Contents

This text is a companion to *Introductory Statistics for the Life and Biomedical Sciences*; while the main text focuses on statistical concepts and ideas, this supplement provides details about how to use the statistical computing language R.

All the datasets used in the text can be accessed by downloading the OIBiostat package from R. Run the following command to download the package:

```
install.packages("OIBioStat")  ## make sure to include the quotations
```

The above command only needs to be run once. Each time a dataset in the package is needed, run the following command:

```
require(OIBioStat) ## note the lack of quotations here
```

This chapter introduces basic commands for manipulating datasets, including how to calculate numerical summaries, create tables from data, and make graphical plots.

1 Case study: preventing peanut allergies

The `LEAP` dataset contains the results of the "Learning Early About Peanut Allergy" (LEAP) study, an experiment conducted to assess whether early exposure to peanut products reduces the probability of peanut allergies developing. To access the documentation associated with the dataset, run the following command:

```
help(LEAP)
```

A file will appear in the Help pane that provides some basic information about the dataset:

- **Description:** A general overview of the dataset.
- **Usage:** Instructions for how to load the dataset.
- **Format:** The names and descriptions of each variable in the dataset, including information about variable type and measurement units.
- **Details:** Additional details about the conditions under which the data were collected.
- **Source**: Information about where the data originates from.

To view the dataset itself, run:

```
View(LEAP)
```

In the main console pane, a window will appear that shows the entire dataset. The variable names are displayed across the top, as the names of the columns. Data for each case in the study are contained within the rows. Note that the farthest left column shows the **indices**, which are computer-generated values that allow specific rows in the dataset to be accessed. These can be used to view a specific portion of the data.

For example, to print out data contained in the first 5 rows and first 3 columns of the data, use the following syntax:

```
LEAP[1:5,1:3]
```

```
##   participant.ID    treatment.group age.months
## 1    LEAP_100522 Peanut Consumption     6.0780
## 2    LEAP_103358 Peanut Consumption     7.5893
## 3    LEAP_105069   Peanut Avoidance     5.9795
## 4    LEAP_105328 Peanut Consumption     7.0308
## 5    LEAP_106377   Peanut Avoidance     6.4066
```

The bracket notation after the dataset name implies location; the syntax `dataset[rows,columns]` specifies rows 1 through 5 and columns 1 through 6. To access only the first 5 rows, but all the columns, leave an empty space where the columns would usually be specified:

```
## note the space (or lack of text after the comma)
LEAP[1:5, ]

##   participant.ID    treatment.group age.months    sex primary.ethnicity
## 1    LEAP_100522 Peanut Consumption     6.0780 Female             Black
## 2    LEAP_103358 Peanut Consumption     7.5893 Female             White
## 3    LEAP_105069   Peanut Avoidance     5.9795   Male             White
## 4    LEAP_105328 Peanut Consumption     7.0308 Female             White
## 5    LEAP_106377   Peanut Avoidance     6.4066   Male             White
##   overall.V60.outcome
## 1            PASS OFC
## 2            PASS OFC
## 3            PASS OFC
## 4            PASS OFC
## 5            PASS OFC
```

Alternatively, since there are 6 columns in the dataset, the command `LEAP[1:5,1:6]` would also achieve the same result. This is a common theme in R – there can be several ways to accomplish a desired result.

OI Biostat Table 1.1 shows the participant ID, treatment group, and overall outcome for five patients. It is not specified in the main text, but the data are specifically from rows 1, 2, 3, 529, and 530. The command `c()` can be used to bind the row numbers into a list, as well as to create a list of the desired columns. Columns can be referred to by name, instead of by number:

```
## OI Biostat Table 1.1
LEAP[c(1, 2, 3, 529, 530),c("participant.ID", "treatment.group",
                            "overall.V60.outcome")]

##     participant.ID    treatment.group overall.V60.outcome
## 1      LEAP_100522 Peanut Consumption            PASS OFC
## 2      LEAP_103358 Peanut Consumption            PASS OFC
## 3      LEAP_105069   Peanut Avoidance            PASS OFC
## 639    LEAP_994047   Peanut Avoidance            PASS OFC
## 640    LEAP_997608 Peanut Consumption            PASS OFC
```

Two-way summary tables organize the data according to two variables and display the number of counts matching each combination of variable categories. The following code corresponds to *OI Biostat* Table 1.2, which groups participants into categories based on treatment group and overall outcome. In the `table()` command, the first variable specifies the rows and the second variable

specifies the columns. The addition of the `addmargins()` command prints the sums of the rows and columns on the sides of the table.

```
## Table 1.2
table(LEAP$treatment.group, LEAP$overall.V60.outcome)

##
##                        FAIL OFC PASS OFC
##   Peanut Avoidance          36       227
##   Peanut Consumption         5       262

addmargins(table(LEAP$treatment.group, LEAP$overall.V60.outcome))

##
##                        FAIL OFC PASS OFC Sum
##   Peanut Avoidance          36       227 263
##   Peanut Consumption         5       262 267
##   Sum                       41       489 530
```

2 Data Basics

Entire datasets can be partitioned into data frames using bracket notation. For example, *OI Biostat* Table 1.3 shows a data frame consisting of rows 1-3 and 150 (and all columns) from the `frog.altitude` dataset. The data frame can then be given a specific name, such as `frog.df`, and directly called on for later operations.

```
## Table 1.3
frog.df = frog.altitude.data[c(1:3, 150),]
frog.df

##     altitude latitude clutch.size body.size clutch.volume egg.size
## 1   3,462.00    34.82    181.9701  3.630781      177.8279 1.949845
## 2   3,462.00    34.82    269.1535  3.630781      257.0396 1.949845
## 3   3,462.00    34.82    158.4893  3.715352      151.3561 1.949845
## 150 2,597.00    34.05    537.0318        NA      776.2471 2.238721
```

Similarly, matrix notation can be used to create *OI Biostat* Table 1.5.

```
## Table 1.5
famuss[c(1,2,3,595),c( "sex", "age", "race", "height", "weight", "actn3.r577x",
                       "ndrm.ch")]

##         sex age      race height weight actn3.r577x ndrm.ch
## 1    Female  27 Caucasian   65.0    199          CC    40.0
```

```
## 2      Male  36 Caucasian   71.7    189          CT    25.0
## 3    Female  24 Caucasian   65.0    134          CT    40.0
## 1348 Female  30 Caucasian   64.0    134          CC    43.8
```

4 Numerical Data

Numerical summaries can be quickly and easily calculated using R. The `summary()` command is one way to access several numerical summaries at once, including the minimum and maximum values of a variable:

```
summary(frog.altitude.data$clutch.volume)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   151.4   609.6   831.8   882.5  1096.0  2630.0
```

4.1 Measures of center: mean and median

The **mean** of a numerical value is the sum of all observations divided by the number of observations, where $x_1, x_2, \ldots, x_n$ represent the n observed values:

$\overline{x} = \frac{x_1+x_2+\cdots+x_n}{n}$

This calculation can be completed in R the same way as a hand calculation is done:

```
# identify n, the number of observations in the data
n = length(frog.altitude.data$clutch.volume)
n

## [1] 431

# calculate the sum of all observations and divide by n
sum(frog.altitude.data$clutch.volume)/n

## [1] 882.474
```

Alternatively, the R function `mean()` can be directly applied to the variable of interest:

```
x.bar = mean(frog.altitude.data$clutch.volume)
x.bar

## [1] 882.474

## round to the first decimal
round(x.bar, 1)

## [1] 882.5
```

To identify the **median** value, use the following command:

```
median(frog.altitude.data$clutch.volume)

## [1] 831.7638
```

4.2 Measures of spread: standard deviation and variance

The distance of a single observation from the mean is its **deviation**. The following command produces the deviations for the 1^{st}, 2^{nd}, 3^{rd}, and 431^{th} observations in the `clutch.volume` variable.

```
frog.altitude.data$clutch.volume[c(1,2,3,431)]-x.bar

## [1] -704.64604 -625.43440 -731.11786   50.78032
```

The sample **standard deviation** is computed as the square root of the **variance**, which is the sum of squared deviations divided by the number of observations minus 1.

$s = \sqrt{\frac{(x_1-\overline{x})^2+(x_2-\overline{x})^2+\cdots+(x_n-\overline{x})^2}{n-1}}$

The following steps illustrate how to calculate the variance and standard deviation using the formula:

```
# calculate all deviations
dev = frog.altitude.data$clutch.volume - x.bar

# sum the squares of the deviations and divide by n - 1
var = (sum(dev^2))/(n-1)
var

## [1] 143680.9

# take the square root of the variance
sd = sqrt(var)
sd

## [1] 379.0527
```

Alternatively, use the R functions `var()` and `sd()`:

```
var(frog.altitude.data$clutch.volume)

## [1] 143680.9

sd(frog.altitude.data$clutch.volume)

## [1] 379.0527
```

Variability can also be measured using the **interquartile range (IQR)**, which equals the third quartile (the 75^{th} percentile) minus the first quartile (the 25^{th} percentile).

```
IQR(frog.altitude.data$clutch.volume)

## [1] 486.9009
```

4.3 Robust statistics

In the `frog.altitude` dataset, there are four extreme values for clutch volume that are larger than 2,000 mm^3. To illustrate how the summary statistics are influenced by extreme values, *OI Biostat* Table 1.15 shows summary statistics for the data without the four largest observations.

The subset of the data with values less than 2,000 mm^3 can be pulled out by first specifying a logical condition, which assigns `TRUE` or `FALSE` to each entry.

```
# logical condition
less.than.2000 = frog.altitude.data$clutch.volume<= 2000

# view the first 5 values
less.than.2000[1:5]

## [1] TRUE TRUE TRUE TRUE TRUE
```

Bracket notation can then be used to calculate summary statistics specifically for the values in `clutch.volume` that satisfy the logical condition.

```
## robust estimates
median(frog.altitude.data$clutch.volume[less.than.2000])

## [1] 831.7638

IQR(frog.altitude.data$clutch.volume[less.than.2000])

## [1] 493.9186

## non-robust estimates
mean(frog.altitude.data$clutch.volume[less.than.2000])

## [1] 867.9425

sd(frog.altitude.data$clutch.volume[less.than.2000])

## [1] 349.1596
```

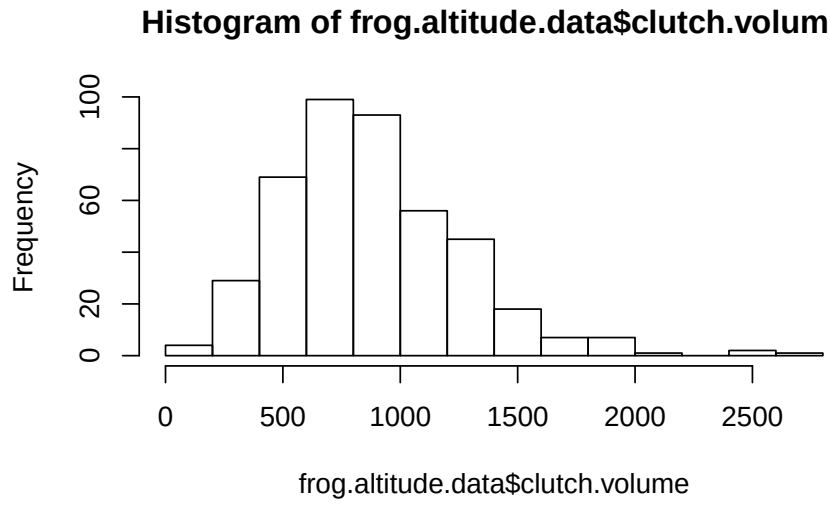
4.4 Visualizing distributions of data: histograms and boxplots

While most numerical summaries can be calculated by hand, R is essential for creating graphical summaries.

4.4.1 Histograms

A **histogram** provides a view of data density. Observations are sorted into different bins based on their value, and the histogram shows the number of observations in each bin. The following command produces a histogram of the `clutch.volume` variable.

```
hist(frog.altitude.data$clutch.volume)
```



The `hist()` function takes several arguments:

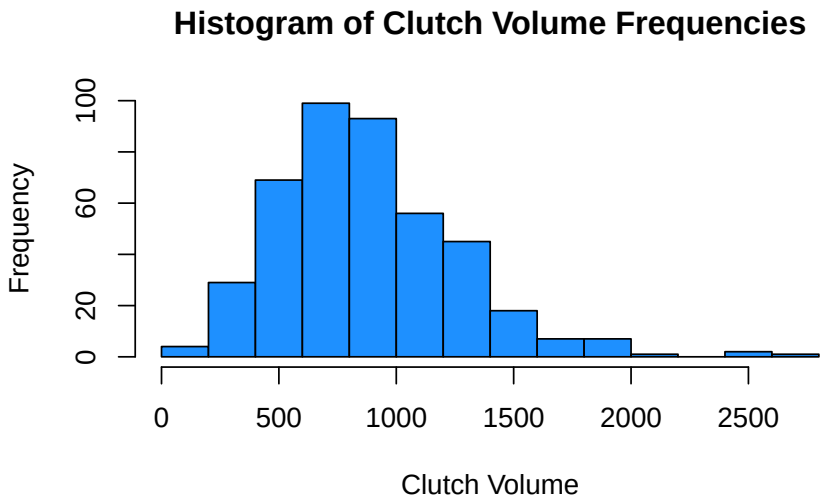
- `x`: variable of interest
- `breaks`: number of bins
- `col`: color of the bars, enclosed in `""`
- `xlab`: x-axis label, enclosed in `""`
- `ylab`: y-axis label, enclosed in `""`
- `xlim`: range of values for the x-axis, in the form `c(lower bound, upper bound)`

- `ylim`: range of values for the y-axis, in the form `c(lower bound, upper bound)`
- `main`: main title of the plot, enclosed in `""`
- `plot`: if `TRUE` (default), a histogram is plotted; otherwise, data about the histogram is returned

As seen above, not all arguments must be specified; only the `x` argument is necessary. When options are not specified, R uses the default options, such as the default color of white for the histogram bars.

The following command reproduces *OI Biostat* Table 1.16 and Figure 1.17.

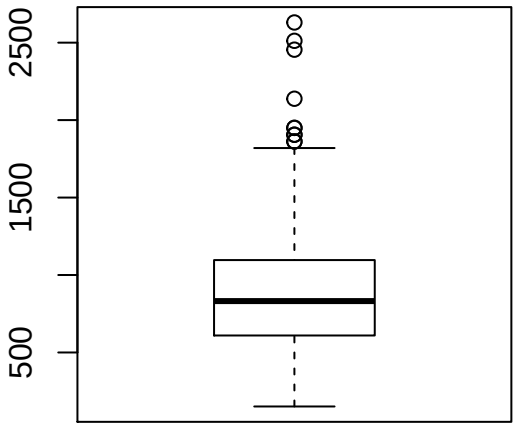
```
## Table 1.16
hist(frog.altitude.data$clutch.volume, breaks = 14, plot = FALSE)$counts

##  [1]  4 29 69 99 93 56 45 18  7  7  1  0  2  1

## Figure 1.17
hist(x = frog.altitude.data$clutch.volume, breaks = 14, col = "dodgerblue",
     xlab = "Clutch Volume", ylab = "Frequency", ylim = c(0, 100),
     main = "Histogram of Clutch Volume Frequencies")
```



4.4.2 Boxplots

A **boxplot** uses five statistics to summarize a dataset, in addition to showing unusual observations. The following command produces a boxplot of the `clutch.volume` variable.

```
boxplot(frog.altitude.data$clutch.volume)
```

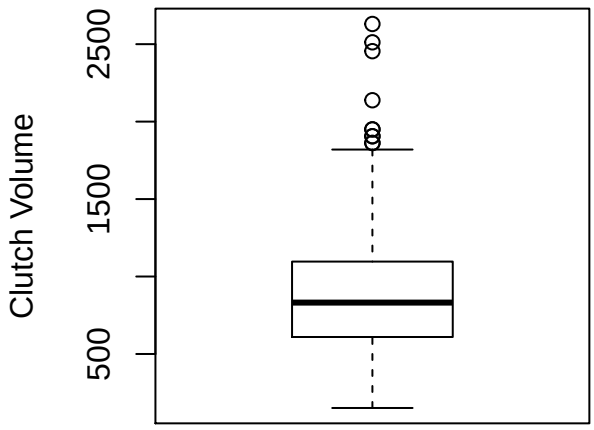


Like the `hist()` function, the `boxplot()` function also takes several arguments that allow for certain parameters to be specified:

- `x`: variable of interest
- `axes`: if `TRUE`, numbers are shown on the axes
- `col`: color of the boxplot, enclosed in `""`
- `xlab`: x-axis label, enclosed in `""`
- `ylab`: y-axis label, enclosed in `""`
- `xlim`: range of values for the x-axis, in the form `c(lower bound, upper bound)`
- `ylim`: range of values for the y-axis, in the form `c(lower bound, upper bound)`
- `main`: main title of the plot, enclosed in `""`

The following command reproduces a simplified version of *OI Biostat* Figure 1.19.

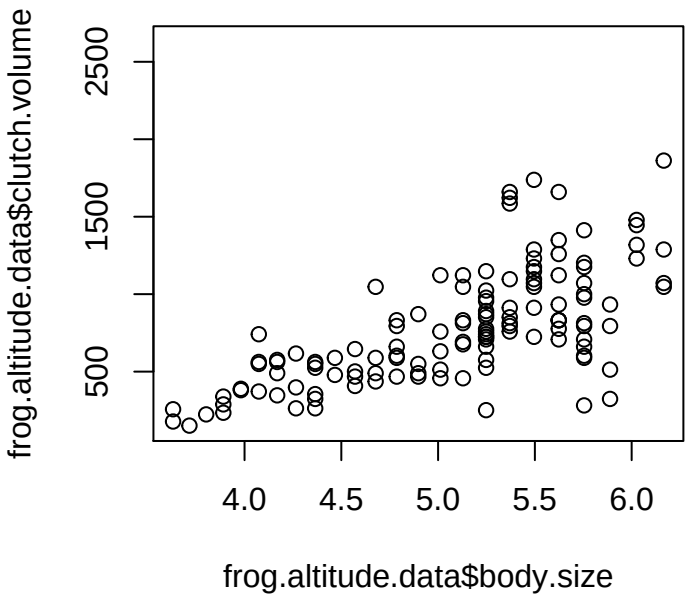
```r
## Figure 1.19
boxplot(x = frog.altitude.data$clutch.volume, ylab = 'Clutch Volume', axes = TRUE,
        ylim = range(frog.altitude.data$clutch.volume))
```



4.5 Scatterplots and correlation

Scatterplots can be used to visualize the relationship between two numerical variables. In the `plot()` command, either a comma or a tilde can be used between the variable names; i.e., `plot(x,y)` versus `plot(y ~ x)`.

```r
plot(frog.altitude.data$body.size, frog.altitude.data$clutch.volume)
```

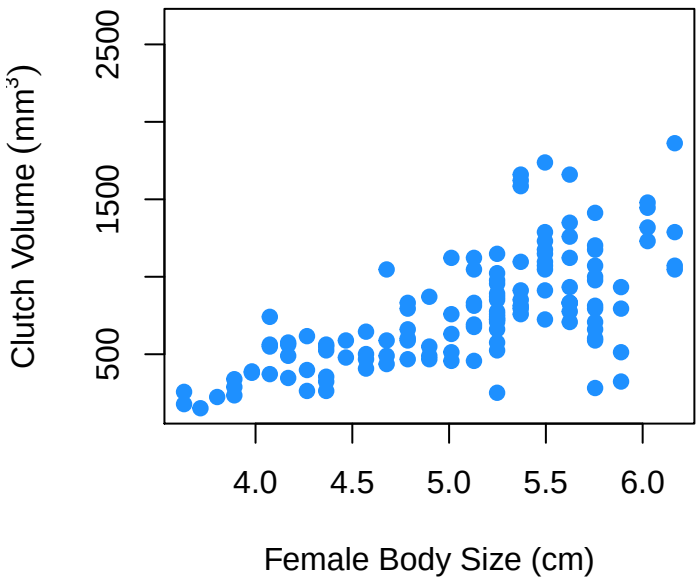


The `plot()` command takes the following arguments:

- `x`: variable defining the x-coordinates
- `y`: variable defining the y-coordinates
- `col`: color of the dots, enclosed in `""`
- `type`: by default, data points are marked with dots; other options include `"l"` which draws a line chart, or `"b"` which includes both dots and lines
- `xlab`: x-axis label, enclosed in `""`
- `ylab`: y-axis label, enclosed in `""`
- `xlim`: range of values for the x-axis, in the form `c(lower bound, upper bound)`
- `ylim`: range of values for the y-axis, in the form `c(lower bound, upper bound)`
- `main`: main title of the plot, enclosed in `""`

The following command reproduces a simplified version of *OI Biostat* Figure 1.20. The additional argument `pch` changes the appearance of the dots to filled dots, which is specified by `19`.

```
## Figure 1.20
plot(frog.altitude.data$clutch.volume~frog.altitude.data$body.size, col = "dodgerblue",
     pch = 19, xlab = "Female Body Size (cm)", ylab = expression("Clutch Volume" ~ (mm^3)))
```



Simplified versions of *OI Biostat* Figures 1.21, 1.22, and 1.23 can also be reproduced using `plot()`.

4.5.1 Correlation

Correlation is a numerical measure of the strength of a linear relationship. The formula for correlation uses the pairing of the two variables, just as the scatterplot does in a graph, so the data used in calculating correlation is a set of n ordered pairs $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$.

The correlation between two variables x and y is given by: $r = \frac{1}{n-1}\sum_{i=1}^{n}\left(\frac{x_i-\bar{x}}{s_x}\right)\left(\frac{y_i-\bar{y}}{s_y}\right)$

where $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ are the n paired values of x and y, and s_x and s_y are the sample standard deviations of the x and y variables, respectively.

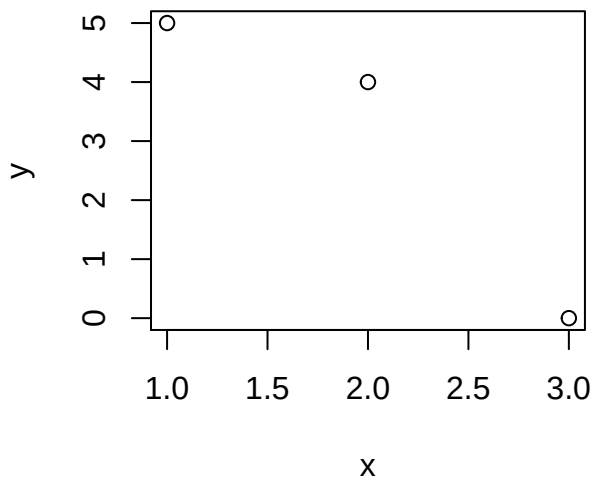
The following illustrates the calculations for finding the correlation coefficient of (1,5), (2, 4), and (3,0), as shown in *OI Biostat* Example 1.13.

```
# define variables
x = c(1, 2, 3)
```

```r
y = c(5, 4, 0)

# calculate sample means
x.bar = mean(x)
y.bar = mean(y)

# calculate sample sd's
sd.x = sd(x)
sd.y = sd(y)

# calculate products and sum of products
x.component = (x - x.bar)/(sd.x)
y.component = (y - y.bar)/(sd.y)
products = x.component * y.component
products.sum = sum(products)

# divide by n - 1
n = length(x)   ## n also equals length(y)
r = products.sum / (n - 1)
r

## [1] -0.9449112
```

It is much easier to use the `cor()` function to find correlation.

```r
## Figure 1.22
cor(x,y)

## [1] -0.9449112

plot(x,y)
```



The correlation can also be calculated for the income versus life expectancy data plotted in *OI Biostat* Figure 1.23. In this case, the command includes the argument `use = "complete.obs"` to allow for the computation to disregard any missing values in the dataset.

```
cor(life.expectancy.income$income, life.expectancy.income$life.expectancy,
    use = "complete.obs")

## [1] 0.6308783
```

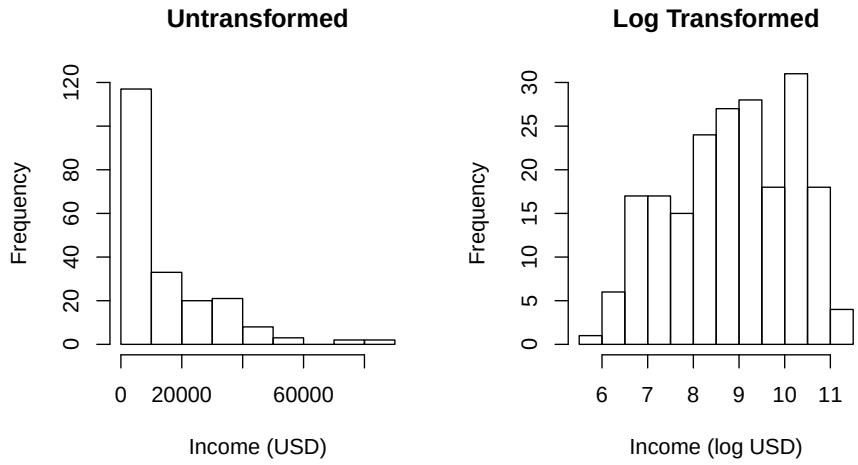
4.6 Transforming Data

A **transformation** is a rescaling of data using a function. The figure below shows the original plot of income data as well as the plot of the log-transformed data. Note that the `log` command in R computes natural logarithms.

The function `par()` creates partitions in the graphing output. In this case, specifying `mfrow = c(1,2)` produces 1 row and 2 columns.

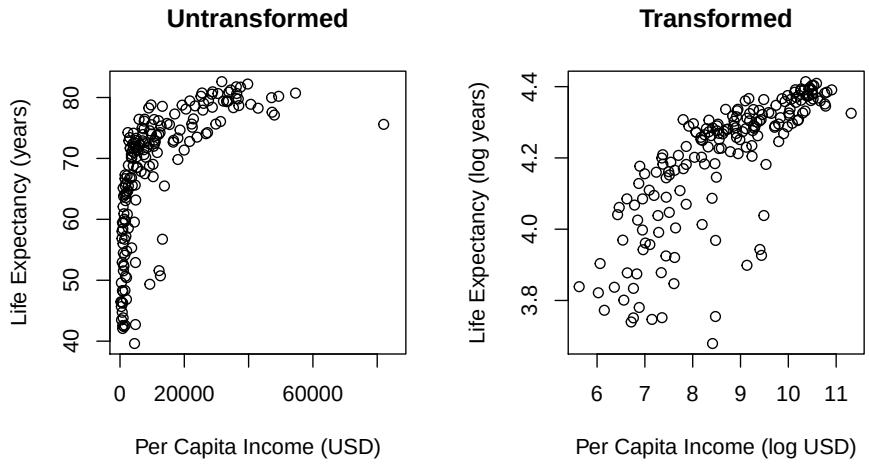
```
## Figure 1.26
par(mfrow = c(1, 2))  ## this line allows two plots to print at the same time
hist(life.expectancy.income$income, breaks = 12, xlab = "Income (USD)",
    ylab = "Frequency", ylim = c(0, 120), main = "Untransformed")

hist(log(life.expectancy.income$income), breaks = 12, xlab = "Income (log USD)",
```

```
    ylab = "Frequency", ylim = c(0, 30), main = "Log Transformed")
```



Transformations can also be applied to both variables when exploring a relationship. The following figure shows simplified versions of *OI Biostat* Figures 1.23 and 1.27.

```
## Figure 1.23 and Figure 1.27
par(mfrow = c(1, 2))
plot(life.expectancy.income$income, life.expectancy.income$life.expectancy,
    ylab = "Life Expectancy (years)", xlab = "Per Capita Income (USD)",
    main = "Untransformed")
plot(log(life.expectancy.income$income), log(life.expectancy.income$life.expectancy),
    ylab = "Life Expectancy (log years)", xlab = "Per Capita Income (log USD)",
    main = "Transformed")
```



5 Categorical Data

5.1 Contingency Tables

A **frequency table** shows the counts for each category within a variable. The following `table()` command produces a frequency table for the `actn3.r577x` variable.

```
## Table 1.28
addmargins(table(famuss$actn3.r577x))

##
##  CC  CT  TT Sum
## 173 261 161 595
```

A **contingency table** summarizes data for two categorical variables, such as race and genotype in *OI Biostat* Table 1.29.

```
## Table 1.29
addmargins(table(famuss$race, famuss$actn3.r577x))

##
##                CC  CT  TT Sum
##   African Am   16   6   5  27
##   Asian        21  18  16  55
##   Caucasian   125 216 126 467
##   Hispanic      4  10   9  23
##   Other         7  11   5  23
##   Sum         173 261 161 595
```

OI Biostat Table 1.30 shows a contingency table with row proportions, computed as the counts divided by their row totals. The `prop.table()` command produces a table with row proportions, as specified by the `1` in the argument.

```
## Table 1.30
counts.table = table(famuss$race, famuss$actn3.r577x)
row.prop.table = prop.table(counts.table, 1)
row.prop.table

##
##                      CC        CT        TT
##   African Am 0.5925926 0.2222222 0.1851852
##   Asian      0.3818182 0.3272727 0.2909091
##   Caucasian  0.2676660 0.4625268 0.2698073
##   Hispanic   0.1739130 0.4347826 0.3913043
##   Other      0.3043478 0.4782609 0.2173913
```

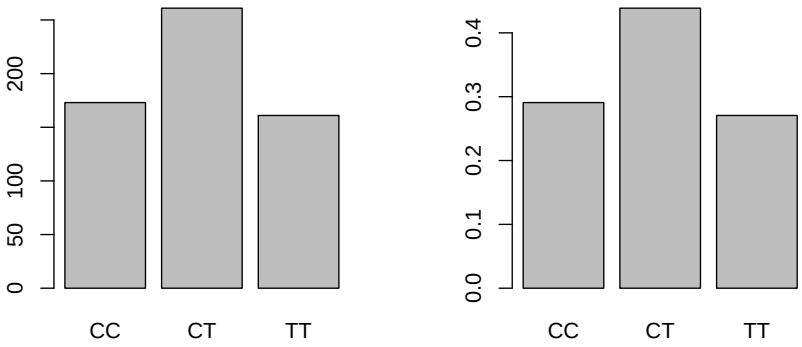
Alternatively, a contingency table can be created with column proportions by changing the `1` in `prop.table()` to a `2`.

```
## Table 1.31
counts.table = table(famuss$race, famuss$actn3.r577x)
col.prop.table = prop.table(counts.table, 2)
col.prop.table

##
##                       CC         CT         TT
##   African Am 0.09248555 0.02298851 0.03105590
##   Asian      0.12138728 0.06896552 0.09937888
##   Caucasian  0.72254335 0.82758621 0.78260870
##   Hispanic   0.02312139 0.03831418 0.05590062
##   Other      0.04046243 0.04214559 0.03105590
```

5.2 Bar Plots

A **bar plot** is a common way to display a single categorical variable, either from count data or from proportions. The `barplot()` command requires values to be input in a table format. In the below example, the `table()` command is nested within the `barplot()` command.

```
## Figure 1.32
par(mfrow = c(1, 2))
barplot(table(famuss$actn3.r577x))  ## count barplot
barplot(table(famuss$actn3.r577x)/sum(table(famuss$actn3.r577x)))  ## frequency barplot
```

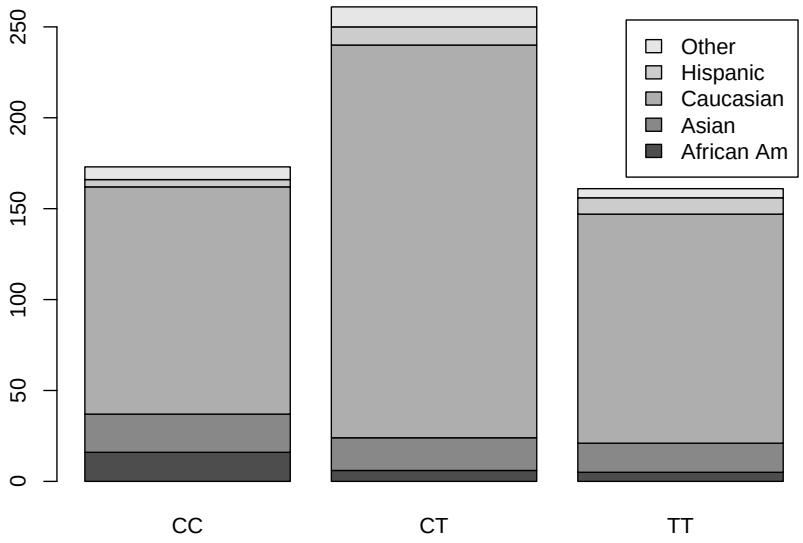

0
50
100
200
CC
CT
TT
0.0
0.1
0.2
0.3
0.4
CC
CT
TT

5.2.1 Segmented Bar Plot

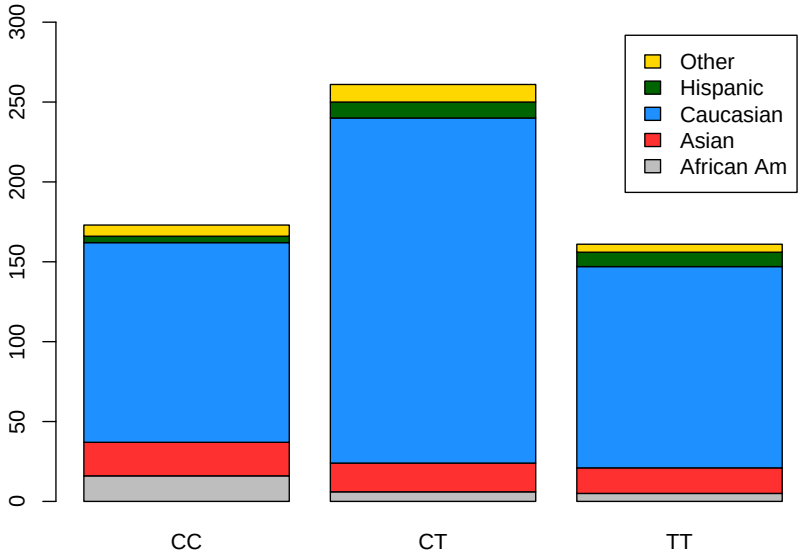
A **segmented bar plot** draws from a contingency table to display information about two categorical variables. The following code reproduces *OI Biostat* Figure 1.33a, in which a bar plot was created using the `actn3.r577x` variable, with each group divided by the levels of race. The simplified version of the plot uses the default greyscale shading; it is also possible to specify a list of colors using `c()`.

The argument `legend` can be used to specify whether the row names or the column names are used for the legend.

```
## Figure 1.31a
counts.table = (table(famuss$race, famuss$actn3.r577x))
barplot(counts.table, legend = rownames(counts.table))
```

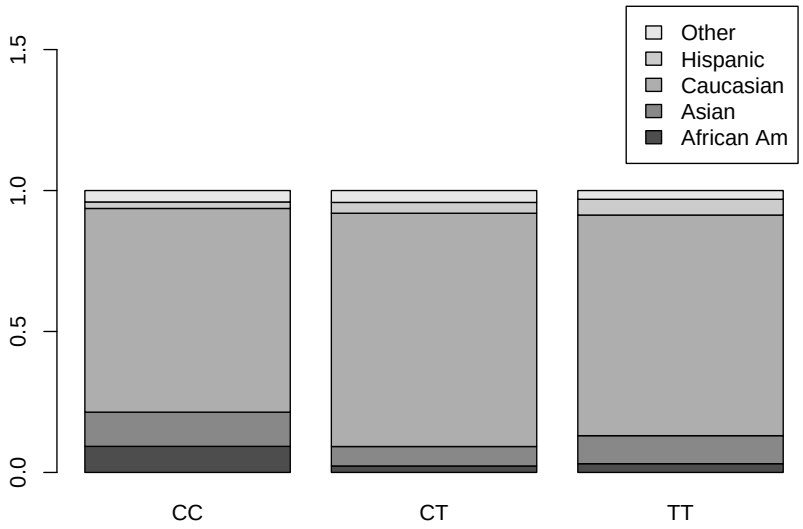



```
barplot(counts.table, col = c("gray", "firebrick1", "dodgerblue",
    "darkgreen", "gold"), ylim = c(0, 300), legend = rownames(counts.table))
```

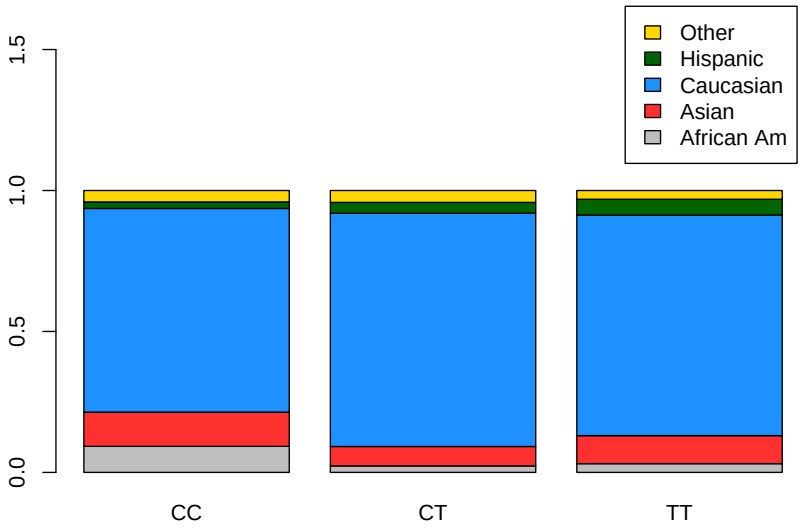

300
250
200
150
100
50
0
Other
Hispanic
Caucasian
Asian
African Am
CC
CT
TT

A **standardized segmented bar plot** uses proportions to scale the data, and draws from a contingency table with proportions. Setting `ylim` from 0 to 1.7 allows for enough empty space on the plot so that the legend does not overlap the bars.

```
## Figure 1.33b
counts.table = (table(famuss$race, famuss$actn3.r577x))
row.prop.table = prop.table(counts.table, 2)
barplot(row.prop.table, ylim = c(0, 1.7), legend = rownames(counts.table))
```

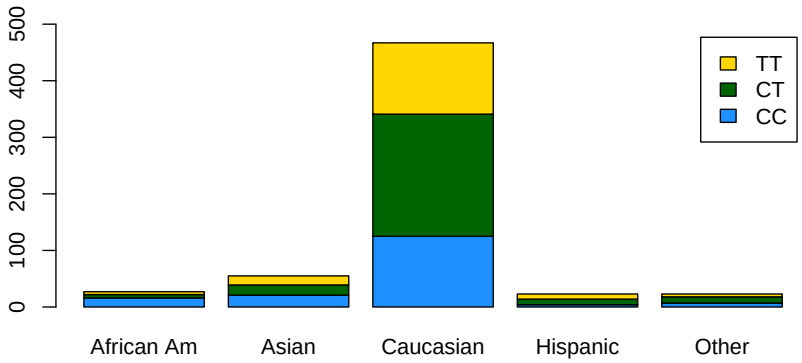



```
barplot(row.prop.table, col = c("gray", "firebrick1", "dodgerblue",
    "darkgreen", "gold"), ylim = c(0, 1.7), legend = rownames(counts.table))
```

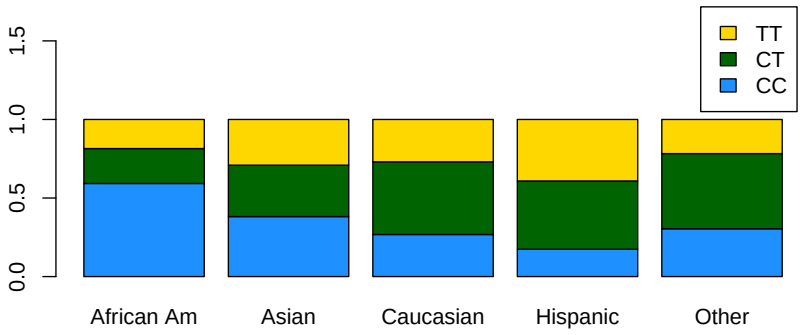


In *OI Biostat* Figure 1.34, the data from the contingency table are organized differently, with each bar representing a level of `race`. To make this change, reverse the order of the variables in `counts.table`

```
## Figure 1.34a
counts.table = (table(famuss$actn3.r577x, famuss$race))  ## change variable order

barplot(counts.table, col = c("dodgerblue", "darkgreen", "gold"),
    ylim = c(0, 500), legend = rownames(counts.table))
```




```
## Figure 1.34b
row.prop.table = prop.table(counts.table, 2)
barplot(row.prop.table, col = c("dodgerblue", "darkgreen", "gold"),
    ylim = c(0, 1.8), legend = rownames(counts.table))
```



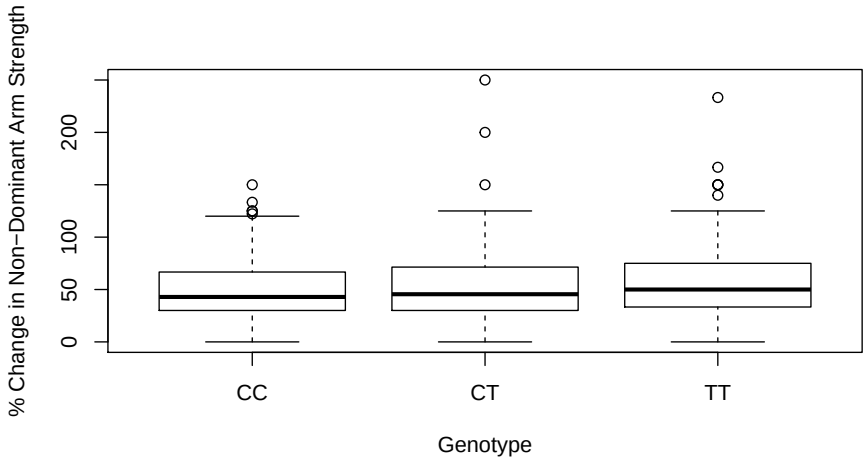
5.3 Comparing numerical data across groups

5.3.1 Side-by-side boxplots

The **side-by-side** boxplot is a useful tool for comparing the distribution of numerical data across categories. *OI Biostat* Figure 1.35a shows a side-by-side

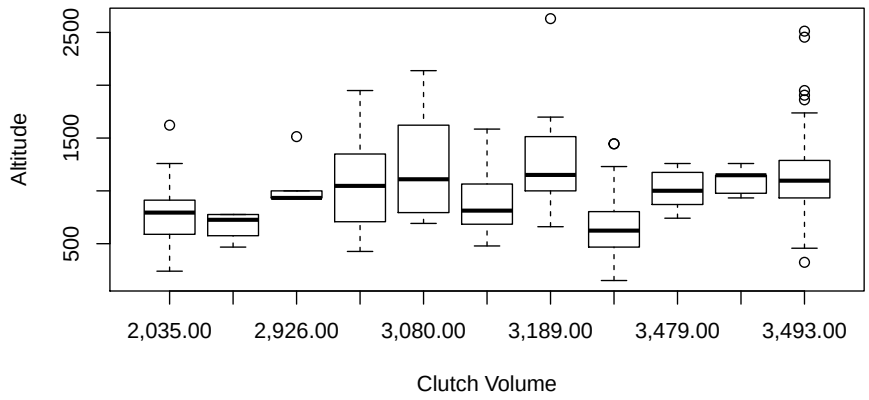
boxplot for percent change in non-dominant arm strength grouped by genotype. The response variable y comes before the tilde and the explanatory variable x; in this case, the response variable is percent change in strength.

```
## Figure 1.335(a)
boxplot(famuss$ndrm.ch ~ famuss$actn3.r577x,
    ylab = "% Change in Non-Dominant Arm Strength",
    xlab = "Genotype")
```



OI Biostat Figure 1.36 shows a larger side-by-side boxplot which compares the distribution of frog clutch sizes for different altitudes.

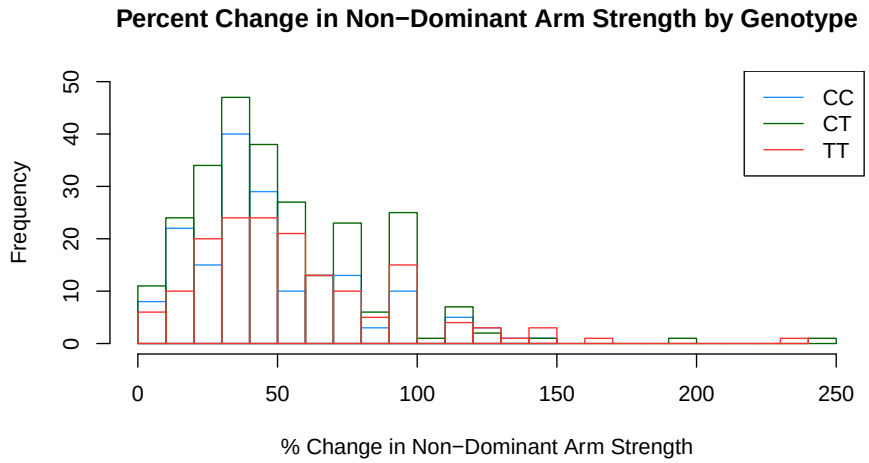
```
## Figure 1.36
boxplot(frog.altitude.data$clutch.volume ~ frog.altitude.data$altitude,
    xlab = "Clutch Volume", ylab = "Altitude")
```



5.3.2 Hollow histograms

A **hollow histogram** plots the outlines of histograms for each group onto the same axes. To plot a hollow histogram, specify any axis limits and labels in the command for the first histogram; the subsequent histograms require the argument `add = T` in order for them to be overlaid on top of the first plot.

```
## Figure 1.35(b)
hist(famuss$ndrm.ch[famuss$actn3.r577x == "CC"], breaks = 20, border = "dodgerblue",
     xlab = "% Change in Non-Dominant Arm Strength", ylim = c(0,50), xlim = c(0,250),
     main = "Percent Change in Non-Dominant Arm Strength by Genotype")
hist(famuss$ndrm.ch[famuss$actn3.r577x == "CT"], breaks = 20, border = "darkgreen",
     add = T)
hist(famuss$ndrm.ch[famuss$actn3.r577x == "TT"], breaks = 20, border = "firebrick1",
     add = T)

legend('topright',
       col = c("dodgerblue", "darkgreen", "firebrick1"),
       lwd = c(1, 1, 1),      ## line width
       legend = c('CC', 'CT', 'TT'))    ##legend labels
```



6 Genomic Data

```
boxplot(golub.exprs.pheno[, 7:9])

## Error in boxplot(golub.exprs.pheno[, 7:9]):  object 'golub.exprs.pheno'
not found
```