Chapter 2
Two-Way Analysis of Variance (ANOVA) Sample 1: Comparison of Scores on a Final Examination by Teaching Method and by Status as a Community College Graduate

Abstract R is used in this chapter to support investigations on the assessment of student learning outcomes in higher education. A tab-separated file was used as the data source and from this file it was demonstrated how R is used to organize and label data, prepare simple graphical figures for quality assurance purposes, provide descriptive statistics overall and by breakout groups, and conduct a two-way analysis of variance (ANOVA). This chapter provides a simple introduction on the use of R, with an emphasis on easy-to-follow confidence building actions that model a real assessment-type setting faced daily by those who work in education and the social sciences.

This study was designed to examine if there are differences in final examination test scores for students in a software engineering course by teaching method (four breakout groups: (1) traditional lecture (e.g., lecture), (2) computer based training (e.g., CBT), (3) instructional video, placed on a tablet computing device (e.g., video), and (4) independent study (e.g., IDS)) and by Status as a Community College graduate (two breakout groups: (1) graduate of a Community College and (2) not a graduate of a Community College).

The motivation for this study was to react to university-level faculty concerns about the skills two-year community college graduates bring into advanced courses (overall and by different teaching methods), as compared to the skills of students who enroll in their first two years of undergraduate study at a four-year university. For background on methods, students were all enrolled in a university senior-level software engineering course. Students were assigned, through an alpha-sort by last name random selection process, to placement into one of the four teaching method groups. Community College graduation status was determined by transcript data maintained in the Registrar's Office. All students sat for the same final examination. This design will allow investigations by teaching method, by Community College graduation status, and by possible interactions between these two factor-type object variables (teaching method and Community College graduation status).

T.W. MacFarland, *Two-Way Analysis of Variance: Statistical Tests and Graphics Using R*, SpringerBriefs in Statistics 1, DOI 10.1007/978-1-4614-2134-4_2,
© Thomas W. MacFarland 2012

The principal investigator was confident that final examination scores represented interval data. As such, two-way analysis of variance (ANOVA) was judged to be the appropriate test for this factorial-type analysis of summative differences in final examination scores by teaching method (four breakout groups) and by Status as a Community College graduate (two breakout groups). The header, the first three lines of data, and the last three lines of data, presented in tab-separated format, are shown below, but of course the tab characters although present do not show:

```
ID Method ComCol Final
01 1 1 089
02 1 1 081
03 1 2 073
73 4 2 062
74 4 1 056
75 4 1 085
```

Sample 1 Ho (Null Hypothesis): There is no difference between teaching method, graduation status from a Community College, and interaction between teaching method and graduation status from a Community College regarding final examination test scores of students enrolled in a university senior-level software engineering course (`p <= 0.05`).

2.1 Data Import of a .txt Tab-Delimited Data File into R

For this lesson, notice how the dataset has been prepared in .txt file format (not .csv file format) and that the data are separated by tab spacings, not commas. Experienced researchers will work with data in many formats, so it is desirable to gain experience with this type of file format even though .csv (comma-separated values) data files are certainly quite common when using R and other data analysis tools. The data for this lesson, in .txt tab-delimited format, have been placed at the `F:\R_Lessons\Inferential_Statistics_Parametric` directory on a standalone personal computer.

All analyses begin from this starting point, working with a previously prepared .txt file. The emphasis will be on: (1) overall analysis of Final, (2) breakout analyses of Final; Final by teaching method and Final by Status as a Community College graduate, (3) graphical representation of overall and breakout findings, (4) two-way ANOVA of the data, and (5) summative interpretation of outcomes.

```
#####################################################################
# Housekeeping                          Use for all analyses
#####################################################################
setwd("F:/R_Lessons/Inferential_Statistics_Parametric")
                  # Set to a new working directory.
                  # Note the single forward slash and double
                  # quotes.
```

```
                    # This new directory should be the directory
                    # where the data file is located, otherwise
                    # the data file will not be found.
getwd()             # Confirm the working directory.
search()            # Attached packages and objects.
###################################################################
```

2.1.1 Data Import or Data Entry

```
Final.table <- read.table (file =
  "SoftwareEngineeringFinal_PriorCC_tab-separated.txt",
  header = TRUE,
  sep = "\t")          # Import the tab-separated .txt
                       # file.
getwd()                # Identify the working directory.
ls()                   # List objects.
attach(Final.table)    # Attach the data, for later use.
names(Final.table)     # Identify names.
head(Final.table)      # Show the head.
tail(Final.table)      # Show the tail.
Final.table            # Show the entire data frame.
```

By completing this action, an object called Final.table has been created. This object consists of the data included in the tab-separated file. Make sure that there are no prior R-based datasets called Final.table available. Note how it was only necessary to key the filename for the .txt file and not the full pathname since the R working directory is currently set to the directory and subdirectory where this .txt file is located (see Sect. 2.1 at the beginning of this lesson).

2.2 Organize the Data and Display the Code Book

Now that the data have been imported into R, it is usually necessary to check the data for format and then make any changes that may be needed, to organize the data. As a typical example, consider the common practice of numeric codes, as factors, for Gender. If Gender is coded as 1 and 2 instead of Female (1) and Male (2), it is necessary to do something so that 1 and 2 are seen as factor (e.g., group) values and not integers more suited for math operations. This concept applies to all other cases where numeric codes are used to identify factors (e.g., groups). In this example, that concept applies to the numeric codes used to identify the four teaching methods and the two options regarding status as a Community College graduate.

```
class(Final.table)
class(Final.table$ID)         # DataFrame$ObjectName notation.
class(Final.table$Method)     # DataFrame$ObjectName notation.
```

```
class(Final.table$ComCol)     # DataFrame$ObjectName notation.
class(Final.table$Final)      # DataFrame$ObjectName notation.
```

Now that the `class()` function has been applied against each object, consult the code book and coerce each object, as needed, into its correct class. Typically, integers that serve as numeric codes (e.g., 1 represents Female and 2 represents Male) are coerced into factor format.

```
# Code Book ###################################################
################################################################
# Software Engineering Final Examination Results by
  Teaching Method and by Status as a Community
  College Graduate
#
# Variable Labels
#   ID          Student Identification Number
#   Method      Teaching Method
#   ComCol      Status as a Community College Graduate
#   Final       Final Examination Score
#
# Variable Values
#   ID            Nominal    LOW to HIGH
#   Method        Nominal    1 Lecture
#                            2 CBT (Computer-based Training)
#                            3 Video
#                            4 IDS (Independent Study)
#   ComCol        Nominal    1 Yes Is a Community College
                                 Graduate
#                            2 No  Is not a Community
#                                  Community Graduate
#   Final         Interval   000 Lowest Possible Score
#                            100 Highest Possible Score
################################################################
```

In an effort to promote self-documentation and readability, it is often desirable to label all object variables. The `epicalc::label.var()` function can serve this purpose. Of course, be sure to load the epicalc package, if it is not operational from prior analyses.

```
install.packages("epicalc")
library(epicalc)          # Load the epicalc package.
help(package=epicalc)     # Show the information page.
sessionInfo()             # Confirm all attached packages.
```

Comment: Use help(library) and help(require) to see the functional difference, if any, between these two functions.

```
epicalc::des(Final.table)
```

Use the `epicalc::des()` function to see the nature of the data frame. Then, provide a useful description of each object variable by using the `epicalc::label.var()` function.

```
epicalc::label.var(ID,     "Subject ID",
  dataFrame=Final.table)
epicalc::label.var(Method, "Teaching Method",
  dataFrame=Final.table)
epicalc::label.var(ComCol, "Community College Status",
  dataFrame=Final.table)
epicalc::label.var(Final,  "Final Examination Score",
  dataFrame=Final.table)

epicalc::des(Final.table)
  # Confirm the description of each object variable.
```

Coerce objects into correct format. Notice how variables are named: `DataFrame$ObjectName`. At first this action may seem somewhat cumbersome, but it is actually very useful to ensure that actions are performed against the correct object. Most text editors allow the use of copy/paste and find/replace, so it should be a simple operation to organize the syntax.

```
# Object class before coercion
class(Final.table)
class(Final.table$ID)       # DataFrame$ObjectName notation.
class(Final.table$Method)   # DataFrame$ObjectName notation.
class(Final.table$ComCol)   # DataFrame$ObjectName notation.
class(Final.table$Final)    # DataFrame$ObjectName notation.

# Coercion
Final.table$ID       <- as.factor(Final.table$ID)
Final.table$Method   <- as.factor(Final.table$Method)
Final.table$ComCol   <- as.factor(Final.table$ComCol)
Final.table$Final    <- as.numeric(Final.table$Final)

# Object class after coercion
class(Final.table)
class(Final.table$ID)       # DataFrame$ObjectName notation.
class(Final.table$Method)   # DataFrame$ObjectName notation.
class(Final.table$ComCol)   # DataFrame$ObjectName notation.
class(Final.table$Final)    # DataFrame$ObjectName notation.
```

As a sidebar comment, at the R prompt, key help(as.numeric) and then key help(as.integer) to see the differences between these two R functions and when it may be best to use each.

Use the `str()` function to confirm object format. Note the details for `str()` output, especially the output against the data frame Final.table.

```
str(Final.table)
str(Final.table$ID)
```

```
str(Final.table$Method)
str(Final.table$ComCol)
str(Final.table$Final)
```

The epicalc package has many useful functions. Saying this, use the `epicalc::des()` function to again describe the data frame currently in use.

```
epicalc::des(Final.table)
```

Then, merely to further confirm the nature of the dataset, use the `levels()` function against the factor object variables, to reinforce understanding of the data.

```
levels(Final.table$Method)
levels(Final.table$ComCol)
```

Use the `summary()` function against the object Final.table, which is a data frame, to gain an initial sense of descriptive statistics and frequency distributions.

```
summary(Final.table)
```

Although the dataset seems to be in correct format, it is somewhat difficult to work with numeric values for factor object variables: Method and ComCol. Use the code book to review the meaning for each factor code and then note how this problem is easy to accommodate. As is nearly always the case with R, there is no one-and-only-one way to apply labels and recode data.

```
# Apply the labels() function.
Final.table$Method.recode  <- factor(Final.table $Method,
  labels = c("Lecture", "CBT", "Video", "IDS"))

head(Final.table$Method)
head(Final.table$Method.recode) # View the first lines of data.

# Apply the labels() function.
Final.table$ComCol.recode  <- factor(Final.table $ComCol,
  labels = c("Yes - is a CC Graduate",
             "No  - not a CC Graduate"))

head(Final.table$ComCol)
head(Final.table$ComCol.recode) # View the first lines of data.
```

To recap, the object variable `Final.table$Method.recode` was created by applying the `factor()` function against the object variable `Final.table$Method`. The `labels()` function was used to embellish future output into simple English. This same approach was also used to create the object variable `Final.table$ComCol.recode` and then create labels for this new object variable.

Now, merely use the `attach()` function again to confirm that all data are attached to the data frame.

```
attach(Final.table)
head(Final.table)
tail(Final.table)
summary(Final.table)  # Quality assurance data check.
str(Final.table)      # List all objects, with finite detail.
```

As an additional data check, use the `table()` function to see how data have been summarized using the newly created names (factor object variables) as well as the original names for the numeric object variables.

```
table(Final.table$Method,         useNA = c("always"))
table(Final.table$Method.recode, useNA = c("always"))

table(Final.table$ComCol,         useNA = c("always"))
table(Final.table$ComCol.recode, useNA = c("always"))

table(Final.table$Method.recode,
      Final.table$ComCol.recode, useNA = c("always"))
```

Note how the argument `useNA = c("always")` is used with the table function, to force identification of missing values.

This type of redundancy and attention to detail at this stage of development may seem unnecessary, but it more than helps in reducing later errors caused by a simple oversight.

2.3 Conduct a Visual Data Check

The `summary()` function, `min()` function, and `max()` function are all certainly useful for data checking, but there are also many advantages to a visual data check process. In this case, simple plots and bar charts can be very helpful in an attempt to look for data that may be either illogical or out-of-range. These initial plots and barcharts will be, by design, simple, and should be considered throwaways as they are intended only for initial diagnostic purposes. They will then be followed by graphical images that provide more detail and may have future use in any possible presentation(s).

```
names(Final.table)    # Confirm all object variables.
```

2.3.1 Simple Plots

```
par(ask=TRUE)
plot(Final.table$ID,
  main="Final.table$ID Visual Data Check",
```

```
  col = c(rainbow(75)))
  # rainbow refers to the 75 datapoints for ID.

par(ask=TRUE)
plot(Final.table$Method,
  main="Final.table$Method Visual Data Check",
  col = c(heat.colors(4)))
  # heat.colors refers to the 4 groups for Method.

par(ask=TRUE)
plot(Final.table$Method.recode,
  main="Final.table$Method.recode Visual Data Check",
  col = c(terrain.colors(4)))
  # terrain.colors refers to the 4 groups for Method.recode.

par(ask=TRUE)
plot(Final.table$ComCol,
  main="Final.table$ComCol Visual Data Check",
  col = c(topo.colors(2)))
  # topo.colors refers to the 2 groups for ComCol.

par(ask=TRUE)
plot(Final.table$ComCol.recode,
  main="Final.table$ComCol.recode Visual Data Check",
  col = c(cm.colors(2)))
  # cm.colors refers to the 2 groups for ComCol.recode.

par(ask=TRUE)
plot(Final.table$Final,
  main="Final.table$Final Visual Data Check",
  pch=19,
  col = c("black"))
  # pch=19 displays datapoints as black solid circles.
```

The purpose of these initial plots is to gain a general sense of the data and to equally look for outliers. In an attempt to look for outliers, the `ylim` argument has been avoided, so that all data are plotted. Extreme values may or may not be outliers, but they are certainly interesting and demand attention.

2.3.2 Histogram of the Summary Object Variable

This sample lesson has been designed to look into the nature of object variable Final and the factor object variables Method and ComCol, recoded into a more verbose format as object variables Method.recode and ComCol.recode. Given the nature of Final values, it may also be a good idea to supplement the `plot()` function with

other functions, to gain a different view of the continuous values of Final, overall and by breakout groups.

```
par(ask=TRUE)
hist(Final.table$Final,
  main="Final.table$Final Visual Data Check(Histogram)",
  font=2, cex.lab=1.15, col="red")
```

For questions about this function and all other functions, simply key `help(function.name)` to learn more about the R function and the many arguments and options supported by the function. The use of arguments and attention to axis scales can greatly improve presentation, as shown in the next set of syntax.

```
par(ask=TRUE)
hist(Final.table$Final,
  main="Histogram of Final Exam Values",
  xlab="Final Exam Values (Limit = 0 to 100)",
  ylab="Frequency",
  xlim=c(0,120),    # Note the selection for xlim.
  ylim=c(0,25),     # Note the selection for ylim.
  cex.lab=1.15, cex.axis=1.15, freq=TRUE,
  border="blue", col="red")
  # Again, note the xlim=c(0,120) argument. By design,
  # the X axis is pushed out to 120 instead of 100, to
  # accommodate a full presentation of output.
```

Compare the output of the histogram to the normal curve and decide if overall distribution allows the use of planned inferential tests. Again, many tests are sufficiently robust to allow their use even if normal distribution is desired, but not fully met.

2.3.3 Horizontal and Vertical Boxplots of the Summary Object Variable

The boxplot is another tool typically used to gain a more complete understanding of the values represented by an object variable. Largely depending on preference and need, a boxplot can be displayed in either horizontal or vertical orientation while still highlighting the lower quartile (Q1), the median (Q2), the upper quartile (Q3), and any possible outliers.

```
par(ask=TRUE)
boxplot(Final.table$Final,
  horizontal=TRUE,
  main="Horizontal Boxplot of Final Exam Values",
```

```
  xlab="Final Exam Values (Limit = 0 to 100)",
  ylim=c(0,125),    # Note the selection for ylim.
  cex.lab=1.15, cex.axis=1.15,
  lty=1,            # Note the line type.
  lwd=3,            # Note the line width.
  border="blue", col="red")
box()

par(ask=TRUE)
boxplot(Final.table$Final,
  horizontal=FALSE,
  main="Vertical Boxplot of Final Exam Values",
  ylab="Final Exam Values (Limit = 0 to 100)",
  ylim=c(0,125),    # Note the selection for ylim.
  cex.lab=1.15, cex.axis=1.15,
  lty=1,            # Note the line type.
  lwd=3,            # Note the line width.
  border="blue", col="red")
box()
```

In the same way that descriptive statistics at the breakout level provide detail about selected object variables for specific groups (e.g., details on Female subjects v details on Male subjects), graphical displays at the breakout level are also useful. It is especially helpful to graphically display object variables at the breakout level to compare distribution, extreme values, etc. This is all done in an effort to determine, later, if there are (or are not) differences by breakout groups.

2.3.4 Sorted Dot Chart of the Summary Object Variable by Breakout Object Variables

```
par(ask=TRUE) #Method.recode
epicalc::summ(Final.table$Final,
  by=Final.table$Method.recode,
  graph=TRUE, # Use graph=TRUE, if desired.
  pch=20, ylab="auto",
  main="Sorted Dotplot of Final Exam Values
  by Teaching Method",
  cex.X.axis=1.25, # Note X axis label size.
  cex.Y.axis=1.25, # Note Y axis label size.
  font.lab=2, dot.col="auto")
```

```
par(ask=TRUE) # ComCol.recode
epicalc::summ(Final.table$Final,
  by=Final.table$ComCol.recode,
  graph=TRUE, # Use graph=TRUE, if desired.
  pch=20, ylab="auto",
  main="Sorted Dotplot of Final Exam Values
  by Community College Graduation Status",
  cex.X.axis=1.25, # Note X axis label size.
  cex.Y.axis=0.95, # Note Y axis label size.
  font.lab=2, dot.col="auto")
```

These sorted dot charts provide a clear view of final exam values by teaching method and then by Community College graduation status. The `epicalc::summ()` function and the accompanying `graph=TRUE` argument should always be among the first graphical tools used to examine distributions by breakout group.

2.3.5 Histogram of the Summary Object Variable by Breakout Object Variables

It is certainly possible to put all final exam values for Lecture into a separate object and to then prepare a histogram of only lecture-based final exam values. This action could then be repeated so that there are separate object variables for all teaching methods: lecture, CBT, video, and IDS. By using the lattice package, however, this excessively redundant action can be avoided, with separate histograms for each teaching method placed into the same graphical image. This action can then be repeated for the two Community College graduation status groups: Yes and No. This general approach and subsequent use of the lattice package can be used for other graphic techniques, as needed.

Breakout Histograms by Count

```
install.packages("lattice")
library(lattice)        # Load the lattice package.
help(package=lattice)   # Show the information page.
sessionInfo()           # Confirm all attached packages.

par(ask=TRUE) # Method.recode
lattice::histogram(~Final.table$Final |
                   Final.table$Method.recode,
  type="count",     # Note:  count
  par.settings=simpleTheme(lwd=2),
  par.strip.text=list(cex=1.15, font=2),
```

```
  scales=list(cex=1.15),
  main="Histograms (Count) of Final Exam Values
  by Teaching Method",
  xlab=list("Final Exam Values (Limit = 0 to 100)",
  cex=1.15, font=2),
  xlim=c(0,120),
  ylab=list("Count", cex=1.15, font=2),
  aspect=0.2,
  layout = c(1,4),  # Note:  1 Column by 4 Rows.
  col="red")

par(ask=TRUE) # ComCol.recode
lattice::histogram(~Final.table$Final |
                   Final.table$ComCol.recode,
  type="count",     # Note:  count
  par.settings=simpleTheme(lwd=2),
  par.strip.text=list(cex=1.15, font=2),
  scales=list(cex=1.15),
  main="Histograms (Count) of Final Exam Values by
  Community College Graduation Status",
  xlab=list("Final Exam Values (Limit = 0 to 100)",
  cex=1.15, font=2),
  xlim=c(0,120),
  ylab=list("Count", cex=1.15, font=2),
  aspect=0.2,
  layout = c(1,2),  # Note:  1 Column by 2 Rows.
  col="red")
```

Breakout Histograms by Percent

```
par(ask=TRUE) # Method.recode
lattice::histogram(~Final.table$Final |
                   Final.table$Method.recode,
  type="percent",   # Note:  percent
  par.settings=simpleTheme(lwd=2),
  par.strip.text=list(cex=1.15, font=2),
  scales=list(cex=1.15),
  main="Histograms (Percent) of Final Exam Values by
  Teaching Method",
  xlab=list("Final Exam Values (Limit = 0 to 100)",
  cex=1.15, font=2),
  xlim=c(0,120),
  ylab=list("Percent", cex=1.15, font=2),
  aspect=0.2,
```

```
  layout = c(4,1),  # Note:  4 Columns by 1 Row.
  col="red")

par(ask=TRUE) # ComCol.recode
lattice::histogram(~Final.table$Final |
                   Final.table$ComCol.recode,
  type="percent",   # Note:  percent
  par.settings=simpleTheme(lwd=2),
  par.strip.text=list(cex=1.15, font=2),
  scales=list(cex=1.15),
  main="Histograms (Percent) of Final Exam Values by
  Community College Graduation Status",
  xlab=list("Final Exam Values (Limit = 0 to 100)",
  cex=1.15, font=2),
  xlim=c(0,120),
  ylab=list("Percent", cex=1.15, font=2),
  aspect=0.2,
  layout = c(1,2),  # Note:  1 Column by 2 Rows.
  col="red")
```

The layout argument can be used to place output into a variety of configurations, in a column by row format. Experiment with this argument.

The use of both count and percent breakout histograms from the `lattice::histogram()` function is clearly demonstrated in this set of teaching method and Community College figures. The count argument produced a set of histograms that may be somewhat difficult to interpret given the largely unequal *N* for each teaching method group. The histograms based on percent give a different perspective of distribution. Whether these figures are ever published, these distribution patterns should always be confirmed before the use of any inferential statistical tests.

The syntax for production of these many images is easily modified for the current study or for future studies. Once R syntax has been put into desired form, it is common practice to use and reuse this tested syntax for other applications.

2.4 Descriptive Analysis of the Data

Now that the data have been organized into the desired format and the `.recode` objects have also been created, a few more actions should be attempted before inferential tests (e.g., Student's *T*-test, ANOVA, MANOVA, etc.) are organized. Specifically, it is useful to carefully examine the dataset using summary and breakout descriptive statistics. Common graphical presentations are also helpful, to have a good understanding of the data and to see if the assumptions associated with most inferential tests, such as normal distribution, can be accepted.

2.4.1 Summary Descriptive Statistics

For this presentation, descriptive statistics will be prepared for the continuous object variable `Final.table$Final` and the factor object variables `Final.table$Method.recode` and `Final.Table$ComCol.recode`. A variety of R functions that support descriptive statistics at the summary level include: `mean()`, `median()`, `summary()`, `psych::describe()`, and `epicalc::summ()`. Review the level of detail for each function.

```
mean(Final.table$Final, na.rm=TRUE)
median(Final.table$Final, na.rm=TRUE)
summary(Final.table$Final)

par(ask=TRUE)
epicalc::summ(Final.table$Final, # Use the epicalc package.
  by=NULL,
  graph=TRUE,
  box=TRUE,      # Generate a boxplot.
  pch=18,
  ylab="auto",
  main="Sorted Dotplot and Boxplot of Final.table$Final",
  cex.X.axis=1.15, cex.Y.axis=1.15,
  font.lab=2, dot.col="auto")
```

Using the `by=NULL` argument, the `epicalc::summ()` function does not provide breakout statistics, but instead provides descriptive statistics and an accompanying graphic for all `Final.table$Final values`.

```
install.packages("psych")
library(psych)          # Load the psych package.
help(package=psych)     # Show the information page.
sessionInfo()           # Confirm all attached packages.

psych::describe(Final.table$Final)
```

The `psych::describe()` function can be applied against the entire dataset. The desired output can then be copied and pasted into a formal report. Merely delete the output applied against numbers that are instead factor-type object variables, which are marked with a * symbol.

```
psych::describe(Final.table)
```

The `epicalc::summ()` function provides output that is somewhat similar to `psych::describe()` when applied against the full set of data, organized in the data frame. As a simple quality assurance tool, be sure to always look at the min (Minimum) and max (Maximum) output, to see if there are data that are simply

out-of-range (e.g., a 3, 7, etc., showing as min or max output when the scale only supports 1 and 2, as codes for a factor object variable).

```
epicalc::summ(Final.table)
```

2.4.2 Breakout Descriptive Statistics

Descriptive statistics at the summary level are essential, but breakout statistics are also needed to gain a more complete understanding of the data. There are many ways to obtain breakout statistics, but the `tapply()` function, the `psych::describe.by()` function, the `epicalc::summ()` function, the `doBy::summaryBy()`, and the `prettyR::brkdn()` function are among the most detailed and easiest to use, to discern differences between breakout groups such as the four breakout groups for the object variable `Final.table$Method.recode` associated with this sample lesson (lecture, CBT, video, and IDS) and the two breakout groups for the object variable `Final.table$ComCol.recode` associated with this sample lesson (Yes—is a CC Graduate, and No—not a CC Graduate).

```
# tapply() Function
tapply(Final, Method.recode, summary, na.rm=TRUE,
  data=Final.table)
tapply(Final, ComCol.recode, summary, na.rm=TRUE,
  data=Final.table)

# psych::describe.by() Function
psych::describe.by(Final.table$Final, Final.table$Method.recode)
psych::describe.by(Final.table$Final, Final.table$ComCol.recode)

# epicalc::summ() Function
par(ask=TRUE)
epicalc::summ(Final.table$Final,
   by=Final.table$Method.recode,  # Note the use of by.
   graph=TRUE, # Use graph=TRUE,
   pch=20,     # if desired.
   ylab="auto",
   main="Sorted Dotplot of Final.table$Final
   by Final.table$Method.recode",
   cex.X.axis=0.95, cex.Y.axis=0.95,
   font.lab=2, dot.col="auto")

# epicalc::summ() Function
par(ask=TRUE)
epicalc::summ(Final.table$Final,
   by=Final.table$ComCol.recode,
   graph=TRUE, # Use graph=TRUE,  # Note the use of by.
   pch=20,     # if desired.
   ylab="auto",
```

```
   main="Sorted Dotplot of Final.table$Final
   by Final.table$ComCol.recode",
   cex.X.axis=0.95, cex.Y.axis=0.95,
   font.lab=2, dot.col="auto")

# doBy::summaryBy() Function
install.packages("doBy")
library(doBy)            # Load the doBy package.
help(package=doBy)       # Show the information page.
sessionInfo()            # Confirm all attached packages.

doBy::summaryBy(Final ~ Method.recode +
                        ComCol.recode,
                        data=Final.table,
                        FUN=c(mean,sd),
                        na.rm=TRUE,
                        keep.names=TRUE,
                        order=TRUE)
```

It will be difficult to accommodate missing values for length. The enumerated function below takes care of this problem.

```
descriptivefun <- function(x, ...){
  c(m=mean(x, ...), sd=sd(x, ...), l=length(x))
}

doBy::summaryBy(Final ~ Method.recode +
                        ComCol.recode,
                        data=Final.table,
                        FUN=descriptivefun,
                        na.rm=TRUE,
                        keep.names=TRUE,
                        order=TRUE)

# prettyR::brkdn() Function
install.packages("prettyR")
library(prettyR)         # Load the prettyR package.
help(package=prettyR)    # Show the information page.
sessionInfo()            # Confirm all attached packages.

prettyR::brkdn(Final ~ Method.recode,
  data=Final.table,
  maxlevels=4,
  num.desc=c("mean", "sd", "valid.n"),
  width=25, round.n=2)

prettyR::brkdn(Final ~ ComCol.recode,
  data=Final.table,
  maxlevels=4,
  num.desc=c("mean", "sd", "valid.n"),
  width=25, round.n=2)
```

```
prettyR::brkdn(Final ~ (Method.recode +
                        ComCol.recode),
  data=Final.table,
  maxlevels=4,
  num.desc=c("mean","sd","valid.n"),
  width=25, round.n=2)

prettyR::brkdn(Final ~ (ComCol.recode +
                        Method.recode),
  data=Final.table,
  maxlevels=4,
  num.desc=c("mean","sd","valid.n"),
  width=25, round.n=2)
```

With the collapsed and breakout statistics completed, the data are now ready for further analyses, such as two-way ANOVA for this sample.

A listing and demonstration of other R functions related to descriptive statistics, either at the summary level or for breakout groups, could go on for pages. More examples are demonstrated in the remaining lessons.

2.5 Use R for Two-Way Analysis of Variance (ANOVA)

The data have now been brought into this R session, data were organized and labeled to accommodate human cognition, graphical images have been produced at the summary level and breakout levels, descriptive statistics were generated at the summary level and breakout levels, and data have also been organized into simple and complex contingency tables.

With all actions now in final form, the data are ready for inferential analyses, such as two-way ANOVA for this sample.

The preceding graphical images and descriptive statistics, both summary descriptive statistics and breakout descriptive statistics, provide a fairly good idea of the final exam values, overall and more importantly for the purpose of this two-way ANOVA sample by breakout groups (`Method.recode` and `ComCol.recode`).

The task now is to use two-way ANOVA, as supported in R by the many available packages and functions, to determine if there are statistically significant differences between (1) the four teaching method breakout groups, (2) the two Community College graduation status breakout groups, and (3) interaction between teaching method and Community College graduation status.

The descriptive statistics and graphical images serve as a basis for an initial suggestion that differences may or may not exist, but only an inferential test can provide the precise assessment needed for informed judgment.

R supports many possible ways to perform a two-way ANOVA. A few methods that support two-way ANOVA are detailed below.

2.5.1 Two-Way ANOVA: `aov()` Function

Use the formula for a two-way factorial design ANOVA, which is typically represented as:

```
fit1 <- aov(y ~ A + B + A:B, data=dataframe)
summary(fit1)

fit2 <- aov(y ~ A*B, data=dataframe)
summary(fit2)

y   = Measured datum, (e.g., weight, exam score, etc.)
A   = Factor Variable A (e.g., Gender, Race-Ethnicity, etc.)
B   = Factor Variable B (e.g., Soil Type, Breed Type, etc.)
A:B = Interaction of Factor Variable A and Factor Variable B
```

Both ANOVA formulas yield the same result, but the following formula (e.g., algorithm) is likely more illustrative:

```
fit1 <- aov(y ~ A + B + A:B, data=dataframe)

Sample1.fit1 <- aov(Final ~ Method.recode +
                            ComCol.recode +
                            Method.recode:ComCol.recode,
                data=Final.table)
summary(Sample1.fit1)
#                        Df Sum Sq Mean Sq F value Pr(>F)
# Method.recode           3 5372.3 1790.78 23.7512 1.388e-10 ***
# ComCol.recode           1  153.8  153.84  2.0404    0.1578
# Method.recode:
#   ComCol.recode         3  178.0   59.33  0.7869    0.5055
# Residuals              67 5051.6   75.40

Sample1.fit2 <- aov(Final ~ Method.recode*ComCol.recode,
                data=Final.table)
summary(Sample1.fit2)

#                        Df Sum Sq Mean Sq F value Pr(>F)
# Method.recode           3 5372.3 1790.78 23.7512 1.388e-10 ***
# ComCol.recode           1  153.8  153.84  2.0404    0.1578
# Method.recode:
#  ComCol.recode          3  178.0   59.33  0.7869    0.5055
# Residuals              67 5051.6   75.40
```

To gain a sense of the descriptive statistics (summary and breakout), use the `model.tables()` function for another view of Grand Mean, Mean, and *N* (`"rep"` for replication in this output) for each cell in the factorial table.

```
print(model.tables(Sample1.fit1,"means"), digits=3)
print(model.tables(Sample1.fit2,"means"), digits=3)
```

As a throwaway diagnostic, use the `plot.design()` function shown below to see general trends for final examination scores by each breakout group.

```
par(ask=TRUE)
plot.design(Final ~ Method.recode + ComCol.recode,
  data=Final.table,
  main="Final Exam Values by Teaching Method and
  Community College Graduation Status",
  lwd=3, font=2, cex.lab=1.50, cex.axis=1.50,
  font.main=2, font.lab=2, font.axis=2)
```

Then, use the `interaction.plot()` function for more details, but now focusing on the object variables of greatest interest.

```
par(ask=TRUE)
interaction.plot(Final.table$Method.recode,
                 Final.table$ComCol.recode,
                 Final.table$Final,
  main="Interaction Plot:  Instructional Method, Status as a
  Community College Graduate, and Final Examination Score",
  font.lab=2, col=2:9, lty="solid")
```

2.5.2 Outcome to Sample 1

The many R functions demonstrated in this sample have been used to `"hammer"` the data, to use an American expression. These many functions were used to demonstrate the important nuances of the data. It is a minimal set of analyses for a two-way ANOVA that only provides descriptive statistics, a simple plot or two, and the corresponding ANOVA output table with *F*-values and *p*-values. Data are too valuable and outcomes are too important to conduct only the minimal actions.

In conclusion for Sample 1, there are overall statistically significant difference (`p <= 0.05`) in final exam values by teaching method (view the three `***` symbols in the two-way ANOVA table, indicating that *p* is actually far less than 0.05). There was no observed statistically significant difference (`p <= 0.05`) in final exam values by Community College graduation status. Equally, there was no observed statistically significant interaction (`p <= 0.05`) between teaching method and Community College graduation status.

The following output (gained by using copy and paste) from the `doBy::summaryBy()` function provides a fairly good idea of where differences occur by teaching method.

```
# Method.recode Final.mean Final.sd
#       Lecture     76.500  12.2774
#           CBT     92.000   4.1675
#         Video     91.381   5.1037
#           IDS     73.000  11.4601
```

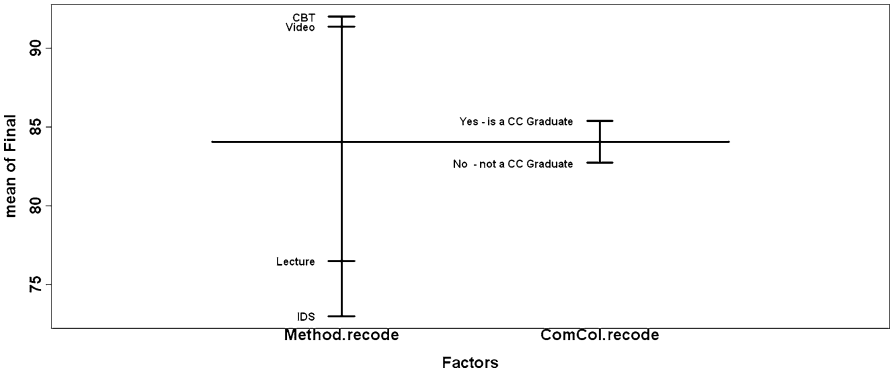

Fig. 2.1 Final exam values by Teaching Method and Community College Graduation status

However, to be precise, it is best to now apply the post hoc `TukeyHSD()` function, which is associated with one-way ANOVA, against final exam scores and teaching method (see Fig. 2.1).

```
Final.by.TeachingMethod.OnewayANOVA <-
  aov(Final ~ Method.recode, data=Final.table)

Final.by.TeachingMethod.OnewayANOVA

summary(Final.by.TeachingMethod.OnewayANOVA)

TukeyHSD(Final.by.TeachingMethod.OnewayANOVA)
  # Multiple comparisons.
  # This analysis accommodates missing data.

par(ask=TRUE)
plot(TukeyHSD(Final.by.TeachingMethod.OnewayANOVA),
  main="\n\nTukeyHSD Mean Comparison of Teaching Method
  and Final Exam Scores",
  cex.main=0.95, cex.lab=0.75, cex.axis=0.75,
  col.axis="darkblue", font.lab=2, font.axis=2, col="red")
  # Plot a graphical reinforcement of Mean Comparison
  # results, based on the TukeyHSD() function.
```

The output from use of the `TukeyHSD()` function provides the evidence needed to determine that there is a statistically significant difference (`p <= 0.05`) in final exam scores between: (1) lecture and CBT, (2) lecture and video, (3) IDS and CBT, and (4) IDS and video.

Conversely, there is no statistically significant difference (`p <= 0.05`) in final exam scores between: (1) IDS and lecture and (2) CBT and video.

```
#   Tukey multiple comparisons of means
#
```

```
# Fit: aov(formula = Final ~ Method.recode, data = Final.table)
#
# Method.recode
#                    diff       lwr      upr   p adj
# CBT-Lecture    15.50000   8.0569  22.9431 0.00000
# Video-Lecture  14.88095   7.5223  22.2396 0.00001
# IDS-Lecture    -3.50000 -11.3715   4.3715 0.64771
# Video-CBT      -0.61905  -7.7768   6.5387 0.99581
# IDS-CBT       -19.00000 -26.6840 -11.3160 0.00000
# IDS-Video     -18.38095 -25.9832 -10.7787 0.00000
```

Given these outcomes, it may be best to ignore any concern about prior status for students who are Community College graduates. Instead, focus should be given to teaching method. Assuming these outcomes stand the test of replication, efforts should be directed to teaching by use of CBT and video and efforts should be directed away from lecture and IDS.

Given the need for attention to policy considerations related to documentation of student learning outcomes, replication is a keyword in this broad conclusion. Outcomes across various scenarios, various faculty members, various courses, etc., would be needed before policy would ever be directed away from lecture as a desired teaching method. This sample lesson is merely one attempt in this overall assessment of student learning outcomes.

```
#################################################################
Prepare to Exit, Save, and Later Retrieve This R Session #
#################################################################

getwd()           # Identify the current working directory.
ls.str()          # List all objects, with finite detail.
save.image("Two-Way_ANOVA_Education-FinaExam.rdata")
list.files()      # List files at the PC directory.
q()               # Quit this session.
                  # Prepare for Save workspace image? query.
######################### END ###################################
```

Use the R graphical user interface (GUI) to load the saved .rdata file: File and then load workspace. Otherwise, use the `load()` function, keying the full pathname, to load the .rdata file and retrieve the session. Recall, however, that it may be just as useful to simply use the .R script file and recreate the analyses and graphics, provided the data files remain available.

http://www.springer.com/978-1-4614-2133-7

Two-Way Analysis of Variance
Statistical Tests and Graphics Using R
MacFarland, Th.W.
2012, VII, 150 p. 3 illus., 2 illus. in color., Softcover
ISBN: 978-1-4614-2133-7