# BIOL 4001 - Biostatistics

## R Tutorial 2: Graphics and summary statistics

### *Acknowledgment*

Thank you to Michael C. Whitlock and Dolph Schluter. This tutorial is based on their Labs using R, which accompanies their book, *The Analysis of Biological Data* (3rd edition).

### *Set working directory and download/open R script file*

Remember after starting R Studio to set the working directory by selecting Session > Set Working Directory > Choose Directory, e.g.:

```
setwd("~/Documents/BIOL 4001/RTutorials")
```

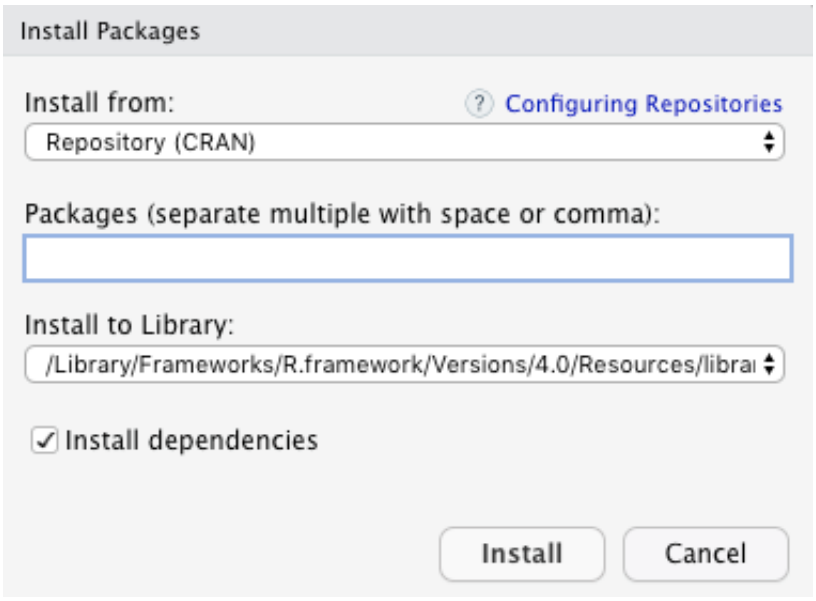
A script file containing all the code for this tutorial is available here: `RTutorial2.R`. Download it to the folder on your computer that you are using as your R working directory and then open it in R Studio by selecting File > Open. If you get stuck, remember you can use "Help" in R Studio for further information on the code used in this tutorial.

A zip file containing all the data files used in this tutorial is available here: `RTutorial2Data.zip`. Download it to the folder on your computer that you are using as your R working directory and uncompress.

### *R packages*

R packages are freely available collections of routines created by researchers for implementing just about any statistical procedure you can think of. We'll make use of a number of packages in this course. To install a package in R Studio click the "Packages" tab in the lower right window and then click the Install button 

. The following window should appear:




Type the name of the package into the middle box and click "Install." For example, we're going to use the package named `dplyr` in this tutorial, so follow the preceding instructions to install this package. You should see output in the Console confirming the download of this package.

Once a package has been downloaded on your computer, it will always be available (in other words, you only need to do the preceding install step once). However, every time you want to use the routines in the package, you have to load it into R using the `library` function:

```
library(dplyr)
```

R will confirm attachment of the package. Don't worry if you see a warning about object masking.

### *Reading data files*

As discussed in *Section 2.6 How to make data files* in the textbook, the easiest way to work with data in R is to use plain text files in which the data is stored in a "comma-separated variable" (CSV) format. The first row in a CSV file should be a "header" row with variable names separated by commas. Recall the syntax requirements for variable names from R Tutorial 1, particularly no spaces or special characters other than underscores. Each subsequent row should have the values of each variable for a given individual, again separated by commas.

The CSV datafile `chap02e2aDeathsFromTigers.csv` contains data for *Example 2.2A* on deaths from tigers. This is one of the files in the zip file `RTutorial2Data.zip` , which you saved earlier and uncompressed (do this now if you didn't do it earlier). Make sure the file `chap02e2aDeathsFromTigers.csv` is in your R working directory and not in any other folder or subfolder and use the `read.csv` function to load the data into R:

```
tigerData <- read.csv("chap02e2aDeathsFromTigers.csv")
```

If you get an error message that the file cannot be opened, it is probably because R cannot find the file in the working directory. Check the data has loaded correctly by using the `summary` function:

```
summary(tigerData)
```

which should result in:

```
     person        activity
 Min.   : 1.00   Length:88
 1st Qu.:22.75   Class :character
 Median :44.50   Mode  :character
 Mean   :44.50
 3rd Qu.:66.25
 Max.   :88.00
```

View the first 6 rows of the data by using the `head` function:

```
head(tigerData)
```

which should result in:

```
  person              activity
1      1 Disturbing tiger kill
2      2       Forest products
3      3          Grass/fodder
4      4       Fuelwood/timber
5      5          Grass/fodder
6      6       Forest products
```

### *Data frames*

One way to work with data in R is to use *data frames*, which you can visualize as a spreadsheet with individuals as the rows and variables as the columns. For example, `tigerData` is a dataframe with 88 rows (representing 88 people killed by tigers) and 2 columns (representing two variables: person, a number that labels each person consecutively from 1 to 88, and activity, a categorical variable describing each person's activity when they were killed).

Variables in data frames are referenced by writing the name of the data frame, then a $, and then the variable name. For example, if you run `tigerData$activity`, R will output all the sample values of this variable. To create a frequency table for this variable, run the following code:

```
tigerTable <- table(tigerData$activity)
tigerTable
```

### *Creating a new variable in a data frame*

We can create a new variable in a data frame by assigning a new vector to a new variable name appended to the data frame name with a $. For example, the following code creates a new variable `disturbAtKill` that has the value "yes" for people who disturbed the tiger at its kill and "no" otherwise:

```
tigerData$disturbAtKill <-
  ifelse(tigerData$activity == "Disturbing tiger kill", "yes", "no")
```

Note the use of the "double equals sign" to evaluate whether a statement is true or false. Now if you run `summary(tigerData)`, you'll see a new variable added to the data frame called `disturbAtKill`, and you can see its values if you run `tigerData$disturbAtKill`.

### *Selecting subsets of a data frame*

One way to select a subset of a data frame is to use the filter function in the `dplyr` package. For example, the following code creates a new data frame from the `tigerData` data frame with everyone except those people who disturbed the tiger at its kill:

```
tigerDataNotAtKill <- filter(tigerData, DisturbAtKill == "no")
```

Now if you run `table(tigerDataNotAtKill$activity)`, you'll see a summary table for only those people who did not disturb the tiger at its kill.

### *Factor variables*

As noted in *Section 2.5 How to make good tables*, we can improve the frequency table of activities by ordering the activities according to their frequencies. One way to do this is to use the following code to create a new "factor" variable to represent activities ordered by their frequencies:

```
tigerData$activity_ordered <- factor(tigerData$activity,
                                     levels = names(sort(tigerTable,
                                                         decreasing = TRUE)))
```

Now if you run `summary(tigerData)`, you'll see a new factor variable added to the data frame called `activity_ordered`, and if you run `table(tigerData$activity_ordered)`, you'll see the table has been reordered according to the activity frequencies.

### *Create graphs with ggplot2 package*

The `ggplot2` package makes creating elegant data visualizations easy. Install the `ggplot2` package and load it using `library(ggplot2)`. Also run `theme_set(theme_classic())` to set

the default visual look of graphs created in ggplot2 to a classic-looking theme, with x and y axis lines and no gridlines.

### *Bar charts*

The following code creates a bar graph for the tiger data similar to Figure 2.2-1 in the textbook:

```
ggplot(data = tigerData, aes(x = activity_ordered)) +
  geom_bar(stat = "count") +
  labs(x = "Activity", y = "Frequency") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```

### *Frequency table and histogram*

Use the `read.csv` function to load `chap02e2bZikaBiparietalDiameter.csv`, which contains data for *Example 2.2B* on the effects of Zika virus. The following code creates a frequency table similar to Table 2.2-2 and a histogram similar to Figure 2.2-3:

```
zikaTable <- table(cut(zikaData$biparietalDiamMm,
                       breaks = seq(60, 95, by=5), right = FALSE))
zikaTable

ggplot(data = zikaData, aes(x = biparietalDiamMm)) +
  geom_histogram(col = "black", binwidth = 2,
                 boundary = 0, closed = "left") +
  labs(x = "Head diamater (mm)", y = "Frequency")
```

### *Contingency table, grouped bar graph, and mosaic plot*

Use the `read.csv` function to load `chap02e3aBirdMalaria.csv`, which contains data for *Example 2.3A* on bird malaria. The following code creates a contingency table similar to Table 2.3-1, a grouped bar graph similar to Figure 2.3-1, and a mosaic plot similar to Figure 2.3-2:

```
birdMalariaTable <- table(birdMalariaData$response,
                          birdMalariaData$treatment)
birdMalariaTable
addmargins(birdMalariaTable, margin = c(1,2), FUN = sum, quiet = TRUE)

ggplot(birdMalariaData, aes(x = treatment, fill = response)) +
  geom_bar(stat = "count", position = position_dodge2(preserve="single")) +
  labs(x = "Treatment", y = "Frequency")

mosaicplot(t(birdMalariaTable), sub = "Treatment",
           ylab = "Relative frequency", main = "")
```

*Scatter plot*

Use the `read.csv` function to load `chap02f3_3GuppyFatherSonAttractiveness.csv`, which contains data on male guppies and their sons. The following code creates a scatter plot similar to Figure 2.3-3:

```
ggplot(data = guppyFatherSonData,
       aes(x = fatherOrnamentation, y = sonAttractiveness)) +
  geom_point(size = 2) +
  labs(x = "Father's ornamentation", y = "Son's attractiveness")
```

*Strip chart, violin plot, and multiple histograms*

Use the `read.csv` function to load `chap02e3bHumanHemoglobinElevation.csv`, which contains data for *Example 2.3B* on human hemoglobin. The following code creates a strip chart and violin plot similar to Figure 2.3-4 and multiple histograms similar to Figure 2.3-5:

```
ggplot(data = hemoglobinData, aes(x = population, y = hemoglobin)) +
  geom_jitter(shape = 1, size = 2, width = 0.15) +
  labs(x = "Male population", y = "Hemoglobin concentration (g/dL)")

ggplot(data = hemoglobinData, aes(y = hemoglobin, x = population)) +
  geom_violin() +
  labs(x = "Male population", y = "Hemoglobin concentration (g/dL)") +
  stat_summary(fun = mean,  geom = "point")

ggplot(data = hemoglobinData, aes(x = hemoglobin)) +
  geom_histogram(binwidth = 1, col = "black",
                 boundary = 0, closed = "left") +
  labs(x = "Hemoglobin concentration (g/dL)", y = "Frequency") +
  facet_wrap(~ population, ncol = 1, scales = "free_y",
             strip.position = "right")
```

*Line graph*

Use the `read.csv` function to load `chap02f4_1MeaslesOutbreaks.csv`, which contains data on measles cases. The following code creates a line graph similar to Figure 2.4-1:

```
ggplot(data = measlesData, aes(x = yearByQuarter, y = confirmedCases)) +
  geom_line() +
  geom_point() +
  labs(x = "Year", y = "Number of new measles cases")
```

*Histogram, mean, variance, standard deviation, coefficient of variation*

Use the `read.csv` function to load `chap03e1GlidingSnakes.csv`, which contains data for *Example 3.1* on gliding snakes. The following code creates a histogram similar to Figure 3.1-1 and calculates the mean, variance, standard deviation, and coefficient of variation.

```
ggplot(data = snakeData, aes(x = undulationRateHz)) +
  geom_histogram(col = "black",
                 breaks = seq(0.8, 2.2, by = 0.2),
                 boundary = 0, closed = "left") +
  labs(x = "Undulation rate (Hz)", y = "Frequency")

mean(snakeData$undulationRate)
var(snakeData$undulationRate)
sd(snakeData$undulationRate)

100 * sd(snakeData$undulationRate)/mean(snakeData$undulationRate)
```

*Calculating summary statistics from a frequency table*

Use the `read.csv` function to load `chap03t1.2ConvictionsFreq.csv`, which contains the frequency data on criminal convictions in Table 3.1-2. The following code recreates the full data from which we can calculate the mean and standard deviation:

```
convictions <- rep(convictionsFreq$convictions, convictionsFreq$frequency)

mean(convictions)
sd(convictions)
```

*Median, quartiles, interquartile range, boxplots, cumulative frequency distribution*

Use the `read.csv` function to load `chap03e2SpiderAmputation.csv`, which contains data for *Example 3.2* on spider running speed. The following code calculates the median, quartiles, and interquartile range of the "before" speeds, and creates boxplot similar to Figure 3.2-2 and a cumulative frequency distribution similar to Figure 3.4-1.

```
# select subset of before measurements
spiderDataBefore <- filter(spiderData, treatment == "before")

median(spiderDataBefore$speed)
quantile(spiderDataBefore$speed, probs = c(0.25, 0.75), type = 5)
IQR(spiderDataBefore$speed, type = 5)

# reorder treatment levels so "before" comes before "after"
spiderData$treatment <- factor(spiderData$treatment,
                               levels = c("before", "after"))
```

```
ggplot(spiderData, aes(x = treatment, y = speed)) +
  geom_boxplot()

ggplot(spiderDataBefore, aes(x = speed)) +
  stat_ecdf() +
  labs(x = "Running speed before amputation (cm/s)",
       y = "Cumulative relative frequency")
```

*Multiple histograms and summary statistics by group*

Use the `read.csv` function to load `chap03e3SticklebackPlates.csv`, which contains data for *Example 3.3* on stickleback lateral plates. The following code creates multiple histograms similar to Figure 3.3-1 and calculates summary statistics by group.

```
# reorder genotype levels
sticklebackData$genotype <- factor(sticklebackData$genotype,
                                   levels = c("MM","Mm","mm"))

ggplot(sticklebackData, aes(x = plates)) +
  geom_histogram(col = "black",
                 boundary = 0, closed = "left", binwidth = 2) +
  labs(x = "Number of lateral plates", y = "Frequency") +
  facet_wrap(~ genotype, ncol = 1, scales = "free_y")

sticklebackTable <- summarise(group_by(sticklebackData, genotype),
                              n = n(),
                              Mean = round(mean(plates), digits = 1),
                              Median = median(plates),
                              Std.dev = round(sd(plates), digits = 1),
                              Interq.range = IQR(plates, type = 5))
sticklebackTable$Proportion <-
  round(sticklebackTable$n/sum(sticklebackTable$n), 2)
sticklebackTable
```