431 Class 10

Thomas E. Love

2017-09-28

Today's Agenda

- The Western Collaborative Group Study
  - Building Tables Effectively
  - Dealing with Missingness
  - Scatterplot and Correlation Matrices

Today's R Setup

```r
library(Epi); library(viridis)
library(GGally); library(mice)
library(tidyverse)

source("Love-boost.R")
```

The Western Collaborative Group Study

Original description: 3524 men aged 39-59 and employed in the San Francisco Bay or Los Angeles areas were enrolled in 1960 and 1961. In addition to determinations of behavior pattern, the initial examination included medical and parental history, socioeconomic factors, exercise, diet, smoking, alcohol consumption, diet, serum lipid and lipoprotein studies, blood coagulation studies, and cardiovascular examination.

- http://www.epi.umn.edu/cvdepi/study-synopsis/western-collaborative-group-study/

The WCGS data describe 3,154 subjects and 22 variables. For now, let's examine a few interesting variables, and a sample of 500 of the observations.

```
wcgs.full <- read.csv("wcgs.csv")
wcgs.full <- tbl_df(wcgs.full)
```

Select the variables of interest, and sample 500 subjects

```r
set.seed(43101)
wcgs1 <-
  wcgs.full %>%
  select(id, age, chol, arcus, dibpat, bmi,
         wghtcat, smoke, ncigs, chd69) %>%
  sample_n(500, replace = FALSE)
```

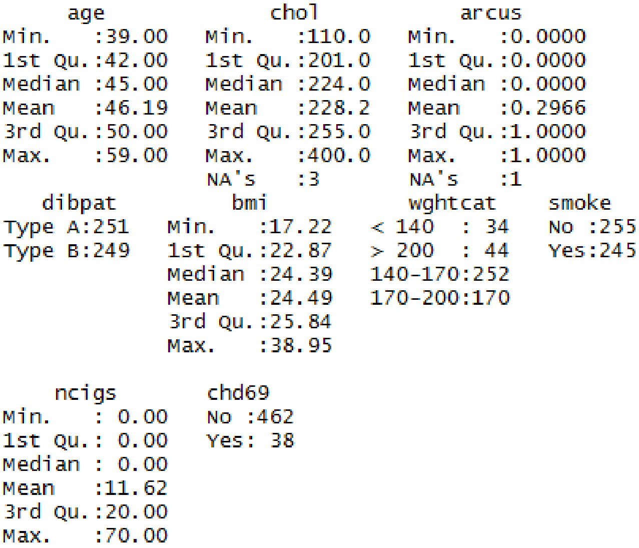
Resulting tibble

```
# A tibble: 500 x 10
      id   age  chol arcus dibpat        bmi wghtcat
   <int> <int> <int> <int> <fctr>      <dbl>  <fctr>
 1  6039    42   218     0 Type A 25.05680 140-170
 2  3713    44   207     0 Type B 22.87093 170-200
 3  3465    50   249     0 Type A 24.38980 140-170
 4  3923    42   236     0 Type B 25.10714 170-200
 5  3503    46   247     1 Type A 21.47629 140-170
 6 13167    40   206     0 Type B 26.45372 170-200
 7 11325    39   267     1 Type B 23.56510 140-170
 8  3800    51   214     0 Type A 23.67245 140-170
 9 12582    43   235     1 Type A 19.25676 140-170
10 19096    48   234     0 Type B 28.12608 170-200
# ... with 490 more rows, and 3 more variables:
#   smoke <fctr>, ncigs <int>, chd69 <fctr>
```

Codebook

Name	Stored As	Type	Details (units, levels, etc.)
`id`	integer	(nominal)	ID #, nominal and uninteresting
`age`	integer	quantitative	age, in years - no decimal places
`chol`	integer	quantitative	total cholesterol, mg/dL
`arcus`	integer	(nominal)	arcus senilis present (1) or absent (0)
`dibpat`	factor (2)	(binary)	behavioral pattern: A or B
`bmi`	number	quantitative	body-mass index
`wghtcat`	factor (4)	(ordinal)	wt: < 140, 140-170, 170-200, > 200
`smoke`	factor (2)	(binary)	cigarette smoker: Yes or No
`ncigs`	integer	quantitative	number of cigarettes smoked per day
`chd69`	factor (2)	(binary)	CHD event: Yes or No

Summary of this sample (without `id`)



```
     age             chol            arcus
 Min.   :39.00   Min.   :110.0   Min.   :0.0000
 1st Qu.:42.00   1st Qu.:201.0   1st Qu.:0.0000
 Median :45.00   Median :224.0   Median :0.0000
 Mean   :46.19   Mean   :228.2   Mean   :0.2966
 3rd Qu.:50.00   3rd Qu.:255.0   3rd Qu.:1.0000
 Max.   :59.00   Max.   :400.0   Max.   :1.0000
                 NA's   :3       NA's   :1
   dibpat          bmi              wghtcat      smoke
 Type A:251   Min.   :17.22   < 140  : 34    No :255
 Type B:249   1st Qu.:22.87   > 200  : 44    Yes:245
              Median :24.39   140-170:252
              Mean   :24.49   170-200:170
              3rd Qu.:25.84
              Max.   :38.95

    ncigs          chd69
 Min.   : 0.00   No :462
 1st Qu.: 0.00   Yes: 38
 Median : 0.00
 Mean   :11.62
 3rd Qu.:20.00
 Max.   :70.00
```

```r
summary(wcgs1)[,-1]
```

A Key Research Question

Were the men with Type A behavioral patterns more likely to suffer a CHD event?

```r
table(wcgs1$dibpat, wcgs1$chd69)
```

```
          No Yes
  Type A 222  29
  Type B 240   9
```

What's not so great about this table?

Re-specify the factor to re-order of the levels

```r
wcgs1$chdevent <-
  factor(wcgs1$chd69, levels = c("Yes", "No"))
tab1 <- table(wcgs1$dibpat, wcgs1$chdevent)
tab1
```

```
         Yes  No
  Type A  29 222
  Type B   9 240
```

and this is **standard epidemiological format**.

Aha! A Two-by-Two Table!

```r
twoby2(tab1) ## twoby2 is part of the Epi package
```

```
2 by 2 table analysis:
------------------------------------------------------
Outcome   : Yes
Comparing : Type A vs. Type B

       Yes  No    P(Yes) 95% conf. interval
Type A  29 222    0.1155    0.0815   0.1613
Type B   9 240    0.0361    0.0189   0.0680

                                   95% conf. interval
             Relative Risk: 3.1965    1.5450   6.6134
         Sample Odds Ratio: 3.4835    1.6132   7.5221
Conditional MLE Odds Ratio: 3.4754    1.5589   8.5398
    Probability difference: 0.0794    0.0334   0.1279
```

A 4 by 2 table

```r
table(wcgs1$wghtcat, wcgs1$chdevent)
```

```
        Yes  No
< 140     1  33
> 200     3  41
140-170  20 232
170-200  14 156
```

Why is this a poor table?

An Improved 4 x 2 table

```r
wcgs1$wghtcat <- factor(wcgs1$wghtcat,
       levels = c("< 140", "140-170", "170-200", "> 200"))
tab2 <- table(wcgs1$wghtcat, wcgs1$chdevent)
knitr::kable(tab2)
```

	Yes	No
< 140	1	33
140-170	20	232
170-200	14	156
> 200	3	41

A Three-Way Table (Not Flattened)

```
table(wcgs1$dibpat, wcgs1$wghtcat, wcgs1$chdevent)
```

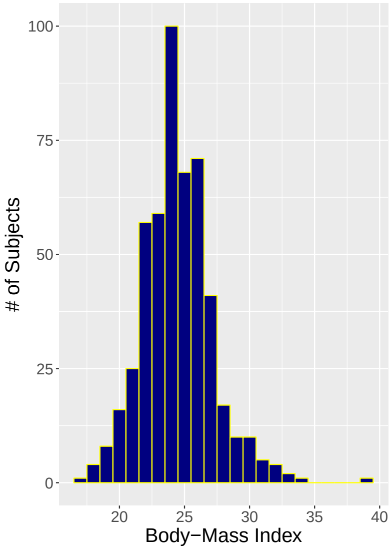
```
, ,  = Yes


         < 140 140-170 170-200 > 200
  Type A     0      15      13     1
  Type B     1       5       1     2

, ,  = No


         < 140 140-170 170-200 > 200
  Type A    15     104      82    21
  Type B    18     128      74    20
```

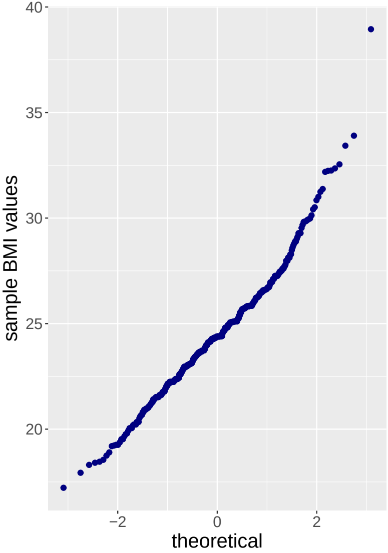
A Three-Way Table (Flattened)

```
ftable(wcgs1$dibpat, wcgs1$wghtcat, wcgs1$chdevent)
```

```
                Yes  No

Type A < 140      0  15
       140-170   15 104
       170-200   13  82
       > 200      1  21
Type B < 140      1  18
       140-170    5 128
       170-200    1  74
       > 200      2  20
```

How can we describe the distribution of BMI?





Summary of BMI data

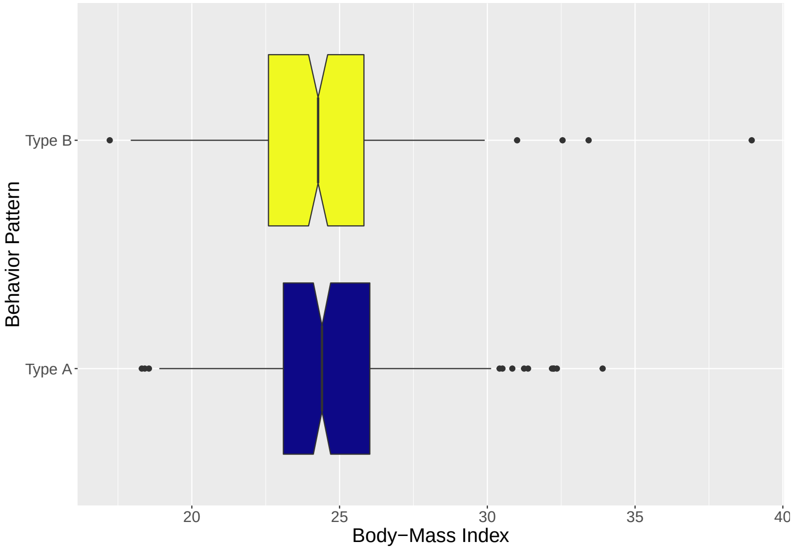
```
mosaic::favstats(wcgs1$bmi)
```

```
      min       Q1  median       Q3      max     mean
 17.22242 22.87091 24.3898 25.84016 38.94737 24.48819
       sd   n missing
 2.680243 500       0
```

```
psych::describe(wcgs1$bmi)
```

```
   vars   n  mean   sd median trimmed  mad   min   max
X1    1 500 24.49 2.68  24.39   24.39 2.15 17.22 38.95
   range skew kurtosis   se
X1 21.72 0.65     2.11 0.12
```

Comparing BMI by Behavior Pattern



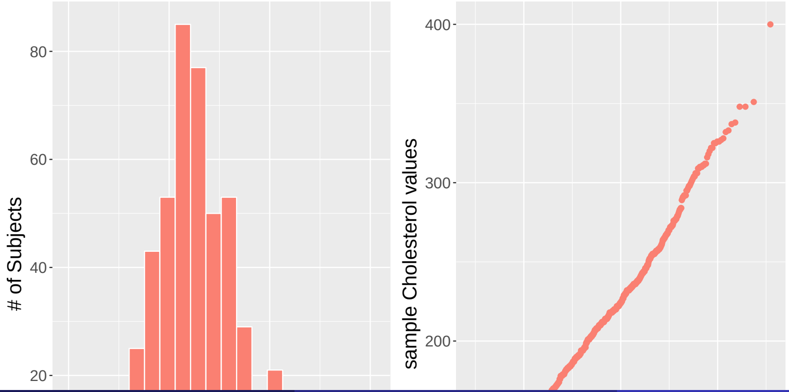
Numerical BMI summaries, by behavior pattern

```
by(wcgs1$bmi, wcgs1$dibpat, mosaic::favstats)
```

```
wcgs1$dibpat: Type A
      min       Q1   median       Q3      max     mean
 18.30729 23.09703 24.40514 26.02431 33.90239 24.64689
       sd   n missing
 2.700341 251       0
-----------------------------------------
wcgs1$dibpat: Type B
      min       Q1   median       Q3      max     mean
 17.22242 22.59412 24.27378 25.82449 38.94737 24.32823
      sd   n missing
 2.65565 249       0
```

How about Total Cholesterol, instead?

```
Warning: Removed 3 rows containing non-finite values
(stat_bin).

Warning: Removed 3 rows containing non-finite values
(stat_qq).
```



Why did we get error warnings?

```
mosaic::favstats(wcgs1$chol)
```

```
 min  Q1 median  Q3 max    mean       sd   n missing
 110 201    224 255 400 228.167 44.20081 497       3
```

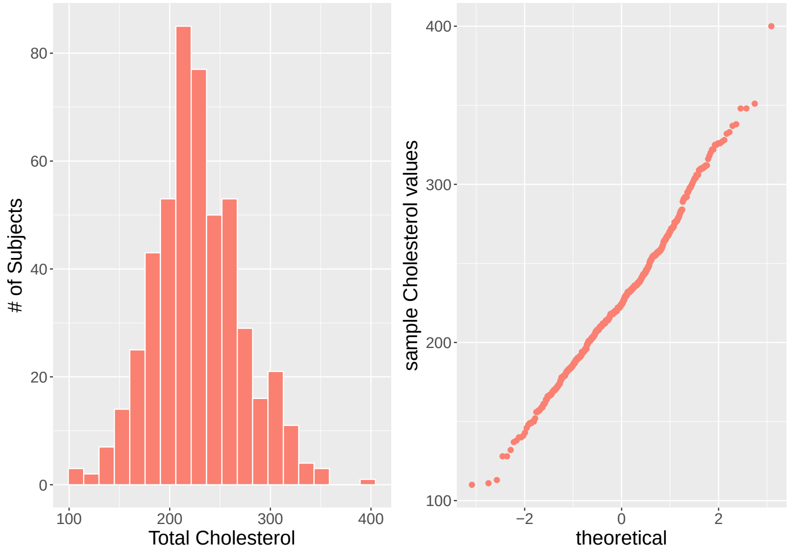
```
psych::describe(wcgs1$chol)
```

```
   vars   n   mean   sd median trimmed   mad min max
X1    1 497 228.17 44.2    224  226.64 41.51 110 400
   range skew kurtosis   se
X1   290 0.33     0.36 1.98
```

What should we do about missing values?

1. ggplot2 doesn't include missing values in the plot, but it does warn that they've been removed.
   - We could suppress that warning by setting `na.rm = TRUE` in the call to a geom like `geom_histogram` or `geom_qq`.

Plot with `na.rm = TRUE`



Classification of Missing Data

There are three classifications we will think about in 431. Subtleties abound, but these three will suffice for most practical work.

- MCAR: Missing *Completely at Random*. This is the desirable scenario for us. MCAR means that there is no relationship between the probability of a data point being missing, and any values in the data set, either missing or observed.
    - The missing data are just a random subset of the data.
    - This is one kind of "ignorable" missingness.

Classification of Missing Data

- MAR: Missing *at Random*, which is definitely an unfortunate name. Less desirable, but there is still some hope. MAR means that the probability of a data point being missing has nothing to do with the missing value that would have been observed, but does have something to do with the values of some other variable that you did observe.
  - The idea is that if we can control for this other variable in our analysis, then we can treat this missingness as just a random subset of the data after that adjustment, which will eventually be pretty straightforward.
  - This is another kind of "ignorable" missingness.

Classification of Missing Data

- MNAR: Missing *not at Random* is a more serious issue. Now, additional thought and some special methods may well be required. Here, there is a relationship between the probability that a value is missing and what the actual (missing) value is.
  - This is what we mean by "non-ignorable" missingness.
  - Multiple Imputation methods (and Maximum Likelihood approaches) assume the data are at least MAR, so that's the important distinction to make, generally.
- Some of these explanations come from `this link`

What should we do about missing values?

2. At times you will want to try to understand what makes observations with missing values different from observations with meaningful recorded values, especially if we're thinking that the missing mechanism is MCAR or MAR.
    - We might, for instance, compare the BMI values or perhaps the smoking status for those with and without missing cholesterol values, using the `is.na()` function to make a new variable to indicate those subjects without a cholesterol level.

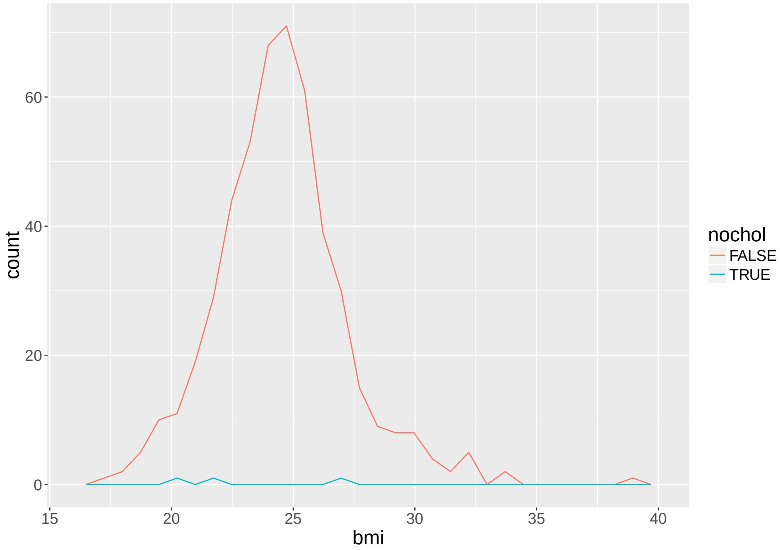
- Some of this material is drawn from `R for Data Science`

Do those missing cholesterol look unusual in terms of BMI?

First, we'll build a new (logical) variable (TRUE/FALSE) to indicate a missing cholesterol level, and then we'll plot the BMI distributions for each level of the new variable.

```
wcgs1 %>%
  mutate(
    nochol = is.na(chol)
  ) %>%
  ggplot(aes(x = bmi)) +
  geom_freqpoly(aes(col = nochol), bins = 30) +
  theme(text = element_text(size = 18))
```

Do the BMIs of people without chol look different?



Are the people without a cholesterol value unusual in terms of their smoking status?

```r
temp1 <- table(is.na(wcgs1$chol), wcgs1$smoke)
knitr::kable(temp1)
```

	No	Yes
FALSE	253	244
TRUE	2	1

What should we do about missing values?

3. How many missing values are there?
    - If the missing values are more than, say, 5% of a variable, we're going to need some strong, almost heroic assumptions in order to feel confident about using such a variable in building a model or making an inference.
    - If the amount of missing data is very small relative to the size of the data as a whole, then leaving out a few samples and just running models or comparisons ignoring those observations may not be too damaging.
    - Depending on the situation, you may want to look for other fixes besides just dropping these cases and wiping out potentially useful data.

- Some of this material comes from `this R-bloggers post`

What should we do about missing values?

Could we **impute** missing values?

- One approach is *simple* imputation, where a single value is created to "fill in" the missing observation. This is pretty easy to do, but very rarely a good idea.
  - Rarely, substituting the mean is a reasonable thing to do, as it reduces variance in your estimate of the distribution, among other problems.
  - Sometimes, but still pretty rarely, substituting in a random value observed in the rest of the data set is a reasonable thing to do.
  - Better, although still problematic, imputation approaches use other variables in the data set to predict the missing value, and contain a random component. Using other variables preserves the relationships among variables in the imputations. The random component is important so that all missing values of a single variable are not all exactly equal. One example would be to use a regression equation to predict missing values, then add a random error term.
- See http://www.theanalysisfactor.com/multiple-imputation-in-a-nutshell/

What's so bad about simple imputation?

Although there are several simple imputation approaches that solve many of the problems inherent in mean imputation, one problem remains. Because the imputed value is an estimate - a predicted value - there is uncertainty about its true value. Every statistic has uncertainty, measured by its standard error. Statistics computed using imputed data have even more uncertainty than its standard error measures. Your statistical package cannot distinguish between an imputed value and a real value.

Since the standard errors of statistics based on imputed values, such as sample means or regression coefficients, are too small, corresponding reported p-values are also too small. P-values that are reported as smaller than they, in reality, are, lead to Type I errors.

- It turns out that *multiple* imputation is a much better approach.
- Various types of "hot deck" procedures can help, too. See the `HotDeckImputation` package in R, or `this link`

So what is multiple imputation?

Multiple imputation has solved this problem by incorporating the uncertainty inherent in imputation. It has four steps:

1. Create *m* sets of imputations for the missing values using an imputation process with a random component.
2. The result is *m* full data sets. Each data set will have slightly different values for the imputed data because of the random component.
3. Analyze each completed data set. Each set of parameter estimates will differ slightly because the data differs slightly.
4. Combine results, calculating the variation in parameter estimates.

Multiple Imputation is amazing

Remarkably, m, the number of sufficient imputations, can be only 5 to 10 imputations, although it depends on the percentage of data that are missing. The result is unbiased parameter estimates and a full sample size, when done well.

Doing multiple imputation well, however, is not always quick or easy. First, it requires that the missing data be ignorable. Second, it requires a very good imputation model. Creating a good imputation model requires knowing your data very well and having variables that will predict missing values.

Source:
http://www.theanalysisfactor.com/multiple-imputation-in-a-nutshell/

What will we do in 431?

- Often, we'll be willing to simply exclude the data with missing values from our graphs or other analyses.
- Sometimes, we'll be willing to assume (heroically) that the data are missing at random and we'll use a simple imputation approach, via the `mice` package.
- Later in the term (and definitely in 432) we'll move on up to multiple imputation, using `mice` sometimes and `Hmisc` at other times.

Using mice to build imputations for chol

```r
mice::md.pattern(wcgs1)
```

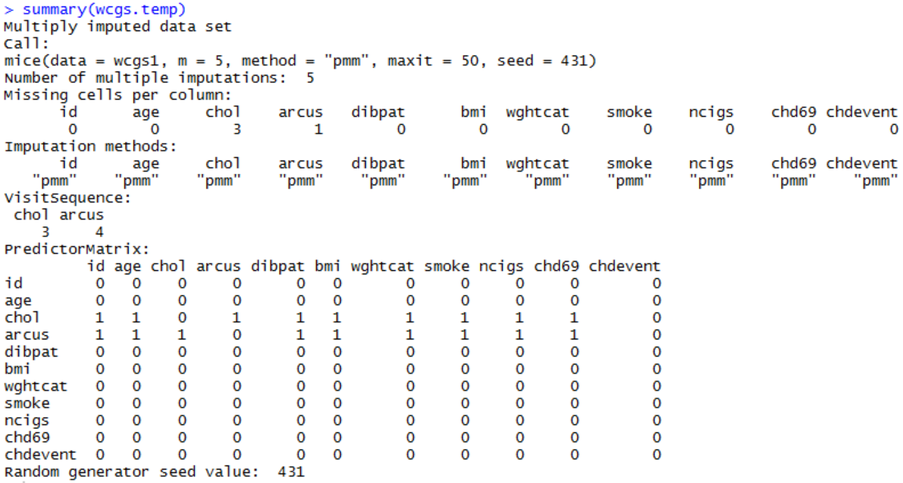
```
    id age dibpat bmi wghtcat smoke ncigs chd69
496  1   1      1   1       1     1     1     1
  3  1   1      1   1       1     1     1     1
  1  1   1      1   1       1     1     1     1
     0   0      0   0       0     0     0     0
    chdevent arcus chol
496        1     1    1 0
  3        1     1    0 1
  1        1     0    1 1
           0     1    3 4
```

Build 5 actual imputations using the "predictive mean matching" (pmm) approach

```
wcgs.temp <- mice(wcgs1,m=5,maxit=50,meth='pmm',seed=431)
```

```
 iter imp variable
  1    1  chol  arcus
  1    2  chol  arcus
  1    3  chol  arcus
  1    4  chol  arcus
  1    5  chol  arcus
  2    1  chol  arcus
  2    2  chol  arcus
  2    3  chol  arcus
  2    4  chol  arcus
  2    5  chol  arcus
  3    1  chol  arcus
```

View imputation results, summarized



```
> summary(wcgs.temp)
Multiply imputed data set
Call:
mice(data = wcgs1, m = 5, method = "pmm", maxit = 50, seed = 431)
Number of multiple imputations:  5
Missing cells per column:
      id      age     chol    arcus   dibpat      bmi  wghtcat    smoke    ncigs    chd69 chdevent
       0        0        3        1        0        0        0        0        0        0        0
Imputation methods:
      id      age     chol    arcus   dibpat      bmi  wghtcat    smoke    ncigs    chd69 chdevent
   "pmm"    "pmm"    "pmm"    "pmm"    "pmm"    "pmm"    "pmm"    "pmm"    "pmm"    "pmm"    "pmm"
VisitSequence:
 chol arcus
    3     4
PredictorMatrix:
         id age chol arcus dibpat bmi wghtcat smoke ncigs chd69 chdevent
id        0   0    0     0      0   0       0     0     0     0        0
age       0   0    0     0      0   0       0     0     0     0        0
chol      1   1    0     1      1   1       1     1     1     1        0
arcus     1   1    1     0      1   1       1     1     1     1        0
dibpat    0   0    0     0      0   0       0     0     0     0        0
bmi       0   0    0     0      0   0       0     0     0     0        0
wghtcat   0   0    0     0      0   0       0     0     0     0        0
smoke     0   0    0     0      0   0       0     0     0     0        0
ncigs     0   0    0     0      0   0       0     0     0     0        0
chd69     0   0    0     0      0   0       0     0     0     0        0
chdevent  0   0    0     0      0   0       0     0     0     0        0
Random generator seed value:  431
```

Inspect the imputed values, if you like

```
wcgs.temp$imp$chol
```

```
      1   2   3   4   5
62  137 268 211 269 224
237 212 244 295 258 282
472 255 205 252 205 178
```

```
wcgs.temp$imp$arcus
```

```
    1 2 3 4 5
480 0 1 0 0 1
```

Simple Imputation: Complete data with, let's say, the fourth of the five imputations we built

```
completedwcgs1 <- mice::complete(wcgs.temp,4)
```

favstats with and without imputation

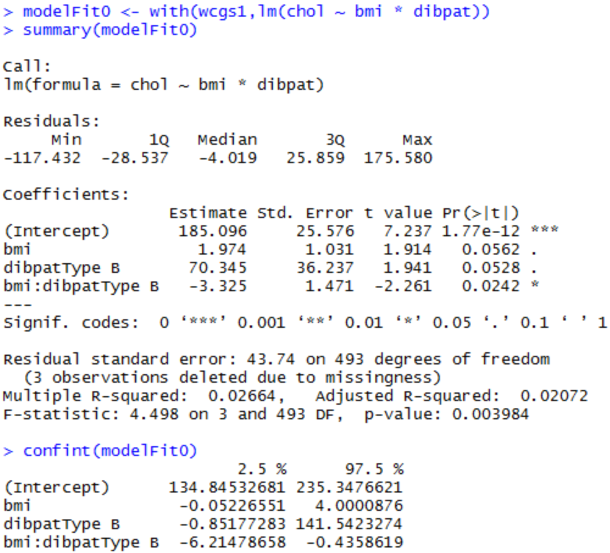
```
mosaic::favstats(wcgs1$chol)
```

```
 min  Q1 median  Q3 max    mean       sd   n missing
 110 201    224 255 400 228.167 44.20081 497       3
```

```
mosaic::favstats(completedwcgs1$chol)
```

```
 min  Q1 median  Q3 max    mean       sd   n missing
 110 201    224 255 400 228.262 44.13794 500       0
```

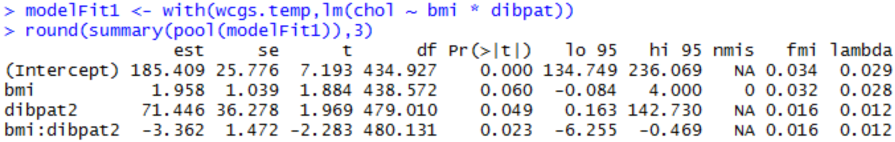
Build a Linear Model without imputation



```
> modelFit0 <- with(wcgs1,lm(chol ~ bmi * dibpat))
> summary(modelFit0)

Call:
lm(formula = chol ~ bmi * dibpat)

Residuals:
     Min       1Q   Median       3Q      Max 
-117.432  -28.537   -4.019   25.859  175.580 

Coefficients:
                Estimate Std. Error t value Pr(>|t|)    
(Intercept)      185.096     25.576   7.237 1.77e-12 ***
bmi                1.974      1.031   1.914   0.0562 .  
dibpatType B      70.345     36.237   1.941   0.0528 .  
bmi:dibpatType B  -3.325      1.471  -2.261   0.0242 *  
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 43.74 on 493 degrees of freedom
  (3 observations deleted due to missingness)
Multiple R-squared:  0.02664,	Adjusted R-squared:  0.02072 
F-statistic: 4.498 on 3 and 493 DF,  p-value: 0.003984

> confint(modelFit0)
                        2.5 %      97.5 %
(Intercept)      134.84532681 235.3476621
bmi               -0.05226551   4.0000876
dibpatType B      -0.85177283 141.5423274
bmi:dibpatType B  -6.21478658  -0.4358619
```

Multiple imputation and pooling

Suppose that the next step in our analysis is to fit a linear model to the data. You may ask what imputed data set to choose. The `mice` package makes it again very easy to fit a a model to each of the imputed data sets and then pool the results together



```
> modelFit1 <- with(wcgs.temp,lm(chol ~ bmi * dibpat))
> round(summary(pool(modelFit1)),3)
                est     se      t      df Pr(>|t|)   lo 95   hi 95 nmis   fmi lambda
(Intercept) 185.409 25.776  7.193 434.927    0.000 134.749 236.069   NA 0.034  0.029
bmi           1.958  1.039  1.884 438.572    0.060  -0.084   4.000    0 0.032  0.028
dibpat2      71.446 36.278  1.969 479.010    0.049   0.163 142.730   NA 0.016  0.012
bmi:dibpat2  -3.362  1.472 -2.283 480.131    0.023  -6.255  -0.469   NA 0.016  0.012
```

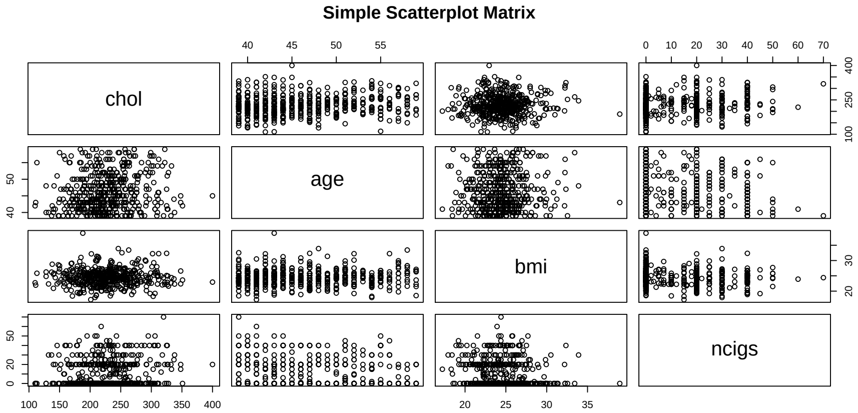
Details of linear model after pooling

`modelFit1` contains the results of the fitting performed over the imputed data sets, while the `pool()` function pools them all together.

- `fmi` = fraction of missing information
- `lambda` = proportion of total variance attributable to the missing data
- Note that if we were looking at a strict alpha of 0.05, we'd have a significant `dibpat2` main effect now, when we didn't before.

Multivariable Descriptions: A Scatterplot Matrix

```
pairs (~ chol + age + bmi + ncigs,
       data=wcgs1, main="Simple Scatterplot Matrix")
```



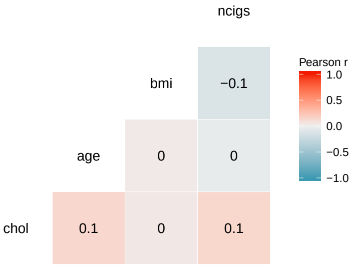
Correlation Matrix (after imputation)

```
completedwcgs1 <- mice::complete(wcgs.temp,1)
round(cor(completedwcgs1[c("chol", "age", "bmi")]),3)
```

```
      chol   age   bmi
chol 1.000 0.133 0.036
age  0.133 1.000 0.029
bmi  0.036 0.029 1.000
```

Using GGally for a Correlation Matrix

```
tempdat <- completedwcgs1 %>%
  select(chol, age, bmi, ncigs)

ggcorr(tempdat, name = "Pearson r", label = TRUE)
```

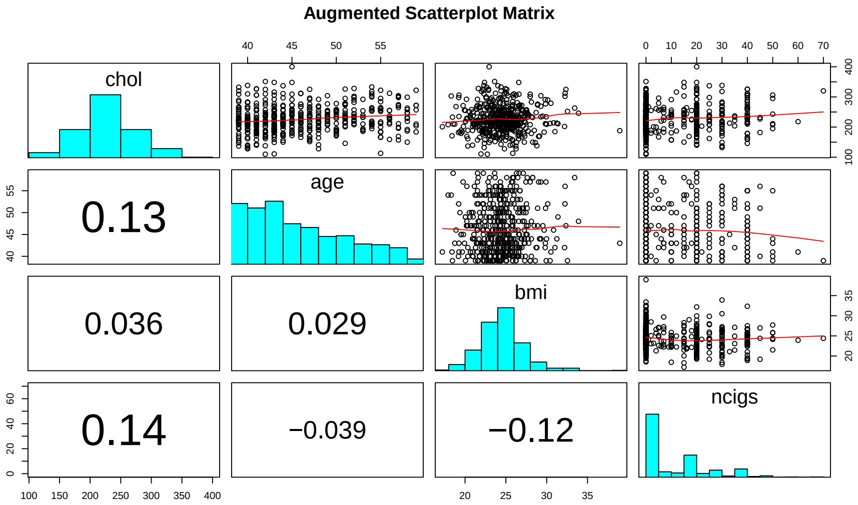


My Favorite Scatterplot Matrix

My favorite way to augment this plot adds loess smooths to the upper panel, and correlations in the lower panel, with histograms down the diagonal. To do this, we first create two functions (these modifications come from Chang's R Graphics Cookbook), called `panel.hist` and `panel.cor`.

These functions are in the Love-boost.R script.

Augmented Scatterplot Matrix



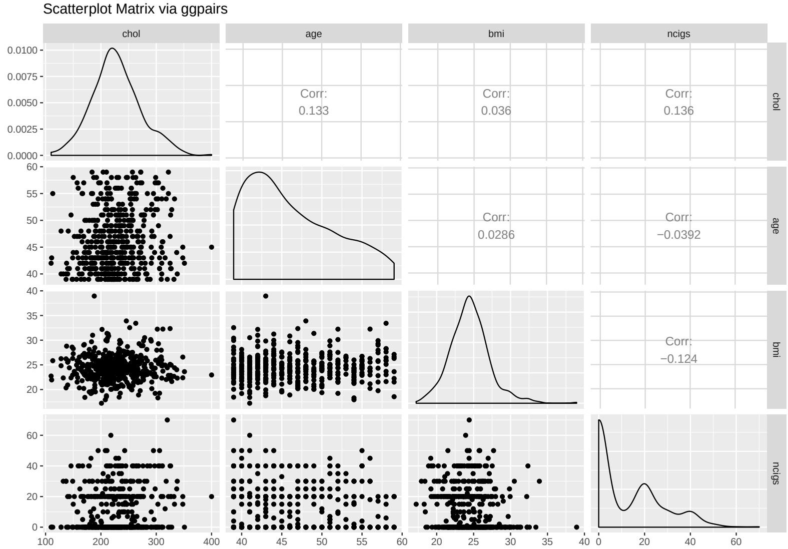
Code for Augmented Scatterplot Matrix

```
pairs (~ chol + age + bmi + ncigs, data=completedwcgs1,
       main="Augmented Scatterplot Matrix",
       upper.panel = panel.smooth,
       diag.panel = panel.hist,
       lower.panel = panel.cor)
```

Using GGally for a Scatterplot Matrix (Code)

```r
tempdat <- completedwcgs1 %>%
  select(chol, age, bmi, ncigs)

ggpairs(tempdat, title = "Scatterplot Matrix via ggpairs")
```

Using GGally for a Scatterplot Matrix (Result)



For Tuesday: Working with Tables

For Class 11, I'd like you to read a 1981 paper by A.S.C. Ehrenberg that you'll find linked on the Class 10 README.

The paper is called *The Problem of Numeracy* and it provides some very helpful tips for working with tables, in particular.

There are three key tips related to the development of tables, in practice, as described by Ehrenberg, and also by Howard Wainer[1] who concisely states them as:

1. Order the rows and columns in a way that makes sense.
2. Round - a lot!
3. ALL is different and important.

[1] Visual Revelations (1997), Chapter 10.