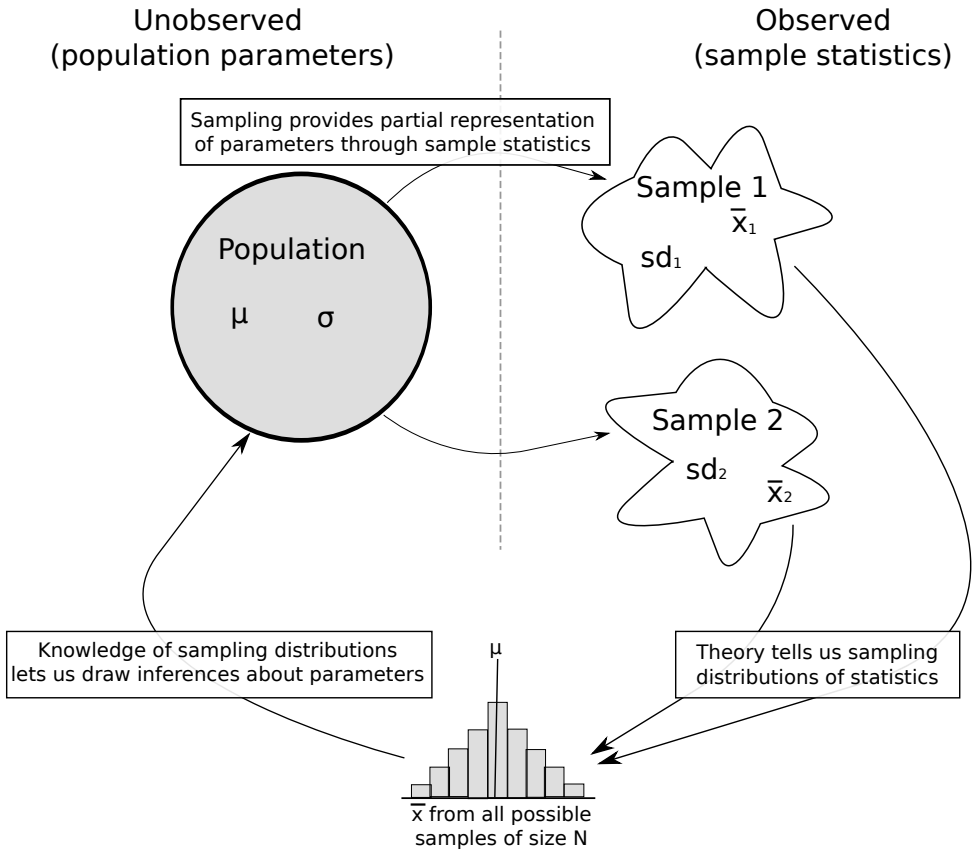
Introduction to Statistical Analysis Using R

Kezia Manlove

April 1, 2014

1 Statistical background



Planning and describing a statistical analysis

In my mind, a statistical analysis consists of three major components:

- **The inferential philosophy employed.** This could be Bayesian, frequentist, information-theoretic, or something else. The basic distinction is whether you think of parameters as fixed constants (as frequentists do), or coming from their own distributions (i.e., Bayesian), or if you instead focus on weight-of-evidence associated with blocks of competing hypotheses (information-theoretic).
- **The estimation procecure used.** This could be Monte Carlo Markov chain (MCMC), maximum likelihood, resampling-based methodologies like bootstrap, jackknife, or permutation-based methods, etc.

- **The model assumed.** The "model" refers to different entities in different contexts, but at its most basic level, it is the set of assumptions governing your analysis

Methods cannot be replicated exactly unless each of these things is known. I try to deal with each of them separately when describing an analysis.

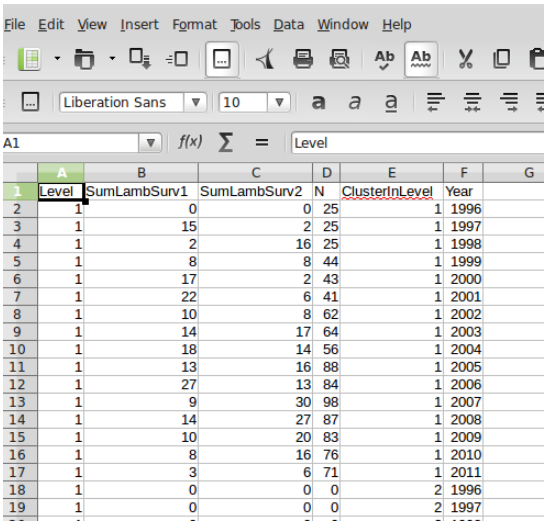
2 Data setup

Do not make your data pretty in Excel – R hates that. Instead, lay it out very simply. I generally follow these rules when working with data in Excel

- Avoid including commas in cells (allows for straighforward reformatting as a .csv for easy reading into R)
- Avoid surrounding your data with empty cells; keep it flush to the corner of the spreadsheet
- Avoid visual formatting in excel sheets that you'll read into R
- Avoid using spaces between words in column names

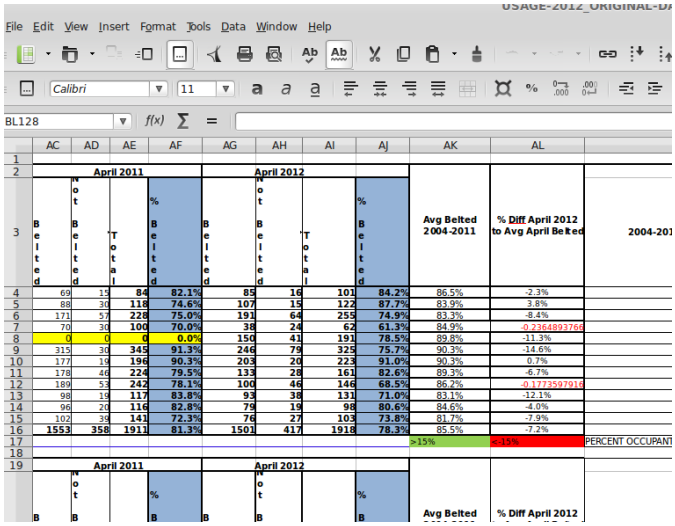
Below are examples of "ready", and "not-so-ready" spreadsheets.

Good data, ready for R



	A	B	C	D	E	F	G
1	Level	SumLambSurv1	SumLambSurv2	N	ClusterInLevel	Year	
2	1	0	0	25	1	1996	
3	1	15	2	25	1	1997	
4	1	2	16	25	1	1998	
5	1	8	8	44	1	1999	
6	1	17	2	43	1	2000	
7	1	22	6	41	1	2001	
8	1	10	8	62	1	2002	
9	1	14	17	64	1	2003	
10	1	18	14	56	1	2004	
11	1	13	16	88	1	2005	
12	1	27	13	84	1	2006	
13	1	9	30	98	1	2007	
14	1	14	27	87	1	2008	
15	1	10	20	83	1	2009	
16	1	8	16	76	1	2010	
17	1	3	6	71	1	2011	
18	1	0	0	0	2	1996	
19	1	0	0	0	2	1997	

Not-so-ready for R



I do all the data post-processing I can in R so that I've got a code record of exactly what modifications were made (this can be done in Excel as well by recording actions, but for me, it's easier to do systematically in R).

3 R background

The R project for statistical computing

R (www.r-project.org) is a statistical computing environment and language, primarily designed for statistical data analysis and graphical displays. It began as an open-source version of the S language and environment developed at Bell Laboratories. Early R-project initiation is generally credited to John Chambers and colleagues.

Referencing R

R is licensed under a General Public License (GPL) and should be cited in publication. You can access the appropriate citations for R, and for all of its individual packages using the `citation()` function shown below. I am a firm believer that all packages used in an analysis should be cited.

```
# Calling, and citing a particular package: example with lme4
require(lme4)

## Loading required package:  lme4
## Loading required package:  lattice
## Loading required package:  Matrix

citation("lme4")

##
## To cite package 'lme4' in publications use:
##
##   Douglas Bates, Martin Maechler, Ben Bolker and Steven Walker (2013). lme4:
##   Linear mixed-effects models using Eigen and S4. R package version 1.0-5.
##   http://CRAN.R-project.org/package=lme4
##
## A BibTeX entry for LaTeX users is
##
##   @Manual{,
##     title = {lme4: Linear mixed-effects models using Eigen and S4},
##     author = {Douglas Bates and Martin Maechler and Ben Bolker and Steven Walker},
##     year = {2013},
##     note = {R package version 1.0-5},
##     url = {http://CRAN.R-project.org/package=lme4},
##   }
```

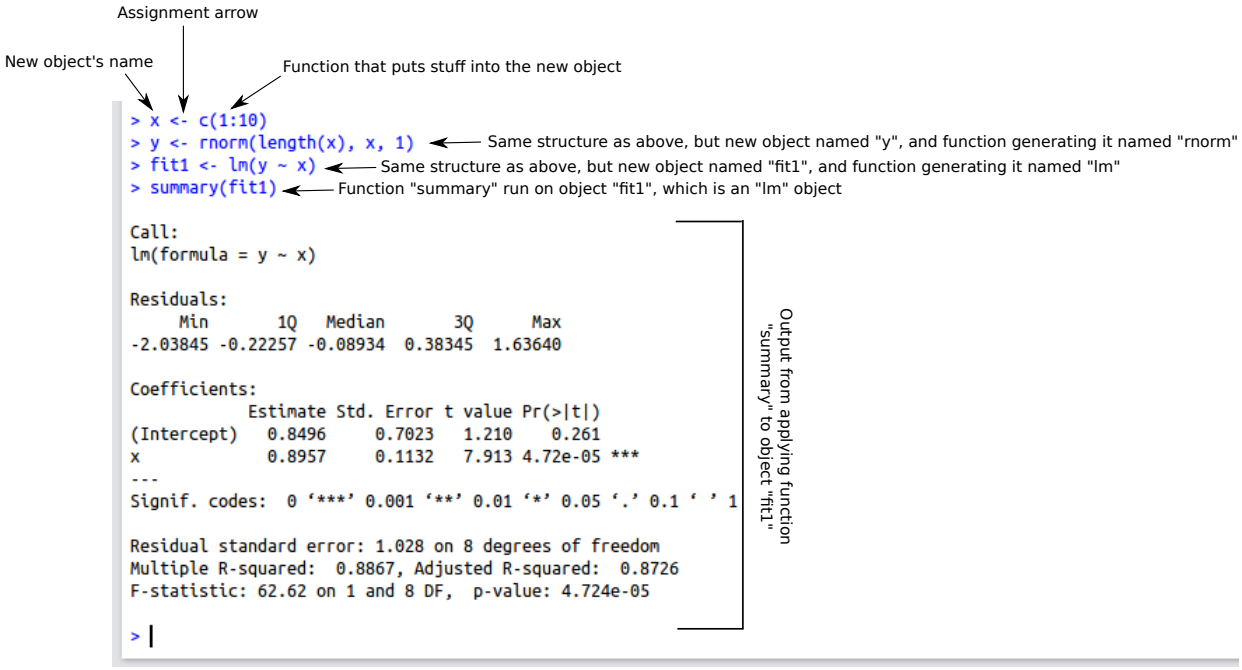
Objects and functions in R

R is an object-oriented language, which means that

Calculations

R has functions to calculate many basic quantities. Here are a few I use all the time:

Calculation	Function
`mean(x)`	Calculate mean of a vector x
`sd(x)`	Calculate standard deviation of a vector x
`sqrt(x)`	Calculate squareroot of number x
`exp(x)`	Exponentiate x
`log(x)`	Take natural log of x
`log10(x)`	Take log base 10 of x
`min(x)`	Returns minimum value of vector x
`max(x)`	Returns maximum value of vector x
`which.min(x)`	Returns index corresponding to minimum value in x
`which.max(x)`	Returns index corresponding to maximum value in x



4 Simulation as a unifying theme

Why bother?

1. All statistical analyses hinge on a set of assumptions. By simulating data under those assumptions, you gain some perspective on how well your model actually works.
2. Simulation forces you to carefully consider the data generating process.
3. Simulation is a first step toward determining statistical power, and helps in study design.

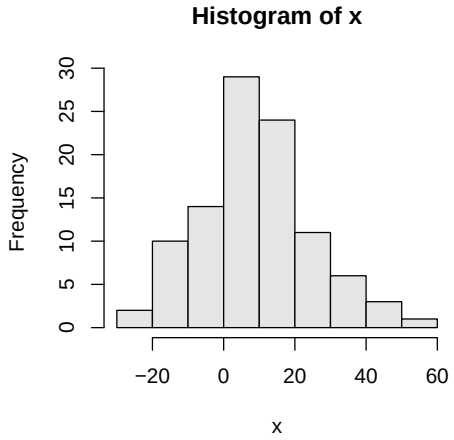
Simulating data in and accessing probability density information using R

`dnorm(x, mean = <mean>, sd = <sd>)` returns height of normal density curve at given value of x
`pnorm(k, mean = <mean>, sd = <sd>)` returns cumulative area under distribution up until point x = k
`qnorm(p, mean = <mean>, sd = <sd>)` returns the value of x that has p % of the distribution below it
`rnorm(n, mean = <mean>, sd = <sd>)` generates random numbers from the normal distribution

R can simulate and describe most commonly-used distributions. The four functions listed above are the normal distribution version of R's simulation and density tools. Parallel functions for Student's T distribution are `dt, pt, qt, rt`; for the Poisson distribution, they are `dpois, ppoit, qpois, rpois`, etc.

Example 1: Simulate from `rnorm` and build a histogram:

```
# -- simulating 100 draws from a normal distribution with mean = 10 and standard deviation
# = 15 -- (i.e., N(10, 15))
x <- rnorm(n = 100, mean = 10, sd = 15)  # simulate draws using rnorm and stick them in x
hist(x, col = "grey90")  # build a histogram of the vector of values in x
```

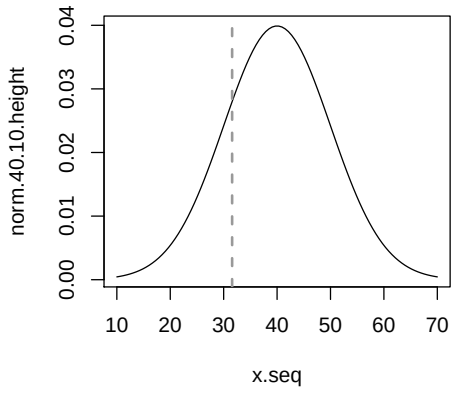


Example 2: Look up the value in a N(40, 10) distribution that has 20% of the area below it.

```
x.20perc <- qnorm(p = 0.2, mean = 40, sd = 10)
x.20perc

## [1] 31.58
```

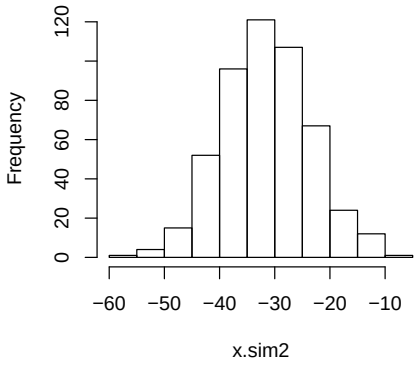
Verify that that mades sense with a plot.

```
# -- 1) build a sequence of x-values from 20 to 60
x.seq <- seq(from = 10, to = 70, length.out = 100)

# -- 2) calculate N(40, 10) desnities at each value of x.seq
norm.40.10.height <- dnorm(x.seq, mean = 40, sd = 10)

# -- 3) plot those x.seq - norm.40.10.height pairs as a line
plot(norm.40.10.height ~ x.seq, type = "l")
# -- add a dashed vertical line at x.20perc from above
abline(v = x.20perc, lty = 2, lwd = 2, col = "grey60")
```



Practice problem 1: Set your simulation seed to 7 (by typing `set.seed(7)`). In this problem, we'll work with a normal distribution that has a mean of -32 and standard deviation of 8.

1. Simulate 500 draws from this normal distribution, and build a histogram of those draws.



2. Find the minimum value of your 500 draws. Report the minumum value, as well as its corresponding position within your vector of simulations.

```
## [1] -55.79
## [1] 252
```

3. What value of N(-32, 8) distribution has 40% of the distribution's area below it?

```
## [1] -34.03
```

4. What is the distribution's height at -24?

```
## [1] 0.03025
```

5 Linear Models

```
lm()
```

Linear models relate a response to a predictor using a straight line. They assume that the expected value of a reponse variable Y can be efficiently expressed as a linear combination of predictors variables (aka "covariates"), written in the matrix **X**.

Structure (1) and counterexample that isn't linear (2)

$$y_i = \beta_0 + X_{i,1}\beta_1 + X_{i,2}\beta_2 + ... + X_{i,k}\beta_k + \epsilon_i \quad (1) \qquad y_i = \frac{\beta_1 X_i}{\beta_2 + X_i} + \epsilon_i \quad (2)$$

Residuals

A residual is the vertical distance from an observed point to a line. The residual associated with the i^{th} point in a model is usually denoted as ϵ_i. Points below a regression line have negative residuals; points above the regression line have positive residuals. Because linear models draw lines through the middle of a cloud of points, residuals in a linear model are restricted to sum to zero (i.e., $\sum_{i=1}^{N} \epsilon_i = 0$).

Assumptions

1. Mean of Y = some linear combination of available covariates
2. Distribution of residuals: $\epsilon_i \overset{iid}{N}(0, \sigma^2)$
   - Residual variance is constant
   - Residuals (and thus observations) are independent of one another
3. Covariates are fairly independent of one another

Questions Addressed

1. What is a minimal set of covariates necessary to predict a given phenomenon? (model selection)
2. Does changing the value of a particular variable X result in a linear change in the mean of some response, Y? (covariate-specific tests)

Simulating a linear relationship

Let's say we want to simulate data under the following linear model:

$$y_i = \beta_0 + \beta_1 x_i + \epsilon_i$$

$$\epsilon_i \overset{iid}{\sim} N(0, \sigma),$$

where $\beta_0 = 36$, $\beta_1 = 0.70$, and $\sigma = 1.2$. To simulate the relationship between x and y dictated by this model, use the following steps:

1. Specify the parameter values for β_0, β_1, and σ.
2. Specify a set of x values to simulate across
3. Calculate the mean value of y for each value of x you specified above (this is called the *expectation of Y given X*, written as $E(Y|X)$), using the specified β_0, and β_1 values

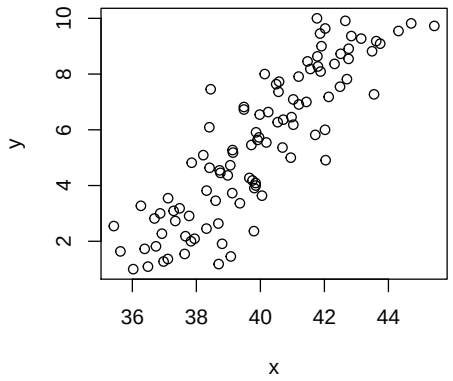
   ASIDE: when you see "expectation" in statistical writing, it's usually safe to think "*mean*".

4. Simulate values of y, using the `rnorm` function with mean = $E(Y|X)$ and standard deviation = σ

Example 3: Simulating data under a linear model

```
# -- 1) specify parameter values for the intercept term, beta_0, and the slope, beta_1
beta_0 <- 36  # at x = 0, E(Y) = 36
beta_1 <- 0.7  # for each 1-unit increase in X, E(Y|X) increases by 0.7 --#
sigma <- 1.2

# -- 2) make a vector of x values in a sequence from 1 to 10 --#
x <- seq(from = 1, to = 10, length.out = 100)

# -- 3) calculate E(Y|X) for all values of X --#
mean.y <- beta_0 + beta_1 * x

# -- set simulation seed --#
set.seed(123)

# -- 4) simulate some y values that have mean = mean.y, and have normally distributed
```

```r
# residuals with standard deviation = sigma
y <- rnorm(n = length(x), mean = mean.y, sd = sigma)

plot(x ~ y, xlab = "x", ylab = "y")
```



Fitting a simple linear regression model

The strategy for estimating the intercept and slope terms in a linear model is to minimize the sum of the squared residuals.

The `lm` function in `R` fits a linear model of the form "response $\sim$ predictors". The default is to add an intercept term, in addition to all specified models. To see the help for the `lm` function, use `?lm`.

```r
# -- fit a simple linear regression model on the data we simulated above
sim.lm.fit <- lm(y ~ x)
summary(sim.lm.fit)

##
## Call:
## lm(formula = y ~ x)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.9443 -0.6628 -0.0415  0.7782  2.5138
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  35.9262     0.2549   141.0   <2e-16 ***
## x             0.7331     0.0418    17.5   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.1 on 98 degrees of freedom
## Multiple R-squared:  0.758,Adjusted R-squared:  0.756
## F-statistic:  307 on 1 and 98 DF,  p-value: <2e-16

sd(sim.lm.fit$residuals)

## [1] 1.092
```

Hypothesis testing and model interpretation

The `lm` function returns two kinds of statistical tests, an overall F-test, describing whether any of the covariates in your model are important, and covariate-specific T-tests, describing the evidence that each specific covariate impacts the mean response.

Overall F-test: At the very bottom of the model summary, you see an overall F-test. This test evaluates whether any of your X variables have a significant relationship with the response (here we only have one X). The null hypothesis for the F-test is that none of the covariates impact the mean response; the alternative is that at least one has a significant impact on the mean of Y.

Hypotheses	Test-statistic
$H_0 : \beta_1 = \beta_2 = ... = \beta_k = 0$ H_A : at least one $\beta_j \neq 0$	$\frac{VariabilityExplainedByModel/(k-1)}{UnexplainedVariability/(n-k)}$

Reference distribution: $F(k-1, N-(k))$, where N is sample size, and k is number of βs

P-value: Area of reference distribution as far or further out the upper tail than the F statistic; low p-values mean that the covariates you included have an impact on the mean response

Covariate-specific T-tests: The coefficients table in the summary output contains test output corresponding to each separate β-term. These tests are of the following form:

Hypotheses	Test-statistic
$H_0 : \beta_1 = 0$ $H_A : \beta_1 \neq 0$	$\frac{\hat{\beta_1}-0}{\frac{sd}{\sqrt{n}}}$

Reference distribution: Student's T-distribution with (n-k) degrees of freedom, where k is number of parameters and n is number of observations

P-value: Area of reference distribution as far or further out the tails than the test statistic

Model interpretation

Estimates in a linear regression model's coefficient table reflect the mean change in Y associated with a one-unit increase in X. So, in the model fit above, a one-unit increase in X is associated with a 0.73-unit increase in the mean Y value. Does this make sense, based on our simulation inputs?

Practice problem 2

Set your seed to 451. Simulate linear regression data with an intercept of -1.6, a slope of .9, and a residual standard deviation of 18 on a sequency of 100 x-values running from 1 to 25.

1. Fit a simple linear regression model to your simulated data. You should produce the output shown below.

```
##
## Call:
## lm(formula = y2 ~ x2)
##
## Residuals:
##    Min     1Q Median     3Q    Max
##  -55.8  -11.5   -1.5   11.7   51.9
##
## Coefficients:
```

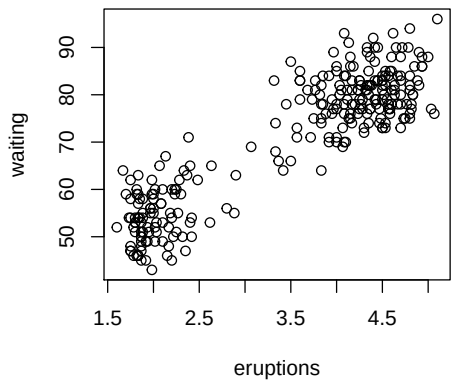
```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)    3.190      3.923    0.81    0.418
## x2             0.512      0.266    1.93    0.057 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 18.6 on 98 degrees of freedom
## Multiple R-squared:  0.0366,Adjusted R-squared:  0.0267
## F-statistic: 3.72 on 1 and 98 DF,  p-value: 0.0567
```

2. Report the p-value associated with a t-test of whether $\beta_0 = 0$. Is this a statistically significant relationship? At what signficance level?
3. What is the test-statistic corresponding to the hypothesis that $\beta_1 = 0$?
4. What is the probability that we would get a stronger relationship between X and Y than the one that we actually observe here just be random chance?
5. What is the reference distribution for the t-tests used here?

Real data example

Stage 1: Pull in dataset, check it, and plot out the potential predictors.

```
require(datasets)
# -- we'll use the faithful dataset --# -- display the names of variables it contains --#
names(faithful)

## [1] "eruptions" "waiting"

# -- plot faithful variables waiting time ~ eruption time --#
plot(waiting ~ eruptions, data = faithful)
```



Stage 2: Develop hypotheses and corresponding models. In this case, let the null hypothesis be that waiting time to the next eruption is independent of the last eruption's duration.

```
# -- fit linear model of faithful data --#
faithful.fit <- lm(waiting ~ eruptions, data = faithful)
summary(faithful.fit)
```

```
##
## Call:
## lm(formula = waiting ~ eruptions, data = faithful)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -12.080  -4.483   0.212   3.925  15.972
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   33.474      1.155    29.0   <2e-16 ***
## eruptions     10.730      0.315    34.1   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.91 on 270 degrees of freedom
## Multiple R-squared:  0.811,Adjusted R-squared:  0.811
## F-statistic: 1.16e+03 on 1 and 270 DF,  p-value: <2e-16
```

Variations on the X matrix

X is sometimes referred to as the design matrix. It contains n rows, where n is your number of observations, and k columns, where k is the number of βs in the model (note that k includes the **intercept**).

Example: Say we want to fit a linear model of weight (Y = weight) as a function of height (X_1 = height). We might not need an intercept in this model, since at height = 0, we might also expect weight to = 0, but we'll include one anyway, for good measure.

Then, a linear model of weight as a function of height is

$$Y_i = \beta_0 + \beta_1 X_{i,height} + \epsilon_i$$

Note that this is the same as

$$Y_i = \beta_0 \times 1 + \beta_1 X_{i,height} + \epsilon_i$$

If we say $Height_i$ = height of individual i, then we can write **X** as:

$$\mathbf{X} = \begin{bmatrix} 1 & Height_1 \\ 1 & Height_2 \\ ... & \\ 1 & Height_n \end{bmatrix}$$

Two-sample t-test as simple linear model

`lm()` or `t.test()`

Two-sample t-tests are a special class of linear model. The objective of a two-sample t-test is to compare two groups, and determine whether their means differ. This is the same as comparing a linear model with an intercept term and an indicator variable telling whether each observation is in group 1 or not to a model with only an intercept term.

Question: Do two groups have different means (i.e., does $\mu_0 = \mu_1$)?

Hypotheses and corresponding models

Traditional formulation	Formulation under linear model
$H_0 : \mu_0 = \mu_1$, aka $\mu_0 - \mu_1 = 0$	$H_0 : \beta_1 = 0$, aka both groups' means = β_0
$H_A : \mu_0 \neq \mu_1$, aka $\mu_0 - \mu_1 \neq 0$	$H_A : \beta_1 \neq 0$; aka one group's mean = β_0, other group's = $\beta_0 + \beta_1$

Test statistic

1. Calculate $\bar{x_0}$, $\bar{x_1}$, $\hat{s_0}$, and $\hat{s_1}$ from your data
2. $t.s. = \frac{\bar{x_0} - \bar{x_1}}{\sqrt{\left(\frac{s_0^2}{n_0}\right) + \left(\frac{s_1^2}{n_1}\right)}}$

Reference distribution Student's T distribution with 1 degree of freedom (because models differ by 1 parameter: first model only contains one mean, since $\mu_1 = \mu_2$; second model contains two means, since $\mu_1 \neq \mu_2$).

P-value Area as or more extreme than your observed test statistic in the distribution under the null hypothesis.

Example: Say we want to fit a linear model of weight (Y = weight) as a function of sex. To do this, we need to build an **indicator variable**, which indicates when individuals are in a given group by taking on the value 1, and taking on 0 when they aren't in the group. If we think of $X_{,sex}$ as an indicator variable that takes on the value 1 when the individual is male, and 0 otherwise, then we can address this question with a linear model that looks very similar to the one above.

Then, a linear model of weight as a function of height is

$$Y_i = \beta_0 + \beta_1 X_{i,sex} + \epsilon_i$$

Note that this is the same as

$$Y_i = \beta_0 \times 1 + \beta_1 X_{i,sex} + \epsilon_i$$

If we say Sex_i = indicator for sex of individual i, with individual 1 being Male, individual 2 being Female, ... individual n being Male, then we can write $\mathbf{X}$ as:

$$\mathbf{X} = \begin{bmatrix} 1 & Sex_1 \\ 1 & Sex_2 \\ \dots & \\ 1 & Sex_n \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \\ \dots & \\ 1 & 1 \end{bmatrix}$$

Note that under this model, $\beta_0 + \beta_1 X_{i,sex}$ can only take on two values:

- For females (remember, X_{sex} for females = 0), $\beta_0 + \beta_1 X_{i,sex} = \beta_0 + \beta_1 \times 0 = \beta_0$.
- For males (remember, X_{sex} for males = 1), $\beta_0 + \beta_1 X_{i,sex} = \beta_0 + \beta_1 \times 1 = \beta_0 + \beta_1$.

If males and females don't differ in their expected weights, then β_1 shouldn't differ significantly from 0.

Here's some code to simulate data that might reflect this process:

```
# -- 1) specify parameter values for the intercept term (i.e., the female-specific mean),
# beta_0, and the adjustment for males, beta_1
beta_0.weight <- 135  # female mean weight
beta_1.weight <- 40  # adjustment to get from female mean to male mean
sigma.weight <- 8  # standard deviation of weights around group means

# -- 2) make a vector of x.sex values that consists of 50 0s and 50 1s -- (so the first 50
# points are females, the latter 50 are males)
x.sex <- c(rep(0, 50), rep(1, 50))
```

```r
# -- 3) calculate E(Y|X) for all values of X --#
mean.y.weight <- beta_0.weight + beta_1.weight * x.sex

# -- set simulation seed --#
set.seed(451)

# -- 4) simulate some y values that have mean = mean.y, --# -- and have normally
# distributed residuals with standard deviation = sigma --#
y.weight <- rnorm(n = length(x.sex), mean = mean.y.weight, sd = sigma.weight)
```

Now we'll fit a linear regression model (using `lm`) and a t-test (using `t.test`) to our simulated data, and compare outputs.

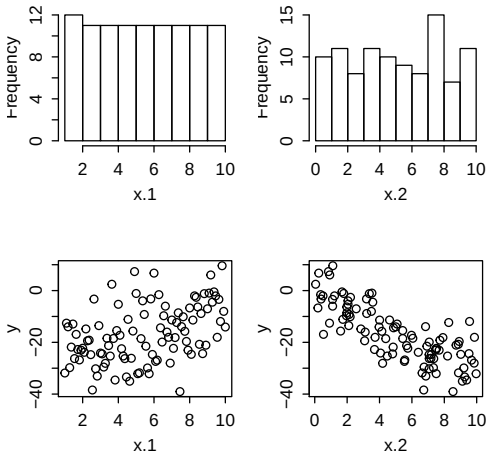
```r
# -- fit linear model --#
weight.lm.fit <- lm(y.weight ~ x.sex)
summary(weight.lm.fit)

##
## Call:
## lm(formula = y.weight ~ x.sex)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -24.016  -5.247  -0.498   5.338  23.957
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   135.92       1.17   116.0   <2e-16 ***
## x.sex          37.95       1.66    22.9   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.29 on 98 degrees of freedom
## Multiple R-squared:  0.842,Adjusted R-squared:  0.841
## F-statistic:  524 on 1 and 98 DF,  p-value: <2e-16

# -- fit same model using t.test function
weight.ttest.fit <- t.test(y.weight ~ x.sex)
weight.ttest.fit

##
##  Welch Two Sample t-test
##
## data:  y.weight by x.sex
## t = -22.89, df = 96.04, p-value < 2.2e-16
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -41.24 -34.66
## sample estimates:
## mean in group 0 mean in group 1
##           135.9           173.9
```

Multiple linear regression

So far, we've looked at models that contain only a single covariate beyond the intercept term (i.e., X contains only a vector of 1s and one vector reflecting a covariate). Linear regression that includes multiple covariates in addition to the intercept terms is referred to as "multiple regression".

Here's code that simulates a multiple regression model where `mean.y` is a function of two predictors, `x.1` and `x.2`

```
# -- 1) specify parameter values for the intercept term, beta_0, and the slope, beta_1
beta_0 <- -10  # at x.1 = 0 and x.2 = 0, E(Y) = -10
beta_1 <- 1.4  # for each 1-unit increase in x.1, E(Y|x.1) increases by 1.4
beta_2 <- -3  # for each 1-unit increase in x.2, E(Y|x.2) decreases by -3
sigma <- 6

# -- 2) make vectors of x.1 (in a sequence from 1 to 10) --# and x.2 (that are draws from a
# uniform distribution from 0 to 10) --#
x.1 <- seq(from = 1, to = 10, length.out = 100)
x.2 <- runif(100, min = 0, max = 10)

# -- 3) calculate E(Y|x.1, x.2) for all values of x.1 and x.2 --#
mean.y <- beta_0 + beta_1 * x.1 + beta_2 * x.2

# -- set simulation seed --#
set.seed(123)

# -- 4) simulate some y values that have mean = mean.y, --# -- and have normally
# distributed residuals with standard deviation = sigma --#
y <- rnorm(n = length(x.1), mean = mean.y, sd = sigma)

# -- 5) plot all data --#
par(mfrow = c(2, 2), mex = 0.6)
hist(x.1, main = "", xlab = "x.1")
hist(x.2, main = "", xlab = "x.2")
plot(y ~ x.1, xlab = "x.1", ylab = "y")
plot(y ~ x.2, xlab = "x.2", ylab = "y")
```



Next, use the `lm` function to fit the multiple regression model

$$y_i = \beta_0 + \beta_1 X_{i,1} + \beta_2 X_{i,2}$$

. Before fitting, consider: what do you expect the coefficient estimates associated with β_1 and β_2 to be?

```
multreg.sim.fit <- lm(y ~ x.1 + x.2)
summary(multreg.sim.fit)

##
## Call:
## lm(formula = y ~ x.1 + x.2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -15.307  -3.283  -0.446   3.634  13.235
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -11.238      1.645   -6.83  7.4e-10 ***
## x.1            1.580      0.210    7.52  2.8e-11 ***
## x.2           -2.842      0.189  -15.05  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.5 on 97 degrees of freedom
## Multiple R-squared:  0.758,Adjusted R-squared:  0.753
## F-statistic:  152 on 2 and 97 DF,  p-value: <2e-16
```

Interpretting multiple regression output

T-tests for coefficient significance in multiple regression still compare each β-term to a null hypothesized value of 0. They compare test-statistics to T-distributions with $n - k$ degrees of freedom, where k is the number of parameters in the model (here, $k = 3$). Estimates associated with significant coefficients are interpretted as reflecting the mean change in y associated with a one-unit increase in that particular x (here, $x.1$ or $x.2$), ***conditional on all other covariates being present in the model***.

The overall-F test compares the fitted model to an intercept-only model. Since in this case, that means a comparison of a model with 3 parameters (β_0, β_1, and β_2) to a model with only 1 parameter (β_0), the reference distribution is an F with 2 and (100-2-1 =) 97 degrees of freedom.

Comparing two regression models with different covariates

Identifying an appropriate set of predictors is a critical step in model fitting. This identification process is referred to as *model selection*. It is typically done using one of two methods: **likelihood ratio tests**, or *information-theoretic approaches* like AIC or BIC.

With Likelihood Ratio Tests

`anova()`

Likelihood ratio tests (LRTs) are used to compare two models, one of which must be nested in the other one. This means that all the X-variables contained in one model must also be included in the other, plus the other has some additional variables in it.

LRTs test the hypothesis that the residuals from the smaller model do not exhibit substantially more spread than residuals from the larger model. The null hypothesis is that the two models do not differ in their residual variance.

LRTs use an F-test, which is implemented through R's ANOVA function. Build two `lm` objects, and then pass them both to `anova` as shown below. The contents of the resulting `anova` object contains the LRT

output.

With information-theoretic approaches

```
AIC()
```

The major advantage of information theoretic approaches like AIC is that they do not require the models to be nested in one another. The major weakness is that the statistical community remains a little nebulous as to the most appropriate penalty term to use. AIC is calculated as

$$AIC = 2k - 2ln(L)$$

where k is the number of parameters in your model, and $ln(L)$ is the log-likelihood associated with your model. The AIC value associated with a given model can be extracted by applying the `AIC` function to the model object. AIC values from different models can be compared then. Lower AIC values indicate better model performance. Differences in AIC of less than 2 points are not thought to be substantial; differences between 2 and 7 are typically viewed as suggesting the lower-scoring model performs marginally better than the higher one, and differences greater than 7 are generally thought to indicate that the lower-scoring model is substantially superior to the higher-scoring one.

Example 4: Multiple regression and model selection. In this example, we'll fit several multiple regression models to try and explain illiteracy rates using R's state.x77 dataset contained in the datasets package. First, let's take a look at the dataset.

```
# -- use the state.x77 dataset from the datasets library --#
require(datasets)
names(state.x77)  # returns names of all columns in the dataset

## NULL

head(state.x77)  # returns the first few rows of the dataset

##            Population Income Illiteracy Life Exp Murder HS Grad Frost   Area
## Alabama          3615   3624        2.1    69.05   15.1    41.3    20  50708
## Alaska            365   6315        1.5    69.31   11.3    66.7   152 566432
## Arizona          2212   4530        1.8    70.55    7.8    58.1    15 113417
## Arkansas         2110   3378        1.9    70.66   10.1    39.9    65  51945
## California      21198   5114        1.1    71.71   10.3    62.6    20 156361
## Colorado         2541   4884        0.7    72.06    6.8    63.9   166 103766
```

We'll fit three different models:

1. Illiteracy as a function of Population and HS Grad
2. Illiteracy as a function of Population, HS Grad, and Income
3. Illiteracy as a function of Population and Life Expectancy

Here we build objects for each of these models.

```
# -- use the state.x77 dataset from the datasets library -- first, need to build a new HS
# Grad and a new Life Exp column, with no spaces
state.x77.new <- as.data.frame(state.x77)
state.x77.new$HSGrad <- state.x77[, 6]
state.x77.new$LifeExp <- state.x77[, 4]
illit.fit1 <- lm(Illiteracy ~ Population + HSGrad, data = state.x77.new)
illit.fit2 <- lm(Illiteracy ~ Population + HSGrad + Income, data = state.x77.new)
illit.fit3 <- lm(Illiteracy ~ Population + LifeExp, data = state.x77.new)
```

Let's print out one of these models to see what's being returned

```
# -- use the state.x77 dataset from the datasets library --#
illit.fit1

##
## Call:
## lm(formula = Illiteracy ~ Population + HSGrad, data = state.x77.new)
##
## Coefficients:
## (Intercept)   Population       HSGrad
##    3.76e+00     5.91e-06    -4.93e-02
```

Next, we want to do model selection. Let's first get AIC values for all three models.

```
# -- use the AIC function to extract AIC values associated with each model --#
aic.illit.fit1 <- AIC(illit.fit1)
aic.illit.fit2 <- AIC(illit.fit2)
aic.illit.fit3 <- AIC(illit.fit3)

aic.vec <- c(aic.illit.fit1, aic.illit.fit2, aic.illit.fit3)
aic.vec

## [1] 70.94 72.66 77.77
```

We can also use an LRT to compare nested fits. Here, illit.fit1 is nested in illit.fit2, so they can be compared via an LRT. Recall, we use the `anova` function to do this.

```
lrt.illit <- anova(illit.fit1, illit.fit2)
lrt.illit

## Analysis of Variance Table
##
## Model 1: Illiteracy ~ Population + HSGrad
## Model 2: Illiteracy ~ Population + HSGrad + Income
##   Res.Df  RSS Df Sum of Sq    F Pr(>F)
## 1     47 10.3
## 2     46 10.2  1    0.0573 0.26   0.61
```

Practice Problem 3: Still using the state.x77.new data we used above, fit a linear model of Murder as a function of Population, Income, and Illiteracy.

1. Fit the model described above. You should get the following coefficient table.

```
##
## Call:
## lm(formula = Murder ~ Population + Income + Illiteracy, data = state.x77.new)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -4.785 -1.677 -0.084  1.478  7.642
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 1.34e+00   3.37e+00    0.40    0.693
```

```
## Population  2.22e-04   8.42e-05    2.64    0.011 *
## Income      6.44e-05   6.76e-04    0.10    0.925
## Illiteracy  4.11e+00   6.71e-01    6.13  1.9e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.51 on 46 degrees of freedom
## Multiple R-squared:  0.567,Adjusted R-squared:  0.539
## F-statistic: 20.1 on 3 and 46 DF,  p-value: 1.84e-08
```

2. Are any of the predictors contained in this model relevant? What test or tests tell you this, and what are their corresponding p-values?

3. Which of the three covariates that you included in your model had a significant bearing on Murder?

4. Build a new model that omits the insigificant predictor in your original model. Compare the new model to the old model using a likelihood ratio test (output shown below). Why is this test appropriate in this case? Which model does the test suggest is more appropriate?

```
## Analysis of Variance Table
##
## Model 1: Murder ~ Population + Illiteracy
## Model 2: Murder ~ Population + Income + Illiteracy
##   Res.Df RSS Df Sum of Sq    F Pr(>F)
## 1     47 289
## 2     46 289  1     0.057 0.01   0.92
```

Considerations when fitting linear models

- Interactions
- Colinearity

6 Generalized Linear Models

```
glm()
```

Introduction

Generalized linear models ("GLMs") are variants on traditional linear models. They still model an attribute of the response as a linear function of covariates. The difference comes in how the response variable gets treated.

In a linear model, Y_i itself can be expressed as a linear combination of covariates (plus ϵ_i. That formulation requires that Y be able to take on any real value, and that's obviously not the case for all data.

For example, consider a situation where outcomes are binary, such that a given sampling unit experiences an event or not. In those cases, the response can ONLY be 0 or 1; it cannot take on all possible values (there is not such thing as a half a succes).

Link functions and residuals

GLMs allow us to apply linear modeling approaches in situations where responses are not distributed over the entire set of real numbers. This extension is done by transforming the responses via a ***link function***, which is matched to a reasonable probability density function for the response data in hand.

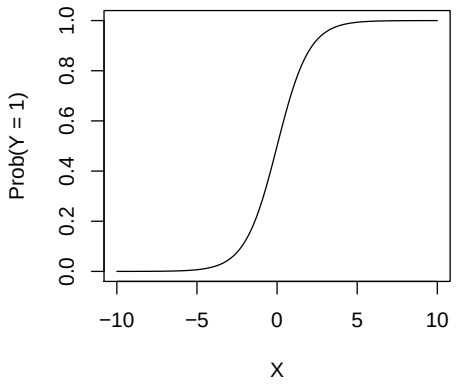
Probability density functions commonly used in GLMs are *binomial, Poisson, negative binomial, hypergeometric,* and *gamma* functions, although the structure extends to many other density functions as well. All of these families of probability functions are accessible for generalized linear modeling through the `glm` function in `R`.

Logistic Regression

Logistic regression is used to model binary or binomial response data as a function of various predictors.

Consider a continuous variable X, and a binary response Y, which can only take on the values 0 and 1. Now, imagine that the chance that Y = 1 changes with changing X, such that Y is very likely to be 0 for low values of X, and very likely to be 1 for high values of X. Then, a picture of the Probability that Y = 1 as a function of X might look like this:

```
x <- seq(-10, 10, length.out = 100)
y <- 1/(1 + exp(-x))
plot(y ~ x, type = "l", xlab = "X", ylab = "Prob(Y = 1)")
```



You may recognize the function I've used to calculate y as the logistic function. We're basically saying that the probability that Y = 1 follows a logistic function, with a "carrying capacity" of 1.

In logistic regression, we model the probability of a success. Since a probability must fall between 0 and 1, the bounding properties of the logistic function make it a nice option for us. To transform binary or binomial data to this probability scale, we use the logit function, which is the inverse of the logistic.

So, in logistic regression model, we fit models of the general form

$$logit(Y_i) = \beta_0 + \beta_1 X_{1,i} + \beta_2 X_{2,i} + ... + \beta_k X_{k,i} + \epsilon_i$$

Logistic Regression Example 1: Simulating logistic data

```
# -- Anorexia example --#
require(datasets)  #call the MASS package, which contains the anorexia data
names(infert)  #show the column names for the anorexia dataset
```

```
## [1] "education"      "age"            "parity"         "induced"        "case"
## [6] "spontaneous"    "stratum"        "pooled.stratum"

head(infert)  #show the first few rows of the anorexia dataset

##   education age parity induced case spontaneous stratum pooled.stratum
## 1    0-5yrs  26      6       1    1           2       1              3
## 2    0-5yrs  42      1       1    1           0       2              1
## 3    0-5yrs  39      6       2    1           0       3              4
## 4    0-5yrs  34      4       2    1           0       4              2
## 5   6-11yrs  35      3       1    1           1       5             32
## 6   6-11yrs  36      4       2    1           1       6             36

# -- plot the response --#
infert.fit <- glm(case ~ age + spontaneous, data = infert, family = "binomial")
summary(infert.fit)

##
## Call:
## glm(formula = case ~ age + spontaneous, family = "binomial",
##     data = infert)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -1.560  -0.709  -0.663   0.899   1.848
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -1.8532     0.9111   -2.03    0.042 *
## age           0.0150     0.0278    0.54    0.588
## spontaneous   1.0738     0.1978    5.43  5.7e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 316.17  on 247  degrees of freedom
## Residual deviance: 283.47  on 245  degrees of freedom
## AIC: 289.5
##
## Number of Fisher Scoring iterations: 4
```

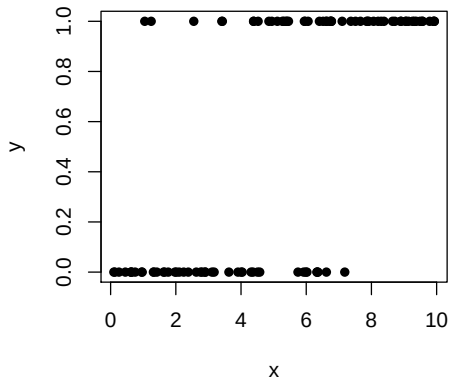
Logistic Regression Example 2: Infertility

The `infert` dataset, distributed in R's `datasets` package, contains information on infertility status in women (in the `case` variable), as well as information on education, age, parity, previous number of induces and spontaneous abortion evets, and more (use `?infert` to see more information).

Our goal is to model a woman's probability of being infertile as a function of these covariates. Infertility status (i.e., `case`) is a binary response, therefore we need to use a GLM.

```
# -- Logistic regression simulation example --#
set.seed(321)
x <- runif(100, 0, 10)
# pick 100 numbers from a uniform distribution ranging from 0 to 10
```

```r
y <- rbinom(100, 1, prob = x/10)
# simulate response values, such that the probability that Y = 1 is equal to x/10 for each
# simulated x value
plot(y ~ x, pch = 16)
```




```r
logistic.fit1 <- glm(y ~ x, family = "binomial")
summary(logistic.fit1)

##
## Call:
## glm(formula = y ~ x, family = "binomial")
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -1.968  -0.564   0.244   0.640   2.330
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)   -3.402      0.718   -4.74  2.2e-06 ***
## x              0.722      0.135    5.35  8.6e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 137.63  on 99  degrees of freedom
## Residual deviance:  82.32  on 98  degrees of freedom
## AIC: 86.32
##
## Number of Fisher Scoring iterations: 5
```

Poisson regression: Insect Sprays

The `InsectSprays` dataset contains information on insect counts after application of various insecticides. We want to develop modesl that compare the efficacy of different treatments (there are six, labeled A through F). Counts by nature can only take on non-negative integer values. Poisson random variables fill

that particular form, so they are often used as a first cut for fitting counts data. So, here we'll fit a GLM with a Poisson family, with `spray` as the predictor. The link-function typically associated with Poisson responses is a log function, so our model will be of this form:

$$log(Count_i) = \beta_0 + \beta_1 I_{SprayB} + \beta_2 I_{SprayC} + \beta_3 I_{SprayD} + \beta_4 I_{SprayE} + \beta_5 I_{SprayF} + \epsilon_i$$

We are particularly interested in the significance of the β-coefficients associated with each spray.

```
# -- InsectSprays example --# require(datasets) #call the MASS package, which contains the
# anorexia data
names(InsectSprays)  #show the column names for the anorexia dataset

## [1] "count" "spray"

head(InsectSprays)  #show the first few rows of the anorexia dataset

##   count spray
## 1    10     A
## 2     7     A
## 3    20     A
## 4    14     A
## 5    14     A
## 6    12     A

# -- plot the response --#
sprays.fit1 <- glm(count ~ spray, data = InsectSprays, family = "poisson")
summary(sprays.fit1)

##
## Call:
## glm(formula = count ~ spray, family = "poisson", data = InsectSprays)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -2.385  -0.888  -0.148   0.606   2.692
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)   2.6741     0.0758   35.27   <2e-16 ***
## sprayB        0.0559     0.1057    0.53     0.60
## sprayC       -1.9402     0.2139   -9.07   <2e-16 ***
## sprayD       -1.0815     0.1507   -7.18    7e-13 ***
## sprayE       -1.4214     0.1719   -8.27   <2e-16 ***
## sprayF        0.1393     0.1037    1.34     0.18
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 409.041  on 71  degrees of freedom
## Residual deviance:  98.329  on 66  degrees of freedom
## AIC: 376.6
##
## Number of Fisher Scoring iterations: 5
```

Additional considerations when fitting GLMs

GLMs come with some risks, and you should research them further prior to application. Here are a few:

Overdispersion (i.e., are your data REALLY Poisson??)

- Pseudo-replication
- Zero-inflation
- Incorrect mean-variance relationship

Tranforming back to the original scale

This can be tricky. It's generally a good idea to transform your coefficient estimates and confidence intervals back to the original scale (since people want to think about changes in probability Y occurs associated with a single unit increase in X). That transformation depends on the particular link function you used. Act with caution, and note that your confidence intervals on the original scale will not be symmetric (and that's okay – that's how they should be).

Notation conventions throughout this document

Notation	Name	Meaning
$\sim$	"tilde"	"is distributed as" or "is a function of"
<Stuff>	hard brackets	fill in this region with your own specifications
<-	Assignment arrow	In R, assigns values generated by function on right-hand side to object on left-hand side
μ	"mu"	Parameteric population mean
σ	"sigma"	Parametric population standard deviation
$E(Y)$	Expected value of Y	Mean value of variable Y
$E(Y\|X)$	Expected value of Y, given X	Mean value of variable Y, associated with a particular value of variable X (for example, mean weight (Y) associated with a particular height (X)