תרגיל 3

מועד הגשה: 27.01.2023. הגשה במודל. עבודה עצמאית בלבד.

יש להתמקד בתשובות עם הסברים, טבלאות וגרפים ברורים וקריאים (כותרות, צירים והסברים). את הקוד יש לצרף כנספח בסוף העבודה. נא להגיש כקובץ PDF יחיד.

בדיקות PCR לקורונה נחשבות מדויקות למדי. בכל זאת, לפעמים קוראות שגיאות (גם בגלל ביצוע לא נכון של הבדיקה וגם בגלל שהיא לפעמים שגויה).

נניח שהתוצאות של בדיקות שונות הן בלתי תלויות.

את הסיכוי שבבדיקה של חולה נקבל תשובה חיובית (TP = true positive) נסמן ב p.

את הסיכוי שבבדיקה של בריא נקבל תשובה חיובית (FP = false positive) נסמן ב q.

את אחוז החולים באוכלוסייה נסמן ב s.

המטרה היא להעריך את s, p ו q מתוך דאטה, בשיטה בייסיאנית כמובן.

1. הגרילו דאטה: 1000 אנשים באו להיבדק. כל אחד חולה בסיכוי s=20%, p=95%, q=10%.

יש שלושה פרמטרים: s, p ו q. כלומר, מרחב הפרמטרים הוא תלת ממדי.

2. עבור ה prior, נניח שההתפלגויות של הפרמטרים הן בלתי תלויות.

   אין ידע מוקדם על s, ולכן ניקח $s \sim U(0,1)$.

   אנחנו מעריכים שהבדיקה די מדויקת ולכן ל p ו q ניקח prior מוטה:

$$f_q(q) = \begin{cases} 2(1-q) & 0 \le q \le 1 \\ 0 & \text{otw} \end{cases}, \quad f_p(p) = \begin{cases} 2p & 0 \le p \le 1 \\ 0 & \text{otw} \end{cases}$$

3. בעזרת שיטת MCMC, צרו 10,000 דגימות בלתי תלויות (בקרוב) מה posterior.

4. שרטטו את ההתפלגויות השוליות החד והדו ממדיות של ה posterior.

5. מצאו את המאקסימום (בשלושה ממדים), ממוצע וחציון.

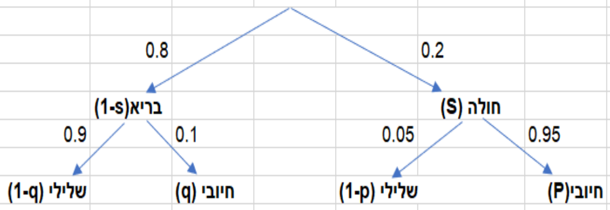
6. לכל פרמטר, מצאו את התחום הקטן ביותר בו הפרמטר נמצא בסיכוי 90%.

בפתרון התרגיל יש לפרט את הנוסחאות לחישובים השונים (למשל, של המונה ב posterior) ולהסביר (במילים ונוסחאות) את האלגוריתם (למשל, באיזו שרשרת עזר השתמשתם, מה הנוסחה לקבלת צעד וכדומה).

תרגיל בית 3

סעיף א:

S	20%
P	95%
Q	10%



סיכוי לקבלת תוצאה חיובית	a=s*p+(1-s)*(q)	0.27
סיכוי לקבלת תוצאה שלילית	b=s*(1-p)+(1-s)*(1-q)	0.73

התבקשנו להגריל data בהתחשב בסיכויים הנ"ל כך שהנתונים יהיו למעשה תוצאות בדיקות קורנה.

ניתן להתייחס לכל בדיקה כאל ניסוי ברנולי עם סיכוי 0.27 להצלחה (כאשר מגדירים תוצאה חיובית כהצלחה)

הגרלתי 1000 תצפיות מהתפלגות ברנולית עם הפונקציה binomial של numpy (כאשר מגדירים את n להיות שווה ל1 מקבלים הגרלה מהתפלגות ברנולית). כל תצפית מקבלת 0 או 1.

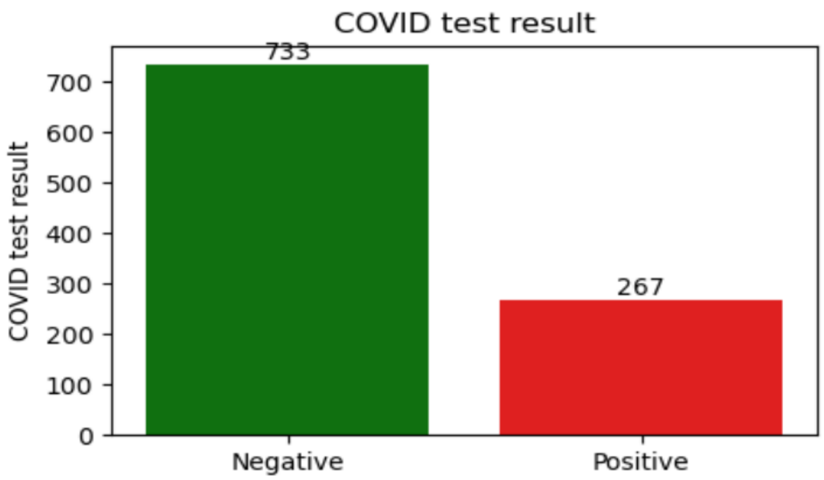
```
n = 1000
p = 0.27
data = pd.DataFrame({'COVID test result':np.random.binomial(n=1, p=p, size=n)})
```

המדגם שהוגרל:

```
data.T
```

	0	1	2	3	4	5	6	7	8	9	...	990	991	992	993	994	995	996	997	998	999
COVID test result	0	1	0	0	0	0	0	1	0	1	...	0	1	0	0	0	1	1	0	0	0

1 rows × 1000 columns



סעיף ב:

בסעיף הזה נשתמש במשפט בייס כדי לחשב: Posterior, likelihood, Prior ולהתייחס ל evidence.

Data -

מאחר ותוצאות הבדיקות הן ב״ת ובסעיף 1 הוצאנו מהתפלגות ברנולית, ניתן להתייחס למס׳ התוצאות החיוביות כעל מס׳ בינומי כאשר גודל המדגם הוא 1000 והסיכוי להצלחה הוא הסיכוי לקבלת תוצאת בדיקה חיובית. נסמן k מס׳ התוצאות החיוביות:

$$k \sim Bin(n,t)$$

$$t = s \cdot p + (1-s)\cdot q = s(p-q)+q$$

t סיכוי לקבלת תוצאה חיובית

ה data הוא למעשה k (כי כך הוגדר). $n=1000$ ו k משתנה מהרצה להרצה (כמשתנה מקרי).

מדגם חדש (למשל) בסעיף 1 הוא 267 אך מכיוון שהקוד ירוץ שוב ולכן הקוד הוא נכתב בצורה כללית בהתאם לתוצאות המדגם.

$$\underbrace{P(s,p,q \mid Data)}_{Posterior} = \frac{\overbrace{P(Data \mid s,p,q)}^{likelihood}\cdot \overbrace{P(s,p,q)}^{Prior}}{\underbrace{P(Data)}_{evidence}}$$

Prior

נתון: (1) $s \sim U(0,1)$, לכן:

$$f_s(s) = \begin{cases} 1, & 0 \le s \le 1 \\ 0, & else \end{cases}$$

(2)
$$f_p(p) = \begin{cases} 2p, & 0 \le p \le 1 \\ 0, & else \end{cases}$$

(3)
$$f_q(q) = \begin{cases} 2(1-q), & 0 \le q \le 1 \\ 0, & else \end{cases}$$

בנוסף נתון שעבור ה Prior, ההתפלגויות של הפרמטרים הן ב״ת.

לכן פונק׳ הצפיפות $f(s,p,q)$ שתבטא את ה Prior תהיה שווה למכפלת הצפיפויות הנ״ל:

$$Prior = f(s,p,q) = \begin{cases} 4p(1-q), & 0 \le s,p,q \le 1 \\ 0, & else \end{cases}$$

likelihood

ה-likelihood כאן הוא הסבירות הגבוהה ביותר לקבל את ה-data (במקרה שלנו k)

בהינתן s,p,q. לכן נשתמש בפונק' צפיפות בינומית:

$$\text{likelihood} = f(Data|s,p,q) = \binom{n}{k}\left(s(p-q)+q\right)^{k}\cdot\left(1-(s(p-q)+q)\right)^{n-k}$$

סיכוי לקבלת תוצאה חיובית ↑   סיכוי לקבלת תוצאה שלילית ↑

Posterior

נעזר במשפט בייס לחישוב ה Posterior:

$$P(s,p,q|Data) = \frac{\binom{n}{k}(s(p-q)+q)^{k}\cdot(1-s(p-q)-q)^{n-k}\cdot 4p(1-q)}{P(Data)}$$

$$P(Data) = \int P(data|p)P(p)\,dp$$

במקרה שלנו מרחב הפרמטרים הוא תלת מימדי, לכן:

evidence

$$P(data) = \int_{s=0}^{1}\int_{p=0}^{1}\int_{q=0}^{1} P(Data|s,p,q)\cdot P(s,p,q)\,ds\,dp\,dq$$

מאחר ואין צורך לנרמל את ה-Posterior אפשר להתעלם מה-evidence ולהישאר רק עם המונה. לכן בסעיפים הבאים ה Posterior יחושב רק כמכפלה של ה likelihood ב-Prior.

*MCMC הוא אלגוריתם שמאפשר לדגום את ה-Posterior ע"י דגימה של דגימות ממנו ללא התייחסות ל-evidence שלעיתים קרובות קשה לחישוב ואף לא אפשרי.

סעיף 3 - MCMC

נדגום בעזרת MCMC 10,000 דגימות בת"ל (בקירוב) מה Poterior.

האלגוריתם שאפשר ליצור ~~?~~ דגימות (תוך התעלמות מה-evidence) ע"י הצעה של ערכים חדשים לפרמטרים ודחיה או ~~?~~ קבלה שלהם לפי תנאי.

נניח שהשרשרת עבור Q של $\theta=[s,p,q]$ שההתפלגות ~~?~~ הסטציונרית שלה אחידה.

בכל איטרציה נחשב ערכים חדשים לפרמטרים לפי Q ונחליט האם לקבל או לדחות אותם כך:

נסמן θ הערכים הישנים או ההתחלתיים ; θ' ערכים חדשים מוצעים.

Acceptance Ratio

$$a(\theta'|\theta)=\min\left\{1,\frac{Q(\theta|\theta')\,\pi(\theta')}{Q(\theta'|\theta)\,\pi(\theta)}\right\}$$

π הוא ההתפלגות הסטציונרית של s,p,q - פונק' ה Posterior שהוגדרה בסעיף הקודם.

מאחר והשרשרת הנוצרת עבור Q בעלת התפלגות סטציונרית אחידה, סיכויי המעבר שלה ממצב למצב זהים: $Q(\theta|\theta')=Q(\theta'|\theta)$ ולכן מצטמצמים.

המימוש של האלגוריתם בפייתון:

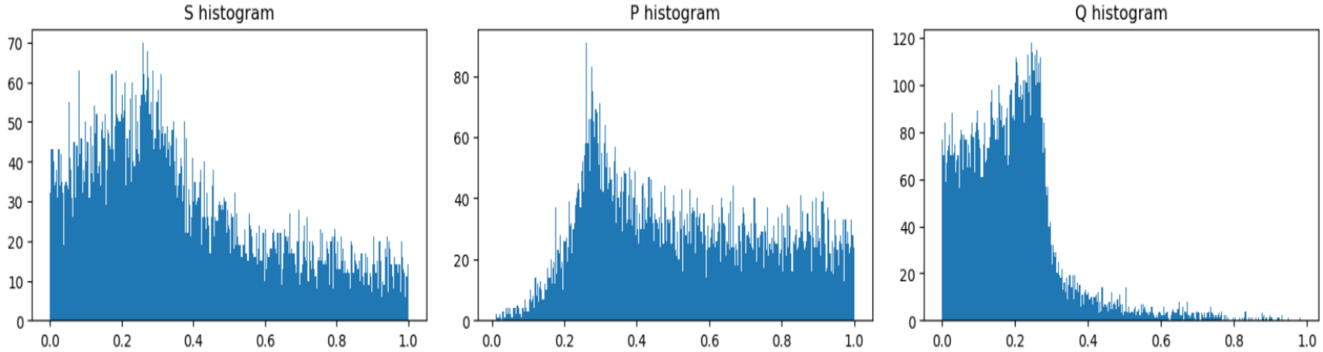
1. נגדיר ערכים התחלתיים ל s,p,q ע"י דגימה מהתפלגות אחידה בין 0 ל 1. נסמן אותם $\theta=[s,p,q]$
2. נגזור ערכים חדשים באותו אופן ונסמן θ'
3. נחשב את סיכוי הקבלה: $$R=\min\left\{1,\frac{Posterior(\theta')}{Posterior(\theta)}\right\}$$
4. נגריל ערך מהתפלגות אחידה: $u\sim U(0,1)$
5. אם $u<R$ ~~?~~ נעדכן את $\theta=\theta'$ ואם לא נישאר עם הערכים ההתחלתיים.
6. נחזור על ~~?~~ סעיפים 2-5 עד שנקבל 10,000 דגימות.

* כדי לנסות ליצור מדגם בת"ל, האלגוריתם יקח כל 100 דגימות. המונח הזה נקרא "Lag". דגימה חדשה תלויה לעיתים קרובות בקודמת מכיוון שעדיין הם אולי לא מקבלים ערך חדש ונשארים עם הישן. לכן לא ניקח כל דגימה אלא כל 100 דגימות. ניסיתי להגדיל את ה Lag כדי לוודא אי תלות אבל זה היה איטי מידי לחישוב. (לכן לקחתי רק כל 100 ולא יותר).

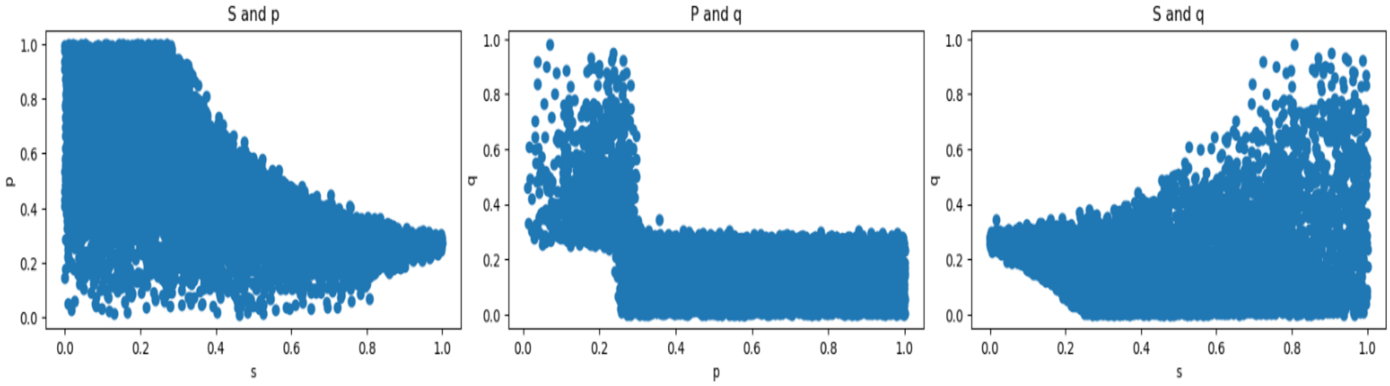
לבסוף נקבל מטריצה בגודל 3X10,000 שתכיל את הדגימות.

סעיף 4:

שרטטתי את ההתפלגויות החד מימדיות עבור כל פרמטר בנפרד ואת ההתפלגות הדו מימדיות עבור כל זוג פרמטרים: s-q,p-q,s-p:



ההתפלגויות הדו מימדיות:



סעיף 5:

מצאתי את הפרמטרים שממקסמים את ה Posterior ע"י הצבה של כל שורה במטריצת הדגימות(כל שלשת פרמטרים) ב posterior ובחירת השלשה שממקסמת אותו.

```
The max value (for 3D) is:0.11313388635762155
The parameters that maximazed the posteriro is:
S:0.27102065748205284
P:0.9975391775213402
Q:0.0006281668743051139
```

ממוצע וחציון:

```
The evarage is:
S:0.3784403637120403
P:0.5285348607612216
Q:0.1918006110098227

The median is:
S:0.31538697459813914
P:0.49515130416565406
Q:0.18330782731111994
```

סעיף 6:

ישנם 10 אופציות לטווח המכיל 90% מהדאטה. בין האחוזון ה0 ל90, בין ה1 ל91, בין ה2 ל92 וכן הלאה. בניתי לולאה שתעבור על כל טווח אפשרי בכל פרמטר ותחשב את הגודל שלו ותשמור את הערכים שנותנים את הטווח הקטן ביותר. התוצאות:

```
The smallest range in which the parameter S lies with a 90% chance: [5.784071560832604e-05, 0.7934371086679619]
The smallest range in which the parameter p lies with a 90% chance:[0.20357451207436889, 0.9559014122809707]
The smallest range in which the parameter q lies with a 90% chance:[8.010756206378034e-06, 0.3231264161797986]
```

In [1]:
```
import pandas as pd
import numpy as np
import statsmodels as sm
import sklearn as skl
import seaborn as sns
import scipy as sc
import matplotlib.pyplot as plt
```

1.

In [2]:
```
data = pd.DataFrame({'COVID test result':np.random.binomial(n=1, p=0.27, size=1000)})
```

In [3]:
```
data.T
```

Out[3]:

	0	1	2	3	4	5	6	7	8	9	...	990	991	992	993	994	995	996	997	998	999
COVID test result	0	0	1	1	0	0	0	0	0	0	...	1	0	0	1	0	0	0	1	0	0

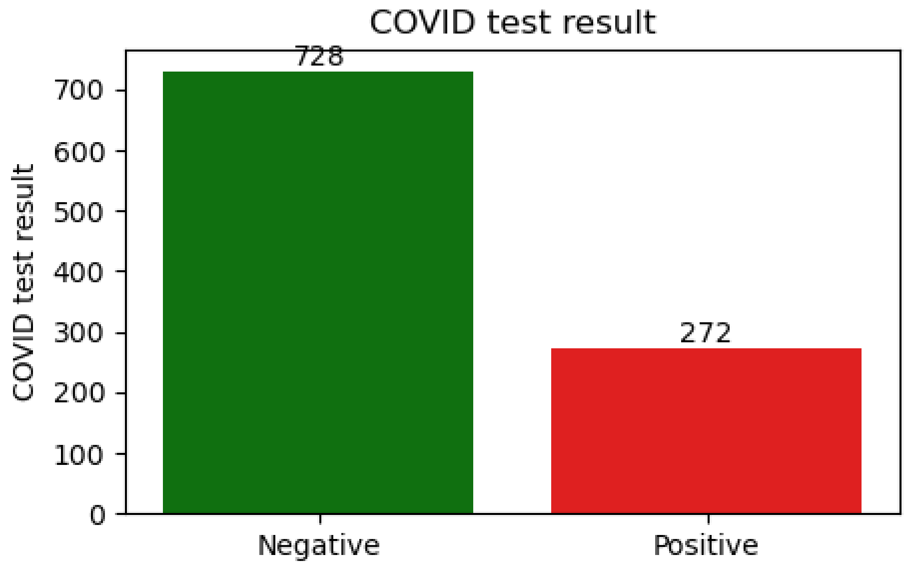
1 rows × 1000 columns

In [4]:
```
positive_result=data[data['COVID test result']==1].count()
negative_result=data[data['COVID test result']==0].count()
```

In [5]:
```
data['COVID test result'].value_counts()
```

Out[5]:
```
0    728
1    272
Name: COVID test result, dtype: int64
```

In [6]:
```
plt.figure(figsize=(5,3))
plt.title("COVID test result")
ax=sns.barplot(x=['Negative','Positive'], y=data['COVID test result'].value_counts(),palette=['green','red'])
for i in ax.containers:
    ax.bar_label(i,)
plt.show()
```


COVID test result
728
272
700
600
500
400
300
200
100
0
COVID test result
Negative
Positive

2.

In [7]:
```
n=1000
k=positive_result

def likelihood(s,p,q):
    t=s*p+(1-s)*q #The chance of a positive result in a covid test
    return sc.stats.binom.pmf(k, n, t)

def prior(s,p,q):
    s_prior=sc.stats.uniform.ppf(0,1)
    p_prior=2*p
    q_prior=2*(1-q)
    if any(x >= 0 and x<=1 for x in (s,p,q)):
        return s_prior*p_prior*q_prior
    else:
        return 0

# Create function to compute acceptance ratio
# This function will accept the current and proposed values of p,q and s
def acceptance_ratio(s,p,q,s_new,p_new,q_new):
    # Return R, using the functions we created before
    return min(1, ((likelihood(s_new,p_new,q_new) / likelihood(s,p,q)) * (prior(s_new,p_new,q_new) / prior(s,p,q))))
```

Lag

We need the samples (from the posteriro) to be independents. A new sample here is often dependent on the previous one as occasionally we do not accept a new random value and keep the old. To address this problem, I implement what is called "lag". Lag is where rather than save every sample, we save every other, or perhaps every fifth or tenth sample.

MCMC algorithm implement

In [8]:
```
# Create empty list to store  the posterior samples
samples = []

# Initialzie a value of p,q,s
s = np.random.uniform(0, 1)
p = np.random.uniform(0, 1)
q = np.random.uniform(0, 1)

# Define model parameters
n_samples = 1000000
lag = 100
```

```python
# Create the MCMC loop
for i in range(n_samples):
    # Propose a new value of p randomly from a uniform distribution between 0 and 1
    s_new = np.random.uniform(0, 1)
    p_new = np.random.uniform(0, 1)
    q_new = np.random.uniform(0, 1)

    # Compute acceptance probability
    R = acceptance_ratio(s,p,q,s_new,p_new,q_new)
    # Draw random sample to compare R to
    u = np.random.random_sample()
    # If R is greater than u, accept the new value of s,p and q (set p = p_new,etc.)
    if u < R:
        s = s_new
        p = p_new
        q = q_new
    teta=[s,p,q]
    # Record values after lag
    if i%lag == 0:
        samples.append(teta)
```

In [9]:
```python
np.shape(samples)
```

Out[9]:
```
(10000, 3)
```

In [14]:
```python
s_samples=[]
p_samples=[]
q_samples=[]
s_samples.append([item[0] for item in samples])
p_samples.append([item[1] for item in samples])
q_samples.append([item[2] for item in samples])
s_samples=s_samples[0]
p_samples=p_samples[0]
q_samples=q_samples[0]
```

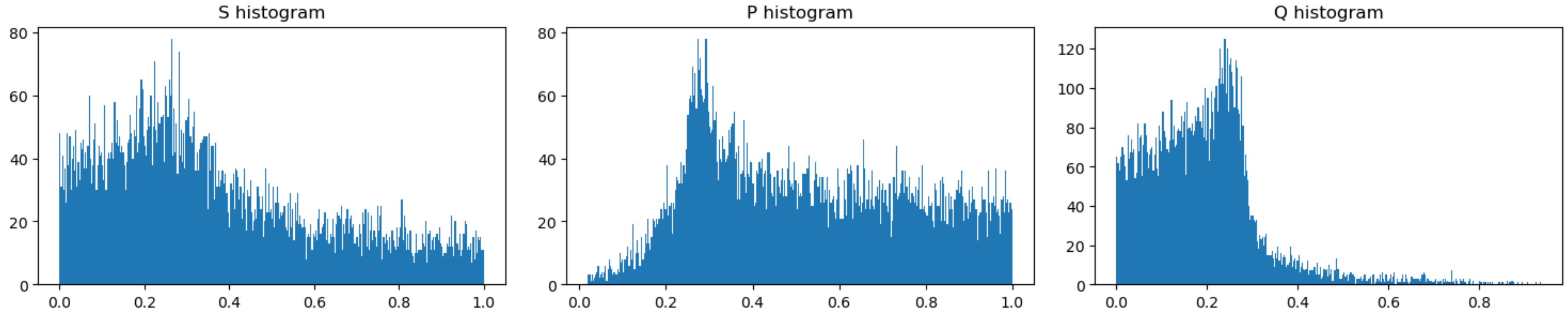
In [16]:
```python
np.shape(s_samples)
```

Out[16]:
```
(10000,)
```

3.

One-dimensional marginal distributions

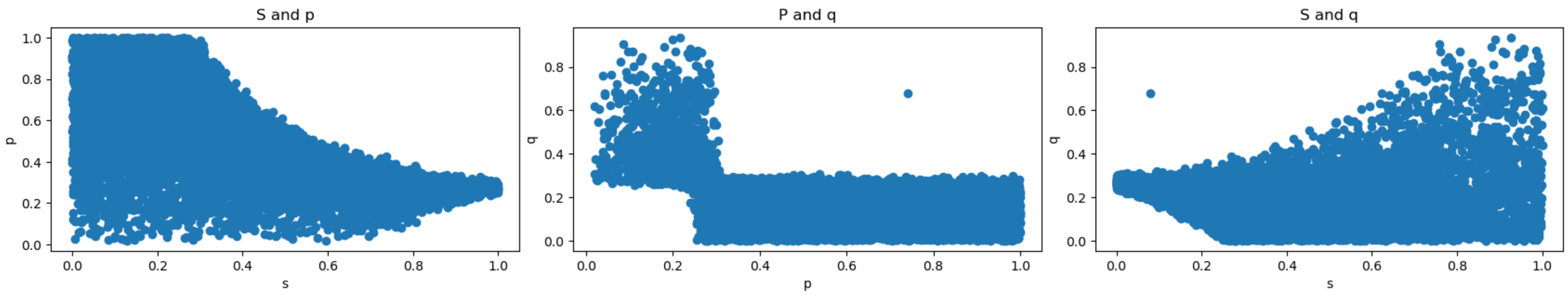
In [17]:
```
#plt.figure(figsize=(4,7))
fig,(ax1,ax2,ax3) = plt.subplots(1,3,figsize=(15, 3))
fig.tight_layout()
ax1.hist(s_samples,bins=350)
ax2.hist(p_samples,bins=350)
ax3.hist(q_samples,bins=350)
ax1.title.set_text('S histogram')
ax2.title.set_text('P histogram')
ax3.title.set_text('Q histogram')
plt.show()
```



Two-dimensional marginal distributions

In [18]:
```
#plt.figure(figsize=(4,7))
fig,(ax1,ax2,ax3) = plt.subplots(1,3,figsize=(17, 3))
fig.tight_layout()
ax1.scatter(s_samples,p_samples)
ax1.set_xlabel('s')
ax1.set_ylabel('p')
ax2.scatter(p_samples,q_samples)
ax2.set_xlabel('p')
ax2.set_ylabel('q')
ax3.scatter(s_samples,q_samples)
ax3.set_xlabel('s')
ax3.set_ylabel('q')
ax1.title.set_text('S and p')
ax2.title.set_text('P and q')
```

```
ax3.title.set_text('S and q')
plt.show()
```



5.

max,evarage and median

In [23]:
```
value=0
for item in samples:
    new_value=likelihood(item[0],item[1],item[2])*prior(item[0],item[1],item[2])
    if new_value>value:
        value=new_value
        parameters=item
print("The max value (for 3D) is:" +str(value[0]),
      "\nThe parameters that maximazed the posteriro is:\nS:"+str(parameters[0]),"\nP:"+str(parameters[1]),
      "\nQ:"+str(parameters[2]))
```

```
The max value (for 3D) is:0.11145187492346192
The parameters that maximazed the posteriro is:
S:0.26351253091662086
P:0.9958933437552981
Q:0.012634616157119694
```

In [24]:
```
print("The evarage is:\nS:"+str(np.mean(s_samples)),"\nP:"+str(np.mean(p_samples)),"\nQ:"+str(np.mean(q_samples)),"\n",
     "\nThe median is:\nS:"+str(np.median(s_samples)),"\nP:"+str(np.median(p_samples)),"\nQ:"+str(np.median(q_samples)))
```

```
The evarage is:
S:0.3740632268522812
P:0.5282107508558151
Q:0.19247544304329123

The median is:
S:0.31090421977935856
P:0.49894669384243606
Q:0.18505075101920765
```

6.

In [36]:
```python
max_q=0.9
min_q=0
quantiles=[[min_q,max_q]]
for i in range(0,10):
    max_q=np.round(max_q+0.01,3)
    min_q=np.round(min_q+0.01,3)
    new_Q=[min_q,max_q]
    quantiles.append(new_Q)
```

In [37]:
```python
quantiles
```

Out[37]:
```
[[0, 0.9],
 [0.01, 0.91],
 [0.02, 0.92],
 [0.03, 0.93],
 [0.04, 0.94],
 [0.05, 0.95],
 [0.06, 0.96],
 [0.07, 0.97],
 [0.08, 0.98],
 [0.09, 0.99],
 [0.1, 1.0]]
```

In [38]:
```python
def confidence(list):
    range=1
    for item in quantiles:
        lower=np.quantile(list,q=item[0])
        upper=np.quantile(list,q=item[1])
        if (upper-lower)<range:
            range=upper-lower
```

```python
            r=[lower,upper]
    return r
```

In [40]:
```python
print("The smallest range in which the parameter S lies with a 90% chance:",confidence(s_samples),
     "\nThe smallest range in which the parameter p lies with a 90% chance:" +str(confidence(p_samples)),
     "\nThe smallest range in which the parameter q lies with a 90% chance:" +str(confidence(q_samples)))
```

```
The smallest range in which the parameter S lies with a 90% chance: [3.5902820645916655e-05, 0.7880315426528474]
The smallest range in which the parameter p lies with a 90% chance:[0.2194116877296459, 0.9782801804613789]
The smallest range in which the parameter q lies with a 90% chance:[6.35918349255471e-05, 0.3220013705494933]
```

In [ ]:

In [ ]: