R Reference Guide
Suggested R Commands for Math 118

0. Getting Started

library(mosaic)

1. Inspecting a data frame

names()
head()
tail()
dim()
nrow()
ncol()

2. Numeric summaries for a single quantitative variable

sum()
mean()
median()
max()
min()
quantile()
IQR() or iqr
range()
sd()
var()
qdata()
favstats()

3. Data summaries for a single qualitative variable

tally()

4. Graphic summaries for a single quantitative variable

histogram()
densityplot()
freqpolygon()
dotplot()
bwplot()

5. Graphic summaries for a single qualitative variable

bargraph()
pie()

6. Numeric summaries for two quantitative variables

cor()
lm()

7. Graphic summary for two quantitative variables

xyplot()

8. Numeric summaries for a quantitative response and a qualitative explanatory variable.

sum()
mean()
median()  
max()  
min()  
quantile()  
IQR()  
range()  
sd()  
var()  
qdata()  
favstats()  

9. Graphical summary for a quantitative response and a qualitative explanatory variable  
bwplot()  

10. Numeric summary for two qualitative variables  
tally()  

11. Some useful functions  
log()  
exp()  

12. Probability Distributions  
sqrt()  
round()  
abs()  

13. Statistical Inference  
t.test() [one sample]  
t.test() [two independent samples]  

14. Data organization  
filter()  
arrange()  
sort()  

15. Other useful commands  
qqnorm() / qqline()  
plotDist()
0. Getting Started

library(mosaic)

Most all of the commands used in Math 118 come from an R package called mosaic. To use these commands you need to:

(a) Download and install the mosaic package. You need do this only once!

(b) On each occasion that you open R you need to load the mosaic package. You can do this by checking to the left of the name of the package in the lower-right page of RStudio.

Alternatively, in the Console (lower-left window), you can type:

library(mosaic)
1. Inspecting a data frame

names()

This function is used to obtain the names of the variables/columns in the data frame.

The format of this function is:

names(data frame name)

Example 1

When you have imported the data frame `exitpoll.csv`, use

```r
names(exitpoll)
```

```
[1] "Sex"       "Age"       "Candidate" "WaitTime" "Educ"
```

Example 2

When you have imported the data frame `hd.csv`, use

```r
names(hd)
```

```
[1] "SBP"    "Age"    "BMI"    "Height" "Smoke" "Race"
```
1. Inspecting a Data Frame

head()

By default, this function is used to obtain the contents of the first six rows of a designated data frame. For a data frame with a large number of rows, this gives you a sense of the data frame’s contents without having to examine every row.

The format of this function is:

head(data frame name)

The head function will also work with a variable that is not part of a data frame. In this case, the format is:

head(variable name)

(See Example 4 below)

Example 1

When you have imported the data frame exitpoll.csv, use

head(exitpoll)

<table>
<thead>
<tr>
<th>Sex</th>
<th>Age</th>
<th>Candidate</th>
<th>WaitTime</th>
<th>Educ</th>
</tr>
</thead>
<tbody>
<tr>
<td>Male</td>
<td>70</td>
<td>Romney</td>
<td>2</td>
<td>10</td>
</tr>
<tr>
<td>Female</td>
<td>75</td>
<td>Obama</td>
<td>1</td>
<td>13</td>
</tr>
<tr>
<td>Female</td>
<td>48</td>
<td>Obama</td>
<td>10</td>
<td>16</td>
</tr>
<tr>
<td>Female</td>
<td>27</td>
<td>Obama</td>
<td>1</td>
<td>16</td>
</tr>
<tr>
<td>Female</td>
<td>59</td>
<td>Obama</td>
<td>2</td>
<td>15</td>
</tr>
<tr>
<td>Female</td>
<td>60</td>
<td>Obama</td>
<td>3</td>
<td>12</td>
</tr>
</tbody>
</table>

Example 2

The default number of rows to print is six. But you can select a number other than six. For instance, if you want to see the first 10 rows of the data frame exitpoll, use:

head(exitpoll, 10)
<table>
<thead>
<tr>
<th>Sex</th>
<th>Age</th>
<th>Candidate</th>
<th>WaitTime</th>
<th>Educ</th>
</tr>
</thead>
<tbody>
<tr>
<td>Male</td>
<td>70</td>
<td>Romney</td>
<td>2</td>
<td>10</td>
</tr>
<tr>
<td>Female</td>
<td>75</td>
<td>Obama</td>
<td>1</td>
<td>13</td>
</tr>
<tr>
<td>Female</td>
<td>48</td>
<td>Obama</td>
<td>10</td>
<td>16</td>
</tr>
<tr>
<td>Female</td>
<td>27</td>
<td>Obama</td>
<td>1</td>
<td>16</td>
</tr>
<tr>
<td>Female</td>
<td>59</td>
<td>Obama</td>
<td>2</td>
<td>15</td>
</tr>
<tr>
<td>Female</td>
<td>60</td>
<td>Obama</td>
<td>3</td>
<td>12</td>
</tr>
<tr>
<td>Female</td>
<td>76</td>
<td>Obama</td>
<td>2</td>
<td>13</td>
</tr>
<tr>
<td>Female</td>
<td>62</td>
<td>Obama</td>
<td>5</td>
<td>13</td>
</tr>
<tr>
<td>Female</td>
<td>26</td>
<td>Obama</td>
<td>7</td>
<td>15</td>
</tr>
<tr>
<td>Male</td>
<td>45</td>
<td>Obama</td>
<td>26</td>
<td>16</td>
</tr>
</tbody>
</table>

**Example 3**

Suppose you want to not only see but also save the first 10 rows of exitpoll.csv. Use the command:

```
exit10 <- head(exitpoll, 10)
```

to save these rows as a new data frame called exit10. (the name is arbitrary). R won’t print the new data frame but will do so if you type its name.

**Example 4**

Here are final exam scores for the seven students in a seminar

```
scores <- c(86, 91, 81, 93, 88, 84, 95)
```

Here are the first four elements of the variable scores.

```
head(scores, 4)
```

```
[1] 86 91 81 93
```
1. Inspecting a Data Frame

**tail()**

By default, this function is used to obtain the contents of the last six rows of a designated data frame. For a data frame with a large number of rows, this gives you a sense of the data frame’s contents without having to examine every row.

The format of this function is:

`tail(data frame name)`

The tail function will also work with a variable that is not part of a data frame. In this case, the format is:

`tail(variable name)`

(See Example 4 below)

**Example 1**

When you have imported the data frame `exitpoll.csv`, use

```
tail(exitpoll)
```

<table>
<thead>
<tr>
<th>Sex</th>
<th>Age</th>
<th>Candidate</th>
<th>WaitTime</th>
<th>Educ</th>
</tr>
</thead>
<tbody>
<tr>
<td>Male</td>
<td>27</td>
<td>Obama</td>
<td>18</td>
<td>13</td>
</tr>
<tr>
<td>Male</td>
<td>86</td>
<td>Romney</td>
<td>28</td>
<td>6</td>
</tr>
<tr>
<td>Male</td>
<td>86</td>
<td>Obama</td>
<td>20</td>
<td>11</td>
</tr>
<tr>
<td>Female</td>
<td>53</td>
<td>Obama</td>
<td>33</td>
<td>12</td>
</tr>
<tr>
<td>Female</td>
<td>70</td>
<td>Obama</td>
<td>20</td>
<td>13</td>
</tr>
<tr>
<td>Female</td>
<td>84</td>
<td>Romney</td>
<td>36</td>
<td>12</td>
</tr>
</tbody>
</table>

**Example 2**

The default number of rows to print is six. But you can select a number other than six. For instance, if you want to see the last 10 rows of the data frame `exitpoll`, use:
tail(exitpoll, 10)

<table>
<thead>
<tr>
<th>Sex</th>
<th>Age</th>
<th>Candidate</th>
<th>WaitTime</th>
<th>Educ</th>
</tr>
</thead>
<tbody>
<tr>
<td>Male</td>
<td>80</td>
<td>Obama</td>
<td>0</td>
<td>11</td>
</tr>
<tr>
<td>Female</td>
<td>89</td>
<td>Obama</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>Male</td>
<td>63</td>
<td>Romney</td>
<td>0</td>
<td>11</td>
</tr>
<tr>
<td>Female</td>
<td>54</td>
<td>Obama</td>
<td>5</td>
<td>14</td>
</tr>
<tr>
<td>Male</td>
<td>27</td>
<td>Obama</td>
<td>18</td>
<td>13</td>
</tr>
<tr>
<td>Male</td>
<td>86</td>
<td>Romney</td>
<td>28</td>
<td>6</td>
</tr>
<tr>
<td>Male</td>
<td>86</td>
<td>Obama</td>
<td>20</td>
<td>11</td>
</tr>
<tr>
<td>Female</td>
<td>53</td>
<td>Obama</td>
<td>33</td>
<td>12</td>
</tr>
<tr>
<td>Female</td>
<td>70</td>
<td>Obama</td>
<td>20</td>
<td>13</td>
</tr>
<tr>
<td>Female</td>
<td>84</td>
<td>Romney</td>
<td>36</td>
<td>12</td>
</tr>
</tbody>
</table>

**Example 3**

Suppose you want to not only see but also save the last 10 rows of exitpoll.csv. Use the command:

```
ext10 <- tail(exitpoll, 10)
```

to save these rows as a new data frame called exit10. (the name is arbitrary). R won’t print the new data frame but will do so if you type its name.

**Example 4**

Here are final exam scores for the seven students in a seminar

```
scores <- c(86, 91, 81, 93, 88, 84, 95)
```

Here are the last three elements of the variable scores.

```
tail(scores, 3)
```

```
[1] 88 84 95
```
1. Inspecting a data frame

dim()

This function is used to obtain the dimensions (the number of rows and columns) for a data frame.

The format of this function is:

dim(data frame name)

Example 1

What are the dimensions of the data set exitpoll.csv.

dim(exitpoll)
[1] 208 5

The exitpoll data has 208 rows (voters) and five columns (variables).

Example 2

What are the dimensions of the data set hd.csv.

dim(hd)
[1] 32 6

The hd data has 32 rows (patients) and six columns (variables).
1. Inspecting a data frame

nrow()

As the name suggests, this function is used to obtain the number of rows (individuals) in a data frame.

The format of this function is:

\[
\text{nrow(data frame name)}
\]

Example 1

The data set \textit{exitpoll.csv} contains how many individuals (voters)?

\[
\text{nrow(exitpoll)}
\]

[1] 208

The exitpoll data has 208 rows (voters).

Example 2

The data set \textit{hd.csv} contains how many individuals (patients)?

\[
\text{nrow(hd)}
\]

[1] 32

The hd data has 32 rows (patients).
1. Inspecting a data frame

ncol()

As the name suggests, this function is used to obtain the number of columns (variables) in a data frame.

The format of this function is:

\texttt{ncol(data frame name)}

**Example 1**

The data set \texttt{exitpoll.csv} contains how many variables?

\texttt{ncol(exitpoll)}

\begin{verbatim}
[1] 5
\end{verbatim}

The exitpoll data has five columns (variables).

**Example 2**

The data set \texttt{hd.csv} contains how many variables (variables)?

\texttt{dim(hd)}

\begin{verbatim}
[1] 6
\end{verbatim}

The hd data has six columns (variables).
2. Numeric summaries for a single quantitative variable

sum()

This function is used to find the sum of all the values for a quantitative/numeric variable.

If the variable is a column in a data frame, the format for the function is:

\[ \text{sum} \left( \sim \text{variable name}, \text{data} = \text{data frame name} \right) \]

If the variable is not part of a data frame, the format is simpler

\[ \text{sum} \left( \sim \text{variable name} \right) \]

(See Example 4 below)

Example 1

What is the total wait time for the 208 voters in the exitpoll data?

\[ \text{sum} \left( \sim \text{WaitTime}, \text{data} = \text{exitpoll} \right) \]

\[ [1] \ 2154 \]

The total wait time is 2154 minutes.

Example 2

Express the total wait time for the 208 voters in the exitpoll data in hours and save your answer as an object called tot_wait.

\[ \text{tot\_wait} \leftarrow \text{sum} \left( \sim \text{WaitTime}, \text{data} = \text{exitpoll} \right) / 60 \]

\[ \text{tot\_wait} \]

\[ [1] \ 35.9 \]

The total wait time is 35.9 hours.
**Example 3**

The variable Smoke in the hd data frame takes the value 1 if the patient is a smoker and 0 otherwise. How many of the 32 patients are smokers?

If we sum the values in Smoke, we will have the number of 1’s and the number of smokers.

```r
sum(~Smoke, data = hd)
```

```
[1] 17
```

Seventeen (17) of the patients report smoking.

**Example 4**

Here are final exam scores for the seven students in a seminar.

```r
scores <- c(86, 91, 81, 93, 88, 84, 95)
sum(~scores)
```

```
[1] 618
```

The sum of all the 7 scores is 618.
2. Numeric summaries for a single quantitative variable

mean()

This function is used to obtain the mean/average value for a quantitative/numeric variable.

If the variable is a column in a data frame, the format for the function is:

\[ \text{mean(\sim variable name, data = data frame name)} \]

If the variable is not part of a data frame, the format is simpler:

\[ \text{mean(\sim variable name)} \]

(See Example 3 below)

**Example 1**

What is the mean systolic blood pressure of the 32 patients in the hd data?

\[ \text{mean(\sim variable name, data = data frame name)} \]

\[ \text{mean(\sim SBP, data = hd)} \]

\[ [1] \quad 144.5312 \]

The mean SBP is 144.5 mm.

**Example 2**

What is the mean wait time to vote for the 208 voters in the exitpoll data?

\[ \text{mean(\sim WaitTime, data = exitpoll)} \]

\[ [1] \quad 10.35577 \]

The mean wait time to vote is 10.4 minutes.
**Example 3**

Here are final exam scores for the seven students in a seminar

\[
\text{scores} \leftarrow c(86, 91, 81, 93, 88, 84, 95)
\]

\[
\text{mean}(~\text{scores})
\]

[1] 88.28571

The mean score in the seminar is 88.3.

The mean is sensitive to extreme/outlying values. In the presence of such values we may choose to compute a *trimmed mean*. An \( \alpha \)-trimmed mean is obtained by eliminating the smallest 100\( \alpha \)% of values and the largest 100\( \alpha \)% of values and computing the mean of the middle 100(1-2\( \alpha \))% of values. For example, a 0.1-trimmed mean is obtained by eliminating the smallest 10% and the largest 10% of values and computing the mean of the middle 80% of values.

To get R to compute an \( \alpha \)-trimmed mean, the function must have the following form:

\[
\text{mean}(~ \text{variable name}, \text{data} = \text{data frame name}, \text{trim} = \alpha)
\]

**Example 4**

What is the 0.1-trimmed mean wait time to vote for the 208 voters in the exitpoll data?

\[
\text{mean}(~\text{WaitTime}, \text{data} = \text{exitpoll}, \text{trim} = 0.1)
\]

[1] 8.494048

The 0.1-trimmed mean wait time to vote is 8.5 minutes.
2. Numeric summaries for a single quantitative variable

**median()**

This function is used to obtain the median value for a quantitative/numeric variable.

If the variable is a column in a data frame, the format for the function is:

```r
median(~ variable name, data = data frame name)
```

If the variable is not part of a data frame, the format is simpler:

```r
median(~ variable name)
```

(See Example 3 below)

**Example 1**

What is the median wait time to vote for the 208 voters in the exitpoll data?

```r
median(~WaitTime, data = exitpoll)
```

```
[1] 7
```

The median wait time is 7 minutes.

**Example 2**

What is the median systolic blood pressure for the 32 patients in the hd data?

```r
median(~ SBP, data = hd)
```

```
[1] 143
```

**Example 3**

Here are final exam scores for the seven students in a seminar

```r
scores <- c(86, 91, 81, 93, 88, 84, 95)
```
The median score is 88.

2. Numeric summaries for a single *quantitative* variable
max()

You won’t be surprised to learn that this function is used to obtain the largest value for a quantitative/numeric variable.

If the variable is a column in a data frame, the format for the function is:

```
max(~ variable name, data = data frame name)
```

If the variable is not part of a data frame, the format is simpler

```
max(~ variable name)
```

(See Example 2 below)

**Example 1**

What is the largest number of years of formal education for the voters in the exitpoll data?

```
max(~Educ, data = exitpoll)
```

```
[1] 19
```

The greatest number of years of formal education is 19.

**Example 2**

Here are final exam scores for the seven students in a seminar

```
scores <- c(86, 91, 81, 93, 88, 84, 95)
max(~scores)
```

```
[1] 95
```

The highest score on the seminar is 95.

2. Numeric summaries for a single quantitative variable
min()

You won’t be surprised to learn that this command is used to obtain the smallest value for a quantitative/numeric variable.

If the variable is a column in a data frame, the format for the function is:

```
min(~ variable name, data = data frame name)
```

If the variable is not part of a data frame, the format is simpler

```
min(~ variable name)
```

(See Example 2 below)

Example 1

What is the least number of years of formal education for the voters in the exitpoll data?

```
min(~Educ, data = exitpoll)
```

```
[1] 6
```

The smallest number of years of formal education is 6.

Example 2

Here are final exam scores for the seven students in a seminar

```
scores <- c(86, 91, 81, 93, 88, 84, 95)
min(~scores)
```

```
[1] 81
```

The lowest score on the seminar is 81.

2. Numeric summaries for a single quantitative variable
quantile()

This command is used to obtain the five number summary (minimum, $Q_1 = X_{25}$, $Q_2 = \text{median}$, $Q_3 = X_{75}$, and maximum) for a quantitative/numeric variable.

Note: R uses the term *quantile* rather than the more familiar and common term *percentile*.

If the variable is a column in a data frame, the format for the function is:

`quantile(~ variable name, data = data frame name)`

If the variable is not part of a data frame, the format is simpler

`quantile(~ variable name)`

(See Example 2 below)

**Example 1**

Obtain the five-number summary of SBP for the 32 patients in the `hd` data?

`quantile(~SBP, data = hd)`

```
0%   25%   50%   75%   100%
120.00 134.75 143.00 152.00 180.00
```

The smallest SBP is 120 mm, $Q_1 = X_{25} = 134.75$ mm, $Q_2 = \text{median} = 143$ mm, $Q_3 = X_{75} = 152$ mm, and the largest SBP is 180 mm.

**Example 2**

Here are final exam scores for the seven students in a seminar.

`scores <- c(86, 91, 81, 93, 88, 84, 95)`

Here is the five number summary:
2. Numeric summaries for a single *quantitative* variable
IQR() or iqr()

This function is used to find the inter-quartile range \((Q_3 - Q_1)\) for a quantitative/numeric variable. This is a rare R command where you may use either upper-case or lower-case letters.

If the variable is a column in a data frame, the format for the function is:

\[
\text{IQR}(\sim \text{variable name}, \text{data} = \text{data frame name})
\]

If the variable is not part of a data frame, the format is simpler

\[
\text{IQR}(\sim \text{variable name})
\]

(See Example 3 below)

**Example 1**

Obtain the IQR of SBP for the 32 patients in the hd data?

\[
\text{IQR}(\sim \text{SBP}, \text{data} = \text{hd})
\]

\[
[1] 17.25
\]

The IQR is 17.25 mm.

**Example 2**

Obtain the iqr of wait-time for the 208 voters in the exitpoll data.

\[
\text{iqr}(\sim \text{WaitTime}, \text{data} = \text{exitpoll})
\]

\[
[1] 11.25
\]

The iqr of wait-time is 11.25 minutes.

**Example 3**
Here are final exam scores for the seven students in a seminar.

\[
\text{scores} \leftarrow c(86, 91, 81, 93, 88, 84, 95)
\]

\[
\text{IQR(scores)}
\]

\[
\left[ 1 \right] 7
\]

The IQR of the scores is 7.

---

2. Numeric summaries for a single \textit{quantitative} variable
range()

This function is used to obtain the range of values for a quantitative/numeric variable.

If the variable is a column in a data frame, the format for the function is:
range(~ variable name, data = data frame name)

If the variable is not part of a data frame, the format is simpler
range(~ variable name)

(See Example 2 below)

Actually, the range function does not produce the range; it produces the minimum value and the maximum value, as you will see.

Example 1
What is the range of years of formal education for the voters in the exitpoll data?
range(~ Educ, data = exitpoll)
[1]  6 19

The smallest number of years is 6; the largest is 19. So the range is 19 - 6 = 13 years.

It is simple to get R to compute the actual range by asking for the difference between the largest and the smallest values.

Example 1 again
range_educ <- max(~Educ, data = exitpoll) - min(~Educ, data = exitpoll)
range_educ
[1] 13

I saved the difference in the new object, range_educ. To see the result you simply have to type the name of the object. If you had omitted the name of the difference, R would have printed out (but not saved) the result.

max(~Educ, data = exitpoll) - min(~Educ, data = exitpoll)
[1] 13

Example 2
Here are final exam scores for the seven students in a seminar

scores <- c(86, 91, 81, 93, 88, 84, 95)
range(~scores)
[1] 81 95

The range of scores is from a minimum of 81 to a maximum of 95 (a range of 14 points.)

2. Numeric summaries for a single quantitative variable
sd()

This function is used to find the standard deviation for a quantitative/numeric variable.

If the variable is a column in a data frame, the format for the function is:

\[ sd(~ \text{variable name, data = data frame name}) \]

If the variable is not part of a data frame, the format is simpler

\[ sd(~ \text{variable name}) \]

(See Example 3 below)

**Example 1**

What is the standard deviation of wait time for the 208 voters in the exitpoll data?

\[
\text{sd(~ WaitTime, data = exitpoll)}
\]

\[
[1] \quad 10.67338
\]

The standard deviation of wait time is \( S = 10.7 \) minutes. Recall that the standard deviation is in the same units as the data—in this case, minutes.

**Example 2**

What is the standard deviation of age for the 32 patients in the hd data?

\[
\text{sd(~ Age, data = hd)}
\]

\[
[1] \quad 6.956083
\]

The standard deviation of wait time is 7.0 years.
Example 3

Here are final exam scores for the seven students in a seminar.

```r
scores <- c(86, 91, 81, 93, 88, 84, 95)
sd(~scores)
[1] 5.023753
```

The standard deviation of the 7 scores is 5.0.

Recall that the variance of a variable is simply the square of the standard deviation. Though it is not necessary (because the there is a var command) it is easy to find the variance of the 7 scores from the standard deviation.

```r
sd(~scores)^2
[1] 25.2381
```

The variance of the 7 scores is 5.0 squared points.

2. Numeric summaries for a single *quantitative* variable
var()

This function is used to find the variance for a quantitative/numeric variable.

If the variable is a column in a data frame, the format for the function is:

```
sd(~ variable name, data = data frame name)
```

If the variable is not part of a data frame, the format is simpler

```
sd(~ variable name)
```

(See Example 3 below)

**Example 1**

What is the variance of wait time for the 208 voters in the exitpoll data?

```
var(~WaitTime, data = exitpoll)
```

```
[1] 113.9211
```

The variance of wait time is $S^2 = 113.9$ minutes$^2$. Recall that variance is in squared units—in this case squared minutes.

**Example 2**

What is the variance of age for the 32 patients in the hd data?

```
var(~ Age, data = hd)
```

```
[1] 48.3871
```

The variance of age is 48.4 years$^2$.

**Example 3**
Here are final exam scores for the seven students in a seminar.

\[ \text{scores} <- \text{c(86, 91, 81, 93, 88, 84, 95)} \]

\[ \text{var(~scores)} \]

\[ [1] \ 25.2381 \]

The variance of the 7 scores is 25.2 squared points.

Recall that the standard deviation is simply the square root of the variance. Though it is not necessary (because there is a sd command) here are a couple of ways to find the standard deviation of the 7 scores from the variance.

\[ \text{sqrt(var(~scores))} \]

\[ [1] \ 5.023753 \]

\[ \text{var(~scores)}^{0.5} \]

\[ [1] \ 5.023754 \]

The standard deviation of the 7 scores is 5.0.

2. Numeric summaries for a single quantitative variable
The purpose of this function is not obvious from the name. In fact, it is used to obtain $X_{100q}$, the 100$q$th percentile value for a quantitative/numeric variable. For instance, with $q = 0.6$, $X_{60}$ is the 60th percentile—i.e. 60% of the scores will be less than $X_{60}$.

Note: R uses the term *quantile* rather than the more familiar and common term *percentile*.

Note: The format for the qdata function is a tad more tricky than for other quantities.

If the variable is a column in a data frame, the format for the function is:

```
qdata(~ variable name, p = q, data = data frame name)
```

If the variable is not part of a data frame, the format is simpler

```
qdata(~ variable name, p = q)
```

(See Example 3 below)

Note: The value for $q$ must be a number between 0 and 1.

**Example 1**

What is the 60th percentile of the ages of the 208 voters in the exitpoll data?

```
qdata(~Age, p = 0.6, data = exitpoll)
```

```
p quantile
0.6  63.0
```

The 60th percentile of the ages is $X_{60} = 63$ years; that is 60% of the 208 voters are younger than 63.
Example 2
What is the 25th percentile of the SBP for the 32 patients in the hd data?

```r
qdata(~SBP, p = 0.25, data = hd)
  p quantile
0.25 134.75
```

The 25th percentile (also known as the first quartile, Q₁) for SBP is $X_{25} = Q₁ = 48$ mm.

Example 3
Here are final exam scores for the seven students in a seminar

```r
scores <- c(86, 91, 81, 93, 88, 84, 95)
```

Example 4
You can use the qdata (or the median) command to obtain the median score:

```r
qdata(~scores, p = 0.5)
  p quantile
0.5  88.0
median(~scores)
[1] 88
```

2. Numeric summaries for a single quantitative variable
As the name suggests, this command provides the most widely used (favorite) statistics for a quantitative/numeric variable.

If the variable is a column in a data frame, the format for the function is:

```
favstats(~ variable name, data = data frame name)
```

If the variable is not part of a data frame, the format is simpler

```
favstats(~ variable name)
```

(See Example 2 below)

**Example 1**
Here are the favstats for the 208 wait times for the 208 voters in the exitpoll data.

```
favstats(~ WaitTime, data = exitpoll)
```

```
min Q1 median Q3 max  mean  sd  n missing
 0  3   7 14.25  67 10.35577 10.67338 208   0
```

The first five numbers are the values associated with the five-number summary, the minimum, the first, second, third, quartiles, and the maximum values. The last four values are the mean, the standard deviation, the number of observations (n), and the number of missing values. In this example, there are 208 values and none are missing.

**Example 2**
Here are final exam scores for the seven students in a seminar.

```
scores <- c(86, 91, 81, 93, 88, 84, 95)
```

Here are the favstats statistics for these data.

```
favstats(~scores)
```

```
min Q1 median Q3 max  mean  sd  n missing
 0  3  14.5  17 10.0 10.55 208   0
```
Example 3

Referring again to the scores in Example 2, suppose that the scores for students two and six were lost or could not be computed (perhaps they had failed to submit an exam). In this case we represent the missing values by NA (Not Available) in R. Thus, we would enter these data as follows

```r
sc <- c(86, NA, 81, 93, 88, NA, 95)
```

2. Numeric summaries for a single quantitative variable
length()

The length function can be used with both quantitative and qualitative variables. It returns the number of elements in a vector/variable/column of data. In most all cases it will give you the same output as the nrow() function.

The length command is a base R command rather than a mosaic command and so the syntax is base R. There is no ~ involved.

If the variable is a column in a data frame, the format for the function is:

```
length(data frame name$variable name)
```

If the variable is not part of a data frame, the format is simpler

```
length(variable name)
```

(See Example 3 below)

Example 1
How many ages are listed in the variable Age for the exitpoll data?

```
length(exitpoll$Age)
[1] 208
```

In a data frame, every column has the same length and so you can obtain the same answer with

```
nrow(exitpoll)
[1] 208
```

Example 2
How many elements are there in the variable Race in the hd data set?
length(hd$Race)
[1] 32

There are 32 entries for the variable race.

**Example 3**

Here are final exam scores for the seven students in a seminar.

scores <- c(86, 91, 81, 93, 88, 84, 95)

You can find the number of scores with:

length(scores)
[1] 7

**Example 4**

Referring again to the scores in Example 3, suppose that the scores for students two and six were lost or could not be computed (perhaps they had failed to submit an exam). In this case we represent the missing values by NA (not available) in R. Thus, we would enter these data as follows

sc <- c(86, NA, 81, 93, 88, NA, 95)

length(sc)
[1] 7

Notice that the length command counts the number of elements in the variable whether the data is missing or not.

3. Data summaries for a single qualitative variable
tally()

The tally command is one of the most useful R commands. It is used primarily with qualitative data and, at its most basic, provides the number of times that each category occurs.

If the variable is a column in a data frame, the format for the function is:
\[ \text{tally(} \sim \text{ variable name, data = data frame name)} \]
If the variable is not part of a data frame, the format is simpler
\[ \text{tally(} \sim \text{ variable name)} \]
(See Example 4 below)

**Example 1**
Obtain a tally of the 32 ‘Race’s in the hd data set.
\[ \text{tally(} \sim \text{Race, data = hd)} \]
Race
\[
\begin{array}{ccc}
\text{Black} & \text{Hispanic} & \text{White} \\
7 & 5 & 20
\end{array}
\]
Of the 32 patients, seven are Black, five are Hispanic, and 20 are White.

**Example 2**
Referring to the exitpoll data, how many votes did each of the candidates, Obama and Romney receive?
\[ \text{tally(} \sim \text{ Candidate, data = exitpoll)} \]
Candidate
\[
\begin{array}{ccc}
\text{Obama} & \text{Romney} \\
160 & 48
\end{array}
\]
If, instead of looking for the number of votes for each candidate, you wanted the proportion of votes received by each candidate or, more commonly, the percentage of votes received by each candidate. To obtain either, you need only add an extra component/argument to the tally command.

**Example 3**

What percentage of the 208 voters voted for Obama?

```r
tally(~ Candidate, data = exitpoll, format = "percent")
```

| Candidate | Obama   | Romney | 76.92308 | 23.07692 |

In this poll, Obama received 76.9% of the votes (and Romney, 23.1%).

What proportion of the 208 voters voted for Obama?

```r
tally(~ Candidate, data = exitpoll, format = "proportion")
```

| Candidate | Obama   | Romney | 0.7692308 | 0.2307692 |

The proportion voting for Obama is 0.7692.

Note: it is rarely necessary for percentages to contain more than one decimal place. You can ask R to round percentages to one decimal place.

```r
t <- tally(~ Candidate, data = exitpoll, format = "percent")
round(t, 1)
```

| Candidate | Obama   | Romney | 76.9   | 23.1   |

It is more cumbersome but you can perform this same rounding operation in one step:
round(tally(~ Candidate, data = exitpoll, format = "percent"), 1)
Candidate
  Obama  Romney
    76.9  23.1

Example 4
Here are the final grades for the 18 students in my Math 118 section several years ago.


tally(~ grades)
grades
  A  B  C
  5 10  3

Of the 18 students, five received A’s, 10 received B’s, and 3 received C’s.

The tally command can also be useful in summarizing a quantitative variable where just a few different values are repeated many times. Here is an example:

Example 5
The variable typos contains the number of typing errors on each page of a 70-page document. What number and percent of pages were free of errors?

typos
  [1]  2  2  2  0  1  1  1  1  2  1  1  0  2  0  0  0  0  0  0  1  0  1  2  2  0  1  2  0  2  
  2  1  1  1  1  0  2  0  3  0  0  2  3  1  1  0  0  1  1  2  1  2  3  2  2  1  1  3  2  2  
  2  0  3  2
[63]  2  4  2  3  0  0  2  0

tally(~typos)
typos
  0  1  2  3  4
  20 20 23  6  1

38
tally(~typos, format = "percent")
typos

0 1 2 3 4
28.571429 28.571429 32.857143 8.571429 1.428571

A total of 20 or 28.6% of the pages were error free.

4. Graphic summaries for a single quantitative variable

histogram()
If the variable is a column in a data frame, the format for the function is:

\[
\text{histogram}(\sim \text{variable name}, \text{data} = \text{data frame name})
\]

If the variable is not part of a data frame, the format is simpler

\[
\text{histogram}(\sim \text{variable name})
\]

(See Example 8 below)

**Example 1**

Obtain a histogram for the 32 SPB’s in the hd data set.

\[
\text{histogram}(\sim \text{SBP, data} = \text{hd})
\]

The default color is light green. The scale on the horizontal axis is the scale of SBP’s. The default scale on the vertical axis is ‘Density’. Without going into more detail, using the density scale enables us to superimpose a normal curve on the histogram, (as you will see in a bit). More typically, the vertical axis would be either the number of observations in each interval or the percentage of the observations in each interval.

**Example 2**

Repeat Example 1 but have the vertical axis be the number of observations in each interval.
histogram(~ SBP, data = hd, type = "count")

You can obtain percentages on the vertical axis by replacing "count" with "percent" in the command.

You can easily modify the color of your histogram:

**Example 3**

Make the histogram in Example 1 red.

```r
histogram(~ SBP, data = hd, col = "red")
```

Note: Go to [http://www.stat.columbia.edu/~tzheng/files/Rcolor.pdf](http://www.stat.columbia.edu/~tzheng/files/Rcolor.pdf) for a look at the colors available in R

**Example 4**

Obtain a lavender colored histogram of the 208 wait times for the voters in the exitpoll data.

```r
histogram(~WaitTime, data = exitpoll, col = "lavender")
```
R chooses a ‘convenient’ number of intervals for a histogram (in the above case, 9). However, it is simple to specify the number of intervals you want using the \textit{nint} argument.

\textbf{Example 5}

An often used guide to determining a number of intervals is to take the (rounded) value for the square-root of the number of observations. In the exitpoll data there are $n = 208$ wait times. So the number of intervals is 14.

\begin{verbatim}
round(sqrt(208))
\end{verbatim}

\begin{verbatim}
[1] 14
\end{verbatim}

Require a default histogram of the wait times with 14 intervals using the \textit{nint} argument:

\begin{verbatim}
histogram(~ WaitTime, data = exitpoll, nint = 14)
\end{verbatim}

Frequently in statistics we need to consider whether a quantitative variable could plausibly be considered a sample from a normal distribution. One simple way to
check this is to superimpose a normal distribution (with the same mean and standard deviation as the data) on top of a histogram.

**Example 6**

Obtain a default histogram of the 32 patient SBP’s in the hd data set. Superimpose a normal distribution on the histogram.

```r
histogram(~ SBP, data = hd, fit = "normal")
```

Note: The `fit` command can be used only when the default option “Density” is used.

If you are obtaining a histogram simply to get a sense of the shape of the distribution, there is little need to provide helpful annotation (like a title). However, if you intend to include your graph in an article, you will want to include such annotation. In the next example you will provide a title and a new label for the horizontal (x) axis.

**Example 7**
Obtain a default histogram of the 208 wait times for the voters in the exitpoll data. Add the title “Histogram of the 208 Wait Times” and change the x-label from the default “WaitTime” to “Wait Times”.

```r
histogram(~ WaitTime, data = exitpoll,
  main = "Histogram of the 208 Wait Times",
  xlab = "Wait Times")
```

Always use `main =` to create a title for a plot; always use `xlab =` to create a label for the X (horizontal) axis. Can you guess what you would use to create a label for the Y (vertical) axis?

**Example 8**

Here are final exam scores for the seven students in a seminar.

```r
scores <- c(86, 91, 81, 93, 88, 84, 95)
histogram(~scores)
```
4. Graphic summaries for a single *quantitative* variable

`densityplot()`

This function provides a density plot (something like a continuous histogram) for the data in a quantitative/numeric variable.

If the variable is a column in a data frame, the format for the function is:
densityplot(~ variable name, data = data frame name)

If the variable is not part of a data frame, the format is simpler

densityplot(~ variable name)

(See Example 5 below)

**Example 1**

Obtain a density plot for the 32 SPB’s in the hd data set.

densityplot(~SBP, data = hd)

![Density plot example](image)

You can adjust the smoothness of the plot with the *adjust* argument. The default value for *adjust* is 1; higher values smooth more heavily; lower values, less so.

**Example 2**

Obtain density plots for the 32 SPB’s in the hd data set using (a) adjust = 0.5, and (b) adjust = 1.5.

```
densityplot(~SBP, data = hd, adjust = 0.5)
```
You can easily modify the color of your densityplot:

**Example 3**

Make the default densityplot in Example 1 red.

```r
densityplot(~SBP, data = hd, col = "red")
```
If you are obtaining a density plot simply to get a sense of the shape of the distribution, there is little need to provide helpful annotation (like a title). However, if you intend to include your graph in an article, you will want to include such annotation. In the next example you will provide a title and a label for the horizontal (X) axis.

**Example 4**

Obtain a default density plot of the 208 wait times for the voters in the exitpoll data. Add the title “Density Plot of the 208 Wait Times” and create the x-label “Wait Times”.

```r
densityplot(~WaitTime, data = exitpoll, 
    main = "Density Plot of the 208 Wait Times", 
    xlab = "Wait Times")
```
Always use `main = ` to create a title for a plot; always use `xlab = ` to create a label for the X (horizontal) axis. Can you guess what you would use to create a label for the Y (vertical) axis?

**Example 5**

Here are final exam scores for the seven students in a seminar.

```r
scores <- c(86, 91, 81, 93, 88, 84, 95)
densityplot(~scores)
```
4. Graphic summaries for a single *quantitative* variable

**freqpolygon()**

This function provides a frequency polygon for the data in a quantitative/numeric variable.

If the variable is a column in a data frame, the format for the function is:

```
freqpolygon(~ variable name, data = data frame name)
```

If the variable is not part of a data frame, the format is simpler

```
freqpolygon(~ variable name)
```

(See Example 5 below)

**Example 1**

Obtain a frequency polygon for the 32 SPB’s in the hd data set.

```
freqpolygon(~SBP, data = hd)
```

Frequency polygons use the same data as histograms but use lines that connect the points that would have been the center of the upper bars of the blocks of the histogram.
You can easily modify the color of your frequency polygon:

**Example 2**

Obtain a red frequency polygon of the 208 wait times for the voters in the exitpoll data.

```
freqpolygon(~waitTime, data = exitpoll, col = "red")
```

Note: Go to [http://www.stat.columbia.edu/~tzheng/files/Rcolor.pdf](http://www.stat.columbia.edu/~tzheng/files/Rcolor.pdf) for a look at the colors available in R

To some extent you can control the smoothness of the frequency polygon by specifying the number of intervals (nint) (as if you were drawing a histogram).

**Example 3**

Obtain a the frequency polygon for the waiting times with just six intervals.

```
freqpolygon(~waitTime, data = exitpoll, 
             col = "red",
             nint = 6)
```
If you are obtaining a frequency polygon simply to get a sense of the shape of the distribution, there is little need to provide helpful annotation (like a title). However, if you intend to include your graph in an article, you will want to include such annotation. In the next example you will provide a title and a label for the horizontal (X) axis.

**Example 4**

Obtain a default red frequency polygon of the 208 wait times for the voters in the exitpoll data. Add the title “Frequency Polygon of the 208 Wait Times” and create the x-label “Wait Times”.

```r
freqpolygon(~WaitTime, data = exitpoll, 
  col = "red", 
  main = "Frequency Polygon of the 208 Wait Times", 
  xlab = "Wait Times")
```
Always use `main =` to create a title for a plot; always use `xlab =` to create a label for the X (horizontal) axis. Can you guess what you would use to create a label for the Y (vertical) axis?

**Example 5**

Here are final exam scores for the seven students in a seminar.

```r
scores <- c(86, 91, 81, 93, 88, 84, 95)
freqpolygon(~scores)
```

With so few observations the frequency polygon is not terribly insightful.
4. Graphic summaries for a single *quantitative* variable

**dotplot()**

The `dotplot()` command will produce, for a quantitative variable, a simple dot plot in which each observation is represented by a dot. This plot is most useful for small data sets.

If the variable is a column in a data frame, the format for the function is:

```
dotplot(~ variable name, data = data frame name)
```

If the variable is not part of a data frame, the format is simpler

```
dotplot(~ variable name)
```

(See Example 5 below)

**Example 1**

Obtain a dot plot for the 32 SPB’s in the hd data set.

```
dotplot(~SBP, data = hd)
```

You can easily modify the color of your frequency polygon:
Example 2

Repeat Example 2 with red dots.

dotplot(~SBP, data = hd, col = "red")

Note: Go to http://www.stat.columbia.edu/~tzheng/files/Rcolor.pdf for a look at the colors available in R

You can change the symbols used in a dotplot (using the pch argument) and the size of the symbols (using the cex argument).

(pch stands for printing character and, I think, cex stands for character expansion.)

Example 3

Rework Example 1 with red dots, pch = 19 and cex = 1.5.

dotplot(~SBP, data = hd,
        col = "red",
        pch = 19,
        cex = 1.5)
For lists of printing characters in R, Google R pch

The default value for cex is 1. The larger the value, the larger the symbol size. A value of 1.5 produces a symbol 50% bigger than the default; a value of 0.5 produces a symbol 50% smaller than the default.

**Example 5**

Here are final exam scores for the seven students in a seminar.

```r
scores <- c(86, 91, 81, 93, 88, 84, 95)
dotplot(~scores)
```
4. Graphic summaries for a single quantitative variable

bwplot()

This function will produce a simple box-and-whisker display (often referred to more simply as a boxplot).

If the variable is a column in a data frame, the format for the function is:

```
bwplot(~variable name, data = data frame name)
```

If the variable is not part of a data frame, the format is simpler

```
bwplot(~variable name)
```

(See Example 6 below)

**Example 1**

Obtain a boxplot for the 32 SPB’s in the hd data set.

```
bwplot(~SBP, data = hd)
```
Example 2

Obtain a boxplot for the 208 wait times for the voters in the exitpoll data set.

\texttt{bwplot(\sim \text{WaitTime}, \text{data} = \text{exitpoll})}

By default, the ‘box’ in the box-and-whisker display is white. You can color the box with the \textit{fill} argument.

Example 3

Make the ‘box’ in Example 2 lavender.

\texttt{bwplot(\sim \text{WaitTime}, \text{data} = \text{exitpoll}, \text{fill} = "lavender")}
If you wish, you can change the color of the dot representing the median value in the box-and-whisker display with the `col` argument.

**Example 4**

Change the color of the dot in Example 3 to red.

```r
bwplot(~ WaitTime, data = exitpoll, fill = "lavender", col = "red")
```
If you are obtaining a box-and-whisker display simply to get a sense of the shape of the distribution, there is little need to provide helpful annotation (like a title). However, if you intend to include your graph in an article, you will want to include such annotation. In the next example you will provide a title and a label for the horizontal (X) axis.

**Example 5**

Obtain a default box-and-whisker display of the 208 wait times for the voters in the exitpoll data. Add the title “Box-and-Whisker Display of the 208 Wait Times” and create the x-label “Wait Times”.

```r
bwplot(~WaitTime, data = exitpoll, main = "Box-and-Whisker Display of the 208 Wait Times", xlab = "Wait Times")
```

Always use `main =` to create a title for a plot; always use `xlab =` to create a label for the X (horizontal) axis. Can you guess what you would use to create a label for the Y (vertical) axis?
Example 6

Here are final exam scores for the seven students in a seminar.

```r
scores <- c(86, 91, 81, 93, 88, 84, 95)
bwplot(~scores)
```
5. Graphic summaries for a single qualitative variable

\texttt{bargraph()}

The bargraph function will produce, for a qualitative variable (usually), a simple bar graph (sometimes called a bar chart).

If the variable is a column in a data frame, the format for the function is:
\texttt{bargraph(\sim \text{variable name}, \text{data} = \text{data frame name})}

If the variable is not part of a data frame, the format is simpler
\texttt{bargraph(\sim \text{variable name})}

(See Example 5 below)

\textbf{Example 1}

Obtain a bar chart for the variable Race in the hd data set.
\texttt{bargraph(\sim \text{Race, data} = \text{hd})}

The default color is light green and the default vertical scale is the counts (of the number of patients of each race). Both defaults can be modified.
Example 2
Repeat Example 1 but with red bars.

```
bargraph(~Race, data = hd, col = "red")
```

Example 3
Repeat Example 2 but with percentages, rather than counts, of each race on the vertical axis.

```
bargraph(~Race, data = hd, 
          col = "red", 
          type = "percent")
```

If you wish, you can replace counts on the vertical axis of the bar chart with proportions by changing “percent” to “proportion” in the bargraph command.
If you are obtaining a bar graph simply to get a sense of the relative frequency of the various categories of a qualitative variable, there is little need to provide helpful annotation (like a title). However, if you intend to include your graph in an article, you will want to include such annotation. In the next example you will provide a title and a label for the horizontal (X) axis.

**Example 4**

Obtain a default bar chart of the voters choice of candidate from the exitpoll data. Add the title “Bar Graph of the Voter’s Choice of Candidate” and create the x-label “Voter’s Choice”.

```r
bargraph(~Candidate, data = exitpoll, 
main = "Bar Graph of the Voter’s Choice of Candidate", 
  xlab = "Voter’s Choice")
```

Always use `main =` to create a title for a plot; always use `xlab =` to create a label for the X (horizontal) axis. Can you guess what you would use to create a label for the Y (vertical) axis?

**Example 5**

Here are the final grades for the 17 students in my Math 118 section several years ago.

```r

bargraph(~ grades)
```
5. Graphic summaries for a single qualitative variable

**pie()**

This function will produce a pie chart for a qualitative variable. R does not like pie charts and it does not make it easy to produce one. You must provide not the name of the variable but rather the tallies associated with the variable.

If the variable is a column in a data frame, the format for the function is:

```
pie(tally(~ variable name, data = data frame name))
```

If the variable is not part of a data frame, the format is simpler

```
pie(tally(~ variable name))
```

(See Example 4 below)

**Example 1**

Obtain a pie chart of the variable Race for the hd data.

```
pie(tally(~Race, data = hd))
```
You can do the same thing in two steps by saving the tallies as an object (t).

**Example 2**
Repeat Example 1 in two steps.

```r
t <- tally(~ Race, data = hd)
pie(t)
```

If you would like to add a title to the pie chart, you can do so with the `main` argument.

**Example 3**
Reproduce Example 1 but with the title “Pie Chart of Race”.

```r
pie(t, main = "Pie Chart of Race")
```
Example 4

Here are the final grades for the 17 students in my Math 118 section several years ago.

```r
T <- tally(~ grades)
pie(T)
```
6. Numeric summaries for two quantitative variables

cor()

This command will compute the Pearson correlation coefficient \((r)\) between two quantitative variables.

If both variables are columns in a data frame, the format for the function is:

\[
\text{lm}(\text{Y-variable name} \sim \text{X-variable name}, \text{data} = \text{data frame name})
\]

or

\[
\text{lm}(\text{X-variable name} \sim \text{Y-variable name}, \text{data} = \text{data frame name})
\]

If both variables are not part of a data frame, the format is simpler

\[
\text{lm}(\text{Y-variable name} \sim \text{X-variable name})
\]

or

\[
\text{lm}(\text{X-variable name} \sim \text{Y-variable name})
\]

(See Example 3 below)

Example 1

Using the data for the 32 patients in the hd data set, compute the correlation between SBP and Age.

Use either of the following:

\[
\text{cor}(\text{SBP} \sim \text{Age, data} = \text{hd})
\]

\[
[1] 0.7752041
\]

\[
\text{cor}(\text{Age} \sim \text{SBP, data} = \text{hd})
\]

\[
[1] 0.7752041
\]
Example 2
Using the data for the 208 voters in the exitpoll data set, compute the correlation between Age and years of education.

Use either of the following:

\[
\text{cor(Age ~ Educ, data = exitpoll)}
\]
\[
\text{[1]} \ -0.5708678
\]

\[
\text{cor(Educ ~ Age, data = exitpoll)}
\]
\[
\text{[1]} \ -0.5708678
\]

\[ r = -0.5709 \]

Note: As you know, the formula for the correlation coefficient is symmetric with respect to the naming of X and Y. Thus, each command above will give the same answer. However, to be consistent with other commands involving two variables you may prefer the form

\[
\text{cor(Y-variable name ~ X-variable name, data = data frame name)}
\]

Thus:

\[
\text{cor(SBP ~ Age, data = hd)}
\]
\[
\text{[1]} \ 0.7752041
\]

and

\[
\text{cor(Educ ~ Age, data = exitpoll)}
\]
\[
\text{[1]} \ -0.5708678
\]
Example 3

Here are the final semester GPA’s and the starting salaries (to the closest $1000) for five recent college graduates.

```
gpa <- c(3.2, 1.8, 2.7, 3, 2.3)
salary <- c(43, 34, 40, 46, 42)
```

What is the correlation between gpa and salary.

It is reasonable to think of salary as the Y variable in this case.

```
cor(salary ~ gpa)
[1] 0.83666
```
6. Numeric summaries for two quantitative variables

\texttt{lm()}

The \texttt{lm} command is used to obtain the equation of the ‘best’ linear relationship for predicting the value for a quantitative response variable (Y) from a quantitative explanatory variable (X). The letters \texttt{lm} stand for ‘linear model’

If both variables are columns in a data frame, the format for the function is:

\texttt{lm(Y-variable name \sim X-variable name, data = data frame name)}

If both variables are not part of a data frame, the format is simpler

\texttt{lm(Y-variable name \sim X-variable name)}

(See Example 4 below)

\textbf{Example 1}

For the hd data set, find the regression line predicting SBP from Age.

\texttt{lm(SBP \sim Age, data = hd)}

\begin{center}
\textbf{Coefficients:}
\begin{tabular}{ll}
\texttt{(Intercept)} & 59.092 \\
\texttt{Age} & 1.605 \\
\end{tabular}
\end{center}

The equation of the regression/least squares line is:

\hat{Y} = 59.092 + 1.605 X \quad \text{or, more usefully,}

\textbf{SBP} = 59.092 + 1.605 \text{Age}
Almost invariably we wish to access more information related to the regression line than just the intercept and the slope (for example, the predicted values of $Y$, the residuals, and the value for $r^2$). In this case it is better to save the regression as an object.

**Example 2**

As in Example 1, find the regression line relating SBP to Age but save the regression as an object called REG. Obtain the coefficient of determination ($r^2$).

```
REG <- lm(SBP ~ Age, data = hd)
REG
```

```
Call:
lm(formula = SBP ~ Age, data = hd)

Coefficients:
(Intercept)          Age
59.092        1.605

summary(REG)
```

```
Multiple R-squared:  0.6009
```

The command `REG <- lm(SBP ~ Age, data = hd)` will not generate any output but you need only type the name of the object to obtain the intercept and the slope as in Example 1. The summary command will generate a lot of information about the regression line including the value for $r^2$ (0.6009).

**Example 3**

Using the exitpoll data set, obtain the regression line relating years of education to age. Create the predicted/fitted values for years of education and the corresponding residuals. Create a data frame consisting of the ages, the actual years of education, the predicted years of education and the residuals. Name the data frame df and print the first 7 rows of df.
model <- lm(Educ ~ Age, data = exitpoll)
model

Coefficients:
(Intercept)          Age
 16.48921       -0.06815

The equation of the regression/least squares line is:

\[ \text{Educ} = 16.489 - 0.06815 \text{ Age} \]

fit <- fitted(model)
res <- resid(model)
df <- data.frame(exitpoll$Age, exitpoll$Educ, fit, res)
head(df, 7)

Example 4

Here are the final semester GPA’s and the starting salaries (to the closest $1000) for five recent college graduates.

gpa <- c(3.2, 1.8, 2.7, 3, 2.3)
salary <- c(43, 34, 40, 46, 42)

What is the equation of the regression line relating salary (Y) to gpa (X)?

mod <- lm(salary ~ gpa)
mod

Coefficients:
(Intercept)          gpa
  23.667          6.667
The equation of the regression/least squares line is:

\[ \text{salary} = 23.667 + 6.667 \text{ gpa} \]

7. Graphic summary for two quantitative variables
**xyplot()**

This command is used to obtain a scatterplot showing the relationship between a quantitative response variable (Y)—on the vertical axis and a quantitative explanatory variable (X)—on the horizontal axis.

If both variables are columns in a data frame, the format for the function is:

```r
xyplot(Y-variable name ~ X-variable name, data = data frame name)
```

If both variables are not part of a data frame, the format is simpler

```r
xyplot(Y-variable name ~ X-variable name)
```

(See Example 7 below)

**Example 1**

Using the data in hd, obtain a scatterplot showing the relationship between SBP (Y) and Age (X).

```r
xyplot(SBP ~ Age, data = hd)
```
You can easily modify the color of the symbols in the scatterplot:

**Example 2**

Repeat Example 1 with red symbols.

```r
xyplot(SBP ~ Age, data = hd, col = "red")
```

![Scatterplot with red symbols](image)

Note: Go to [http://www.stat.columbia.edu/~tzheng/files/Rcolor.pdf](http://www.stat.columbia.edu/~tzheng/files/Rcolor.pdf) for a look at the colors available in R.

You can change the symbols used in a dotplot (using the `pch` argument) and the size of the symbols (using the `cex` argument).

*(`pch` stands for printing character and, I think, `cex` stands for character expansion.)*

**Example 3**

Rework Example 1 with red dots, `pch = 19` and `cex = 1.5`.

```r
xyplot(~SBP, data = hd, 
       col = "red", 
       pch = 19, 
       cex = 1.5)
```
For lists of printing characters in R, Google R pch

The default value for cex is 1. The larger the value, the larger the symbol size. A value of 1.5 produces a symbol 50% bigger than the default; a value of 0.5 produces a symbol 50% smaller than the default.

With a little care, you can add a third variable to a scatterplot by using different colors for different categories of a qualitative variable. For instance, in the example below we use a red symbol for smokers (the variable Smoke is 1) and a blue symbol for non-smokers (the variable Smoke = 0). To do this we use the ifelse command.

**Example 4**

Reproduce the plot in Example 3 but using red symbols for smokers and blue symbols for non-smokers.

```r
xyplot(SBP ~ Age, data = hd, 
    col = ifelse(hd$Smoke == 1, "red", "blue"), 
    pch = 19, 
    cex = 1.5)
```
There are three components to the ifelse argument. The first, `hd$Smoke == 1` sets the *if* condition. For a smoker, `hd$Smoke` takes the value 1 and the corresponding symbol is defined by the second argument, "red". For non-smokers `hd$Smoke` is not 1 (but 0) and so the *else* condition is met and the third argument "blue" is used.

Notice the use of the double “=” in the argument `hd$Smoke == 1`.

**Example 5**

Using the exitpoll data, obtain a scatterplot of years of education against age. Use pink for female voters and blue for male voters.

```r
xyplot(Educ ~ Age, data = exitpoll, 
col = ifelse(exitpoll$Sex == "Female", "pink", "blue"), 
pch = 19, cex = 1.3)
```
Note: You would have obtained the same result if you had entered

\texttt{ifelse(exitpoll$Sex == "Male", "blue", "pink")}

Again, note the use of the double “=” in the argument \texttt{exitpoll$Sex == "Female"}.

If you are obtaining a scatterplot simply to get a sense of the nature of the relationship between two quantitative variables, there is little need to provide helpful annotation (like a title). However, if you intend to include your graph in an article, you will want to include such annotation. In the next example you will provide a title and a label for the vertical (Y) axis.

**Example 6**

Repeat Example 5 but add the title “Scatterplot of Years of Education against Age with Sex” and create the Y-label “Years of Education”.

\texttt{xyplot(Educ ~ Age, data = exitpoll,}
\texttt{col = ifelse(exitpoll$Sex == "Female", "pink", "blue"),}
\texttt{pch = 19,}
\texttt{cex = 1.3,}
\texttt{main = "Scatterplot of Years of Education against Age with Sex",}
\texttt{ylab = "Years of Education" )}
Always use *main* = to create a title for a plot; always use *ylab* = to create a label for the Y (vertical) axis. Can you guess what you would use to create a label for the X (horizontal) axis?

**Example 7**

Here are the final semester GPA’s and the starting salaries (to the closest $1000) for five recent college graduates.

```r
gpa <- c(3.2, 1.8, 2.7, 3, 2.3)
salary <- c(43, 34, 40, 46, 42)
```

Obtain a default scatterplot of salary (Y) against gpa (X) (except use cex = 2).

```r
xyplot(salary ~ gpa, cex = 2)
```
8. Numeric summaries for a quantitative response and a qualitative explanatory variable.

`sum()`, `mean()`, `median()`, `max()`, `min()`, `quantile()`, `IQR()`, `range()`, `sd()`, `var()`, `qdata()`, `favstats()`

In this context, we are interested in obtaining one of these summary statistics for a quantitative variable (Y) for each category of a qualitative variable (X).

If both variables are columns in a data frame, the format for the function is:

`sum(Y-variable name ~ X-variable name, data = data frame name)`
mean(Y-variable name ~ X-variable name, data = data frame name)
favstats(Y-variable name ~ X-variable name, data = data frame name)

If both variables are not part of a data frame, the format is simpler
sum(Y-variable name ~ X-variable name)
mean(Y-variable name ~ X-variable name)
favstats(Y-variable name ~ X-variable name)
(See Example 6 below)

**Example 1**
Using the data in hd, obtain the mean SBP for smokers and for non-smokers.
\[
\text{mean(SBP ~ Smoke, data = hd)} \\
0 \quad 1 \quad 137.7333 \quad 150.5294 \\
\]

**Example 2**
Using the data in hd, obtain the mean and the median SBP for smokers and for non-smokers.
\[
\text{mean(SBP ~ Smoke, data = hd); median(SBP ~ Smoke, data = hd)} \\
0 \quad 1 \quad 137.7333 \quad 150.5294 \\
0 \quad 1 \quad 137 \quad 149 \\
\]

**Example 3**
For the exitpoll data obtain all the favstats statistics for the ages for Obama voters and for Romney voters.
favstats(Age ~ Candidate, data = exitpoll)

<table>
<thead>
<tr>
<th>Candidate</th>
<th>min</th>
<th>Q1</th>
<th>median</th>
<th>Q3</th>
<th>max</th>
<th>mean</th>
<th>sd</th>
<th>n</th>
<th>missing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Obama</td>
<td>18</td>
<td>40.0</td>
<td>57</td>
<td>71.00</td>
<td>89</td>
<td>55.92500</td>
<td>19.08951</td>
<td>160</td>
<td>0</td>
</tr>
<tr>
<td>Romney</td>
<td>20</td>
<td>40.5</td>
<td>67</td>
<td>81.75</td>
<td>89</td>
<td>60.20833</td>
<td>22.53221</td>
<td>48</td>
<td>0</td>
</tr>
</tbody>
</table>

Example 4

For the exitpoll data, obtain the five-number summary of the years of education for Obama voters and for Romney voters.

quantile(Age ~ Candidate, data = exitpoll)

<table>
<thead>
<tr>
<th>Candidate</th>
<th>0%</th>
<th>25%</th>
<th>50%</th>
<th>75%</th>
<th>100%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Obama</td>
<td>18</td>
<td>40.0</td>
<td>57</td>
<td>71.00</td>
<td>89</td>
</tr>
<tr>
<td>Romney</td>
<td>20</td>
<td>40.5</td>
<td>67</td>
<td>81.75</td>
<td>89</td>
</tr>
</tbody>
</table>

Example 5

For the exitpoll data, obtain the 80th percentile of the years of education for Obama voters and for Romney voters.

qdata(Age ~ Candidate, p = 0.8, data = exitpoll)

<table>
<thead>
<tr>
<th>Candidate</th>
<th>p</th>
<th>quantile</th>
</tr>
</thead>
<tbody>
<tr>
<td>Obama</td>
<td>0.8</td>
<td>75.2</td>
</tr>
<tr>
<td>Romney</td>
<td>0.8</td>
<td>84.6</td>
</tr>
</tbody>
</table>

Example 6

Here are the final grades for the 18 students in my Math 118 section several years ago.


For the same students, here are their scores on the first exam:
Here are the mean and median first exam scores for each final grade.

```r
mean(exam1 ~ grades)
  A     B     C
89.80 85.90 72.33

median(exam1 ~ grades)
  A  B  C
  90 86 76
```

9. Graphical summary for a quantitative response and a qualitative explanatory variable

`bwplot()`

A box-and-whisker display is most effective when used to show not a single distribution but to compare the distribution of a quantitative response variable (Y) for the different categories of a qualitative explanatory variable (X).

If both variables are columns in a data frame, the format for the function is:
bwplot(Y-variable name ~ X-variable name, data = data frame name)

If both variables are not part of a data frame, the format is simpler

bwplot(Y-variable name ~ X-variable name)

(See Example 6 below)

**Example 1**

Using the hd data, obtain a comparative box-and-whisker display of SBP (Y) for each of the three races of patient (X).

bwplot(SBP ~ Race, data = hd)

![Box-and-whisker plot](image)

You can select the color of the dot representing the median with the `color` argument and you can select a color to fill the ‘box’ with the `fill` argument.

**Example 2**

Repeat Example 1 but with a red dot for the median and lavender fill.

bwplot(SBP ~ Race, data = hd, 
col = "red", 
fill = "lavender")
By default, R will produce vertical box-and-whisker displays. If you would like horizontal plots, you need only reverse the placement of X and Y in the bwplot command.

**Example 3**

Repeat Example 2 but with the box-and-whisker plots aligned horizontally.

```r
bwplot(Race ~ SBP, data = hd,
       col = "red",
       fill = "lavender")
```

Notice that the only annotation on these plots is the label associated with the quantitative variable (SBP). This fine if you are interested only in exploring the relationship between SBP and Race. However, if you plan to include your plot as part of a publication, you will want to add at least a title.
Example 4

To the plot in Example 3, add a title “Boxplots of SBP by Race’ and the label “Race” for the vertical (Y) axis.

```r
bwplot(Race ~ SBP, data = hd,
       col = "red",
       fill = "lavender",
       ylab = "Race",
       main = "Boxplots of SBP by Race")
```

Always use `main` = to create a title for a plot; always use `ylab` = to create a label for the Y (vertical) axis. Can you guess what you would use to create a label for the X (horizontal) axis?

Example 5

Again, refer to the hd data set. Obtain default box-and-whisker displays of SBP for smokers and non-smokers.

Recall that the variable Smokes, though qualitative, uses the values 0 and 1; 1 for a smoker and 0 for a non-smoker.

```r
bwplot(SBP ~ Smoke, data = hd)
```
This is a crazy mess! What is going on? R has seen that the second variable, Smoke has numbers (0’s and 1’s) and so assumes that we want horizontal boxplots of Smoke for each value of SBP. Hence the result above. The simplest solution to this problem is to create a new variable (let’s call it Smokes rather than Smoke) that takes the label Smoker if Smoke = 1 and Non-smoker if Smoke = 0. We will add this new variable to the data frame. We can create this new variable easily with the *ifelse* command as shown below.

```r
hd$Smokes <- ifelse(hd$Smoke == 1, "Smoker", "Non-smoker")
```

If the original variable hd$Smoke takes the 1, then the new variable hd$Smokes will take the label Smoker; *else* it will take the label Non-smoker.

Notice the use of the double ‘=’ in the initial condition.

Now we will be able to obtain box-and-whisker displays of SBP for those that smoke and those that don’t.

```r
bwplot(SBP ~ Smokes, data = hd)
```
Example 6

Here are the final grades for the 18 students in my Math 118 section several years ago.

```r
```

For the same students, here are their scores on the first exam:

```r
exam1 <- c(81, 91, 60, 81, 76, 86, 91, 81, 91, 76, 90, 78, 93, 99, 92, 86, 86, 87)
```

Obtain (default) box-and-whisker displays of first-exam scores by final grade.

```r
bwplot(exam1 ~ grades)
```

10. Numeric summary for two qualitative variables

`tally()`

For a single qualitative variable, the tally command, at its most basic, counts the number of times that each category of the variable occurs. In the context of two qualitative variables, the tally command counts the number (or percent or proportion) of times that a response variable (Y) occurs separately for each category of a response variable (X).
If both variables are columns in a data frame, the format for the function is:

`tally(Y-variable name ~ X-variable name, data = data frame name)`

If both variables are not part of a data frame, the format is simpler

`tally(Y-variable name ~ X-variable name)`

(See Example 6 below)

**Example 1**

For the hd data, obtain counts of the number of smokers (and non-smokers) by race. Here, X is the explanatory variable race and Y, the response variable Smoke.

`tally(Smoke ~ Race, data = hd)`

<table>
<thead>
<tr>
<th>Race</th>
<th>Smoke 0</th>
<th>Smoke 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Black</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>Hispanic</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>White</td>
<td>10</td>
<td>10</td>
</tr>
</tbody>
</table>

The explanatory variable (X) forms the columns of the table; the response variable (Y) form the rows. Two (2) of the Black patients smoke while 10 of the White patients smoke.

It is more insightful to ask for the percentage of each race that smoke (and not smoke). We can obtain such percentages with the *format* argument.

**Example 2**

For the hd data, obtain the percent of each race that are smokers (and non-smokers).

`tally(Smoke ~ Race, data = hd, format = "percent")`
Fifty-percent of White patients smoke.

Referring to Example 2, you don’t really need percentages to five decimal places; one decimal place will do. In the next example, we save the tally output as an object and round the percentages to one decimal place.

**Example 3**

```r
t <- tally(Smoke ~ Race, data = hd, format = "percent")
round(t, 1)
```

<table>
<thead>
<tr>
<th>Race</th>
<th>Smoke</th>
<th>Black</th>
<th>Hispanic</th>
<th>White</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>71.4</td>
<td>0.0</td>
<td>50.0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>28.6</td>
<td>100.0</td>
<td>50.0</td>
<td></td>
</tr>
</tbody>
</table>

Note: If you prefer proportions to percentages, simply change the format argument to

`format = "proportion"`.

In the examples above we can add the column totals to the table by using the ‘margins = TRUE’ argument (actually, margins = T will be sufficient.)

**Example 4**

For the hd data, obtain counts of the number of smokers (and non-smokers) by race. Add the column totals.

```r
tally(Smoke ~ Race, data = hd, margins = T)
```
Race
Smoke  Black  Hispanic  White
0       5      0       10
1       2      5       10
Total   7      5       20

Example 5
For the exitpoll data, obtain counts of the number of voters that voted for each candidate by sex. Add the column totals.

Repeat the table above but with percentages rather than counts.

tally(Candidate ~ Sex, data = exitpoll, margins = T)

<table>
<thead>
<tr>
<th>Sex</th>
<th>Candidate</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Obama</td>
</tr>
<tr>
<td>Female</td>
<td>90</td>
</tr>
<tr>
<td>Male</td>
<td>70</td>
</tr>
<tr>
<td>Total</td>
<td>109</td>
</tr>
</tbody>
</table>

tally(Candidate ~ Sex, data = exitpoll, format = "percent", margins = T)

<table>
<thead>
<tr>
<th>Sex</th>
<th>Candidate</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Obama</td>
</tr>
<tr>
<td>Female</td>
<td>82.56881</td>
</tr>
<tr>
<td>Male</td>
<td>70.70707</td>
</tr>
<tr>
<td>Total</td>
<td>100.00000</td>
</tr>
</tbody>
</table>

Example 6
Here are the final grades for the 18 students in my Math 118 section several years ago.


For the same students, the variable level contains an indication of whether each student was upper class (Junior or Senior) or lower class (Freshman or Sophomore)

Obtain a table showing the counts of grades (Y) by level (X).

\[
\text{tally(grades } \sim \text{ level)}
\]

<table>
<thead>
<tr>
<th>level</th>
<th>grades</th>
<th>L</th>
<th>U</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>3</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>7</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>2</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

11. Some useful functions

log(), exp(), sqrt(), round(), abs()

For these functions, the parentheses may contain a scalar (a single number) or a variable (a column of numbers).

log() computes the logarithm to the base e of the content of the parentheses.
exp() computes the value for $e$ raised to the power of the content of the parentheses.
sqrt() computes the square root of the content of the parentheses.
round() will round the content of the parentheses.
abs() computes the absolute value of the content of the parentheses.

The first example applies these functions to scalars (single values):

**Example 1**

(a) Find $\sqrt{9.56}$

```
sqrt(9.56)
[1] 3.091925
```

(b) Find $e^{0.5}$

```
exp(0.5)
[1] 1.648721
```

(c) Find $\log_e(118.6) = \ln(118.6)$

```
log(118.6)
[1] 4.775756
```

(d) Find $\sqrt{\frac{63.9}{24.1}}$

```
sqrt(63.9/24.1)
[1] 1.628328
```

(e) Find the absolute value for $\frac{21.6 - 43.2}{24.1}$

```
abs((21.6 - 43.2)/24.1)
[1] 0.8962656
```
Note the internal parentheses. Without them we would have obtained the absolute value for 21.6 – 43.2/24.1.

(f) Round the number 4.68 to the nearest whole number.

`round(4.68)`

[1] 5

(g) Round the number 0.468 to the nearest whole number.

`round(0.468)`

[1] 0

(h) Round the number 0.468 to two decimal places.

`round(0.468, 2)`

[1] 0.47

(i) Round the number 71.468 to one decimal place.

`round(71.468, 1)`

[1] 71.5

Note: `round(x)` rounds x to the nearest whole number and is equivalent to `round(x, 0)`.

These are base-R rather than mosaic functions and, as you can see below, the format is different from the format in mosaic functions.

If the variable is a column in a data frame, the format for the function is:

`log(data frame name$variable name)`

`exp(data frame name$variable name)`

:::
abs(data frame name$variable name)

If the variable is not part of a data frame, the format is simpler

log(variable name)

exp(variable name)

abs(variable name)

(See Example 2 below)

**Example 2**

The variable gain contains the one-year percentage gain in value for four stocks (a negative value indicates a percentage loss).

gain <- c(4.36, -1.61, 2.42, 6.66)

(a) Find the absolute values for these gains.

```r
abs(gain)
[1]  4.36  1.61  2.42  6.66
```

(a) Round these gains to one decimal place and save them as a new variable, gain1.

```r
gain1 <- round(gain, 1)
gain1
[1]  4.4  -1.6  2.4  6.7
```

Note: Notice that you function you use is applied to each value for the variable.

**Example 3**

Refer to the hd data set. Obtain a new variable called SBP_SQ which contain the square roots of each of the SBPs.

```r
SBP_SQ <- sqrt(hd$SBP)
```
Example 4

Repeat Example 3 but this time make the new variable a part of the hd data set.

```r
hd$SBP_SQ <- sqrt(hd$SBP)
```

The `log()` and the `sqrt()` functions are helpful when a variable takes values over several orders of magnitude. This invariably means that the data is very skewed. Taking logs or square roots often brings the new values closer to bell-shaped—which is often desirable in statistics. The next example will illustrate this point.

Example 5

The data frame `statepop.csv` contains the names of the 50 states and their population size (pop).

(a) Obtain a histogram of population size.

```r
histogram(~pop, data = statepop)
```

(b) Create a new variable (called `lnpop`) that is the logarithm to the base e of population size. Obtain a histogram of the `lnpop` values.

```r
lnpop <- log(statepop$pop)
histogram(~lnpop)
```
(c) Create a new variable (called sqpop) that is the square root of population size. Obtain a histogram of the sqpop values.

```
sqpop <- sqrt(statepop$pop)
histogram(~sqpop)
```

12. Probability Distributions

`xpnorm()`
The `xpnorm` function will compute the area under a normal distribution with mean \( \mu \) and standard deviation \( \sigma \) to the left of a specified value, \( A \) (as shown below).

The \( \text{N}(\mu, \sigma) \) distribution

The format for the `xpnorm` function is `xpnorm(A, \mu, \sigma)`.

**Example 1**

The distribution of midday temperature in the summer in Boston is approximately normal with mean 79°F and standard deviation 7.5°F.

(a) On what proportion of days will the midday temperature be below 75°F?

We require the area under the \( \text{N}(79, 7.5) \) distribution to the left of 75.

\[
\text{xpnorm}(75, 79, 7.5)
\]

If \( X \sim \text{N}(79, 7.5) \), then

\[
\begin{align*}
P(X \leq 75) &= P(Z \leq -0.5333) = 0.2969 \\
P(X > 75) &= P(Z > -0.5333) = 0.7031
\end{align*}
\]
Almost 30% of summer days have a midday temperature below 75.

The output mimics the calculation used to find this area using the Z-tables. The area to the left of 75 under the $N(79, 7.5)$ distribution is equal to the area under the $N(0, 1)$ distribution to the left of $-0.5333$. The output also includes a helpful picture showing the required area in orange.

(b) On what proportion of days will the midday temperature exceed 85°F?

We require the area under the $N(79, 7.5)$ distribution to the right of 85.

Recall that the total area under a normal distribution is 1 so we need only subtract the area to the left of 85 from 1.

$$1 - \text{xpnorm}(85, 79, 7.5)$$

If $X \sim N(79, 7.5)$, then
\[ P(X \leq 85) = P(Z \leq 0.8) = 0.7881 \]
\[ P(X > 85) = P(Z > 0.8) = 0.2119 \]

[1] 0.2118554

A little over 21% of summer days have midday temperatures above 85.

The area to the right of 85 under the N(79, 7.5) distribution is equal to the area under the N(0, 1) distribution to the right of 0.8. The output also includes a helpful picture showing the required area in teal.

![Diagram showing areas under normal distributions](image)

The `xpnorm` function can be used to compute areas/probabilities associated with the (Z) distribution with mean 0 and standard deviation 1.

**Example 2**

For the Z distribution, compute \( P(Z < 1.07) \), the probability that Z is less than 1.07. This is the area under the Z-distribution to the left of 1.07.

\[ xpnorm(1.07, 0, 1) \]
If $X \sim N(0, 1)$, then

\[
\begin{align*}
P(X \leq 1.07) &= P(Z \leq 1.07) = 0.8577 \\
P(X > 1.07) &= P(Z > 1.07) = 0.1423
\end{align*}
\]

[1] 0.8576903

The required area is 0.8577.

In fact, there is no need to include the 0 and the 1 in the function in Example 2. If you are computing an area under the $Z$ distribution you need only include the value for $A$ in the pnorm function; with only a single value, R will assume that $\mu = 0$ and $\sigma = 1$.

`xpnorm(1.07)`

If $X \sim N(0, 1)$, then

\[
\begin{align*}
P(X \leq 1.07) &= P(Z \leq 1.07) = 0.8577 \\
P(X > 1.07) &= P(Z > 1.07) = 0.1423
\end{align*}
\]

[1] 0.8576903
12. Probability Distributions

\texttt{xqnorm()}
The `xqnorm` function does the opposite of `xpnorm`. Instead of finding the area to the left of a specified value under a $N(\mu, \sigma)$ distribution, `xqnorm` finds the value (the quantile) that has a specified area to the left.

The $N(\mu, \sigma)$ distribution

![Diagram of normal distribution with quantile](image)

The format for the `qnorm` command is:

```
xqnorm(K, \mu, \sigma)
```

where $K$ is a value between 0 and 1.

**Example 1**

The distribution of midday temperature in the summer in Boston follows approximately a normal distribution with mean $79^\circ F$ and standard deviation $7.5^\circ F$.

(a) What is the 25th percentile of temperature ($X_{25}$)?
If $X \sim N(79, 7.5)$, then

$$P(X \leq 73.94133) = 0.25 \quad P(X > 73.94133) = 0.75$$

[1] 73.94133

The value for temperature that has 25% of the distribution to the left is 73.9°F. That is, on 25% of summer days the midday temperature is less than 73.9°F.

R includes a picture showing the specified area (0.25) in orange and noting that $P(X \leq 74) = 0.25$. In standard (Z) units, the 25th percentile is -0.67.

(b) What is the 70th percentile of temperature ($X_{70}$)?
If $X \sim N(79, 7.5)$, then

$$P(X \leq 82.933) = 0.7$$
$$P(X > 82.933) = 0.3$$

[1] 82.933

On 70% of summer days the midday temperature is less than 82.9°F.

R includes a picture showing the specified area (0.70) in orange and noting that $P(X \leq 83) = 0.70$. In standard (Z) units, the 70th percentile is 0.52.
The `xqnorm` function can be used to compute percentiles associated with the \((Z)\) distribution with mean 0 and standard deviation 1.

**Example 2**

What is the 90\(^{th}\) percentile of the Z distribution \((Z_{90})\)?

\[
xqnorm(0.9, 0, 1)
\]

If \(X \sim N(0, 1)\), then

\[
\begin{align*}
P(X \leq 1.281552) &= 0.9 \\
P(X > 1.281552) &= 0.1
\end{align*}
\]
The 90th percentile of the Z distribution is 1.282.

In fact, there is no need to include the 0 and the 1 in the command in Example 2. If you are finding a percentile for the Z distribution you need only include the value for K/100 in the qnorm function; with only a single value, R will assume that \( \mu = 0 \) and \( \sigma = 1 \).

\[ x_{\text{qnorm}}(0.9) \]

If \( X \sim N(0, 1) \), then

\[
\begin{align*}
P(X \leq 1.281552) &= 0.9 \\
P(X > 1.281552) &= 0.1
\end{align*}
\]
12. Probability Distributions

\textbf{rnorm()}

The \texttt{rnorm} function is used to select a random sample of specified size \(n\) from a normal distribution with specified mean \(\mu\) and specified standard deviation, \(\sigma\).

\begin{equation*}
\text{N(}\mu, \sigma) \text{ distribution}
\end{equation*}

The format for the \texttt{xpnorm} function is:

\texttt{rnorm(n µ, σ)}

\textbf{Example 1}

Select a random sample of size \(n = 20\) from the normal distribution (of IQ) with mean 100 and standard deviation 15. Save the sample with the name \texttt{sample}.

Have R print the first five values in the sample. Compute the sample mean and the sample standard deviation.

\begin{verbatim}
sample <- rnorm(20, 100, 15)
head(sample, 5)
[1]  95.91556  95.26977  90.57617  98.40304 106.42022
mean(~ sample)
[1] 99.30154
sd(~ sample)
[1] 12.04059
\end{verbatim}
12. Probability Distributions

xpt()

The xpt function will compute the area to the left of a specified value \( A \) of a \( t \) distribution with a specified number of degrees of freedom \( (v) \).

\[ \text{xpt}(A, \text{df} = v) \]

where \( \text{df} \) is the number of degrees of freedom.

**Example 1**

Refer to the \( t \) distribution with 5 degrees of freedom. Find the area to the left of -2.03 under this distribution.
\texttt{xpt(-2.03, df = 5)}

\begin{verbatim}
[1] 0.04905559
\end{verbatim}

The required area is 0.049.

Here is the picture that accompanies the output.

The required area (0.049) is in orange.

\textbf{Example 2}

Refer to the t distribution with 24 degrees of freedom. Find the area to the \textit{right} of 1.45 under this distribution.
The area under every t distribution is 1. So, the area to the right of 1.45 is $1 - \text{the area to the right}$.

\[ 1 - \text{xpt}(1.45, \text{df} = 24) \]

[1] 0.08000109

The required area is 0.080.

The required area (0.08) is in teal.

**Example 3**

Refer to the t distribution with 24 degrees of freedom. Find the sum of two areas; the area to the left of $-1.45$ and the area to the right of 1.45.

(a) Find the two areas separately and add them.
\[ xpt(-1.45, \text{df} = 24) + (1 - xpt(1.45, \text{df} = 24)) \]

\[ [1] \ 0.1600022 \]

(b) Since t distributions are symmetric, the two areas are equal; so find the area to the left of \(-1.45\) and multiple it by 2.

\[ 2 \times xpt(-1.45, \text{df} = 24) \]

\[ [1] \ 0.1600022 \]

The required area is 0.160.
12. Probability Distributions

The xqt function does the opposite of xpt. Instead of finding the area to the left of a specified value under a t distribution with a specified number of degrees of freedom (v), xqt finds the value (the quantile) that has a specified area to the left.

The format for the xqt function is:

\[
xqt(K, \text{df} = v)
\]

where df stands for the number of degrees of freedom and K is a value between 0 and 1.

Example 1

Obtain the 95th percentile of the t_{24} distribution.
The 95\textsuperscript{th} percentile of the $t_{24}$ distribution is 1.711.

By definition, an area of 0.05 lies to the right of the 95\textsuperscript{th} percentile, 1.711. Recall that $t$ distributions are symmetric around 0. So, if we break off a similar area of 0.05 to the left of $-1.711$ we have an area of 0.9 between $-1.711$ and 1.711.

The 95\textsuperscript{th} percentile (1.711) is also the value for $t_{24}$ that contains the central 90\% of the $t_{24}$ distribution; that is 90\% of the $t_{24}$ distribution lies between $-1.711$ and 1.711.
In statistics we frequently need values for \( t \) that contain the *central 90%*, the *central 95%*, the *central 99%* ... of a \( t \) distribution.

You can easily translate central areas into areas entirely to the left with the following chart.

<table>
<thead>
<tr>
<th>Area in middle</th>
<th>Area to the left</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.80</td>
<td>0.9</td>
</tr>
<tr>
<td>0.85</td>
<td>0.925</td>
</tr>
<tr>
<td>0.90</td>
<td>0.95</td>
</tr>
<tr>
<td>0.92</td>
<td>0.96</td>
</tr>
<tr>
<td>0.95</td>
<td>0.975</td>
</tr>
<tr>
<td>0.98</td>
<td>0.99</td>
</tr>
<tr>
<td>0.99</td>
<td>0.995</td>
</tr>
</tbody>
</table>

For instance, if you need the value that contains the central 80% of the \( t_{10} \) distribution, find the 90\(^{th}\) percentile.

**Example 2**

Find the value that contains the central 80% of the \( t_{10} \) distribution. You need only find the 90\(^{th}\) percentile of the \( t_{10} \) distribution.
The value that contains the central 80% of the $t_{10}$ distribution is 1.372.

**Example 3**

Find the value that contains the central 95% of the $t_{40}$ distribution. You need only find the 97.5\textsuperscript{th} percentile of the $t_{40}$ distribution.

The value that contains the central 95% of the $t_{40}$ distribution is 2.021.
13. Statistical Inference

\texttt{t.test()} \hspace{1cm} \textbf{(one sample)}

In the context of a single sample, the \texttt{t.test} function is used to test a hypothesis about a population mean, $\mu$ (the Greek mu) and to find confidence intervals for $\mu$.

If the variable is a column in a data frame, the default format for the function is:

\texttt{t.test(~ variable name, data = data frame name, mu = $\mu_0$)}

If the variable is not a column in a data frame, the format is simpler

\texttt{t.test(~ variable name, mu = $\mu_0$)}

(See Example 4 below)

\textbf{Example 1}

Think of the 32 patients in the \texttt{hd} data set as a random sample of heart disease patients. Perform a two-sided test of the null hypothesis that $\mu$, the mean BMI in the population is 24.9 (this is the upper point of the range of ‘normal’ BMI values).

\texttt{t.test(~BMI, data = hd, mu = 24.9)}

\begin{verbatim}
One Sample t-test
data:  hd$BMI
t = 1.7598, df = 31, p-value = 0.08831 alternative hypothesis: true mean is not equal to 24.9
95 percent confidence interval:
 24.76936 26.67439 sample estimates:
mean of x
 25.72188
\end{verbatim}

Note: We did not specify a two sided test; that is the default.
Note: By default the output includes a 95% confidence interval for the unknown population mean, \( \mu \).

Note: It is simple to use R as a calculator to verify the value for the test statistic,

\[
t = \frac{\bar{X} - \mu_0}{\frac{S}{\sqrt{n}}}
\]

\[
m <- \text{mean}(\sim \text{BMI}, \text{data} = \text{hd})
\]
\[
s <- \text{sd}(\sim \text{BMI}, \text{data} = \text{hd})
\]
\[
t <- (m - 24.9)/(s/sqrt(32))
\]
\[
t
\]
\[
[1] 1.75978
\]

**Example 2**

In Example 1 we may prefer to perform a one-sided test with the alternative being \( H_A: \mu > 24.9 \). R will do this if we add the argument `alternative = "greater"`.

\[
t.\text{test}(\sim \text{BMI}, \text{data} = \text{hd}, \text{mu} = 24.9, \text{alternative} = "greater")
\]

One Sample t-test

data:  hd$BMI
t = 1.7598, df = 31, p-value = 0.04416 alternative hypothesis: true mean is greater than 24.9 95 percent confidence interval:
24.93001     Inf sample estimates:
mean of x
25.72188

Note: Asking R for a one-sided test also produces a one-sided 95% confidence interval (in this case we can be 95% confident that \( \mu > 24.93 \)).

Note: If you prefer the alternative ‘less than’ rather than ‘greater than’ simply replace ‘greater’ with ‘less’ in the t.test format.
Example 3

By default, R produces a 95% confidence interval for $\mu$, but you can select a confidence level other than 95%. Repeat Example 1, but ask for an 80% confidence interval.

```r
t.test(~BMI, data = hd, mu = 24.9, conf.level = 0.8)
```

One Sample t-test

data:  BMI
t = 1.7598, df = 31, p-value = 0.08831
alternative hypothesis: true mean is not equal to 24.9
80 percent confidence interval: 
25.11031 26.33344
sample estimates:
mean of x
25.72188

Example 4

Here are final exam scores for the seven students in a seminar

```r
scores <- c(86, 91, 81, 93, 88, 84, 95)
```

It is claimed by the manufacturer that cartons of its cereal weigh an average of 510 grams. A family wonders if this figure is an overstatement. When they carefully weigh seven unopened cartons of the cereal the results are as below.

```r
weight <- c(503, 512, 510, 505, 508, 507, 506)

```

t.test(~ weight, mu = 510, alternative = "less")

One Sample t-test

data:  weight
t = -2.3627, df = 6, p-value = 0.02804
alternative hypothesis: true mean is less than 510
95 percent confidence interval:
These results suggest that the sample average weight (507.3 grams) is statistically significantly less than 510 grams—though probably the difference is not of any practical significance.

**Example 5**

In some cases we want to test the null hypothesis $H_0: \mu = 0$. This arises frequently when the variable of interest is the *difference* between two measurements. Here is an example. In a town in West Virginia there are two real estate agents Allen and Baldwin. Each of six house owners in the town ask each broker for an approximate asking price for their house. The results (in $1000$s) were entered into R, saved as the data frame *hprice.csv* and shown below. The last column (variable) is the (Allen – Baldwin) difference in prices.

```
head(hprices, 6)
  Allen Baldwin  diff
  1   325     280    45
  2   199     210 -11
  3   345     325    20
  4   270     250    20
  5   380     355    25
  6   280     255    25
```

Think of $\mu$ in this case as the (hypothetical) mean (Allen – Baldwin) difference in suggested price. We will perform a two-sided test of $H_0: \mu = 0$ and find a 95% confidence interval for $\mu$.

As it happens, if you omit a null value for $\mu$ in the *t.test* function, R assumes that you mean a value of 0.

```
t.test(~ diff, data = hprices)
  One Sample t-test

data:  diff
```

-Inf 509.518
sample estimates:
mean of x
507.2857
t = 2.8009, df = 5, p-value = 0.03795
alternative hypothesis: true mean is not equal to 0
95 percent confidence interval:
  1.699248 39.634085
sample estimates:
mean of x
  20.66667

The p-value (0.03795) suggests that $\mu$ is significantly greater than 0 and we can be 95% confident that the (population) mean (Allen – Baldwin) price suggestions lies between $1700 and $3963.
13. Statistical Inference

t.test() (two independent samples)

In the context of two independent samples, the t.test function is used to test for the equality of two population means ($\mu_1$ and $\mu_2$) and to find a confidence interval for $\mu_1 - \mu_2$. The response variable (Y) is quantitative and the explanatory variable (X) is (usually) qualitative with just two categories.

If both variables are columns in a data frame, the default format for the t.test function is:

```
t.test(Y-variable name ~ X-variable name, data = data frame name)
```

If both variables are not part of a data frame, the format is simpler

```
t.test(Y-variable name ~ X-variable name)
```

(See Example 5 below)

**Example 1**

An educator believes that new reading activities for elementary school children will improve reading comprehension scores. She randomly assigns third-graders to an eight-week program in which some will use these activities and others will experience traditional teaching methods. At the end of the experiment both groups take a reading comprehension exam. Their scores are in the data set reading.csv. We will use the default two-sample t test to compare the two sets of scores.

```
t.test(score ~ group, data = reading)
```

Welch Two Sample t-test

data:  score by group
t = 2.3109, df = 37.855, p-value = 0.02638
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval: 
  1.23302 18.67588
sample estimates: 
  mean in group active mean in group control 
  51.47619 41.52174

The p-value suggests that the sample mean score in the active reading group is significantly greater than that for the control group.

Note: We did not specify a two sided test; that is the default.

Note: By default the output includes a 95% confidence interval for the difference in population means, \( \mu_{\text{Active}} - \mu_{\text{Control}} \).

Note: It is not too difficult to use R as a calculator to verify the value for the test statistic,

\[
t = \frac{\bar{Y}_1 - \bar{Y}_2}{\sqrt{\frac{S_1^2}{n_1} + \frac{S_2^2}{n_2}}}
\]

\[
favstats(score ~ group, data = reading)
group  min   Q1 median   Q3 max     mean       sd  n missing
1  active  24 44.0     53 58.0  71 51.47619 11.00736 21       0
2  control 10 30.5     42 53.5  85 41.52174 17.14873 23       0

\[
t \leftarrow \frac{(51.47619 - 41.52174)/\sqrt{(11.00736^2/21 + 17.148732^2/23)}}
\]

\[
[1] 2.310889
\]

How did R know that we were interested in the difference

\[
\bar{Y}_{\text{Active}} - \bar{Y}_{\text{Control}} = 51.47619 - 41.52174 = 9.95445 \rightarrow t = 2.310889
\]

rather than:

\[
\bar{Y}_{\text{Control}} - \bar{Y}_{\text{Active}} = 41.52174 - 51.47619 = -9.95445 \rightarrow t = -2.310889
\]
The answer is that the A in Active is closer to the beginning of the alphabet than the C in Control!
You always have to keep the implied rule in mind particularly with one-sided tests.

**Example 2**

In the context of Example 1, it might be more plausible to perform a one-sided test of the form

\[ H_A: \mu_{\text{Active}} - \mu_{\text{Control}} > 0. \]

R will do this if we add the argument `alternative = "greater"`.

```r
t.test(score ~ group, data = reading, alternative = "greater")
```

```
Welch Two Sample t-test
data:  score by group
t = 2.3109, df = 37.855, p-value = 0.01319
alternative hypothesis: true difference in means is greater than 0
95 percent confidence interval:
  2.691293      Inf
sample estimates:
mean in group active mean in group control
 51.47619              41.52174
```

Note: Asking R for a one-sided test also produces a one-sided 95% confidence interval (in this case we can be 95% confident that \( \mu_{\text{Active}} - \mu_{\text{Control}} > 2.69 \)).

Note: If you prefer the alternative ‘less than’ rather than ‘greater than’ simply replace ‘greater’ with ‘less’ in the `t.test` format.
Example 3

By default, R produces a 95% confidence interval for \( \mu_{\text{Active}} - \mu_{\text{Control}} \), but you can select a confidence level other than 95%. Repeat Example 1, but ask for an 80% confidence interval.

```r
t.test(score ~ group, data = reading, conf.level = 0.8)
```

Welch Two Sample t-test

data:  score by group
t = 2.3109, df = 37.855, p-value = 0.02638
alternative hypothesis: true difference in means is not equal to 0
80 percent confidence interval: 4.335933 15.572970
sample estimates:
mean in group active  mean in group control
51.47619              41.52174

Example 4

Refer to the exitpoll data set. Perform a two-sided, two-sample t test to check whether there is a significant difference in mean age by candidate.

```r
t.test(Age ~ Candidate, data = exitpoll)
```

Welch Two Sample t-test

data:  Age by Candidate
t = -1.1947, df = 68.482, p-value = 0.2363
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval: -11.436853  2.870187
sample estimates:
mean in group Obama  mean in group Romney
55.92500              60.20833
The mean age of the Romney voters ($\bar{Y}_{\text{Romney}} = 60.2$ years) is not significantly
greater (p-value = 0.2363) than that for the Obama voters ($\bar{Y}_{\text{Obama}} = 55.9$ years).
Notice that the value for t (- 1.1947) is negative since R computes $\bar{Y}_{\text{Obama}} - \bar{Y}_{\text{Romney}}$
(O being closer to the beginning of the alphabet than R).

**Example 5**
A random sample of seven people selected from close to the front of a large line
waiting for 400 tickets for a major sporting event. Each person is asked to estimate
the total number of people in line. Another six people randomly selected from
toward the back of the line are asked the same questions. Here are the 13 estimates:

```r
estimate <- c(427, 432, 401, 520, 450, 480, 460, 442,
        421, 364, 389, 394, 452)
```

Here are the corresponding positions for the 13 people:

```r
position <- c("Front", "Front", "Front", "Front",
        "Front", "Front", "Back", "Back", "Back",
        "Back","Back","Back")
```

Do these data suggest that, on average, people toward the front of the line make
significantly larger estimates than those at the back? Perform the appropriate one
sided t-test.

The two groups to be compared are ‘Front’ and ‘Back’. So, R will test the null hypothesis $H_0$: $\mu_{\text{Back}} - \mu_{\text{Front}} = 0$ (since B is closer than F to the beginning of the alphabet). To answer the research question above, we need to use the alternative
$H_A$: $\mu_{\text{Back}} - \mu_{\text{Front}} < 0$.

```r
t.test(estimate ~ position, alternative = "less")
```

```
Welch Two Sample t-test
data:  estimate by position
t = -2.107, df = 10.992, p-value = 0.02945
alternative hypothesis: true difference in means is less than 0
```
95 percent confidence interval:
  \(-\infty \text{ -6.27721}\)

Sample estimates:
mean in group Back mean in group Front
  \(410.3333 \text{ \text{ 452.8571}}\)

The p-value (0.02945) suggest that \(\mu_{\text{Back}} - \mu_{\text{Front}} < 0\); that is that \(\mu_{\text{Front}} > \mu_{\text{Back}}\).
14. Data organization

filter()

This function is used to select a subset of the rows of a data frame that satisfy a set of stated conditions.

The format of the filter function is:

```
filter(data frame name, selection condition(s))
```

Example 1

Refer to the hd data set. Suppose we wish to focus not on all 32 patients but only the 20 White patients. Create a new data frame called `hd_w` that contains the data for only the White patients. Obtain the dimensions of this new data frame and print out the first six rows.

```r
hd_w <- filter(hd, Race == "White")
dim(hd_w)
[1] 20  6
head(hd_w)
   SBP Age  BMI Height Smoke Race
1  122  41 24.7     67    0  White
2  148  52 27.4     70    0  White
3  146  54 23.3     71    1  White
4  162  60 26.9     79    1  White
5  160  48 26.6     67    1  White
6  144  44 20.0     75    0  White
```

Note the use of the double “=“ in the filter expression. Take this to mean ‘exactly equal to’.
Example 2

Suppose now we want to focus on the non-White patients. Create a new data set that contains just the non-White patients. Obtain the dimensions of this new data frame.

There are a couple of ways of specifying non-White patients;

(a)
```r
hd_nonW <- filter(hd, Race != "White")
dim(hd_nonW)
```

[1] 12  6

Read the notation != as ‘not equal to’.

(b)
```r
hd_nonw <- filter(hd, Race == "Black" | Race == "Hispanic")
dim(hd_nonw)
```

Read the notation | as ‘or’.

Example 3

Refer to the exitpoll data. We want to focus on the data for senior voters. Create a new data frame (called seniors) which contains just the results for those voters aged 65 or older. How many rows are there in this new data frame?

```r
seniors <- filter(exitpoll, Age >= 65)
nrow(seniors)
```

[1] 79

In each of the three examples we created a new data frame by selecting a subset of the original data frame.
14. Data organization

**arrange()**

This function is used to arrange a data frame based on the order of a specified variable (or variables).

The format of the arrange function is:

```
arrange(data frame name, name of variable(s))
```

**Example 1**

Refer to the hd data set. Create a new data frame called hd_SBP in which the observations are listed in order of SBP. Print the first six rows of hd_SBP. What is the third smallest SBP?

```
hd_SBP <- arrange(hd, SBP)
head(hd_SBP)
```

<table>
<thead>
<tr>
<th>SBP</th>
<th>Age</th>
<th>BMI</th>
<th>Height</th>
<th>Smoke</th>
<th>Race</th>
</tr>
</thead>
<tbody>
<tr>
<td>120</td>
<td>43</td>
<td>22.3</td>
<td>68</td>
<td>0</td>
<td>Black</td>
</tr>
<tr>
<td>122</td>
<td>41</td>
<td>24.7</td>
<td>67</td>
<td>0</td>
<td>White</td>
</tr>
<tr>
<td>126</td>
<td>43</td>
<td>23.1</td>
<td>74</td>
<td>0</td>
<td>White</td>
</tr>
<tr>
<td>129</td>
<td>47</td>
<td>22.3</td>
<td>76</td>
<td>1</td>
<td>Black</td>
</tr>
<tr>
<td>130</td>
<td>49</td>
<td>23.9</td>
<td>69</td>
<td>0</td>
<td>Black</td>
</tr>
<tr>
<td>132</td>
<td>50</td>
<td>24.5</td>
<td>70</td>
<td>0</td>
<td>White</td>
</tr>
</tbody>
</table>

The third smallest SBP is 126 mm.

Notice that the arrange function orders the rows of hd_SBP in ascending order of SBP. If you prefer to arrange the data in descending order of SBP, you need only add a – (negative) sign in front of SBP.
Example 2

Refer to the hd data set. Create a new data frame called hd_SBP in which the observations are listed in descending order of SBP. Print the first six rows of hd_SBP. What is the third largest SBP?

```r
hd_SBP <- arrange(hd, -SBP)
head(hd_SBP)
```

<table>
<thead>
<tr>
<th>SBP</th>
<th>Age</th>
<th>BMI</th>
<th>Height</th>
<th>Smoke</th>
<th>Race</th>
</tr>
</thead>
<tbody>
<tr>
<td>180</td>
<td>64</td>
<td>32.1</td>
<td>74</td>
<td>1</td>
<td>Hispanic</td>
</tr>
<tr>
<td>170</td>
<td>63</td>
<td>29.4</td>
<td>81</td>
<td>1</td>
<td>Black</td>
</tr>
<tr>
<td>166</td>
<td>59</td>
<td>28.0</td>
<td>70</td>
<td>1</td>
<td>White</td>
</tr>
<tr>
<td>164</td>
<td>65</td>
<td>28.7</td>
<td>66</td>
<td>1</td>
<td>Hispanic</td>
</tr>
<tr>
<td>162</td>
<td>60</td>
<td>26.9</td>
<td>79</td>
<td>1</td>
<td>White</td>
</tr>
<tr>
<td>161</td>
<td>63</td>
<td>27.7</td>
<td>70</td>
<td>0</td>
<td>White</td>
</tr>
</tbody>
</table>

The third largest SBP is 166 mm.

Example 3

Refer to the hd data set. Create a new data frame called hd_A in which the observations are listed in ascending order of SBP and then by Age within SBP. Print the first 10 rows of hd_A.

```r
hd_A <- arrange(hd, SBP, Age)
head(hd_A, 10)
```

<table>
<thead>
<tr>
<th>SBP</th>
<th>Age</th>
<th>BMI</th>
<th>Height</th>
<th>Smoke</th>
<th>Race</th>
</tr>
</thead>
<tbody>
<tr>
<td>120</td>
<td>43</td>
<td>22.3</td>
<td>68</td>
<td>0</td>
<td>Black</td>
</tr>
<tr>
<td>122</td>
<td>41</td>
<td>24.7</td>
<td>67</td>
<td>0</td>
<td>White</td>
</tr>
<tr>
<td>126</td>
<td>43</td>
<td>23.1</td>
<td>74</td>
<td>0</td>
<td>White</td>
</tr>
<tr>
<td>129</td>
<td>47</td>
<td>22.3</td>
<td>76</td>
<td>1</td>
<td>Black</td>
</tr>
<tr>
<td>130</td>
<td>49</td>
<td>23.9</td>
<td>69</td>
<td>0</td>
<td>Black</td>
</tr>
<tr>
<td>132</td>
<td>48</td>
<td>23.5</td>
<td>74</td>
<td>1</td>
<td>White</td>
</tr>
<tr>
<td>132</td>
<td>50</td>
<td>24.5</td>
<td>70</td>
<td>0</td>
<td>White</td>
</tr>
<tr>
<td>134</td>
<td>50</td>
<td>23.3</td>
<td>67</td>
<td>1</td>
<td>Hispanic</td>
</tr>
<tr>
<td>135</td>
<td>45</td>
<td>22.8</td>
<td>70</td>
<td>0</td>
<td>Black</td>
</tr>
<tr>
<td>135</td>
<td>57</td>
<td>24.2</td>
<td>75</td>
<td>0</td>
<td>Black</td>
</tr>
</tbody>
</table>

Notice that the two patients with SBP = 132 are ordered by Age.
It is not necessary to provide a new name for the re-arranged data frame but if you use simply

```r
arrange(hd, SBP, Age)
```

R will replace the original data frame hd with the new arrangement and you probably don’t want this.
14. Data organization

sort()

The sort function is used to sort a variable (not a data frame) in order.

If the variable to be sorted is a column in a data frame, the format for the function is:

\[
\text{sort(data frame name$variable name)}
\]

If the variable is not part of a data frame, the format is simpler

\[
\text{sort(variable name)}
\]

(See Examples 1-3 below)

**Example 1**

Here are final exam scores for the seven students in a seminar

\[
scores \leftarrow c(86, 91, 81, 93, 88, 84, 95)
\]

Sort these scores in ascending order.

\[
\text{sort(scores)}
\]

\[
[1] \, 81 \, 84 \, 86 \, 88 \, 91 \, 93 \, 95
\]

If you would like to save these ordered values, simply give them a name.

**Example 2**

Sort and save the values in scores as a new variable, sscores.

\[
\text{sscores} \leftarrow \text{sort(scores)}
\]

\[
\text{sscores}
\]

\[
[1] \, 81 \, 84 \, 86 \, 88 \, 91 \, 93 \, 95
\]
You can also sort in descending order with the `decreasing = TRUE` argument.

**Example 3**

Create a new variable, `sc` which contains the scores ordered from largest to smallest.

```r
sc <- sort(scores, decreasing = TRUE)
sc
[1] 95 93 91 88 86 84 81
```

**Example 4**

Refer to the `hd` data set. Create a new variable called `sBMI` which contains the BMI’s for the 32 patients listed is increasing order. Print the first 10 values in `sBMI`. What is the fifth smallest BMI?

```r
sBMI <- sort(hd$BMI)
head(sBMI, 10)
[1] 20.0 22.3 22.3 22.8 23.1 23.3 23.3 23.5 23.5 23.9
```

The fifth smallest BMI is 23.1.
15. Other useful commands

`qqnorm()`/`qqline()`

The `qqnorm` function is a base R command that creates a qqplot (sometimes known as a Normal Probability plot) for a quantitative variable. The closer the points in the plot are to a straight line the closer the data is to normal. The `qqline` function simply adds the best fitting line to the plot in order to make it easier to judge how close the points are to linear. The `qqline` is not used outside of this context.

The format for the `qqnorm` function and for the `qqline` are similar.

If the variable is a column in a data frame, the format is:

`qqnorm(data frame name$variable name)`  
`qqline(data frame name$variable name)`

If the variable is not part of a data frame, the format is simpler

`qqnorm (variable name)`  
`qqline (variable name)`

(See Example 5 below)

**Example 1**

For the `hd` data, obtain a qqplot for the 32 SBP’s.

`qqnorm(hd$SBP)`
Example 2

To the plot in Example 1, add the best-fitting line.

```r
qqnorm(hd$SBP)
qqline(hd$SBP)
```

You can select the color of the dots in the qqplot and the color of the qqline.
Example 3

Reproduce the plot in Example 2 with red dots and a blue line.

```r
qqnorm(hd$SBP, col = "red")
qqline(hd$SBP, col = "blue")
```

Example 4

For the exitpoll data, obtain a normal probability plot (with the fitted line) for the wait times of the 208 voters. Make the dots red and the line blue. Add the title “NPP of the 208 Wait Times Showing the Non-Normality of the Times”.

```r
qqnorm(exitpoll$WaitTime, col = "red",
       main = "NPP of the 208 Wait Times Showing the Non-Normality of the Times")
qqline(exitpoll$WaitTime, col = "blue")
```
Example 5

Here are final exam scores for the seven students in a seminar.

scores <- c(86, 91, 81, 93, 88, 84, 95)

Obtain a normal probability plot (with the fitted line) for these scores.

qqnorm(scores)
qqline(scores)
15. Other commands

plotDist()

plotDist is a mosaic function that creates plots of theoretical distributions like the normal and the t. We will focus on drawing normal distributions.

The basic format for using plotDist to draw a normal distribution is:

```
plotDist("norm", mean = \mu, sd = \sigma)
```

Example 1

Obtain a plot of the N(10, 2) distribution.

```
plotDist("norm", mean=10, sd=2)
```

![Normal distribution plot](image)

It is straightforward to specify the color of the plot.

Example 2
Reproduce the normal plot in Example 1 in red.

\[
\text{plotDist("norm", mean=10, sd=2, col="red")}
\]

You can use the \textit{type = "h"} argument to ‘fill’ the plot.

\textbf{Example 3}

Obtain a plot of the N(10, 2) distribution with red fill.

\[
\text{plotDist("norm", mean=10, sd=2, col="red", type = "h")}
\]
By using the `under = TRUE` argument you can superimpose one distribution on another.

**Example 4**

Obtain a plot of the N(10, 2) distribution and the N(10, 1) distribution.

```r
plotDist("norm", mean = 10, sd = 2, col = "red")
plotDist("norm", mean = 10, sd = 1, col = "blue", under = TRUE)
```

![Graph showing superimposed distributions](image)

**Example 5**

In this example, we produce a N(100, 15) distribution with the area to the right of \( x = 100 \) given a different color to the area to the left of \( x = 100 \).

```r
plotDist("norm", mean = 100, sd = 15, groups = x > 110, type = "h")
```
The colors in Example 5 are default. You can easily specify the two colors you wish to use.

**Example 6**

Reproduce the plot in Example 5 but with the colors blue and red.

```r
plotDist("norm", mean = 100, sd = 15, groups = x > 110,
        type = "h",
        col = c("red", "blue"))
```
Example 7

Referring back to Example 6, I wanted to emphasize the area to the right of 100 by making the area to the left white. This is how it came out.

```r
plotDist("norm", mean = 100, sd = 15, groups = x > 110, type = "h", col = c("white", "blue") )
```

Not quite what I had in mind!

Example 8

I created what I had in mind in Example 7 by (i) first creating a N(100, 15) distribution and the using the `under = TRUE` argument to superimpose the same distribution but with white and blue fill as before.

```r
plotDist("norm", mean = 100, sd = 15) plotDist("norm", mean = 100, sd = 15, groups = x > 110, type = "h", col = c("white", "blue"), under = T )
```
Example 9

Finally, here is a plot of the standard normal (Z) distribution with a t4 distribution superimposed. The `main` argument is used to add a title to the plot.

```r
plotDist("norm", col="blue", main = "The N(0,1) distribution (in blue) and the t4 distribution (in red)")
plotDist("t", df = 4, col="red", under=TRUE)
```