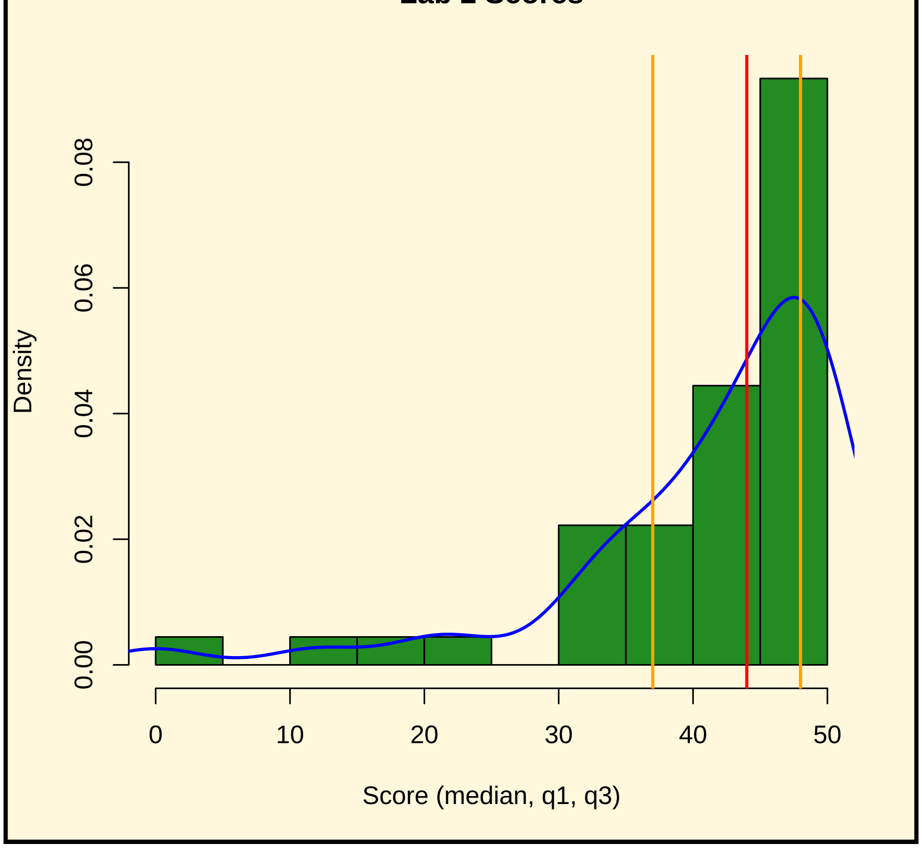
ISTA 116 Lab: Week 7

Colin Dawson

Last Revised October 3, 2011

1 HW2 Scores



2 HW3

- **Go over HW3**

3 Correlation

Two numeric variables for *each observation* (that is, we have numeric *pairs*).

3.1 Scatterplots

Each observation varies along two dimensions. A single point has two "coordinates".

- A *scatterplot* displays each data point in two dimensions, with the value of one variable on the x-axis, and the value of the other on the y-axis.
- In `R`, we can just use the `plot()` function.

The `blood` data set (in the `UsingR` library) contains blood pressure readings for 15 individuals. `Machine` contains readings by an automated machine; `Expert` contains readings by an expert.

```
> library(UsingR)
> data(blood)
> attach(blood)
```

Plot Machine "as a function of" Expert. That is, for each expert reading (on the x-axis) plot the corresponding machine reading on the y-axis. What do you expect to see?

```
> plot(Machine ~ Expert)
```

We can get a sense of the relationship from a scatterplot, but our eyes can sometimes deceive us. It's often a good idea to measure the relationship quantitatively.

- The *Pearson Correlation* is a measure of the strength of a *linear* relationship between two variables, ranging from -1 (perfect negative linear relationship) to $+1$ (perfect positive linear relationship)
- Computed in `R` using `cor()`
- It is a *symmetric* measure of the relationship.

```
> cor(Machine, Expert)
[1] 0.9068599
```

But what happens if we don't expect to find a linear relationship, but we still want to measure correlation?

Two possibilities:

- Transform one or both variables (e.g. by log or sqrt) to create a linear relationship
- Use a different measure of correlation

Spearman's rho is a measure of the degree to which two variables tend to move in the same or different directions.

- It uses only *rank* information, so the relationship need not be linear.
- As a result, any transformation that doesn't change the ranks of the data won't change Spearman's rho
- In the `cor()` function, set `method=''spearman''` (lower-case)

```
> cor(Machine, Expert, method = "spearman")
```

```
[1] 0.8878956
```

```
> data(kid.weights)
> with(kid.weights, cor(weight, height, method = "pearson"))
```

```
[1] 0.8237564
```

```
> with(kid.weights, cor(weight, height, method = "spearman"))
```

```
[1] 0.8822136
```

Exercise: Make scatterplots and calculate the Pearson and Spearman correlation for some of the other data sets we've looked at. See what happens to the correlation values if you transform one or both variables.

4 Linear Regression

From Section 3.4 in Verzani: In this section, we introduce the simple linear regression model for describing paired data sets that are related in a linear manner. When we say that variables x and y have a linear relationship in a mathematical sense we mean that $y = mx + b$, where m is the slope of the line and b the intercept. We call x the *independent* variable and y the *dependent* one. In statistics, we don't assume these variables have an exact linear relationship: rather, the possibility for noise or error is taken into account.

To fit the linear regression ("least-squares") model to data, we use the `lm()` function. With data loaded in, you only need to specify the linear model desired. Many other models are possible, today we'll only look at:

```
> lm(y ~ x)
```

The model is specified by the formula `y ~ x` which implies the linear relationship between y (dependent/response variable) and x (independent/predictor variable).

The `lm()` function creates a linear model object from which a wealth of information can be extracted.

Example: Consider the `cars` dataset. The data give the speed (`speed`) of cars and the distances (`dist`) taken to come to a complete stop. Here, we will fit a linear regression model using `speed` as the independent variable and `dist` as the dependent variable (these variables should be plotted first to check for evidence of a linear relation).

```
> data(cars)
> attach(cars)
> plot(speed, dist)
```

To compute the least-squares:

```
> fit <- lm(dist ~ speed)
```

The object `fit` is a linear model object. To see what it contains, type:

```
> attributes(fit)

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

$class
[1] "lm"
```

To get the least squares estimates of the slope and intercept, type:

```
> fit

Call:
lm(formula = dist ~ speed)

Coefficients:
(Intercept)        speed
    -17.579        3.932
```

So, the fitted regression model has an intercept of -17.579 and a slope of 3.932.

We can add the fitted regression line to a scatterplot:

```
> plot(speed, dist)
> abline(fit)
```

5 Questions?