STAT 513/413: Lecture 11
Sampling and resampling

(bootstrap, permutation tests, and all that)

What we cannot establish by probabilistic calculations, we accomplish by simulations

Assignment Project Exam Help

Rizzo 7.1, 7.2, 7.3, 8.1

https://powcoder.com

Add WeChat powcoder

The median: recall

The *median* of a probability distribution P is defined to be a number m such that

$$P((-\infty, m]) \geqslant 1/2 \qquad \text{and} \qquad P[(m, +\infty)) \geqslant 1/2$$

A very common situation is that

$$P((-\infty, m] = P[(m, +\infty)) = 1/2$$

In such a case, the median is $Q(0.5) = F^{-1}(0.5)$

Assignment Project Exam Help

where F is the cumulative distribution function of P

https://powcoder.com

Add WeChat powcoder

Right now, we are not interested in what is the median, μ, of the distribution, but we are interested its estimator M_{10}, the *sample median*, taken out of sample of 10 numbers that look like the outcomes of independent random variables with exponential distribution with $\lambda = 1$: how does it perform?

For starters: one of the performance measures of an estimator is its variance

Variance is easy: but is it the right thing here?

```
> var(replicate(100,median(rexp(10))))
[1] 0.09402476
> var(replicate(100,median(rexp(10))))
[1] 0.09007331
> var(replicate(100,median(rexp(10))))
[1] 0.08155923
> var(replicate(1000,median(rexp(10))))
[1] 0.09270415
> var(replicate(1000,median(rexp(10))))
[1] 0.09723465
> var(replicate(1000,median(rexp(10))))
[1] 0.09982106
> var(replicate(100000,median(rexp(10))))
[1] 0.09628878
> var(replicate(100000,median(rexp(10))))
[1] 0.09590175
```

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Note the difference: M_{10} estimates m out of $n = 10$ numbers

while T_N estimates Var(M_{10}) out of $N = 100000$ results

Is M_{10} an unbiased estimator of m?

A general measure of the quality of the estimator is the mean squared error $\mathsf{E}\left[(M_{10} - m)^2\right]$. The variance is linked to it through bias-variance decomposition

$$
\begin{aligned}
\mathsf{E}\left[(M_{10} - m)^2\right] &= \mathsf{E}\left[(M_{10} - \mathsf{E}(M_{10}))^2\right] + \mathsf{E}\left[(\mathsf{E}(M_{10}) - m)^2\right] \\
&\quad + \mathsf{E}\left[2(M_{10} - \mathsf{E}(M_{10}))(\mathsf{E}(M_{10}) - m)\right] \\
&= \mathsf{Var}(M_{10}) + (m - \mathsf{E}(M_{10}))^2
\end{aligned}
$$

Assignment Project Exam Help

as

$$
\begin{aligned}
\mathsf{E}\left[2(M_{10} - \mathsf{E}(M_{10}))(\mathsf{E}(M_{10}) - m)\right] &= 2(\mathsf{E}(M_{10}) - m)\,\mathsf{E}\left[M_{10} - \mathsf{E}(M_{10})\right] \\
&= 2(\mathsf{E}(M_{10}) - m)(\mathsf{E}(M_{10}) - \mathsf{E}(M_{10})) = 0
\end{aligned}
$$

https://powcoder.com
Add WeChat powcoder

The expression $m - \mathsf{E}(M_{10})$ is called *bias* - hence *bias-variance decomposition* (which apparently holds in greater generality, not ust for M_{10})

The estimates for which is bias equal to zero - and thence their expected value is equal to the estimated quantity - are called *unbiased*

Is $\mathsf{E}(M_{10}) = m$ - that is, is M_{10} an unbiased estimator of m?

A quick check of bias

```
> mean(replicate(100000,median(rexp(10))))
[1] 0.745127
> median(rexp(100000))
[1] 0.6890036
```

We can see that bias is likely not equal to 0 - even without knowing the exact value

Assignment Project Exam Help

```
> mm=-log(0.5)
> mm
[1] 0.6931472
```

https://powcoder.com

Add WeChat powcoder

We estimated it by 0.6890036, and the difference with 0.745127 seems to be big enough to be just attributed to chance

So M_{10} is apparently not unbiased. However, if we increase n and take M_{100} instead, we can see the situation improving

```
> mean(replicate(100000,median(rexp(100))))
[1] 0.6987744
```

Seems like some consistency holds there...

Back to mean squared error

So, let us do mean squared error now. First, there is a possibility that we actually know an estimated value

```
> mm=-log(0.5)
> mm
[1] 0.6931472
> mean(replicate(10,(median(rexp(10))-mm)^2))
[1] 0.06314839
> mean(replicate(100,(median(rexp(10))-mm)^2))
[1] 0.08828418
> mean(replicate(100,(median(rexp(10))-mm)^2))
[1] 0.1249923
> mean(replicate(100,(median(rexp(10))-mm)^2))
[1] 0.06841089
> mean(replicate(100000,(median(rexp(10))-mm)^2))
[1] 0.09911663
> mean(replicate(100000,(median(rexp(10))-mm)^2))
[1] 0.09856143
> mean(replicate(100000,(median(rexp(10))-mm)^2))
[1] 0.0979045
```

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

What if we do not know the estimated value?

Well, then instead of `-log(0.5)` we have to do something else, and if it involves random numbers, we may need to wait a bit longer

```
> mm=median(rexp(10000000))    ## consistency, remember?
> mm
[1] 0.6930446
> mean(replicate(100000,(median(rexp(10))-mm)^2))
[1] 0.09919107
> mean(replicate(100000,(median(rexp(10))-mm)^2))
[1] 0.0988128
> mean(replicate(100000,(median(rexp(10))-mm)^2))
[1] 0.09927002
```

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Confronting with large n approximation

Actually, theory does not help us too much here. The only useful theoretical result concerns M_n for n large. In such a case, M_n is very close to μ, and the mean squared error is thus very close to variance, which in turn can be approximated by

$$\frac{1}{4nf^2(\mu)} \qquad \text{(theorem by Kolmogorov)}$$

where f is the density of the distribution underlying the sample in our case.

Assignment Project Exam Help

Looks like this not that bad approximation already for $n = 100$

https://powcoder.com

Add WeChat powcoder

```
> mm = -log(0.5)
> mean(replicate(100000,(median(rexp(100))-mm)^2))
[1] 0.01004371
> 1/(4*100*dexp(log(2))^2)
[1] 0.01
```

but not that good for $n = 10$ (which is apparently not that large)

```
> 1/(4*10*dexp(log(2))^2)
[1] 0.1      # compare to the Monte Carlo value above
```

A research story

So, we have the estimator, sample median M_{10}, of m

This estimator works in general - for various distributions

say, for the exponential distribution with $\lambda = 1$

or for the exponential distribution with unknown λ

However, when the data follow an exponential distribution

that is, an exponential distribution with unknown λ

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

In the latter case, we may actually devise a better estimator for m

How about $Q(0.5)$? Well, actually it is $Q_\lambda(0.5)$

and λ we do not know

but we can estimate it by $1/\bar{X}$

Well, and how do we know it is better?

- well, mean squared error, is it not?

Theory for mean-squared error

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Computer experiment

Let us calculate the mean squared error - in a way that was already shown above - for a couple of selected λ and we will see (as a blind said to a deaf). A tiny script:

```
N <- 100000
n <- 10
las <- c(0.01,0.1,1,10,100)
rslt <- matrix(0,2,length(las))
for (k in (1:length(las))) {
  mm <- qexp(0.5,las[k])
  rslt[1,k] <-
    mean(replicate(N,(median(rexp(n,las[k])) - mm)^2))
  rslt[2,k] <-
    mean(replicate(N,(qexp(0.5,1/mean(rexp(n,las[k]))) - mm)^2))
}
print(rslt)
```

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

And the result

```
> source("twomed.R")
         [,1]     [,2]       [,3]         [,4]         [,5]
[1,] 998.0411 9.915687 0.09901681 0.0009827265 9.896574e-06
[2,] 477.9990 4.822185 0.04794337 0.0004825622 4.823859e-06
```

So, we seem to be good...

for five values of λ

and the $N = 100000$ used

and when the data come from an exponential distribution

and if we did not have an error in the code

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

An error???

For instance, we could change n to $n = 100$ now

```
> source("twomed.R")
         [,1]      [,2]        [,3]         [,4]         [,5]
[1,] 99.81847 0.9905156 0.009954382 1.010295e-04 9.962715e-07
[2,] 47.99285 0.4794193 0.004777554 4.797323e-05 4.821234e-07
```

and then try the large-sample approximation for the first line

```
> las
[1] 1e-02 1e-01 1e+00 1e+01 1e+02
> 1/(4*100*dexp(qexp(0.5,las),las)^2)
[1] 1e+02 1e+00 1e-02 1e-04 1e-06
```

and for the second one

```
> (-log(0.5)/(las*sqrt(n)))^2
[1] 4.80453e+01 4.80453e-01 4.80453e-03 4.80453e-05 4.80453e-07
```

Seems like we are good...

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Aside: an even a bit better way

```
N <- 100000
n <- 10
las = c(0.01,0.1,1,10,100)
rslt <- matrix(0,2,length(las))
mses <- matrix(0,2,N)
for (k in (1:length(las))) {
  for (rep in 1:N) {
    mm <- qexp(0.5,las[k])
    sss <- rexp(n,las[k])
    mses[1,rep] <- (median(sss) - mm)^2
    mses[2,rep] <- (qexp(0.5,1/mean(sss)) - mm)^2
  }
  rslt[1,k] <- mean(mses[1,])
  rslt[2,k] <- mean(mses[2,])
}
print(rslt)
```

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Not that different results, however

```
> source("twomod.R")
          [,1]     [,2]       [,3]         [,4]         [,5]
[1,] 987.1046 9.903584 0.09875735 0.0009891487 9.820579e-06
[2,] 483.8226 4.801344 0.04818387 0.0004835474 4.800250e-06
```

Recall the earlier

Assignment Project Exam Help

```
> source("twomed.R")
          [,1]     [,2]       [,3]         [,4]         [,5]
[1,] 998.0411 9.915687 0.09901681 0.0009827265 9.896574e-06
[2,] 477.9990 4.822185 0.04794337 0.0004825622 4.823859e-06
```

https://powcoder.com

Add WeChat powcoder

And now for something a bit different

Fine... so the above gave me the mean squared error (which may be close to variance or not) for a sample median out of $n = 10$ *when sampled from the exponential distribution with* $\lambda = 1$

But now I have the following 10 numbers; I compute the median from them - any possibility to assess the precision of that?

Assignment Project Exam Help

```
> exs
 [1]  2.710660  1.100322 11.934400  1.419022  1.523077
 [6]  5.384740  2.801879  2.163556  4.837161 10.374159
> mm=median(exs)
> mm
[1] 2.75627
> var(exs)
[1] 14.67475
```

https://powcoder.com

Add WeChat powcoder

Yes, but does this say anything at all here?

```
> mean((exs-mm)^2)
[1] 15.99159
```

And this one??

Bootstrap!

How about taking the mean squared differences of the sample median of “the sample” (=original batch of numbers) from the sample medians repeatedly calculated... from the data at hand!

```
> mean((replicate(10,median(sample(exs,10)))-mm)^2)
[1] 0
```

Well, if we sample n numbers from n numbers (“resample”), then we always get the same thing; we need to do it *with replacement*

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

```
> mean((replicate(10,median(sample(exs,10,replace=TRUE)))-mm)^2)
[1] 2.362092
> mean((replicate(10,median(sample(exs,10,replace=TRUE)))-mm)^2)
[1] 3.502714
> mean((replicate(10,median(sample(exs,10,replace=TRUE)))-mm)^2)
[1] 3.041924
> mean((replicate(100000,median(sample(exs,10,replace=TRUE)))-mm)^2)
[1] 2.125004
> mean((replicate(100000,median(sample(exs,10,replace=TRUE)))-mm)^2)
[1] 2.111911
> mean((replicate(100000,median(sample(exs,10,replace=TRUE)))-mm)^2)
[1] 2.125408
```

The scheme of bootstrap

We are trying to figure out the quantity that may involve

an unknown constant(s) - here let it be just one, m

And also depends on random variables $X_1, \ldots X_n$

independent and with the same distribution

which are believed to be those whose outcomes model the behavior of the observations $x_1, \ldots x_n$ ("sample")

Assignment Project Exam Help

A *bootstrap estimate* of the desired quantity is then obtained by

https://powcoder.com

estimating the unknown constant(s) from the original $x_1, x_2, \ldots x_n$

Add WeChat powcoder

and then estimating the quantity itself from N batches of n numbers $x^*_{1i}, x^*_{1i} \ldots x^*_{ni}$ sampled from the original $x_1, x_2, \ldots x_n$ *with replacement* ("resampling")

For instance, we are after

$\mathsf{E}\left[(M_{10}(X_1, \ldots, X_{10}) - m)^2\right]$ which we estimate by

$$\frac{1}{N}\sum_{i=1}^{N}\left(M_{10}(x^*_{1i}, \ldots, x^*_{10,i}) - M_{10}(x_1, \ldots, x_{10})\right)^2$$

Excercise: bias, variance

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

Bias and variance

We know the sample median is in general biased... Can we obtain the estimate of the bias? (bias = the difference between the estimated value and the expected value of the estimator; equal zero for unbiased estimators)

```
> mm-mean(replicate(10000,median(sample(exs,10,replace=TRUE))))
[1] -0.4691844
```

Assignment Project Exam Help

What is the variance (may be needed for confidence intervals, say)

https://powcoder.com

```
> var(replicate(10000,median(sample(exs,10,replace=TRUE))))
[1] 1.895233
```

Add WeChat powcoder

Efron & Tibshirani say that 200 would be enough (instead of 10000)... They may be right...

```
> var(replicate(200,median(sample(exs,10,replace=TRUE))))
[1] 1.831597
> var(replicate(200,median(sample(exs,10,replace=TRUE))))
[1] 2.062268
> var(replicate(10000,median(sample(exs,10,replace=TRUE))))
[1] 1.931406
```

And now, hocus-pocus

Bias-variance decomposition again: is it true here?

```
> set.seed(007); mse=mean((replicate(1000000,
+ median(sample(exs,10,replace=TRUE)))-mm)^2)
> mse
[1] 2.116876
> set.seed(007); mvr=var(replicate(1000000,
+ median(sample(exs,10,replace=TRUE))))
> mvr
[1] 1.881946
> set.seed(007); bias=mm-mean(replicate(1000000,
+ median(sample(exs,10,replace=TRUE))))
> bias
[1] -0.4846973
> mvr+bias^2
[1] 2.116877
> mse
[1] 2.116876
```

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

If time permits, we may return to cover bootstrap confidence intervals

And again different: permutation tests

We have two batches of 10 numbers:

```
> s1
[1]  3.7551030 11.7892438  4.1296516  0.9881743  1.1722081
[6]  1.1131551  7.5318461  4.9694114  0.6259583  1.3886535
> s2
[1] 1.2574818 1.9363749 1.4094235 0.9201450 2.4456553
[6] 4.9157286 0.7597842 0.5466244 0.7560977 1.5443949
```

Their means turn out to be quite different

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

```
> mean(s1)
[1] 3.746341
> mean(s2)
[1] 1.649171
> dss=mean(s1)-mean(s2)
> dss
[1] 2.097169
```

Is this difference significant at level 0.05?

The usual take on it

```
> t.test(s1,s2,alternative="greater")

Welch Two Sample t-test
...
t = 1.7312, df = 11.26, p-value = 0.05534
alternative hypothesis: true difference in means is greater than 0
...
mean of x mean of y
 3.746341  1.649171
```

Assignment Project Exam Help

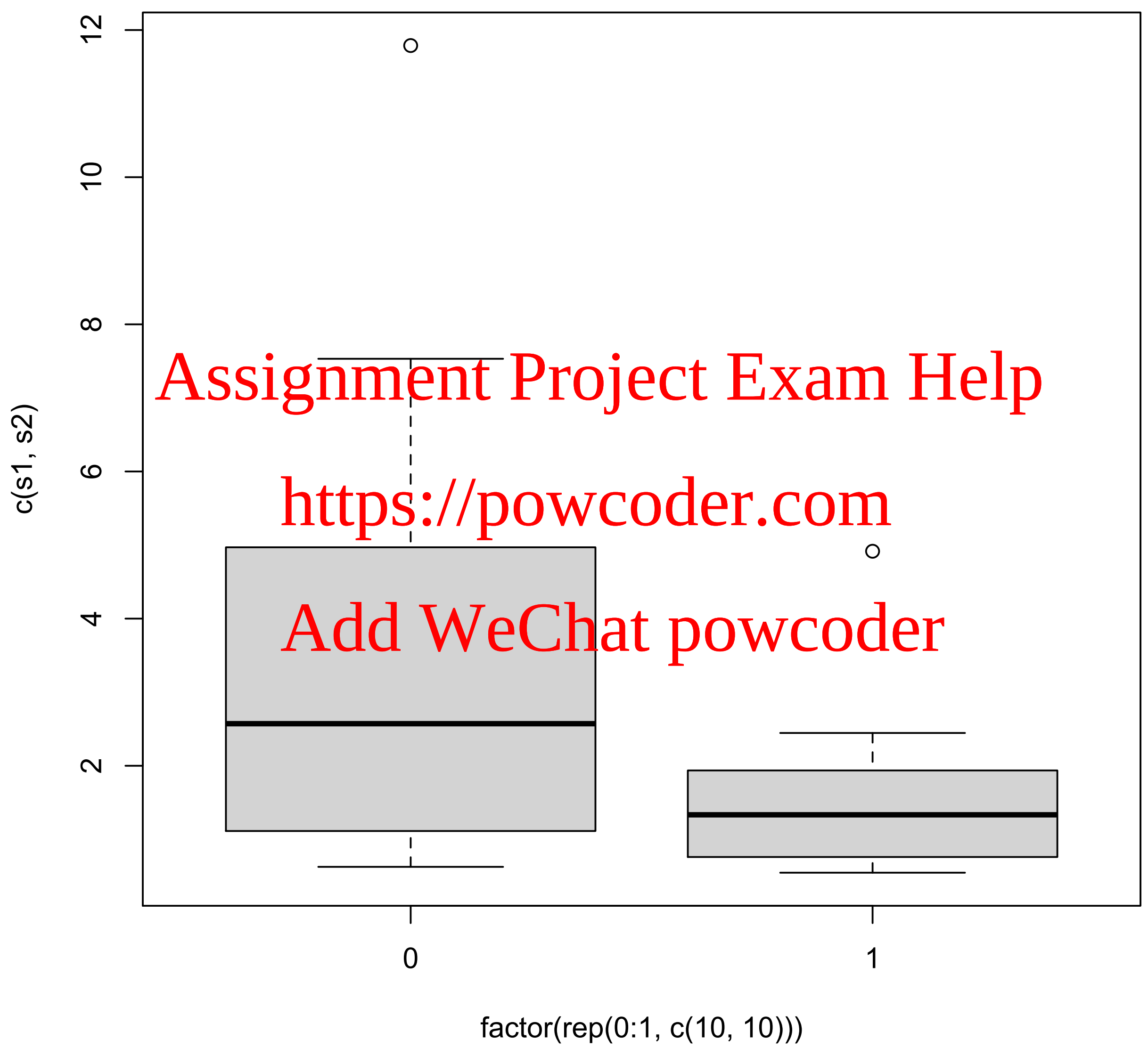
https://powcoder.com

```
> t.test(s1,s2,var.equal=TRUE,alternative="greater")
```

Add WeChat powcoder

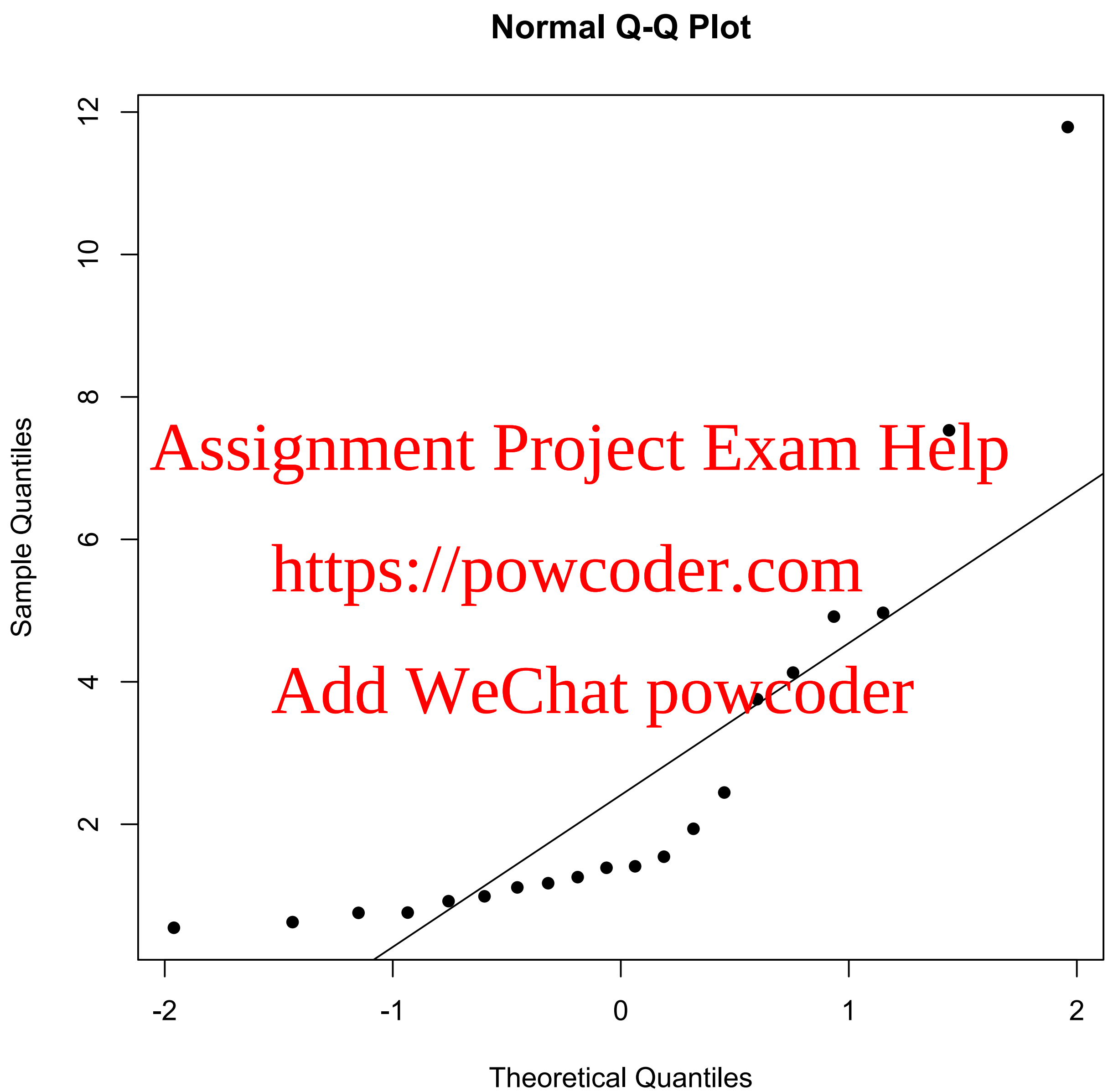
```
Two Sample t-test

t = 1.7312, df = 18, p-value = 0.05026
alternative hypothesis: true difference in means is greater than 0
...
mean of x mean of y
 3.746341  1.649171
```

The underlying assumptions?



Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

In particular, normality?



Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Note: putting both together is a bit tricky here... is it clear why?

Permutation test!

If you think of both as having no difference, you can think about them as that their assignment into `s1` and `s2` is by pure chance: it is then equally likely that they end up as they did, with difference in means `dss = 2.097169` as well as they do other way round, difference in means being then `dss = -2.097169` - and most differences will be around 0 anyway...

Assignment Project Exam Help

However, the observed difference `dss = 2.097169`: is it not somewhat unlikely? To see that, we could take all possible outcomes under random allocations into `s1` and `s2` - and figure out how many differences of means exceed 2.097169

https://powcoder.com

Add WeChat powcoder

That certainly can be done, if we have enough time to wait for the result of $\binom{20}{10}$ allocations... if not, then...

... then we can do just random sampling of those, cannot we?

So, now only how to do it

We first illustrate the code on a very simple numbers - so that we can see what is to be done

```
> s1
[1] 15 16 17 18 19
> s2
[1] 5 6 7 8 9
> s12 = c(s1,s2) ; s12
 [1] 15 16 17 18 19  5  6  7  8  9
> mean(s1)
[1] 17
> mean(s2)
[1] 7
> dss = mean(s1) - mean(s2) ; dss
[1] 10
> sss = sum(s12)/5 ; sss
[1] 24
> sss - 2*mean(s2)      ## this is my trick to have the code short
[1] 10
```

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

So, here is what we are going to do

The above was for the observed `s1` and `s2`. Now we combine their values into `s12` and then we are going to sample *new* `s1` and `s2`, again and again. For each of those, we compare the difference of their means with `dss`, the original difference of the means of `s1` and `s2`, and count the proportion of how many times it gets exceeded

```
> sample(s12,5)
[1] 17 16  5  6 19
> sss-2*mean(sample(s12,5))  ## note: sample is different - know why?
[1] 7.6
> replicate(10,sss-2*mean(sample(s12,5)))
 [1]  1.6 -6.0 -5.6  1.6 -1.6 -1.6  3.6  1.2  2.4  5.2
> replicate(10,sss-2*mean(sample(s12,5))) > dss
 [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
> mean(replicate(10,sss-2*mean(sample(s12,5))) > dss)
[1] 0   ## also here it could be different, but isn't - guess why
```

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

And now with real `s1` and `s2`

```
> s12=c(s1,s2)
> s12
 [1]  3.7551030 11.7892438  4.1296516  0.9881743  1.1722081
 [6]  1.1131551  7.5318461  4.9694114  0.6259583  1.3886535
[11]  1.2574818  1.9363749  1.4094235  0.9201450  2.4456553
[16]  4.9157286  0.7597842  0.5466244  0.7560977  1.5443949
> sss=sum(s12)/10 ; sss
[1] 5.395512
> dss
[1] 2.097169
> mean(replicate(100,sss-2*mean(sample(s12,10))) > dss)
[1] 0.06
> mean(replicate(100,sss-2*mean(sample(s12,10))) > dss)
[1] 0.07
```

Seems like it works now; only $N = 100$ does not yield stable result

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

The final run

```
> mean(replicate(100000,sss-2*mean(sample(s12,10))) > dss)
[1] 0.04684
> mean(replicate(100000,sss-2*mean(sample(s12,10))) > dss)
[1] 0.04713
> mean(replicate(100000,sss-2*mean(sample(s12,10))) > dss)
[1] 0.04674
> mean(replicate(100000,sss-2*mean(sample(s12,10))) > dss)
[1] 0.04633
```

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Seems like this is stable enough, and consistently below 0.05 - that is, the test would reject the null hypothesis that there is no difference. We can still give it a final run

```
> set.seed(007)
> mean(replicate(10000000,sss-2*mean(sample(s12,10))) > dss)
```

and when we return after getting a beer from the fridge, we find

```
[1] 0.0467867
```