Using R for Educational Research: An Introductory Workshop to Break the Learning Curve

Dr. Kim Nimon[1] Dr. J. Kyle Roberts[2]

[1]University of North Texas

[2]Southern Methodist University

SERA Training Course, February 3, 2012

Table of Contents

Basics of R

- *R* is an Open Source (freely available) environment for statistical computing and graphics.
- Provides data manipulation and display facilities and most statistical procedures. Can be extended with "packages" containing data, code and documentation. Currently over 1500 contributed packages in the Comprehensive R Archive Network (CRAN).
- Beauty of *R* is object oriented language.
- You can get solutions to most of your problems with *R* by typing "RSiteSearch("Your Question")" at the command prompt.
- `http://www.r-project.org/`

Simple calculator usage

- Arithmetic expressions can be typed in the console window. If the expresssion on a line is complete it is evaluated and the result is printed.
- Applies to scalar expressions and to vector expressions.

```
> 2 + 3
[1] 5
> exp(2)
[1] 7.389056
> sin(2 * pi/3)
[1] 0.8660254
> 1:10
 [1]  1  2  3  4  5  6  7  8  9 10
> 5 + 6 * 8
[1] 53
```

Object Oriented Language

- Arithmetic operations and most functions apply to vectors.
- When results are printed the number in square brackets at the beginning of the line is the index of the element at the start of the line.
- The `c` function (concatenate) is a quick way of creating a vector.
- The assignment operator is the two-character sequence `<-`. (The `=` sign can also be used except in a few cases.)

```
> weight <- c(120, 180, 135, 260, 225, 198)
> weight
```

```
[1] 120 180 135 260 225 198
```

```
> height <- c(50, 60, 50, 73, 74, 52)
```

Object Oriented Language

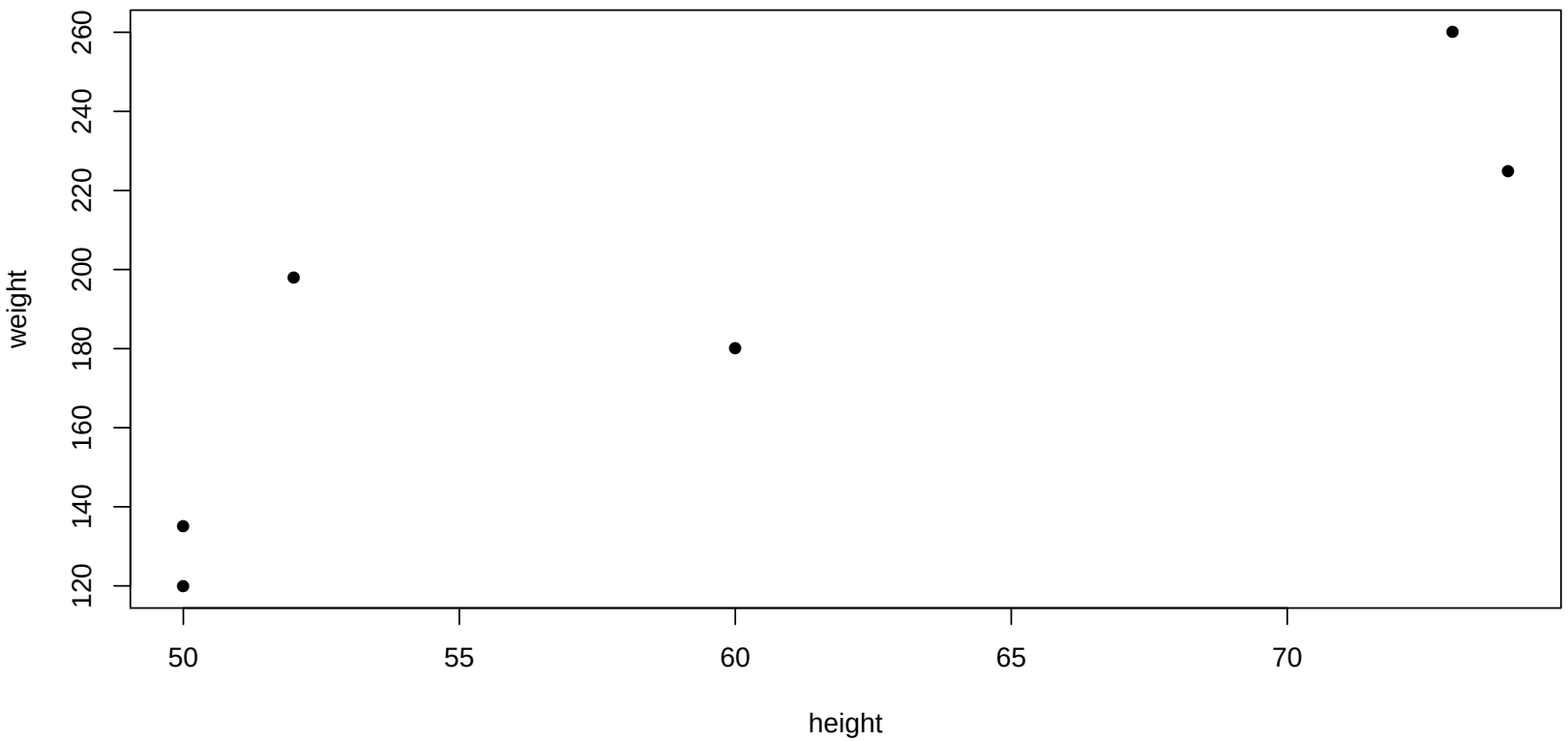
- R recognizes objects as having certain "classes."
- The object `weight` is a numeric object with 6 elements.
- Functions can then be computed with objects as long as they do not differ in size.

```
> str(weight)
 num [1:6] 120 180 135 260 225 198
> weight * 2
[1] 240 360 270 520 450 396
> weight + height
[1] 170 240 185 333 299 250
```

Simple plots

- The `plot` function, given two vectors, produces a simple scatter plot.

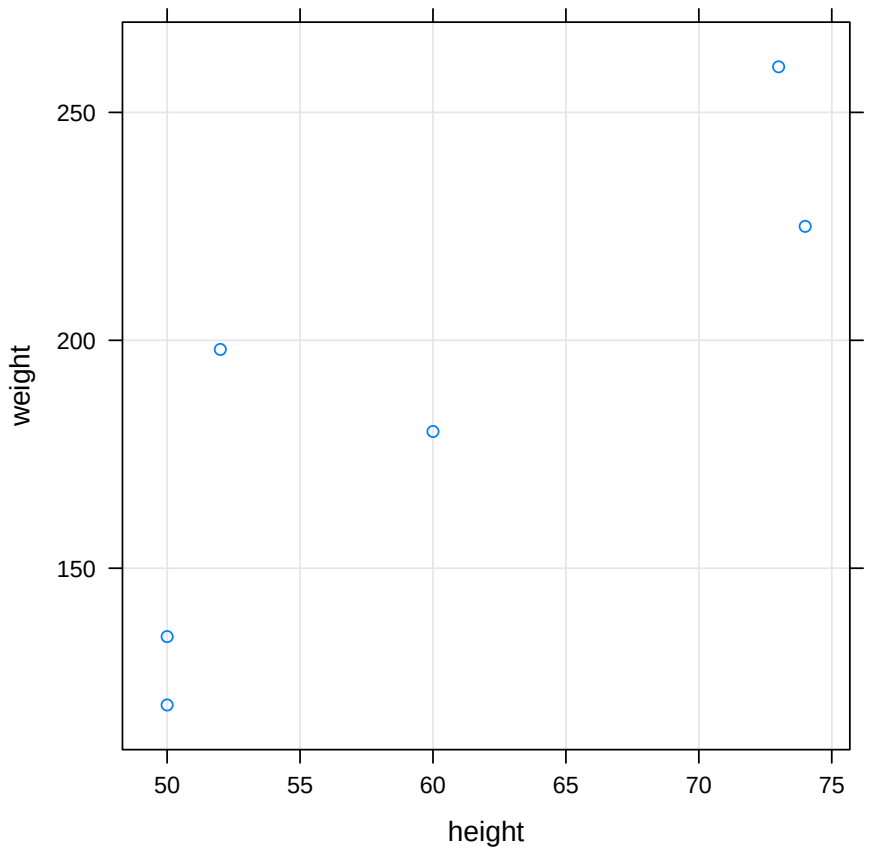
```
> plot(height, weight, pch = 16)
```



`lattice` graphics functions

- The `lattice` package provides higher-level graphics functions, including `xyplot`. (`library(lattice)` for access to them)

```
> library(lattice)
> xyplot(weight ~ height, type = c("p", "g"))
```



Summary functions

- Several functions such as `sum`, `prod`, `mean`, `median`, `var` and `sd`, produce a single summary result from a vector. The `length` function returns the number of elements in the vector, not the geometric length.

```
> sum(weight)
```

```
[1] 1118
```

```
> c(n = length(weight), sum = sum(weight), mean = mean(weight),
+     median = median(weight), variance = var(weight),
+     sd = sd(weight))
```

```
         n        sum       mean     median   variance
   6.00000 1118.00000  186.33333  189.00000 2826.66667
        sd
  53.16641
```

Operations with missing data

- R handles missing data by letting an "NA" be the placeholder for the missing piece.

```
> str(missdata <- c(1:5, NA, 7))

 num [1:7] 1 2 3 4 5 NA 7

> mean(missdata)

[1] NA

> mean(missdata, na.rm = TRUE)

[1] 3.666667
```

- Note that the structure above is different from:

```
> str(missdata2 <- c(1:5, "A", 7))

 chr [1:7] "1" "2" "3" "4" "5" "A" "7"
```

Matrices vs. Data Frames

- Datasets can be imported in either a `matrix` or `dataframe` fashion. We will start with a matrix.

```
> matr <- matrix(1:12, ncol = 4, byrow = TRUE)
> matr
```

```
     [,1] [,2] [,3] [,4]
[1,]    1    2    3    4
[2,]    5    6    7    8
[3,]    9   10   11   12
```

```
> str(matr)
```

```
 int [1:3, 1:4] 1 5 9 2 6 10 3 7 11 4 ...
```

Indexing Parts of the matrix

```
> matr[1, ]
[1] 1 2 3 4
> matr[, 2]
[1]  2  6 10
> matr[2, 4]
[1] 8
```

Matrices vs. Data Frames (cont.)

```
> dfram <- data.frame(y = 1:3, x1 = 4:6, x2 = 7:9,
+     x3 = 10:12)
> dfram
```

```
  y x1 x2 x3
1 1  4  7 10
2 2  5  8 11
3 3  6  9 12
```

```
> str(dfram)
```

```
'data.frame': 3 obs. of  4 variables:
 $ y : int  1 2 3
 $ x1: int  4 5 6
 $ x2: int  7 8 9
 $ x3: int  10 11 12
```

Indexing the Data Frame

```
> dfram$y

[1] 1 2 3

> cbind(dfram$x1, dfram$x3)

     [,1] [,2]
[1,]    4   10
[2,]    5   11
[3,]    6   12

> attach(dfram)
> x2

[1] 7 8 9

> detach(dfram)
```

Table of Contents

Data in Simple Structures

- There are multiple ways to import data into *R*.
- Use the `read.table` command to read in data files that are in ASCII text format.
- With the `read.table` function, we must specify if the variable names are in the first row of data. To do this, we use the command `header=T`.
- You can import data from a data file off of your harddrive, or you can import data directly from web addresses.
- For example, we might import a dataset as:

```
> new.data <- read.table("http://www.site.com/data.txt",
+     header = T)
```

Data in Complex Structures

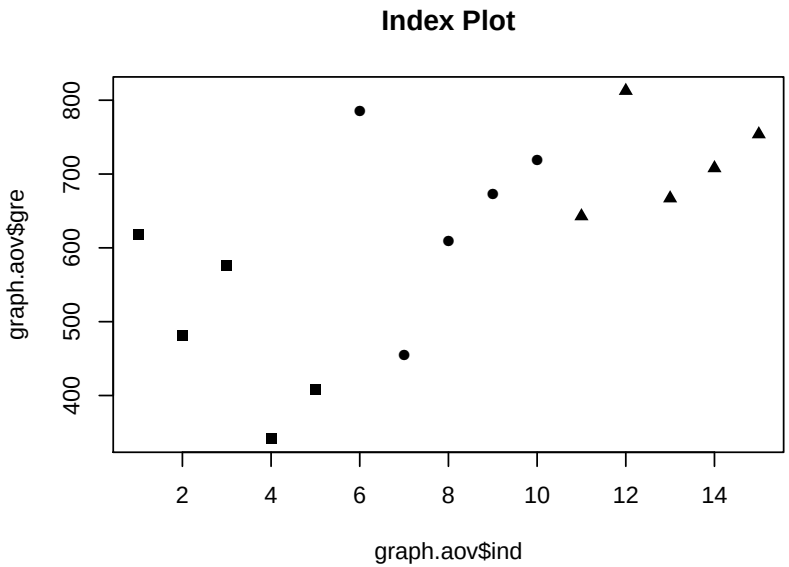
- You can also use the `foreign` package to import datasets from SPSS(`read.spss`), Excel (`read.csv`), SAS (`read.ssd` or `read.xport`), STATA (`read.dta`) and other packages.
- For more documentation on the `foreign` library, see `http://cran.r-project.org/web/packages/foreign/foreign.pdf`

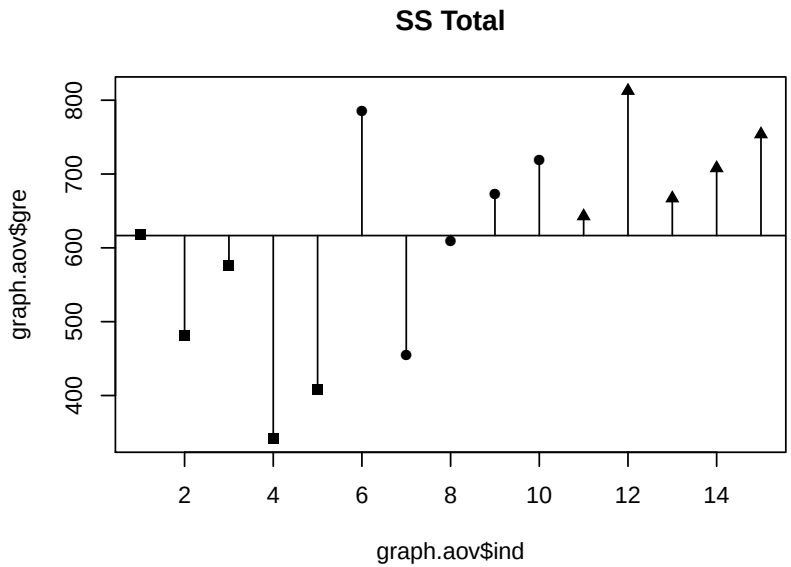
Viewing Data

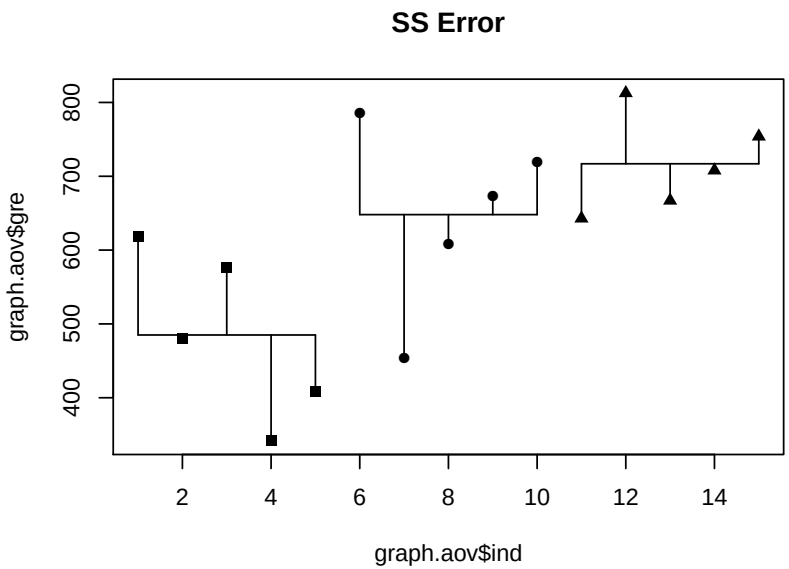
- You can easily view the data by typing the dataset name in the command box.
- If you wish to see the data in a more "Excel-like" format, use the command `edit(dataset)`.
- You can also use the `edit` command to make changes to the dataset, however you must assign it to a new dataset like `new.data<-edit(old.data)`.

Table of Contents

Graphical Representation of Computation of SS







Running ANOVA in R

- Consider the following dataset:
- Ethnicity 1-8,7,6,7,9,11,13
- Ethnicity 2-11,13,14,18,17,14,12,15
- Ethnicity 3-14,13,15,15,20,21,22

```
> ethdata <- data.frame(ethn = factor(rep(1:3, c(7,
+     8, 7))), score = c(8, 7, 6, 7, 9, 11, 13,
+     11, 13, 14, 18, 17, 14, 12, 15, 14, 13, 15,
+     15, 20, 21, 22))
> tapply(ethdata$score, ethdata$ethn, mean)

       1        2        3
 8.714286 14.250000 17.142857

> tapply(ethdata$score, ethdata$ethn, sd)

       1        2        3
2.497618 2.375470 3.716117
```

Assumptions for ANOVA

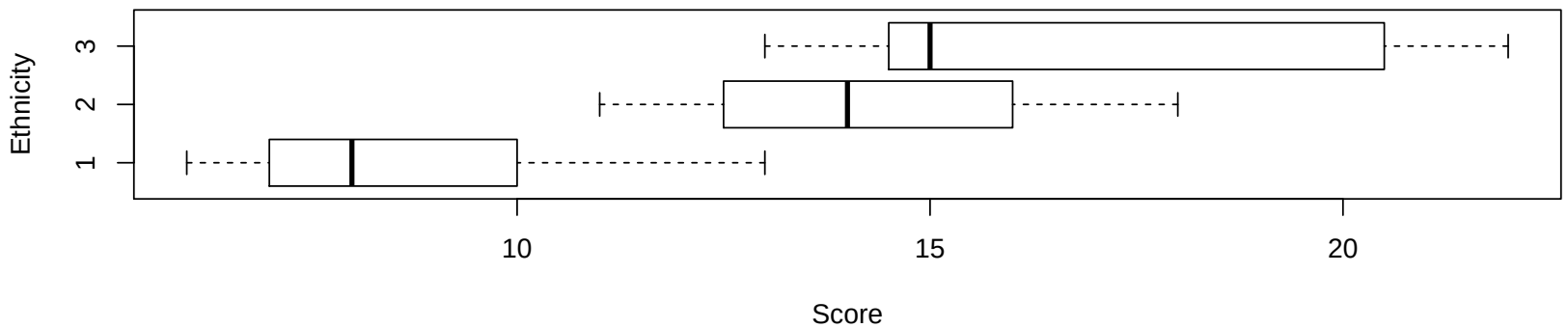
- Relatively the same number of people in each level
- Normality in the population for each of the levels
- Homogeneity of variance

$$H_0 : \sigma_1^2 = \sigma_2^2 = \cdots = \sigma_k^2$$

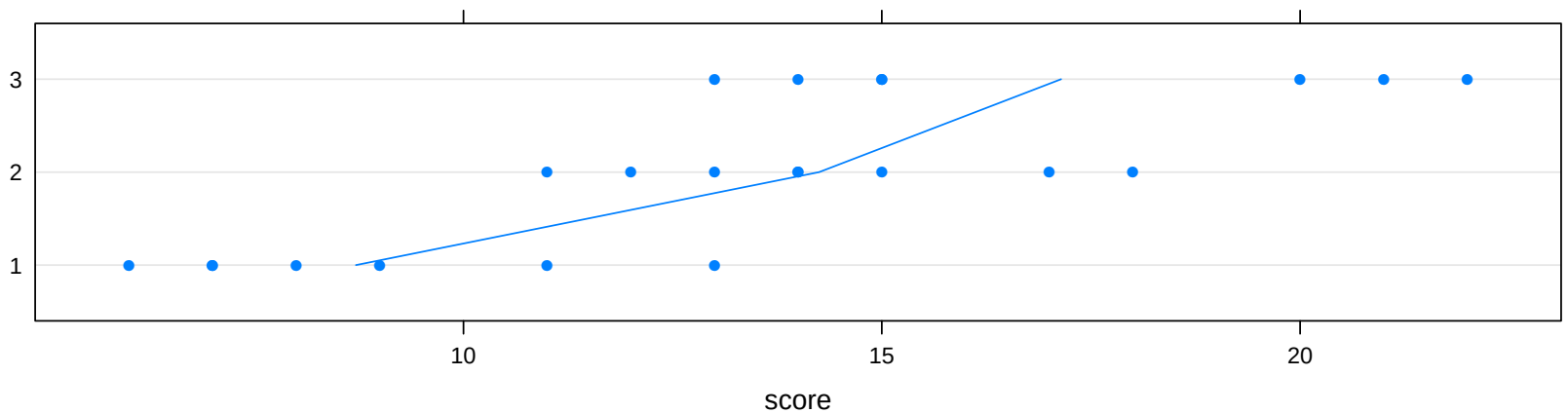
```
> bartlett.test(score ~ ethn, ethdata)

Bartlett test of homogeneity of variances
data:  score by ethn
Bartlett's K-squared = 1.5034, df = 2, p-value =
0.4716
> par(mfrow = c(1, 1))
```

Graphing Ethnicity Data

```
> boxplot(score ~ ethn, ethdata, ylab = "Ethnicity",
+     xlab = "Score", horizontal = TRUE)
```

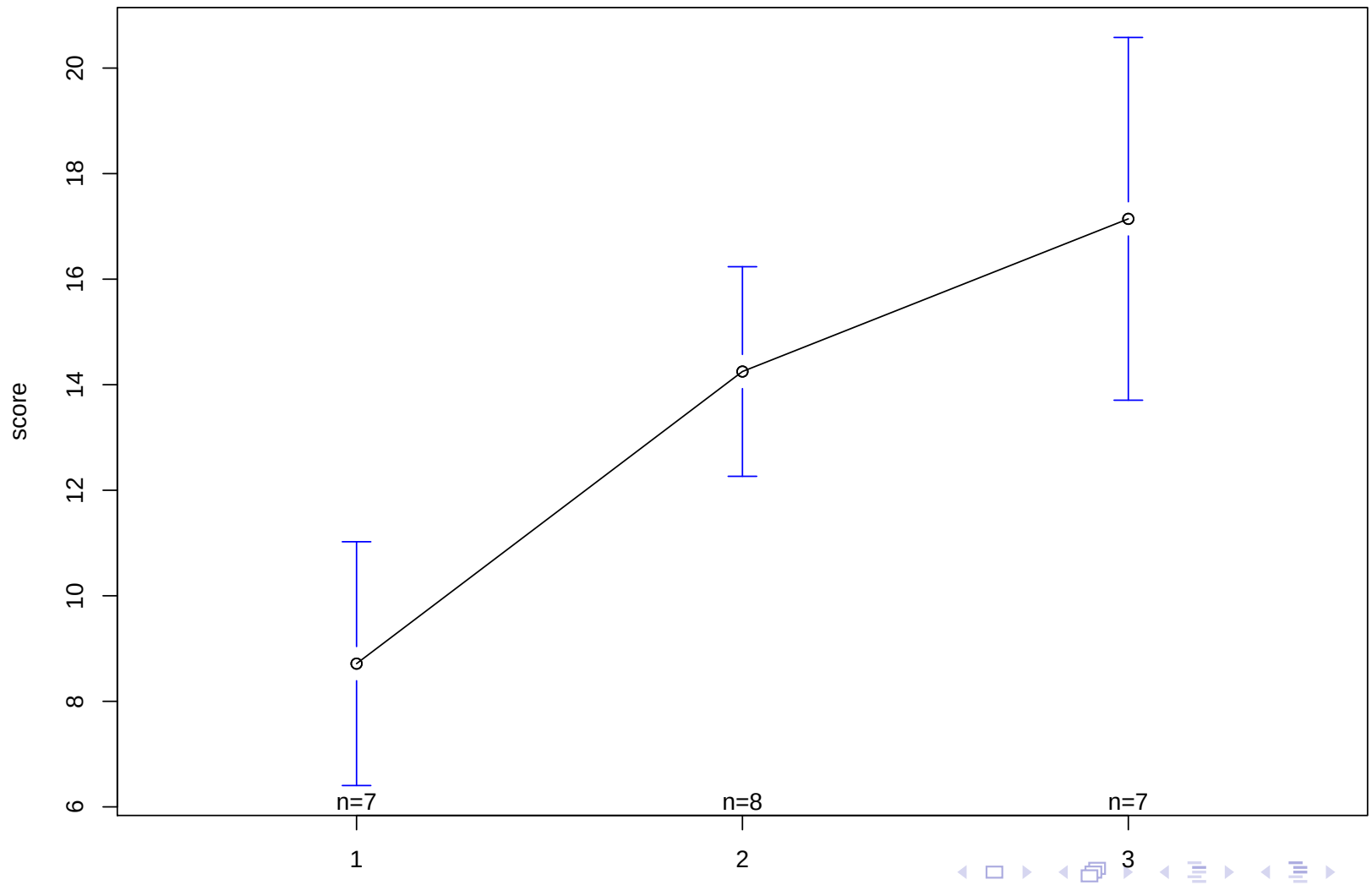



```
> print(dotplot(ethn ~ score, ethdata, type = c("p",
+     "a")))
```



Graphing Ethnicity Data, cont.

```
> library(gregmisc)
> plotmeans(score ~ ethn, ethdata)
```



ANOVA in R cont.

```
> m1 <- aov(score ~ ethn, ethdata)
> anova(m1)

Analysis of Variance Table
Response: score
          Df Sum Sq Mean Sq F value    Pr(>F)
ethn       2 257.53  128.77  15.312 0.0001094
Residuals 19 159.79    8.41
```

- Note that R puts the ANOVA summary table in a slightly different format than we will report. There is no "Total" row and there is no η^2.
- For this case η^2 is computed as $257.53/(257.53 + 159.79) = 0.617$.

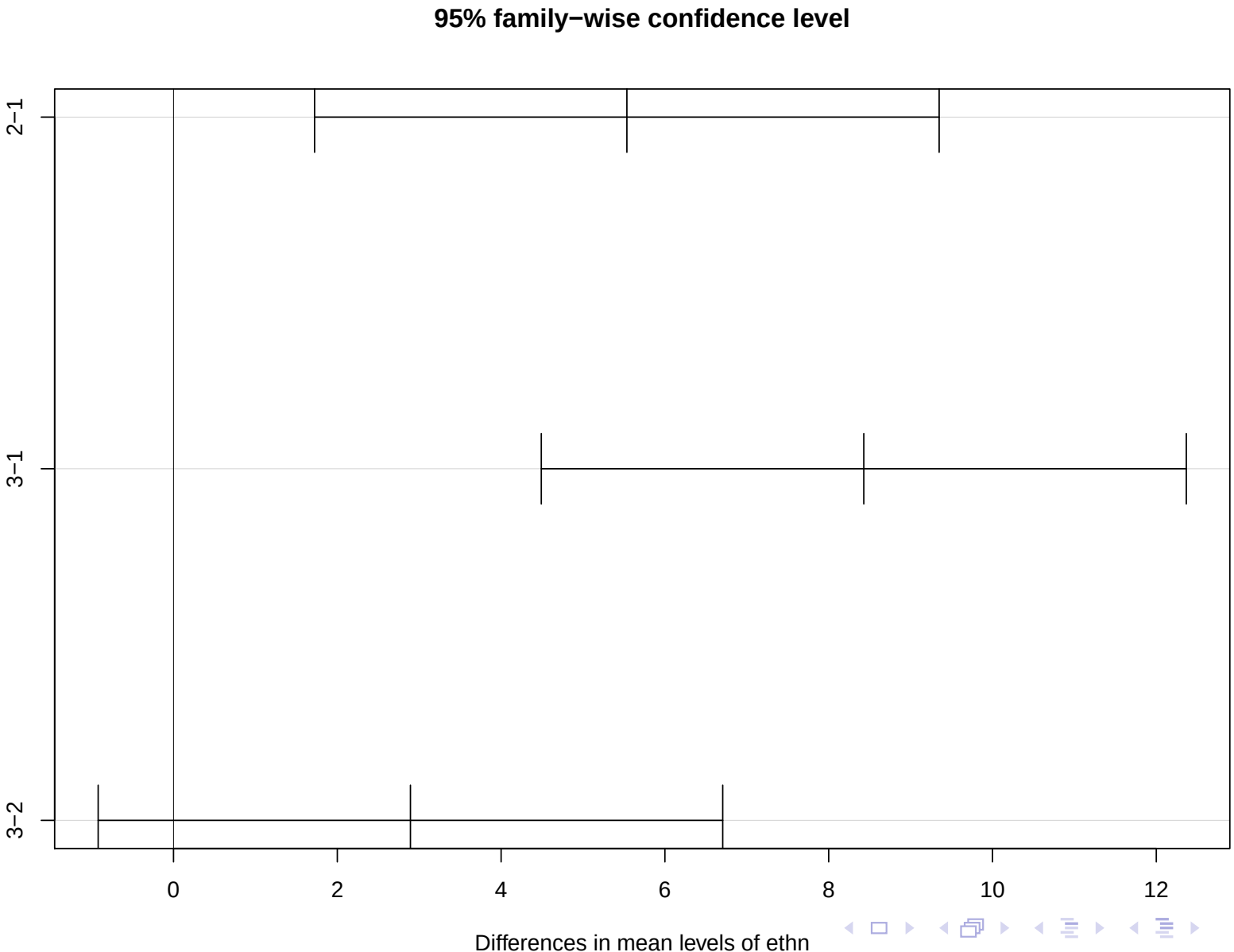
Tukey's HSD for ANOVA

- The Tukey's HSD provides a correction factor to the pairwise comparisons such that the p-value is slightly inflated.
- These adjustments are based on the number of comparisons.

```
> TukeyHSD(m1)

  Tukey multiple comparisons of means
    95% family-wise confidence level
Fit: aov(formula = score ~ ethn, data = ethdata)

$ethn
        diff        lwr       upr     p adj
2-1 5.535714  1.7228228  9.348606 0.0042454
3-1 8.428571  4.4906341 12.366509 0.0000860
3-2 2.892857 -0.9200343  6.705749 0.1582960
```

Plotting Tukey HSD

```
> plot(TukeyHSD(m1))
```



Practice Example with 5 Levels

- Create the following dataset in R.
- Test all assumptions and run all appropriate post-hoc tests.

Program 1	Program 2	Program 3	Program 4	Program 5
30	32	31	43	44
33	35	34	47	50
31	28	33	53	50
25	29	32	54	49
26	19	29	52	47
29	20	30	55	49
29	20	31	45	49
31		31		

Two-Way ANOVA Data

- Read in a table using `read.table` which resides on this website at `http://faculty.smu.edu/kyler/training/sera_r_2012/twoway1.txt`

```
> twoway <- read.table("http://faculty.smu.edu/kyler/training/se
+     header = T)
> head(twoway)
```

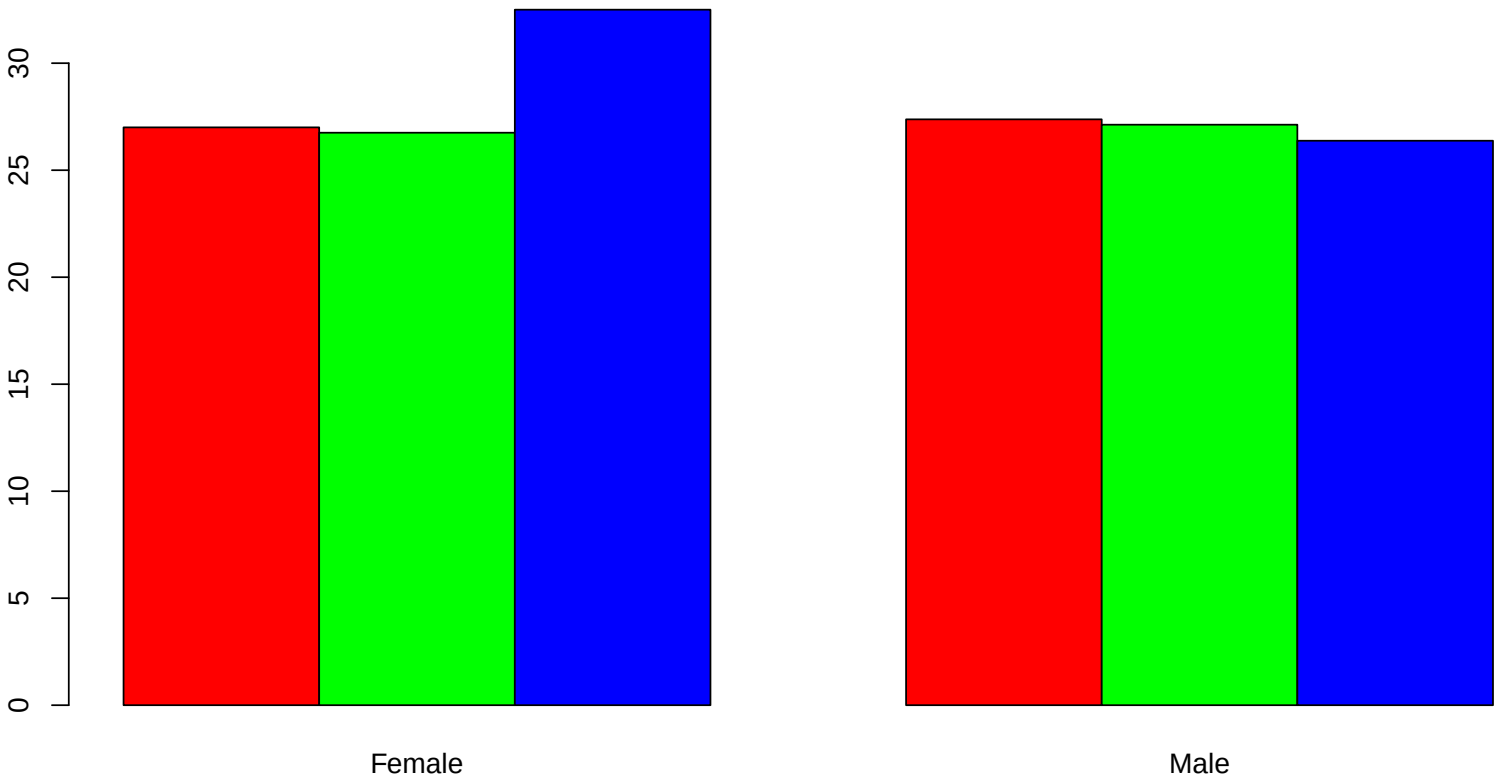
```
  gender program gre gender2 program2
1      2       1  24  Female        A
2      2       1  27  Female        A
3      2       1  33  Female        A
4      2       1  25  Female        A
5      2       1  26  Female        A
6      2       1  30  Female        A
```

Structuring and viewing data

```
> str(twoway)

'data.frame': 48 obs. of  5 variables:
 $ gender  : int  2 2 2 2 2 2 2 2 1 1 ...
 $ program : int  1 1 1 1 1 1 1 1 1 1 ...
 $ gre     : int  24 27 33 25 26 30 22 29 33 26 ...
 $ gender2 : Factor w/ 2 levels "Female","Male": 1 1 1 1 1 1 1 1 2 2 ..
 $ program2: Factor w/ 3 levels "A","B","C": 1 1 1 1 1 1 1 1 1 1 ...

> table(twoway$gender2, twoway$program2)

        A B C
  Female 8 8 8
  Male   8 8 8
```

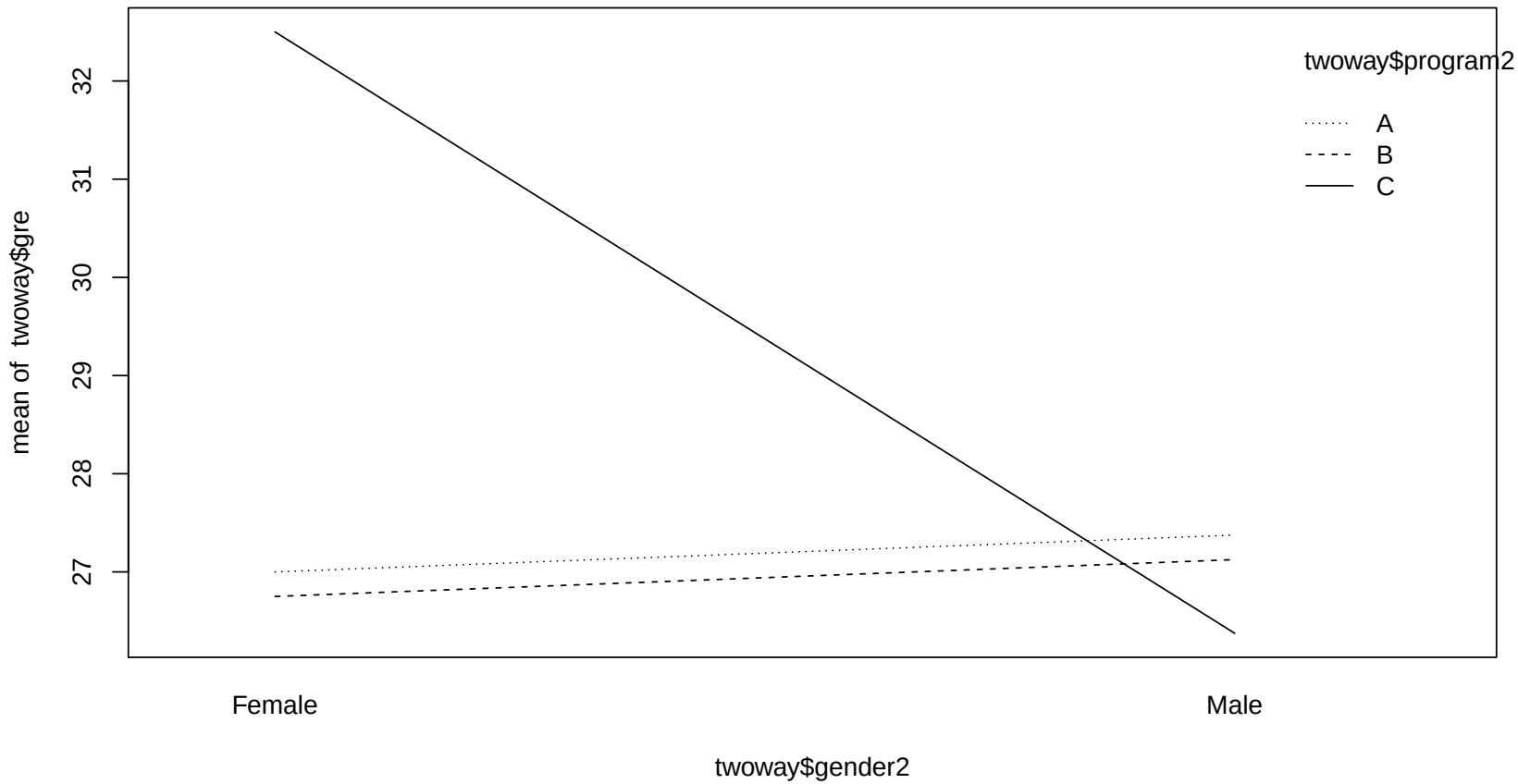
Program vs. Gender

```
> barplot(tapply(twoway$gre, list(twoway$program2,
+     twoway$gender2), mean), beside = T, col = rainbow(3))
```



Interaction Plot

```
> interaction.plot(twoway$gender2, twoway$program2,
+     twoway$gre)
```



Running the Data

```
> m1 <- aov(gre ~ gender2 + program2, twoway)
> summary(m1)
```

```
            Df Sum Sq Mean Sq F value Pr(>F)
gender2      1  38.52  38.521  2.2942 0.1370
program2     2  60.67  30.333  1.8066 0.1762
Residuals   44 738.79  16.791
```

```
> m2 <- aov(gre ~ gender2 * program2, twoway)
> summary(m2)
```

```
                Df Sum Sq Mean Sq F value  Pr(>F)
gender2          1  38.52  38.521  2.5839 0.11544
program2         2  60.67  30.333  2.0347 0.14340
gender2:program2 2 112.67  56.333  3.7788 0.03097
Residuals       42 626.13  14.908
```

Testing Assumptions

```
> bartlett.test(gre ~ gender2 * program2, twoway)

Bartlett test of homogeneity of variances
data:  gre by gender2 by program2
Bartlett's K-squared = 3.865, df = 1, p-value =
0.0493

> bartlett.test(gre ~ program2 * gender2, twoway)

Bartlett test of homogeneity of variances
data:  gre by program2 by gender2
Bartlett's K-squared = 6.5532, df = 2, p-value =
0.03776

> fligner.test(gre ~ gender2 * program2, twoway)

Fligner-Killeen test of homogeneity of variances
data:  gre by gender2 by program2
Fligner-Killeen:med chi-squared = 1.6152, df = 1,
p-value = 0.2038
```

Investigation of Means

```
> model.tables(m2, "means")

Tables of means
Grand mean
27.85417

 gender2
gender2
Female   Male
28.750 26.958

 program2
program2
     A      B      C
27.187 26.938 29.437

 gender2:program2
        program2
gender2  A     B     C
  Female 27.00 26.75 32.50
  Male   27.38 27.12 26.37
```

Post Hoc Tests

```
> TukeyHSD(m2)$"gender2:program2"
```

	diff	lwr	upr	p adj
Male:A-Female:A	0.375	-5.38810348	6.1381035	0.99995875
Female:B-Female:A	-0.250	-6.01310348	5.5131035	0.99999450
Male:B-Female:A	0.125	-5.63810348	5.8881035	0.99999983
Female:C-Female:A	5.500	-0.26310348	11.2631035	0.06898988
Male:C-Female:A	-0.625	-6.38810348	5.1381035	0.99949105
Female:B-Male:A	-0.625	-6.38810348	5.1381035	0.99949105
Male:B-Male:A	-0.250	-6.01310348	5.5131035	0.99999450
Female:C-Male:A	5.125	-0.63810348	10.8881035	0.10654536
Male:C-Male:A	-1.000	-6.76310348	4.7631035	0.99516934
Male:B-Female:B	0.375	-5.38810348	6.1381035	0.99995875
Female:C-Female:B	5.750	-0.01310348	11.5131035	0.05082404
Male:C-Female:B	-0.375	-6.13810348	5.3881035	0.99995875
Female:C-Male:B	5.375	-0.38810348	11.1381035	0.08000676
Male:C-Male:B	-0.750	-6.51310348	5.0131035	0.99876887
Male:C-Female:C	-6.125	-11.88810348	-0.3618965	0.03144884

Table of Contents

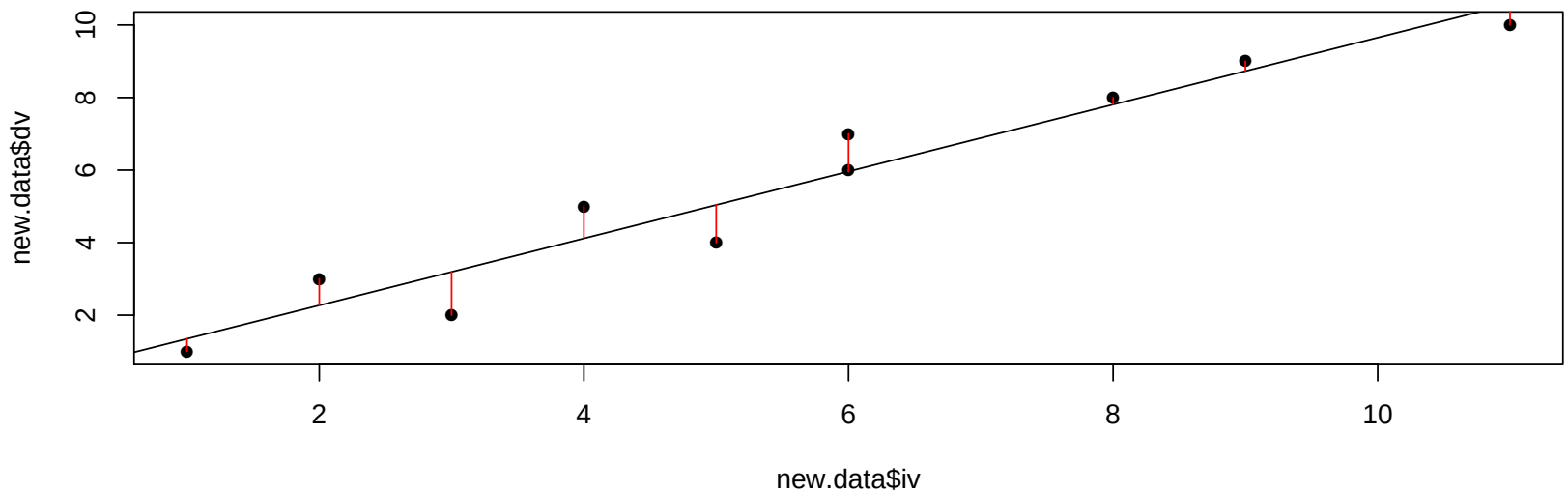
The `lm` Function in R

- In the `lm` function, we specify the dependent variable as modeled by $(\sim)$ the independent variable.

```
> new.data <- data.frame(dv = 1:10, iv = c(1, 3,
+     2, 5, 4, 6, 6, 8, 9, 11))
> summary(m1 <- lm(dv ~ iv, new.data))

Call:
lm(formula = dv ~ iv, data = new.data)
Residuals:
    Min      1Q  Median      3Q     Max
-1.1934 -0.5180  0.1160  0.6146  1.0387

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.42541    0.54433   0.782    0.457
iv           0.92265    0.08683  10.626 5.39e-06

Residual standard error: 0.826 on 8 degrees of freedom
Multiple R-squared: 0.9338, Adjusted R-squared: 0.9256
F-statistic: 112.9 on 1 and 8 DF,  p-value: 5.385e-06
```

Plotting Heuristic Data

```
> plot(new.data$iv, new.data$dv, pch = 16)
> abline(lm(new.data$dv ~ new.data$iv))
> segments(new.data$iv, fitted(m1), new.data$iv,
+     new.data$dv, col = "red")
```



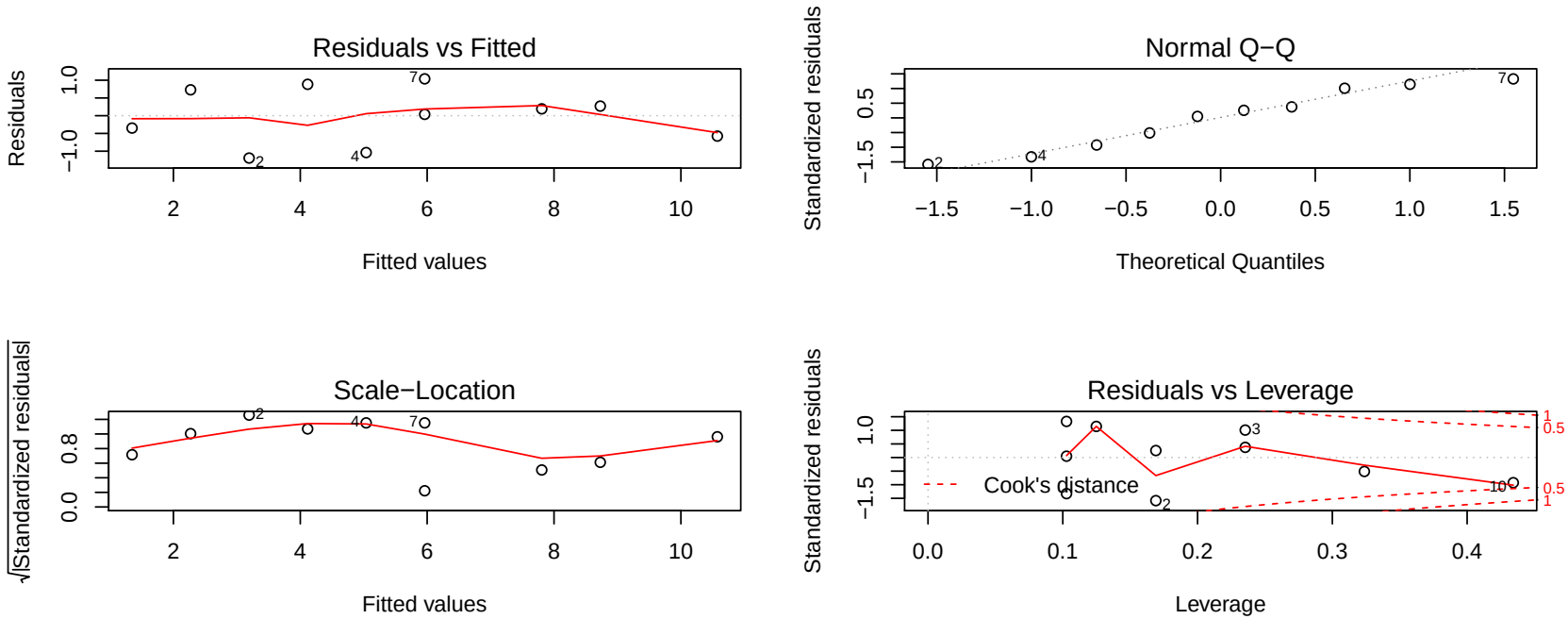
Examining Fitted Values and Residuals

```
> cbind(new.data, fit = m1$fit, resid = m1$resid)
```

	dv	iv	fit	resid
1	1	1	1.348066	-0.34806630
2	2	3	3.193370	-1.19337017
3	3	2	2.270718	0.72928177
4	4	5	5.038674	-1.03867403
5	5	4	4.116022	0.88397790
6	6	6	5.961326	0.03867403
7	7	6	5.961326	1.03867403
8	8	8	7.806630	0.19337017
9	9	9	8.729282	0.27071823
10	10	11	10.574586	-0.57458564

Checking Assumptions of the Linear Model

```
> par(mfrow = c(2, 2))
> plot(m1)
```



Identifying Outliers

- We can look at the influence of each individual variable pairs by looking at their influence on the coefficients (a and b) when they are removed.

```
> influence(m1)$coefficients
```

```
    (Intercept)            iv
1  -0.192232694  0.0255931102
2  -0.361819678  0.0396732103
3   0.298246735 -0.0368856386
4  -0.150940315  0.0063957761
5   0.193091132 -0.0167420057
6   0.003000572  0.0002381406
7   0.080586778  0.0063957761
8  -0.012085635  0.0064285294
9  -0.039903554  0.0136923961
10  0.237914365 -0.0617230663
```

Adding a Second Predictor

- First reconsider our original model for a single predictor where:

$$y_i = a + b * x_i + \epsilon_i \quad i = 1, \ldots, n.$$

and

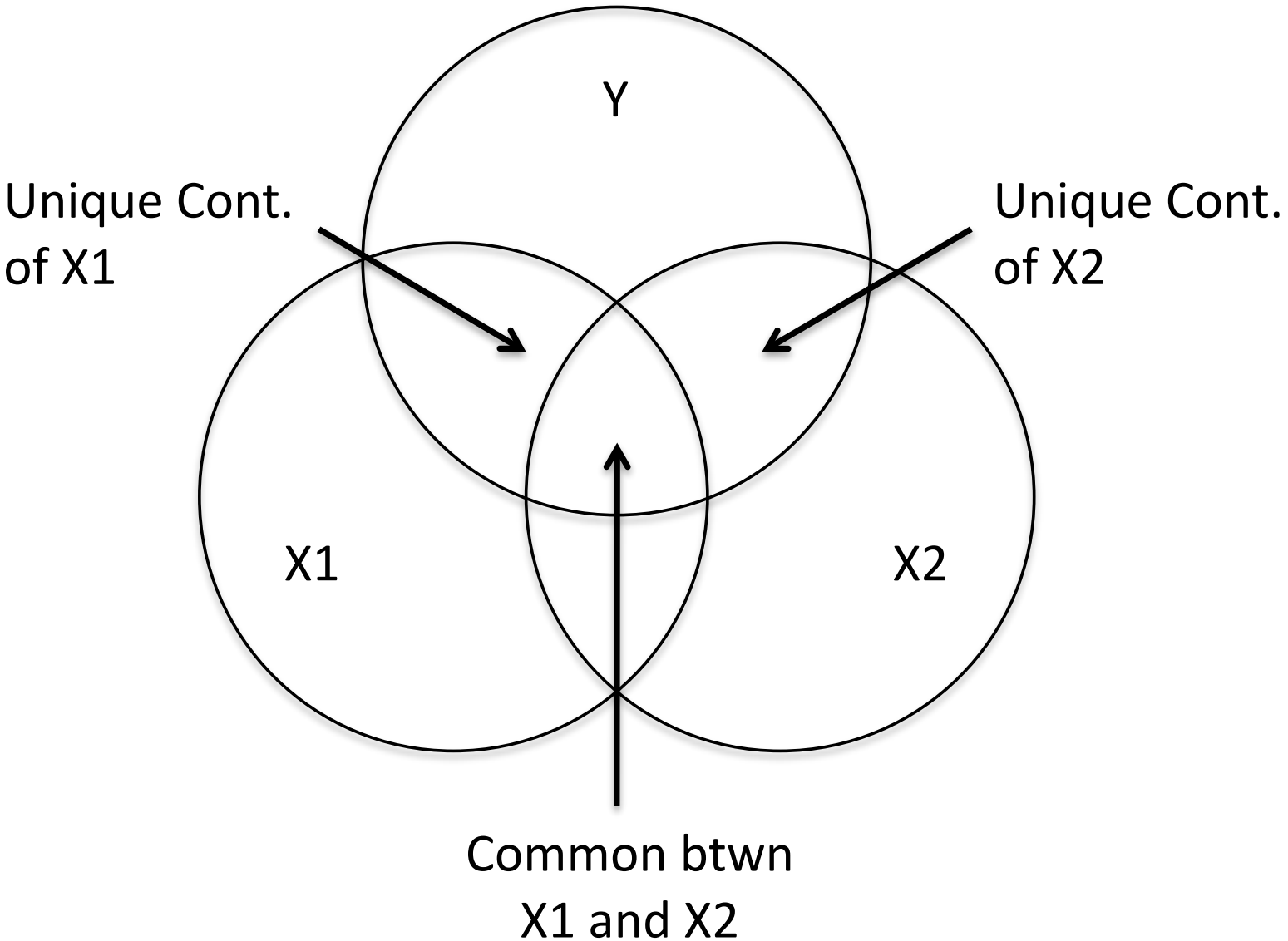
$$\epsilon_i \sim \mathcal{N}(0, \sigma^2)$$

- Were we to add another predictor to the above model, this would change the model to:

$$y_i = a + b_1 * x_1 + b_2 * x_2 + \epsilon_i.$$

where b_1 represents the partial regression coefficient for y on x_1 when x_2 is in the equation and b_2 represents the partial regression coefficient for y on x_2 when x_1 is in the equation.

Graphical Example of Commonality Coefficients



An R Example

- We will be working through the example in the `yhat` library. The dataset we will be looking at is a classic dataset from the Holzinger and Swineford (1939) study which involved 301 subject responses on 26 items.
- In order to access the data, we must first run the following commands:

```
> library(yhat)
> data(HS.data)
```

- The total output from the `regr` command is too large to look at in these slides, so we will look at specific pieces in the next few slides.

The yhat Library

```
> reg1 <- lm(deduct ~ numeric + arithmet, HS.data)
> regr(reg1)$LM_Output

Call:
lm(formula = deduct ~ numeric + arithmet, data = HS.data)
Residuals:
    Min      1Q  Median      3Q     Max
-51.476 -11.278  -1.726   8.845  55.349

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  -8.6450     5.1550  -1.677  0.09459
numeric       1.2723     0.2403   5.294 2.32e-07
arithmet      0.7314     0.2318   3.155  0.00177

Residual standard error: 17.1 on 298 degrees of freedom
Multiple R-squared: 0.1848, Adjusted R-squared: 0.1793
F-statistic: 33.78 on 2 and 298 DF,  p-value: 5.994e-14
```

More from regr

```
> regr(reg1)$Beta_Weights

  numeric  arithmet
0.3116765 0.1857667

> regr(reg1)$Structure_Coefficients

       numeric  arithmet
[1,] 0.9233733 0.7649353

> regr(reg1)$Commonality_Data

$CC
                                    Coefficient     % Total
Unique to numeric                        0.0767       41.49
Unique to arithmet                       0.0272       14.74
Common to numeric, and arithmet          0.0809       43.77
Total                                    0.1848      100.00
$CCTotalbyVar
         Unique Common  Total
numeric  0.0767 0.0809 0.1576
arithmet 0.0272 0.0809 0.1081
```

More from `regr`

```
> regr(reg1)$Effect_Size
```

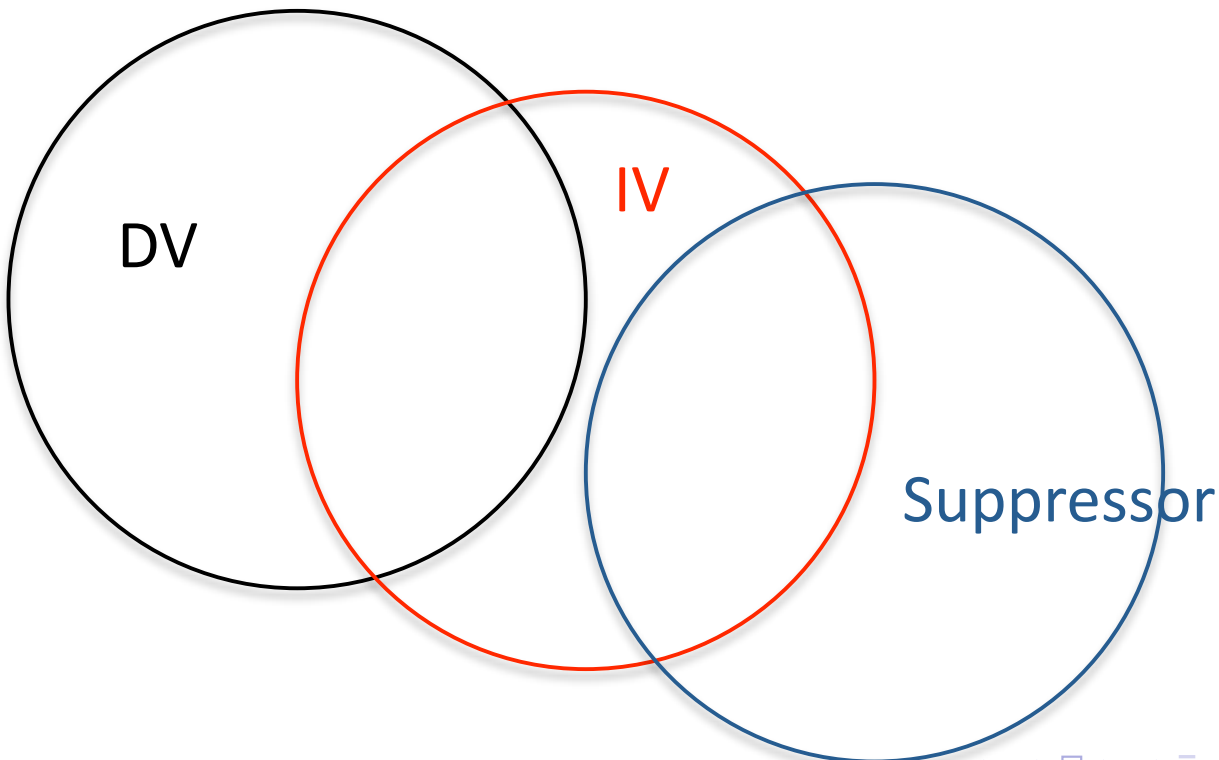
	Effect.Size	Recommended
Wherry1	0.1766	No
Claudy3	0.1804	Yes
Smith	0.1766	No
Wherry2	0.1793	No
Olkin & Pratt	0.1776	No
Pratt	0.1776	No

```
> regr(reg1)$Comment
```

```
[1] "The Effect Size recommendations are based on Yin and Fan (2001). Y
```

Thinking Graphically About Suppressors

- A suppressor effect occurs when a variable has a non-zero β weight but a zero structure coefficient.
- The inclusion of a suppressor in a regression equation removes the unwanted variance from the predictor variable, thus enhancing the relationship between the other independent variable and the dependent variable.



Illustrating Suppressors in R

```
> library(MASS)
> correlation <- matrix(c(1, 0.5, 0, 0.5, 1, 0.5,
+     0, 0.5, 1), ncol = 3, nrow = 3, dimnames = list(c("dv",
+     "iv1", "iv2"), c("dv", "iv1", "iv2")))
> correlation
```

```
     dv iv1 iv2
dv  1.0 0.5 0.0
iv1 0.5 1.0 0.5
iv2 0.0 0.5 1.0
```

```
> set.seed(12346)
> suppressor.set <- data.frame(mvrnorm(n = 1000,
+     rep(10, 3), correlation))
```

Illustration cont.

```
> reg2 <- lm(dv ~ iv1 + iv2, suppressor.set)
> summary(reg2)

Call:
lm(formula = dv ~ iv1 + iv2, data = suppressor.set)
Residuals:
     Min       1Q   Median       3Q      Max
-2.75190 -0.54804  0.03108  0.53789  2.93339

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  6.00115    0.29836  20.114   <2e-16
iv1          0.67842    0.03044  22.285   <2e-16
iv2         -0.28112    0.03015  -9.323   <2e-16

Residual standard error: 0.8217 on 997 degrees of freedom
Multiple R-squared: 0.3356, Adjusted R-squared: 0.3343
F-statistic: 251.8 on 2 and 997 DF,  p-value: < 2.2e-16
```

Illustration cont.

```
> library(yhat)
> regr(reg2)$Beta_Weights
```

```
       iv1        iv2
 0.6733602 -0.2816942
```

```
> regr(reg2)$Structure_Coefficients
```

```
           iv1       iv2
[1,] 0.9096263 0.1178144
```

```
> regr(reg2)$Commonality_Data$CC
```

	Coefficient	% Total
Unique to iv1	0.3310	98.61
Unique to iv2	0.0579	17.26
Common to iv1, and iv2	-0.0533	-15.87
Total	0.3356	100.00

Table of Contents

Null Hypothesis for MANOVA

- We could test to see if the vector of means of the dependent variables is equal for multiple independent groups and our new null would be:

$$H_0 : \begin{bmatrix} \overline{X}_{11} \\ \overline{X}_{21} \\ \vdots \\ \overline{X}_{p1} \end{bmatrix} = \begin{bmatrix} \overline{X}_{12} \\ \overline{X}_{22} \\ \vdots \\ \overline{X}_{p2} \end{bmatrix} = \begin{bmatrix} \overline{X}_{13} \\ \overline{X}_{23} \\ \vdots \\ \overline{X}_{p3} \end{bmatrix} = \cdots = \begin{bmatrix} \overline{X}_{1k} \\ \overline{X}_{2k} \\ \vdots \\ \overline{X}_{pk} \end{bmatrix}$$

- where p represents the total number of dependent variables for k levels.

Example from Stevens (2002) p. 213

- Create the following dataset in R where there are 2 dependent variables and three groups.

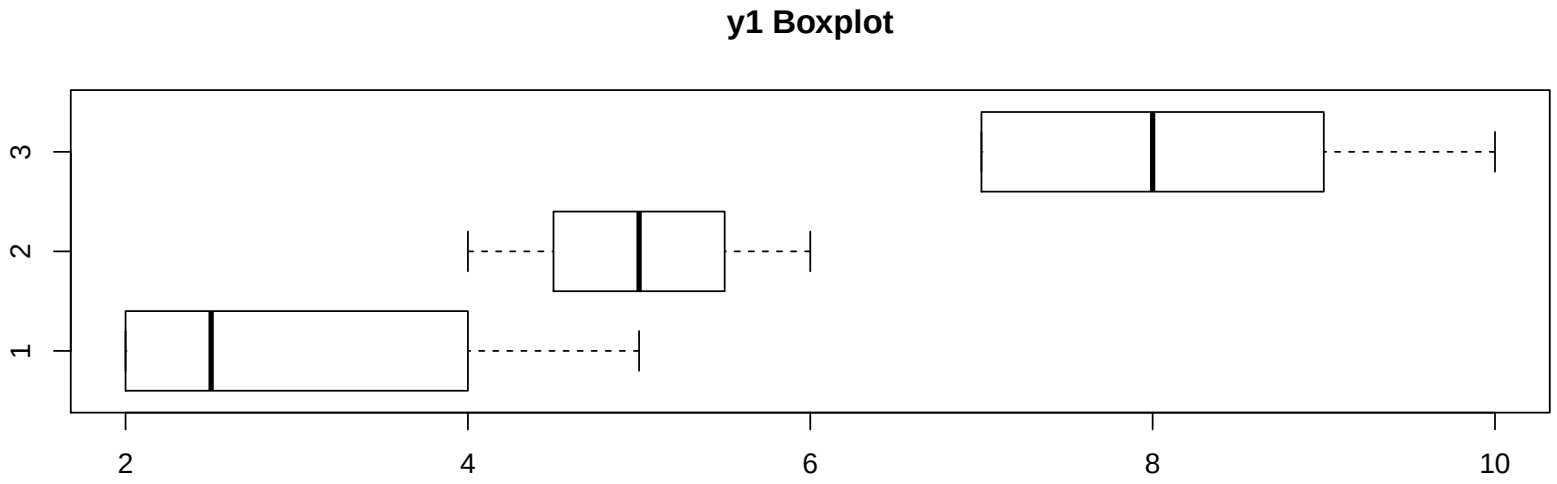
K_1		K_2		K_3	
y_1	y_2	y_1	y_2	y_1	y_2
2	3	4	8	7	6
3	4	5	6	8	7
5	4	5	7	10	8
2	5			9	5
				7	6
$\overline{y}_{11} = 3$	$\overline{y}_{21} = 4$	$\overline{y}_{12} = 5$	$\overline{y}_{22} = 7$	$\overline{y}_{13} = 8.2$	$\overline{y}_{13} = 6.4$

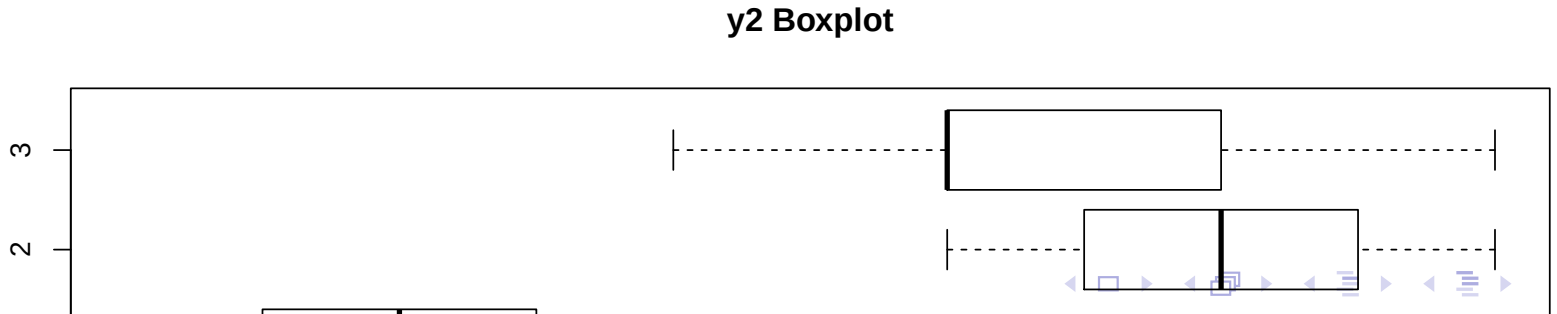
Setting Up the Data in R

```
> manova.data <- data.frame(group = as.factor(rep(1:3,
+     c(4, 3, 5))), y1 = c(2, 3, 5, 2, 4, 5, 6,
+     7, 8, 10, 9, 7), y2 = c(3, 4, 4, 5, 8, 6,
+     7, 6, 7, 8, 5, 6))
> with(manova.data, tapply(y1, group, mean))

  1   2   3
3.0 5.0 8.2

> with(manova.data, tapply(y2, group, mean))

  1   2   3
4.0 7.0 6.4
```

Graphical Analysis of manova.data

```
> par(mfrow = c(2, 1))
> boxplot(y1 ~ group, manova.data, main = "y1 Boxplot",
+     horizontal = T)
> boxplot(y2 ~ group, manova.data, main = "y2 Boxplot",
+     horizontal = T)
```





Running the MANOVA in R

```
> (m1 <- manova(cbind(y1, y2) ~ group, manova.data))

Call:
   manova(cbind(y1, y2) ~ group, manova.data)
Terms:
                    group Residuals
resp 1           61.86667  14.80000
resp 2              19.05      9.20
Deg. of Freedom         2         9

Residual standard error: 1.282359 1.01105
Estimated effects may be unbalanced

> summary(m1, test = "Wilks")

          Df    Wilks approx F num Df den Df    Pr(>F)
group      2 0.089674   9.3575      4     16 0.0004271
Residuals  9
```

Table of Contents

CCA Setup in R

- CCA is used when a researcher wants to look at the relationship between two meaningful sets of variables.
- In order to do this in `R`, we will use both the `yacca` and `yhat` libraries.

```
> library(yacca)
> library(yhat)
```

- We previously loaded the `yhat` library, but it may be necessary to load it again if you have closed R and then re-launched it.

Data Setup for CCA

- We first create two canonical variable sets using again the Holzinger and Swineford dataset `HS.data`.

```
> MATH_REASON <- HS.data[, c("deduct", "problemr")]
> MATH_FUND <- HS.data[, c("numeric", "arithmet",
+     "addition")]
```

- Then we can perform our canonical correlation analysis using the default parameters using the function `cca`.

```
> canon <- cca(MATH_FUND, MATH_REASON)
> canons <- cca(MATH_FUND, MATH_REASON, xcenter = T,
+     ycenter = T, xscale = T, yscale = T, standardize.scores =
```

- We will see part of the output in the next slide.

CCA Output

```
> canon$lambda <- exp(-canon$chisq/(nrow(MATH_FUND) -
+     1 - 0.5 * (ncol(MATH_FUND) + ncol(MATH_REASON) +
+     1)))
> canon$lambda

     CV 1      CV 2
0.6983461 0.9921095

> canon$chisq

      CV 1       CV 2
106.635002   2.352762

> canon$df

CV 1 CV 2
   6    2

> canon$corr

      CV 1       CV 2
0.54415050 0.08882828
```

```
> canon$corrsq
      CV 1       CV 2
0.296099772 0.007890463

> canon$xcoef
                CV 1        CV 2
numeric   -0.14588540 -0.19884621
arithmet  -0.12569673  0.19298681
addition   0.01325338  0.01094449

> canons$xcoef
                CV 1       CV 2
numeric   -0.6745010 -0.9193653
arithmet  -0.6025274  0.9250825
addition   0.3316284  0.2738550

> canon$xstructcorr
                CV 1       CV 2
numeric   -0.8343516 -0.3983097
arithmet  -0.7929838  0.6014613
addition  -0.1223222  0.2826557
```

```
> canon$xstructcorrsq
```

```
              CV 1       CV 2
numeric  0.69614268 0.15865058
arithmet 0.62882330 0.36175564
addition 0.01496271 0.07989422
```

```
> canon$ycoef
```

```
               CV 1        CV 2
deduct   -0.03139923 -0.04833813
problemr -0.06548871  0.09781825
```

```
> canons$ycoef
```

```
              CV 1       CV 2
deduct   -0.5926241 -0.9123263
problemr -0.6052307  0.9040124
```

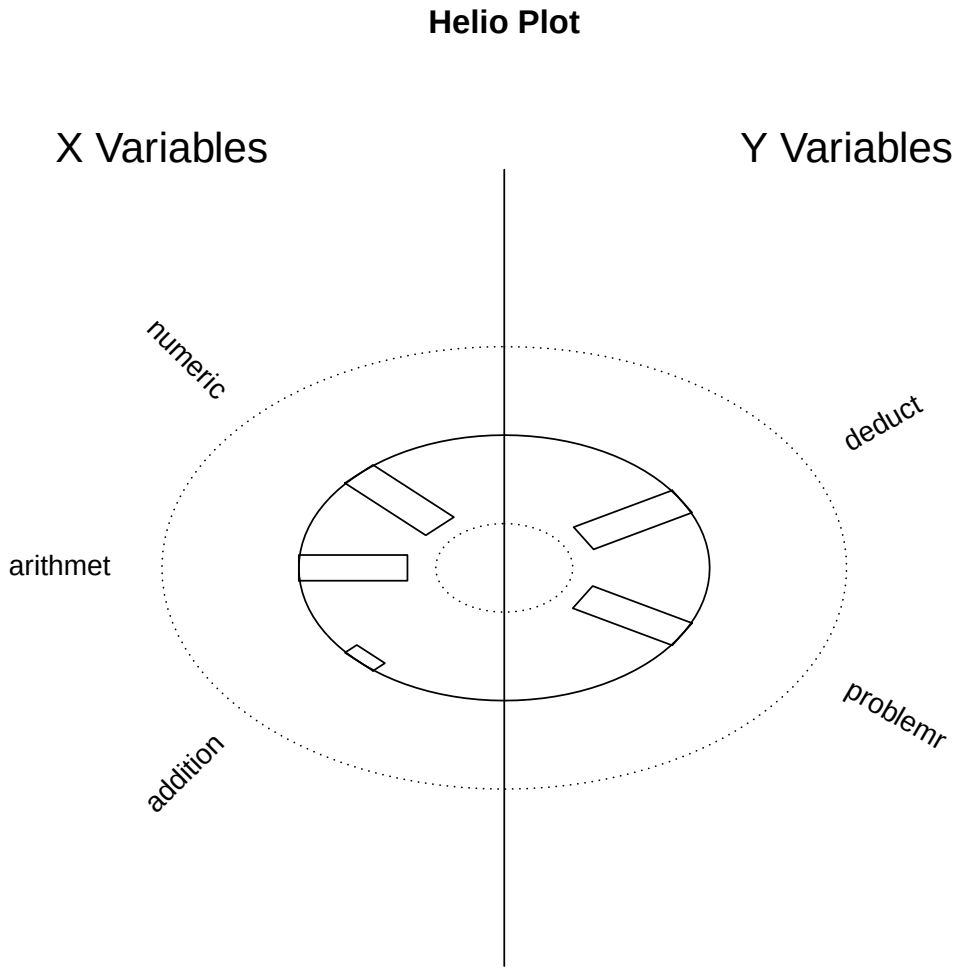
```
> canon$ystructcorr
```

```
              CV 1       CV 2
deduct   -0.8309644 -0.5563255
problemr -0.8386065  0.5447376
```

```
> canon$ystructcorrsq

              CV 1      CV 2
deduct    0.6905019 0.3094981
problemr 0.7032609 0.2967391
```

Helio Plots for the First Canonical Variate

```
> helio.plot(canon)
```



Canonical Commonality Analysis

- After loading the `yhat` library, we can now run a commonality analysis on our canonical data.
- This analysis can only look at one canonical variate (function) at a time. Therefore, we specify the first variate with the command below.

```
> canonCommonData <- canonCommonality(MATH_FUND,
+     MATH_REASON, 1)
```

Canonical Commonality Output

```
> canonCommonData[1]

[[1]]
[[1]][[1]]
[[1]][[1]]$CC
                                  Coefficient     % Total
Unique to deduct                       0.0879       29.67
Unique to problemr                     0.0916       30.95
Common to deduct, and problemr         0.1166       39.38
Total                                  0.2961      100.00
[[1]][[1]]$CCTotalbyVar
          Unique Common  Total
deduct    0.0879 0.1166 0.2045
problemr 0.0916 0.1166 0.2082
```

Canonical Commonality Output, cont.

```
> canonCommonData[2]

[[1]]
[[1]][[1]]
[[1]][[1]]$CC
```

	Coefficient
Unique to numeric	0.1009
Unique to arithmet	0.0800
Unique to addition	0.0269
Common to numeric, and arithmet	0.1107
Common to numeric, and addition	-0.0179
Common to arithmet, and addition	-0.0170
Common to numeric, arithmet, and addition	0.0124
Total	0.2961
	% Total
Unique to numeric	34.08
Unique to arithmet	27.03
Unique to addition	9.09
Common to numeric, and arithmet	37.39
Common to numeric, and addition	-6.05
Common to arithmet, and addition	-5.74
Common to numeric, arithmet, and addition	4.20

Table of Contents

Heuristic Example of Multilevel ANOVA

- Suppose that we have a dataset in which we have scores from 160 students nested inside 16 different schools.
- The dataset may be found at `http://faculty.smu.edu/kyler/training/sera_r_2012/sciach.txt`

```
> sciach <- read.table("http://faculty.smu.edu/kyler/training/se
+     header = T)
> str(sciach)

'data.frame': 160 obs. of  10 variables:
 $ ID          : int  1 2 3 4 5 6 7 8 9 10 ...
 $ GROUP       : int  1 1 1 1 1 1 1 1 1 1 ...
 $ SCIENCE     : int  1 1 2 2 3 3 4 4 5 5 ...
 $ URBAN       : int  8 7 7 6 6 5 5 5 3 2 ...
 $ GENDER      : int  1 1 1 1 1 2 2 2 2 2 ...
 $ CONS        : int  1 1 1 1 1 1 1 1 1 1 ...
 $ URB.MEAN    : num  -6.43 -7.43 -7.43 -8.43 -8.43 ...
 $ SCH.RES     : num  5 5 5 5 5 5 5 5 5 5 ...
 $ SCH.RES.MEAN: num  -4.97 -4.97 -4.97 -4.97 -4.97 ...
 $ GEND.FAC    : Factor w/ 2 levels "Female","Male": 2 2 2 2 2 1 1 1 1

> sciach$GROUP <- factor(sciach$GROUP)
```

The Multilevel ANOVA

- Recall from the multilevel ANOVA notation that we want to test:

$$Y_{ij} = \gamma_{00} + u_{0j} + e_{ij}$$

```
> library(lme4)
> (lmer0 <- lmer(SCIENCE ~ 1 + (1 | GROUP), sciach))

Linear mixed model fit by REML
Formula: SCIENCE ~ 1 + (1 | GROUP)
   Data: sciach
   AIC   BIC logLik deviance REMLdev
 643.9 653.1 -318.9    640.2   637.9
Random effects:
 Groups   Name        Variance Std.Dev.
 GROUP    (Intercept) 25.5313  5.0528
 Residual              1.9792  1.4068
Number of obs: 160, groups: GROUP, 16
Fixed effects:
            Estimate Std. Error t value
(Intercept)   10.687      1.268   8.429
```

Single Predictor with Random Effects for the Intercept

- We may modify our previous model to include one covariate:

$$y_{ij} = \gamma_{00} + \gamma_{10} x_{ij} + u_{0j} + \epsilon_{ij}$$

where:

- y_{ij} - is the response variable for individual i in group j
- γ_{00} - the y-intercept or the expected value when the covariate is 0
- γ_{10} - the expected change in the response variable (y) for every one unit change in the covariate
- x_{ij} - the covariate term for each individual; the subscripts i and j mean that this variable is measured at the first level
- u_{0j} - the residual term defining the random variation of each of the group intercepts around the grand intercept γ_{00}
- ϵ_{ij} - the residual term defining the random variation of each person around their predicted group regression equation

Breaking Down the Mixed Effects Model into Levels

- From the previous slide, our mixed effects model:

$$y_{ij} = \gamma_{00} + \gamma_{10} x_{ij} + u_{0j} + \epsilon_{ij}$$

may be thought of as a 2-level model where:

- Level 1:

$$y_{ij} = \beta_{0j} + \beta_{1j} x_{ij} + \epsilon_{ij}$$

- Level 2:

$$\beta_{0j} = \gamma_{00} + u_{0j}$$
$$\beta_{1j} = \gamma_{10}$$

- where:

$$u_{0j} \sim \mathcal{N}(0, \sigma^2_{u_{0j}})$$
$$\epsilon_{ij} \sim \mathcal{N}(0, \sigma^2_{\epsilon_{ij}})$$

Running the Multilevel Model in R

```
> (lmer1 <- lmer(SCIENCE ~ URBAN + (1 | GROUP),
+     sciach))

Linear mixed model fit by REML
Formula: SCIENCE ~ URBAN + (1 | GROUP)
   Data: sciach
   AIC   BIC logLik deviance REMLdev
 508.1 520.4 -250.0    499.4   500.1
Random effects:
 Groups   Name        Variance Std.Dev.
 GROUP    (Intercept) 86.45599 9.29817
 Residual              0.65521 0.80945
Number of obs: 160, groups: GROUP, 16
Fixed effects:
            Estimate Std. Error t value
(Intercept)  22.3029     2.4262   9.193
URBAN        -0.8052     0.0480 -16.776

Correlation of Fixed Effects:
      (Intr)
URBAN -0.285
```

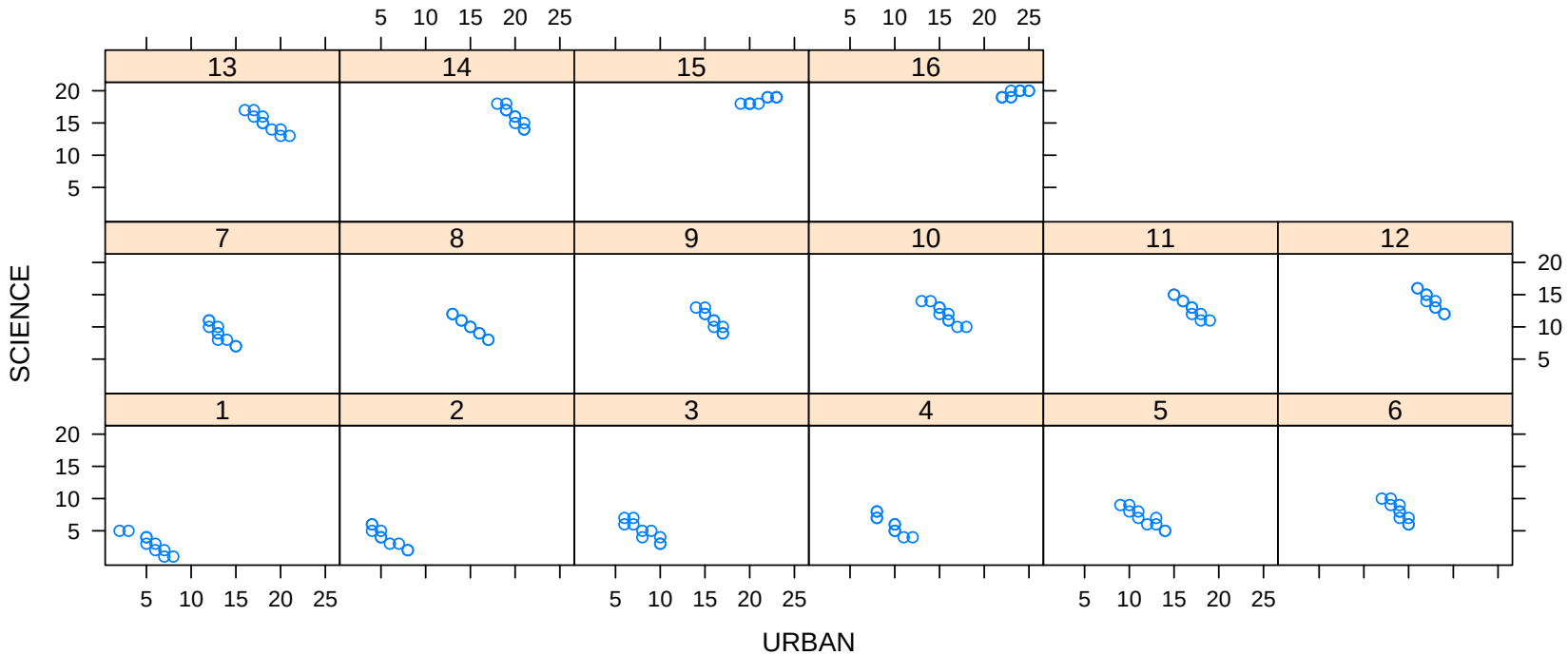
Comparing Models

```
> anova(lmer0, lmer1)

Data: sciach
Models:
lmer0: SCIENCE ~ 1 + (1 | GROUP)
lmer1: SCIENCE ~ URBAN + (1 | GROUP)
      Df    AIC    BIC  logLik  Chisq Chi Df Pr(>Chisq)
lmer0  3 646.17 655.39 -320.08
lmer1  4 507.37 519.67 -249.69 140.79      1  < 2.2e-16
```

- So we can see that by the addition of a single fixed effect to our model, we reduced the AIC by $\sim$ 139 and the BIC by $\sim$ 135.

Model Estimates (cont.)

```
> print(xyplot(SCIENCE ~ URBAN | GROUP, sciach))
```



Adding a Random Effect

- We will continue using the `SCIACH` dataset to illustrate adding random effects.
- Recall that we already specified the model:

$$\text{science}_{ij} = \gamma_{00} + \gamma_{10}\text{urban} + u_{0j} + e_{ij}$$

- We now want to add a random effect for `urban` such that we allow the slope for `urban` to be random for each school. Before, we were forcing each school to have the same relationship between `urban` and `science`.
- Now we model:

$$\text{science}_{ij} = \gamma_{00} + \gamma_{10}\text{urban} + u_{0j} + u_{1j} + e_{ij}$$

Random Effects in R with `lmer`

```
> (lmer2 <- lmer(SCIENCE ~ URBAN + (URBAN | GROUP),
+     sciach))

Linear mixed model fit by REML
Formula: SCIENCE ~ URBAN + (URBAN | GROUP)
   Data: sciach
   AIC   BIC logLik deviance REMLdev
 424.2 442.6 -206.1    413.2   412.2
Random effects:
 Groups   Name        Variance  Std.Dev. Corr
 GROUP    (Intercept) 113.60330 10.65848
          URBAN         0.25200  0.50200 -0.625
 Residual               0.27066  0.52025
Number of obs: 160, groups: GROUP, 16
Fixed effects:
            Estimate Std. Error t value
(Intercept)  22.3912     2.7170   8.241
URBAN        -0.8670     0.1298  -6.679

Correlation of Fixed Effects:
      (Intr)
URBAN -0.641
```

Checking Data Fit

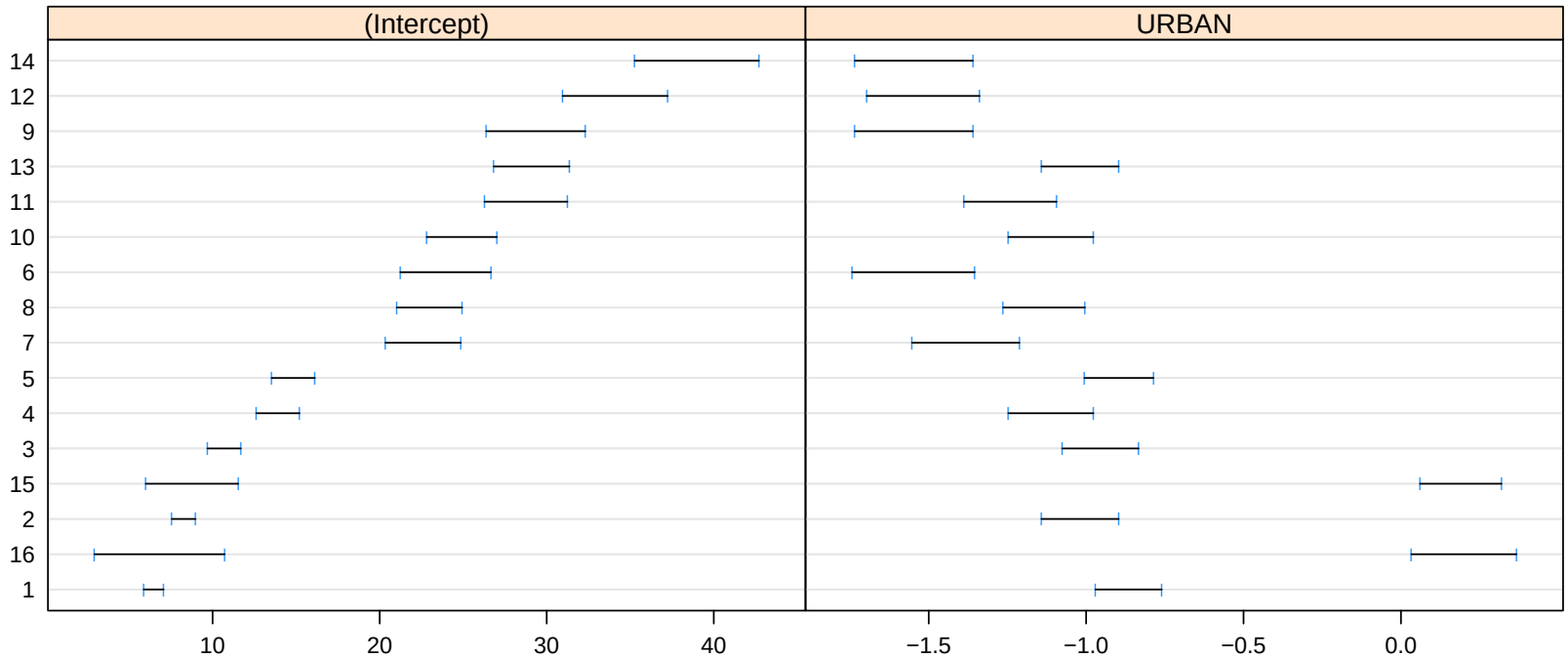
```
> anova(lmer0, lmer1, lmer2)

Data: sciach
Models:
lmer0: SCIENCE ~ 1 + (1 | GROUP)
lmer1: SCIENCE ~ URBAN + (1 | GROUP)
lmer2: SCIENCE ~ URBAN + (URBAN | GROUP)
      Df    AIC    BIC  logLik  Chisq Chi Df Pr(>Chisq)
lmer0  3 646.17 655.39 -320.08
lmer1  4 507.37 519.67 -249.69 140.79      1  < 2.2e-16
lmer2  6 425.22 443.67 -206.61  86.15      2  < 2.2e-16
```

Model Estimates

```
> coef(lmer2)

$GROUP
   (Intercept)      URBAN
1     7.038481 -0.7468711
2     8.901292 -0.8742823
3    11.668625 -0.8227981
4    15.130283 -0.9607326
5    16.185686 -0.7849307
6    26.028960 -1.2969888
7    24.550955 -1.1780440
8    24.894022 -0.9929563
9    31.570301 -1.3020013
10   26.967206 -0.9657083
11   30.982600 -1.0705294
12   36.360105 -1.2779096
13   31.267605 -0.8842923
14   41.201523 -1.2730680
15   12.724031  0.2710713
16   12.788238  0.2879613
```

Model Estimates (cont.)

```
> print(plot(confint(lmList(SCIENCE ~ URBAN | GROUP,
+     sciach), pooled = TRUE), order = 1))
```



Good R Resources

- Crawley, M. J. (2007). "The R Book." West Sussex, England: John Wiley and Sons.
- Quick R - `http://www.statmethods.net/`
- Kyle's homepage - `http://faculty.smu.edu/kyler/`
- IF you have a problem with a function inside R, use `RSiteSearch("your question")`.
- Verzani, J. (2005). "Using R for Introductory Statistics." London: Chapman and Hall,CRC.
- Maindonald, J. and Braun, J. (2003). "Data Analysis and Graphics Using R." New York, NY: Cambridge University Press.
- For more advanced books, see `http://www.r-project.org/doc/bib/R-books.html`.
- `http://mypage.iu.edu/~haguinis/R.pdf`

More R Stuff

- We made this entire presentation in R!
- By using the Sweave function in R, we are able to generate a LaTeXdocument of the "beamer" class for presentations. We then run this through any LaTeXprocessor to produce the slide presentation.
- We also use the Stangle function to generate the code blocks that are used in the script file.
- Helpful Websites:
  - LaTeXhelp - http://www.latex-project.org/
  - Sweave help - http://users.stat.umn.edu/~geyer/Sweave/
  - TeXnicCenter (Windows) - http://www.texniccenter.org/
  - TeXShop (Macs) - http://pages.uoregon.edu/koch/texshop/