# BIOS 621/821 Session 1

Levi Waldron

## Welcome and outline - session 1

- syllabus review
- software usage
- in-person / online course format
- multiple regression
  - continuous & categorical predictors
  - interactions
  - ANOVA tables
  - Model formulae
- introduction to R

## A bit about me - research interests

- High-dimensional statistics (more variables than observations)
- Predictive modeling
- Cancer genomics
- Metagenomic profiling of the human microbiome
- HIV treatment effectiveness
- `http://www.waldronlab.org` —— `http://waldronlab.github.io`

## Some of my activities that may interest you

- "Statistical Learning" book club and data competitions:
  - `https://groups.google.com/forum/#!forum/stat_learning`
- Research opportunities available

## Multiple Linear Regression Model (sec. 4.2)

Systematic part of model:

$$E[y|x] = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_p x_p$$

- $E[y|x]$ is the expected value of $y$ given $x$
- $y$ is the outcome, response, or dependent variable
- $x$ is the vector of predictors / independent variables
- $x_p$ are the individual predictors or independent variables
- $\beta_p$ are the regression coefficients

## Multiple Linear Regression Model (cont'd)

Random part of model:

$y_i = E[y_i|x_i] + \epsilon_i$

$y_i = \beta_0 + \beta_1 x_{1i} + \beta_2 x_{2i} + ... + \beta_p x_{pi} + \epsilon_i$

- $x_{ji}$ is the value of predictor $x_j$ for observation $i$

Assumption: $\epsilon_i \overset{iid}{\sim} N(0, \sigma_\epsilon^2)$

- Normal distribution
- Mean zero at every value of predictors
- Constant variance at every value of predictors
- Values that are statistically independent

## Continuous predictors

- **Coding:** as-is, or may be scaled to unit variance (which results in *adjusted* regression coefficients)
- **Interpretation for linear regression:** An increase of one unit of the predictor results in this much difference in the continuous outcome variable
  - *additive model*

## Binary predictors (2 levels)

- **Coding:** indicator or dummy variable (0-1 coding)
- **Interpretation for linear regression:** the increase or decrease in average outcome levels in the group coded "1", compared to the reference category ("0")
- *e.g.* $E(y|x) = \beta_0 + \beta_1 x$
- where x=$\{$ 1 if male, 0 if female $\}$

## Multilevel Categorical Predictors (Ordinal or Nominal)

- **Coding:** $K - 1$ dummy variables for $K$-level categorical variables *
- **Interpretation for linear regression:** as above, the comparisons are done with respect to the reference category
- Testing significance of multilevel categorical predictor: partial F-test, a.k.a. nested ANOVA

* STATA and R code dummy variables automatically, behind-the-scenes

## Inference from multiple linear regression

- Coefficients are t-distributed when assumptions are correct
- Variance in the estimates of each coefficient can be calculated
- The t-test of the null hypothesis $H_0 : \beta_1 = 0$ and from confidence intervals tests whether $x_1$ predicts $y$, *holding other predictors constant*
  - often used in causal inference to control for confounding: see section 4.4

## Interaction (effect modification)

Interaction is modeled as the product of two covariates:

$$E[y|x] = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_{12} x_1 * x_2$$

## Interaction (effect modification)

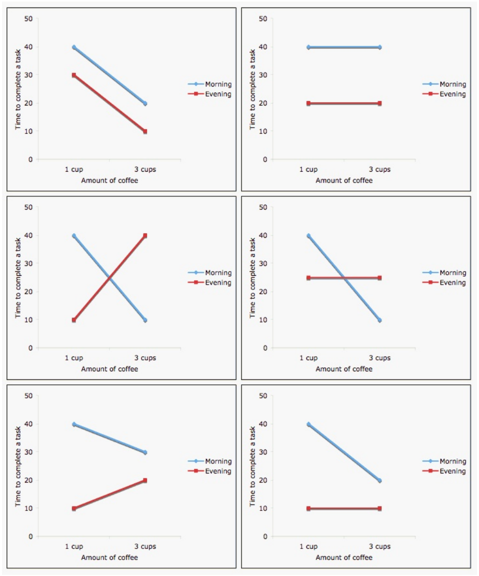


Figure 1: Interaction between coffee and time of day on performance

## ANOVA table

| Source of Variation | Sum Sq | Deg Fr | Mean Sq | F |
|---|---|---|---|---|
| Model | MSS | k | MSS/k | (MSS/k)/MSE |
| Residual | RSS | n-(k-1) | RSS/(n-k-1) | |
| Total | TSS | n-1 | | |

- $k$ = Model degrees of freedom = coefficients - 1
- $n$ = Number of observations
- **F** is F-distributed with $k$ numerator and $n-(k-1)$ denominator degrees of freedom

## Regression in R: model formulae

Model formulae tutorial

- regression functions in R such as `aov()`, `lm()`, `glm()`, and `coxph()` use a “model formula” interface.
- The formula determines the model that will be built (and tested) by the R procedure. The basic format is:

  *response variable* $\sim$ *explanatory variables*

- The tilde means “is modeled by” or “is modeled as a function of.”

## Model formulae (cont'd)

Model formula for simple linear regression:

$y \sim x$

- where “x” is the explanatory (independent) variable
- “y” is the response (dependent) variable.

## Model formulae (cont'd)

Additional explanatory variables would be added as follows:

$y \sim x + z$

Note that "+" does not have its usual meaning, which would be achieved by:

$y \sim I(x + z)$

## Types of standard linear models

```
lm( y ~ u + v)
```

`u` and `v` factors: **ANOVA**
`u` and `v` numeric: **multiple regression**
one factor, one numeric: **ANCOVA**

## Model formulae (cont'd)

| symbol | example | meaning |
|---|---|---|
| + | + x | include this variable |
| - | - x | delete this variable |
| : | x : z | include the interaction |
| | x * z | include these variables and their interactions |
| / | x / z | nesting: include z nested within x |
| \| | x \| z | conditioning: include x given z |
| ^ | (u + v + w)^3 | include these variables and |
| | | all interactions up to three way |
| 1 | -1 | intercept: delete the intercept |

## Model formulae (cont'd)

How to interpret the following model formulae?

y ~ u + v + w + u:v + u:w + v:w

y ~ u * v * w - u:v:w

y ~ (u + v + w)^2

## Model formulae (cont'd)

How to interpret the following model formulae?

$y \sim u + v + w + u{:}v + u{:}w + v{:}w + u{:}v{:}w$

$y \sim u * v * w$

$y \sim (u + v + w)\hat{}3$

## Introduction to the R language

- `5 + 2  #addition`
- `5 - 2  #subtraction`
- `5 * 2  #multiplication`
- `5 / 2  #division`
- `5 ^ 2  #exponentiation`
- `5 ** 2 #exponentiation`
- `5 %% 2 #modulus (a.k.a. remainder)`

## Logic

- `5 < x  #less than`
- `5 <= x #less than or equal to`
- `5 > x  #greater than`
- `5 >= x #greater than or equal to`
- `5 == x #equal to`
- `5 != x #not equal to`
- `!x     #logical NOT`
- `True || False  #stepwise logical OR`
- `True && False  #stepwise logical AND`

## Storing Data: The Rules

- Letters, numbers, dots, underscores
- Must start with a letter or a dot not followed by a number
- No reserve words, No spaces

```
x <- 5
x * 2
```

```
## [1] 10
```

```
x <- x + 1
y <- 4
x * y
```

```
## [1] 24
```

## Basic Data Types

- numeric (set seed to sync random number generator):

```
set.seed(1)
rnorm(5)
```

```
## [1] -0.6264538  0.1836433 -0.8356286  1.5952808  0.32950
```

- integer:

```
1:5
```

```
## [1] 1 2 3 4 5
```

```
sample( 1:5 )
```

```
## [1] 2 1 3 4 5
```

## Basic Data Types

- character:

```
c("yes", "no")
```

```
## [1] "yes" "no"
```

- factor (play with this to show character/integer properties):

```
factor(c("yes", "no"))
```

```
## [1] yes no
## Levels: no yes
```

## Basic Data Types

- ordered factor:

```
factor(c("good", "very good", "poor"),
       levels=c("poor", "good", "very good"),
       ordered=TRUE)
```

```
## [1] good      very good poor
## Levels: poor < good < very good
```

- logical:

```
1:5 %in% 4:5
```

```
## [1] FALSE FALSE FALSE  TRUE  TRUE
```

- Missing Values and others - **IMPORTANT**

```
c(NA, NaN, -Inf, Inf)
```

```
## [1]   NA  NaN -Inf  Inf
```

## Vectors Must Be of One Data Mode

```r
c( 1, "2", FALSE)
```

```
## [1] "1"     "2"     "FALSE"
```

```r
c( 1, FALSE )
```

```
## [1] 1 0
```

## Selecting Vector Elements

- One element

```
x <- 1:4
x[ 2 ]
```

```
## [1] 2
```

- A slice of a vector

```
x <- 1:10
x[ 4:7 ]
```

```
## [1] 4 5 6 7
```

## Selecting Vector Elements

- Multiple elements ( not contiguous )

```
x <- c( "a", "b", "c", "d", "e", "f" )
x[ c(5,3,1) ]
```

```
## [1] "e" "c" "a"
```

- Removing elements

```
x[ -1 ]
```

```
## [1] "b" "c" "d" "e" "f"
```

```
x[-1:-2]
```

```
## [1] "c" "d" "e" "f"
```

## Selecting Vector Elements

- Using logical vector

```
x <- 1:10
y <- x%%2 == 0
x[y]
```

```
## [1]  2  4  6  8 10
```

## 2-Dimensional Vectors are Matrices

```r
matrix( 1:20, nrow = 5, ncol = 4 )
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    6   11   16
## [2,]    2    7   12   17
## [3,]    3    8   13   18
## [4,]    4    9   14   19
## [5,]    5   10   15   20
```

## Indexing Matrices

- matrix[ r, c ]

```
boring.matrix <- matrix( 1:20, nrow = 5, ncol = 4 )
dim( boring.matrix )
```

```
## [1] 5 4
```

```
boring.matrix[ ,1 ]
```

```
## [1] 1 2 3 4 5
```

```
boring.matrix[ 2, 1 ]
```

```
## [1] 2
```

```
boring.matrix[ 2, ]
```

```
## [1]  2  7 12 17
```

## Indexing Matrices

```
boring.matrix
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    6   11   16
## [2,]    2    7   12   17
## [3,]    3    8   13   18
## [4,]    4    9   14   19
## [5,]    5   10   15   20
```

```
boring.matrix[ boring.matrix[ ,1 ] ==3,]
```

```
## [1]  3  8 13 18
```

## Matrix Operations

- Transpose

```
boring.matrix <- matrix(1:9, nrow = 3)
boring.matrix
```

```
##      [,1] [,2] [,3]
## [1,]    1    4    7
## [2,]    2    5    8
## [3,]    3    6    9
```

```
t(boring.matrix)
```

```
##      [,1] [,2] [,3]
## [1,]    1    2    3
## [2,]    4    5    6
## [3,]    7    8    9
```

## Matrix Operations (cont'd)

- Adding

```
boring.matrix + 1
```

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

```
boring.matrix + 1:3
```

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

## Matrix Operations (cont'd)

- Adding

```
boring.matrix
```

```
##      [,1] [,2] [,3]
## [1,]    1    4    7
## [2,]    2    5    8
## [3,]    3    6    9
```

```
boring.matrix + boring.matrix
```

```
##      [,1] [,2] [,3]
## [1,]    2    8   14
## [2,]    4   10   16
## [3,]    6   12   18
```

## Matrix Operations (cont'd)

- Multiplying

```
boring.matrix * boring.matrix
```

```
##      [,1] [,2] [,3]
## [1,]    1   16   49
## [2,]    4   25   64
## [3,]    9   36   81
```

```
boring.matrix %*% boring.matrix
```

```
##      [,1] [,2] [,3]
## [1,]   30   66  102
## [2,]   36   81  126
## [3,]   42   96  150
```

## Naming rows and columns

```
colnames(boring.matrix) <- c("col.1", "col.2", "col.3")
rownames(boring.matrix) <- c("row.1", "row.2", "row.3")
boring.matrix
```

```
##       col.1 col.2 col.3
## row.1     1     4     7
## row.2     2     5     8
## row.3     3     6     9
```

```
boring.matrix["row.1", ]
```

```
## col.1 col.2 col.3
##     1     4     7
```

## Lists are Filing Cabinets

- e.g. if we have 5 medical measurements, 10 self-reported measurements, a sex, two parent names:

```
my.person <- list( measurements, self.reporting,
                   sex, parents)
my.person
```

```
## [[1]]
## [1]  1.3  1.6  3.2  9.8 10.2
##
## [[2]]
##  [1] 13  6  4  7  6  5  8  9  7  4
##
## [[3]]
## [1] FALSE
##
## [[4]]
## [1] "Parent1.name" "Parent2.name"
```

## Lists are Filing Cabinets

- Single bracket accessing

```
my.person[1:2]
```

```
## [[1]]
## [1]  1.3  1.6  3.2  9.8 10.2
##
## [[2]]
##  [1] 13  6  4  7  6  5  8  9  7  4
```

- Double bracket accessing

```
my.person[[1]]
```

```
## [1]  1.3  1.6  3.2  9.8 10.2
```

## Lists are Filing Cabinets

```
my.person <- list( measure = measurements,
                   self.measure = self.reporting,
                   s = sex,
                   parents = parents )
my.person
```

```
## $measure
## [1]  1.3  1.6  3.2  9.8 10.2
##
## $self.measure
##  [1] 13  6  4  7  6  5  8  9  7  4
##
## $s
## [1] FALSE
##
## $parents
## [1] "Parent1.name" "Parent2.name"
```

## The data.frame object

- a data.frame is:
- a matrix with columns of potentially different data types, and
- a `list` with vector elements of equal length

```
x <- 11:16
y <- seq(0,1,.2)
z <- c( "one", "two", "three", "four", "five", "six" )
a <- factor( z )
```

```
test.dataframe <- data.frame(x,y,z,a)
```

## Accessing data.frame elements

```
test.dataframe[[4]]
```

```
## [1] one   two   three four  five  six
## Levels: five four one six three two
```

```
test.dataframe$parents
```

```
## NULL
```

## Columns of a data.frame May Contain Different Data Modes

```
class( test.dataframe[[1]] )
```

```
## [1] "integer"
```

```
class( test.dataframe[[2]] )
```

```
## [1] "numeric"
```

```
class( test.dataframe[[3]] )
```

```
## [1] "factor"
```

```
class( test.dataframe[[4]] )
```

```
## [1] "factor"
```

## Combining Data Frames

- binding columns with common row names

```
mini.frame.one <- data.frame( "one" = 1:5 )
mini.frame.two <- data.frame( "two" = 6:10 )
```

```
cbind( mini.frame.one, mini.frame.two )
```

```
##   one two
## 1   1   6
## 2   2   7
## 3   3   8
## 4   4   9
## 5   5  10
```

Alternatively: `c( mini.frame.one, mini.frame.two )`

- rbind for binding rows ( with common column names )

## Updating Data Frames

```
test.dataframe
```

```
##    x   y     z     a
## 1 11 0.0   one   one
## 2 12 0.2   two   two
## 3 13 0.4 three three
## 4 14 0.6  four  four
## 5 15 0.8  five  five
## 6 16 1.0   six   six
```

```
test.dataframe[[1]] = 21:26
test.dataframe
```

```
##    x   y     z     a
## 1 21 0.0   one   one
## 2 22 0.2   two   two
## 3 23 0.4 three three
## 4 24 0.6  four  four
## 5 25 0.8  five  five
## 6 26 1.0   six   six
```

## Navigating Directories

- Paths start at where you opened R in the terminal
- Home directory for RStudio
- "/" inside a folder
- "parent_folder/inside_folder"
- ".." move up one
- "../.." move up two
- getwd()
- setwd()

## Reading Tables

- `read.table`
- `read.csv`
- `read.delim`

## Example: Cholesterol vs. Age

```
chol <- read.delim("http://www.statlab.uni-heidelberg.de/da
summary(chol)
```

```
##   cholesterol         age             state
##  Min.   :112.0   Min.   :18.00   Iowa    :11
##  1st Qu.:181.2   1st Qu.:39.50   Nebraska:19
##  Median :199.0   Median :48.00
##  Mean   :213.7   Mean   :48.57
##  3rd Qu.:247.0   3rd Qu.:58.00
##  Max.   :356.0   Max.   :78.00
```

## Example: Cholesterol vs. Age linear model

```
fit <- lm(cholesterol ~ age * state, data=chol)
summary(fit)
```

```
##
## Call:
## lm(formula = cholesterol ~ age * state, data = chol)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -73.480 -31.907  -4.303  22.829  85.833
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)        35.8112    55.1166   0.650  0.52156
## age                 3.2381     1.0088   3.210  0.00352 **
## stateNebraska      65.4866    61.9834   1.057  0.30045
## age:stateNebraska  -0.7177     1.1628  -0.617  0.54247
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 43.14 on 26 degrees of freedom
## Multiple R-squared:  0.5326, Adjusted R-squared:  0.4786
## F-statistic: 9.875 on 3 and 26 DF,  p-value: 0.00016
```

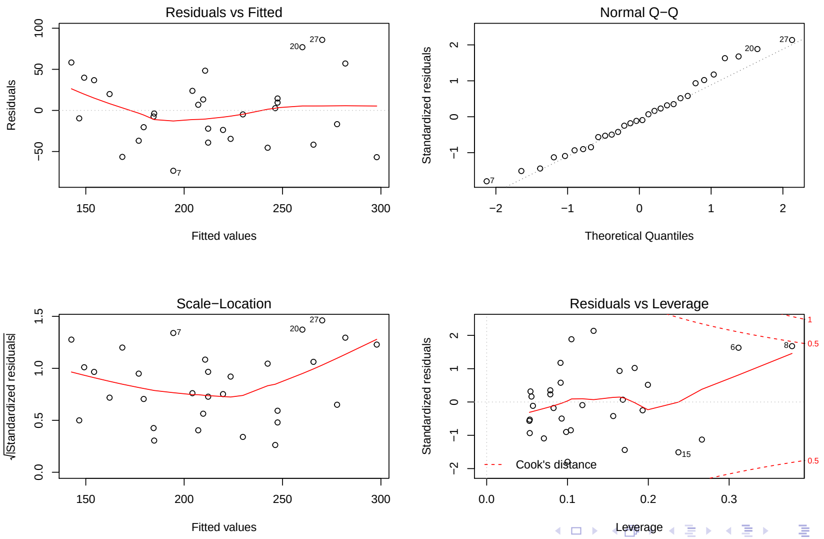
## Example: Cholesterol vs. Age ANOVA table

```r
anova(fit)
```

```
## Analysis of Variance Table
##
## Response: cholesterol
##           Df Sum Sq Mean Sq F value    Pr(>F)
## age        1  48976   48976 26.3124 2.388e-05 ***
## state      1   5456    5456  2.9315   0.09877 .
## age:state  1    709     709  0.3809   0.54247
## Residuals 26  48395    1861
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.
```

## Example: Cholesterol vs. Age diagnostic plots

```r
par(mfrow=c(2, 2))
plot(fit)
```



## Example: Cholesterol vs. Age partial F-test

```
fit1 <- lm(cholesterol ~ state, data=chol)
fit2 <- lm(cholesterol ~ state + age, data=chol)
anova(fit1, fit2)
```

```
## Analysis of Variance Table
##
## Model 1: cholesterol ~ state
## Model 2: cholesterol ~ state + age
##   Res.Df    RSS Df Sum of Sq      F    Pr(>F)
## 1     28 102924
## 2     27  49104  1     53820 29.593 9.361e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.
```

## Example: Cholesterol vs. Age backwards selection

```r
library(MASS)
fit <- lm(cholesterol ~ age * state, data=chol)
step <- stepAIC(fit, direction="backward")
```

```
## Start:  AIC=229.58
## cholesterol ~ age * state
##
##             Df Sum of Sq   RSS    AIC
## - age:state  1    709.05 49104 228.01
## <none>                   48395 229.58
##
## Step:  AIC=228.01
## cholesterol ~ age + state
##
##         Df Sum of Sq    RSS    AIC
## <none>                49104 228.01
## - state  1      5456  54560 229.18
## - age    1     53820 102924 248.22
```

AIC = Akaike's Information Criterion