Lab_05

March 10, 2025

Probability for Data Science

UC Berkeley, Spring 2025

Michael Xiao and Ani Adhikari

[1]:
```
from datascience import *
from prob140 import *
import numpy as np
from scipy import stats
import matplotlib.pyplot as plt
plt.style.use('fivethirtyeight')
%matplotlib inline
```

1 Lab 5: Multinomial Correlations (Due Monday, March 10th at 5PM)

Many questions in data science involve populations that can be partitioned into categories of individuals. For example, in Data 8 you classified individuals into one of two classes. Data 100 examines logistic regression as a different approach to binary classification. These methods can be extended to multiple categories.

At a more fundamental level, it is important to understand how a random sample from a categorical population reflects the population. As you know, the multinomial is the joint distribution of the counts of the different categories in a random sample of a fixed size.

More precisely, suppose the population has k categories of individuals in proportions $p_1, p_2, \ldots, p_k$ with $\sum_{i=1}^{k} p_i = 1$. For $1 \le i \le k$ let N_i be the number of elements in Category i in a sample of size n drawn at random with replacement from the population. Then the joint distribution of of $N_1, N_2, \ldots, N_k$ is multinomial with parameters n and $p_1, p_2, \ldots, p_k$.

The counts $N_1, N_2, \ldots, N_k$ are dependent since $\sum_{i=1}^{k} N_i = n$ which is a constant. In this lab you will study the dependence between pairs N_i and N_j and develop a way to predict N_j based on the value of N_i. You will see that the method is closely connected to the linear regression that you studied in Data 8.

What you will learn in this lab:

- How to simulate multinomial counts in Python
- How to find the covariance of two multinomial counts and hence the correlation between them
- How to predict one multinomial count based on another, and how this is connected to linear regression

A few computational preliminaries will help you get started.

1.1 Instructions

Similar to your homeworks, your labs will generally have two components: a written portion and a portion that also involves code.
- Written work should be completed on paper, and coding questions should be done in the notebook.
- Start the work for the written portions of each section on a new page. - You are welcome to LaTeX your answers to the written portions, but staff will not be able to assist you with LaTeX related issues. - Show your work. Give reasoning. The question isn't always going to ask for it, because we assume that you will provide justification for your answers. Every answer should contain a calculation, reasoning, or diagrams that are clearly labeled to show what's going on. - It is your responsibility to ensure that both components of the lab are submitted completely and properly to Gradescope. **Make sure to assign each page of your pdf to the correct question. - Refer to the bottom of the notebook for submission instructions.**

1.2 Identify Your Lab Partner

This is a multiple choice question. Please select **ONE** of following options that best describes how you complete this lab.

- I am doing this lab by myself and I don't have a partner.
- My partner for this lab is [PARTNER'S NAME] with email [berkeley.edu email address]. [SUBMITTER'S NAME] will submit to Gradescope and add the other partner to the group on Gradescope after submission.

Please copy and paste **ONE** of above statements and fill in blanks if needed. If you work with a partner, make sure only one of you submit on Gradescope and that the other member of the group is added to the submission on Gradescope. Refer to the bottom of the notebook for submission instructions.

I am doing this lab by myself.

1.3 Section 1: Preliminaries

1.3.1 1a) Simulating a Multinomial

`np.random.multinomial(n, p_array)` simulates `n` draws at random with replacement from the categorical distribution `p_array` and returns an array of the observed counts in all the categories.

You have seen a version of this in Data 8. The `datascience` function `sample_proportions` is just `np.random.multinomial` with the output converted to proportions.

Run the cell below to draw 10 times from a population in which 20% are in Category A, 70% in Category B, and 10% in Category C. The output array contains the observed counts in Categories A, B, and C.

Confirm that it makes sense, by looking at the total count and the count in each category. Then run the cell a few times to see how the observations change.

[2]:
```
num_draws = 10
population_proportions = [0.2, 0.7, 0.1] # make_array(0.2, 0.7, 0.1) is also␣
  ↪fine
np.random.multinomial(num_draws, population_proportions)
```

[2]:
```
array([2, 5, 3])
```

1.3.2 1b) Adding Rows to a Table

You know that `t.with_columns` can be used to add a column or columns to a table `t`. Sometimes it is useful to grow a table by adding rows. Start by creating the table with just the column labels and no rows, as follows.

[3]:
```
t = Table(['Column A', 'Column B', 'Column C'])
t
```

[3]:
```
Column A | Column B | Column C
```

Now `append` a row to the table. It is important to note that `t.append` mutates (that is, changes) the table `t`. Unlike the usual Table operations, it doesn't just display a temporary copy of `t` that you have to name if you want to save it. Run the two cells below and keep track of what happens to `t`.

[4]:
```
t.append([1, 2, 3])
```

[4]:
```
Column A | Column B | Column C
1        | 2        | 3
```

[5]:
```
# t has been changed
t
```

[5]:
```
Column A | Column B | Column C
1        | 2        | 3
```

To append another row, use `append` again.

[6]:
```
t.append([4, 5, 6])
```

[6]:
```
Column A | Column B | Column C
1        | 2        | 3
4        | 5        | 6
```

[7]:
```
# t has been changed
t
```

[7]:
```
Column A | Column B | Column C
1        | 2        | 3
4        | 5        | 6
```

1.3.3 1c) Putting the Two Together

Create a table `simulated_counts` that has column labels `A`, `B`, and `C`. The table should have two rows. Each row should contain the observed counts in Categories A, B, and C in 10 multinomial trials, each of which results in Category A with chance 0.2, Category B with chance 0.7, and Category C with chance 0.1 as in Part **a** above. The trials in each row should be independent of those in the other row.

You should use no more than 3 lines of code.

[8]:
```
# Answer to 1c

simulated_counts = Table(['A', 'B', 'C'])
simulated_counts.append(np.random.multinomial(num_draws,␣
 ↪population_proportions))
simulated_counts.append(np.random.multinomial(num_draws,␣
 ↪population_proportions))
```

[8]:
```
A    | B    | C
1    | 8    | 1
6    | 4    | 0
```

1.4 Section 2: Marginals

Each spin of a Nevada roulette wheel results in the winning color being red with chance 18/38, black with chance 18/38, and green with chance 2/38, independent of all other spins.

A Nevada roulette wheel is spun 380 times. Let R be the number of times red wins, let B be the number of times black wins, and let G be the number of times green wins.

1.4.1 2a) The Distributions

Fill in the blanks below with the names of famous distributions and the relevant parameters.

(i) The joint distribution of R, B, and G is _____ with parameters _____.

(ii) R has the _____ ( ) distribution.

(iii) B has the _____ ( ) distribution.

(iv) G has the _____ ( ) distribution.

2 ALL independent

i. multinomial with parameters n = 380 and probabilities = (18/38, 18/38, 2/38)

ii. binomial(380, 18/38)

iii. binomial(380, 18/38)

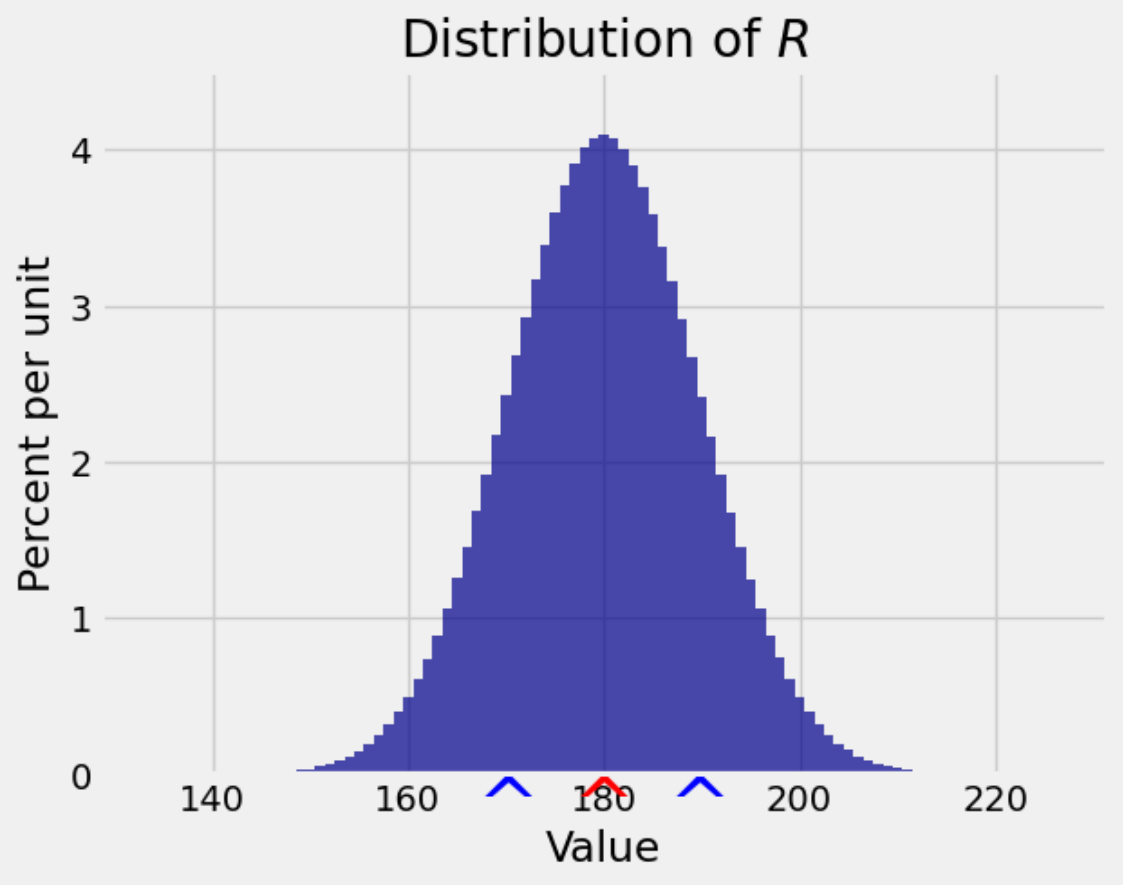
iv. binomial(380, 2/38)

2.0.1 2b) Probability Histogram of R

Draw the probability histogram of R.

[9]:
```
# Answer to 2b

k = np.arange(130, 231)
p_k = stats.binom.pmf(k, 380, 18/38)
dist_R = Table().values(k).probabilities(p_k)
Plot(dist_R, show_ev=True, show_sd=True)
plt.title('Distribution of $R$');
```



2.0.2 2c) Expectations and SDs

Use Part **a** to write numerical expressions for the expectations and SDs of the three random counts. Check that the values of $E(R)$ and $SD(R)$ are consistent with the histogram in Part **b**.

[10]:
```
# Answer to 2c

exp_R = 380 * (18/38)  # E(R)
sd_R = np.sqrt(exp_R * (1 - 18/38))    # SD(R)
exp_B = 380 * (18/38)
sd_B = np.sqrt(exp_B * (1 - 18/38))
exp_G = 380 * (2/38)
sd_G = np.sqrt(exp_G * (1 - 2/38))
exp_R, sd_R, exp_B, sd_B, exp_G, sd_G
```

[10]:
```
(180.0, 9.7332852678457531, 180.0, 9.7332852678457531, 20.0, 4.35285750066007)
```

2.0.3 2d) Empirical Histogram of R

Create a table `simulated` that has three columns labeled `R`, `B`, and `G`. The table should have 10,000 rows. Each row should consist of the observed values of R, B, and G in one set of 380 spins of the roulette wheel. The spins in each row should be independent of those in the other rows.

Then draw an empirical histogram of the distribution of R and compare it briefly with the histogram in Part **b**. Use as many lines as you need before the last two lines provided.

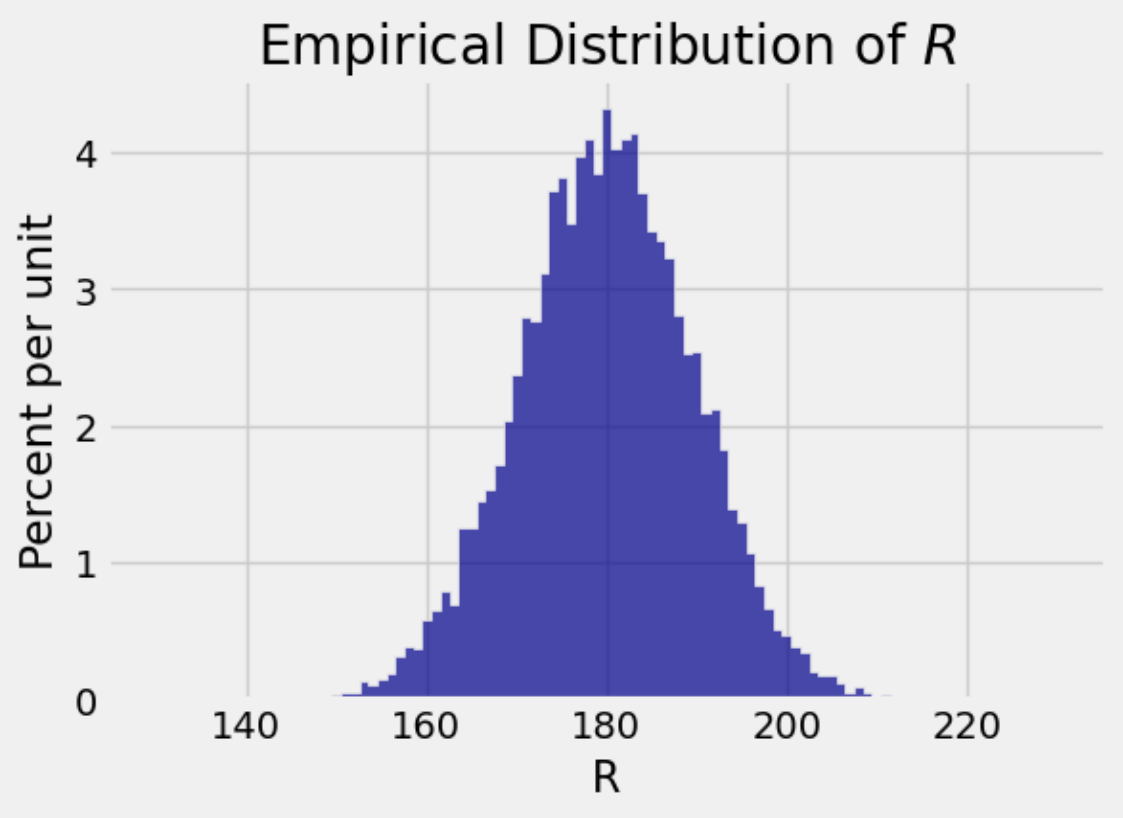
[11]:
```
# Answer to 2d

pop_proportions = [18/38, 18/38, 2/38]
spins = 380
repetitions = 10000

simulated = Table(['R','B','G'])

for i in np.arange(repetitions):
    temp = np.random.multinomial(spins, pop_proportions)
    simulated.append(temp)


simulated.hist('R', bins=np.arange(129.5, 231, 1))
plt.title('Empirical Distribution of $R$'); # COMPARE WITH 2b
```



2.0.4 2e) Empirical Mean and SD

Find the mean and SD of the 10,000 simulated values of R and compare them to $E(R)$ and $SD(R)$ as found in Part **c**.

[12]:
```
# Answer to 2e

emp_mean_R = np.mean(simulated.column('R'))
emp_SD_R = np.std(simulated.column('R'))
emp_mean_R, emp_SD_R  # COMPARE TO E(R) AND SD(R)
```

[12]:
```
(179.95599999999999, 9.7387301020204902)
```

2.1 Section 3: Covariance and Correlation

The multinomial is a joint distribution, so now let's look at the relation between R and the other two variables.

2.1.1 3a) Sign (+ or −) of Correlation

Just based on your Data 8 recollection of properties of correlation, what would you guess as the sign of the correlation between R and B, and why?

The sign of the correlation between R and B would be negative because the number of red wins and black wins are dependent, constrained by 380 spins. If there are more red wins, there are fewer black wins, and if there are more black wins, there are fewer red wins. In addition, we take into account three possible outcomes (red, black, and green), where the number of one outcome will indefinitely lead to fewer outcomes for the other colors.

2.1.2 3b) [On Paper] Covariance of R and B

Find $Cov(R, B)$ using both the methods below. The first is a direct calculation using the most important property of covariance. The second is a consequence of the particular structure of R, B, and G.

Method 1, using bilinearity: For each spin i, define indicators X_i and Y_i such that $R = \sum_{j=1}^{380} X_j$ and $B = \sum_{k=1}^{380} Y_k$. Then use bilinearity to find $Cov(R, B)$ and check whether its sign is the same as in Part **a**.

[It will help to keep in mind that for $j \neq k$, X_j is independent of Y_k. Then think carefully about $Cov(X_j, Y_j)$.]

Method 2, using the variance of a sum: Identify the distribution of $R + B$ and hence find $Var(R + B)$. Then use Part **2c** and the formula for the variance of a sum to find $Cov(R, B)$.

2.1.3 3c) [On Paper] Correlation

The covariance of random variables X and Y has nasty units: the product of the units of X and the units of Y. Dividing the covariance by the two SDs results in an important pure number.

Define the *correlation coefficient* between random variables X and Y as

$$Corr(X, Y) = \frac{Cov(X, Y)}{SD(X)SD(Y)}$$

and is called *correlation* for short.

Find $Corr(R, B)$.

2.1.4 3d) Empirical Correlation

Let `v_1` and `v_2` be numerical arrays of the same length. The expression `np.corrcoef(v_1, v_2)` evaluates to a 2×2 *correlation matrix* whose (i, j) element is the correlation (as defined in Data 8) between `v_i` and `v_j`. Thus both the diagonal elements are equal to 1, and both the off-diagonal elements are the correlation between `v_1` and `v_2` just as you would have calculated it in Data 8.

Use **2d** to get the empirical correlation between R and B based on 10,000 simulations, in a correlation matrix `corr_matrix`. Make sure the result is consistent with your answer to Part **c** above.

Make sure the result is consistent with your answer to Part **c** above.

Later in the lab, to access just the off-diagonal element you can use `corr_matrix[0, 1]`.

[13]:
```
# Answer to 3d

v_1 = simulated.column('R')
v_2 = simulated.column('B')

corr_matrix = np.corrcoef(v_1, v_2)
corr_matrix
```

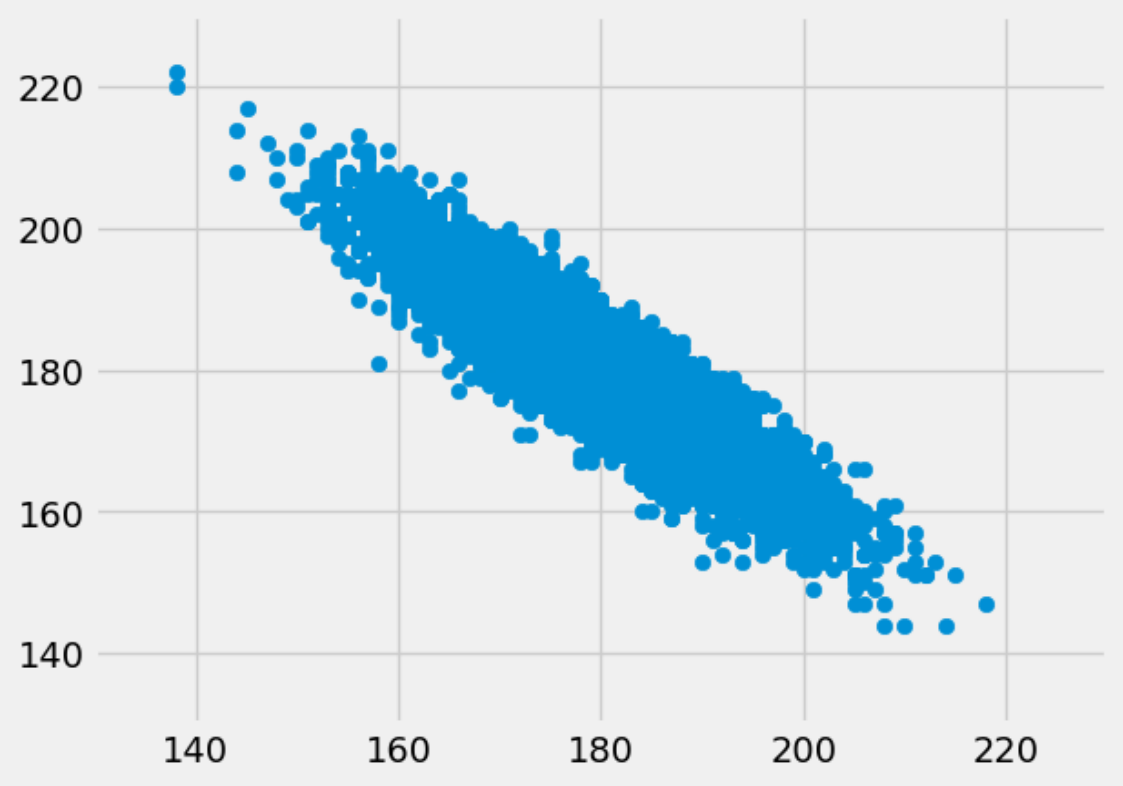
[13]:
```
array([[ 1.        , -0.90148029],
       [-0.90148029,  1.        ]])
```

2.1.5 3e) Visualization

Draw the scatter plot of the 10,000 (R, B) points simulated in **2d**, with R on the horizontal axis. Use only one line of code before the two lines provided. Check that the plot looks consistent with the correlation you found in Part **d**.

[14]:
```
# Answer to 3e

plt.scatter(v_1, v_2)
plt.xlim(130, 230)
plt.ylim(130, 230);
```


220
200
180
160
140
140
160
180
200
220

2.2 Section 4: Conditional Expectation

Clearly, there's a straight line going through the scatter plot in **3e**. Let's try to find a plausible candidate for that line.

2.2.1 4a) [On Paper] A Conditional Distribution

Let $0 \leq r + b \leq 380$.

Find $P(B = b \mid R = r)$. Use your formula to answer the following.

(i) Identify the conditional distribution of B given $R = r$ as one of the famous ones and provide its parameters.

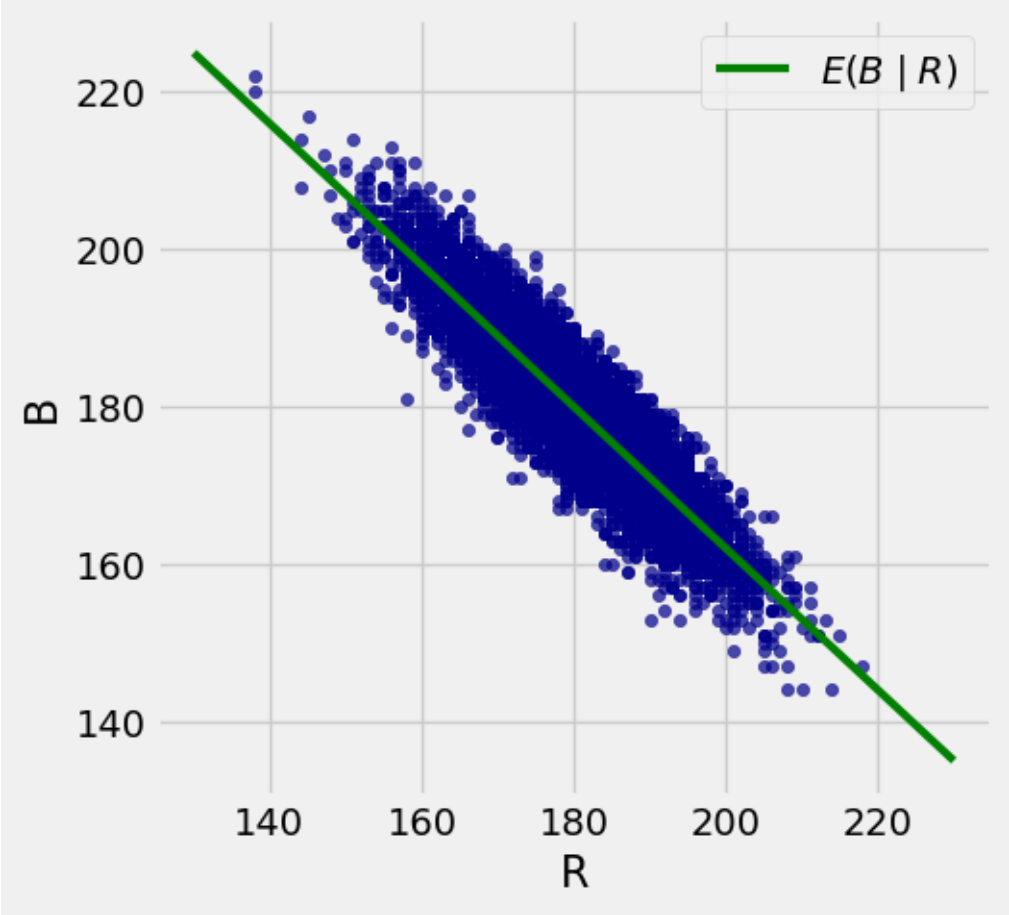
(ii) Find $E(B \mid R)$ and show that it is a linear function of R.

2.2.2 4b) Another Visualization

Redraw the scatter plot in **3e** and overlay the line $R \to E(B \mid R)$.

[15]:
```
# Answer to 4b

r = np.arange(130, 231)
exp_B_given_r = 342 - 0.9 * r  # E(B | R=r)
simulated.scatter('R', 'B')
plt.plot(r, exp_B_given_r, color='green', lw=3, label='$E(B \\mid R)$')
plt.legend();
```



Later in the term we'll see that $E(B \mid R)$ is the best among all predictors of B based on R, in the sense of least squared error. The result is valid for any two jointly distributed random variables and does not involve observed data.

2.2.3 4c) The Regression Line Based on Data

In Data 8, the regression line was calculated as the best straight line for prediction using the data points at hand. In our current setting that would be the least squares line based on the 10,000 simulated points, not based on the underlying distribution of those points like the conditional expectation line you found in Part **b** above.

Let's calculate the Data 8 line based on the empirical averages and SDs in **2e**. Recall the formulas for the slope and intercept of the regression line, and find the slope and intercept using the simulated data in **2d**.

Remember that in **2e** you already found `emp_mean_R`, the empirical mean of R, and `emp_SD_R`, the empirical SD of R. In **3d** you found the empirical correlation between R and B. You just need a couple of quantities related to B, and then you can find the slope and intercept of the regression line based on the 10,000 simulated points.

[16]:
```
# Answer to 4c

emp_mean_B = np.mean(simulated.column('B'))
emp_SD_B = np.std(simulated.column('B'))
reg_slope = corr_matrix[0,1] * (emp_SD_B / emp_SD_R)
reg_intercept = emp_mean_B - reg_slope * emp_mean_R
reg_slope, reg_intercept
```

[16]:
```
(-0.90511232980058431, 342.89299442159393)
```

The resulting line is an excellent estimate of the underlying conditional expectation line $R \to E(B \mid R)$ that you plotted in **4b**.

2.3 Conclusion

What you have learned in this lab:

- How to simulate multinomial variables in Python
- How probability distributions and properties such as expectation, standard deviation, and correlation are related to empirical distributions and their corresponding properties
- How to find the correlation between two multinomial counts
- That conditional expectation can be thought of as a way to make predictions
- That for predicting one multinomial count based on another, based on a large amount of data, the empirical regression line looks very much like the conditional expectation of the response given the predictor

2.4 Submission Instructions

Many assignments throughout the course will have a written portion and a code portion. Please follow the directions below to properly submit both portions.

2.4.1 Written Portion

- Scan all the pages into a PDF. You can use any scanner or a phone using applications such as CamScanner. Please **DO NOT** simply take pictures using your phone.
- Please start a new page for each question. If you have already written multiple questions on the same page, you can crop the image in CamScanner or fold your page over (the old-fashioned way). This helps expedite grading.
- It is your responsibility to check that all the work on all the scanned pages is legible.
- If you used LaTeX to do the written portions, you do not need to do any scanning; you can just download the whole notebook as a PDF via LaTeX.

2.4.2 Code Portion

- Save your notebook using `File > Save Notebook`.
- Generate a PDF file using `File > Save and Export Notebook As > PDF`. This might take a few seconds and will automatically download a PDF version of this notebook.
  - If you have issues, please post a follow-up on the general Lab 5 Ed thread.

2.4.3 Submitting

- Combine the PDFs from the written and code portions into one PDF. Here is a useful tool for doing so.
- Submit the assignment to Lab 5 on Gradescope.
- **Make sure to assign each page of your pdf to the correct question.**
- **It is your responsibility to verify that all of your work shows up in your final PDF submission.**

If you are having difficulties scanning, uploading, or submitting your work, please read the Ed Thread on this topic and post a follow-up on the general Lab 5 Ed thread.

2.5 We will not grade assignments that do not have pages selected for each question.

3b. Method 1

$$R = \sum_{j=1}^{380} X_j \qquad B = \sum_{k=1}^{380} Y_k$$

$$\mathrm{Cov}(R, B) = \mathrm{Cov}\left(\sum_{j=1}^{380} X_j, \sum_{k=1}^{380} Y_k \right)$$

$$= \sum_{j=1}^{380} \sum_{k=1}^{380} \mathrm{Cov}(X_j, Y_k)$$

$$= \sum_{j=1}^{380} \sum_{k=1}^{380} \left[\underbrace{E[X_j Y_k]} - E[X_j] E[Y_k] \right]$$

expectation of getting red and black in one spin → 0

∴ can only ever be one color

$$= \sum_{j=1}^{380} \sum_{k=1}^{380} \left[0 - \left(\frac{18}{38} \times \frac{18}{38} \right) \right]$$

$$= \sum_{j=1}^{380} \sum_{k=1}^{380} -\left(\frac{18}{38} \times \frac{18}{38} \right)$$

$$= -380 \left(\frac{18}{38} \right) \left(\frac{18}{38} \right)$$

$$\approx -85.27$$

Method 2

$R \sim$ # red in 380 spins

$B \sim$ # blue in 380 spins

$G \sim$ # green in 380 spins

$$R + B + G = 380$$

$$R + B = 380 - G$$

$$\therefore R + B \sim \text{Binomial}\left(380, \frac{18}{38} + \frac{18}{38}\right)$$

$$\sim \text{Binomial}\left(380, \frac{36}{38}\right)$$

$$\text{Var}(R + B) = np(1-p)$$

$$= 380\left(\frac{36}{38}\right)\left(1 - \frac{36}{38}\right)$$

$$= 380\left(\frac{36}{38}\right)\left(\frac{2}{38}\right)$$

$$\approx 18.95$$

$$\text{Var}(R + B) = \text{Var}(R) + \text{Var}(B) + 2\,\text{Cov}(R + B)$$

$$\text{Var}(R) = np(1-p)$$

$$= 380\left(\frac{18}{38}\right)\left(1-\frac{18}{38}\right)$$

$$= 380\left(\frac{18}{38}\right)\left(\frac{20}{38}\right)$$

$$\approx 94.74$$

$$Var(B) = np(1-p)$$

$$= 380\left(\frac{18}{38}\right)\left(1-\frac{18}{38}\right)$$

$$= 380\left(\frac{18}{38}\right)\left(\frac{20}{38}\right)$$

$$\approx 94.74$$

$$\therefore Var(R+B) = Var(R) + Var(B) + 2Cov(R+B)$$

$$18.95 = 94.74 + 94.74 + 2Cov(R+B)$$

$$-170.53 = 2Cov(R+B)$$

$$Cov(R+B) = -85.27$$

3c. $$Corr(R,B) = \frac{Cov(R,B)}{SD(R)\,SD(B)}$$

$$= \frac{-85.27}{(9.73)(9.73)}$$

$$= \frac{-85.27}{94.6729}$$

$$\approx -0.9$$

4ai. $P(B = b \mid R = r) = ?$

$$R + B + G = 380$$
$$B + G = 380 - R$$
$$B + G = 380 - r$$

$$n = 380 - r$$

$$p = \frac{\frac{18}{38}}{\frac{18}{38} + \frac{2}{38}} \cdot \frac{38}{38}$$

$$= \frac{18}{20}$$

$$B \mid R = r \sim \text{Binomial}\left(380 - r, \frac{18}{20}\right)$$

4aii. $E[B \mid R] = np$

$$= (380 - r)\left(\frac{18}{20}\right)$$

$$= 342 - \frac{18}{20} r$$

$$= 342 - 0.9r$$