Example 2

qqnorm(dat[, 1], cex = 0.5)

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 $y$.

```r
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”).

```r
> cor(x, y_1)  # Check against the scatterplots, to get a feeling for r (correlation).
[1] 1
> cor(x, y_2)
[1] 0.9972
> cor(x, y_3)
```

[1] 0.9298

cor(x, y_4)
[1] 0.769

cor(y_4, x)  # r is symmetric.
[1] 0.769

cor(y_4, x + 10)  # r is invariant under shifts.
[1] 0.769

cor(x, 10 * y_4)  # r is invariant under scaling.
[1] 0.769

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 reproducible 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
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.

### 3.2.2 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 (discussed in later sections) can be misleading as well.

```r
dat <- read.table("http://www.stat.washington.edu/marzban/390/3_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.
```
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):

```r
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()
```

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.

```r
rm(list = ls(all = TRUE)) # Start from a clean slate.
set.seed(123) # Ensures reproducible 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"
3.4 OLS Regression on Real Data

```r
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)
```

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

(Intercept)      x
     9.991  1.910
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).

```r
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.
```