An Introduction to Missing Data

Luke Miratrix with Lily Bliznashka and Maxime Rischard

2019-11-17 11:23:34

Introduction

Handling missing data is the icky, unglamorous part of any statistical analysis. It is where the skeletons lie. There's a range of options available, which are, broadly speaking:

1. Delete the observations with missing covariates (this is a "complete case analysis")
2. Plug in some kind of reasonable value for the missing covariate. This is called "imputation." We discuss three ways of doing this that are increasingly sophisticated and layered on each other:

a. Mean imputation. Simply take the mean of all the observations where you know the value, and then use that for anything that is missing.
b. Regression imputation. You generate regression equations describing how all the variables are connected, and use those to predict any missing value.
c. Stochastic Regression imputation. Here we use regression imputation, but we also add some residual noise to all our imputed values so that our imputed values have as much variation as our actual values (otherwise our imputed values will tend to be all clumped together).

3. Multiply impute the missing data, by fully modelling the covariate and the missingness, and generating a range of complete datasets under this model. Here you end up with a bunch of complete datasets that are all "reasonable guesses" as to what the full dataset might have been. You then analyze each one, and aggregate your findings across them to get a final answer.

The first two general approaches are imperfect, while the third is often more work than the original analysis that we were hoping to perform. For this course, doing a 2a, 2b, or 2c are all reasonable choices. If you have very little missing data you can often get away with 1. We have no expectations that people will take the plunge into #3 (multiple imputation). In real life, people will often analyze their data with a complete case analysis and some other strategy, and then compare the results. In Education, if missingness is below 10% people usually just do mean imputation, but regression imputation would probably be superior.

This handout provides an introduction to missing data, and includes a few commands to explore and deal with missing data. In this document we first talk about exploring missing data (in particular getting plots that show you if you have any notable patterns in how things are missing) and then we give a brief walk-through of the 3 methods listed above.

We will the `mice` and `VIM` packages, which you can install using `install.packages()` if you have not yet done so. These are simple and powerful packages for visualizing and imputing missing data. At the end of this document we also describe the `Amelia` package.

```r
library( tidyverse )
library(mice)
library(VIM)
```

Throughout we use a small built in R dataset on air quality as a working example.

```r
  data(airquality)
  nrow(airquality)
```

```
## [1] 153
```

```r
  head(airquality)
```

```
##   Ozone Solar.R Wind Temp Month Day
## 1    41     190  7.4   67     5   1
## 2    36     118  8.0   72     5   2
## 3    12     149 12.6   74     5   3
## 4    18     313 11.5   62     5   4
## 5    NA      NA 14.3   56     5   5
## 6    28      NA 14.9   66     5   6
```

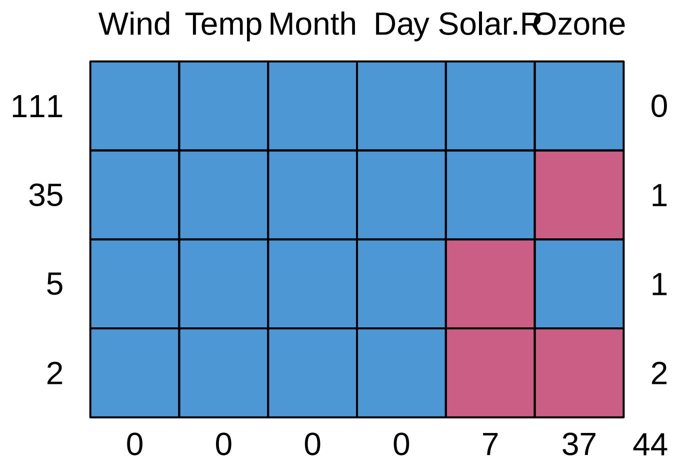
```
summary(airquality[1:4])
```

```
##      Ozone           Solar.R          Wind             Temp
##  Min.   :  1.0   Min.   :  7   Min.   : 1.70   Min.   :56.0
##  1st Qu.: 18.0   1st Qu.:116   1st Qu.: 7.40   1st Qu.:72.0
##  Median : 31.5   Median :205   Median : 9.70   Median :79.0
##  Mean   : 42.1   Mean   :186   Mean   : 9.96   Mean   :77.9
##  3rd Qu.: 63.2   3rd Qu.:259   3rd Qu.:11.50   3rd Qu.:85.0
##  Max.   :168.0   Max.   :334   Max.   :20.70   Max.   :97.0
##  NA's   :37      NA's   :7
```

Visualizing missing data

Just like with anything in statistics, the first thing to do is to look at our data. We want to know which variables are often missing, and if some variables are often missing together. We also want to know how much data is missing. The mice package has a variety of plots to show us patterns of missingness:

```
md.pattern(airquality)
```



```
##     Wind Temp Month Day Solar.R Ozone
## 111    1    1     1   1       1     1  0
## 35     1    1     1   1       1     0  1
## 5      1    1     1   1       0     1  1
## 2      1    1     1   1       0     0  2
##        0    0     0   0       7    37 44
```

This plot gives us the different missing data patterns and the number of observations that have each mising data pattern. For example, the second row in the plot says there are 35 observations that have a missing data pattern where only Ozone is missing.

Easier to understand patterns!

We can also just look at 10 observations to see everything that is going on. Here we take the first 10 rows of our dataset, but could also take a random 10 row with the tidyverse's `sample_n` method.

```
airqualitysub = airquality[1:10, ]
airqualitysub
```

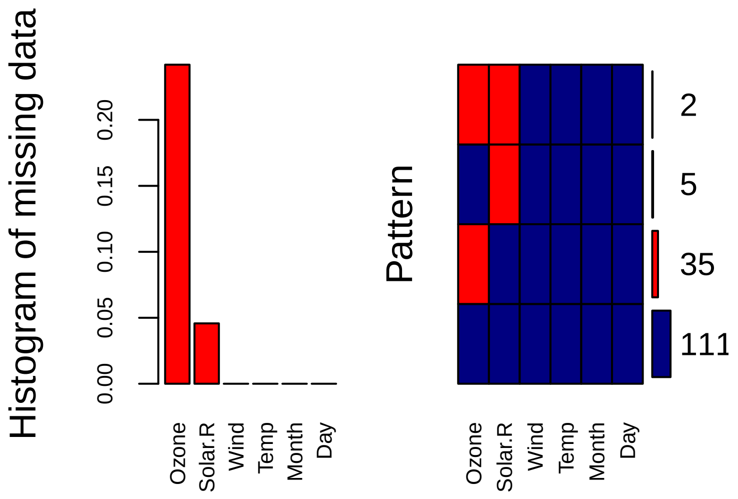
```
##    Ozone Solar.R Wind Temp Month Day
## 1     41     190  7.4   67     5   1
## 2     36     118  8.0   72     5   2
## 3     12     149 12.6   74     5   3
## 4     18     313 11.5   62     5   4
## 5     NA      NA 14.3   56     5   5
## 6     28      NA 14.9   66     5   6
## 7     23     299  8.6   65     5   7
## 8     19      99 13.8   59     5   8
## 9      8      19 20.1   61     5   9
## 10    NA     194  8.6   69     5  10
```

We see that we have one observation missing two covariates and one each of missing Ozone only and Solar.R only.

The VIM Package

The VIM package gives some alternate plots to explore missing data patterns. For example, `aggr()`:

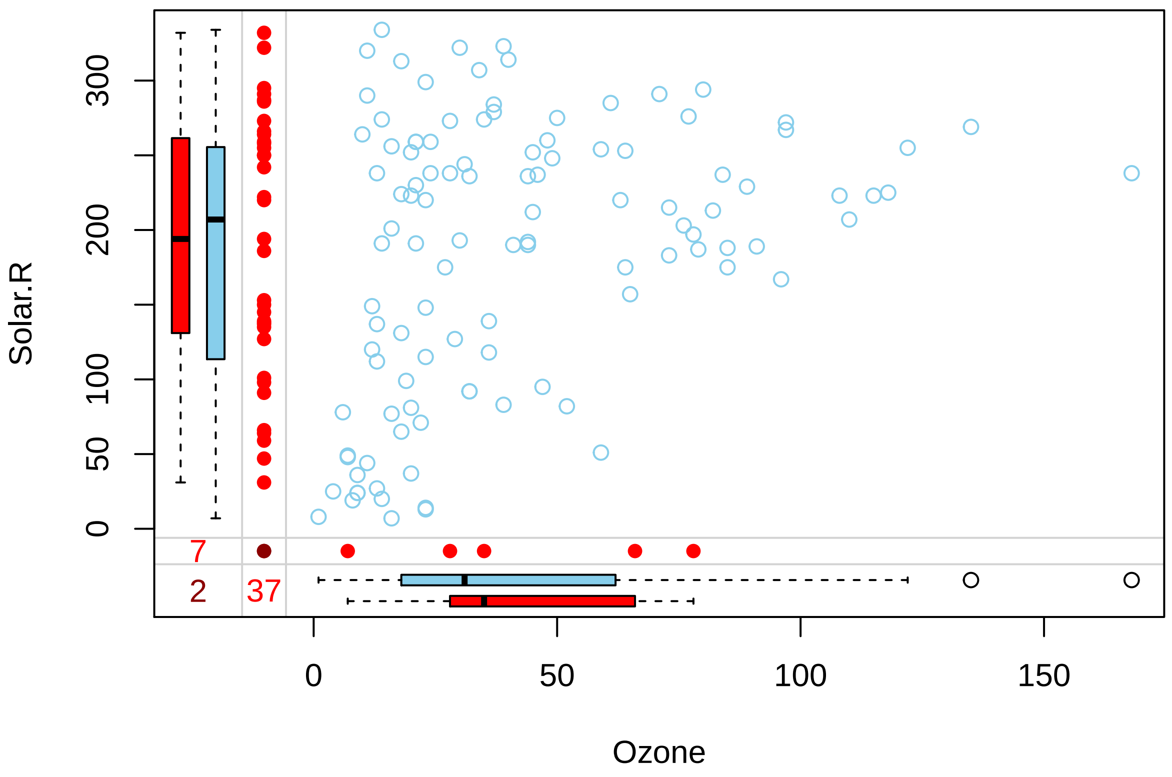
```r
aggr(airquality, col=c('navyblue','red'),
     numbers=TRUE, sortVars=TRUE, labels=names(data),
     cex.axis=.7, gap=3, prop=c(TRUE, FALSE),
     ylab=c("Histogram of missing data","Pattern"))
```



```
##
##  Variables sorted by number of missings:
##  Variable  Count
##     Ozone 0.2418
##   Solar.R 0.0458
##      Wind 0.0000
##      Temp 0.0000
##     Month 0.0000
##       Day 0.0000
```

On the left, we have the proportion of missing data for each variable in our dataset. We can see that Ozone and Solar.R have missing values. On the right, we have the joint distribution of missingness. We can see that 111 observations have no missing values. From those with missing values, the majority have missing values for Ozone, some have missing values for Solar.R and only 2 observations have missing values for both Ozone and Solar.R.

```r
marginplot(airquality[1:2])
```



Here we have a scatterplot for the first two variables in our dataset: Ozone and Solar.R. These are the variables that have missing data. In addition to the standard scatterplot we are familir with, information about missingness is shown in the margins. The red dots indicate observations with one of both values missing (so there can be a bunch of dots stacked up in the bottom-left corner). The numbers (37, 7, and 2 tells us how many observations are missing either or both of these variables).

Complete case analysis

Working with complete cases (dropping observations with any missing data on our outcome and predictors) is always an option. We have been doing this in class and section. However, this can lead to substantial data loss, if we have a lot of missingness and it can heavily bias our results depending on why observations are missing.

Complete case analysis is the R default.

```
fit <- lm(Ozone ~ Wind, data = airquality )
summary(fit)
```

```
##
## Call:
## lm(formula = Ozone ~ Wind, data = airquality)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -51.57 -18.85  -4.87  15.23  90.00
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)    96.87       7.24   13.38  < 2e-16 ***
## Wind           -5.55       0.69   -8.04  9.3e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 26.5 on 114 degrees of freedom
##   (37 observations deleted due to missingness)
## Multiple R-squared:  0.362,  Adjusted R-squared:  0.356
## F-statistic: 64.6 on 1 and 114 DF,  p-value: 9.27e-13
```

Note the listing in the summary of number of items deleted. You can find out which rows were deleted:

```
# which rows/observations were deleted
deleted <- na.action(fit)
deleted
```

```
##   5  10  25  26  27  32  33  34  35  36  37  39  42  43  45  46  52  53
##   5  10  25  26  27  32  33  34  35  36  37  39  42  43  45  46  52  53
##  54  55  56  57  58  59  60  61  65  72  75  83  84 102 103 107 115 119
##  54  55  56  57  58  59  60  61  65  72  75  83  84 102 103 107 115 119
## 150
## 150
## attr(,"class")
## [1] "omit"
```

```
naprint(deleted)
```

```
## [1] "37 observations deleted due to missingness"
```

We have more incomplete rows if we add Solar.R as predictor.

```
fit2 <- lm(Ozone ~ Wind+Solar.R, data=airquality)
naprint(na.action(fit2))
```

```
## [1] "42 observations deleted due to missingness"
```

We can also drop observations with missing data ourselves instead of letting R do it for us. **Dropping data preemptively is generally a good idea, especially if you plan on using predict().**

```r
# complete cases on all variables in the data set
complete.v1 = filter( airquality, complete.cases(airquality) )

# drop observations with missing values, but ignoring a specific variable
complete.v2 = filter(airquality, complete.cases(select( airquality, -Wind) ) )

# drop observations with missing values on a specific variable
complete.v3 = filter(airquality, !is.na(Ozone))
```

Once you have subset your data, you just analyze what is left as normal. Easy as pie!

Mean imputation

Instead of dropping observations with missing values, we can plug in some kind of reasonable value for the missing value, e.g. the grand/global mean. While this can be statistically questionable, it does allow us to use the information provided by that unit's outcome and other covariates, without, we hope, unduly affecting the analysis of the missing covariate.

Generally, people will first plug in the mean value for anything missing, but then also make a dummy variable of whether that observation had a missing value there (or sometimes any missing value). You would then include both the original vector of covariates (with the means plugged in) along with the dummy variable in subsequent regressions and analyses.

Doing Mean Imputation manually

Manually, we can just replace missing values for a variable with the grand/global mean.

```r
# make a new copy of the data
  data.mean.impute = airquality

# select the observations with missing Ozone
  miss.ozone = is.na(data.mean.impute$Ozone)

# replace those NAs with mean(Ozone)
  data.mean.impute[miss.ozone,"Ozone"] = mean(airquality$Ozone, na.rm=TRUE)
```

In a multi-level context, it might make more sense to impute using the group mean rather than the grand mean. Here's a generic function to do it. Here we group by month:

```r
# a function that replaces missing values in a vector
# by the mean of the other values
  mean.impute = function(y) {
      y[is.na(y)] = mean(y, na.rm=TRUE)
      return(y)
  }

  data.mean.impute = airquality %>% group_by(Month) %>%
    mutate(Ozone = mean.impute(Ozone),
           Solar.R = mean.impute(Solar.R) )
```

We have mean imputed the Ozone column and the Solar.R column

Mean imputation with the Mice package

We can use the `mice` package to do mean imputation. The mice package is a package that can do some quite complex imputation, and so when you call `mice()` (which says "impute missing values please") you get back a rather complex object telling you what mice imputed, for whom, etc. This object, which is a `mids` object (see `help(mids)`), contains the multiply imputed dataset (or in our case, so far, singly imputed). The `mice` package then provides a lot of nice functions allowing you to get your imputed information out of this object.

We first demonstrate this for the 10 observations sampled above. Mice is generally going to be a two-step process: impute data, get completed dataset.

For step 1:

```
imp <- mice(airqualitysub, method="mean", m=1, maxit=1)
```

```
##
##  iter imp variable
##   1   1  Ozone  Solar.R
```

```
## Warning: Number of logged events: 1
```

```
imp
```

```
## Class: mids
## Number of multiple imputations:  1
## Imputation methods:
##   Ozone Solar.R    Wind    Temp   Month     Day
##  "mean"  "mean"      ""      ""      ""      ""
## PredictorMatrix:
##         Ozone Solar.R Wind Temp Month Day
## Ozone       0       1    1    1     0   1
## Solar.R     1       0    1    1     0   1
## Wind        1       1    0    1     0   1
## Temp        1       1    1    0     0   1
## Month       1       1    1    1     0   1
## Day         1       1    1    1     0   0
## Number of logged events:  1
##   it im dep     meth   out
## 1  0  0     constant Month
```

For step 2:

```
cmp = complete(imp)
cmp
```

```
##    Ozone Solar.R Wind Temp Month Day
## 1   41.0     190  7.4   67     5   1
## 2   36.0     118  8.0   72     5   2
## 3   12.0     149 12.6   74     5   3
## 4   18.0     313 11.5   62     5   4
## 5   23.1     173 14.3   56     5   5
## 6   28.0     173 14.9   66     5   6
## 7   23.0     299  8.6   65     5   7
## 8   19.0      99 13.8   59     5   8
## 9    8.0      19 20.1   61     5   9
## 10  23.1     194  8.6   69     5  10
```

We see there are no missing values in `cmp`. They were all imputed with the mean of the other non-missing values. This is **mean imputation**.

Now let's impute the full dataset.

```
imp <- mice(airquality, method="mean", m=1, maxit=1)
```

```
##
##  iter imp variable
##   1   1  Ozone  Solar.R
```

```
cmp = complete( imp )
```

We next make a dummy variable for each row of our data noting whether anything was imputed or not. We use the `ici` (Incomplete Case Indication) function to list all rows with any missing values.

```
head( ici(airquality) )
```

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

Note how we have a TRUE or FALSE for each row of our data.

We then store this as a covariate in our completed dataset:

```
cmp$imputed = ici(airquality)
head( cmp )
```

```
##   Ozone Solar.R Wind Temp Month Day imputed
## 1  41.0     190  7.4   67     5   1   FALSE
## 2  36.0     118  8.0   72     5   2   FALSE
## 3  12.0     149 12.6   74     5   3   FALSE
## 4  18.0     313 11.5   62     5   4   FALSE
## 5  42.1     186 14.3   56     5   5    TRUE
## 6  28.0     186 14.9   66     5   6    TRUE
```

How well did mean imputation work?

Mean imputation has problems. The imputed values will all be the same, and thus when we look at how much variation is in our variables after imputation, it will go down. Compare the sd of our completed dataset Ozone values to the SD of the Ozone values for our non-missing values.

```
sd( airquality$Ozone, na.rm=TRUE )
```

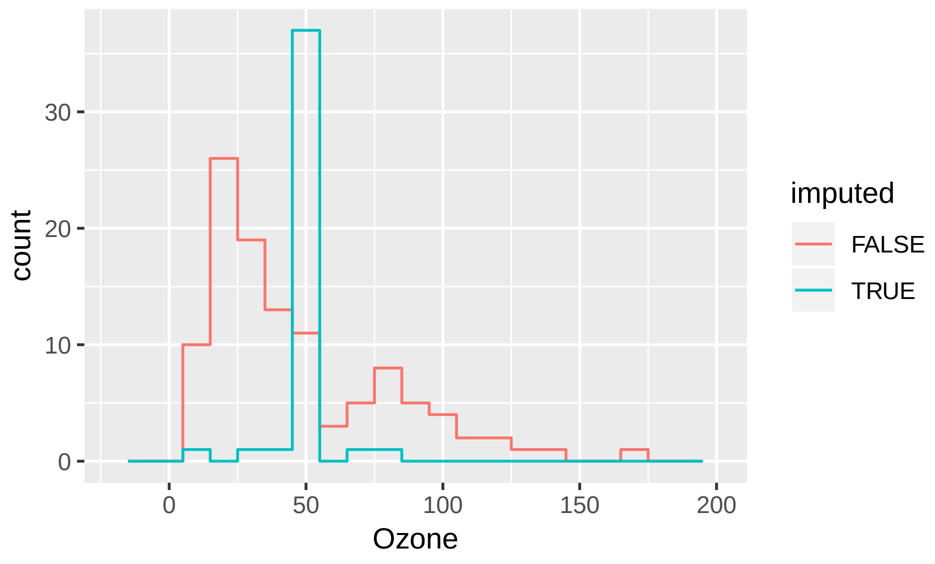
```
## [1] 33
```

```
sd( cmp$Ozone )
```

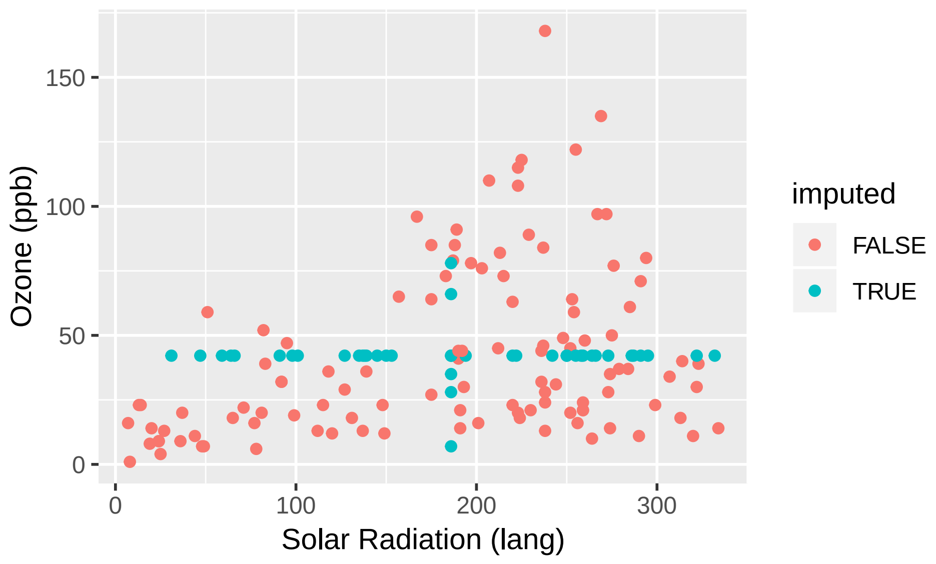
```
## [1] 28.7
```

Next, let's look at some plots of our completed data, coloring the points by whether they were imputed.

```
library(ggplot2)
ggplot( cmp, aes(x=Ozone, col=imputed) ) +
    stat_bin( geom="step", position="identity",
              breaks=seq(-20, 200, 10) )
```



```r
ggplot( cmp, aes(y=Ozone, x=Solar.R, col=imputed) ) +
    geom_point() +
    labs( y="Ozone (ppb)", x="Solar Radiation (lang)" )
```



What we see in the above plots is that our imputed observations do not look like the rest of our data because one (or both) of their values always is in the exact center. This creates the "+" shape. It also gives the big spike at the mean for the histogram.

Important Aside: Namespaces and function collisions

We now need to discuss a sad aspect of R. The short story is, different packages have functions with the same names and so if you have both packages loaded you will need to specify which package to use when calling such a fuction. You can do this by giving the "surname" of the function at the beginning of the function call (like, I believe, the Chinese). This comes up because for us the method `complete()` exists both in the tidyverse and in mice. In tidyverse, `complete()` fills in rows of missing combinations of values. In mice, `complete()` gives us a completed dataset after we have made an imputation call.

It turns out that since we loaded tidyverse first and mice second, the mice's `complete()` method is the default. But if we loaded the packages in the other order, we would get strange errors. To be clear, we thus tell R to use `mice` by writing:

```
cmp = mice::complete( imp )
```

In general, you can detect such "namespace collisions" by noticing weird error messages all of a sudden when you don't expect them. You can then type, for example, `help( complete )` and it will list all the different `complete`s around.

```
help( complete )
```

Also when you load a package it will write down what functions are getting mixed up for you. If you were looking at your R code you would get something like this:

```
tidyr::complete() masks mice::complete()
```

Regression imputation

Regression imputation is half way between mean imputation and multiple imputation. In regression imputation we predict what values we expect for anything missing based on the other values of the observation. For example, if we know that urban/rural is correlated with race, we might impute a different value for race if we know an observation came from an urban environment vs. rural. We do this with regression: we fit a model predicting each variable using the others and then use that regression model to predict any missing values.

We can do this manually, but then it gets very hard when multiple variables are missing for a given observation. The mice package is more clever: it does variables one at a time, and the cycles around so everything can get imputed.

Manually

Here is how to use other variables to predict missing values.

```
ic( airqualitysub )
```

```
##    Ozone Solar.R Wind Temp Month Day
## 5     NA      NA 14.3   56     5   5
## 6     28      NA 14.9   66     5   6
## 10    NA     194  8.6   69     5  10
```

```
fit <- lm(Ozone ~ Solar.R, data=airqualitysub)

# predict for missing ozone
need.pred = subset( airqualitysub, is.na( Ozone ) )
need.pred
```

```
##    Ozone Solar.R Wind Temp Month Day
## 5     NA      NA 14.3   56     5   5
## 10    NA     194  8.6   69     5  10
```

```
pred <- predict(fit, newdata=need.pred)
pred
```

```
##    5   10
##   NA 23.1
```

But now we have to merge back in, and we didn't solve for case 5 because we are missing the variable we would use to predict the other missing variable. Ick. This is where missing data gets *really* hard (when we have multiple missing values on multiple variables). So let's quit now and turn to a package that will handle all of this for us.

Mice

To do regression imputation using mice, we simply call the `mice()` method:

```
imp <- mice(airquality[,1:2], method="norm.predict", m=1, maxit=3,seed=1)
```

```
##
##  iter imp variable
##   1   1  Ozone  Solar.R
##   2   1  Ozone  Solar.R
##   3   1  Ozone  Solar.R
```

We have everything! How did it do it? By *chaining equations*. First we start with mean imputation. Then we use our fit model to predict for one covariate, and then uses those predicted scores to predict for the next covariate, and so forth. We cycle back and then everything is jointly predicting everything else.

The `complete()` method gives us a complete dataset with everything imputed. Like so:

```
cdat = mice::complete( imp )
head( cdat )
```

```
##   Ozone Solar.R
## 1  41.0     190
## 2  36.0     118
## 3  12.0     149
## 4  18.0     313
## 5  42.7     186
## 6  28.0     169
```

```
nrow( cdat )
```

```
## [1] 153
```

```
nrow( airquality )
```

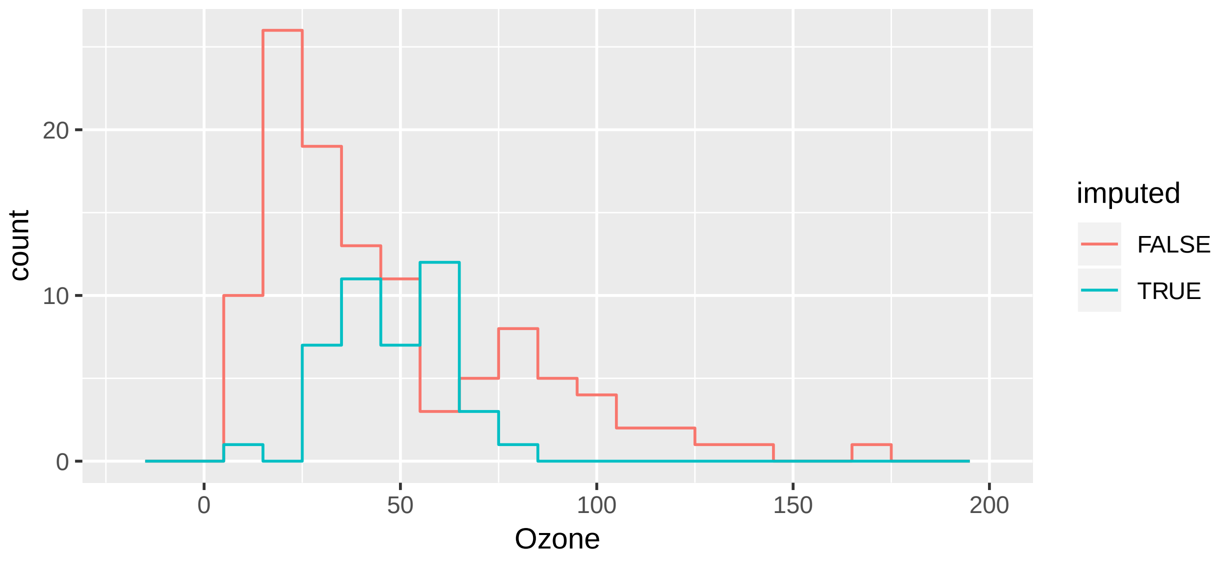
```
## [1] 153
```

Next we make a variable of which cases have imputed values and not (any row with missing data must have been partially imputed.)

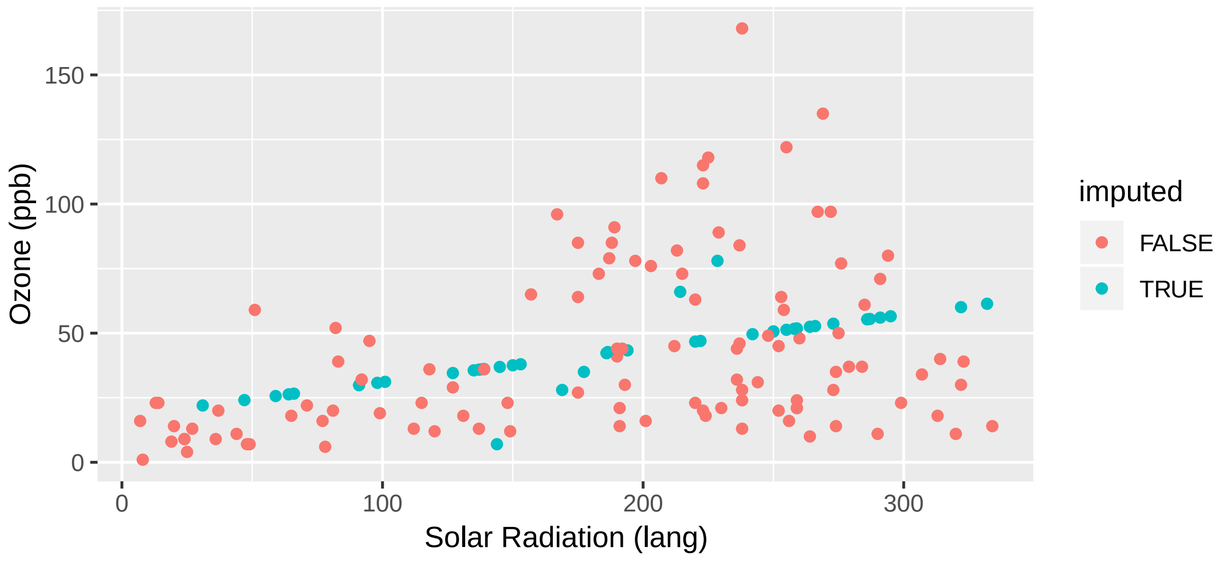
```
cdat$imputed = ici( airquality )
```

And see our results! Compare to mean imputation, above.

```
ggplot( cdat, aes(x=Ozone, col=imputed) ) +
    stat_bin( geom="step", position="identity",
              breaks=seq(-20, 200, 10) )
```



```r
ggplot( cdat, aes(y=Ozone, x=Solar.R, col=imputed) ) +
    geom_point() +
    labs( y="Ozone (ppb)", x="Solar Radiation (lang)" )
```



This is better than mean imputation. See how we impute different Ozone for different Solar Radiation values, taking advantage of the information of knowing that they are correlated? But it still is obvious what is mean imputed and what is not. Also, the variance of our imputed values still does not contain the residual variation around the predicted values that we would get in real data. We can do one more enhancement to fix this.

Stochastic regression imputation

We extend regression imputation by randomly drawing observations that *look like* real ones. See in the two imputations below we get slightly different values for our imputed data.

Here we do it on our mini-dataset and look at the imputed values for our observations with missing values only:

```r
  imp <- mice(airqualitysub[,1:2],method="norm.nob",m=1,maxit=1,seed=1)
```

```
##
##  iter imp variable
##   1   1  Ozone  Solar.R
```

```
imp$imp
```

```
## $Ozone
##       1
## 5  15.6
## 10 44.6
##
## $Solar.R
##       1
## 5 194.0
## 6  83.7
```

```
imp <- mice(airqualitysub[,1:2],method="norm.nob",m=1,maxit=1,seed=4)
```

```
##
##  iter imp variable
##   1   1  Ozone  Solar.R
```

```
imp$imp
```

```
## $Ozone
##       1
## 5  34.4
## 10 31.6
##
## $Solar.R
##     1
## 5 381
## 6 260
```

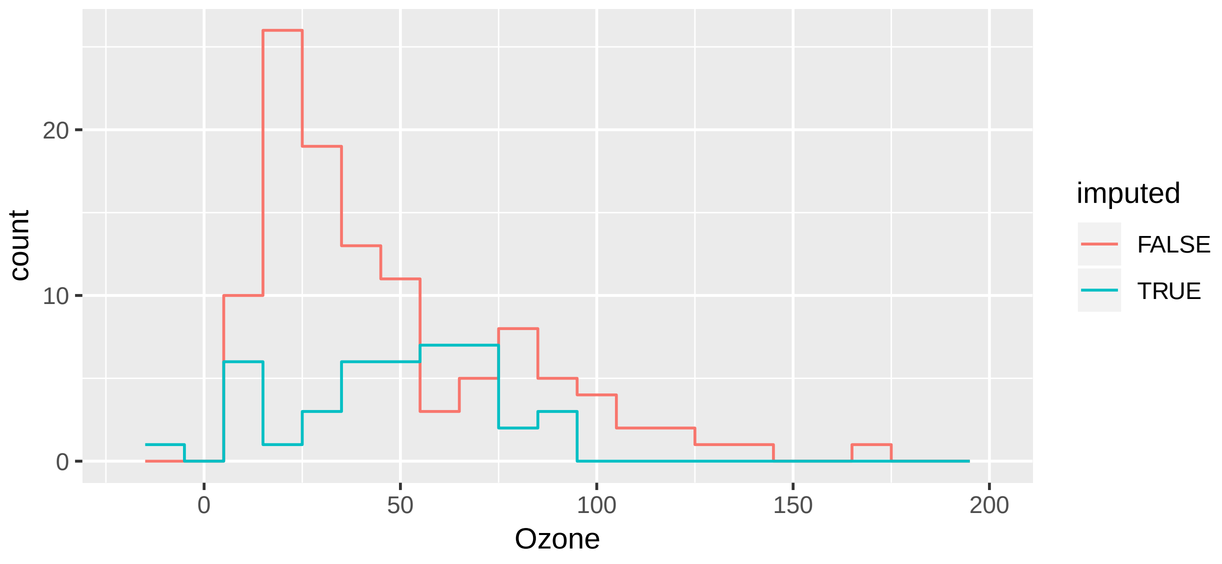
Now let's do it on the full data and look at the imputed values and compare to our plots above.

```
imp <- mice(airquality[,1:2],method="norm.nob",m=1,maxit=1,seed=1)
```

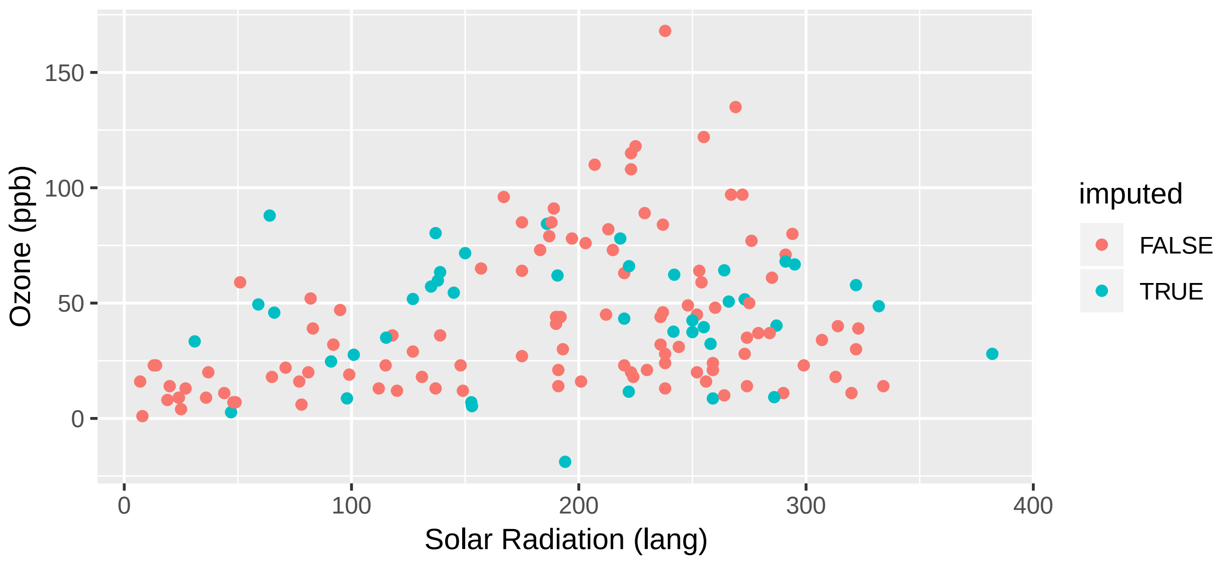
```
##
##  iter imp variable
##   1   1  Ozone  Solar.R
```

```
cdat = mice::complete( imp )
cdat$imputed = ici( airquality )

ggplot( cdat, aes(x=Ozone, col=imputed) ) +
  stat_bin( geom="step", position="identity",
            breaks=seq(-20, 200, 10) )
```



```
ggplot( cdat, aes(y=Ozone, x=Solar.R, col=imputed) ) +
  geom_point() +
  labs( y="Ozone (ppb)", x="Solar Radiation (lang)" )
```



Better, but not perfect. What is better? What is still not perfect?

Multiple imputation

If missing data is a significant issue in your dataset, then mean or regression imputation can be a bit too hacky and approximate. In these contexts, multiple imputation is the way to go.

We do this as follows:

```
imp <- mice(airqualitysub, seed=2, print=FALSE)
```

```
## Warning: Number of logged events: 1
```

```
imp
```

```
## Class: mids
## Number of multiple imputations:  5
## Imputation methods:
##   Ozone Solar.R    Wind    Temp   Month     Day
##   "pmm"   "pmm"      ""      ""      ""      ""
## PredictorMatrix:
##         Ozone Solar.R Wind Temp Month Day
## Ozone       0       1    1    1     0   1
## Solar.R     1       0    1    1     0   1
## Wind        1       1    0    1     0   1
## Temp        1       1    1    0     0   1
## Month       1       1    1    1     0   1
## Day         1       1    1    1     0   0
## Number of logged events:  1
##   it im dep     meth   out
## 1  0  0     constant Month
```

```
imp$imp
```

```
## $Ozone
##     1  2  3  4  5
## 5   8 18 41 36 23
## 10 23 28 18 23 19
##
## $Solar.R
##     1   2   3   4   5
## 5 194 194 149 149 118
## 6  99 118 118  19 118
##
## $Wind
## [1] 1 2 3 4 5
## <0 rows> (or 0-length row.names)
##
## $Temp
## [1] 1 2 3 4 5
## <0 rows> (or 0-length row.names)
##
## $Month
## [1] 1 2 3 4 5
## <0 rows> (or 0-length row.names)
##
## $Day
## [1] 1 2 3 4 5
## <0 rows> (or 0-length row.names)
```

See multiple columns of imputed data? (We have 5 here.)

First aside: All variables you'll be using for your model should be included in the imputation model. Notice we included the full dataset in `mice`, not just the variables with missing values. This way we can account for associations between all the outcome and the predictors in the model we'll be fitting. Your imputation model can be more complicated than your model of interest. That is, you can include additional variables that predict missing values but will not be part of your final model of interest.

Second aside: All variables in your imputation model should be in the correct functional form! Quadratic, higher order polynomials and interaction temrs are just another variable that we need to impute. Although it may seem logical to impute your variables first and then calculate the interaction or non-linear term, this can lead to bias.

Third aside: The ordering of the variables in the dataset you are feeding into `mice` can make a difference in results and model convergence. Generally, you want to order your variables from least to most missing. Here, we reorder the variables from least to most missing, and obtain different results.

```
test = airqualitysub[,c(6,5,4,3,2,1)]
head(test)
```

```
##   Day Month Temp Wind Solar.R Ozone
## 1   1     5   67  7.4     190    41
## 2   2     5   72  8.0     118    36
## 3   3     5   74 12.6     149    12
## 4   4     5   62 11.5     313    18
## 5   5     5   56 14.3      NA    NA
## 6   6     5   66 14.9      NA    28
```

```
test.imp <- mice(test, seed=2, print=FALSE)
```

```
## Warning: Number of logged events: 1
```

```
test.imp$imp
```

```
## $Day
## [1] 1 2 3 4 5
## <0 rows> (or 0-length row.names)
##
## $Month
## [1] 1 2 3 4 5
## <0 rows> (or 0-length row.names)
##
## $Temp
## [1] 1 2 3 4 5
## <0 rows> (or 0-length row.names)
##
## $Wind
## [1] 1 2 3 4 5
## <0 rows> (or 0-length row.names)
##
## $Solar.R
##     1   2   3   4  5
## 5 194 313 190 313 99
## 6  99 149 118  99 19
##
## $Ozone
##     1  2  3  4  5
## 5  19 19 18 28 36
## 10 23 18 18 19 18
```

How to get each complete dataset?

```
# first complete dataset
mice::complete(imp, 1)
```

```
##    Ozone Solar.R Wind Temp Month Day
## 1     41     190  7.4   67     5   1
## 2     36     118  8.0   72     5   2
## 3     12     149 12.6   74     5   3
## 4      18      313 11.5   62      5   4
## 5       8      194 14.3   56      5   5
## 6      28       99 14.9   66      5   6
## 7      23      299  8.6   65      5   7
## 8      19       99 13.8   59      5   8
## 9       8       19 20.1   61      5   9
## 10     23      194  8.6   69      5  10
```

```
# and our second complete dataset
  mice::complete(imp, 2)
```

```
##    Ozone Solar.R Wind Temp Month Day
## 1     41     190  7.4   67     5   1
## 2     36     118  8.0   72     5   2
## 3     12     149 12.6   74     5   3
## 4     18     313 11.5   62     5   4
## 5     18     194 14.3   56     5   5
## 6     28     118 14.9   66     5   6
## 7     23     299  8.6   65     5   7
## 8     19      99 13.8   59     5   8
## 9      8      19 20.1   61     5   9
## 10    28     194  8.6   69     5  10
```

See how they are different? They were randomly imputed. We basically used the stochastic regression thing, above, multiple times.

```
  mice::complete(imp, 1)[ ici(airqualitysub), ]
```

```
##    Ozone Solar.R Wind Temp Month Day
## 5      8     194 14.3   56     5   5
## 6     28      99 14.9   66     5   6
## 10    23     194  8.6   69     5  10
```

```
  mice::complete(imp, 2)[ ici(airqualitysub), ]
```

```
##    Ozone Solar.R Wind Temp Month Day
## 5     18     194 14.3   56     5   5
## 6     28     118 14.9   66     5   6
## 10    28     194  8.6   69     5  10
```

On full data:

```
  imp <- mice(airquality, seed=1, print=FALSE)
```

Now we estimate for each imputed dataset using the `with()` method that, in `mice`, will do the regression for each completed dataset. See `help with.mids`.

```
  fit <- with(imp, lm(Ozone ~ Wind + Temp + Solar.R))
  fit
```

```
## call :
## with.mids(data = imp, expr = lm(Ozone ~ Wind + Temp + Solar.R))
##
## call1 :
## mice(data = airquality, printFlag = FALSE, seed = 1)
##
## nmis :
##   Ozone Solar.R    Wind    Temp   Month     Day
##      37       7       0       0       0       0
```

```
##
## analyses :
## [[1]]
##
## Call:
## lm(formula = Ozone ~ Wind + Temp + Solar.R)
##
## Coefficients:
## (Intercept)         Wind         Temp      Solar.R
##    -73.8134      -2.7690       1.6977       0.0597
##
##
## [[2]]
##
## Call:
## lm(formula = Ozone ~ Wind + Temp + Solar.R)
##
## Coefficients:
## (Intercept)         Wind         Temp      Solar.R
##    -64.5101      -3.0377       1.5995       0.0669
##
##
## [[3]]
##
## Call:
## lm(formula = Ozone ~ Wind + Temp + Solar.R)
##
## Coefficients:
## (Intercept)         Wind         Temp      Solar.R
##    -65.5981      -3.0582       1.6556       0.0541
##
##
## [[4]]
##
## Call:
## lm(formula = Ozone ~ Wind + Temp + Solar.R)
##
## Coefficients:
## (Intercept)         Wind         Temp      Solar.R
##    -60.9246      -2.9043       1.5366       0.0602
##
##
## [[5]]
##
## Call:
## lm(formula = Ozone ~ Wind + Temp + Solar.R)
##
## Coefficients:
## (Intercept)         Wind         Temp      Solar.R
##     -61.987       -3.082        1.589        0.057
```

This can take *any* function call that takes a formula. So `glm`, `lm`, whatever… We can then pool the estimates using the standard theory of combining multiply imputed datasets. The basic idea is to combine the variation/uncertainty of the multiple sets with the average uncertainty we would have for each set if it was truely complete and not imputed.

```
tab <- round(summary(pool(fit)),3)
colnames( tab )
```

```
## [1] "estimate"  "std.error" "statistic" "df"        "p.value"
```

```
tab[,c(1:3,5)]
```

```
##             estimate std.error statistic p.value
## (Intercept)   -65.37    20.210     -3.23   0.002
## Wind           -2.97     0.577     -5.15   0.000
## Temp            1.62     0.226      7.15   0.000
## Solar.R         0.06     0.021      2.85   0.005
```

Aside: You will notice that once we fit our model on the imputed data, `with()` returned an object of class `mira`. `Mira` objects can be pooled to get the pooled estimates, whereas objects of class `glm`, `lm`, `lmer`, etc. cannot be pooled. You will also notice that you cannot use `predict` with a `mira` object. To use `predict`, you can stack the imputed datasets and fit your model on this complete dataset. Parameter estimates generated by `pool` are the average of the parameter estimates from the model fit on each imputed dataset separately. So your coeficients are fine. However, your SEs will be underestimated. How underestimated your SEs will be depends, to an extent, on how much data is missing and whether it is missing at random.

Our old, sad method:

```
fit <- lm(Ozone~Wind+Temp+Solar.R,data=airquality,na.action=na.omit)
summary( fit )
```

```
##
## Call:
## lm(formula = Ozone ~ Wind + Temp + Solar.R, data = airquality,
##     na.action = na.omit)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -40.48 -14.22  -3.55  10.10  95.62
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -64.3421    23.0547   -2.79   0.0062 **
## Wind         -3.3336     0.6544   -5.09  1.5e-06 ***
## Temp          1.6521     0.2535    6.52  2.4e-09 ***
## Solar.R       0.0598     0.0232    2.58   0.0112 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 21.2 on 107 degrees of freedom
##   (42 observations deleted due to missingness)
## Multiple R-squared:  0.606,  Adjusted R-squared:  0.595
## F-statistic: 54.8 on 3 and 107 DF,  p-value: <2e-16
```

```
round(coef(summary(fit)),3)
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   -64.34     23.055   -2.79    0.006
## Wind           -3.33      0.654   -5.09    0.000
## Temp            1.65      0.254    6.52    0.000
## Solar.R         0.06      0.023    2.58    0.011
```

In this case, the missing data estimates are basically the same as the complete case analysis, it appears. We ony had 5% missing data though.

Extensions

Non-continuous variables

Everything shown above can easily be extended to non-continuous variables. The easiest way to do this is using the `mice` package. It allows you to specify the type of variable you are imputing, e.g. dichotmous or categorical. `Mice` will autoamtically detect and handle non-continuous variables. You can also specify these vaiables yourself. Here is an example using **`nhanes`** data (another built in R dataset).

```
# load data
  data(nhanes2)
  head(nhanes2)
```

```
##     age  bmi  hyp chl
## 1 20-39   NA <NA>  NA
## 2 40-59 22.7   no 187
## 3 20-39   NA   no 187
## 4 60-99   NA <NA>  NA
## 5 20-39 20.4   no 113
## 6 60-99   NA <NA> 184
```

```
# create some missing values for an ordered categorical variable
  nhanes2$age[1:5] = NA
  head(nhanes2)
```

```
##     age  bmi  hyp chl
## 1  <NA>   NA <NA>  NA
## 2  <NA> 22.7   no 187
## 3  <NA>   NA   no 187
## 4  <NA>   NA <NA>  NA
## 5  <NA> 20.4   no 113
## 6 60-99   NA <NA> 184
```

```
# impute 5 datasets
  imp.cat <- mice(nhanes2, m = 5, print=FALSE)
  full.cat = mice::complete(imp.cat)           # print the first imputed data set
  head(full.cat)
```

```
##     age  bmi hyp chl
## 1 40-59 27.2 yes 186
## 2 40-59 22.7  no 187
## 3 20-39 27.5  no 187
## 4 60-99 22.7  no 131
## 5 20-39 20.4  no 113
## 6 60-99 22.7 yes 184
```

We can check what imputation method `mice` used for each variable:

```
  imp.cat$method
```

```
##       age       bmi       hyp       chl
## "polyreg"     "pmm"  "logreg"     "pmm"
```

We can see that `mice` used the `polyreg` imputation method for the variable *age*, which means it treated it as an unordered categoical variable. But this is an ordered variable: higher values categories signified older age. We can manually force `mice` to treat *age* as an ordered categorical variable. We will keep the imputation methods for the remaining variables the same.

```r
imp.cat2 <- mice(nhanes2, meth=c("polr","pmm","logreg","pmm"), m=5, print=FALSE)
head(mice::complete(imp.cat2, 1))
```

```
##     age  bmi hyp chl
## 1 40-59 22.7 yes 187
## 2 40-59 22.7  no 187
## 3 60-99 22.7  no 187
## 4 40-59 27.5 yes 199
## 5 20-39 20.4  no 113
## 6 60-99 25.5  no 184
```

```r
imp.cat2$method
```

```
##      age      bmi      hyp      chl
##   "polr"    "pmm" "logreg"    "pmm"
```

Multi-level data

Multilevel data gets more tricky: should we impute taking into account cluster? How do we do that?

For an initial pass, I would recommend simply doing regression imputation *ignoring* cluster/grouping, and then adding in that dummy variable of whether a value is imputed.

Further reading

Some further reading on handling missing data. But this is really a course into itself.

- Gelman & Hill Chapter 25 has a more detailed discussion of missing data imputation.
- White IR, Royston P, Wood AM. Multiple imputation using chained equations: issues and guidance for practice. Statistics in Medicine 2011;30: 377-399.
- Graham, JW, Olchowski, AE, Gilreath, TD, 2007. How Many Imputations are Really Needed? Some Practical Clarifications of Multiple Imputation Theory 206–213. https://doi.org/10.1007/s11121-007-0070-9
- van Buurin S, Groothuis-Oudshoorn K, MICE: Multivariate Imputation by Chained Equations. Journal of Statistical Software. 2011;45(3):1-68.
- Grund S, Lüdtke O, Robitzsch A. Multiple Imputation of Missing Data for Multilevel Models: Simulations and Recommendations. DOI: 10.1177/1094428117703686

Appendix: More about the mice package

The mice package gives back a very complex object that has a lot of information about how values were imputed, which values were imputed, and so forth. In the following we unpack the `imp` variable from above a bit more.

Looking at the imputation object

In the following code, we look at the object we get back from `mice()`. It has lots of parts that we can peek into.

First, the `imp` list inside of `imp` stores all of our newly imputed data. It is itself a list of each variable with their imputed values:

```
imp$imp
```

```
## $Ozone
##        1   2   3   4   5
## 5     19  19  18  13  18
## 10    12  41  16  36  44
## 25    19   8  18  18  14
## 26    14  18   9  37   6
## 27    18  13  16  11  37
## 32    46  35  20  63  29
## 33    16  44  31   7  36
## 34    37  37  28   1   1
## 35    40  23  44  23  59
## 36    37  89  35  39  96
## 37    16  23  44  36  16
## 39    82  82  66  50 135
## 42    78  77  96 115  91
## 43    97  91 168  50  73
## 45    59  45  39  31  59
## 46    52  35  16  29  28
## 52    28   7  89  16  40
## 53    63  49  80  48  23
## 54    28  35  39  40  63
## 55    40  71  78  71  16
## 56    31  36  44  16  65
## 57    28  32  23  29  20
## 58    12  12  23  44  27
## 59    44  18  31  13  29
## 60    14   9  24  24  22
## 61    37  89 108  89  64
## 65    59  65  16  39  32
## 72    44  28  28  52  40
## 75    48  59  52  47  37
## 83    40  40  49  20  16
## 84    47  16  35  52  20
## 102   82  66  96  85  78
## 103   52  59  36  16  40
## 107   21  21  34  44  16
## 115   21  41  36  14  34
## 119  135 135  78  61  78
## 150   32   7  21  14  12
##
## $Solar.R
##       1   2   3   4   5
## 5   259 186 137 131 145
## 6   294 264 118 256 287
## 11  264 139  82  71 259
## 27  238  37 115 223 224
## 96  276 332 323 250 314
## 97 190 190 236 149 190
## 98  92  92 284 273 322
##
## $Wind
## [1] 1 2 3 4 5
## <0 rows> (or 0-length row.names)
##
## $Temp
## [1] 1 2 3 4 5
## <0 rows> (or 0-length row.names)
##
## $Month
## [1] 1 2 3 4 5
## <0 rows> (or 0-length row.names)
##
## $Day
## [1] 1 2 3 4 5
## <0 rows> (or 0-length row.names)
```

```r
  str( imp$imp )
```

```
## List of 6
##  $ Ozone  :'data.frame': 37 obs. of  5 variables:
##   ..$ 1: int [1:37] 19 12 19 14 18 46 16 37 40 37 ...
##   ..$ 2: int [1:37] 19 41 8 18 13 35 44 37 23 89 ...
##   ..$ 3: int [1:37] 18 16 18 9 16 20 31 28 44 35 ...
##   ..$ 4: int [1:37] 13 36 18 37 11 63 7 1 23 39 ...
##   ..$ 5: int [1:37] 18 44 14 6 37 29 36 1 59 96 ...
##  $ Solar.R:'data.frame': 7 obs. of  5 variables:
##   ..$ 1: int [1:7] 259 294 264 238 276 190 92
##   ..$ 2: int [1:7] 186 264 139 37 332 190 92
##   ..$ 3: int [1:7] 137 118 82 115 323 236 284
##   ..$ 4: int [1:7] 131 256 71 223 250 149 273
##   ..$ 5: int [1:7] 145 287 259 224 314 190 322
##  $ Wind   :'data.frame': 0 obs. of  5 variables:
##   ..$ 1: logi(0)
##   ..$ 2: logi(0)
##   ..$ 3: logi(0)
##   ..$ 4: logi(0)
##   ..$ 5: logi(0)
##  $ Temp   :'data.frame': 0 obs. of  5 variables:
##   ..$ 1: logi(0)
##   ..$ 2: logi(0)
##   ..$ 3: logi(0)
##   ..$ 4: logi(0)
##   ..$ 5: logi(0)
##  $ Month  :'data.frame': 0 obs. of  5 variables:
##   ..$ 1: logi(0)
##   ..$ 2: logi(0)
##   ..$ 3: logi(0)
##   ..$ 4: logi(0)
##   ..$ 5: logi(0)
##  $ Day    :'data.frame': 0 obs. of  5 variables:
##   ..$ 1: logi(0)
##   ..$ 2: logi(0)
##   ..$ 3: logi(0)
##   ..$ 4: logi(0)
##   ..$ 5: logi(0)
```

```
str( imp$imp$Ozone )
```

```
## 'data.frame':    37 obs. of  5 variables:
##  $ 1: int  19 12 19 14 18 46 16 37 40 37 ...
##  $ 2: int  19 41 8 18 13 35 44 37 23 89 ...
##  $ 3: int  18 16 18 9 16 20 31 28 44 35 ...
##  $ 4: int  13 36 18 37 11 63 7 1 23 39 ...
##  $ 5: int  18 44 14 6 37 29 36 1 59 96 ...
```

We see that Ozone and Solar.R have imputed values, and the other variables do not.

Next, we see two missing observations in our original data and then see the two imputed values for these two missing observations.

```
airqualitysub$Ozone
```

```
##  [1] 41 36 12 18 NA 28 23 19  8 NA
```

```
imp$imp$Ozone[,1]
```

```
##  [1]  19  12  19  14  18  46  16  37  40  37  16  82  78  97  59  52  28
## [18]  63  28  40  31  28  12  44  14  37  59  44  48  40  47  82  52  21
## [35]  21 135  32
```

We can make (the hard way) a vector of Ozone by plugging our missing values into the original data. But the `complete()` method, above, is preferred.

```
oz = airqualitysub$Ozone
oz[ is.na( oz ) ] = imp$imp$Ozone[,1]
```

```
## Warning in oz[is.na(oz)] = imp$imp$Ozone[, 1]: number of items to replace
## is not a multiple of replacement length
```

```
oz
```

```
##  [1] 41 36 12 18 19 28 23 19  8 12
```

What else is there in imp?

```
names(imp)
```

```
##  [1] "data"            "imp"             "m"
##  [4] "where"           "blocks"          "call"
##  [7] "nmis"            "method"          "predictorMatrix"
## [10] "visitSequence"   "formulas"        "post"
## [13] "blots"           "seed"            "iteration"
## [16] "lastSeedValue"   "chainMean"       "chainVar"
## [19] "loggedEvents"    "version"         "date"
```

What was our imputation method?

```
imp$method
```

```
##   Ozone Solar.R    Wind    Temp   Month     Day
##   "pmm"   "pmm"      ""      ""      ""      ""
```

Mean imputation for each variable with missing values. Later this will say other thing.

What was used to impute what?

```
imp$predictorMatrix
```

```
##         Ozone Solar.R Wind Temp Month Day
## Ozone       0       1    1    1     1   1
## Solar.R     1       0    1    1     1   1
## Wind        1       1    0    1     1   1
## Temp        1       1    1    0     1   1
## Month       1       1    1    1     0   1
## Day         1       1    1    1     1   0
```

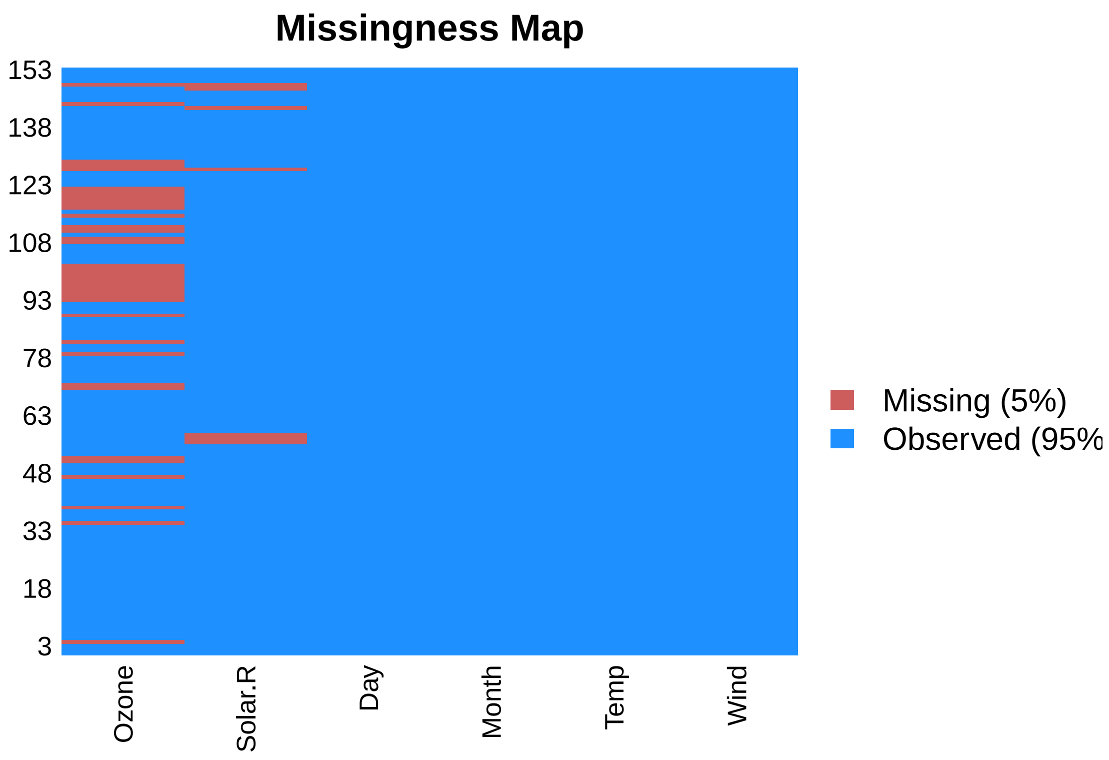
Appendix: The amelia package

Amelia is another multiple imputation and missing data package. We do not prefer it, but have some demonstration code in the following, for reference.

```
library(Amelia)
```

For missingness we can make the following:

```
missmap(airquality)
```



Each row of the plot is a row of the data, and missing values are shown in brown. But ugly! And hard to see any trends in the missingness.

You can use the `Amelia` package to do mean imputation.

```
library(dplyr)

# exclude variables that do not vary
a.airquality = airquality %>% dplyr::select(-Month)
```

```r
# impute data
  a.imp <- amelia(a.airquality, m=5)
```

```
## -- Imputation 1 --
##
##   1  2  3  4  5  6
##
## -- Imputation 2 --
##
##   1  2  3  4  5  6  7
##
## -- Imputation 3 --
##
##   1  2  3  4  5
##
## -- Imputation 4 --
##
##   1  2  3  4  5  6
##
## -- Imputation 5 --
##
##   1  2  3  4  5  6
```

```r
  a.imp
```

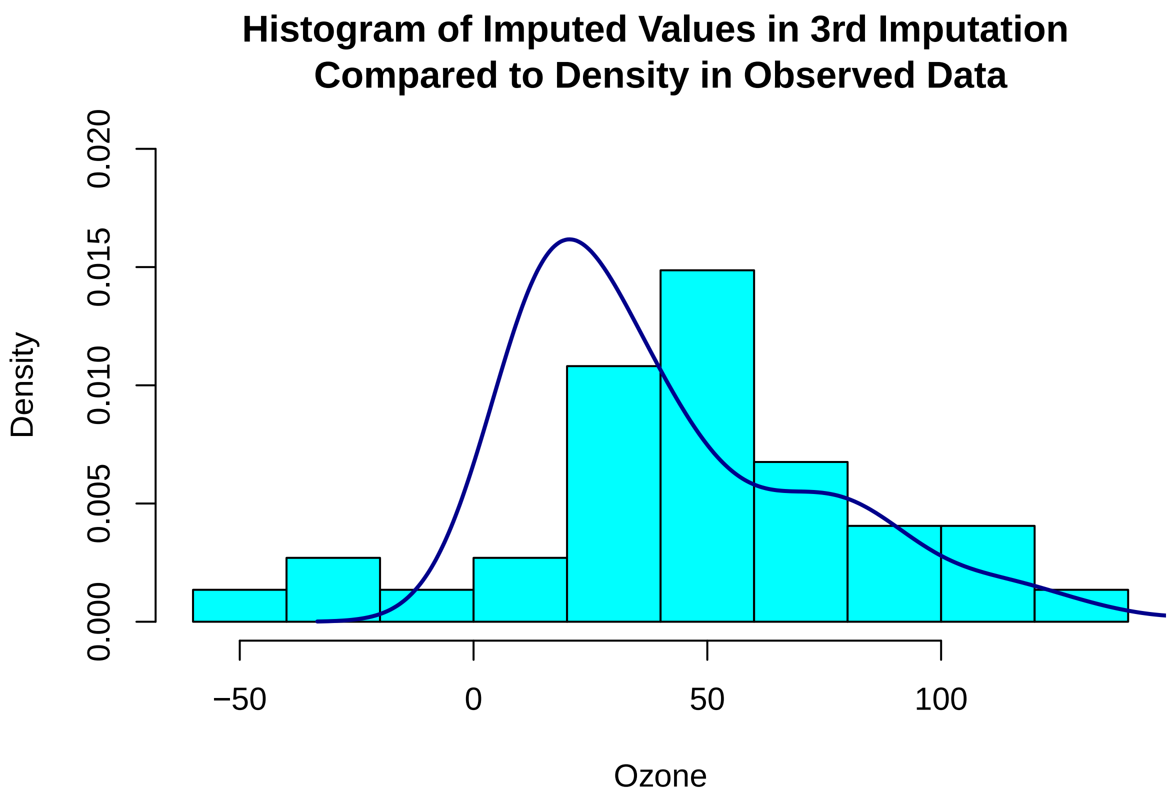
```
##
## Amelia output with 5 imputed datasets.
## Return code:  1
## Message:  Normal EM convergence.
##
## Chain Lengths:
## --------------
## Imputation 1:  6
## Imputation 2:  7
## Imputation 3:  5
## Imputation 4:  6
## Imputation 5:  6
```

We can plot our imputed values against our observed values to check that they make sense. We will do this for just one of five datasets we just imputed using `Amelia`.

```r
# put imputed values from the third dataset in an object
  one_imp <- a.imp$imputations[[3]]$Ozone

# make object with observed values
# from observations without missing Ozone values
  obs_data <- a.airquality$Ozone

# make a plot overlaying observed and imputed values
  hist(one_imp[is.na(obs_data)], prob=TRUE, xlab="Ozone",
       main="Histogram of Imputed Values in 3rd Imputation \nCompared to Density in Observed Data",
       col="cyan", ylim=c(0,0.02))
  lines(density(obs_data[!is.na(obs_data)]), col="darkblue", lwd=2)
```



You can also do multiple imputation in `Amelia`. However, `Amelia` does not have an easy way to combine the estimates from the imputed datasets (no analogue of `with()` in `mice`). You can write a function that fits your model of interest in each imputed dataset and then use a package like `mitools` to pool the estimates and variances.

Much easier to use `mice`!

Aside: A more important limitation of `Amelia` is that the algorithm it uses to impute missing values assumes multivariate normality, which is often questionable, especially when you have binary variables.