```
# quantile.


# Instead of focusing on "50% of the cases," we can ask about 10% of the cases.
# That would be the number in the data that has 10%  of the 11 cases below it,
# i.e., -10, because there is 1 case below -10, as confirmed by

quantile(x, probs = 0.1)

10%
-10
```

2.4 Distribution Quantile

In this section, we will review the notion of distribution quantile. Recall Table 1 in the textbook, which gives us the area under the standard normal distribution to the left of some number. For example, $z = 1.285$ has about 90% of the area to its left. That means that about 90% of the values of z (ranging from $-\infty$ to $+\infty$) are less than 1.285. In other words, the 90th percentile (or 0.9th quantile) of the standard normal distribution is about 1.285. Table 1 is more precisely encoded into the R function **qnorm()**:

```
# Finding the 0.9 quantile of a standard normal distribution
qnorm(0.9, mean = 0, sd = 1, lower.tail = TRUE)

[1] 1.282

# Similarly, we can find the 0.1th, 0.2th, 0.3th, ..., 0.9th quantile:
sequence <- seq(0.1, 0.9, by = 0.1)
qnorm(sequence, mean = 0, sd = 1, lower.tail = TRUE )

[1] -1.2816 -0.8416 -0.5244 -0.2533  0.0000  0.2533  0.5244  0.8416  1.2816

# This way, we can compute any quantile of the standard normal distribution.
# In fact, we can find any quantile of any distribution.
```

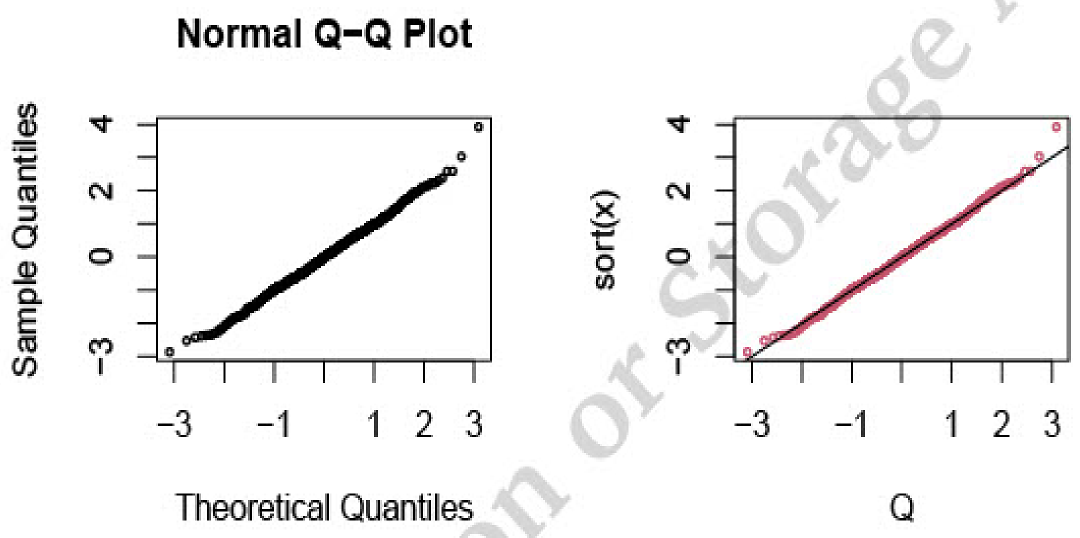
2.5 Q-Q Plots

A q-q plot is a plot of sample quantiles versus distribution quantiles for some specified distribution. If the result is a relatively straight "line," then there is some evidence that the data have come from that distribution. More intuitively, there is evidence that the histogram of the data is consistent with the specified distribution. Most often when people talk about a q-q plot, they are assuming that the distribution is the standard normal distribution. So one plots sample quantiles (along the y-axis) versus quantiles of the of the standard normal (along the x-axis). The R corresponding function is **qqnorm()**.

Example

Now we take a sample from a standard normal distribution and use qqnorm() to make the q-q plot for the sample. Then, we will make the q-q plot "by hand:"

```r
n <- 500  # Sample size = 500.
x <- rnorm(n, 0, 1)  # Sample from a normal distribution with mu = 0, sigma = 1.

qqnorm(x, cex = 0.5)  # The q-q plot, according to qqnorm().

# Doing it by hand:
X <- seq(.5 / n, 1 - .5 / n, length = n)  # Make a sequence of n values between
                                          # (1-.5)/n and (n-.5)/n , and find
Q <- qnorm(X, mean = 0, sd = 1)  # their quantiles under standard normal.
plot(Q, sort(x), col = 2, cex = 0.5)  # plot the data (sorted) vs. the
                                      # quantiles.
abline(0, 1)  # Add a line with slope 1 and intercept 0.
```

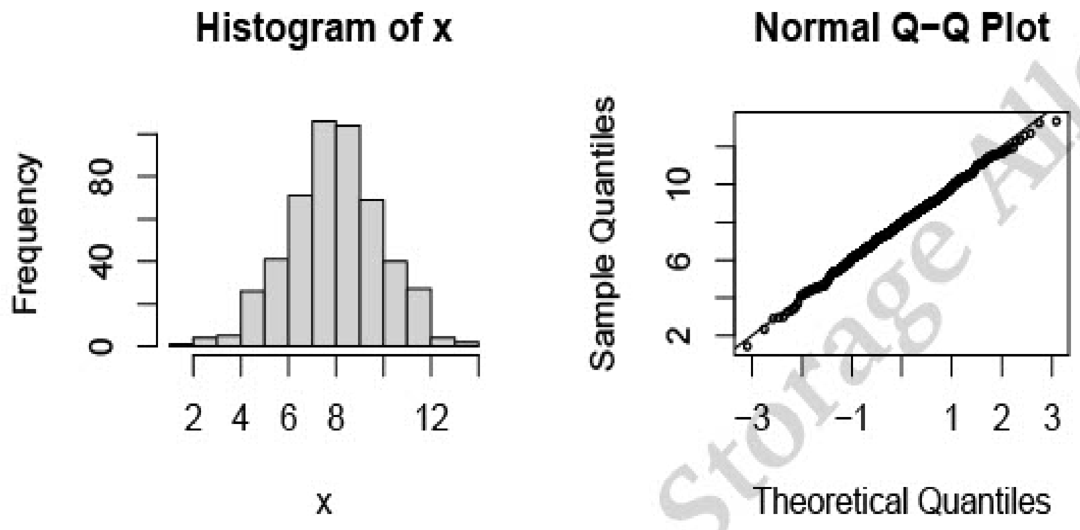


Note that the two q-q plots are the same. Make sure you understand what's done here: When we have 500 observations, each one is associated with a percentile; e.g., the smallest observation has nothing less than it, and so its percentile (rank) is 0. Meanwhile, the largest obs has everything less than it, and so, its percentile (rank) is 100%. In terms of quantiles (instead of percentiles), these two limits are 0 and 1; and that's why we have X go from 0 to 1. In fact, the numbers go from a number close to 0 to a number close to 1, namely from $.5/n$ to $1 - .5/n$; this is mostly a matter of convention for handling the lowest and the largest element in the sample. Then, Q = qnorm(X,0,1) returns the quantiles of the standard normal dist.

It's not obvious but (e.g., see the book) the slope of the "line" is the σ parameter of the normal distribution from which the data have come, and the y-intercept of the line is equal to the μ parameter of the distribution. In this case, we can see that the slope is around 1, and the y-intercept is around 0. We have confirmed this by drawing a line with slope = 1, y-intercept = 0. Note that the line we have just drawn is NOT the "best fit" line to the q-q plot. When we talk about the slope or the intercept of the q-q plot, think of them as the "visual slope" and the "visual y-intercept" instead.

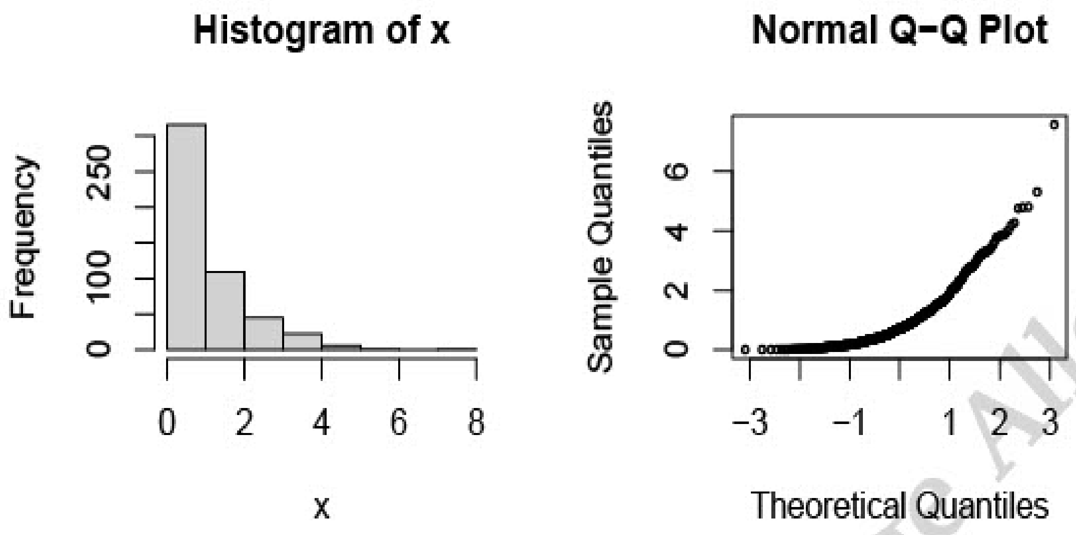
Now, let's consider data coming from a normal distribution with $\mu = 8$, $\sigma = 2$ (i.e., not standard). If we continue using the quantiles of the standard normal along the x-axis, it turns out the slope will then be 2, and the intercept will be 8. In other words, the interpretation of the q-q plot is still the same - the slope is going to be close to the σ of the distribution from which the data were drawn, and the intercept will be approximately the μ of the distribution.

```
n <- 500  # Sample of size 500.
x <- rnorm(n, 8, 2)  # Generate data from normal(mu=8,sigma=2).
hist(x)
# qqnorm() checks the data against a normal distribution.
qqnorm(x, cex = 0.5)  # Note the slope and intercept are about 8 and 2.
abline(8, 2)  # Add a line with slope = 2, intercept = 8.
```



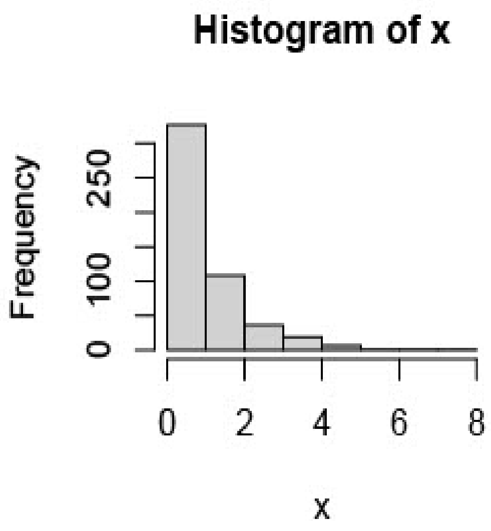
Now, let's get a sense of what a standard normal q-q plots look like for data from non-normal distributions. Recall that if the data come from a normal distribution, their q-q plot should look like a straight line, at least in the bulk of the plot; the tails usually deviate from a straight line, because there are usually few cases there anyway. A normal q-q plot is a visual method for checking whether data are normally distributed. Also, if linear, then the intercept and slope of the line can be used as estimates of the μ and σ of the normal distribution, respectively.

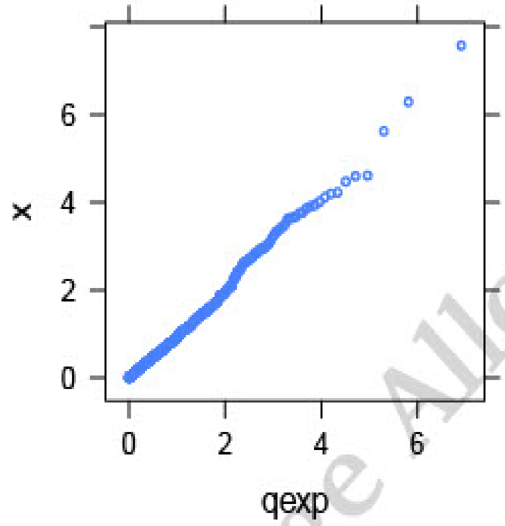
```
x <- rexp(n, 1)  # Sample of size 500 from an exponential dist with lambda = 1.
hist(x)
qqnorm(x, cex = 0.5)  # Note that it is not straight at all.
```

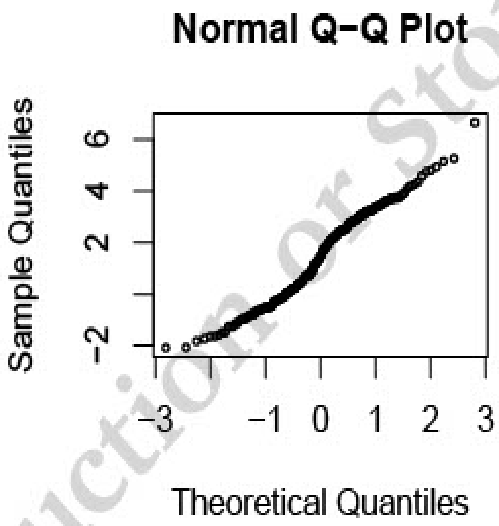


As we can see, a non-normal sample will not produce a linear pattern because qqnorm() checks the data against the standard normal distribution. But how do we identify if some data come from some other distribution, say, from the exponential distribution? The solution is to use analog of qqnorm for the exponential distribution. For example, we can plot quantiles of data versus quantiles of the exponential distribution. In R, the corresponding function is **qqmath()**, which allows for a large number of theoretical distributions.

```
library(lattice)  # Load the library that contains qqmath().
x <- rexp(500, 1)  #Sample of size 500 from an exponential dist with lambda = 1.
hist(x)
qqmath(x, dist = qexp, cex = 0.5)

# The q-q plot is a straight "line" even though the data are from an
# exponential distribution.

# Finally, recall that in section 1, we saw a bimodal histogram.
# Check out its q-q plot:
dat <- read.table('hist_dat.txt', header = F)
qqnorm(dat[, 1], cex = 0.5)
# You can see that there are two linear segments, parallel (i.e., same sigma),
# but different intercepts (i.e., different mu).
```







2.6 Jargon

As explained in class, in statistics, we use distributions to represent populations. In fact, we can think of distributions and populations as one and the same thing. Specifically, when we talk about "a sample from a population," what we really mean is "a sample from a distribution." That jargon can be confusing for some of the distributions (especially, Binomial), but it will help if every time you hear "a sample from a distribution," you translate that into what the actual data in the sample would look like. Consider the following examples, and their translation:

"A sample of size n from a Bernoulli distribution with parameter π." Translation: n numbers, each either 0 or 1, and the proportion of 1's is around π.

"A sample of size n from a Normal distribution with parameters μ and σ." Translation: n numbers, each between $\pm\infty$, with most around μ, and a typical deviation around σ.

"A sample of size m from a Binomial distribution with parameters n and π." Translation: m numbers, each an integer between 0 and n. (Later, you'll see how π affects the sample.)

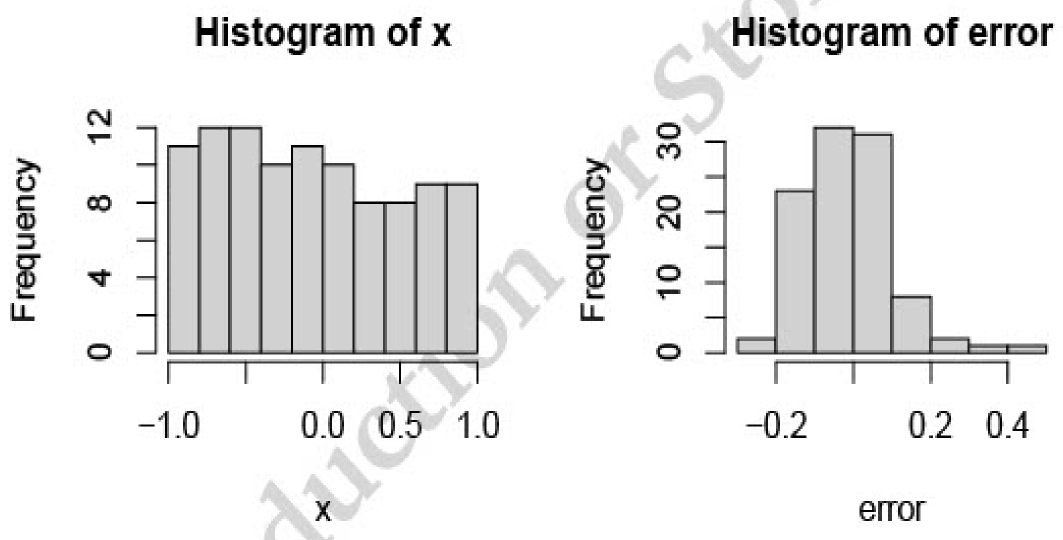
The last one is particularly confusing because of the names R uses to refer to the parameters of the Binomial distribution. Specifically, what we call the n and π parameters of the Binomial distribution are called "size" and "prob." And, to make matters worse, take a look at the help pages for **rbinom**; you'll see "rbinom(n, size, prob)," and so the n that appears in there is NOT what we call the n parameter of Binomial. I did say that it's confusing!

3 Regression

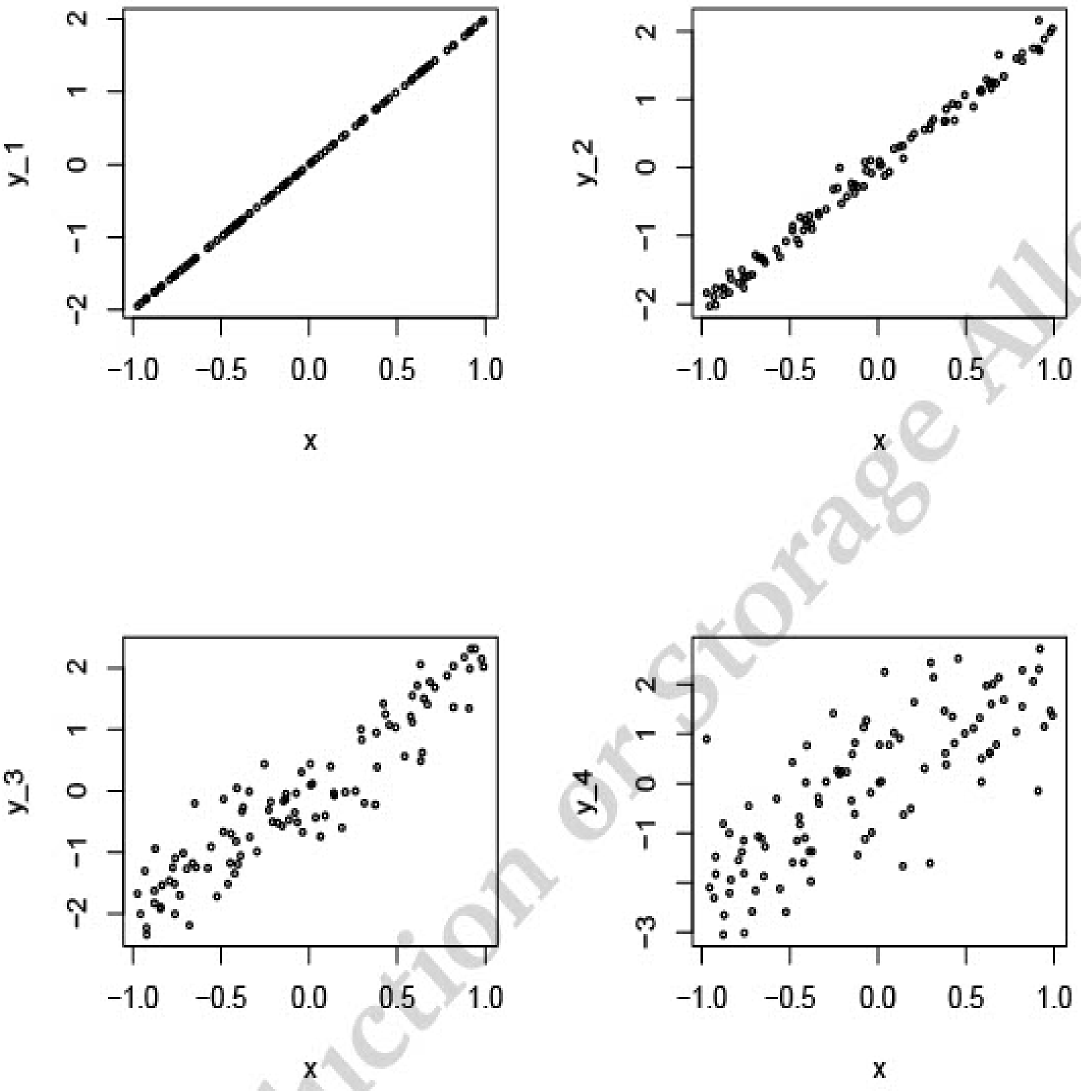
3.1 Scatter Plots

The best way of visualizing the relationship between two continuous random variables is through a scatterplot. (Just in passing, the analog for two categorical variables is the Contingency Table, also called the Confusion Matrix.) It can convey a great deal of information, including whether or not the relationship linear, and the extent of the strength of the relationship. Here, strength refers to the skinniness of the scatterplot. Let's illustrate through an example: Pick 100 random x values, and corresponding y values that have some linear association with x and change the amount of linear association by adding different amounts of "error" to y.

```
par(mfrow = c(1, 2))
x <- runif(100, -1, 1)  # Take 100 points from a uniform distribution between
                        # -1 and 1.
hist(x)  # The shape and looks uniform.
error <- rnorm(100, 0, 0.1)  # Generate a normal variable (the error), with mu=0,
                             # sigma=0.1
hist(error)  # The shape looks normal.
```



```
y_1 <- 2 * x  # Perfect linear relation between x and y.
y_2 <- 2 * x + error  # With some error added to y.
y_3 <- 2 * x + rnorm(100, 0, 0.5)  # With more error added to y.
y_4 <- 2 * x + rnorm(100, 0, 1.0)

par(mfrow = c(2, 2))
plot(x, y_1, cex = 0.5)
plot(x, y_2, cex = 0.5)
plot(x, y_3, cex = 0.5)
plot(x, y_4, cex = 0.5)  # Note that too much noise makes it hard to see
                         # the linear relationship between x and y.
```



3.2 Correlation

To quantify the strength of the association between two continuous variables, Pearson's correlation coefficient (i.e., correlation), can be computed. It measures the 'amount of scatter' (i.e., skinniness) in a linear sense (but NOT about "the fit").

```
cor(x, y_1)  # Check against the scatterplots, to get a feeling for r (correlation).

[1] 1

cor(x, y_2)

[1] 0.995

cor(x, y_3)
```

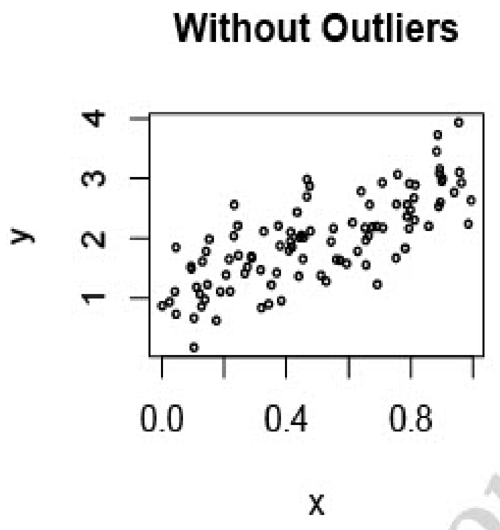
```
[1] 0.9321

cor(x, y_4)

[1] 0.7744

cor(y_4, x)  # r is symmetric.

[1] 0.7744

cor(y_4, x + 10)  # r is invariant under shifts.

[1] 0.7744

cor(x, 10 * y_4)  # r is invariant under scaling.

[1] 0.7744
```

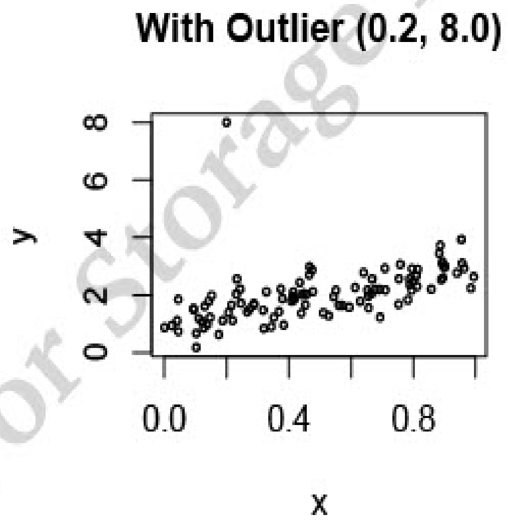
3.2.1 Defects of Correlation

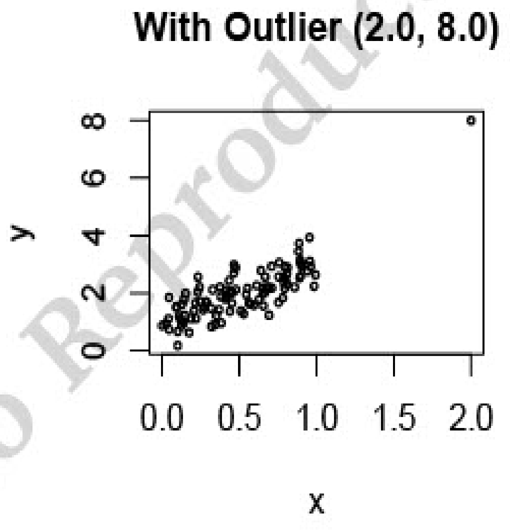
Pearson's correlation coefficient, r, can become misleading in several situations.

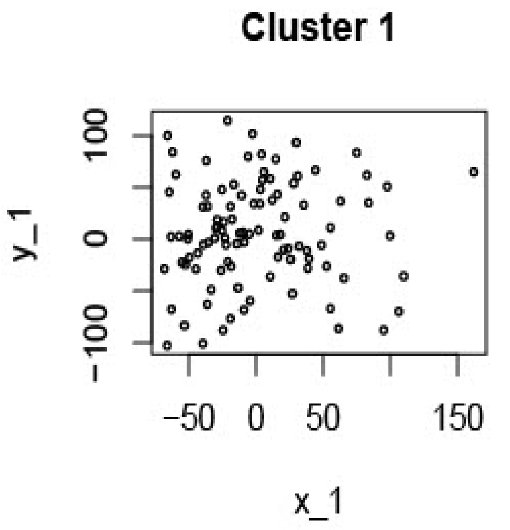
```
set.seed(123)  # Set a seed to get reproducable results.
x <- runif(100, 0, 1)
error <- rnorm(100, 0, 0.5)
y <- 1 + 2 * x + error
x_1 <- rnorm(100, 0, 50)
y_1 <- rnorm(100, 0, 50)
x_2 <- 1000 + rnorm(100, 0, 50)
y_2 <- 1000 + rnorm(100, 0, 50)
plot(x, y, main = 'Without Outliers', cex = 0.5)
cor(x, y)

[1] 0.7562

# Effect of outliers:
x[101] <- 0.2  # Adding one outlier can artificially reduce r.
y[101] <- 8.0
plot(x, y, main = 'With Outlier (0.2, 8.0)', cex = 0.5)
cor(x, y)

[1] 0.516

x[101] <- 2.0  # A different outlier can artificially increase r.
y[101] <- 8.0
plot(x, y, main = 'With Outlier (2.0, 8.0)', cex = 0.5)
cor(x, y)

[1] 0.8129

# Clusters can also make r meaningless.
plot(x_1, y_1, main = 'Cluster 1', cex = 0.5)
cor(x_1, y_1)  # No correlation between x and y in cluster 1
```

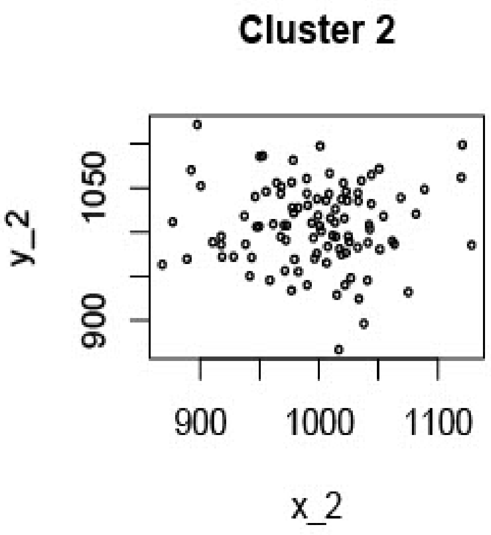
```
[1] 0.05597

plot(x_2, y_2, main = 'Cluster 2', cex = 0.5)
cor(x_2, y_2)  # No correlation between x and y in cluster 2

[1] -0.006664

x <- c(x_1, x_2)  # Combine/concatenate the 2 clusters.
y <- c(y_1, y_2)
plot(x, y, main = 'Combined Clusters', cex = 0.5)
cor(x, y)  # r incorrectly sees a correlation between x and y.

[1] 0.991
```

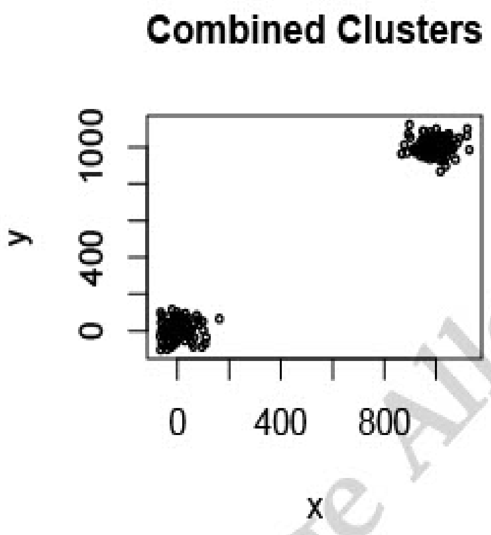










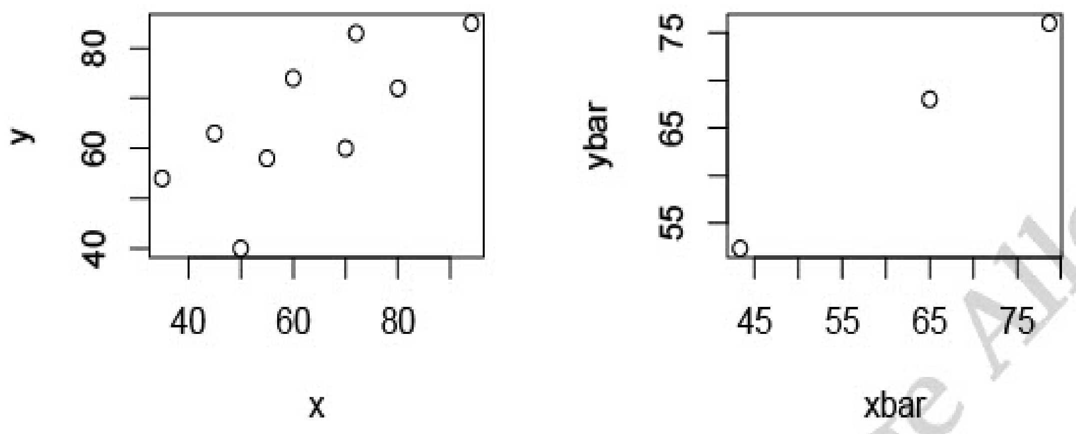


The moral is this: Use r to measure linear correlation, but always examine the data (e.g. with a scatterplot) to make sure things are okay.

Example: Ecological Correlation

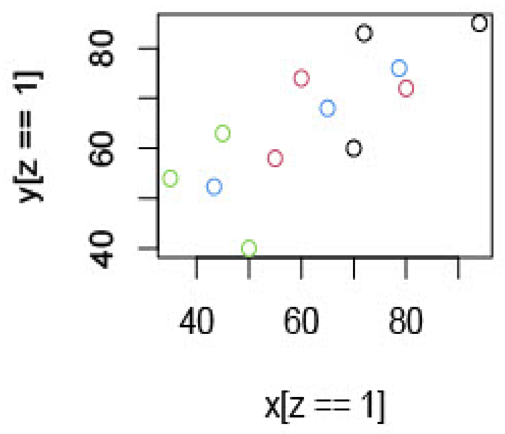
The following example illustrates another way in which the value of r can be "artificially" increased, i.e., by averaging over things before computing r. For similar reasons, regression results (disscussed in later sections) can be misleading as well.

```
dat <- read.table('3_17_dat.txt', header = TRUE)
x <- dat[, 1]
y <- dat[, 2]
z <- dat[, 3]

plot(x, y)  # Making a scatter plot.
cor(x, y)  # Moderate correlation of 0.733 between the 9 pairs.

[1] 0.7329

xbar <- numeric(3)  # Allocating space for storing the time-averaged values of x.
ybar <- numeric(3)  # and of y.

xbar[1] <- mean(x[z == 1])  # This averages x values only when time = 1.
ybar[1] <- mean(y[z == 1])

xbar[2] <-  mean(x[z == 2])  # USE UP-ARROW.
ybar[2] <-  mean(y[z == 2])

xbar[3] <-  mean(x[z == 3])
ybar[3] <-  mean(y[z == 3])

plot(xbar,ybar)  # Scatterplot of the 3 averaged pairs,
cor(xbar,ybar)  # and their extreme correlation of 0.998 .

[1] 0.9985
```



You can see clearly how it is that averaging tends to increase r, by reducing the number of points and their scatter about a line. Looking at the last scatterplot of the original data, but with the three times colored differently, you can see why this magnification of r is happening: Averaging the three pairs for each time, replaces the three points with a single point located in the "middle" of the three. In general, then, averaging tends to reduce the scatter, and hence the resulting r (called the ecological correlation):

```
plot(x[z == 1], y[z == 1], xlim = range(x), ylim = range(y))  # Scatterplot for time 1
points(x[z == 2], y[z == 2], col = 2)  # time 2 (USE UP-ARROW)
points(x[z == 3], y[z == 3], col = 3)  # time 3
points(xbar, ybar, col = 4)  # and the averaged data.

pdf("ecol.pdf")
plot(x[z == 1], y[z == 1], xlim = range(x), ylim = range(y), xlab = "x",
     ylab = "y", pch = 1, cex = 3)
points(x[z == 2], y[z == 2], col = 1, pch = 2, cex = 3)
points(x[z == 3], y[z == 3], col = 1, pch = 3, cex = 3)
dev.off()

pdf
  2
```

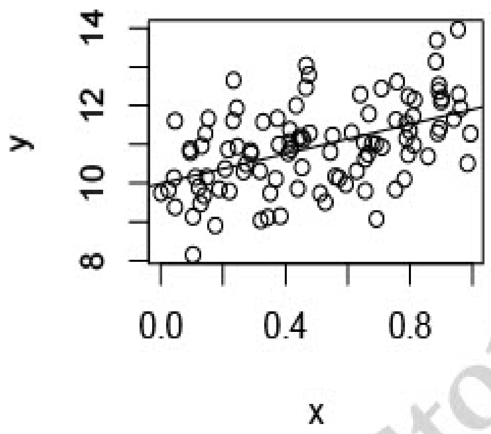


3.3 OLS Regression on Simulated Data

Regression (or a line that fits the scatterplot) can be used for prediction. The function lm(), which stands for linear model, does runs a regression in R. It fits a curve through a scatterplot, or a surface through higher-dimensional data.

```
rm(list = ls(all = TRUE)) # Start from a clean slate.
set.seed(123)  # Ensures reproducable results.
x <- runif(100, 0, 1)  # x is uniform between 0 and 1.
error <- rnorm(100, 0, 1)  # Error is normal with mean = 0, sigma = 1.
y <- 10 + 2*x + error  # The real/true line is y = 10 + 2x.
plot(x, y)  # Plot the scatterplot.
cor(x, y)  # Correlation between x and y.

[1] 0.4916

model.1 <- lm(y ~ x)  # Fitting the regression.
model.1  # Note that the estimated coefficients are pretty close to the true ones


Call:
lm(formula = y ~ x)

Coefficients:
(Intercept)            x
       9.99         1.91

abline(model.1)  # Superimposes the fit on the scatterplot.

# To see what else is returned by lm(), use the following command:
names(model.1)

 [1] "coefficients"  "residuals"     "effects"       "rank"
 [5] "fitted.values" "assign"        "qr"            "df.residual"
 [9] "xlevels"       "call"          "terms"         "model"
```

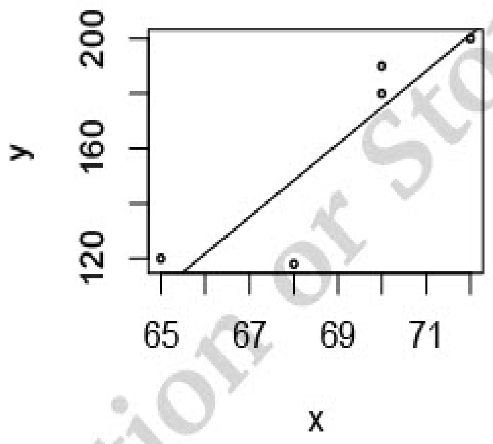
```
# To select one of the items returned in lm(), use the dollar sign:
model.1$coefficients

(Intercept)           x
      9.991       1.910
```



3.4 OLS Regression on "Real" Data

```
x <- c(72, 70, 65, 68, 70)  # Enter data into R.
y <- c(200, 180, 120, 118, 190)  # See 1.1 for alternative ways to enter data.
plot(x, y, cex = 0.5)
cor(x, y)

[1] 0.8892

model.1 <- lm(y ~ x)
abline(model.1)  # Draws the fit
model.1  # Returns the estimated intercept and slope.


Call:
lm(formula = y ~ x)

Coefficients:
(Intercept)            x
     -755.1         13.3

summary(model.1)


Call:
lm(formula = y ~ x)
```

```
Residuals:
     1      2      3      4      5
 -1.46   5.11  11.54 -30.31  15.11

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  -755.11     272.53   -2.77    0.070 .
x              13.29       3.95    3.37    0.044 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 20.9 on 3 degrees of freedom
Multiple R-squared:  0.791,Adjusted R-squared:  0.721
F-statistic: 11.3 on 1 and 3 DF,  p-value: 0.0436
```



Example: Regression on Hail Data

In practice, two quantities called "divergence" and "rotate" are measured by Doppler radar, while hail size is measured directly, i.e., on the ground. But if we can relate hail size to divergence and rotate, then we can predict hail size from Doppler radar. In regression lingo, size is the response (or dependent) variable, and the others are predictors (or independent variables, or covariates).

```
dat <- read.table("hail_dat.txt", header=T)

plot(dat)
cor(dat)  # This shows the correlations between ALL the vars in the hail data

                    Divergence Rotational_velocity Hail_size
Divergence              1.0000              0.5496    0.5214
Rotational_velocity     0.5496              1.0000    0.5386
Hail_size               0.5214              0.5386    1.0000

size <- dat[, 3]  # Name the 3 columns in dat. Size is in 100th-of-an-inch.
```