Getting Results with R

First import the mockdata:

```
>library(readr) # this loads the library that knows how to read files
>mockdata <- read_csv("http://github.org/
                      jon-may/GettingResultsinR/raw/master/mockdata.csv")
```

And tell R to use this data as the default 'data frame' so you do not need to specify it in every command:

```
>attach(mockdata)
```

counting cases

First, count how many males and females were tested, using `table`:

```
>table(Sex)
Sex
 F  M
20 10
```

Report it like this, in the *Participants* section of the *Method*:

"We recruited 30 participants (20 females, 10 males)…"

Descriptive statistics

Find out how old the participants were:

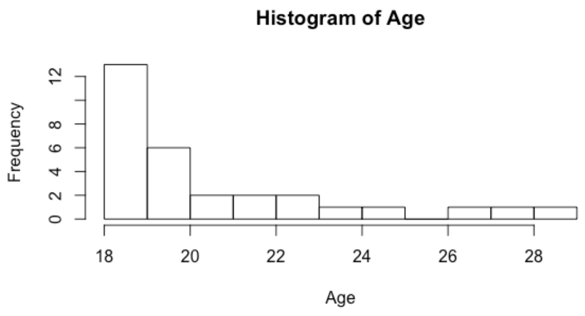
```
>library(psych) # a library that contains handy functions for psychology
>describe(Age)
   vars  n mean   sd median trimmed  mad min max range skew kurtosis   se
X1    1 30 20.9 3.03     20   20.38 1.48  18  29    11 1.26     0.59 0.55
```

"… aged between 18 and 29 (M=20.9, SD=3.0)."

As long as the skew and kurtosis are less than ±2 the data can be treated as normally distributed, but that skew value is a bit high. Draw a histogram to check:

```
>hist(Age) # look in Plot window!
```

This shows why the skew value is so high, but luckily there is no need to do any parametric statistics on this demographic variable.



Chi squared test of association – χ2

`table` can make two-way tables too. For example, how many of each sex were tested by each experimenter:

```
> table(Sex,Experimenter)
   Experimenter
Sex JM LS MK
  F  6  6  8
  M  4  4  2
```

MK didn't seem to test as many men as the others – was this systematic, or just random variation?

Use a Chi Squared test to see if there is an association between Sex and Experimenter, and do so by sending it the results of this `table` command:

```
> chisq.test(table(Sex,Experimenter))

	Pearson's Chi-squared test

data:  table(Sex, Experimenter)
X-squared = 1.2, df = 2, p-value = 0.5488
```

A warning message is also produced, because some of the ‘expected’ cell sizes are less than 5, but there is no association. There is no need to report this, but this is how a χ2 test is reported:

> “There was no difference in the proportion of men and women tested by each experimenter χ^2 (2, N=30)=1.20, p=.549.”

Correlation

The participants have been measured three times, at a baseline session (Value1), after training (Value2) and at a follow-up session (Value3). These can be correlated a pair at a time like this:

```
> cor(Value1,Value2)
[1] 0.3078597
```

With more than two variables, a correlation matrix can be produced by using cor on a data frame or part of a data frame, e.g.,

```
> cor(mockdata[,6:8])
          Value1      Value2      Value3
Value1 1.0000000  0.30785968  0.17061310
Value2 0.3078597  1.00000000 -0.02317446
Value3 0.1706131 -0.02317446  1.00000000
```

A subset of a dataframe

A subset of a data frame is referred to by putting row and column indices in square brackets after the name of the data frame. Here `mockdata[,6:8]` selects all rows (as there is nothing before the comma) and columns 6 to 8 (inclusive). If the indices are not known, they can be found by combining the `which` and `colnames` functions, e.g.:

```
> which(colnames(mockdata)=="Value1")
[1] 6
```

These can be used directly in the `cor` command:

```
> cor(mockdata[,
              which(colnames(mockdata)=="Value1"):
              which(colnames(mockdata)=="Value3")])
```

If the variables that are to be correlated are not adjacent, then this will not work, so a new data frame needs to be built that contains them:

```
> myvals<-mockdata[,c(1,6:8)]
> cor(myvals)
          Subject    Value1      Value2      Value3
Subject 1.0000000 0.2115570  0.26509522  0.52041127
Value1  0.2115570 1.0000000  0.30785968  0.17061310
Value2  0.2650952 0.3078597  1.00000000 -0.02317446
Value3  0.5204113 0.1706131 -0.02317446  1.00000000
```

The `c()` command joins together the values is encloses into a list which the `[,]` then uses to select the columns in the order specified. Here it selects the first column, then columns 6 to 8.

A limitation of `cor` is that it just gives the r value. The `psych` library contains a function `corr.test` (note that there are two rs in its name; there is another function `cor.test`).

```
> corr.test(myvals)
Call:corr.test(x = myvals)
Correlation matrix
        Subject Value1 Value2 Value3
Subject    1.00   0.21   0.27   0.52
Value1     0.21   1.00   0.31   0.17
Value2     0.27   0.31   1.00  -0.02
Value3     0.52   0.17  -0.02   1.00
Sample Size
[1] 30
Probability values (Entries above the diagonal are adjusted for multiple tests.)
        Subject Value1 Value2 Value3
Subject    0.00   0.79   0.63   0.02
Value1     0.26   0.00   0.49   0.79
Value2     0.16   0.10   0.00   0.90
Value3     0.00   0.37   0.90   0.00

 To see confidence intervals of the correlations, print with the short=FALSE option
```

There is nothing much to report here, but this is how it might be reported:

> "The three scores did not correlate with each other (Baseline, After testing: r(30) = .31, p=.49; After testing, Follow-up: r(30)=-.02, p=.90; Baseline, Follow-up: r(30)=.17, p=.79; p values adjusted for multiple comparisons)."

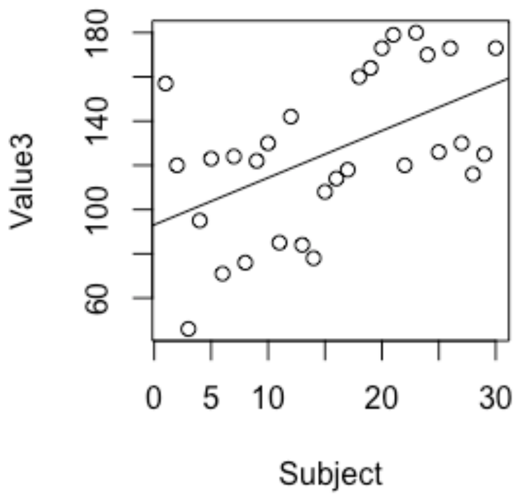
There is a presumably spurious correlation between the Subject and Value3:

> "There was a correlation between Subject Code and participant's score at follow-up r(30)=.52, p=.02"

Scatterplots and regression

To understand a correlation it is always a good idea to draw a scatterplot. The `plot(x,y)` command draws the scatterplot, and then the `abline()` command adds the regression line using the result of a simple linear regression command `lm(y~x)`. Note that the order of x and y are different in these two commands!

```
> plot(Subject,Value3)
> abline(lm(Value3 ~ Subject))
```



The full output from the lm() command looks like this:

```
> summary(lm(Value3 ~ Subject))

Call:
lm(formula = Value3 ~ Subject)

Residuals:
    Min      1Q  Median      3Q     Max 
-53.493 -27.454   1.567  24.302  61.759 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)  93.1149    11.7032   7.956 1.15e-08 ***
Subject       2.1259     0.6592   3.225   0.0032 ** 
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 31.25 on 28 degrees of freedom
Multiple R-squared:  0.2708,	Adjusted R-squared:  0.2448 
F-statistic:  10.4 on 1 and 28 DF,  p-value: 0.003197
```

To check that this is based on the same data as the correlation, the square root of the R^2 value should be .52

```
> sqrt(.2708)
[1] 0.5203845
```

The full regression equation statistics could be reported like this:

> "A simple linear regression model showed that follow-up scores were predicted by Subject code, where Follow-up = 93.11 + Subject x 2.13, R^2=.27, F(1,28)=10.4, p=.003".

Computing and Recoding variables

Existing variables may need to be combined for further analysis:

```
>mockdata$sumOfValues<-Value1 + Value2 + Value3
>mockdata$meanValue <- mockdata$sumOfValues / 3
```

Variables may need to be recoded from raw data into categories:

```
> mockdata$ValueGroup[mockdata$sumOfValues>0]<-"Low"
> mockdata$ValueGroup[mockdata$sumOfValues>250]<-"Medium"
> mockdata$ValueGroup[mockdata$sumOfValues>350]<-"High"
> describeBy(mockdata$sumOfValues,group=mockdata$ValueGroup,mat=TRUE,digits=1)

    item group1 vars  n  mean   sd median trimmed  mad min max range skew kurtosis   se
X11    1   High    1  8 395.9 36.6    394   395.9 51.1 358 444    86  0.2     -1.9 12.9
X12    2    Low    1  5 206.6 32.1    225   206.6 17.8 167 237    70 -0.3     -2.2 14.4
X13    3 Medium    1 17 312.0 25.3    319   313.1 26.7 260 347    87 -0.5     -0.9  6.1
```

> "Participants were divided into groups based on their total score over the three sessions. The Low group (N=5) ranged from167 to 237 (M=207, SD=32), Medium (N=17) from 260 to 247 (M=312, SD=25), and High (N=8) from 358 to 444 (M=396, SD=37)."

Note that in these commands the data frame `mockdata` has been specified explicitly, so that the computed variables are added to it. Even if `mockdata` has previously been made the default data frame with an `attach()` command, the new columns can not yet be used without the data frame prefix. To make them available first `detach()` and then `attach()` again:

```
> table(ValueGroup)
Error in table(ValueGroup) : object 'ValueGroup' not found
> detach(mockdata)
> attach(mockdata)
> table(ValueGroup)
ValueGroup
  High    Low Medium
     8      5     17
```

If you no longer need variables, they can be removed.

```
>mockdata<-mockdata[,1:8]
```

This will retain all rows, but only the first eight columns, deleting the three just computed.

Comparing two means from one group with a paired t test

When people have been measured twice on a variable, so they have a pair of measures, the means can be compared using a paired t test.

```
> t.test(Value1, Value2, paired=TRUE, var.equal = TRUE)

	Paired t-test

data:  Value1 and Value2
t = -3.3941, df = 29, p-value = 0.002011
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -39.530416  -9.802918
sample estimates:
mean of the differences 
              -24.66667 
> describe(Value1)
   vars  n  mean    sd median trimmed   mad min max range skew kurtosis   se
X1    1 30 83.03 34.17   76.5   81.42 37.81  29 163   134 0.43     -0.6 6.24
> describe(Value2)
   vars  n  mean    sd median trimmed mad min max range skew kurtosis   se
X1    1 30 107.7 33.49    101  107.33  43  43 165   122 0.06    -1.06 6.11
```

"Scores increased from 83.0 (34.2) at baseline to 107.7 (33.5) after training, t(29)=3.39, p=.002."

The minus sign before the t value can always be ignored as it simply depends on the order the variables are listed.

Comparing two groups' means with a two sample t test

Where two groups have been measured on the same variable, their means can be compared using a two sample t test. First the data needs to be split into the two groups. This can be done by creating a data frame for each group, which is useful if more commands might need to be run on each group:

```
> control.df<-mockdata[Group=="control",]
> intervention.df<-mockdata[Group=="intervention",]
```

These commands will select all of the rows of mockdata where Group has the specified value, and copy them into the new data frames. Note that two equals signs are needed, and that there is nothing after the comma so that all columns (variables) are selected.

Having created the two separate sets of data, the t.test command can be used to refer to them explicitly using the dollar notation, which takes the format `dataframe$variable`

```
> t.test(control.df$Value1,intervention.df$Value1,var.equal = TRUE)

	Two Sample t-test

data:  control.df$Value1 and intervention.df$Value1
t = -0.16284, df = 28, p-value = 0.8718
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -28.06455  23.93121
sample estimates:
mean of x mean of y 
 82.00000  84.06667 
```

This can also all be done within the t.test command if there is no need to retain the two groups' data for later use:

```
> t.test(mockdata[Group=="control",]$Value1,
         mockdata[Group=="intervention",]$Value1,
         var.equal = TRUE)
```

Here, creating the two subsets of data is helpful because it allows the group descriptives to be obtained easily:

```
> describe(control.df$Value1)
   vars  n mean    sd median trimmed   mad min max range skew kurtosis    se
X1    1 15   82 38.97     73   80.92 54.86  29 149   120 0.16    -1.49 10.06
> describe(intervention.df$Value1)
   vars  n  mean    sd median trimmed   mad min max range skew kurtosis   se
X1    1 15 84.07 29.96     78      81 26.69  45 163   118 1.01     0.68 7.74
```

although this can also be done using this command:

```
> describeBy(Value1, group=Group)
```

> "The control group (M=82.0, SD=39.0) and the intervention group (M=84.1, SD=30.0) did not differ at baseline t(28)=0.16, p=.872"

Oneway between-subjects Analysis of Variance

When there are more than two groups or levels of any factor, ANOVA is needed, not multiple t tests.

Test whether the three experimenters recruited people who were different at baseline.

```
> oneway.test(Value1 ~ Experimenter, var.equal = TRUE)

	One-way analysis of means

data:  mockdata$Value1 and mockdata$Experimenter
F = 3.447, num df = 2, denom df = 27, p-value = 0.04643
```

In the brackets, the formula `Value1 ~ Experimenter` makes `Value1` the IV and `Experimenter` the DV. The `var.equal=TRUE` bit tells the analysis to assume that the SDs for each Experimenter are equivalent. There is a significant effect, reported like this:

> "The participants recruited by each Experimenter did not all have the same means at baseline F(2,27)=3.45, p=.046."

```
> describeBy(mockdata$Value1, group=mockdata$Experimenter)
group: JM
   vars  n mean   sd median trimmed   mad min max range skew kurtosis  se
X1    1 10 64.2 27.5   60.5   63.12 24.46  29 108    79 0.33    -1.37 8.7
--------------------------------------------------------------------------------
group: LS
   vars  n mean    sd median trimmed   mad min max range skew kurtosis   se
X1    1 10 83.6 27.39     85    85.5 37.06  34 118    84 -0.3    -1.33 8.66
--------------------------------------------------------------------------------
group: MK
   vars  n  mean    sd median trimmed   mad min max range skew kurtosis    se
X1    1 10 101.3 38.61   92.5   99.38 47.44  55 163   108 0.27    -1.67 12.21
> with(mockdata,pairwise.t.test(Value1,Experimenter,p.adj="bonferroni",paired=F))
```

MK is clearly higher than JM, but what about LS? Post hoc t tests are needed to compare them, using Bonferroni correction because we are making three tests at once:

```
> pairwise.t.test(Value1,Experimenter,p.adj="bonferroni",paired=F)

       Pairwise comparisons using t tests with pooled SD

data:  Value1 and Experimenter

   JM    LS
LS 0.544 -
MK 0.042 0.664

P value adjustment method: bonferroni
```

This handy command just gives the *p* values, not the t values. It could be reported like this, using the descriptives too:

> "Two-tailed t tests with bonferroni correction indicated that MK's participants (M=101.3, SD=38.6) scored higher than JM's (M=64.2, SD=27.5) p=.042, and that LS's participants (M=83.6, SD=27.4) did not differ to MK's p=.664 or JM's p=.544."

aov - A more general ANOVA command

Another, more general way to do ANOVA is to use this function:

```
> m<-aov(Value1 ~ Experimenter)
> summary(m)
             Df Sum Sq Mean Sq F value Pr(>F)
Experimenter  2   6887    3443   3.447 0.0464 *
Residuals    27  26972     999
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The F and p values are the same as in the `oneway` output, but here the results are being stored in an object called m instead of printing it right away because the output from `aov` is more complicated, and putting it into m allows other commands to be run as well, such as an effect size statistic and post hoc tests (the label 'm' is just a name; it could be called anything).

eta squared ($\eta^{2)}$:

```
> library(lsr) # this library includes the etaSquared command
> etaSquared(m)
                eta.sq eta.sq.part
Experimenter 0.2033986   0.2033986
```

There are two values here: etasq (which is the variance explained by the effect) and the partial eta squared (which supposedly adjusts for the inclusion of multiple effects and interactions in the model). As there is only one effect in this simple oneway anova, they are identical, and the write up would be:

> "The participants recruited by each Experimenter did not all have the same means at baseline F(2,27)=3.45, p=.046, η^2=.20."

Post hoc tests

You can also ask for Tukey's Honestly Significant Difference posthoc test if you need to make unplanned comparisons between more than two levels of a between-subject factor:

```
> TukeyHSD(m)
  Tukey multiple comparisons of means
    95% family-wise confidence level

Fit: aov(formula = Value1 ~ Experimenter)

$Experimenter
      diff       lwr      upr     p adj
LS-JM 19.4 -15.64614 54.44614 0.3690517
MK-JM 37.1   2.05386 72.14614 0.0363937
MK-LS 17.7 -17.34614 52.74614 0.4337410
```

As with the Bonferroni t tests, MK and JM's groups differ. Write this up like this:

> "Tukey's HSD tests conducted post hic indicated that MK's participants (M=101.3, SD=38.6) scored higher than JM's (M=64.2, SD=27.5) p=.036, and that LS's participants (M=83.6, SD=27.4) did not differ to MK's p=.434 or JM's p=.369."

These p values are lower than those from the pairwise t tests, because Tukey's HSD is a more sensitive test.

Two-between ANOVA

The formula in `aov` can be extended to add in multiple between-subject factors, for example, testing Experimenter and Group at the same time, using the following 'star' notation to request the main effects and all interactions:

```
> m<-aov(Value1 ~ Experimenter * Group)
> summary(m)
                   Df Sum Sq Mean Sq F value Pr(>F)
Experimenter        2   6887    3443   3.144 0.0613 .
Group               1     32      32   0.029 0.8656
Experimenter:Group  2    652     326   0.298 0.7451
Residuals          24  26288    1095
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
> etaSquared(m)
                         eta.sq eta.sq.part
Experimenter       0.2033986074 0.207595400
Group              0.0009460812 0.001217089
Experimenter:Group 0.0192701293 0.024219193
```

> "At baseline, there were no differences between the experimenter's groups F(2,24=3.14, p=.061, η_p^2 = .21 or between the conditions F(1,24)=0.03, p=.866, η_p^2 = .00, and there was no interaction of experimenter and condition F(2,24)=0.29, p=.745, η_p^2 =.02"

The same approach works for more than two between-subject factors, but repeated measures designs with within-subject factors requires more work. The data needs to be reshaped.

Reorganising data (Wide to Long)

In mockdata, people were measured three times: Value1 at baseline, Value2 at the end of training, and Value3 in a follow-up session some weeks later. The data file includes all three measurements for each participant on a single line, as is convenient for data entry. This is called WIDE format.

To see the first few rows, use the `head` command:

```
> head(mockdata)
# A tibble: 6 × 8
  Subject Experimenter   Age   Sex   Group Value1 Value2 Value3
    <dbl>        <chr> <dbl> <chr>   <chr>  <dbl>  <dbl>  <dbl>
1       3           JM    18     M control     32     99     46
2       6           LS    19     F control     34    120     71
3       8           LS    20     M control     60     91     76
4      14           MK    18     F control    149    131     78
5      13           MK    18     F control     73     80     84
6      11           MK    20     F control    130    145     85
```

This is nice and compact, but many analyses in R require LONG format, where a single column contains all of the measurements, and another column says which time they are from.

This is how to turn a wide data set into a long one:

```
>library(reshape2) # contains the reshaping commands
>longdata<-melt(data=mockdata, measure.vars = c("Value1","Value2","Value3"))
```

This creates a new dataset called `longdata`. The c(…) lists the three variables that are all the same measure taken at different times. This is how the new file looks:

```
> head(longdata)
  Subject Experimenter Age Sex   Group variable value
1       3           JM  18   M control   Value1    32
2       6           LS  19   F control   Value1    34
3       8           LS  20   M control   Value1    60
4      14           MK  18   F control   Value1   149
5      13           MK  18   F control   Value1    73
6      11           MK  20   F control   Value1   130
```

The level names are now in the column `variable`, and the actual values are in the column `value`

Finally, make this the new default data frame:

```
>detach(mockdata)
>attach(longdata)
```

Renaming levels in a factor and creating factors from variables

The variable names that are now in `variable` are not very meaningful. They can be changed to more meaningful labels, but as variable is a factor, the `levels()` command needs to be used:

```
> levels(variable)[levels(variable)=="Value1"]<-"Baseline"
> levels(variable)[levels(variable)=="Value2"]<-"After Testing"
> levels(variable)[levels(variable)=="Value3"]<-"Follow-up"
> head(longdata)
  Subject Experimenter Age Sex   Group variable value
1       1           JM  19   F control Baseline   108
2       2           JM  18   F control Baseline    63
3       3           JM  18   M control Baseline    32
4       4           JM  21   F control Baseline    29
5       5           JM  22   F control Baseline    58
6       6           LS  19   F control Baseline    34
```

One more thing to do: an integer has been used for the Subject number, and this variable needs to be turned into a factor before use in a repeated measures design:

```
> longdata$SubF<-as.factor(Subject)
```

This adds a new column to the data where each subject number is now the level of a `SubF` factor.

Repeated Measures ANOVA

With long format data, a simple oneway repeated measures ANOVA can be done using the `aov` command. Unlike the between-subjects designs, the Error term now needs to be added explicitly.

```
> m<-aov(value ~ variable + Error(SubF/variable))
```

The Error term is telling `aov` that each level of `SubF` was measured at each level of `variable`.

```
> summary(m)

Error: SubF
          Df Sum Sq Mean Sq F value Pr(>F)
Residuals 29  44954    1550

Error: SubF:variable
          Df Sum Sq Mean Sq F value   Pr(>F)
variable   2  27976   13988   13.77 1.28e-05 ***
Residuals 58  58935    1016
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

(NB: The `etaSquared` and `TukeyHSD` commands do not work with repeated measures designs.)

> “A repeated measures ANOVA showed a significant effect of time F(2,58)=13.8, p<.001”

This isn’t very informative as it just shows that the means differ, not whether they go up or down.

If `Subject` had not been turned into the factor `SubF`, this is what happens:

```
> m<-aov(value ~ variable + Error(Subject/variable))
> summary(m)

Error: Subject
          Df Sum Sq Mean Sq F value Pr(>F)
Residuals  1  11721   11721

Error: Subject:variable
         Df Sum Sq Mean Sq
variable  2  27849   13924

Error: Within
          Df Sum Sq Mean Sq F value Pr(>F)
variable   2   2365    1183   1.105  0.336
Residuals 84  89930    1071
```

This non-significant F is because the integer values of `Subject` are treated as meaningful values of Subjectness, as if participant 24 were twice as much a Subject than participant 12. **This is clearly nonsense. Please do remember to make sure any Subject variable is being treated as a factor.**

Building a table of means and SDs

To understand what the statistically significant value of F in the oneway repeated measures analysis means, you need some descriptive statistics.

The `aggregate` function computes means and SDs, and its results can be combined into a table, with new column headings, and numbers rounded off sensibly.

```
> tab1<-aggregate(data=longdata, value ~ variable, mean)
```

This makes a table of means of value, split by variable.

```
> tab1
       variable     value
1      Baseline  83.03333
2 After Testing 107.70000
3     Follow-up 126.06667
```

Relabel the value column M to show it holds means

```
> colnames(tab1)[2]="M"
```

The [2] makes the command affect the second column of the table.

Add another column called SD by using `aggregate` again to get standard deviations, and asking for the second column of the results by putting [2] afterwards.

```
> tab1["SD"]<-aggregate(data=longdata, value ~ variable, sd)[2]
> tab1
       variable         M       SD
1      Baseline  83.03333 34.16944
2 After Testing 107.70000 33.48922
3     Follow-up 126.06667 35.96256
```

Five decimal places is messy, so round columns 2 and 3 of this table to one decimal place:

```
> tab1[,2:3]<-round(tab1[,2:3],digits=1)
> tab1
       variable     M   SD
1      Baseline  83.0 34.2
2 After Testing 107.7 33.5
3     Follow-up 126.1 36.0
```

The [,2:3] means the rounding is applied to all rows (the row numbers to operate on are listed before the comma; leaving it blank means all of them) of columns two to three (so 2:5 would affect columns 2 to 5, if they all existed).

These descriptives can help improve the reporting of the repeated measures ANOVA:

"A repeated measures ANOVA showed that scores increased over time F(2,58)=13.8, p<.001, from M=83.0 (SD=34.2) at baseline, to M=107.7 (SD=33.5) at the end of training, rising to M=126.1 (SD=36.0) at follow-up."

Mixed measures ANOVA

As well as the within-subject factor, there is a between subject factor.

Again, aov can model this:

```
> m<-aov(value ~ variable * Group + Error(SubF/variable))
> summary(m)

Error: SubF
          Df Sum Sq Mean Sq F value Pr(>F)
Group      1   8841    8841   6.854 0.0141 *
Residuals 28  36114    1290
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Error: SubF:variable
               Df Sum Sq Mean Sq F value   Pr(>F)
variable        2  27976   13988  15.100 5.69e-06 ***
variable:Group  2   7058    3529   3.809   0.0281 *
Residuals      56  51877     926
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

"A mixed measures ANOVA with one within-subject factor (time: baseline, after-training, follow-up) and one between subjects factor (condition: control, intervention) showed that overall the intervention group (M=115.5, SD=39.4) scored higher than the controls (M=95.7, SD=35.3) F(1,28)=6.85, p=.014; that scores increased over time (baseline: M=83.0, SD=34.17; after training: M=107.7, SD=33.5); follow-up: M=126.1, SD=36.0) F(2,56)=15.1, p<.001, and that there was an interaction of time and condition F(2,56)=3.81, p=.028, with the intervention group's scores increasing more over time than the control group's scores (see Figure 1)."

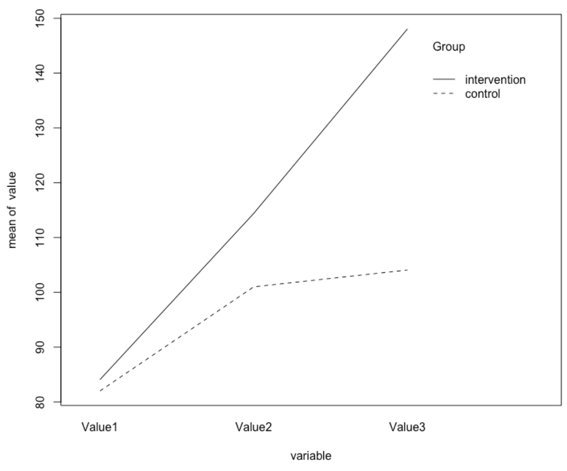
This time the means and SDs came from `describeBy` commands, part of the `psych` library – in R there are always lots of ways of doing things. The `mat` and `digits` parameters simplify the output by producing a 'matrix' instead of listing everything, and only giving two decimal places:

```
> describeBy(value, Group, mat=TRUE, digits=2)
    item       group1 vars  n   mean    sd median trimmed   mad min max range  skew kurtosis   se
X11    1      control    1 45  95.69 35.32     94   95.95 40.03  29 164   135 -0.02    -0.94 5.26
X12    2 intervention    1 45 115.51 39.35    116  115.76 47.44  45 180   135  0.03    -1.15 5.87
> describeBy(value, variable,mat=TRUE,digits=2)
    item        group1 vars  n   mean    sd median trimmed   mad min max range  skew kurtosis   se
X11    1      Baseline    1 30  83.03 34.17   76.5   81.42 37.81  29 163   134  0.43    -0.60 6.24
X12    2 After Testing    1 30 107.70 33.49  101.0  107.33 43.00  43 165   122  0.06    -1.06 6.11
X13    3     Follow-up    1 30 126.07 35.96  123.5  127.38 45.96  46 180   134 -0.20    -0.85 6.57
```

Plotting an interaction chart

A really quick way to see what is happening in a two-way interaction is this:

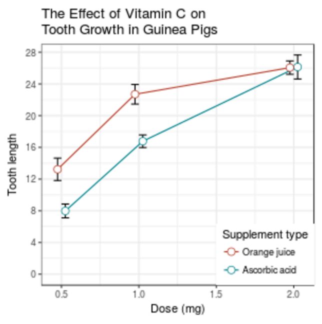
```
>interaction.plot(variable,Group,value)
```

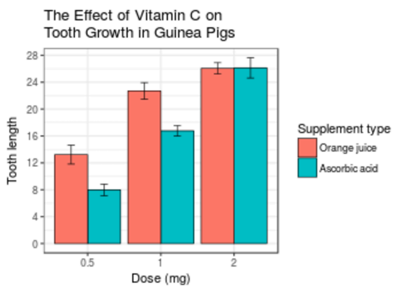


Both groups improve over time, but while the intervention group keep on improving after training, the control group level off. This graph is not good enough for a Results section, though. It also needs to have error bars showing 95% confidence intervals, or standard errors of each mean. These need to be computed and put into a table.

For full details on how to do this, visit :
http://www.cookbook-r.com/Graphs/Plotting_means_and_error_bars_(ggplot2)

where functions and code can be copied to produce graphs like these:



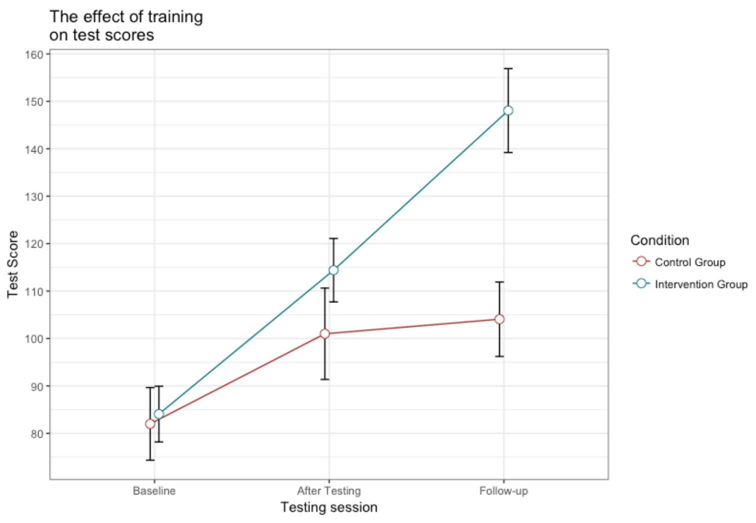


Using the `longdata` data frame and the `summarySEwithin` functions provided on that page, this command produces the required table of descriptive statistics:

```
> wc.df<-summarySEwithin(longdata,
                         measurevar = "value",
                         withinvars = "variable",
                         betweenvars= "Group",
                         idvar=       "Subject",
                         na.rm=FALSE, conf.interval = .95)
Automatically converting the following non-factors to factors: Group
> wc.df
         Group      variable  N     value value_norm       sd       se       ci
1      control After Testing 15 101.00000  110.91111 37.30264 9.631501 20.65752
2      control      Baseline 15  82.00000   91.91111 29.65254 7.656252 16.42103
3      control     Follow-up 15 104.06667  113.97778 30.39301 7.847441 16.83109
4 intervention After Testing 15 114.40000  104.48889 25.89478 6.686004 14.34005
5 intervention      Baseline 15  84.06667   74.15556 22.76258 5.877272 12.60550
6 intervention     Follow-up 15 148.06667  138.15556 34.27920 8.850851 18.98319
```

The following code will then produce an appropriate graph, with standard error bars – the red text shows where variable names or values need to be customised:

```
library(ggplot2)
pd <- position_dodge(0.1) # move series .05 to the left and right to avoid overlap
ggplot(wc.df, aes(x=variable, y=value, colour=Group, group=Group)) +
     geom_errorbar(aes(ymin=value-se, ymax=value+se), colour="black", width=.1, position=pd) +
     geom_line(position=pd) +
     geom_point(position=pd, size=3, shape=21, fill="white") + # 21 is filled circle
     xlab("Testing session") +
     ylab("Test Score") +
     scale_colour_hue(name="Condition",    # Legend label, use darker colors
                      breaks=c("control", "intervention"),
                      labels=c("Control Group", "Intervention Group"),
                      l=40) +                       # Use darker colours, lightness=40
     ggtitle("The effect of training\non test scores") + #\n to force a line-break
     #expand_limits(y=0) +                          # Uncomment this line to expand y range
     scale_y_continuous(breaks=0:20*10) +           # Set y-axis tick every 10 units (max 20 ticks)
     theme_bw() +
     theme(legend.justification=c(1,0),
           legend.position=c(1,0))                 # Position legend in bottom right
```



Note that for strict APA format there should be no legend, with the lines being defined in the caption e.g.

Figure 1: Both groups' test scores improved following training, but the intervention group's scores (blue line) continued to improve over the follow-up interval, while the control group's scores (red line) did not. Error bars show one standard error (within subject).

Cronbach's alpha

Exploratory Factor Analysis

Confirmatory Factor Analysis