STATS 205: Homework Assignment 2

Brian Liu

4/19/2019

Solution to Problem 1

```
patient_before     = c(1.83, 0.50, 1.62, 2.48, 1.68, 1.88, 1.55, 3.06, 1.30)
patient_before_mod = c(1.83, 0.50, 16.2, 2.48, 1.68, 1.88, 1.55, 3.06, 1.30)
patient_after      = c(0.878, 0.647, 0.598, 2.05, 1.06, 1.29, 1.06, 3.14, 1.29)

patient_z     = patient_after - patient_before
patient_z_mod = patient_after - patient_before_mod

# mean before modification
mean(patient_z)
```

```
## [1] -0.4318889
```

```
# mean after modification
mean(patient_z_mod)
```

```
## [1] -2.051889
```

The mean, $\bar{Z}$ is substantially different after the change. However,

```
# wilcox test before modification
wilcox.test(patient_before, patient_after, alternative="greater", paired=TRUE, conf.int = TRUE)
```

```
##
##  Wilcoxon signed rank test
##
## data:  patient_before and patient_after
## V = 40, p-value = 0.01953
## alternative hypothesis: true location shift is greater than 0
## 95 percent confidence interval:
##  0.175   Inf
## sample estimates:
## (pseudo)median
##           0.46
```

```
# wilcox test after modification
wilcox.test(patient_before, patient_after, alternative="greater", paired=TRUE, conf.int = TRUE)
```

```
##
##  Wilcoxon signed rank test
##
## data:  patient_before and patient_after
## V = 40, p-value = 0.01953
## alternative hypothesis: true location shift is greater than 0
## 95 percent confidence interval:
##  0.175   Inf
## sample estimates:
## (pseudo)median
```

```
##            0.46
```

the estimate of θ given by $\hat{\theta}$ is the same for both tests, before and after the modification of a single value:

```
## sample estimates:
## (pseudo)median
##            0.46
```

which is further evidence that supports Comment 16:

> The estimator $\hat{\theta}$ is relatively insensitive to outliers. This is not the case with the classical estimator $\bar{Z} = \sum_{i=1}^{n} Z_i/n$. Thus the use of $\hat{\theta}$ provides protection against gross errors.

Solution to Problem 2

The reason the Hodges-Lehmann estimator is less influenced by outlying observations than the sample mean of the Z's is that the Hodges-Lehmann estimator is computed by:

> *Sensitivity to Gross Errors.* For a dataset with n measurements, the set of all possible one- or two- element subsets of it has $n(n+1)/2$ elements. For each such subset, the mean is computed; finally, the median of these $n(n+1)/2$ averages is defined to be the Hodges-Lehmann estimator of location. – Hodges-Lehmann estimator

This is described by Equation 3.23 in HWC:

$$\hat{\theta} = \text{median}\left\{\frac{Z_i + Z_j}{2}, i \leq j = 1, ..., n\right\}$$

In other words, the Hodges-Lehmann estimator associated with the Wilcoxon test computes a value related to the median (the median of the averages of ranked, consecutive values), which does not take into account edge-cases, or outliers, and is therefore robust to outliers.

Solution to Problem 3

```
beak_dark = c(5.8, 13.5, 26.1, 7.4, 7.6, 23.0, 10.7, 9.1, 19.3, 26.3, 17.5, 17.9, 18.3, 14.2, 55.2, 15.

beak_lite = c(5, 21, 73, 25, 3, 77, 59, 13, 36, 46, 9, 25, 59, 38, 70, 36, 55, 46, 25, 30, 29, 46, 71,

beak_lite_mod = c(5, 21, 173, 25, 3, 77, 59, 13, 36, 46, 9, 25, 59, 38, 70, 36, 55, 46, 25, 30, 29, 46,

beak_diff     = beak_lite - beak_dark
beak_diff_mod = beak_lite_mod - beak_dark

mean(beak_diff)
```

```
## [1] 18.804
```

```
mean(beak_diff_mod)
```

```
## [1] 22.804
```

The mean, $\bar{Z}$ is substantially different after the change. However,

```
library(BSDA) # required to run SIGN.test
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'BSDA'

## The following object is masked from 'package:datasets':
##
##     Orange
```

```r
# sign test before modification
SIGN.test(beak_dark, beak_lite, estimate=TRUE)
```

```
##
##  Dependent-samples Sign-Test
##
## data:  beak_dark and beak_lite
## S = 4, p-value = 0.0009105
## alternative hypothesis: true median difference is not equal to 0
## 95 percent confidence interval:
##  -24.606258  -7.141663
## sample estimates:
## median of x-y
##         -17.6
##
## Achieved and Interpolated Confidence Intervals:
##
##                     Conf.Level   L.E.pt  U.E.pt
## Lower Achieved CI     0.8922 -23.8000 -7.5000
## Interpolated CI       0.9500 -24.6063 -7.1417
## Upper Achieved CI     0.9567 -24.7000 -7.1000
```

```r
# sign test after modification
SIGN.test(beak_dark, beak_lite_mod, estimate=TRUE)
```

```
##
##  Dependent-samples Sign-Test
##
## data:  beak_dark and beak_lite_mod
## S = 4, p-value = 0.0009105
## alternative hypothesis: true median difference is not equal to 0
## 95 percent confidence interval:
##  -24.606258  -7.141663
## sample estimates:
## median of x-y
##         -17.6
##
## Achieved and Interpolated Confidence Intervals:
##
##                     Conf.Level   L.E.pt  U.E.pt
## Lower Achieved CI     0.8922 -23.8000 -7.5000
## Interpolated CI       0.9500 -24.6063 -7.1417
## Upper Achieved CI     0.9567 -24.7000 -7.1000
```

the estimate of θ given by $\tilde{\theta}$ is the same for both tests, before and after the modification of a single value:

```
## sample estimates:
## median of x-y
##         -17.6
```

which is further evidence that supports Comment 40:

Sensitivity to Gross Errors. The estimator $\tilde{\theta}$ (3.58) is even less sensitive to outliers than the estimator $\hat{\theta}$ (3.23) associated with the signed rank statistics T^+ (3.3). (See Comment 16 and Problems 20 and 60.) As a result, $\tilde{\theta}$ protects well against gross errors. However, all the information contained in the collected sample is not utilized in computing $\tilde{\theta}$. Consequently, $\tilde{\theta}$ is rather inefficient for many populations.

Solution to Problem 4

NOTE: I realized after looking at Problem 6 that the notation for "Uniform(0,1)" meant that the mean is 0 and the standard deviation is 1, not that it's a Uniform distribution where the values range from 0 to 1. Oops. Maybe I can fix it later.

Small note about empty vectors and their usage [1]

(i)

```
x <- list()
for(i in 1:100000)
{
  # If min or max are not specified they assume the default values of 0 and 1 respectively.
  x[[i]] = runif(50)
}
maxes <- double(length(x))
for(i in 1:100000)
{
  curr_list = x[[i]]
  maxes[i] = max(curr_list)
}
# maxes
wilcox.test(maxes, mu = 1, conf.int = TRUE)
```

```
##
##  Wilcoxon signed rank test with continuity correction
##
## data:  maxes
## V = 0, p-value < 2.2e-16
## alternative hypothesis: true location is not equal to 1
## 95 percent confidence interval:
##  0.9833297 0.9835415
## sample estimates:
## (pseudo)median
##      0.9834422
```

According to the above, the approximation to the true distribution of $\hat{\theta} = 0.9832971$.

(ii)

```
library(boot)
```

```
##
## Attaching package: 'boot'
```

[1] This is a useful page about empty vectors and habits not to build when making them.

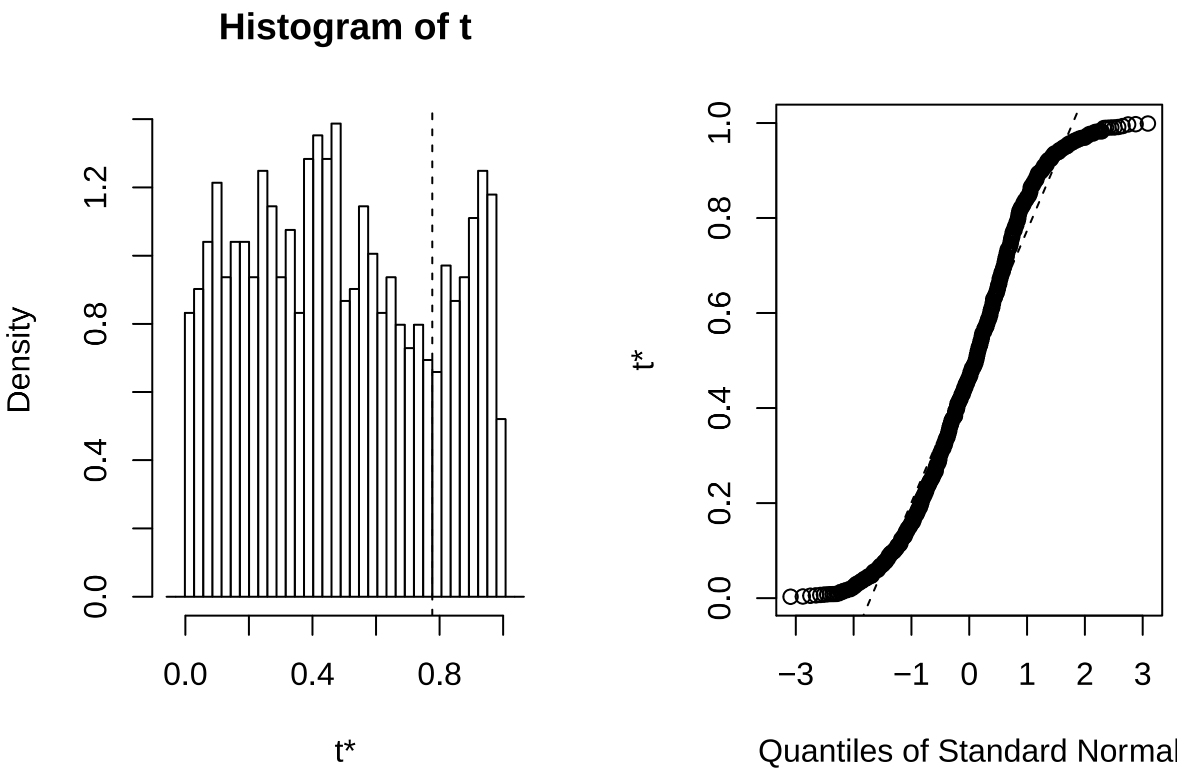
```
## The following object is masked from 'package:lattice':
##
##     melanoma
```

```r
runif_custom <- function(dat, ind) {
  return(runif(50))
}
boot_results <- boot(data = maxes, statistic = runif_custom, R=1000)
boot_results
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = maxes, statistic = runif_custom, R = 1000)
##
##
## Bootstrap Statistics :
##        original        bias    std. error
## t1*  0.77702961 -0.2894595548   0.2861709
## t2*  0.20998933  0.2819276715   0.2888316
## t3*  0.43577126  0.0609291047   0.2861995
## t4*  0.69097621 -0.1846050347   0.2966802
## t5*  0.25761388  0.2517806361   0.2941043
## t6*  0.51454982 -0.0115912590   0.2857544
## t7*  0.13027506  0.3675460549   0.2902551
## t8*  0.15222583  0.3586612692   0.2858322
## t9*  0.17473168  0.3082062584   0.2895240
## t10* 0.99714973 -0.4749641486   0.2914988
## t11* 0.29686621  0.1932820667   0.2857008
## t12* 0.02753882  0.4710144397   0.2878050
## t13* 0.53185019 -0.0263699631   0.2900505
## t14* 0.12522918  0.3599023145   0.2873351
## t15* 0.96438437 -0.4557739523   0.2929972
## t16* 0.52997864 -0.0364980322   0.2934992
## t17* 0.82944131 -0.3296802665   0.2809914
## t18* 0.03554089  0.4735912543   0.2861304
## t19* 0.18396833  0.3000459853   0.2887661
## t20* 0.85788755 -0.3611198272   0.2929694
## t21* 0.63845499 -0.1233940111   0.2892410
## t22* 0.20018703  0.2961391318   0.2855493
## t23* 0.91027357 -0.4169961874   0.2832966
## t24* 0.36768516  0.1209936589   0.2892679
## t25* 0.33928370  0.1538150772   0.2937720
## t26* 0.74090775 -0.2481671514   0.2905488
## t27* 0.09425031  0.3996449021   0.2900537
## t28* 0.86886351 -0.3680655366   0.2882963
## t29* 0.24235178  0.2477816301   0.2931636
## t30* 0.64548235 -0.1377686839   0.2852474
## t31* 0.37858200  0.1338602080   0.2841546
## t32* 0.67421296 -0.1526655841   0.2860526
## t33* 0.85523177 -0.3480227325   0.2926841
## t34* 0.82900100 -0.3177947879   0.2924377
## t35* 0.78557519 -0.2796838341   0.2881925
```

```
## t36* 0.77421602 -0.2603049324   0.2871458
## t37* 0.17364294  0.3059574681   0.2926887
## t38* 0.95398509 -0.4563812012   0.2890344
## t39* 0.86080687 -0.3703940949   0.2896816
## t40* 0.50446802 -0.0085735770   0.2898889
## t41* 0.74288900 -0.2475090606   0.2885765
## t42* 0.63486421 -0.1175754349   0.2942248
## t43* 0.05198426  0.4509870352   0.2878393
## t44* 0.86735136 -0.3647903666   0.2892814
## t45* 0.25914465  0.2360150194   0.2850747
## t46* 0.11453933  0.3756565428   0.2893047
## t47* 0.68521711 -0.1835607027   0.2908744
## t48* 0.65593235 -0.1413831902   0.2902077
## t49* 0.80839529 -0.3177772086   0.2876998
## t50* 0.47925442 -0.0005810513   0.2927867
```

```
plot(boot_results)
```




```
library(car) #qqPlot
```

```
## Loading required package: carData

##
## Attaching package: 'carData'

## The following objects are masked from 'package:BSDA':
##
##     Vocab, Wool

##
## Attaching package: 'car'
```

```
## The following object is masked from 'package:boot':
##
##     logit
```

```
library(wrapr) #qc
# qqPlot(maxes, boot_results)
```

Unfortunately, I get this error when I try to run the commented line. I probably spent too long trying to get qqPlot to work.

Solution to Problem 5

(*i*)

```
placebo = c(rep(1, 30), rep(0, 20))
treated = c(rep(1, 40), rep(0, 10))

nll <- function(p1, p2) {
  return(p1 - p2)
}

# library(stats4)
# mle(minuslogl = nll, start = list(p1 = .6, p2 = .8))
```

This is the error message associated with commented code

Solution to Problem 6

(*i*)

```
n <- rnorm(100, 1, 1)
# jackknife(x, theta = )
```