Statistics

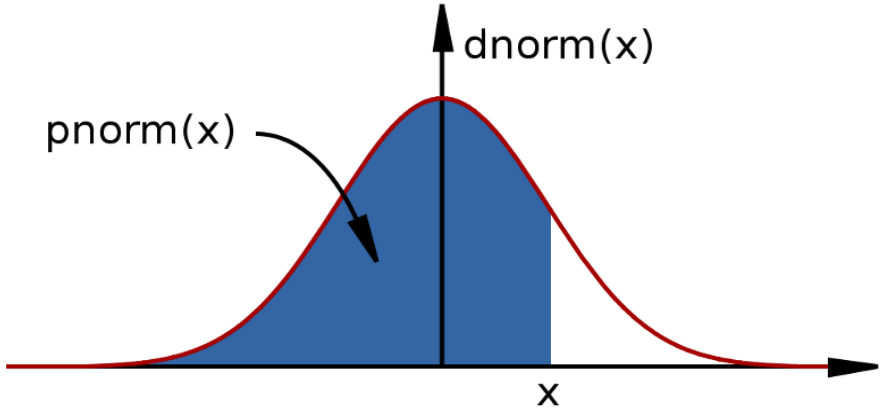
5

Built-in support for statistics

- R is a statistical programming language
  - Classical statistical tests are built-in
  - Statistical modeling functions are built-in
  - Regression analysis is fully supported
  - Additional mathematical packages are available
    - MASS, Waves, sparse matrices, etc

Pseudo-random numbers and distributions

- mostly commonly used distributions are built-in, functions have stereotypical names, e.g. for normal distribution:
    - pnorm - cumulative distribution for x
    - qnorm - inverse of pnorm (from probability gives x)
    - dnorm - distribution density
    - rnorm - random number from normal distribution



- available for variety of distributions: punif (uniform), pbinom (binomial), pnbinom (negative binomial), ppois (poisson), pgeom (geometric), phyper (hyper-geometric), pt (T distribution), pf (F distribution) ...

Two sample tests
Basic data analysis

- Comparing 2 variances
  - Fisher's F test

`var.test()`

- Comparing 2 sample means with normal errors
  - Student's t test

`t.test()`

- Comparing 2 means with non-normal errors
  - Wilcoxon's rank test

`wilcox.test()`

- Comparing 2 proportions
  - Binomial test

`prop.test()`

- Correlating 2 variables
  - Pearson's / Spearman's rank correlation

`cor.test()`

- Testing for independence of 2 variables in a contingency table
  - Chi-squared

`chisq.test()`

  - Fisher's exact test

`fisher.test()`

Comparison of 2 data sets example
Basic data analysis

- Men, on average, are taller than women.
  - The steps
    1. Determine whether variances in each data series are different
       - Variance is a measure of sampling dispersion, a first estimate in determining the degree of difference
         - *Fisher's F test*
    2. Comparison of the mean heights.
       - Determine probability that mean heights really are drawn from different sample populations
         - *Student's t test, Wilcoxon's rank sum test*

1. Comparison of 2 data sets
Fisher's F test

- Read in the data file into a new object, heightData

```
heightData<-read.csv("10_heightData.csv",header=T)
```

- **attach** the data frame so we don't have to refer to it by name all the time:

```
attach(heightData)
```

- Do the two sexes have the same variance?

```
var.test(Female,Male)
```

```
F test to compare two variances

data:  Female and Male
F = 1.0073, num df = 99, denom df = 99, p-value = 0.9714
alternative hypothesis: true ratio of variances is not equal to 1
95 percent confidence interval:
 0.6777266 1.4970241
sample estimates:
ratio of variances
           1.00726
```

2. Comparison of 2 data sets
Student's t test

- Student's t test is appropriate for comparing the difference in mean height in our data.
  - Remember a t test = $\frac{\text{difference in two sample means}}{\text{standard error of the difference of the means}}$

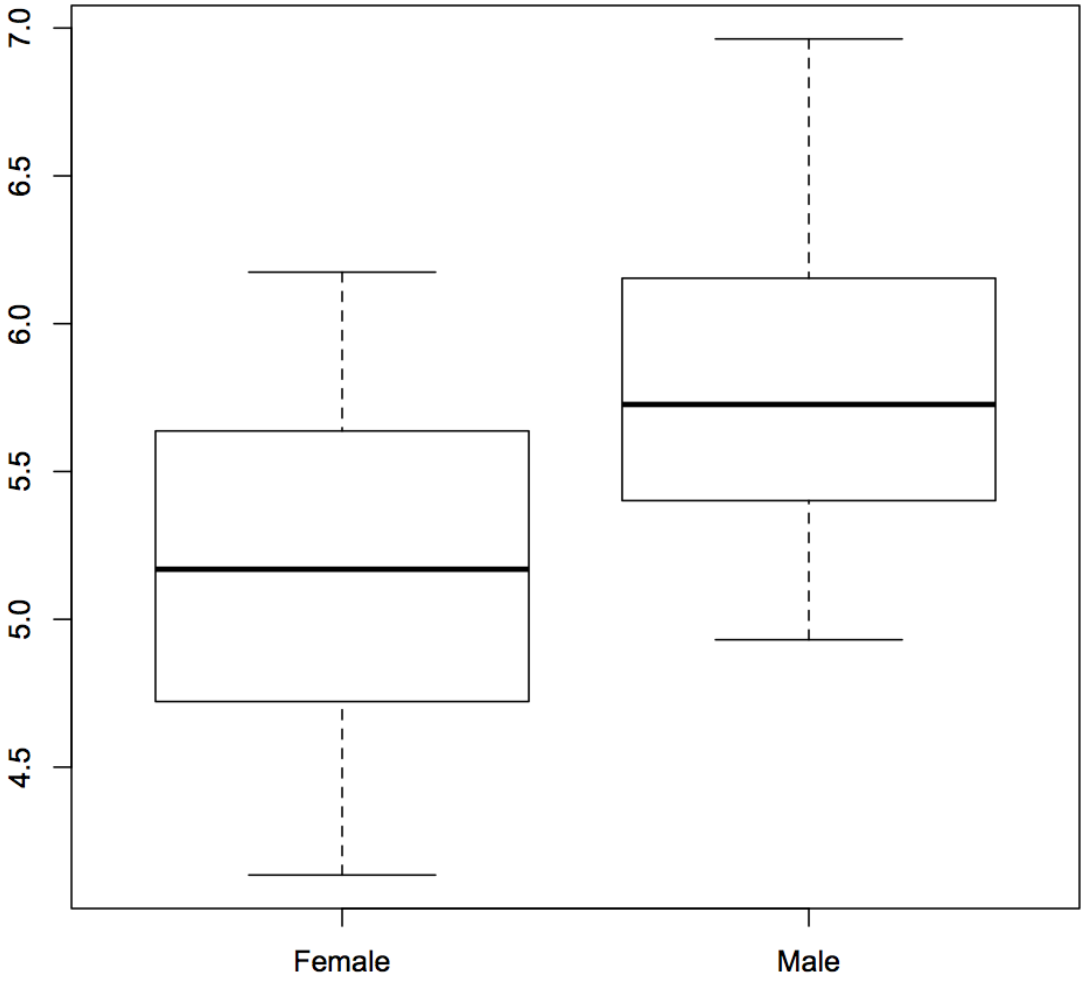
```
t.test(Female,Male)

Welch Two Sample t-test

data:  Female and Male
t = -8.4508, df = 197.997, p-value = 6.217e-15
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.7788497 -0.4841288
sample estimates:
mean of x mean of y
 5.168725  5.800214
```

3. Comparison of 2 data sets
Review findings

```
> boxplot(heightData)
```



Linear regression
Basic data analysis

- Linear modeling is supported by the function **`lm()`**
  - **`example(lm)`** # the output assumes you know a fair bit about the subject

- lm is really useful for plotting lines of best fit to XY data in order to determine, intercept, gradient & Pearson's correlation coefficient
  - This is very easy in R

- Three steps to plotting with a best fit line
  - Plot XY scatter-plot data
  - Fit a linear model
  - Add bestfit line data to plot with abline() function

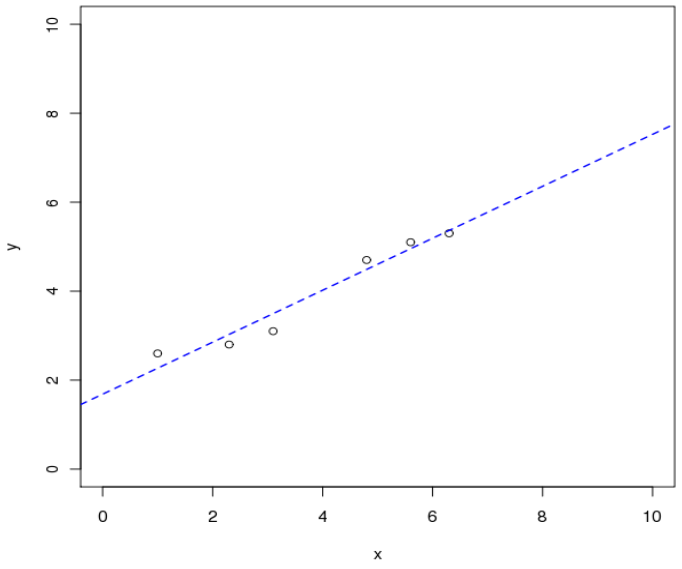
Typical linear regression analysis
Basic data analysis

X	Y
1.0	2.6
2.3	2.8
3.1	3.1
4.8	4.7
5.6	5.1
6.3	5.3

```
> x<-c(1, 2.3, 3.1, 4.8, 5.6, 6.3)
> y<-c(2.6, 2.8, 3.1, 4.7, 5.1, 5.3)
> plot(y~x, xlim=c(0,10),ylim=c(0,10))
```

Note formula notation
(y is given by x)

```
> myModel<-lm(y~x)
> abline(myModel,lty=2,lwd=1.5,col="blue")
```



Get the coefficients of the fit from:
`summary.lm(myModel)` **and**
`coef(myModel)`
`resid(myModel)`
`fitted(myModel)`

Get QC of fit from
`plot(myModel)`

Find out about the fit data from
`names(myModel)`

The linear model object
Basic data analysis

- Summary data describing the linear fit is given by
  - summary.lm(myModel)

```
> summary.lm(myModel)

Call:
lm(formula = y ~ x)

Residuals:
       1        2        3        4        5        6
 0.33159 -0.22785 -0.39520  0.21169  0.14434 -0.06458

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  1.68422    0.29056   5.796   0.0044 **
x            0.58418    0.06786   8.608   0.0010 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.3114 on 4 degrees of freedom
Multiple R-squared: 0.9488,   Adjusted R-squared: 0.936
F-statistic:  74.1 on 1 and 4 DF,  p-value: 0.001001
```

Y intercept

Gradient

Good fit: would happen 1 in 1000 by chance

R^2 , with pValue

Modelling formulae

- It is very easy to extend R model formulas to do multiple regressions, ANOVAs, include interactions...
- Suppose we had two explanatory variables **x** and **z** and one response variable **y**.

```
> y~x         # If x is continuous, this is linear regression
> y~x         # If x is categorical, this is ANOVA
> y~x+z       # If x and z are continuous, this is multiple regression
> y~x+z       # If x and z are categorical this is a two-way ANOVA
> y~x+z+x:z   # : is the symbol for the interaction term
> y~x*z       # * is a shorthand for the above: x+z and their interaction
```

Exercise
The coin toss

To learn how the distribution functions work, try simulating tossing a fair coin 100 times and then show that it is fair.

1) We can model a coin toss using the binomial distribution. Use the **rbinom** function to generate a sample of 100 coin tosses. Look up the binomial distribution help page to find out what arguments this function needs.
2) How many heads or tails were there in your sample? You can do this in two ways; either select the number of successes using indices, or convert your sample to a factor and get a summary of the factor.
3) If we toss a coin 50 times, what is the probability that we get exactly 25 heads? What about 25 heads or less? Use **dbinom** and **pbinom** to find out.
4) The argument to **pbinom** is a vector, so try calculating the probabilities for getting any number of coin tosses from 0 to 50 in fifty trials using **dbinom**. Plot these probabilities using **plot**. Does this plot remind you of anything?

Coin toss answers

- To simulate a coin toss, give **rbinom** a number of observations, the number of trials for each observation, and a probability of success:

```
> coin.toss<-rbinom(100, 1, 0.5)
```

- Because we only specified one trial per observation, we either have an outcome of 0 or 1 successes. To get the number of successes, use indices or a factor to look up the number of 1s in the **coin.toss** vector (your numbers will vary):

```
> length(coin.toss[coin.toss==1])
[1] 50
> summary(factor(coin.toss))
 0  1
50 50
```

Coin toss answers

The probability of getting exactly 25 heads from 50 observations of a fair coin:

```
> dbinom(25, 50, 0.5)
```

The probability of getting 25 heads or less from 50 observations of a fair coin:

```
> pbinom(25, 50, 0.5)
```

The probabilities for getting all numbers of coin tosses from 0 to 50 in fifty trials:

```
> dbinom(0:50, 50, 0.5)
```

To plot this distribution, which should resemble a normal distribution:

```
> plot(dbinom(0:50, 50, 0.5))
```

Exercise
Linear modelling example

Mice have varying numbers of babies in each litter. Does the size of the litter affect the average brain weight of the offspring? We can use linear modelling to find out. (This example is taken from John Maindonald and John Braun's book *Data Analysis and Graphics Using R* (CUP, 2003), p140-143.)

1) Install and load the **DAAG** package. The **litters** data frame is part of this package. Take a look at it. How many variables and observations does it have? Does **summary** tell you anything useful? What about **plot**?
2) Are any of the variables correlated? Look up the **cor.test** function and use it to test for relationships.
3) Use **lm** to calculate the regression of brain weight on litter size, brain weight on body weight, and brain weight on litter size and body weight together.
4) Look at the coefficients in your models. How is brain weight related to litter size on its own? What about in the multiple regression? How would you interpret this result?

Linear modelling answers

- To install and load the package and look at **litters**:

```
> install.packages("DAAG")
> library(DAAG)
> litters
> summary(litters)
> plot(litters)
```

- To calculate correlations between variables:

```
> attach(litters)
> cor.test(brainwt, lsize)
> cor.test(bodywt, lsize)
> cor.test(brainwt, bodywt)
```

Linear modelling answers

- To calculate the linear models:

```
> lm(brainwt~lsize)

Call:
lm(formula = brainwt ~ lsize)

Coefficients:
(Intercept)        lsize
   0.447000    -0.004033
```

```
> lm(brainwt~bodywt)

Call:
lm(formula = brainwt ~ bodywt)

Coefficients:
(Intercept)       bodywt
    0.33555      0.01048
```

```
> lm(brainwt~lsize+bodywt)

Call:
lm(formula = brainwt ~ lsize + bodywt)

Coefficients:
(Intercept)        lsize       bodywt
    0.17825      0.00669      0.02431
```

Interpretation: brain weight decreases as litter size increases, but brain weight increases proportional to body weight (when bodywt is held constant, the lsize coefficient is positive – 0.00669). This is called 'brain sparing'; although the offspring get smaller as litter size increases, the brain does not shrink as much as the body.