

# Explore the data

MONTE CARLO SIMULATIONS IN PYTHON



Izzy Weber

Curriculum Manager, DataCamp

# The diabetes dataset

Ten independent variables:

- Age age
- Sex sex
- Body mass index bmi
- Average blood pressure bp
- Six blood serum measurements: tc , ldl , hdl , tch , ltg , glu

<sup>1</sup> Bradley Efron, Trevor Hastie, Iain Johnstone and Robert Tibshirani (2004) "Least Angle Regression," Annals of Statistics (with discussion), 407-499 <sup>2</sup> <https://www4.stat.ncsu.edu/~boos/var.select/diabetes.html>

# The diabetes dataset

## Dependent variable

- A quantitative measure of disease progression one year after baseline,  $y$

## Size of the dataset

- 442 diabetes patients

# The diabetes dataset

```
dia.head()
```

	age	sex	bmi	bp	tc	ldl	hdl	tch	ltg	glu	y
0	59	2	32.1	101.00	157	93.2	38.0	4.00	4.8598	87	151
1	48	1	21.6	87.00	183	103.2	70.0	3.00	3.8918	69	75
2	72	2	30.5	93.00	156	93.6	41.0	4.00	4.6728	85	141
3	24	1	25.3	84.00	198	131.4	40.0	5.00	4.8903	89	206
4	50	1	23.0	101.00	192	125.4	52.0	4.00	4.2905	80	135

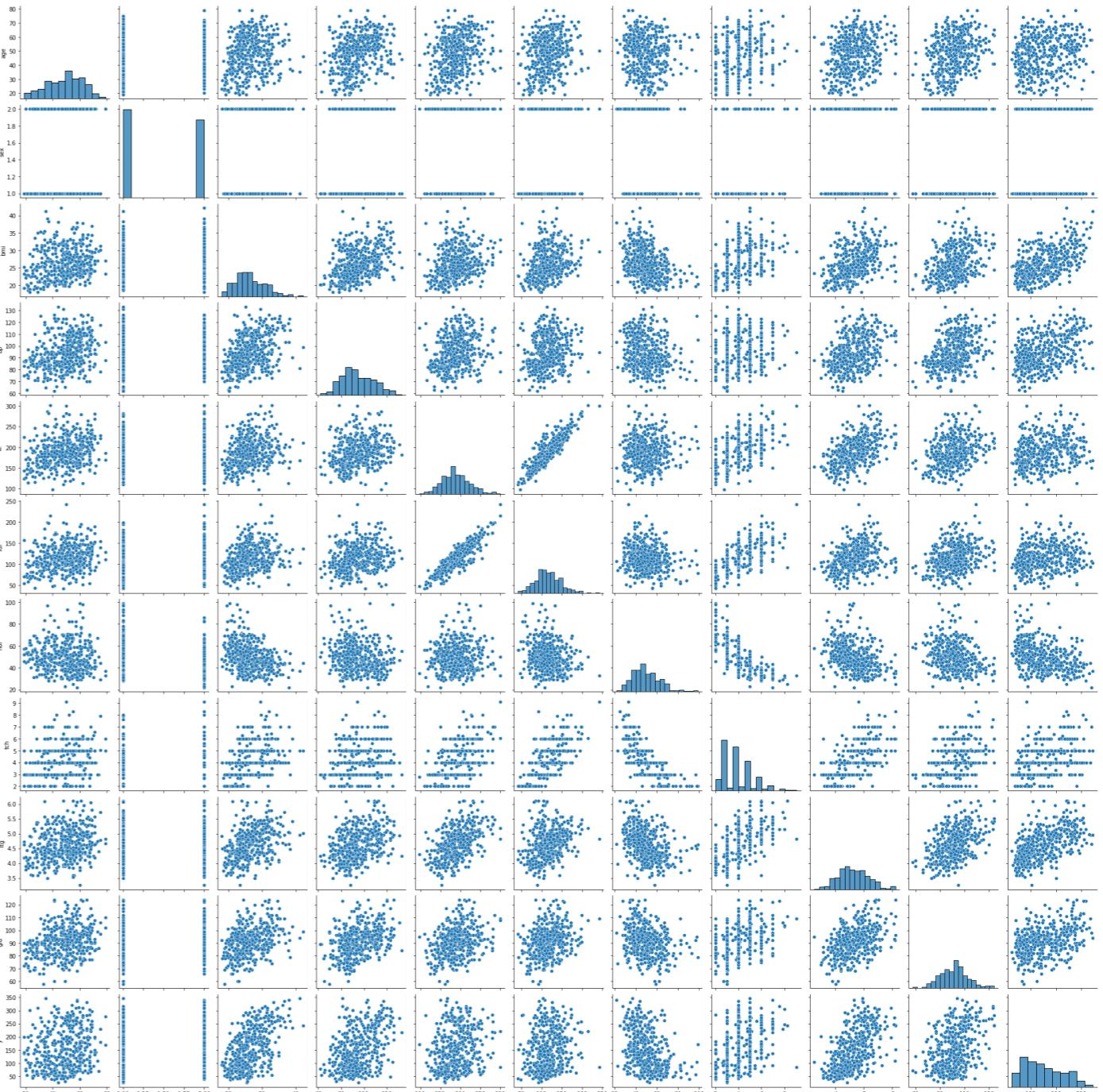
<sup>1</sup> <https://www4.stat.ncsu.edu/~boos/var.select/diabetes.html> <sup>2</sup> <http://statweb.lsu.edu/faculty/li/IIT/diabetes.txt>

# Why do we explore data before simulation?

- Visually inspect the distribution of variables
  - Intuition for probability distribution
- Check and measure the correlation between predictor variables
  - Rationales for modeling covariance structure
- Check and measure the correlation between predictor variables and the response
  - Initial understanding of relationship between predictors and response

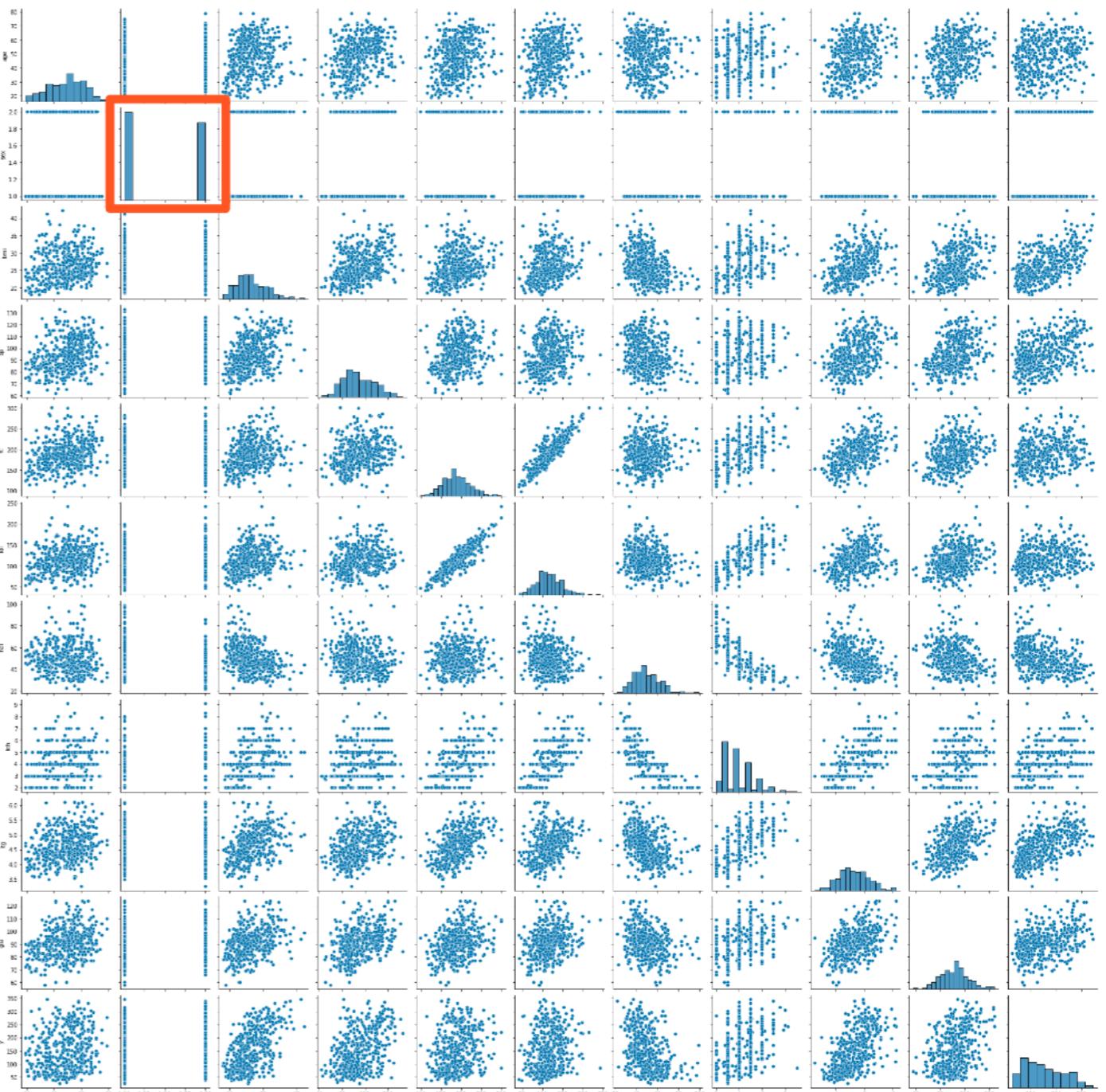
# Pairplot of the dataset

```
sns.pairplot(dia)
```



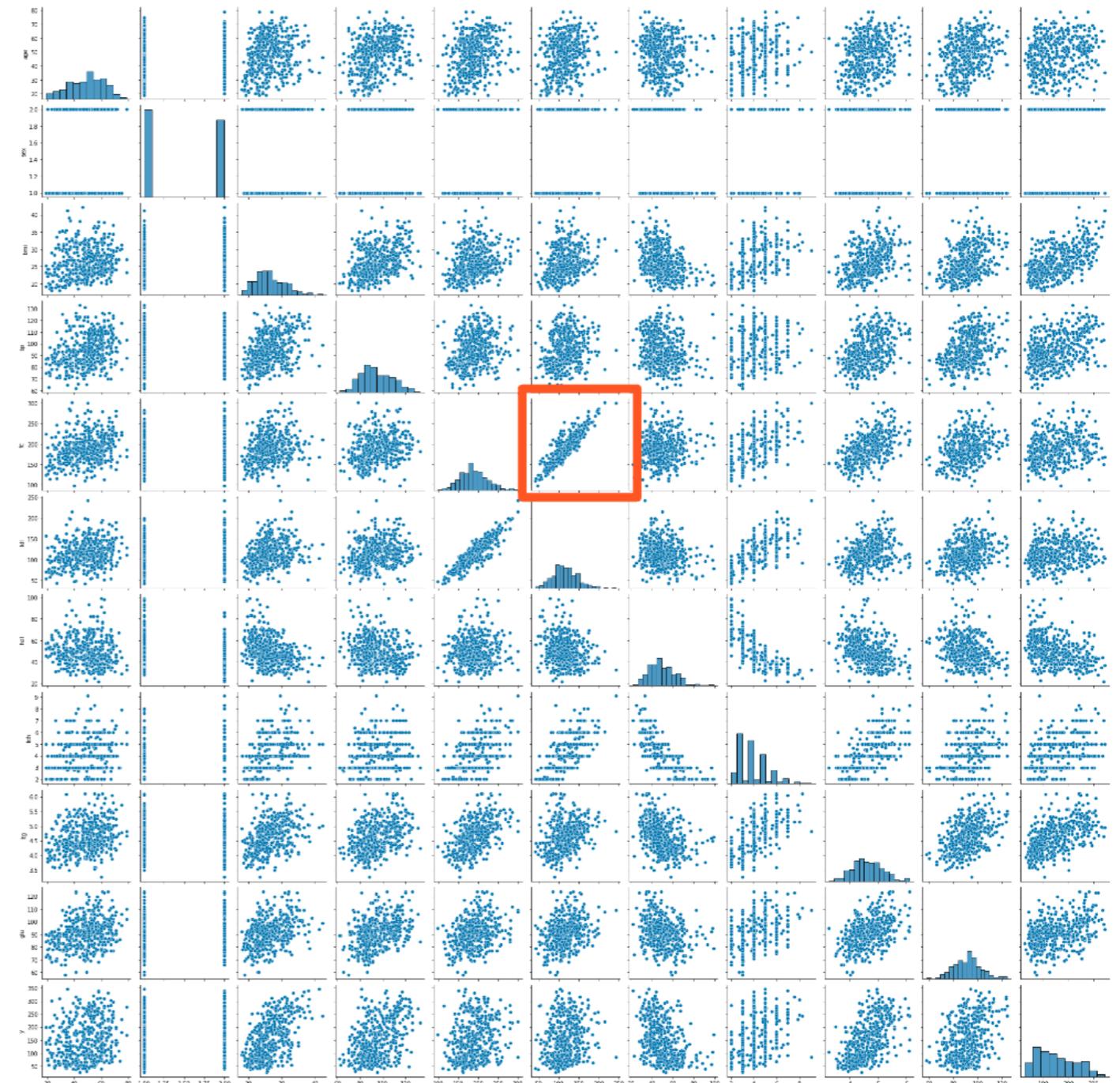
# Pairplot of the dataset

```
sns.pairplot(dia)
```



# Pairplot of the dataset

```
sns.pairplot(dia)
```



# Correlations between different variables

```
dia.corr()
```

	age	sex	bmi	bp	tc	ldl	hdl	tch	ltg	glu	y
age	1.000000	0.173737	0.185085	0.335428	0.260061	0.219243	-0.075181	0.203841	0.270774	0.301731	0.187889
sex	0.173737	1.000000	0.088161	0.241010	0.035277	0.142637	-0.379090	0.332115	0.149916	0.208133	0.043062
bmi	0.185085	0.088161	1.000000	0.395411	0.249777	0.261170	-0.366811	0.413807	0.446157	0.388680	0.586450
bp	0.335428	0.241010	0.395411	1.000000	0.242464	0.185548	-0.178762	0.257650	0.393480	0.390430	0.441482
tc	0.260061	0.035277	0.249777	0.242464	1.000000	0.896663	0.051519	0.542207	0.515503	0.325717	0.212022
ldl	0.219243	0.142637	0.261170	0.185548	0.896663	1.000000	-0.196455	0.659817	0.318357	0.290600	0.174054
hdl	-0.075181	-0.379090	-0.366811	-0.178762	0.051519	-0.196455	1.000000	-0.738493	-0.398577	-0.273697	-0.394789
tch	0.203841	0.332115	0.413807	0.257650	0.542207	0.659817	-0.738493	1.000000	0.617859	0.417212	0.430453
ltg	0.270774	0.149916	0.446157	0.393480	0.515503	0.318357	-0.398577	0.617859	1.000000	0.464669	0.565883
glu	0.301731	0.208133	0.388680	0.390430	0.325717	0.290600	-0.273697	0.417212	0.464669	1.000000	0.382483
y	0.187889	0.043062	0.586450	0.441482	0.212022	0.174054	-0.394789	0.430453	0.565883	0.382483	1.000000

# **Let's practice!**

**MONTE CARLO SIMULATIONS IN PYTHON**

# Choosing probability distributions

MONTE CARLO SIMULATIONS IN PYTHON



Izzy Weber

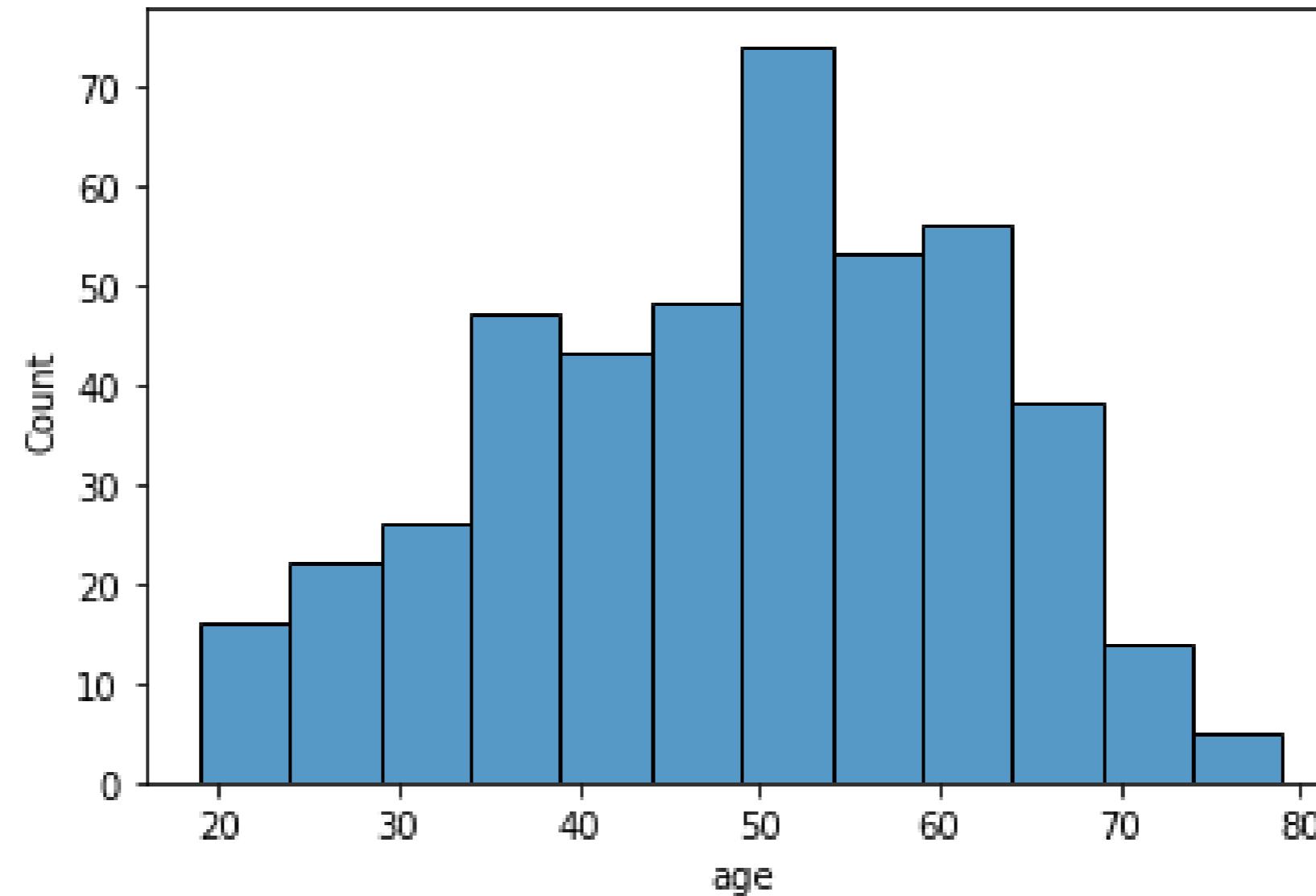
Curriculum Manager, DataCamp

# Maximum Likelihood Estimation (MLE)

- Used to select a probability distribution by measuring fit
  - Distribution yielding highest likelihood given the data is considered optimal
- SciPy's `.nnlf()` used to calculate the *negative* likelihood function
- The lower the MLE value calculated using `.nnlf()`, the better the fit

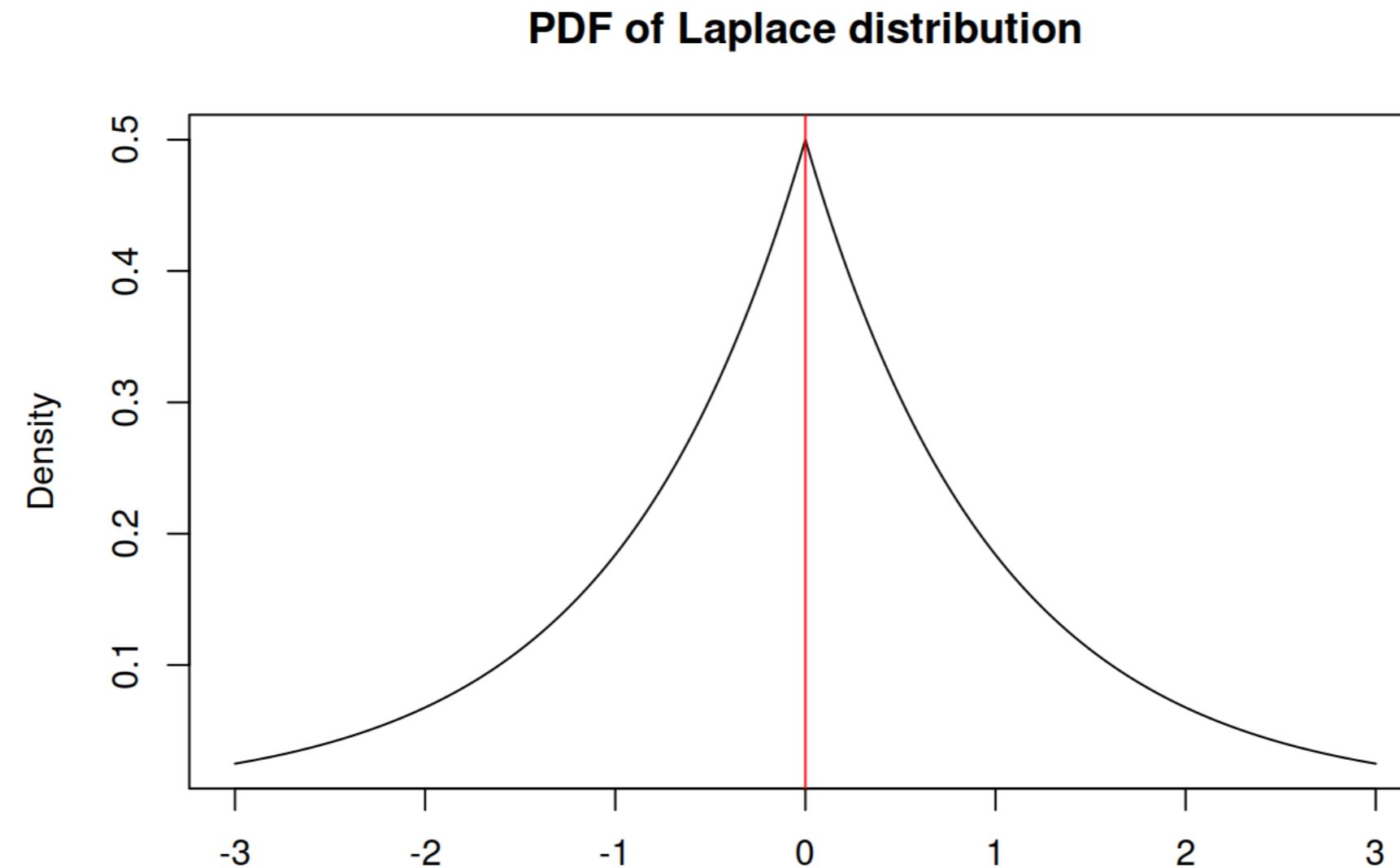
# Picking a distribution for the age variable

```
sns.histplot(dia["age"])
```



# Candidate distributions

```
distributions = [st.laplace, st.norm, st.expon]
```



# Choosing between candidate distributions

```
mles = []

for distribution in distributions:
    pars = distribution.fit(dia["age"])
    mle = distribution.nlf(pars, dia["age"])
    mles.append(mle)

print(mles)
```

```
[1797.8467779878652, 1764.0693689033028, 1938.171599681118]
```

# Choosing between candidate distributions

```
for var in ["age", "bmi", "bp", "tc", "ldl", "hdl", "tch", "ltg", "glu"]:  
    distributions = [st.laplace, st.norm, st.expon]  
    mles = []  
  
    for distribution in distributions:  
        pars = distribution.fit(dia[var])  
        mle = distribution.nnlf(pars, dia[var])  
        mles.append(mle)  
  
    best_fit = sorted(zip(distributions, mles), key=lambda d: d[1])[0]  
    print(f"Best fit reached using {best_fit[0].name}, \  
          MLE value: {best_fit[1]}, for variable {var}")
```

# Results of the evaluation

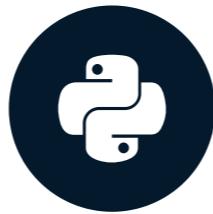
```
Best fit reached using norm, MLE value: 1764.0693689033028, for variable age
Best fit reached using norm, MLE value: 1283.356127017369, for variable bmi
Best fit reached using norm, MLE value: 1787.7746251622739, for variable bp
Best fit reached using norm, MLE value: 2193.1564373753627, for variable to
Best fit reached using norm, MLE value: 2136.0440476305284, for variable ldl
Best fit reached using norm, MLE value: 1758.1350738323013, for variable hdl
Best fit reached using norm, MLE value: 739.3762494786798, for variable tch
Best fit reached using norm, MLE value: 339.6620870566908, for variable ltg
Best fit reached using norm, MLE value: 1706.0467588930867, for variable glu
```

# **Let's practice!**

**MONTE CARLO SIMULATIONS IN PYTHON**

# Inputs with correlations

MONTE CARLO SIMULATIONS IN PYTHON



Izzy Weber

Curriculum Manager, DataCamp

# Parameters for a Multivariate normal simulation

Parameters for multivariate normal simulation:

1. Mean of each variable
2. Covariance matrix

# Parameters for a Multivariate normal simulation

```
mean_dia = dia[["age", "bmi", "bp", "tc", "ldl", "hdl", "tch", "ltg", "glu"]].mean()
```

```
age    48.518100  
bmi   26.375792  
bp    94.647014  
tc    189.140271  
ldl   115.439140  
hdl   49.788462  
tch   4.070249  
ltg   4.641411  
glu   91.260181
```

# Parameters for a Multivariate normal simulation

```
cov_dia = dia[["age", "bmi", "bp", "tc", "ldl", "hdl", "tch", "ltg", "glu"]].cov()
```

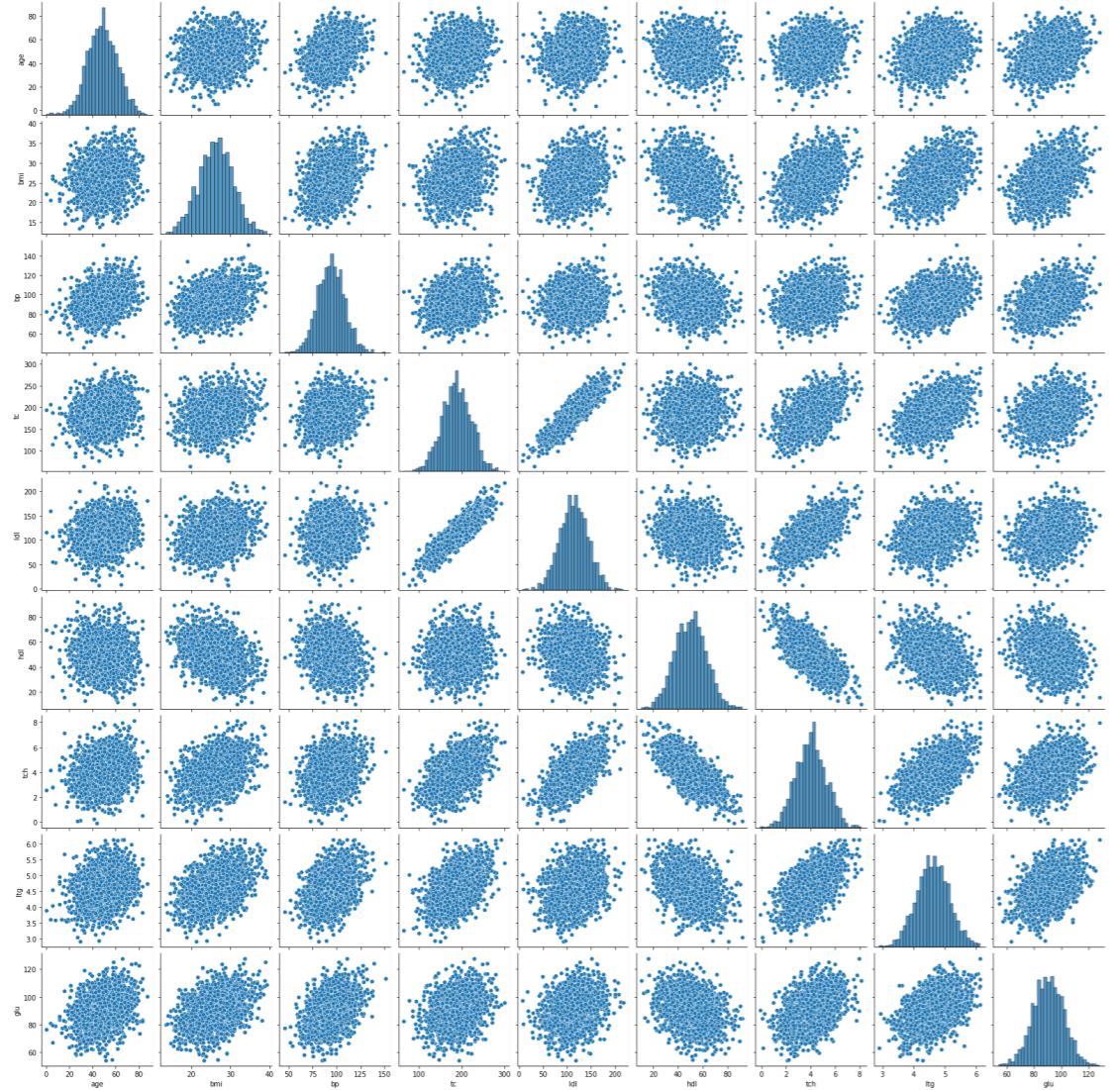
	age	bmi	bp	tc	ldl	hdl	tch	ltg	glu
age	171.846610	10.719600	60.817945	117.983850	87.409154	-12.747296	3.448283	1.854271	45.472604
bmi	10.719600	19.519798	24.162884	38.191612	35.093059	-20.961368	2.359262	1.029723	19.741914
bp	60.817945	24.162884	191.304401	116.061168	78.051321	-31.979851	4.598687	2.843024	62.081913
tc	117.983850	38.191612	116.061168	1197.717241	943.771368	23.061486	24.214954	9.319736	129.591539
ldl	87.409154	35.093059	78.051321	943.771368	924.955494	-77.279343	25.895541	5.057894	101.605213
hdl	-12.747296	-20.961368	-31.979851	23.061486	-77.279343	167.293585	-12.326138	-2.693069	-40.697671
tch	3.448283	2.359262	4.598687	24.214954	25.895541	-12.326138	1.665261	0.416510	6.189527
ltg	1.854271	1.029723	2.843024	9.319736	5.057894	-2.693069	0.416510	0.272892	2.790604
glu	45.472604	19.741914	62.081913	129.591539	101.605213	-40.697671	6.189527	2.790604	132.165712

# Code for simulation

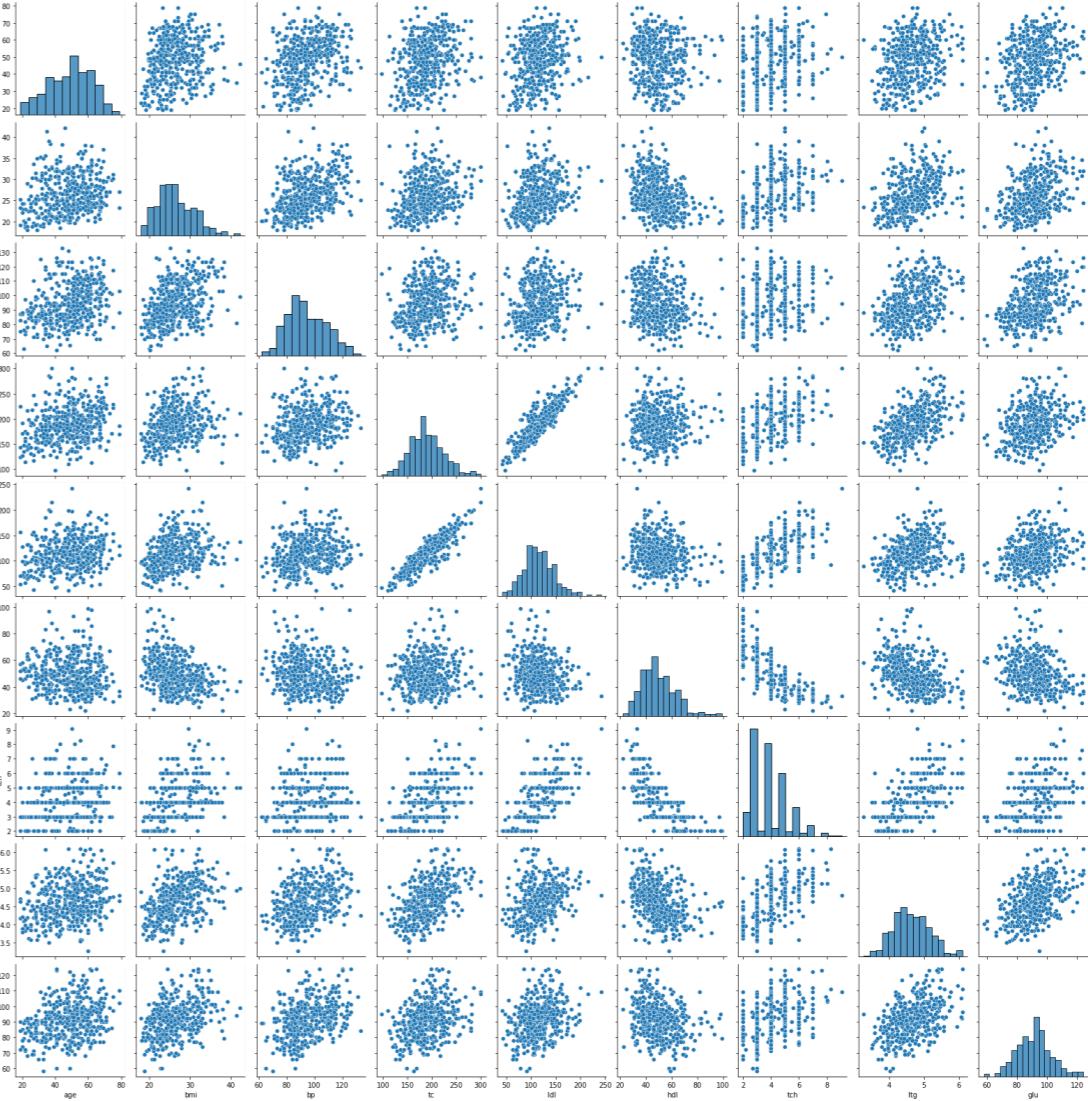
```
simulation_results = st.multivariate_normal.rvs(mean=mean_dia,  
                                                size=2000, cov=cov_dia)  
  
df_results = pd.DataFrame(simulation_results, columns=["age", "bmi", "bp", "tc", "ldl",  
                                                       "hdl", "tch", "ltg", "glu"])
```

# Pairplot of simulation results

```
# Simulated results  
sns.pairplot(df_results)
```

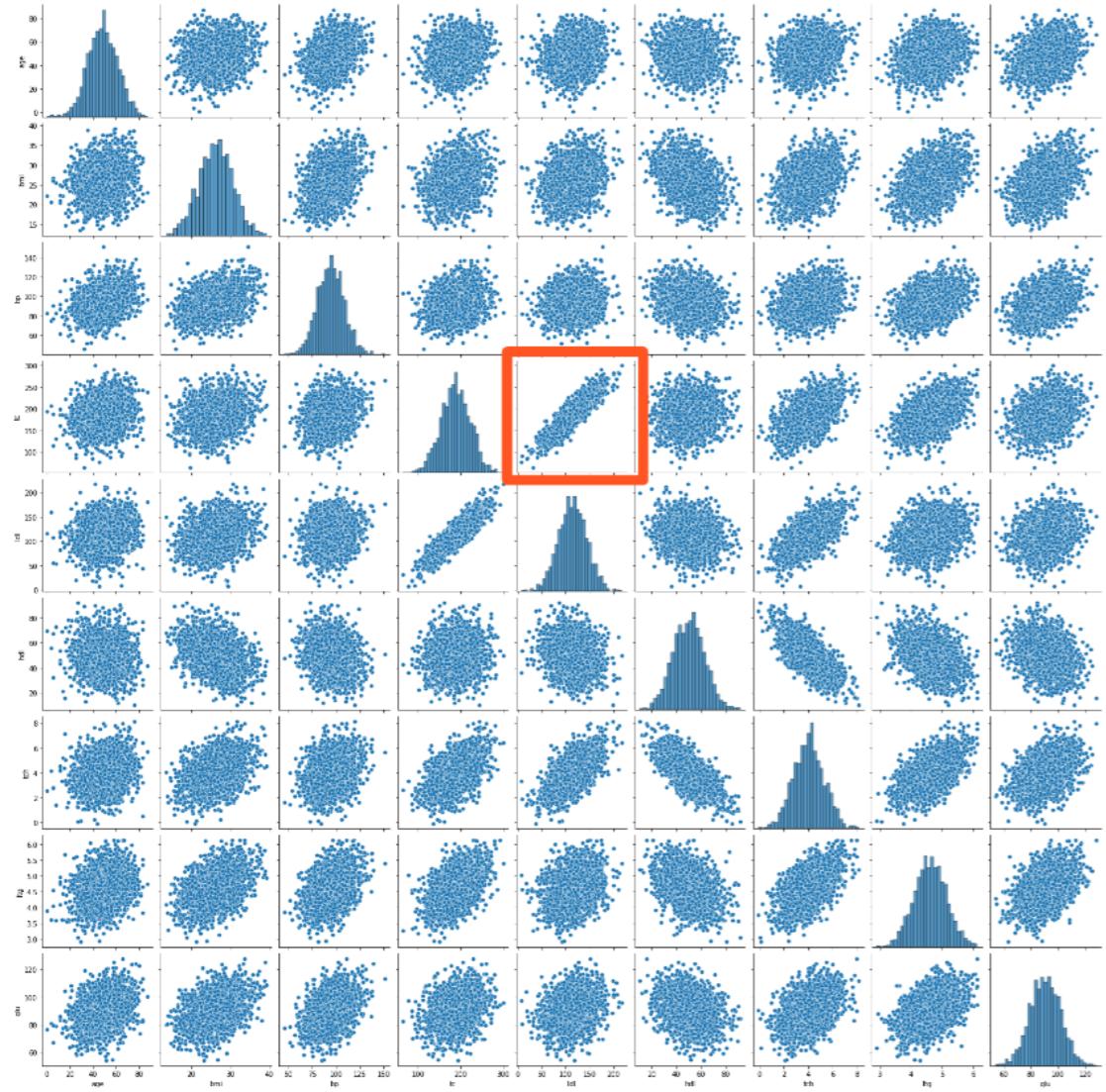


```
sns.pairplot(dia[["age", "bmi", "bp", "tc", "ldl",  
"hdl", "tch", "ltg", "glu"]])
```

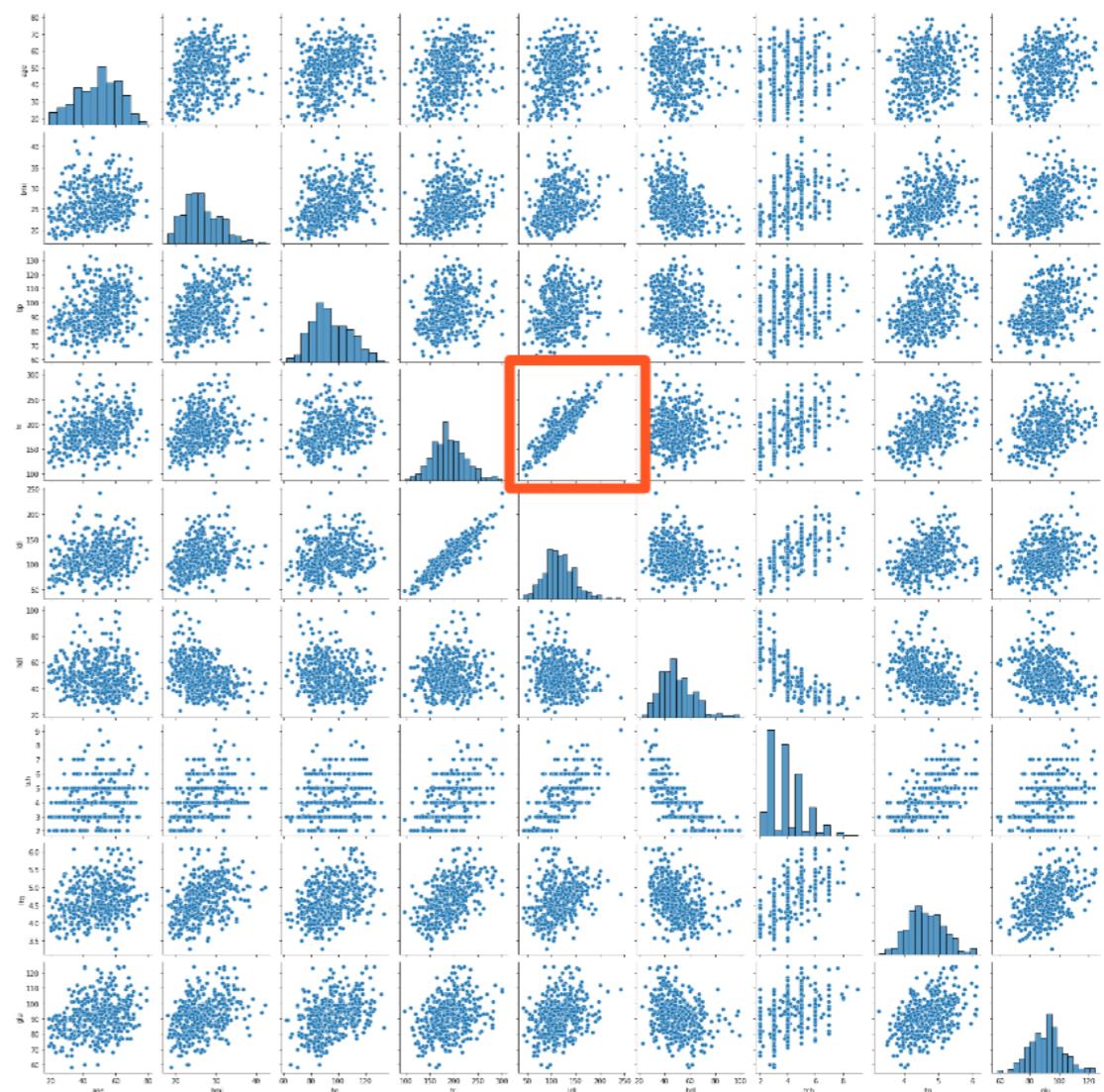


# Pairplot of simulation results

```
# Simulated results  
sns.pairplot(df_results)
```



```
sns.pairplot(dia[["age", "bmi", "bp", "tc", "ldl",  
"hdl", "tch", "ltg", "glu"]])
```



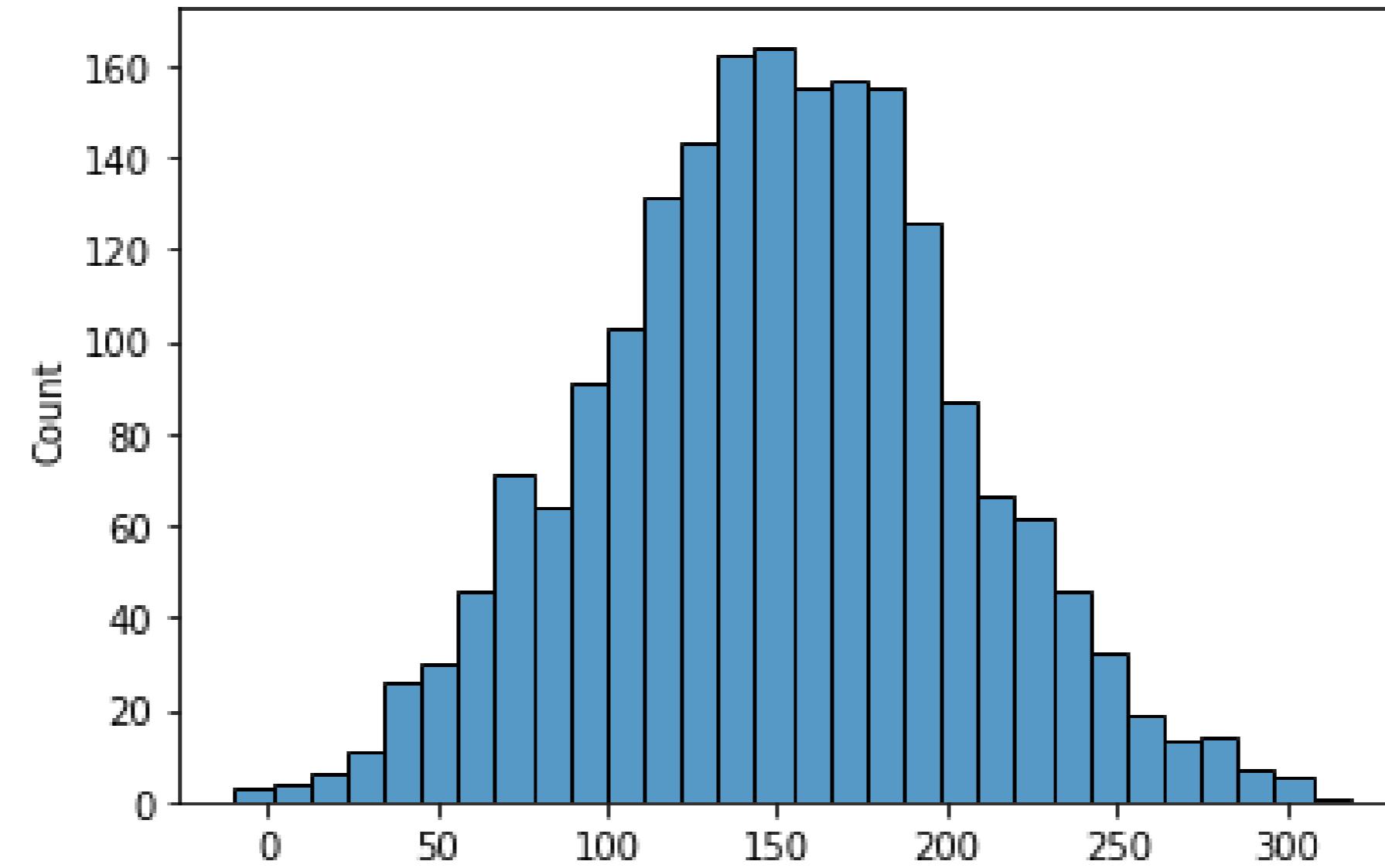
# Calculate the predicted y

```
predicted_y = regr_model.predict(df_results)  
predicted_y[0:5]
```

```
array([209.29768784, 177.08133903, 123.19242521, 84.28490832, 244.90014108])
```

# Histogram of the predicted y

```
sns.histplot(predicted_y)
```



# Simulated predictors + predicted response

```
df_results["predicted_y"] = predicted_y  
df_results.head()
```

	age	bmi	bp	tc	ldl	hdl	tch	ltg	glu	predicted_y
0	54.491842	32.512362	82.131464	203.075420	114.043050	44.820017	5.137683	5.254633	100.815909	209.297688
1	66.380490	29.380708	98.810054	136.474760	68.457982	51.691298	3.455412	4.572478	96.117969	177.081339
2	59.003285	27.015225	92.195168	242.796424	126.541644	86.050629	2.423928	4.640063	87.485747	123.192425
3	34.803821	20.961365	86.852597	168.762268	110.113823	53.158621	3.925988	4.080205	79.187999	84.284908
4	56.732615	32.682115	118.384860	226.152964	136.838283	46.467736	4.376397	5.374001	104.184429	244.900141

# Recap

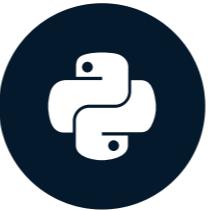
1. Define the input variables and pick probability distributions for them
  - All independent variables except `sex`
  - Multivariate normal distributions based on MLE calculation
2. Generate inputs by sampling from these distributions
  - `st.multivariate_normal.rvs(mean, size, cov)`
3. Perform a deterministic calculation of the simulated inputs
  - `regr_model`
4. Summarize results to answer questions of interest
  - In the next lesson!

# **Let's practice!**

**MONTE CARLO SIMULATIONS IN PYTHON**

# Summary statistics

MONTE CARLO SIMULATIONS IN PYTHON

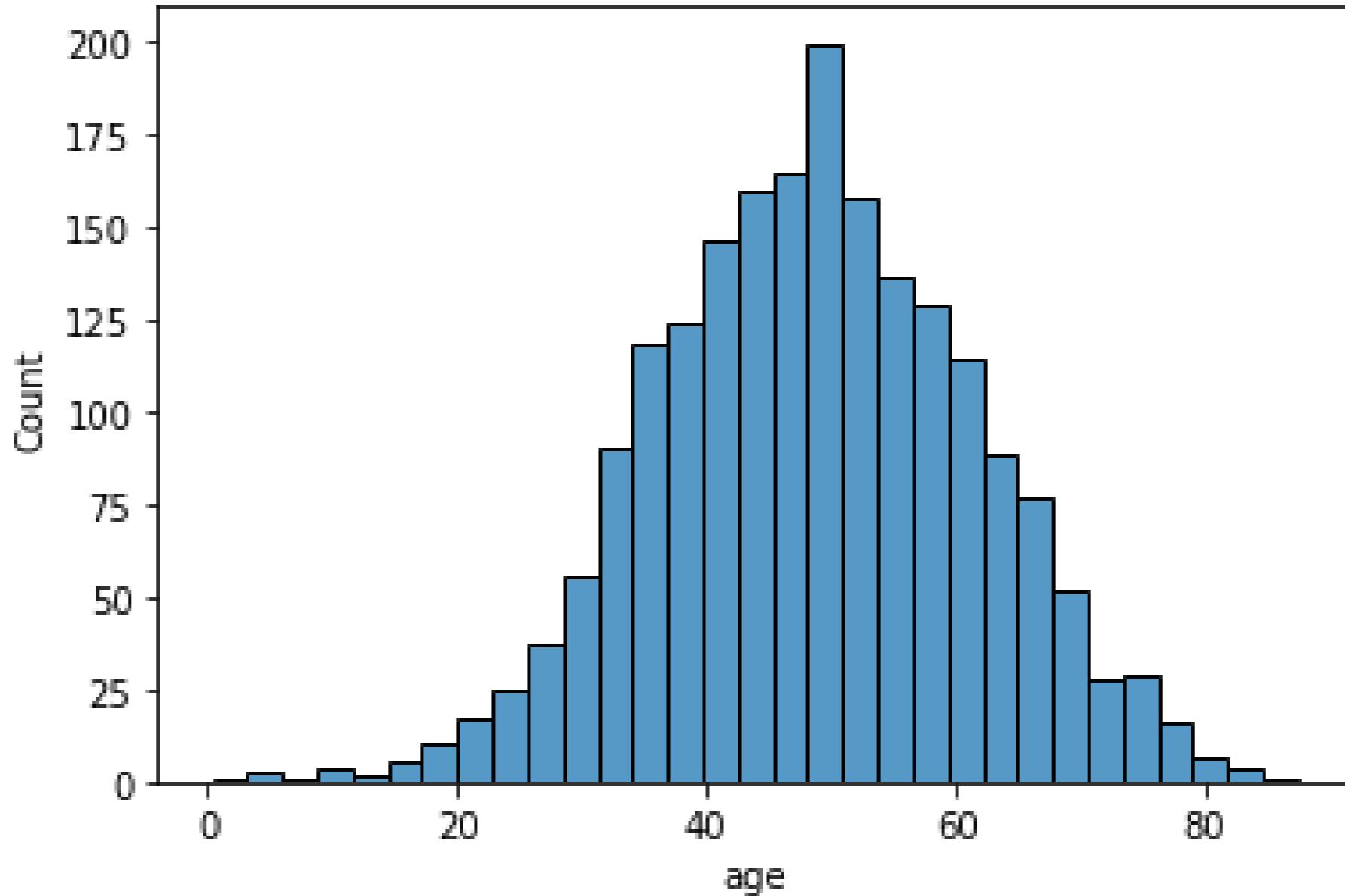


Izzy Weber

Curriculum Manager, DataCamp

# Age outcomes

What is the difference in outcomes for people in the first and fourth age quantiles?



# Age outcomes

```
print(np.quantile(df_summary["age"], 0.25))  
print(np.quantile(df_summary["age"], 0.75))
```

```
39.78584463094688  
57.52092642086441
```

# Age outcomes

```
age_q25 = np.quantile(df_summary["age"], 0.25)
age_q75 = np.quantile(df_summary["age"], 0.75)

mean_age_q75_outcome = np.mean(df_summary[df_summary["age"] > age_q75]["predicted_y"])
mean_age_q25_outcome = np.mean(df_summary[df_summary["age"] < age_q25]["predicted_y"])

mean_age_q75_outcome - mean_age_q25_outcome
```

```
34.09429663553621
```

# Outcome differences based on age and bmi

- What is the difference in outcomes for people in the first and fourth quantiles of bmi *and* age?
- What is the 95 percent confidence interval and standard deviation for this difference?

# Outcome differences based on age and bmi

```
y_diffs = []
for i in range(1000):
    simulation_results = st.multivariate_normal.rvs(mean=mean_dia, size=1000, cov=cov_dia)
    df_results = pd.DataFrame(simulation_results, columns=["age", "bmi", "bp", "tc", "ldl",
                                                            "hdl", "tch", "ltg", "glu"])
    predicted_y = regr_model.predict(df_results)
    df_y = pd.DataFrame(predicted_y, columns=["predicted_y"])
    df_sum = pd.concat([df_results, df_y], axis=1)
    age_q25 = np.quantile(df_sum["age"], 0.25)
    age_q75 = np.quantile(df_sum["age"], 0.75)
    bmi_q25 = np.quantile(df_sum["bmi"], 0.25)
    bmi_q75 = np.quantile(df_sum["bmi"], 0.75)
    q75_outcome = np.mean(df_sum[(df_sum["bmi"] > bmi_q75) & (df_sum["age"] > age_q75)]["predicted_y"])
    q25_outcome = np.mean(df_sum[(df_sum["bmi"] < bmi_q25) & (df_sum["age"] < age_q25)]["predicted_y"])
    y_diff = q75_outcome - q25_outcome
    y_diffs.append(y_diff)
```

# Outcome differences based on age and bmi

- Mean:

```
np.mean(y_diffs)
```

```
132.4948511247819
```

- 95% confidence interval:

```
np.quantile(y_diffs, 0.025)
```

```
120.73340800299707
```

```
np.quantile(y_diffs, 0.975)
```

```
144.1344994507557
```

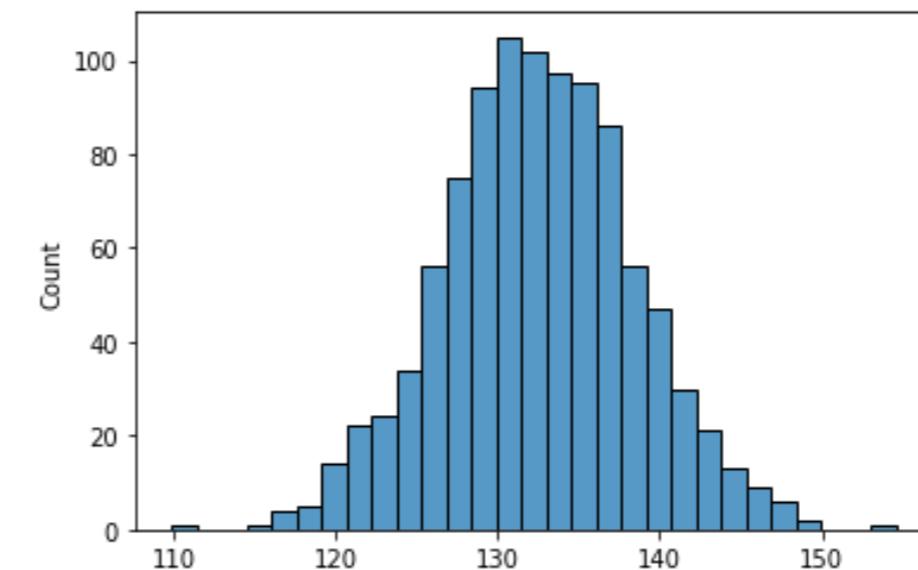
- Standard deviation:

```
np.std(y_diffs)
```

```
5.9322225537128
```

- Histogram of the results:

```
sns.histplot(y_diffs)
```



# **Let's practice!**

**MONTE CARLO SIMULATIONS IN PYTHON**