Problem Set 4: Posterior Predictive Checks and other exercises.

"All models are wrong, but some are useful"

George E.P. Box

Posterior predictive tests medical trial

- Choose test statistic that shows the non-hierarchical model does NOT explain the control group data well.
- generate posterior samples from model
- compare posterior of model and data set for the statistic

To submit:

- Test statistic function
- code for generating samples from the posterior
- code for computing the test statistic
- histogram analogous to Gelman 6.4
- p-value

Non-hierarchical Model Summary

Assumptions:

- all studies have the same probability of success θ_t for treatment, θ_c for control.
- binomial distribution in each study
- uniform priors over θ_t and θ_c

Model:

- Likelihood $\prod_{i=1}^{6} Binomial(s_i|\theta, n_i)$
- Prior: $Beta(\theta|1, 1)$ for θ_t and θ_c
- posterior for treatment group: $Beta(\theta_t|108, 35)$
- posterior for control group: $Beta(\theta_c|58, 65)$

In [1]:

```
import numpy as np
import scipy.stats as sts
from scipy.special import betaln
import matplotlib.pyplot as plt
```

In [2]:

```python
# The control group data set from the original pre-class work
control_group_size = [15,18,10,39,29,10]
control_group_successes = [9,11,4,21,12,0]
```

In [3]:

```python
# a function for generating fake data from the posterior
def nonhierarchical_rvs(n):
    data = []
    # generate n fake data sets
    for i in range(n):
        #sample a probability theta_c from the posterior
        current_p = sts.beta.rvs(58,65, size = 1)
        sample = []
        # generat groups of the same size as the original control groups
        for group_size in control_group_size:
            sample.append(sts.binom.rvs(group_size,current_p))
        data.append(sample)
    return data
```

```
In [4]:
```

```python
# generate 1000 samples for the
posterior_data = nonhierarchical_rvs(100000)
print(posterior_data[:10])
```

```
[[3, 5, 5, 17, 13, 3], [5, 10, 6, 15, 14, 8], [7, 9, 5, 18, 10, 7], [1
2, 8, 4, 19, 17, 4], [7, 13, 5, 16, 15, 2], [7, 9, 3, 23, 20, 3], [8,
10, 4, 18, 12, 5], [7, 9, 4, 17, 7, 4], [3, 7, 5, 16, 13, 1], [6, 11,
6, 18, 17, 8]]
```

After inputting the observed data and generating 100000 datasets with the same number of patients and probability of success sampled from our model, we can look at different scalar parameters that are basically a one-number summary of some aspect of our data.

```
In [5]:
```

```python
# function defining three example test statistics
def test_statistic(data,name):
    stats = {'range': np.ptp,
             'mean'  : np.mean,
             'variance': np.var,
             'minimum': np.min,
             'maximum': np.max}
    return stats[name](data)
```

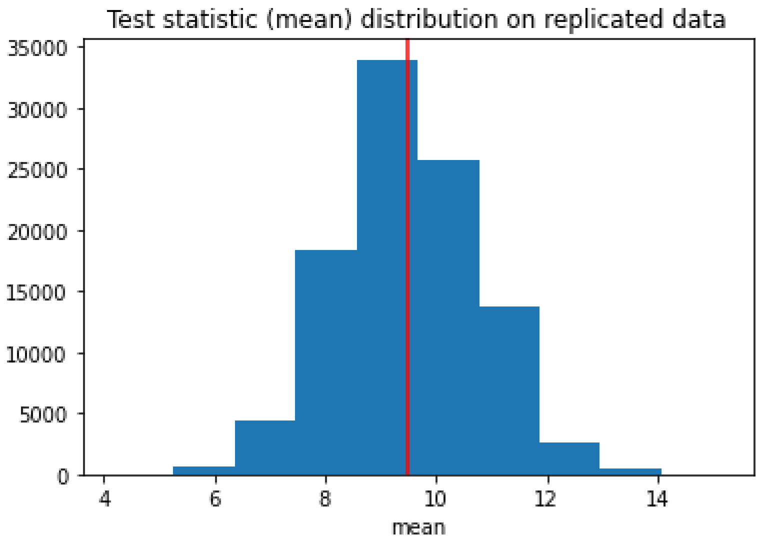
In [6]:

```python
# function calculating the test statistic for the data set and calculating the p
def create_plot(posterior_samples, data, name):
    # calculate the test statistic for the posterior samples
    # and the control group data
    test_statistic_posterior = [test_statistic(x, name) for x in posterior_data]
    test_statistic_control = test_statistic(control_group_successes, name)

    # create plot
    plt.hist(test_statistic_posterior, label = 'posterior samples')
    plt.axvline(test_statistic_control, color = 'red',label = 'data')
    plt.xlabel(name)
    plt.title('Test statistic ({}) distribution on replicated data'.format(name)
    plt.show()

    # calculate p-value
    print(name,'statistic P-value: %.3f' % (
    np.mean(np.array(test_statistic_posterior) > test_statistic_control)))
```

One statistic that is our model is unsurprisingly predicting pretty well is the mean. This makes sense because our model assumes that all experiments have the same probability of success and if that was the case the mean is similar distributed than if there is huge variance between the probabilities of success for each experiment.
The p-value for the mean is 0.476 which is close to 0.5.

In [7]:

```
create_plot(posterior_data, control_group_successes,'mean')
```

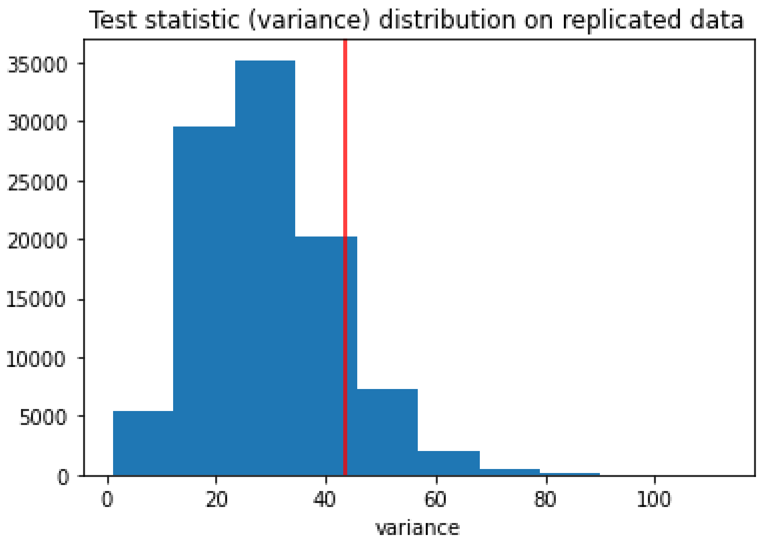


```
mean statistic P-value: 0.476
```

Talking of variance, this is the next test statistic I was interested in. The variance of the simulated dataset is consistently much lower than the variance in the observed data. The p-value here is 0.124, which is not good, but also not below our threshold of 0.05 or above 0.95.

In [8]:

```
create_plot(posterior_data, control_group_successes,'variance')
```



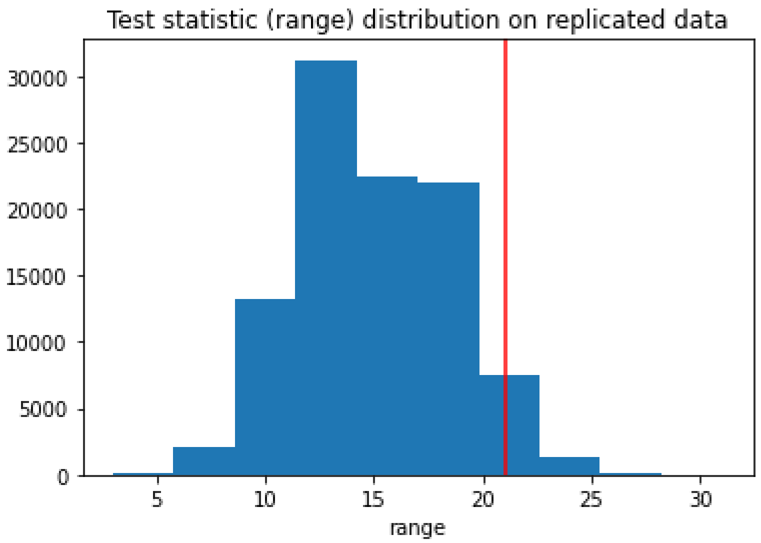
```
variance statistic P-value: 0.124
```

We can do better (or I guess worse).
In my search for a test statistic that is really not captured by the model, I found the range of values so the maximum-minimum of each data set.

The p-value of this metric is 0.028, so below 0.05, which suggests our model really does not capture the extreme values of the data.

In [9]:

```
create_plot(posterior_data, control_group_successes,'range')
```

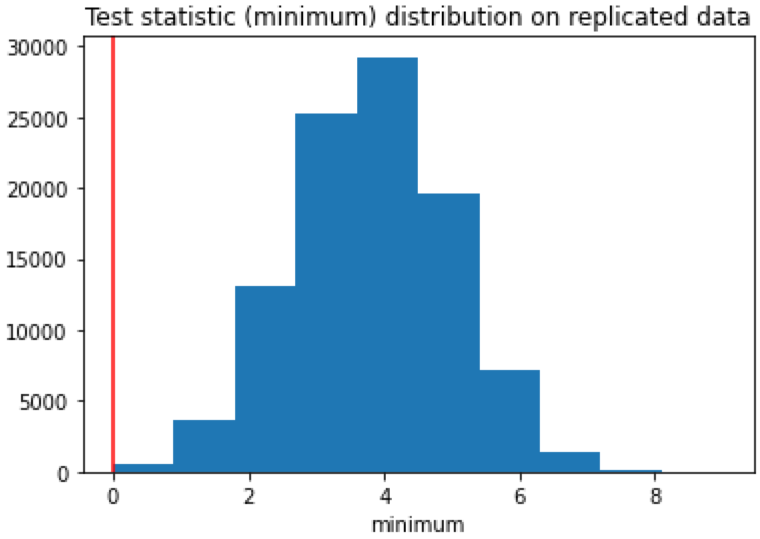


```
range statistic P-value: 0.028
```

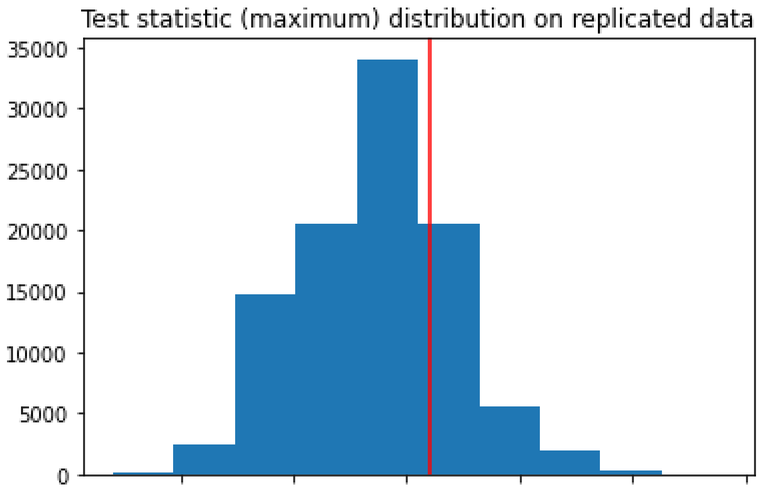
Just for fun, here are two more test statistics, the min and max. The min falls above the 0.95 threshold with a p-value of 0.995, but the max is doing somewhat better with a p-value of 0.195.

In [10]:

```
create_plot(posterior_data, control_group_successes,'minimum')
create_plot(posterior_data, control_group_successes,'maximum')
```



```
minimum statistic P-value: 0.995
```



```
maximum statistic P-value: 0.195
```