Comparing two treatments

christophe@pallier.org

In the previous section, the data from the two groups were assumed to be independent. If there is some pairing, for example if data were acquired in the same unit under two conditions, then the data are not independent. The simplest way to perform the data analysis is to examine the differences between the two conditions computed over each unit.

Here data come organized a long table format with one measure per row, and condition and subject as variables. This less convenient to compute the differences within subjects than a short format with one subject per row, and one column per condition, but better to run linear model. To convert from one representation to the other, see stack, reshape2, plyr...

```
tc <- read.csv("twotreat.csv")
head(tc)
```

```
##   X sub cond         y
## 1 1  s1    1  9.487949
## 2 2  s1    2  9.594552
## 3 3  s2    1  9.554275
## 4 4  s2    2 12.278365
## 5 5  s3    1  9.843633
## 6 6  s3    2 10.460506
```

```
str(tc)
```

```
## 'data.frame':    40 obs. of  4 variables:
##  $ X   : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ sub : Factor w/ 20 levels "s1","s10","s11",..: 1 1 12 12 14 14 15 15 16 16 ...
##  $ cond: int  1 2 1 2 1 2 1 2 1 2 ...
##  $ y   : num  9.49 9.59 9.55 12.28 9.84 ...
```

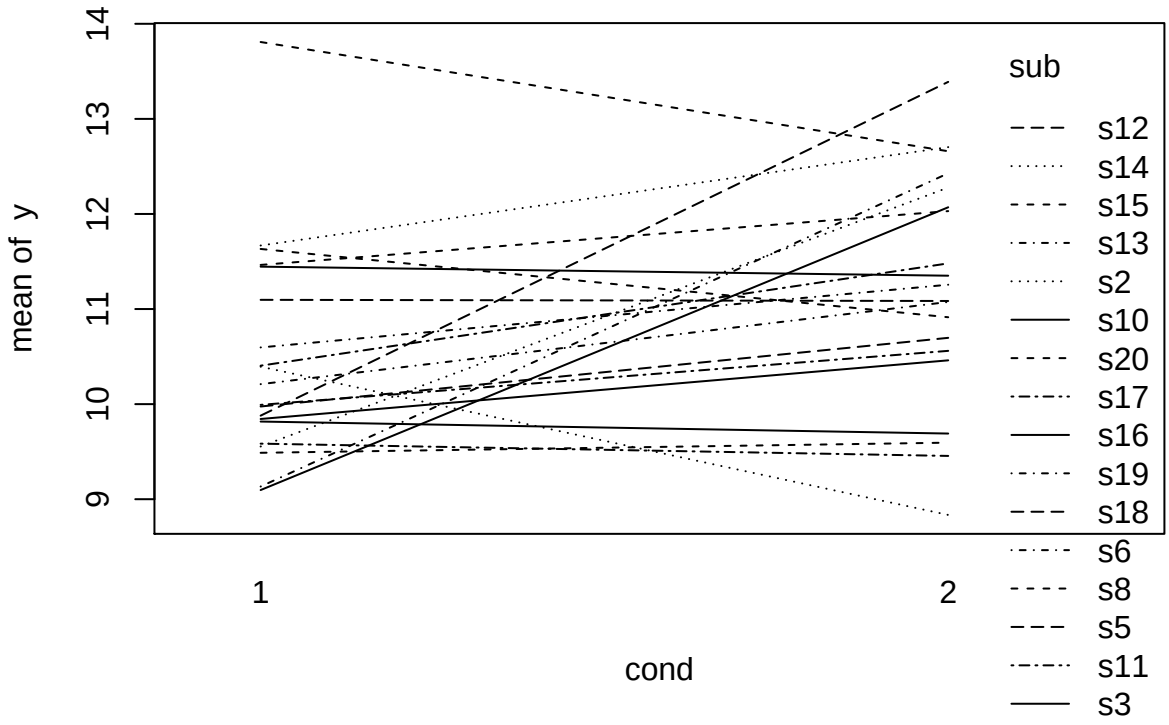
```
tc$sub <- factor(tc$sub) # make sure these vars are factors
tc$cond <- factor(tc$cond)
table(tc$sub)
```

```
##
##  s1 s10 s11 s12 s13 s14 s15 s16 s17 s18 s19  s2 s20  s3  s4  s5  s6  s7
##   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2
##  s8  s9
##   2   2
```

(I assume that thereare no repeated measures within subject and treatment. If this is the case with your dataset, use aggregate or melt)

Graphical explorations

```
with(tc, interaction.plot(cond, sub, y))
```

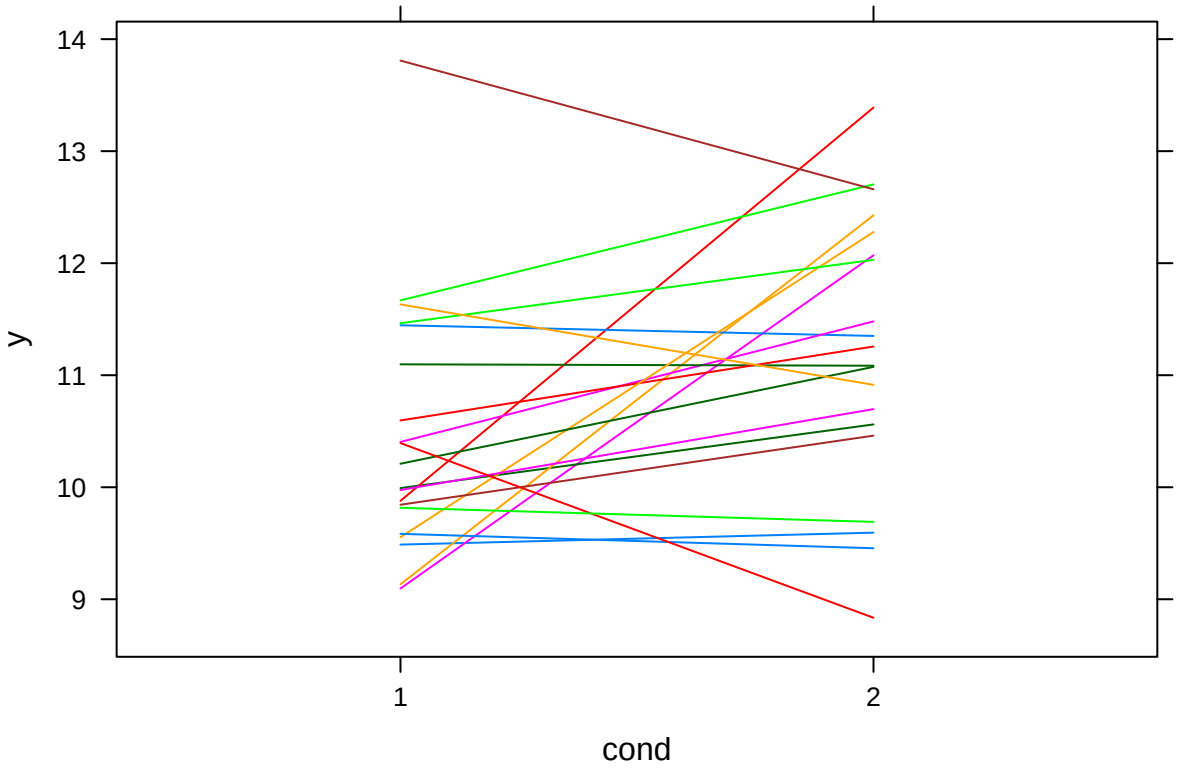


Fancier graphs can be obtained with lattice:

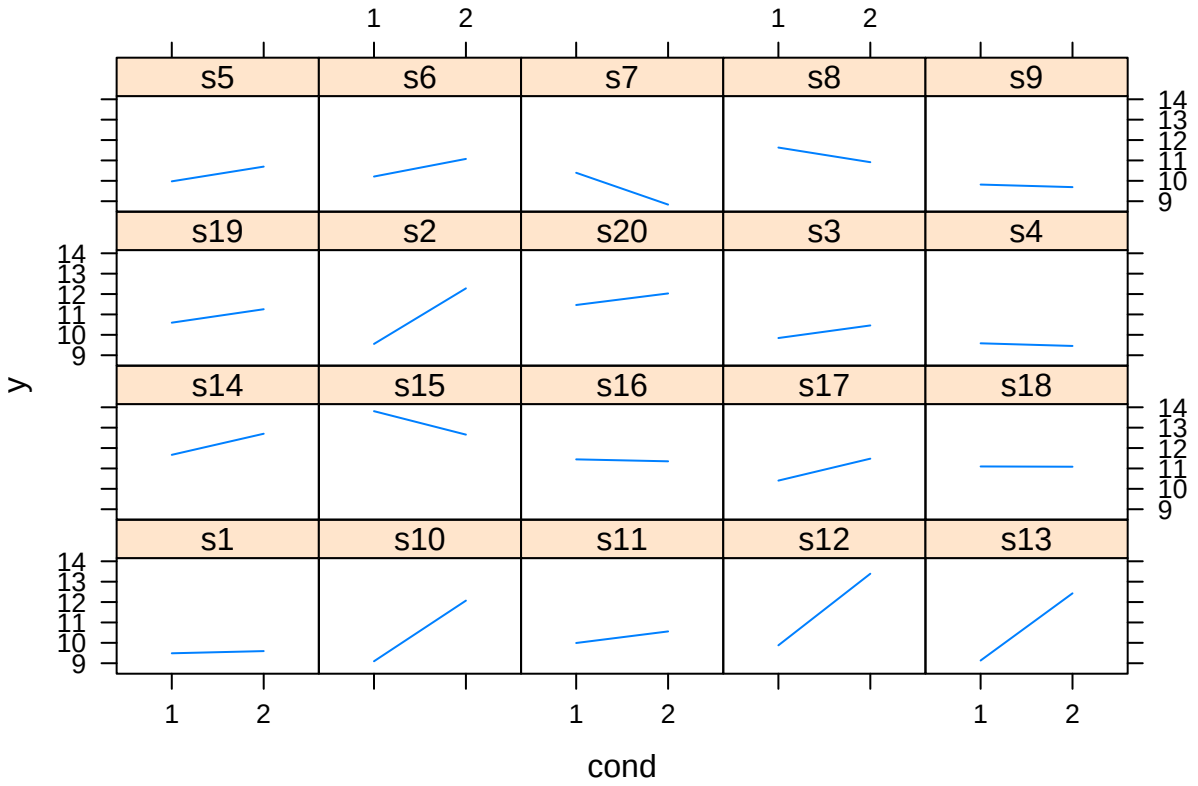
```
require(lattice)
```

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

```
xyplot(y ~ cond, group=sub, data=tc, type='l')
```

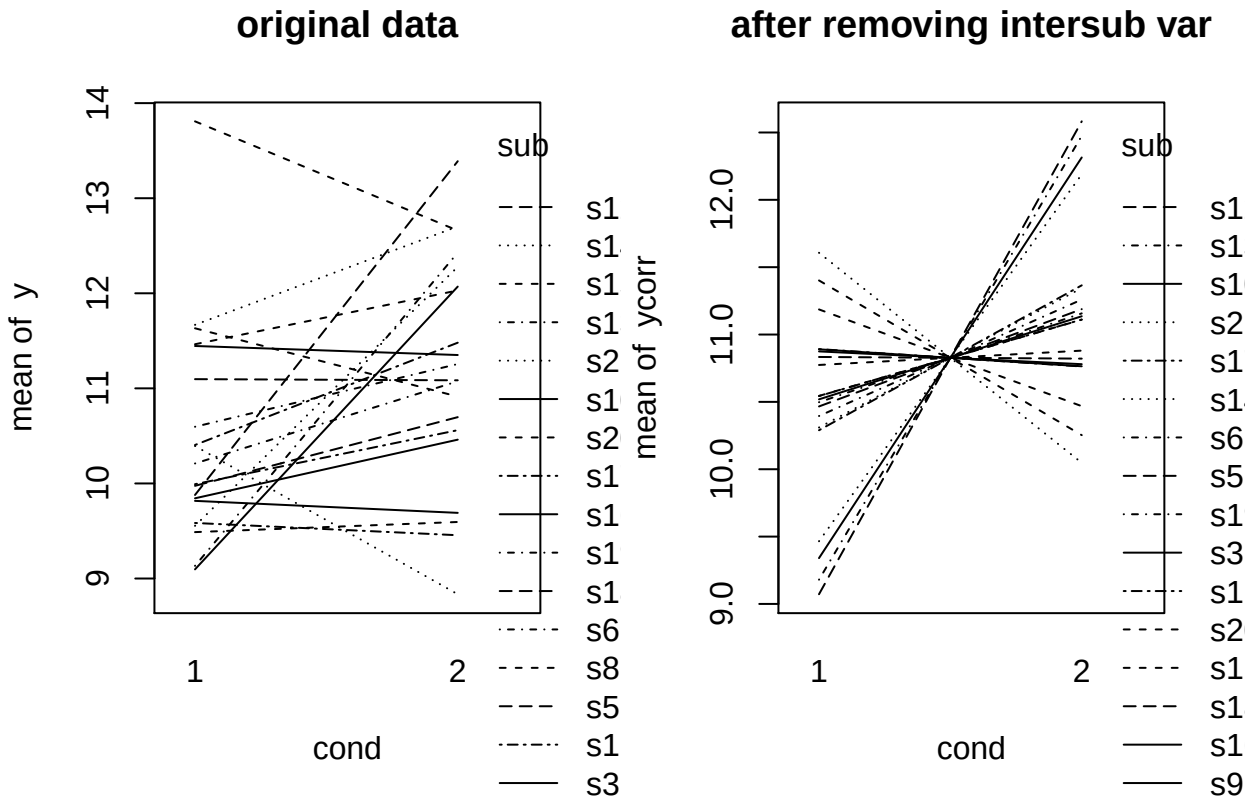


```
xyplot(y ~ cond | sub, data=tc, type='l')
```



We can also remove to main effects of subjects, as we are interested in the difference between condition within subjects:

```
attach(tc)
tc$ycorr <- y + mean(y) - tapply(y, sub, mean)[sub]
detach(tc)
attach(tc)
par(mfcol=c(1,2))
interaction.plot(cond, sub, y, main='original data')
interaction.plot(cond, sub, ycorr, main='after removing intersub var')
```




```
par(mfcol=c(1,1))
detach(tc)
```

Descriptive stats

```
with(tc, signif(tapply(y, cond, mean)))
```

```
##       1       2
## 10.4541 11.2006
```

```
# compute differences
c1 <- levels(tc$cond)[1]
c2 <- levels(tc$cond)[2]

s1 <- tc$sub[tc$cond==c1]
y1 <- tc$y[tc$cond==c1][order(s1)]

s2 <- tc$sub[tc$cond==c2]
y2 <- tc$y[tc$cond==c2][order(s2)]

summary(y1-y2)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -3.5106 -1.0451 -0.5926 -0.7465  0.1028  1.5595
```

```
se(y1-y2) # standard error of the effect
```

```
## [1] 0.3144635
```

```r
# Check if the pairing was useful?
cor.test(y1, y2)
```

```
##
##  Pearson's product-moment correlation
##
## data:  y1 and y2
## t = 1.2601, df = 18, p-value = 0.2237
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  -0.1805511  0.6458640
## sample estimates:
##      cor
## 0.284718
```

Inferential stats

```r
t.test(y1, y2, paired=T)
```

```
##
##  Paired t-test
##
## data:  y1 and y2
## t = -2.3737, df = 19, p-value = 0.02831
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -1.40463520 -0.08827584
## sample estimates:
## mean of the differences
##              -0.7464555
```

Linear model approach

```r
(sm <- summary(model_lm <- lm(y ~ cond + sub, data=tc)))
```

```
##
## Call:
## lm(formula = y ~ cond + sub, data = tc)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.3821 -0.3897  0.0000  0.3897  1.3821
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  9.16802    0.72053  12.724 9.57e-11 ***
## cond2        0.74646    0.31446   2.374  0.02831 *
## subs10       1.04209    0.99442   1.048  0.30782
## subs11       0.73505    0.99442   0.739  0.46883
## subs12       2.09273    0.99442   2.104  0.04888 *
## subs13       1.23933    0.99442   1.246  0.22781
## subs14       2.64426    0.99442   2.659  0.01550 *
## subs15       3.69282    0.99442   3.714  0.00147 **
## subs16       1.85698    0.99442   1.867  0.07735 .
## subs17       1.40099    0.99442   1.409  0.17504
```

```
## subs18       1.54960    0.99442   1.558  0.13566
## subs19       1.38488    0.99442   1.393  0.17981
## subs2        1.37507    0.99442   1.383  0.18277
## subs20       2.20568    0.99442   2.218  0.03894 *
## subs3        0.61082    0.99442   0.614  0.54634
## subs4       -0.02161    0.99442  -0.022  0.98289
## subs5        0.79484    0.99442   0.799  0.43399
## subs6        1.10091    0.99442   1.107  0.28207
## subs7        0.07310    0.99442   0.074  0.94217
## subs8        1.73165    0.99442   1.741  0.09779 .
## subs9        0.21238    0.99442   0.214  0.83315
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9944 on 19 degrees of freedom
## Multiple R-squared:  0.6763, Adjusted R-squared:  0.3356
## F-statistic: 1.985 on 20 and 19 DF,  p-value: 0.07052
```

```
(diff <-sm$coefficients[2,'Estimate'])
```

```
## [1] 0.7464555
```

```
(diffse <- sm$coefficients[2,'Std. Error'])
```

```
## [1] 0.3144635
```

In this simple situation, mixed effect models will yield the same p-values:

```
require(nlme)
```

```
## Loading required package: nlme
```

```
(model_lme <- lme(y ~ cond, random=~1|sub, data= tc))
```

```
## Linear mixed-effects model fit by REML
##   Data: tc
##   Log-restricted-likelihood: -62.24936
##   Fixed: y ~ cond
## (Intercept)       cond2
##  10.4541013   0.7464555
##
## Random effects:
##  Formula: ~1 | sub
##         (Intercept)  Residual
## StdDev:   0.6261402 0.9944209
##
## Number of Observations: 40
## Number of Groups: 20
```

```
summary(model_lme)
```

```
## Linear mixed-effects model fit by REML
##  Data: tc
##        AIC      BIC    logLik
##   132.4987 139.0491 -62.24936
##
## Random effects:
##  Formula: ~1 | sub
```

```
##         (Intercept)  Residual
## StdDev:   0.6261402 0.9944209
##
## Fixed effects: y ~ cond
##                 Value Std.Error DF  t-value p-value
## (Intercept) 10.454101 0.2627665 19 39.78476  0.0000
## cond2        0.746456 0.3144635 19  2.37374  0.0283
##  Correlation:
##       (Intr)
## cond2 -0.598
##
## Standardized Within-Group Residuals:
##        Min         Q1        Med         Q3        Max
## -1.8397652 -0.4106805 -0.1330625  0.5485781  2.3027841
##
## Number of Observations: 40
## Number of Groups: 20
```

```
# plot(ranef(model_lme))
# plot(res_lme <- residuals(model_lme))
# qqnorm(res_lme)
# qqline(res_lme)
# plot(model_lme)
```

```
require(lme4)
```

```
## Loading required package: lme4
```

```
## Loading required package: Matrix
```

```
##
## Attaching package: 'lme4'
```

```
## The following object is masked from 'package:nlme':
##
##     lmList
```

```
(model_lmer <- lmer(y ~ cond + (1|sub), data= tc))
```

```
## Linear mixed model fit by REML ['lmerMod']
## Formula: y ~ cond + (1 | sub)
##    Data: tc
## REML criterion at convergence: 124.4987
## Random effects:
##  Groups   Name        Std.Dev.
##  sub      (Intercept) 0.6261
##  Residual             0.9944
## Number of obs: 40, groups:  sub, 20
## Fixed Effects:
## (Intercept)        cond2
##     10.4541       0.7465
```

```
summary(model_lmer)
```

```
## Linear mixed model fit by REML ['lmerMod']
## Formula: y ~ cond + (1 | sub)
##    Data: tc
##
```

```
## REML criterion at convergence: 124.5
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -1.8398 -0.4107 -0.1331  0.5486  2.3028
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  sub      (Intercept) 0.3921   0.6261
##  Residual             0.9889   0.9944
## Number of obs: 40, groups:  sub, 20
##
## Fixed effects:
##             Estimate Std. Error t value
## (Intercept)  10.4541     0.2628  39.785
## cond2         0.7465     0.3145   2.374
##
## Correlation of Fixed Effects:
##       (Intr)
## cond2 -0.598
```

```
# qqmath(ranef(model_lmer))
```

See http://freshbiostats.wordpress.com/2013/07/28/mixed-models-in-r-lme4-nlme-both/

Bootstrap confidence interval for the difference

```
require(boot)
```

```
## Loading required package: boot
```

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

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

```
samplemean <- function(x, d) { mean(x[d]) }
b <- boot(y1-y2, samplemean, 1000)
boot.ci(b)
```

```
## Warning in boot.ci(b): bootstrap variances needed for studentized intervals
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = b)
##
## Intervals :
## Level      Normal              Basic
## 95%   (-1.3412, -0.1136 )   (-1.3009, -0.0782 )
##
## Level     Percentile            BCa
## 95%   (-1.4147, -0.1920 )   (-1.4530, -0.2087 )
## Calculations and Intervals on Original Scale
```

Plots

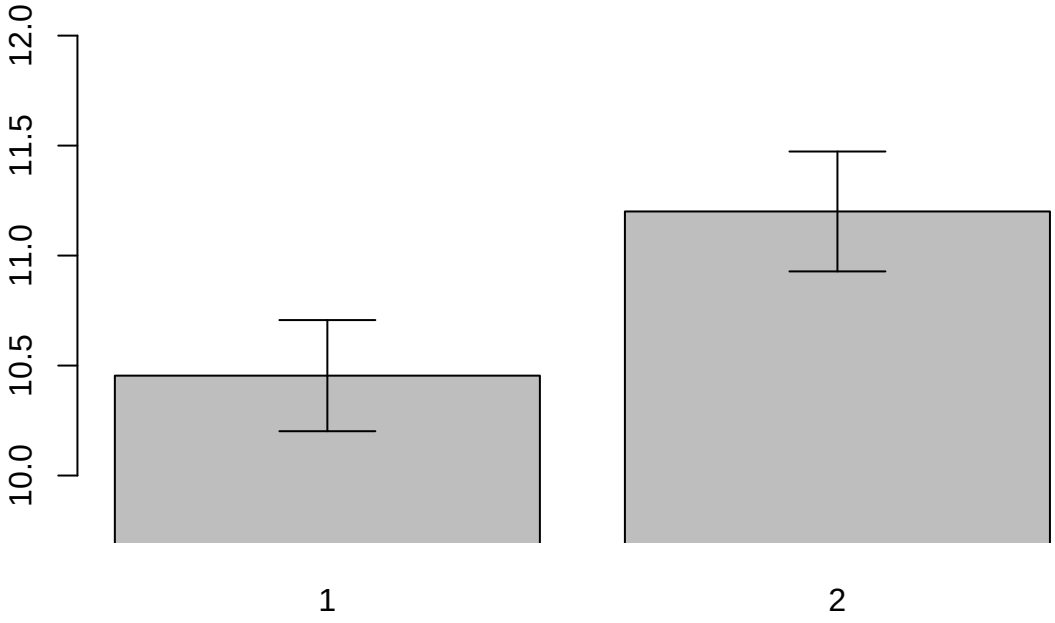
The errors bars can either represent the standard errors (or confidence intervals) of the means of each treatment, *or* the standard error bar for the difference between the two treatments when intersubject variability is taken out.

First graphics: with the std.err. of the means:

```
attach(tc)
par(mfrow=c(1,1))

means <- tapply(y, cond, mean)
(ses <- tapply(y, cond, se))
```

```
##         1         2
## 0.2526511 0.2725067
```

```
ysca = c(min(means-3*ses), max(means+3*ses))

mp <- barplot(means, ylim=ysca, xpd=F)
arrows(mp, means-ses,
       mp, means+ses,
       code=3, angle=90)
```



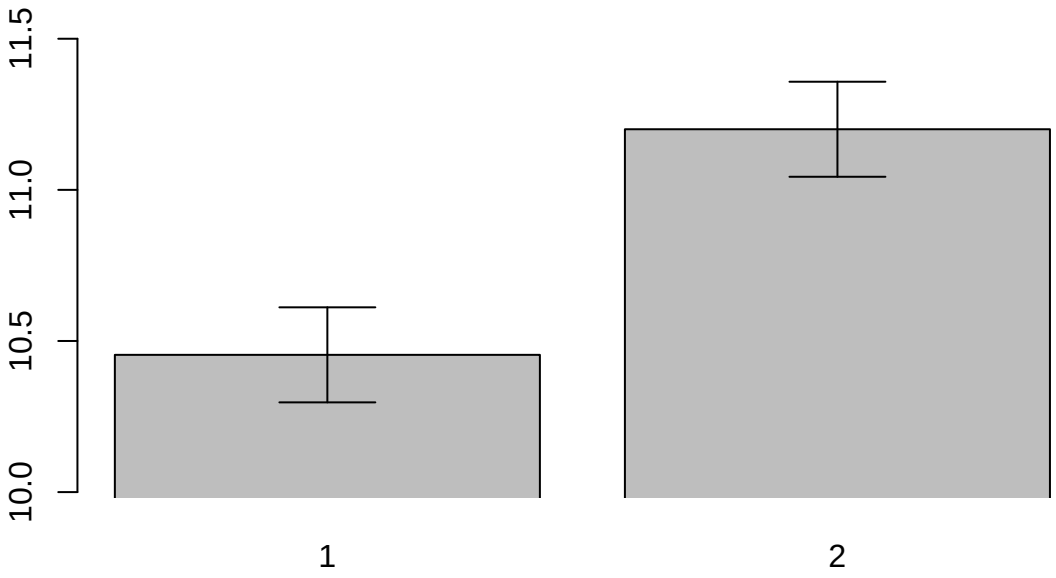

```
detach(tc)
```

If we remove the between Ss variability

```
attach(tc)
par(mfrow=c(1,1))

means <- tapply(y, cond, mean)
(ses <- tapply(ycorr, cond, se))
```

```
##         1         2
## 0.1572318 0.1572318
```

```
ysca = c(min(means-3*ses), max(means+3*ses))

mp <- barplot(means, ylim=ysca, xpd=F)
arrows(mp, means-ses,
       mp, means+ses,
       code=3, angle=90)
```




```
detach(tc)
```

If we take the standard error from the regression:

```
(sm <- summary(model_lm <- lm(y ~ cond + sub, data=tc)))
```

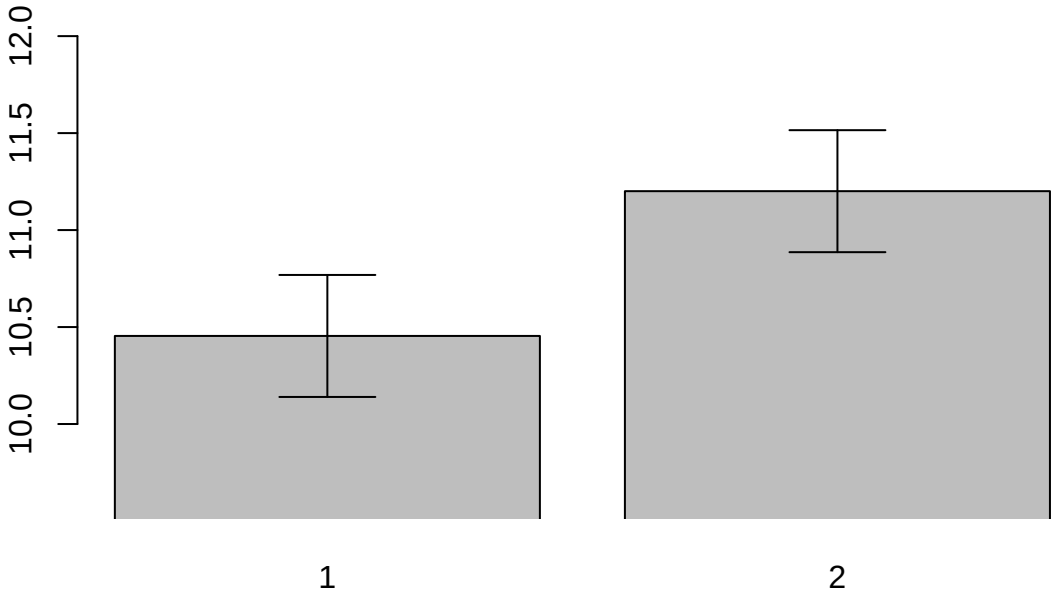
```
##
## Call:
## lm(formula = y ~ cond + sub, data = tc)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.3821 -0.3897  0.0000  0.3897  1.3821
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  9.16802    0.72053  12.724 9.57e-11 ***
## cond2        0.74646    0.31446   2.374  0.02831 *
## subs10       1.04209    0.99442   1.048  0.30782
## subs11       0.73505    0.99442   0.739  0.46883
## subs12       2.09273    0.99442   2.104  0.04888 *
## subs13       1.23933    0.99442   1.246  0.22781
## subs14       2.64426    0.99442   2.659  0.01550 *
## subs15       3.69282    0.99442   3.714  0.00147 **
## subs16       1.85698    0.99442   1.867  0.07735 .
## subs17       1.40099    0.99442   1.409  0.17504
## subs18       1.54960    0.99442   1.558  0.13566
## subs19       1.38488    0.99442   1.393  0.17981
## subs2        1.37507    0.99442   1.383  0.18277
## subs20       2.20568    0.99442   2.218  0.03894 *
```

```
## subs3        0.61082    0.99442   0.614  0.54634
## subs4       -0.02161    0.99442  -0.022  0.98289
## subs5        0.79484    0.99442   0.799  0.43399
## subs6        1.10091    0.99442   1.107  0.28207
## subs7        0.07310    0.99442   0.074  0.94217
## subs8        1.73165    0.99442   1.741  0.09779 .
## subs9        0.21238    0.99442   0.214  0.83315
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9944 on 19 degrees of freedom
## Multiple R-squared:  0.6763, Adjusted R-squared:  0.3356
## F-statistic: 1.985 on 20 and 19 DF,  p-value: 0.07052
```

```
diff <-sm$coefficients[2,'Estimate']
diffse <- sm$coefficients[2,'Std. Error']

attach(tc)
par(mfrow=c(1,1))

means <- tapply(y, cond, mean)
(ses <- rep(diffse, length(means)))
```

```
## [1] 0.3144635 0.3144635
```

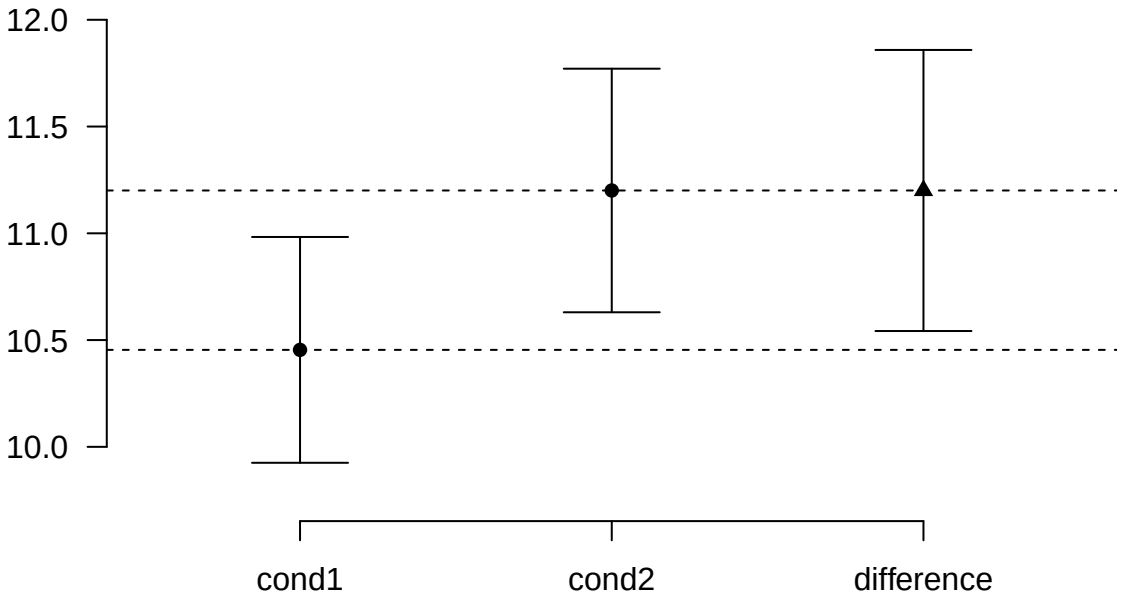
```
ysca = c(min(means-3*ses), max(means+3*ses))

mp <- barplot(means, ylim=ysca, xpd=F)
arrows(mp, means-ses,
       mp, means+ses,
       code=3, angle=90)
```




```
detach(tc)
```

A much nicer plot can be constructed, with confidence intervals for the means and for their difference (Cumming, Geoff, and Sue Finch. 2005. "Inference by Eye: Confidence Intervals and How to Read Pictures of Data." American Psychologist 60 (2): 170–180.)

```
attach(tc)
m1 <- t.test(y[cond==1])$conf.int
m2 <- t.test(y[cond==2])$conf.int
di <- diff(t.test(y1-y2)$conf.int)
ysca <- c(min(c(m1,m2)-0.1*diff(range(c(m1,m2)))),
          max(c(m1,m2)+0.1*diff(range(c(m1,m2)))))

plot(c(Gr1=1, Gr2=2, difference=3),
     c(mean(m1), mean(m2), mean(m2)),
     pch=c(16,16,17), xlim=c(0.5, 3.5), ylim=ysca, axes=F, xlab='', ylab='')
axis(2, las=1)
axis(1,at=1:3,labels=c('cond1','cond2','difference'))
arrows(1:3, c(m1[1], m2[1], mean(m2)-di/2),
       1:3, c(m1[2], m2[2], mean(m2)+di/2),
       code=3, angle=90)
abline(h=mean(m1), lty=2)
abline(h=mean(m2), lty=2)
```




```
detach(tc)
```

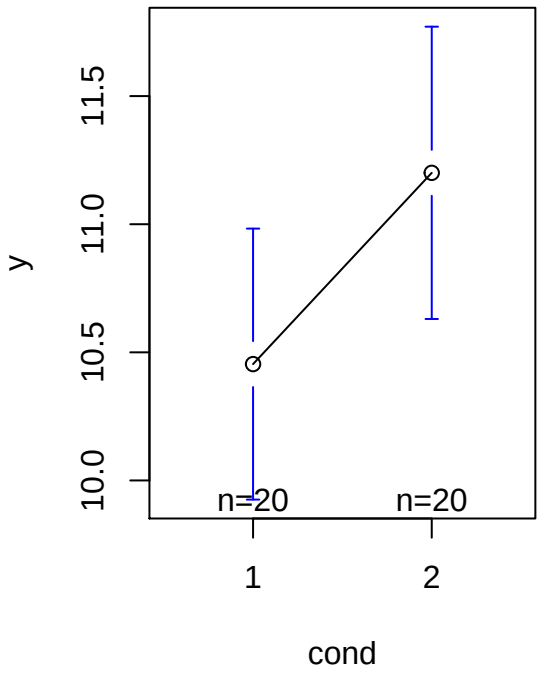
```
require(gplots)
```

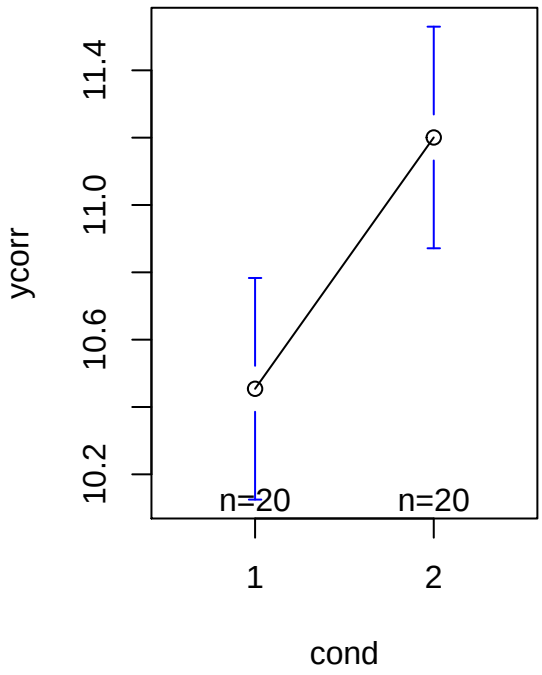
```
## Loading required package: gplots
```

```
##
## Attaching package: 'gplots'
```

```
## The following object is masked from 'package:stats':
##
##     lowess
```

```
par(mfcol=(c(1,2)))
plotmeans(y ~ cond, data=tc)
plotmeans(ycorr ~ cond, data=tc)
```


y
10.0
10.5
11.0
11.5
n=20
n=20
1
2
cond


ycorr
10.2
10.6
11.0
11.4
n=20
n=20
1
2
cond