pitcher})$$



# VBA Output: VIF Worksheet



**Variance Inflation Factor (VIF)** is a statistical measurement of how much multicollinearity exists between the independent variables in a regression model.

$$\text{VIF} = 1 / (1 - R^2)$$

	A	B	C	D	E
1	<b>Variable</b>	<b>VIF</b>			
2	Singles	1.0671			
3	Doubles	1.1089			
4	Triples	1.1094			
5	Home Runs	1.3435			
6	Walks + Hit by Pitcher	1.2934			
7					
8					
9					

Start Regression | Runs (Reg) | **Runs (VIF)** | Runs (Cor) | (+)

## How to Interpret VIF

VIF	Definition
$\leq 4$	No multicollinearity problem. <b>Good</b>
$4 < \text{VIF} \leq 10$	Possible multicollinearity problem. Further investigation is necessary. <b>Maybe Good</b>
$\text{VIF} > 10$	Signs of serious multicollinearity requiring correction. <b>Bad</b>



# VBA Output: Correlation Worksheet



	A	B	C	D	E	F	G	H
1	<b>Correlation</b>	<i>Singles</i>	<i>Doubles</i>	<i>Triples</i>	<i>Home Runs</i>	<i>Walks + Hit by Pitcher</i>	<i>Runs</i>	
2	Singles	1						
3	Doubles	0.117400377	1					
4	Triples	0.187847856	0.048480649	1				
5	Home Runs	-0.131101525	0.233092574	-0.229331386	1			
6	Walks + Hit by Pitcher	-0.089842683	0.200714225	-0.204903229	0.450412469	1		
7	Runs	0.401147318	0.496219735	0.032766113	0.706166539	0.571158503	1	
8								
9								

Dependent variable's correlation

↓

Correlation Coefficient Value	Direction and Strength of Correlation
-1	Perfectly negative
-0.8	Strongly negative
-0.5	Moderately negative
-0.2	Weakly negative
0	No association
0.2	Weakly positive
0.5	Moderately positive
0.8	Strongly positive
1	Perfectly positive



# Excel Data Visualization





# Data Visualization of Regression Result

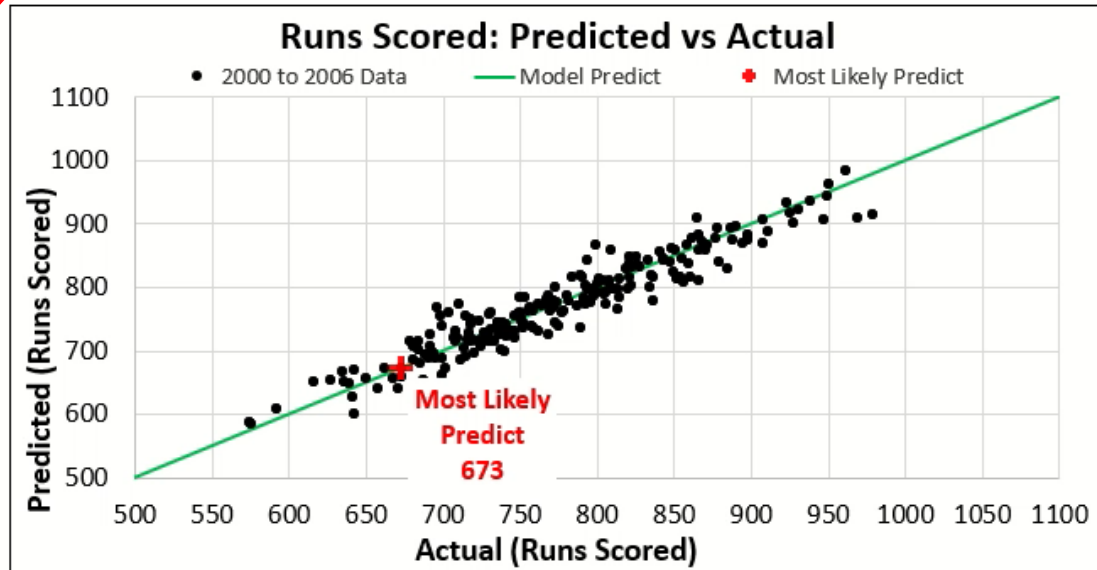


Linear Weights:						
Intercept	Singles	Doubles	Triples	Home Runs	Walks + Hit by Pitcher	
-559.9971	0.6328	0.7059	1.2637	1.4907	0.3466	

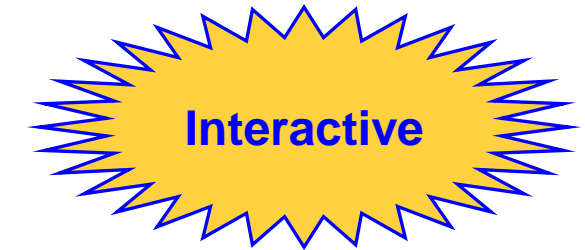
User Inputs:						
	Intercept	Singles	Doubles	Triples	Home Runs	Walks + Hit by Pitcher
Min:	NA	850	201	12	116	427
Max:	NA	1186	373	61	260	823
Input:	-559.99708	900	250	30	150	650

Expected Runs Scored:

673



View and change the model.



The model responds.

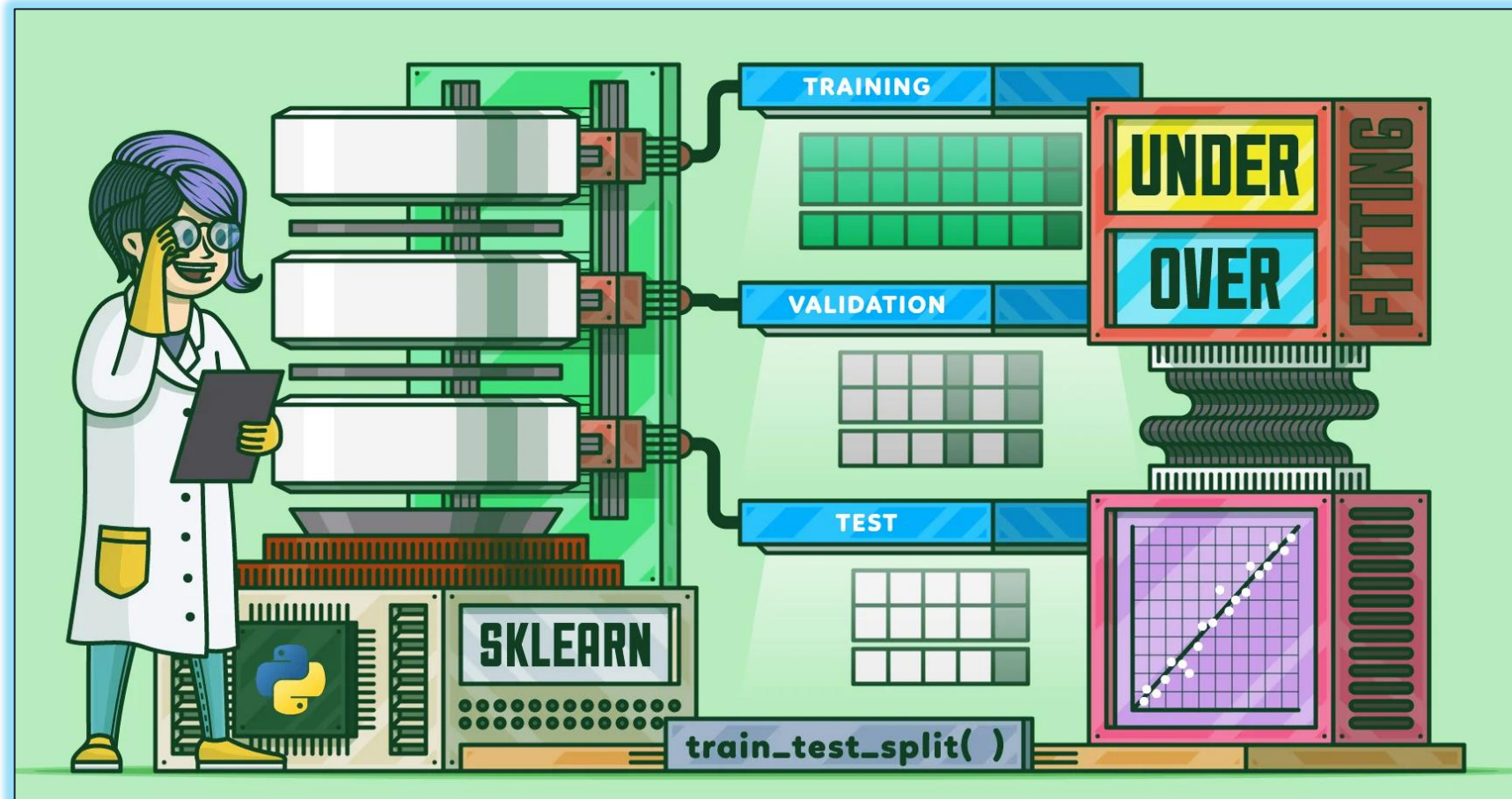


Regression equation is programmed into cell.

**Runs** = -559.997 + (0.63)(Singles) + (0.71)(Doubles) + (1.26)(Triples) + (1.49)(Home Runs) + (0.35)(Walks + Hit by Pitcher)



# Performing Regression Using Python





# Python Regression Software



- **Jupyter Notebook**– interface for designing Python software.

- **Python Linear Regression Software**

Dependent Variable ~ Independent Variables

C() = Categorical

Attendance\_Per\_Home\_Game ~ Payroll\_Ranking + W + Singles + HR + C(franchID) + C(Playoff\_Rank)

```
In [65]: # Library for Linear Regression model.  
import statsmodels.formula.api as smf
```

```
In [66]: # Fit the data to the Linear Regression model by using the Ordinary Least Squares (OLS) method.  
Attendance_Model = smf.ols('Attendance_Per_Home_Game ~ Payroll_Ranking + W + Singles + HR \\  
+ C(franchID) + C(Playoff_Rank)', data=mlb_stats_1998_2023).fit()
```



# Python Regression Output



Python Software



```
In [74]: # View the Linear Regression model.  
Attendance_Model.summary()
```

Python Output



Out[74]:

OLS Regression Results

Dep. Variable:	Attendance_Per_Home_Game	R-squared:	0.785
Model:	OLS	Adj. R-squared:	0.772
Method:	Least Squares	F-statistic:	61.87
Date:	Mon, 03 Feb 2025	Prob (F-statistic):	1.70e-173
Time:	20:41:33	Log-Likelihood:	-6126.5
No. Observations:	630	AIC:	1.232e+04
Df Residuals:	594	BIC:	1.248e+04
Df Model:	35		
Covariance Type:	nonrobust		

Continued...

## Important Results

1. Adjusted R Squared
2. Coefficients
3. P-values



# Python Regression Output



Python Output Continued...

## Important Results

1. Adjusted R Squared
2. **Coefficients**
3. **P-values**

Ind. Variables	coef	std err	t	P> t	[0.025	0.975]
Intercept	1.954e+04	3633.585	5.377	0.000	1.24e+04	2.67e+04
C(franchID)[T.ARI]	-716.1728	1317.409	-0.544	0.587	-3303.518	1871.173
C(franchID)[T.ATL]	-2097.1898	1301.817	-1.611	0.108	-4653.914	459.534
C(franchID)[T.BAL]	-3016.0587	1312.136	-2.299	0.022	-5593.050	-439.068
•			•			•
•			•			•
•			•			•
C(franchID)[T.WSN]	-6463.0043	1321.204	-4.892	0.000	-9057.803	-3868.206
C(Playoff_Rank)[T.NoPlayoffs]	-1333.2160	621.944	-2.144	0.032	-2554.693	-111.739
C(Playoff_Rank)[T.WildCard]	-1192.8129	662.970	-1.799	0.072	-2494.863	109.237
Payroll_Ranking	-481.2423	29.475	-16.327	0.000	-539.129	-423.355
W	104.3063	23.724	4.397	0.000	57.713	150.899
Singles	11.2934	2.738	4.125	0.000	5.917	16.670
HR	10.0529	6.048	1.662	0.097	-1.824	21.930



# Python Multicollinearity



## • GVIF Values

Python Output →

## Generalized Variance Inflation Factor

- Less than 5 is generally considered acceptable.

$$GVIF = VIF^{(1/(2 * df))}$$

## Statsmodels Library

```
# Create GVIF function.
from statsmodels.stats.outliers_influence import variance_inflation_factor
def calculate_gvif(df, target_column):
    """Calculates the GVIF for each feature in a pandas DataFrame."""
    X = df.drop(columns=[target_column])
    vif_data = pd.DataFrame()
    vif_data["Feature"] = X.columns
    vif_data["VIF"] = [variance_inflation_factor(X.values, i) for i in range(X.shape[1])]
    vif_data["df"] = [X.iloc[:, i].nunique() - 1 for i in range(X.shape[1])]
    vif_data["GVIF"] = vif_data["VIF"] ** (1 / (2 * vif_data["df"]))
    return vif_data
```

Feature	GVIF
Payroll_Ranking	1.014922
W	1.009128
Singles	1.000739
HR	1.001904
Playoff_Rank_NoPlayoffs	1.724298
Playoff_Rank_WildCard	1.222865
franchID_ANA	3.995659
franchID_ARI	3.920768
franchID_ATL	3.952842
franchID_BAL	3.990779
•	•
•	•
•	•
franchID_SEA	4.036322
franchID_SFG	3.891175
franchID_STL	3.969560
franchID_TBD	3.989647
franchID_TEX	4.038921
franchID_TOR	4.032929
franchID_WSN	3.912310



# Python Data Visualization

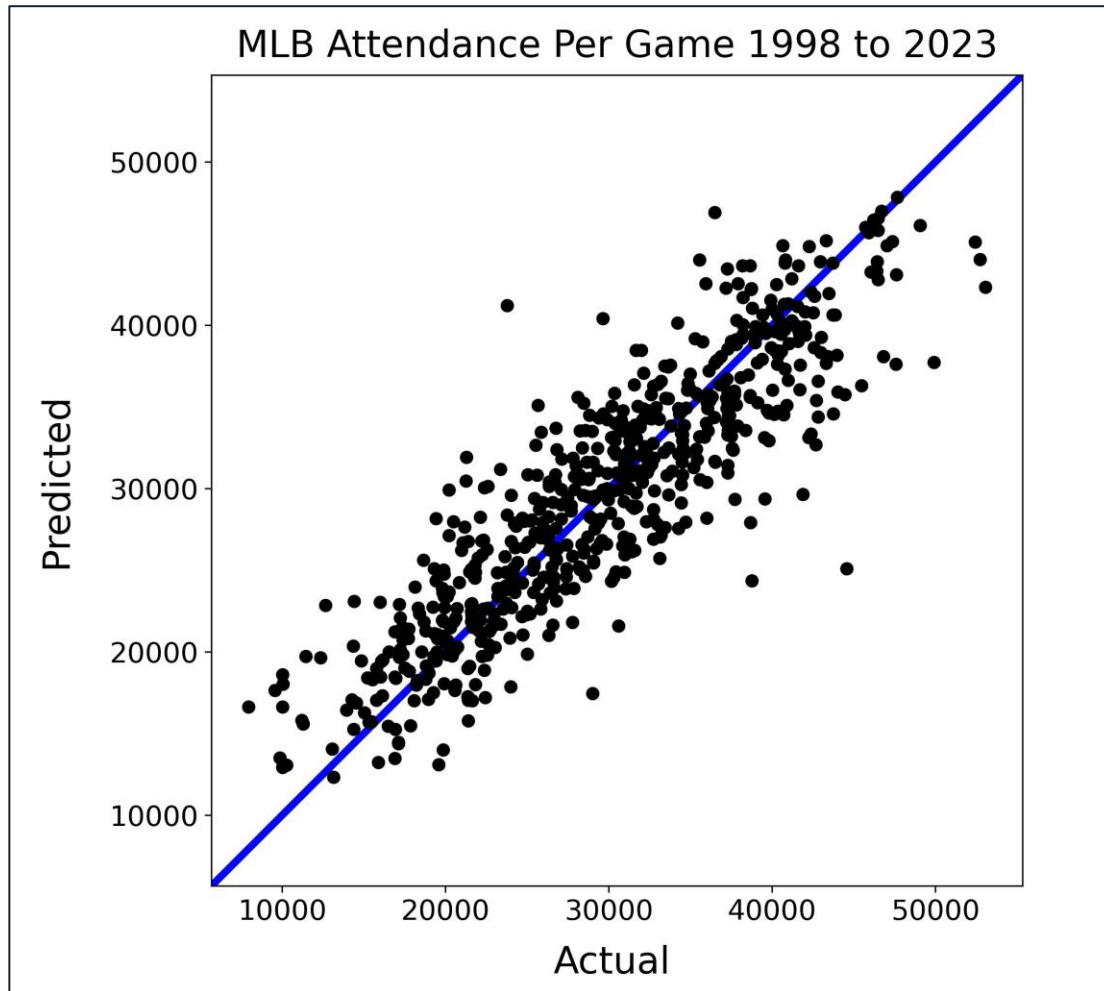




# Data Visualization of Regression Result



## ● Predicted Data vs Actual Data



The closer the datapoints are to the  $y = x$  line, the more accurate the model's prediction is.

### Matplotlib Library

```
# Plot predicted vs actual.  
ax = new_df.plot.scatter(x='actual', y='predicted', title='MLB  
  
# plot y=x line:  
lims = [  
    np.min([ax.get_xlim(), ax.get_ylim()]), # min of both axes  
    np.max([ax.get_xlim(), ax.get_ylim()]), # max of both axes  
]  
  
# now plot both limits against eachother.  
ax.plot(lims, lims, alpha=0.75, zorder=0, color='blue')  
ax.set_aspect('equal')  
ax.set_xlim(lims)  
ax.set_ylim(lims)
```

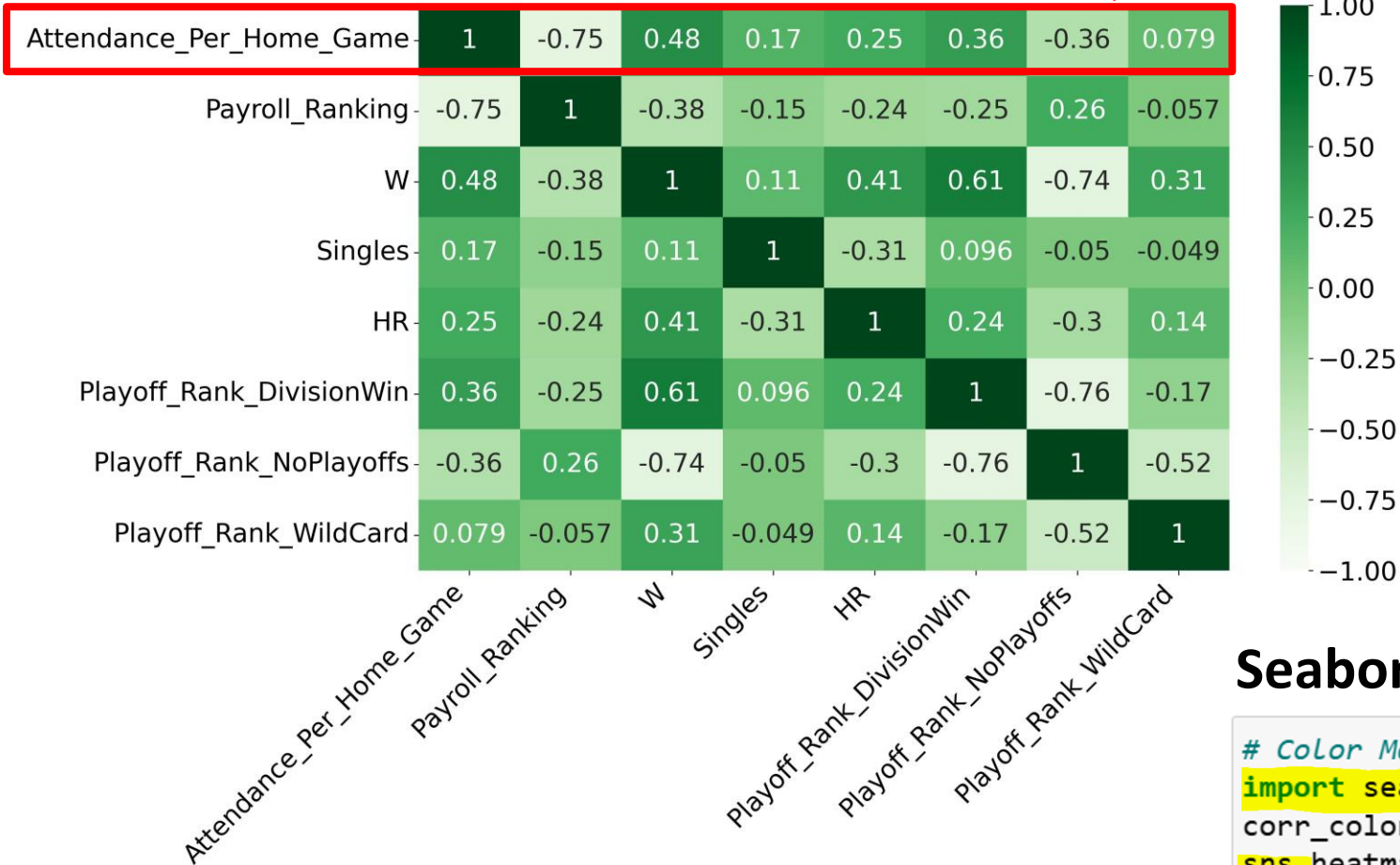


# Data Visualization of Correlation



## Correlation Heatmap

Dependent variable's correlation

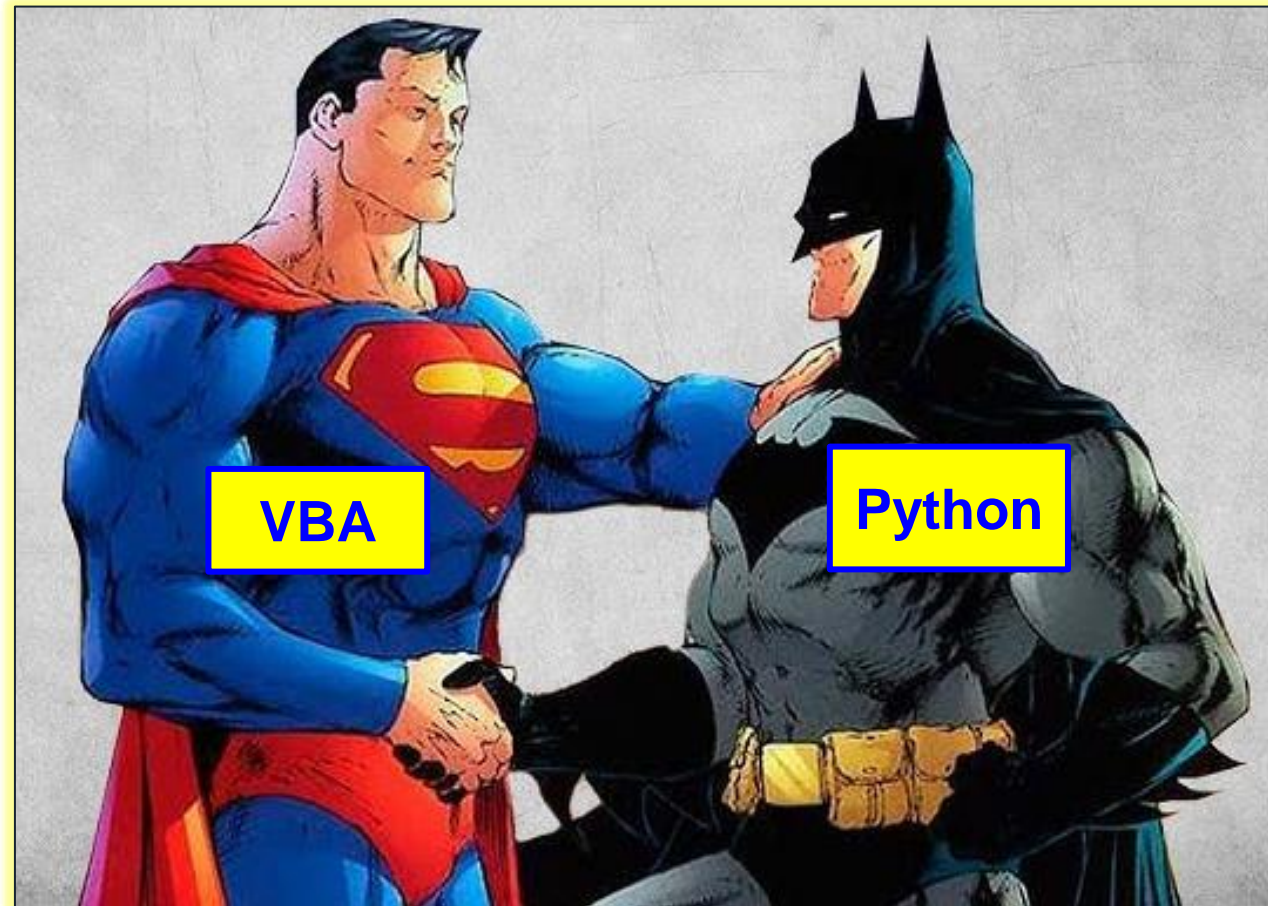


## Seaborn Library

```
# Color Matrix: Plotting correlation matrix.
import seaborn as sns
corr_color_matrix = corr_color_df.corr()
sns.heatmap(corr_color_matrix, cmap="Greens", annot=True)
```



# Performing Regression Using VBA and Python



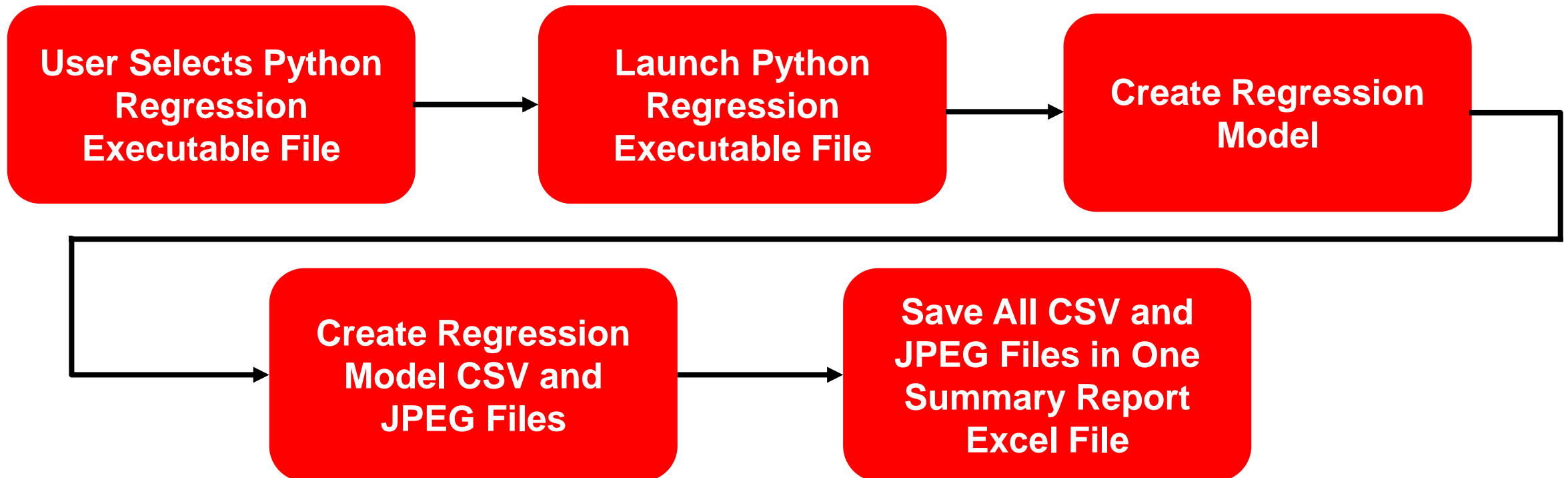


# Use Excel VBA to Run Python Regression



*VBA: For user interaction & regression visualization.*

*Python: For generating regression model.*





# Excel VBA Running Python Regression



## Python Executable Folder



## VBA Automation Tool



The screenshot displays an Excel spreadsheet with a VBA Python Tool interface. The interface includes a title bar "VBA Python Tool" and four buttons: "Start" (green text), "Delete Tabs" (orange text), "Load File" (black text), and "Clear" (black text). Below the buttons, there is a label "Input" and a text field containing the path "C:\Users\imgiann02\Documents\ICEAA 2025 Baseball Stats\dist\MLB\_Stats\_1998". The spreadsheet grid shows columns A through L and rows 1 through 17. The "Start" button is highlighted in the bottom status bar.



# Excel and Python Regression Comparison



	Excel	Python
Types of Regression	Has Limitations	No Limitations
Data Variables	Has Limitations	No Limitations
Size of Data	Has Limitations	Proficient
Multicollinearity	Has Limitations	Easy
Correlation	Easy	Easy
Automation Driven	Easy	Easy
User Interface Driven	Easy	Difficult
Accessibility	Extremely Easy	Labor Intensive

*Eliminate All Weakness*

Two black spotlights are positioned at the top of the slide, angled downwards. They cast bright yellow beams of light that converge on the title text below. The beams create a soft, glowing effect around the text.

# General Steps and Examples



# Combining VBA and Python Flow Diagram



Only complete first time!



1. Create Python Code



2. Convert .py into .exe



3. Collect inputs from user



4. Create arguments variable (if needed)



5. Create command variable



6. Run Shell command to execute .exe file



7. Generate output



8. Integrate output into Excel (if desired)



# How to Combine VBA And Python Code



```
'Path to the .exe file  
exePath = Worksheets("Info").Range("A3").Value
```

```
'Get individual arguments  
TP_arg = Worksheets("Info").Range("A1").Value  
TM_arg = Worksheets("Info").Range("A2").Value  
'Put new excel sheet in the same folder as the .exe file  
excel_output_arg = Worksheets("Info").Range("A3").Value  
excel_output_arg = left$(excel_output_arg, InStrRev(excel_output_arg, "\") - 1)  
excel_output_arg = excel_output_arg & "\test.xlsx"  
  
'Put together all arguments  
arguments = "" & TP_arg & "" & TM_arg & "" & excel_output_arg & ""
```

Gather arguments and define arguments variable

```
'Run the .exe file with arguments, wait for exe to finish before continuing  
command = "" & exePath & "" & arguments  
CreateObject("Wscript.Shell").Run command, 0, True '0 = hide window, 1 = show window; true = wait on return
```

Command String

Runs Shell command to execute .exe file



# Combining VBA and Python



*General Use Case: VBA for user interaction, Python for processing data.*

## Example Scenarios:

1. Gathering (Python) and displaying (VBA) data from a COM API.
2. Create XML document (Python) from Excel data (VBA).
3. Display results (VBA) and use data from separate tools (Python).



# Thank You



# Resources

---

- Useful Python libraries:
  - Pandas documentation: <https://pandas.pydata.org/docs/>
  - Element Tree documentation: <https://docs.python.org/3/library/xml.etree.elementtree.html>
  - NumPy documentation: <https://numpy.org/doc/>
- PyInstaller documentation: <https://pyinstaller.org/en/stable/>
- *Mathletics* by Wayne L. Winston. (2009)
- Clip Art Images:
  - VBA Regression: <https://towardsdatascience.com/linear-regressions-for-causal-conclusions-34c6317c5a11/>
  - Python Regression: <https://realpython.com/train-test-split-python-data/>
  - Python Data Visualization: <https://ioflood.com/blog/sklearn-linear-regression-python/>