Summary statistics by groups

August 25, 2017

When dealing with grouped data, you will often want to have various summary statistics computed within groups; for example, a table of means and standard deviations.
. you can use *tapply*
Here is an example concerning the folate concentration in red blood cells according to three types of ventilation during anesthesia. data utils

```
 attach(red.cell.folate)
 tapply(folate,ventilation,mean)

 N2O+O2,24h     N2O+O2,op   O2,24h
316.6250        256.4444   278.0000

> tapply(folate,ventilation,sd)
N2O+O2,24h   N2O+O2,op    O2,24h
58.71709      37.12180    33.75648

> tapply(folate,ventilation,length)
N2O+O2,24h    N2O+O2,op     O2,24h
    8           9              5
```

A nicer display:

```
> xbar <- tapply(folate, ventilation, mean)
> s <- tapply(folate, ventilation, sd)
> n <- tapply(folate, ventilation, length)
> cbind(mean=xbar, std.dev=s, n=n)
              mean     std.dev n
N2O+O2,24h   316.6250 58.71709 8
N2O+O2,op    256.4444 37.12180 9
O2,24h       278.0000 33.75648 5
```

The functions *aggregate* is useful for presenting many variables at once;

```
> aggregate(juul[c("age","igf1")], list(sex=juul$sex), mean, na.rm=T)
 sex age        igf1
1 M 15.38436 310.8866
2 F 14.84363 368.1006
```

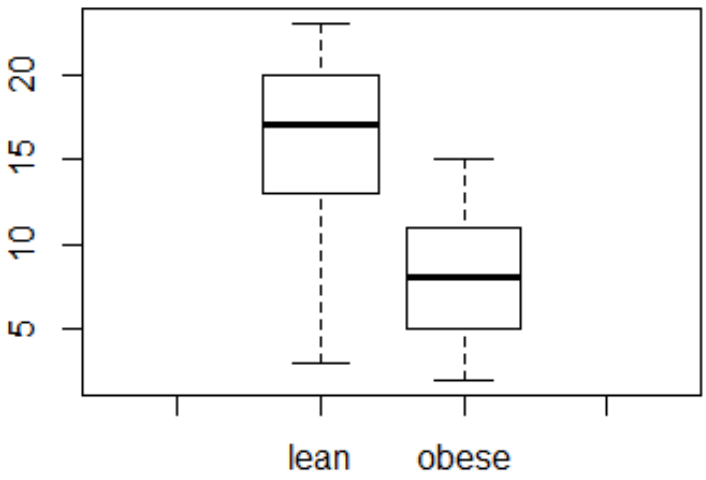

Figure 1: Parallel boxplot.

0.1 Parallel boxplots

```
> attach(energy)
> expend.lean <- expend[stature=="lean"]
> expend.obese <- expend[stature=="obese"]

> boxplot(expend ~ stature)
```

0.2 Generating tables

We deal mainly with two-way tables. A two-way table can be entered as a matrix object . Altman (1991, p. 242) contains an example on caffeine consumption by marital status among women giving birth. That table may be input as follows:

```
> caff.marital <- matrix(c(652,1537,598,242,36,46,38,21,218,327,106,67), nrow=3,byrow=T)
> caff.marital
[,1] [,2] [,3] [,4]
[1,]  652 1537  598  242
[2,]   36   46   38   21
[3,]  218  327  106   67
```

To get readable printouts, you can add row and column names to the matrices.

```
> colnames(caff.marital) <- c("0","1-150","151-300",">300")
>   rownames(caff.marital) <- c("Married","Prev.married","Single")
```

```
>  caff.marital
              0  1-150  151-300 >300
Married      652  1537     598  242
Prev.married  36    46      38   21
Single       218   327     106   67
```

Furthermore, you can name the row and column names as follows. This is particularly useful if you are generating many tables with similar classification criteria.

```
> names(dimnames(caff.marital)) <- c("marital","consumption")
>   caff.marital
                     consumption
marital           0 1-150 151-300 >300
  Married       652  1537     598  242
  Prev.married   36    46      38   21
  Single        218   327     106   67
```

Like any matrix, a table can be transposed with the t function:

```
> t(caff.marital)
           marital
consumption Married Prev.married Single
0             652           36    218
1-150        1537           46    327
151-300       598           38    106
>300          242           21     67
```

1 One- and two-sample tests

Some of the most basic statistical tests deal with comparing continuous data either between two groups or against an a priori stipulated value.

Two functions are introduced here, namely *t.test* and *wilcox.test* for t tests and Wilcoxon tests, respectively. Both can be applied to one and two-sample problems as well as paired data. Notice that the twosample Wilcoxon test is the same as the one called the MannWhitney test in many textbooks.

1.1 One-sample t test

The t tests are based on an assumption that data come from the normal distribution.
In the one-sample case we thus have data $x_1, ..., x_n$ assumed to be independent realizations of random variables with distribution $N(\mu, \sigma^2)$, which denotes the normal distribution with mean μ and variance σ^2, and we wish to test the null hypothesis that $\mu = \mu_0$. We can estimate the parameters μ and σ by the empirical mean $\bar{x}$ and standard deviation s, although we must realize that we could never pinpoint their values exactly.

Example:
Here is an example concerning daily energy intake in kJ for 11 women (Altman, 1991, p. 183).

```
daily.intake <- c(5260,5470,5640,6180,6390,6515, 6805,7515,7515,8230,8770)
```

Assuming that data come from a normal distribution, the object is to test whether this distribution might have mean $\mu = 7725$. This is done with *t.test* as follows:

```
?t.test
> hist(daily.intake)              ####normality assumption
> ?shapiro.test
> shapiro.test(daily.intake)

Shapiro-Wilk normality test

data:  daily.intake
W = 0.95237, p-value = 0.6743

> t.test(daily.intake,mu=7725)         ###t-test

One Sample t-test

data:  daily.intake
t = -2.8208, df = 10, p-value = 0.01814
alternative hypothesis: true mean is not equal to 7725
95 percent confidence interval:
5986.348 7520.925
sample estimates:
mean of x
6753.636
```

2 Wilcoxon signed-rank test

The t tests are fairly robust against departures from the normal distribution especially in larger samples,but sometimes you wish to avoid making that assumption. To this end, the distribution-free methods are convenient. These are generally obtained by replacing data with corresponding order statistics. For the $one-sampleWilcoxon$ test, the procedure is to subtract the theoretical 0 and rank the differences according to their numerical value, ignoring the sign, and then calculate the sum of the positive or negative ranks. The point is that, assuming only that the distribution is symmetric around 0, the test statistic corresponds to selecting each number from 1 to n with probability 1/2 and calculating the sum.
Practical application of the Wilcoxon signed-rank test is done almost exactly like the t test:

```
> wilcox.test(daily.intake, mu=7725)

Wilcoxon signed rank test with continuity correction

data:  daily.intake
V = 8, p-value = 0.0293
alternative hypothesis: true location is not equal to 7725

Warning message:
```

```
In wilcox.test.default(daily.intake, mu = 7725) :
cannot compute exact p-value with ties
```

If the model assumptions of the parametric test are fulfilled, then it will be somewhat more efficient,on the order of 5% percent in large samples,although the difference can be larger in small samples.Notice,for instance,that unless the sample size is 6 or above,the signed-rank test simply cannot become significant at the 5%level.This is probably not too important,though;what is more important is that the apparent lack of assumptions for these tests sometimes misleads people into using them for data where the observations are not independent or where a comparison is biased by an important covariate. The Wilcoxon tests are susceptible to the problem of *ties*,where several observations share the same value.In such cases,you simply use the average of the tied ranks; for example, if there are four identical values corresponding to places 6 to 9, they will all be assigned the value 7.5.

3 R-Code

```
mid.age <- c(2.5,7.5,13,16.5,17.5,19,22.5,44.5,70.5)
acc.count <- c(28,46,58,20,31,64,149,316,103)
age.acc <- rep(mid.age,acc.count)
brk <- c(0,5,10,16,17,18,20,25,60,80)
hist(age.acc,breaks=brk)
n <- length(x)
plot(sort(x),(1:n)/n,type="s",ylim=c(0,1))
plot(sort(x),(1:n)/n,ylim=c(0,1))
qqnorm(x)
data("juul")
par(mfrow=c(1,2))
boxplot( juul$tanner )
boxplot(log( juul$tanner ))
par(mfrow=c(1,2))
############################chp2
attach(energy)
expend.lean <- expend[stature=="lean"]
expend.obese <- expend[stature=="obese"]
boxplot(expend ~ stature)                        ###boxplot
#######################################       Generating Table

caff.marital <- matrix(c(652,1537,598,242,36,46,38,21,218,327,106,67), nrow=3,byrow=T)
caff.marital

colnames(caff.marital) <- c("0","1-150","151-300",">300")       ##### name rows and columns
rownames(caff.marital) <- c("Married","Prev.married","Single")
caff.marital

names(dimnames(caff.marital)) <- c("marital","consumption")
caff.marital
```

```
t(caff.marital)                                          ######transpose
par(mfrow=c(1,2))
barplot(caff.marital, col="white")
barplot(t(caff.marital), col="white")
par(mfrow=c(1,1))
################################################################ One-sample t test
daily.intake <- c(5260,5470,5640,6180,6390,6515, 6805,7515,7515,8230,8770)
summary(daily.intake)
?t.test                                  ###################help on t.test
hist(daily.intake)                       ################### checking the normality assumpti
?shapiro.test
shapiro.test(daily.intake)
t_test<-t.test(daily.intake,mu=7725)
############################################################################ Wilcoxon-ranked sum
?wilcox.test
wilcox.test(daily.intake, mu=7725)
```