R Commands for *Introduction to Statistical Modeling*

DANIEL KAPLAN

OCTOBER 5, 2008

This sheet is intended to help you remember R commands and some of the ways they are used. It's assumed that you already understand the statistics and purpose of the commands.
\> marks the command you type.
\+ marks the second line, if any, of the command.

Installing R

Download and execute the "binary" file appropriate for your operating system: Windows, Mac OS X, others.

Starting R

Datasets and convenience functions for the *Introduction to Statistical Modeling* course are contained in the "workspace" file `ISM.Rdata`. Double-click on that file to start a new session of R.

Functions defined in `ISM.Rdata` are: `resample`, `shuffle`, `r.squared`, `do`.

Reading in Spreadsheet/Tabular Data

A data table (called a "data frame" in R) is organized into *cases* and *variables*.

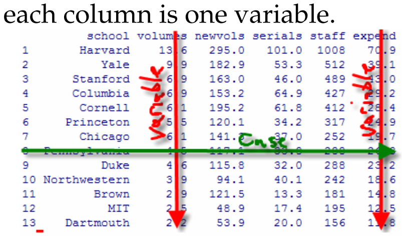
• **Data from the ISM course**

Relevant operators: `ISMdata`. This takes a file name (in quotes) and returns a data frame

```
> kids = ISMdata("kidsfeet.csv")
> runners = ISMdata("ten-mile-race.csv")
```

• **Your own data**

Store your data in a spreadsheet in CSV format. There should be a header row. After that, each row is one case, each column is one variable.



Relevant operators: `read.csv`

```
> mydata = read.csv("fish.csv")
```

Simple Descriptive Statistics

For describing data one variable at a time.

Relevant operators: `mean`, `sd`, `median`, `IQR`, `summary`, `quantile`, `table`, `prop.table`.

There are two basic styles when selecting a variable from a data frame: using `with` or using the `$` reference syntax:

```
> with( kids, mean(width))
[1] 8.992308
> mean( kids$width )
[1] 8.992308
```

Either way is fine. I encourage the `$` method.

• **Quantitative Variables**

```
> sd( kids$width )
[1] 0.5095843
> median( kids$width )
[1] 9
> IQR( kids$width )
[1] 0.7
> quantile( kids$width, 0.60 )
 60%
9.08
> summary( kids$width )
Min   1st Qu.  Median   Mean 3rd Qu.   Max
7.90    8.65    9.00    8.99  9.35    9.80
```

• **Categorical Variables**

Count the number of cases at each level:

```
> table( kids$sex )
 B  G
20 19
```

Convert the count to a proportion.

```
> prop.table(table( kids$sex ))
        B         G
0.5128205 0.4871795
```

Linear Modeling

Constructing linear models.

Relevant operators: `lm`, `r.squared`, `summary`, `anova`

Fit a model. All of these three styles are equivalent, but I recommend the first one:

```
> mod = lm( width ~ length + sex, data=kids)
```

or

```
> mod = with(kids, lm( width ~ length + sex))
```

or even

```
> mod = lm(kids$width~kids$length+kids$sex)
```

Display the coefficients:

```
> mod
Coefficients:
(Intercept)       length          sexG
      3.641        0.221        -0.233
```

• **R-squared**

```
> r.squared(mod)
[1] 0.45954
```

• **Regression table** including standard errors:

```
> summary(mod)
Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)   3.6412     1.2506    2.91   0.0061
length        0.2210     0.0497    4.45    8e-05
sexG         -0.2325     0.1293   -1.80   0.0806
```

• **ANOVA table**

```
> anova(mod)
Analysis of Variance Table

Response: width
            Df Sum Sq Mean Sq  F value   Pr(>F)
(Intercept)  1   3154    3154 21287.88 < 2e-16
length       1      4       4    27.38 7.4e-06
sex          1   0.48    0.48     3.23    0.08
Residuals   36      5    0.15
```

Resampling

Relevant operators: `resample`, `shuffle`, `do`

• **Bootstrapping a Standar Error**

The standard error reflects variability due to sampling.

```
> with( kids, mean(width) )
[1] 8.9923
> trials = do(500)*
+    with( resample(kids), mean(width) )
> sd(trials)
[1] 0.076145
```

For a model:

```
> lm( width~length+sex, data=kids)
(Intercept)       length          sexG
      3.641        0.221        -0.233
> trials = do(500)*
+  lm( width~length+sex, data=resample(kids))
> sd(trials)
(Intercept)       length          sexG
   0.939525     0.036822      0.120385
```

- **Hypothesis Testing**

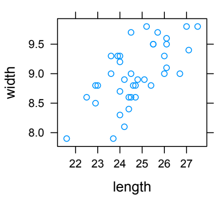
Implement the null hypothesis that sex is not related to width, but length might be:

```
> trials = do(500)*
+   lm( width~length+shuffle(sex), data=kids)
> mean(trials)
  (Intercept)          length shuffle(sex)G
    2.8575581       0.2480365     0.0051818
> sd(trials)
  (Intercept)          length shuffle(sex)G
    0.2228825       0.0085814     0.1326187
```

Graphics

- **Scatter Plot**

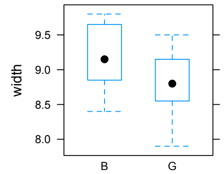
```
> xyplot( width ~ length, data=kids)
```



Try: `xyplot(width~length|sex,data=kids)`

- **Box and Whiskers Plot**

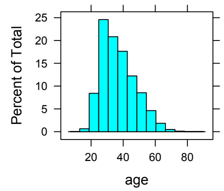
```
> bwplot( width ~ sex, data=kids)
```



- **Histograms**

```
> histogram( ~ age, data=runners)
```

Don't forget the leading tilde (~).

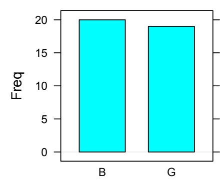


Extra: Try also

```
> histogram( ~ age | sex, data=runners)
```

- **Bar Charts**

```
> barchart( table(kids$sex), horizontal=FALSE)
```



Simulations

Simulations generate data from a hypothetical causal network.

Relevant function: `run.sim`

Available hypothetical causal networks include: `campaign.spending`, `jock`, `university.test`, `heights`, `electro`, `aspirin`, `salaries`, etc.

To see the variables, type the name of the hypothetical causal network

```
> campaign.spending
Causal Network with  4  vars:
==============================
popularity is exogenous
polls <== popularity
spending <== polls
vote <== popularity & spending
```

- **Observational Study**

```
> run.sim(campaign.spending, 4)
  popularity  polls spending   vote
1     44.141 46.875   59.484 42.780
2     25.577 30.991   46.322 18.655
3     47.376 46.616   45.275 52.207
4     48.393 50.003   56.839 54.389
```

- **Experimental Study**

Two types of experiments are supported.

Create the experimental intervention of the desired length:

```
> intervene = rep( c(0,100), length.out = 5)
```

1) Impose the intervention directly.

```
> run.sim( campaign.spending, 4,
+   spending=intervene)
  popularity  polls spending   vote
1     49.931 50.071        0 48.009
2     66.392 69.752      100 74.800
3     73.079 71.553        0 58.396
4     51.752 53.136      100 57.155
```

2) Add the intervention on top of the "natural" values:

```
> run.sim( campaign.spending, 4,
+   spending=intervene, inject=TRUE)
  popularity  polls spending   vote
1     54.129 52.972   59.853 67.638
2     70.974 65.608  126.752 85.580
3     75.549 71.873   34.934 67.062
4     50.391 48.954  160.389 74.577
```