STA 135: Discussion 4

Chris Conley

February 1st, 2017

Inferences about a mean vector

Code and examples will be discussed.

Example 5.2

Import the physiology measurements derived from 20 people's sweat (Ewww!). This data is available on canvas.

```r
datadir <- "~/../Box Sync/sta135-winter17/textbook_data/"
sweat <- read.table(file.path(datadir, "T5-1.DAT"))
names(sweat) <- c("rate", "sodium", "potassium")
head(sweat)
```

```
##   rate sodium potassium
## 1  3.7   48.5       9.3
## 2  5.7   65.1       8.0
## 3  3.8   47.2      10.9
## 4  3.2   53.2      12.0
## 5  3.1   55.5       9.7
## 6  4.6   36.1       7.9
```

Test mean vector with Hotellings T^2 statistic

Write some functions to compute the critical value and the T^2test statistic. Note that these functions, especially the hotellings T^2 is not robust to missing data and assumes the sample covariance matrix is invertible. Which is more computationally efficient: computing T^2 as a function of Wilk's lambda or just the regular T^2 formula that is a generalization of the univariate t-test?

```
critical_value <- function(n,p, alpha) {
  ((n - 1)*p / (n - p)) *
    qf(p = 1 - alpha, df1 = p, df2 = n - p)
}
t2_stat <- function(dat, mu0) {
  X <- as.matrix(dat)
  n <- nrow(X)
  Xbar <- colMeans(X)
  d <- (Xbar - mu0)
  n * t(d) %*%solve(cov(X)) %*% d
}
```

Test mean vector with Hotellings T^2 statistic

$H_0 : \mu = (\mathbf{4}, \mathbf{50}, \mathbf{10})^{\mathsf{T}}$ vs. $H_a : \mu \neq (\mathbf{4}, \mathbf{50}, \mathbf{10})^{\mathsf{T}}$

at $\alpha = 0.10$.

```
alpha <- 0.10; n <- nrow(sweat); p <- ncol(sweat);
crit <- critical_value(n, p, alpha)
T2 <- t2_stat(sweat, mu = c(4,50,10))
```

Since T^2=9.739 $>$ 8.173 $= \frac{(n-1)p}{(n-p)} F_{3,17}(0.10)$, we reject H_0 in favor of H_a at the 10% significance level.

95% T^2 confidence intervals for μ_1, μ_2, μ_3

Using Result 5.3, we have the following code:

```
t2_ci <- function(a, dat, alpha= 0.05) {
  X <- as.matrix(dat)
  Xbar <- colMeans(X)
  n <- nrow(X)
  p <- ncol(X)
  crit <- critical_value(n, p, alpha)
  center <- a %*% Xbar
  me <- sqrt(crit* t(a) %*% cov(X) %*% a / n)
  c(lower = center - me, upper = center + me)
}
```

95% T^2 confidence intervals for μ_1, μ_2, μ_3

```
sci_t2 <- t(apply(X = diag(p), MARGIN = 1,
                  FUN = t2_ci,
                  dat = sweat, alpha = 0.05))
rownames(sci_t2) <- colnames(sweat)
kable(sci_t2,
      caption = "95% $T^2$ confidence intervals for sweat data")
```

Table 1: 95% T^2 confidence intervals for sweat data

	lower	upper
rate	3.397768	5.882232
sodium	35.052408	55.747592
potassium	8.570664	11.359336

Discussion

- Confidence interval methods using the $t_{n-1}(\alpha/2)$ critical value found in the univariate case do not provide simultaneous coverage of the means when considering multiple confidence intervals. For example, the coordinate-confidence intervals for each $\mu_i, i = 1, \ldots, p$ are not simultaneously covered in this naive framework; see formula 5-21 when $a = (0_1, \ldots, 1_i, \ldots, 0_p)^T$ for the ith coordinate.
- To obtain simultaneous confidence statements for arbitrary linear combinations of the mean, we can use a larger critical value that corresponds to the T^2 confidence intervals (see formula 5-23, result 4.3). This is a very powerful tool that can be used under "data-snooping" contexts.
- But need we pay the price of much wider confidence intervals under T^2 or can we do better? Bonferroni can produce more efficient (i.e. tighter) simultaneous confidence intervals than T^2 when we pre-specify the contrasts prior to data analysis.

95% Bonferroni confidence intervals for μ_1, μ_2, μ_3

We can abstract the previous confidence interval function into one that can express either result 4.3 or using the Bonferroni correction (5-29).

```
cif <- function(a, dat, alpha,
                method = c("T2", "Bonf"), m = NULL) {
  X <- as.matrix(dat)
  Xbar <- colMeans(X)
  n <- nrow(X)
  if (method == "T2") {
    p <- ncol(X)
    crit <- critical_value(n, p, alpha)
     me <- sqrt(crit*(t(a) %*% cov(X) %*% a ) / n)
  }
  else if (method == "Bonf") {
    crit <- qt(p = 1 - (alpha/(2*m)), df = n - 1)
    me <- crit*sqrt((t(a) %*% cov(X) %*% a ) / n)
  }
  center <- a %*% Xbar

  c(lower = center - me, upper = center + me)
}
```

95% Bonferroni confidence intervals for μ_1, μ_2, μ_3

Note that we need to "pre-specify" the number of confidence intervals to be $m = 3$ in the Bonferroni case. Hint: on part (d) of exercise 5.10, the number of confidence intervals is the total number of confidence intervals from part (a) and from part (b).

95% Bonferroni confidence intervals for μ_1, μ_2, μ_3

How do these Bonferroni intervals compare to the T^2 intervals?

```
sci_bonf <- t(apply(X = diag(p), MARGIN = 1,
                    FUN = cif,
                    dat = sweat, alpha = 0.05,
                    method = "Bonf", m = 3))
```

Bonferroni intervals are smaller in every case.

Table 2: 95% simultaneous confidence intervals for sweat data

	Bonferroni	T^2
rate	(3.644, 5.636)	(3.398, 5.882)
sodium	(37.103, 53.697)	(35.052, 55.748)
potassium	(8.847, 11.083)	(8.571, 11.359)

Exercise 5.19

Read in the stiffness and bending strength measurements of the lumber data.

```
lumber <- read.table(file = file.path(datadir, "T5-11.DAT"),
                     col.names = c("stiffness", "bending_strength"

head(lumber)
```

```
##   stiffness bending_strength
## 1      1232             4175
## 2      1115             6652
## 3      2205             7612
## 4      1897            10914
## 5      1932            10850
## 6      1612             7627
```

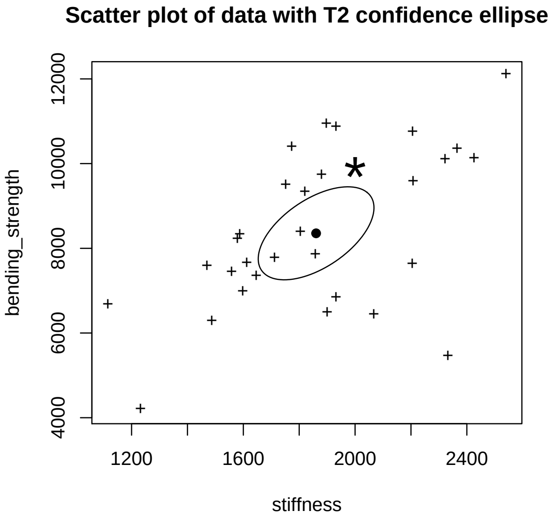
Confidence Ellipse: Exercise 5.19 (a,b)

Define a function to plot the confidence ellipse and overlay with data points and the proposed mean, μ_0.

```
library(ellipse)
ci_ellipse <- function(pw, dat, alpha, mu0) {
  X <- as.matrix(dat); X <- X[,pw];
  Xbar <- colMeans(X)
  crit <- critical_value(n = nrow(X), p = ncol(X),
                         alpha = alpha)
  ell_points <- ellipse(x = cov(X), centre = Xbar,
                        t = sqrt(crit/n))
  plot(ell_points,
       type = "l",
       main = "Scatter plot of data with T2 confidence ellipse",
       ylim = c(min(X[,2]), max(X[,2])),
       xlim  = c(min(X[,1]), max(X[,1])))
  #center of ellipse
  points(Xbar[1], Xbar[2], pch = 19)
  #proposed mu0
  points(mu0[1], mu0[2], pch = "*", cex = 3)
  #data
```

Confidence Ellipse Exercise 5.19 (a,b)

Legend: data points with $+$, the μ_0 with $*$ and the center of the ellipse with a solid black dot.



Exercise 5.19 (b)

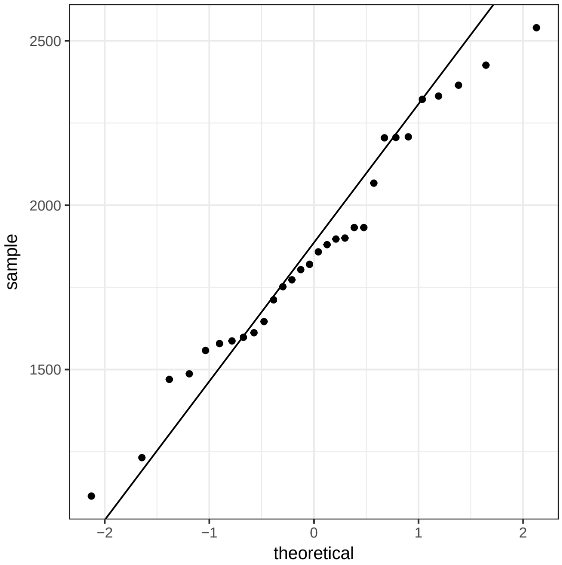
From the previous slide we see that the proposed value, μ_0, is not within the confidence ellipse. Thus, μ_0 does not contain plausible values for the mean stiffness and mean bending strength of the lumber. Compare this visual method with a numerical method (i.e. Hotelling T^2 hypothesis testing, e.g. example 5.4) for testing whether the μ_0 is contained in the confidence ellipse.

Exercise 5.19 (c)

Some of the following code like `qqplot.data` is hidden from view in the .pdf. Please look at the .R file for its definition.

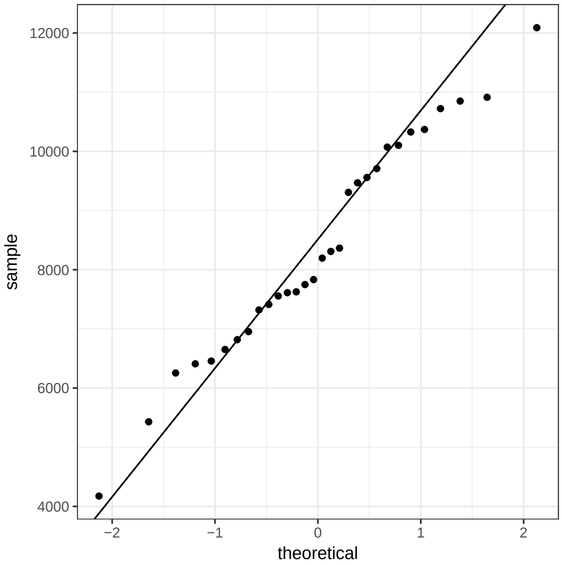
Exercise 5.19 (c) stiffness is sufficiently normal

```
qqplot.data(lumber$stiffness)
```



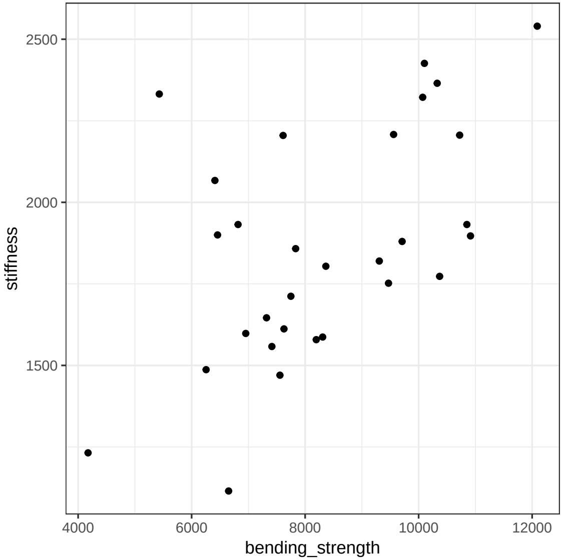
Exercise 5.19 (c) bending strength is sufficiently normal

```
qqplot.data(lumber$bending_strength)
```



Exercise 5.19 (c) Scatter plot is sufficiently elliptical

No strong indication that bi-variate normal is grossly violated.



Exercise 5.19 (c) discussion

Data will almost always exhibit some deviations from normality. The `Q-Q` plots are limited diagnostics, but give us some indication that there are not gros violations of the normality assumption because the sample quantiles match fairly closely with the theoretical quantiles. With the scatter plots, we are looking for some approximation, how ever rough, of an ellipse. This is more judgement and experience about when non-normality is so strong that the interval estimation will be mis-specified for small $n - p$. For large enough $n - p$, we can overcome departures from normality through a χ^2 approximation.

Exercise 5.19 95% Confidence intervals

```
p <- ncol(lumber)
n <- nrow(lumber)
ci_bonf <- t(apply(X = diag(p), MARGIN = 1,
                   FUN = cif,
                   dat = lumber, alpha = 0.05,
                   method = "Bonf", m = 3))
ci_t2 <- t(apply(X = diag(p), MARGIN = 1,
                 FUN = cif,
                 dat = lumber, alpha = 0.05,
                 method = "T2", m = 3))
```

Table 3: 95% simultaneous confidence intervals for lumber data

	Bonferroni	T^2
stiffness	(1697.106, 2023.894)	(1691.347, 2029.653)
bending_strength	(7487.944, 9220.323)	(7457.413, 9250.854)

Visualize Confidence Intervals

We can compare the Bonferroni intervals with the simultaneous T^2 intervals. We write an R function, `ci_compare`, found in the `.R` script version of this file to do this. Note that if two confidence intervals being compared are linear combinations of the original data, then the `ci_compare` function accepts a matrix XA that is the linear combinations applied to the data matrix. For example, if $n = 23, p = 4$ and one inteval was for $\mu_2 - \mu_1$ and the other was $\mu_4 - \mu_3$, then the matrix

$$A = \begin{bmatrix} -1 & 0 \\ 1 & 0 \\ 0 & -1 \\ 0 & 1 \end{bmatrix}$$

would be used to create the right input to the function.

```
XA <- X %*% A
```

But the dimensions $n = 23, p = 4$ of the original data matrix are still passed to the function `ci_compare` to make the right ellipse length. The following example has no special linear combinations, so it is rather straightforward.

Visualize Confidence Intervals

```
ci_compare(X = as.matrix(lumber), n = n, p = p, alpha = 0.05,
           bonf_rect = ci_bonf, t2_rect = ci_t2,
           xlab = "Stiffness", ylab = "Bending Strength")
```