DATA 500

Lecture 3

Data Basics

EDA 1

The syllabus is almost done

For the next two weeks:

23	Data Types	Ch2-Verzani-2014:20-50;
		Ch2-Lantz-2015;
25	Exploratory Data Analysis (EDA)	Ch2-Verzani-2014:70-80;
	Graphical Methods 1	
30	Graphical Methods 2	Ch2-Verzani-2014:80-87;
Feb		
1	Numerical Summaries of data	Ch1-PSDS;
		Ch2-Verzani-2014:50-70;
6	Intro to (Discrete) Probability	CH4-Moore:pgs 231-258;
8	Probability distributions in R	CH6-Verzani-2014: p211-236;
	Continuous probabilities	

Sample Basics

A **data set** or **data table** is a **sample** from a larger **population**

A sample has information from a number ($\boldsymbol{n}$) of
cases or **individuals**

The information is called
observations/characteristics,
or **measurements**

The observations are often called **variables** (beak length)

The data are usually in rows and columns, called a "dataframe" in R

Data Basics

The variables are usually of two types:

A **categorical variable** has data from two or more groupings, or categories.

A **quantitative variable** takes numerical values for which arithmetic operations such as adding and averaging make sense.

- counts of things
- measurements of things

The **distribution** of a variable tells us the values that a variable takes and how often it takes each value.

- frequencies of categories
- distributions of quantitative variables

Basic Sample Notation

Single observations can be designated

$$x_1, x_2, x_3, x_4, \ldots, x_n$$

x_{55} or "the value of the fifty-fifth case" (in the data table)

- the index number is often designated with subscript *i*: x_i

Multiple measurements can be designated by row (1..n) and column (c)

$$x_{11}, x_{12}, x_{13}, x_{21}, x_{22}, x_{23}, \ldots, x_{nc}$$

- the row and column index subscripts are often designated *i,j*: x_{ij}

Data: Characteristics and Observations
in order of increasing information

Non-metric, qualitative variables

1. Nominal: differ in kind, no clear order, at least three.

occupation, religion, color categories (red, green, blue, etc.)

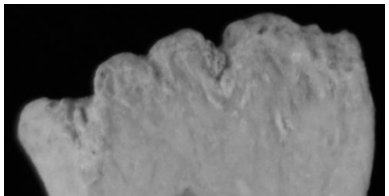
1a. Binary/Dichotomous: two possibilities

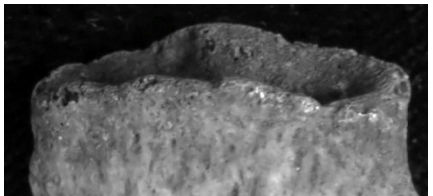
(sex: M or F; feathers: Y or N; yellow vs. not yellow)

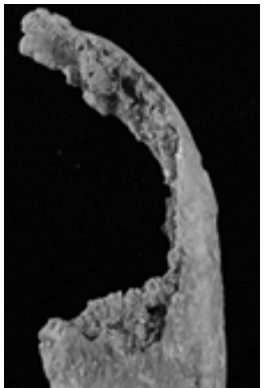
2. Ordinal: have a logical order; **stage, grade**; NO measurement

small, medium, large; light, medium, dark;

Stages 1, 2, 3/A,B,C,D,E. (disease progression; rib ends)







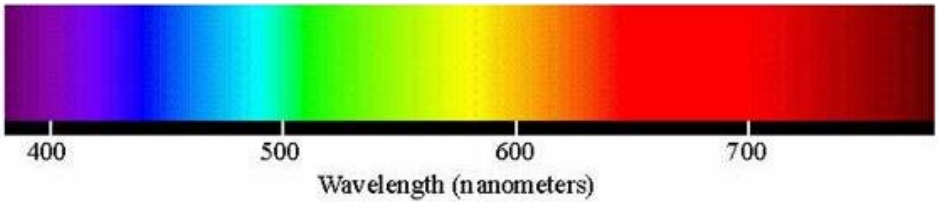
Data: Characteristics and Observations
in order of increasing information

Metric, quantitative variables

3. Interval: equal scale; zero and ratios not meaningful

Fahrenheit, Celsius temperatures; color wavelength

(80 degrees vs. 40 degrees)



4. Ratio: zero and ratios meaningful;

Linear measurements; Time duration; Counts (frequency);

Kelvin temperature;

0 Kelvin = -273˚ C or -460˚ F

Amount of information: N < B < O < I ≤ R

Data Types in R

Record Identifier (should be unique)

HTH302, 43, 45B, A27, GHR, S0022, etc.

Integer or Alphanumeric, **Character in R**

Categorical variables (Nominal, Binary)

- Sex: string, M or F
- Treatment (Y: treatment; N: control)
- Treatment (Drug1, Drug2, Control, 1,2,3 as factors)

Alphanumeric, **Character** or **Factor in R**

Data Types in R

Ordinal categories

- Stage (1,2,3 or A,B,C)

Ordered Factor in R

Integers (discrete numbers)

- Number of visits to a web site **(count)**
- Number of questions answered correctly **(count)**
- Age (rounded to integers: ***quasi-continuous***)
- Blood pressure (rounded to integers: ***quasi-continuous***)

Integer in R (0, 3, 34, 6)

Data Types in R

Continuous measurements with decimals

- head length, shoulder width
- shell diameter
- blood pressure readings

Floating point decimal, **Number in R (2.3, 4.347)**

Simple Data in R: Data Vectors

Assign multiple numbers to list (a vector: one row of numbers)

- use the c function (combine)

NOTE: in R you will get nothing back unless you ask for it

(unless there is an error!)

```
whale <- c(74, 122, 235, 111, 292, 111, 211, 133, 156, 79)
```

```
^^ These data are the number of whales beached in Texas each
   year between 1990 and 1999.
```

year	Beached whales
1990	74
1991	122
...	...
1999	79

```
length(whale)
[1] 10
mean(whale)
[1] 152.4
```

Missing Values

Hip replacement costs (NA means missing data, Not Available)

10,500 45,000 74,100 NA 83,500 86,000 38,200 NA

44,300 12,500 55,700 43,900 71,900 NA 62,000

Do not add commas to numbers:

```
hip_cost <- c(44,300 12,500 55,700 43,900 71,900 NA 62,000)
Error: unexpected numeric constant in "hip_cost <c(44,300 12"

hip_cost <- c(10500, 45000, 74100, NA, 83500 , 86000, 38200, NA,
44300,12500,55700, 43900 ,71900, NA, 62000)

hip_cost
[1] 10500 45000 74100 NA 83500 86000 38200 NA 44300 12500 55700
   43900 71900 NA 62000
length(hip_cost)
[1] 15
```

Missing Values

```
mean(hip_cost)
```

```
[1] NA
```

Problem? we tried to add NAs (missing values) to numbers

```
hip_cost
```

```
[1] 10500 45000 74100 NA 83500 86000 38200 NA 44300 12500 55700
   43900 71900 NA 62000
```

Why not use zeros instead of NAs?

```
# use na.omit to remove NAs
na.omit(hip_cost)
```

(returns all values from hip_cost after removing NAs)

```
mean(na.omit(hip_cost))
```

```
[1] 52300
```

Data Coercion in R

Data get forced into the most inclusive category

character/string/alphanumeric is the loosest category

```
x <- c(1, "two", "III")
x
[1] "1" "two" "III"
```

try a typo

```
num_data <- c(1,3,5,6,4 5, 5a, 2,8.5)
Error: unexpected symbol in "num_data <- c(1,3,5,6,4, 5, 5a"

# numeric data accommodates all numbers
num_data <- c(1,3,5,6,4, 5, 5,2,8.5)
num_data
num_data [1] 1.0 3.0 5.0 6.0 4.0 5.0 5.0 2.0 8.5
```

Coercion

NA is used as NA, not "NA" (it is not a string)

We can force, or try to force data into a category

```
as.numeric("23")
[1] 23
#hmmm - what will happen?
as.numeric("23a")

as.character(23a) #will fail

as.character("23a")

as.character(num_data)
[1] "1" "3" "5" "6" "4" "5" "5"
[8] "2" "8.5"
```

Index numbers and sequences

The index is designated *i*: x_i **(*i* = 1 to *n*)**

```
whale <- c(74, 122, 235, 111, 292, 111, 211, 133,156,79)
# look at an individual value: put index inside brackets
length(whale)  # should be 10!
[1]  10
whale[4]  # (square) brackets!  i = 4, 4th value
[1]  111
```

How do we get a series of individuals?

```
whale [1:5]  # use a colon to get a sequence
[1]  74 122 235 111 292
[1:10] # produces...?
```

Index numbers and sequences

The seq() function: sequence

```
seq(1,10) # parens used; produces the same as 1:10
 [1]  1  2  3  4  5  6  7  8  9 10

# add "by =" for the interval between sequence numbers
# get odd numbers
seq(1,10, by = 2)  # or use by=2
[1] 1 3 5 7 9

# get even numbers
seq(2,10, by = 2)
[1]  2  4  6  8 10
```

How can we get the odd-numbered individuals in *whale*?

Index numbers and sequences

How can we get the odd-numbered individuals in *whale*?

1. remember to use brackets

2. remember to use sequence (seq) INSIDE the brackets

```
# get odd numbers
seq(1,10, by = 2)
# put (nest) that list inside the brackets
whale[seq(1,10, by=2)]
whale[ seq(1,10, by = 2) ]  # more readable?
```

How can we get the odd-numbered individuals in ANY SIZE data?

- use the length() function

```
whale[ seq(1,10, by=2) ]
whale[ seq(1, length(whale), by=2) ]
```

Index numbers and sequences

How can we get all BUT one individual in *whale*?

use the negative sign (means NOT)

 # all in whale except for # 5

```
whale[-5]
[1]  74 122 235 111 111 211 133 156  79
# what answer will the following produce?
length(whale[-5])

#we can add a new value to a data vector
whale [11] <- 99
whale [13] <- 95  # will that work?
whale[15:19] <- 23 # same value inserted for several
```

how to get rid of extra entries?

```
whale <- whale[1:10]  # resets to original 10 individuals
```

Index numbers and sequences

list all records

```
whale
whale[]
```

How can we get TWO values from *whale*?

```
whale[2,3] # will not work
whale[c(2,3)]
whale[c(2,3,6:9)]  # will give us more
```

How can we get all BUT two values in *whale*?

```
whale[-c(2,3)] #negative sign, as before

whale [55] # we know there are only 10
 [NA]
```

Repeating numbers and sequences

The rep function

```
rep(5, times=10)
 [1] 5 5 5 5 5 5 5 5 5 5

rep(1:3, times=4)
 [1] 1 2 3 1 2 3 1 2 3 1 2 3

rep(1:3, 4)
 [1] 1 2 3 1 2 3 1 2 3 1 2 3

rep(rep(1:3, 4),2)
 [1] 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3
```

Logical values

```
is.na(1)
[1] FALSE
is.na("NA")
[1] FALSE

is.na(NA)
[1] TRUE

3 < pi   # pi = 3.14159..
[1] TRUE
```

Use double equal signs to ask a T/F equality question

```
3 * 5 == 15
[1] TRUE
```

Logical values

```
# logical operations will work on a vector
whale == 111
[1] FALSE FALSE FALSE TRUE FALSE TRUE FALSE FALSE FALSE FALSE

whale > 100 # ten answers for all ten values

# ANY values over 100? ONE answer
any(whale > 100)

# count of values over 100 (T = 1, F = 0)
sum(whale > 100)

# are ALL values over 50? ONE answer
all(whale > 50)
```

Logical values

```
whale
[1] 74 122 235 111 292 111 211 133 156 79
diff(whale) # subtract each value from the next value
[1] 48 113 -124 181 -181 100 -78 23 -77

mean(whale)
[1] 152.4
# logical operations will work on a vector
whale > mean(whale)
[1] FALSE FALSE TRUE FALSE TRUE FALSE TRUE FALSE TRUE FALSE

# get the values that are > whale mean
whale[whale > mean(whale)]
[1] 235 292 211 156
```

Dataframes in R

Data in rows and columns

(records and fields)

Good data practices:

Have a “key field”, at least one field with unique values

R will number records but it can be inconsistent!

Set up categorical fields as the first (left) ones

Leave out memo fields and erroneous text fields for analysis

Shorten long column names

"GDP_for year 2001-US"

Remove spaces in field names **ALWAYS**

R will automatically convert field names into funky variants

Spaces are bad in field names

In field names: “head ln” → “HeadLn” or “Head_Ln”

In field values (museum number):

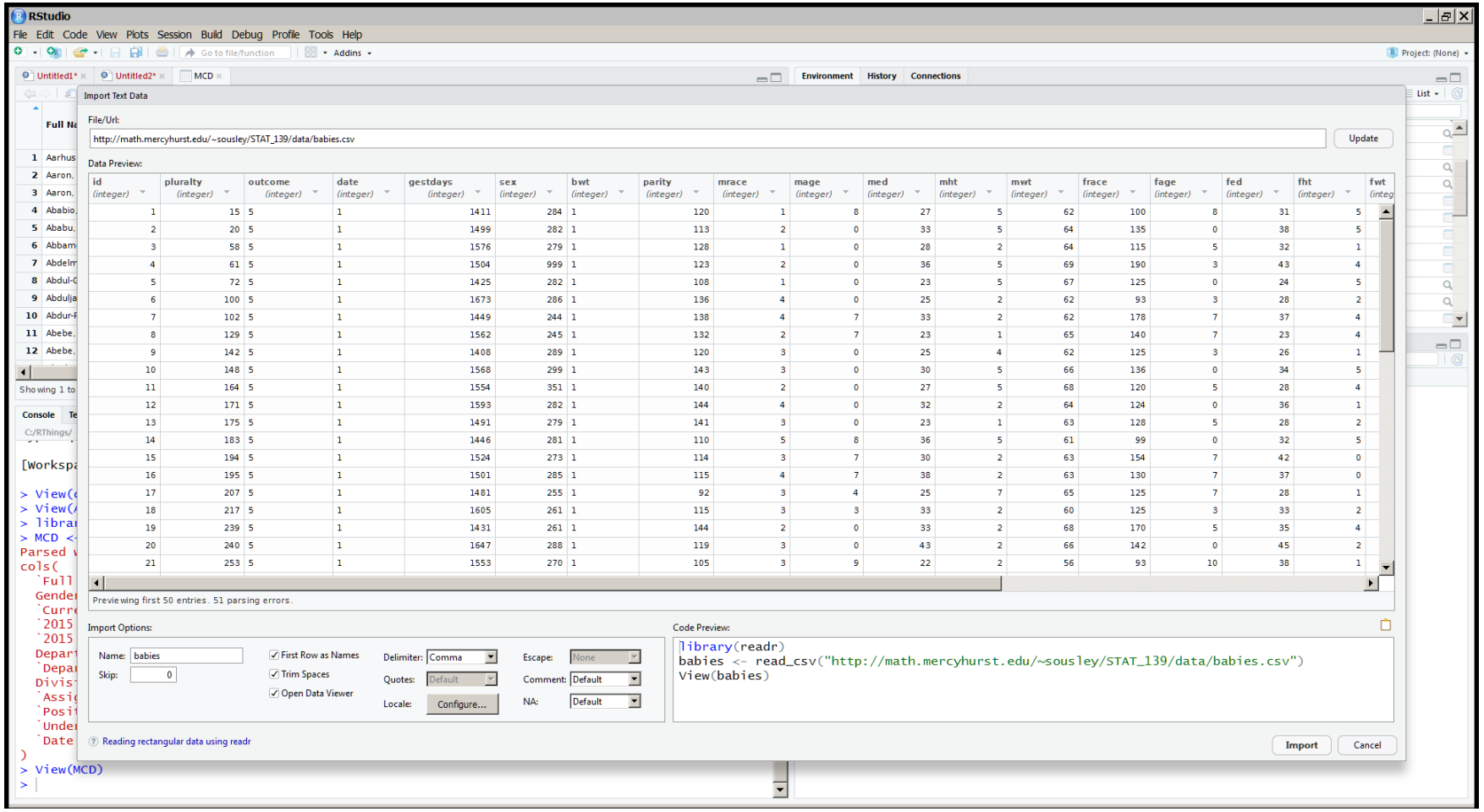
"HTH 102" → "HTH102" or "HTH_102"

Use leading zeros if you care about sorting!

HTH102	→	**HTH102**		**HTH004**
HTH4	→	**HTH004**	**sorted:**	**HTH099**
HTH99	→	**HTH099**		**HTH102**

One way to import data

One way to load the babies data:
In RStudio, top right window:
choose Environment | Import Dataset | From Text(readr) and type in:
`http://math.mercyhurst.edu/~sousley/STAT_139/data/babies.csv`
then press the Update button



Double check that "First Row as Names" is checked, then press the Import button.

Data in babies dataframe

```
# Another way:
babies <- read.csv("http://math.mercyhurst.edu/~sousley/STAT_139/data/babies.csv", header=T);
```

```
str(babies) # get the structure
```

```
'data.frame':    1236 obs. of  23 variables:
 $ id      : int  15 20 58 61 72 100 102 129 142 148 ...
 $ pluralty: int  5 5 5 5 5 5 5 5 5 5 ...
 $ outcome : int  1 1 1 1 1 1 1 1 1 1 ...
 $ date    : int  1411 1499 1576 1504 1425 1673 1449 1562 1408 1568 ...
 $ gestdays: int  284 282 279 999 282 286 244 245 289 299 ...
 $ sex     : int  1 1 1 1 1 1 1 1 1 1 ...
 $ bwt     : int  120 113 128 123 108 136 138 132 120 143 ...
 $ parity  : int  1 2 1 2 1 4 4 2 3 3 ...
 $ mrace   : int  8 0 0 0 0 0 7 7 0 0 ...
 $ mage    : int  27 33 28 36 23 25 33 23 25 30 ...
 $ med     : int  5 5 2 5 5 2 2 1 4 5 ...
 $ mht     : int  62 64 64 69 67 62 62 65 62 66 ...
 $ mwt     : int  100 135 115 190 125 93 178 140 125 136 ...
 $ frace   : int  8 0 5 3 0 3 7 7 3 0 ...
 $ fage    : int  31 38 32 43 24 28 37 23 26 34 ...
 $ fed     : int  5 5 1 4 5 2 4 4 1 5 ...
 $ fht     : int  65 70 99 68 99 64 99 71 70 99 ...
 $ fwt     : int  110 148 999 197 999 130 999 192 180 999 ...
 $ marital : int  1 1 1 1 1 1 1 1 0 1 ...
 $ inc     : int  1 4 2 8 1 4 98 2 2 2 ...
 $ msmoke  : int  0 0 1 3 1 2 0 0 0 1 ...
 $ time    : int  0 0 1 5 1 2 0 0 0 1 ...
 $ number  : int  0 0 1 5 5 2 0 0 0 4 ...
```

babies columns

What kind of data is in each?

id identification number **unique integer; key field**

pluralty 5= single fetus

outcome 1= live birth that survived at least 28 days

date birth date where 1096=January1,1961

gestation length of gestation in days **day count (integer) , 999 = unknown (gestdays)**

sex infant's sex 1=male 2=female 9=unknown **baby's sex, categorical, coded as integer**

wt birth weight in ounces (999 unknown) **measurement, 999 = unknown (bwt)**

parity total number of previous pregnancies including fetal deaths and still births, 99=unknown

race mother's race 0-5=white 6=mex 7=black 8=asian 9=mixed 99=unknown

age mother's age in years at termination of pregnancy, 99=unknown **(mage)**

ed mother's education 0= less than 8th grade, 1 = 8th -12th grade - did not graduate, 2= HS graduate–no other schooling , 3= HS+trade, 4=HS+some college 5= College graduate, 6\&7 Trade school HS unclear, 9=unknown **(med)**

babies columns (continued)

ht mother's height in inches to the last completed inch 99=unknown **(mht)**

wt1 mother prepregnancy wt in pounds, 999=unknown **(mwt)**

drace father's race, coding same as mother's race.

dage father's age, coding same as mother's age.

ded father's education, coding same as mother's education.

dht father's height, coding same as for mother's height

dwt father's weight coding same as for mother's weight

marital 1=married, 2= legally separated, 3= divorced, 4=widowed, 5=never married

inc family yearly income in \$2500 increments 0 = under 2500, 1=2500-4999, ..., 8= 12,500-14,999, 9=15000+, 98=unknown, 99=not asked

smoke does mother smoke? 0=never, 1= smokes now, 2=until current pregnancy, 3=once did, not now, 9=unknown

generally, changed d for dad to f for father

babies

We can refer to a column in a dataframe using $:

```
babies$gestdays
```

```
   [1] 284 282 279 999 282 286 244 245 289 299 351 282 279 281 273 285 255 261 261
  [20] 288 270 274 287 276 294 261 280 266 292 274 270 278 268 275 281 283 279 288
  [39] 267 282 293 278 302 270 248 274 294 272 275 291 258 283 282 286 267 275 278
  [58] 257 273 232 273 288 280 245 283 282 246 274 273 276 289 292 284 274 270 274
  [77] 286 276 277 272 293 280 292 274 287 274 294 296 305 999 281 268 271 999 278
  [96] 282 255 302 999 254 279 274 286 280 273 279 287 273 303 274 269 302 255 293
 [115] 279 276 278 283 264 243 288 284 288 284 276 283 277 267 272 225 278 266 294
...
[1198] 281 300 338 255 285 297 260 273 266 242 247 281 283 273 265 286 271 293 267
[1217] 266 319 285 321 284 290 288 262 281 287 244 278 276 290 270 275 265 291 281
[1236] 297
```

```
summary(babies$gestdays)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  148.0   272.0   280.0   286.9   288.0   999.0
```

babies

We can also refer to a records and columns in a dataframe using numbers

- Similar to a list, but we have TWO dimensions: `dataframe[row, column]`

```
babies[1,] # I can leave one number out - will give me all columns from first record;
  id pluralty outcome date gestdays sex bwt parity mrace mage med mht mwt frace fage
1 15        5       1 1411      284   1 120      1     8   27   5  62 100     8   31
  fed fht fwt marital inc msmoke time number
1   5  65 110       1   1      0    0      0
# first row, 5th value;
babies[1,5];
[1] 284
# all values from 5th column (gestdays);
babies[,5];
   [1] 284 282 279 999 282 286 244 245 289 299 351 282 279 281 273 285 255 261 261
  [20] 288 270 274 287 276 294 261 280 266 292 274 270 278 268 275 281 283 279 288
...
[1217] 266 319 285 321 284 290 288 262 281 287 244 278 276 290 270 275 265 291 281
[1236] 297
```

babies

```
summary(babies[5]);
 gestdays
 Min.   :148.0
 1st Qu.:272.0
 Median :280.0
 Mean   :286.9
 3rd Qu.:288.0
 Max.   :999.0
```

Index numbers and sequences

How can we get the odd-numbered individuals in *babies*?

1. remember to use brackets, AND we have two dimensions

2. remember to use sequence (seq) INSIDE the brackets

```
# put (nest) that list inside the brackets
babies[seq(1,10, by = 2),]
   id pluralty outcome date gestdays sex bwt parity mrace mage med mht mwt frace
1  15        5       1 1411      284   1 120      1     8   27   5  62 100     8
3  58        5       1 1576      279   1 128      1     0   28   2  64 115     5
5  72        5       1 1425      282   1 108      1     0   23   5  67 125     0
7 102        5       1 1449      244   1 138      4     7   33   2  62 178     7
9 142        5       1 1408      289   1 120      3     0   25   4  62 125     3
  fage fed fht fwt marital inc msmoke time number
1   31   5  65 110       1   1      0    0      0
3   32   1  99 999       1   2      1    1      1
5   24   5  99 999       1   1      1    1      5
7   37   4  99 999       1  98      0    0      0
9   26   1  70 180       0   2      0    0      0
```

Homework 2

The rivers data set is built-in to R

```
rivers
```

Problems on page 45 of VCH2

2.1

2.2 (think)

2.3

2.5

2.7

Due before class 1/30

Email to me following the same procedures and format as for Homework 1.