How and Why to use Multiple Imputation

Thomas Lumley

2019-11-04

Contents

Chapter 1

Executive Summary

Multiple imputation is a valuable technique for reducing bias when training predictive models in health data with missing values, and for increasing applicability when deploying the models in data with missing values.

Multiple imputation works by creating a representative sample of plausible values that the missing data could have had, and then averaging the analysis over that sample. Using randomly-sampled representative values reduces bias, and using a sample of multiple plausible values for each missing value allows for valid estimates of uncertainty.

In large datasets, the classical methods of multiple imputation are computationally challenging. One goal of this project was to investigate multiple imputation using modern machine learning methods: random forests, and deep neural networks. These approaches are computationally feasible in large datasets but the representative sampling of the imputations is still not as good as the classical approaches.

We recommend

- using multiple imputation where there is substantial missing data
- using `mice`, multiple imputation by chained equations, if computationally feasible
- considering random-forest if `mice` is not computationally feasible and missing-data bias is a major concern
- updating this document in a year or two, as the machine learning approaches are an active research area.

Chapter 2

Why impute?

Missing data is ubiquitous in statistics. Even theoretically-complete administrative data such as the NMDS for hospital admissions will have some fields not filled in. Additional missing data will be created when impossible values are removed in data cleaning.

Traditionally, records with missing observations have simply been omitted from data analysis and model fitting. This 'complete-case' approach to analysis is automated in all major statistical software; it was the only straightforward approach available when the packages were created.

Complete-case analysis, however, can be substantially biased; people with missing data are not a representative sample of the population. For example, in the 2018 New Zealand Census(External Data Quality Panel, 2019, table 4.8), 16% of those who responded to the question on 'Māori descent' indicated they did have Māori descent, but Stats New Zealand estimate that 48% of those who did not respond had Māori descent. In healthcare settings, people who do not respond may be at systematically lower risk if measurements are made based on clinical need, or at systematically higher risk if missing data reflects poorer access to care or higher clinician workload.

There are two general and statistically principled ways of handling missing data. The first is to use only the complete records, but to reweight them so that they represent the whole target population; this is analogous to direct standardisation for age in epidemiology. Reweighting is straightforward, but inefficient; there may be nearly-complete data on many people, and it is being ignored. The second is imputation.

First, consider single imputation, which replaces each missing value by a single imputed value. There are two approaches to single imputation: replacing a missing value by the *most likely true value*, and replacing by a *randomly chosen plausible value*, in both cases taking into account what else is known about the individual. The first approach corrects some of the bias, but has a tendency to cause stereotyping. Suppose, for example, that we are imputing current smoking status. The most likely value, for almost any group of people in New Zealand, is 'non-smoker', because the smoking prevalence is well below 50%. However, imputing 'non-smoker' for *all* the missing values will lead to a downwards bias in smoking prevalence. Similarly, imputing ethnicity by the most likely value will over-represent NZ Europeans, and imputing diabetes status by the most likely value will under-represent diabetes. Imputing the most likely value is still a helpful strategy with small amounts of missing data, or where the values can be accurately predicted.

The second single-imputation approach is widely used for census data; missing values are replaced by the observed value from some other individual, who is randomly chosen from a set of sufficiently close matches. Stochastic single imputation will preserve the distribution of each variable. For example, a missing smoking status will be imputed as smoking or non-smoking in proportion to the actual prevalence in matching individuals. However, stochastic imputation tends to bias associations between variables and between individuals in a household.

All single-imputation methods suffer from the problem of 'making up data'. The data set ends up complete; there is no simple way to know how much of the statistical information in the data is real and how much was created from nothing. As long as only a few percent of data are imputed this isn't a problem, but if 10-20% is imputed the standard errors are likely to be too small and p-values are likely to overstate the evidence.

Multiple imputation(Rubin, 1987) does allow for a valid assessment of evidence after imputation. Rather than being imputed only once, each missing value is replaced by a *representative sample of plausible values* to create a collection of plausible complete data sets. Each of these data sets is run through the same analysis, and the differences in results between the analyses tells us how much extra uncertainty can be attributed to the 'made-up' data.

Imputation will be most effective when there are variables available that predict being missing or predict the values of the missing variable, but that are not going to be included in the predictive model. They might be excluded because they are not readily available in production use, or because they are not available until after the point when the prediction must be made, or because of face-validity or parsimony concerns.

Finally, it's important to note that imputation is not magic. It's reasonable to expect that missing-data bias will be reduced by multiple imputation, but the bias will only be completely removed if there are observed variables that can predict all the differences in missingness. This is called the *Missing At Random* assumption; it is provably required for any method that completely removes missing-data bias, but is unlikely to be exactly true in practice.

Chapter 3

Available implementations of multiple imputation

We evaluated the following imputation software

- `mi`, an R package using Bayesian generalised linear models (Gelman and Hill, 2011)
- `mice`, standing for *multiple imputation with chained equations*, available in R and Stata (van Buuren and Groothuis-Oudshoorn, 2011)
- `missForest` (Stekhoven and Buehlmann, 2012) and `missRanger` (Mayer, 2019), two R packages using random forests for imputation

- `MIDAS`, a Python program using two forms of deep neural network (variation autoencode and denoising autoencoder), which was in pre-alpha development (Oracen, 2018)

3.1 Computational scalability

We first considered computational scalability(Connor, 2018). We found that `mi` was unusable when the number of variables was more than 18 or the number of observations more than 1,000; it is not suitable for large-scale data analysis.

`mice` is usable for data sets with large numbers of observations and moderate numbers of variables. It ran in 0.25-1s per observation on datasets with 18 observations and 100 to 10,000 observations, but in 10-20s per observation on datasets with 59 variables. The ability to use large numbers of variables is important not only because the models of interest may include many variables, but because imputation will tend to be improved when additional variables are used.

`missRanger` and `MIDAS` were feasible even for quite large data sets, taking about

0.1s per observation with 59 variables and 10,000 observations. Both could be used with even larger numbers of variables or observations. `missForest` was slower, but still usable.

3.2 Imputation accuracy in healthcare settings

We analysed three predictive models using the range of imputation models (Lee, 2018). The first was a model for in-hospital mortality fitted to data from the US MIMIC-III intensive-care database. The second was a model for in-hospital mortality fitted the NZ National Minimum Dataset, and the third was a model for non-cardiac mortality after surgery in New Zealand.

Since `mice` is widely-used and accepted, we assumed it would give well-calibrated imputations on these datasets, and examined whether imputing using the machine-learning approaches gave similar results. The results were comparable for the two neural network approaches.

In contrast, `missRanger` gave imputed values that were much more variable than those from `mice`, suggesting that it had failed to incorporate all the information in the data. Excess variability in the predictions suggests that `missRanger` will be less successful in correcting bias, and will overestimate the uncertainty in the fitted model.

3.3 Conclusion

We have reservations about recommending `missRanger` because of its poorer accuracy in these three examples. We have different reservations about the neural-network methods provided by `MIDAS`: the software was not in a sufficiently finished state to be relied on.

It is likely that both random-forest and neural-network imputation engines will improve rapidly. `MIDAS` has already progressed from the version we evaluated, and has substantially better documentation of the code.

If there is a need to perform imputation in data sets too large for `mice` to be feasible, `missRanger` would be worth using at least as a sensitivity analysis for comparison to a complete-case analysis.

Chapter 4

Worked example

In this section we show how missing data can be described and imputed. The basic steps are

1. Explore the structure of the missing data
2. Run imputation software to produce a collection of possible complete datasets
3. Run the analysis on each data set
4. Combine the results

We will use R packages `mice` and `missRanger` to perform imputations, `mitools` (Lumley, 2019) to combine results, and `naniar` (Tierney et al., 2019) to explore missing data structure.

4.1 Example: Youth Risk Behavior Survey

From the Centers for Disease Control and Prevention (https://www.cdc.gov/healthyyouth/data/yrbs/index.htm)

> The Youth Risk Behavior Surveillance System (YRBSS) monitors six categories of health-related behaviors that contribute to the leading causes of death and disability among youth and adults, including—
>
> - Behaviors that contribute to unintentional injuries and violence
> - Sexual behaviors related to unintended pregnancy and sexually transmitted diseases, including HIV infection
> - Alcohol and other drug use
> - Tobacco use
> - Unhealthy dietary behaviors
> - Inadequate physical activity

> YRBSS also measures the prevalence of obesity and asthma and other health-related behaviors plus sexual identity and sex of sexual contacts.
>
> YRBSS includes a national school-based survey conducted by CDC and state, territorial, tribal, and local surveys conducted by state, territorial, and local education and health agencies and tribal governments.

We will fit a predictive model for asthma using age, exercise, height, weight, and smoking. The asthma variable is coded `1=Yes`, `2=No`, `3=Don't know`

```
library(dplyr)
library(purrr)
load("Data/yrbs15.rda")
dim(yrbs15)
```

```
## [1] 15624   240
```

```
yrbs15 <- select(yrbs15, q1, sex=q2, grade=q3, height=q6, weight=q7,
                 bullying=q23, ebullying=q25, wtperceived=q69, wtgoal=q70, exercise=q8
                 asthma=q87,sleep=q88, english=q99, smoke=qntob2, tobacco=qntob4, qnfr
                 qnfr3, qnveg1, qnveg3, qnsoda1, qnsoda3,overwt=qnowt, raceeth) %>%
          map_df(as.numeric) %>%
          mutate(age=q1+11, has_asthma=ifelse(asthma==3, NA, asthma==1)) %>%
          mutate(raceeth=factor(raceeth)) %>%
          select(-q1, -asthma)
dim(yrbs15)
```

```
## [1] 15624    23
```

First, consider just the complete cases

```
cc_model<-glm(has_asthma~sex*age+I(weight/height^2)+exercise+smoke,
            data=yrbs15, family=binomial,na.action=na.exclude)
summary(cc_model)
```

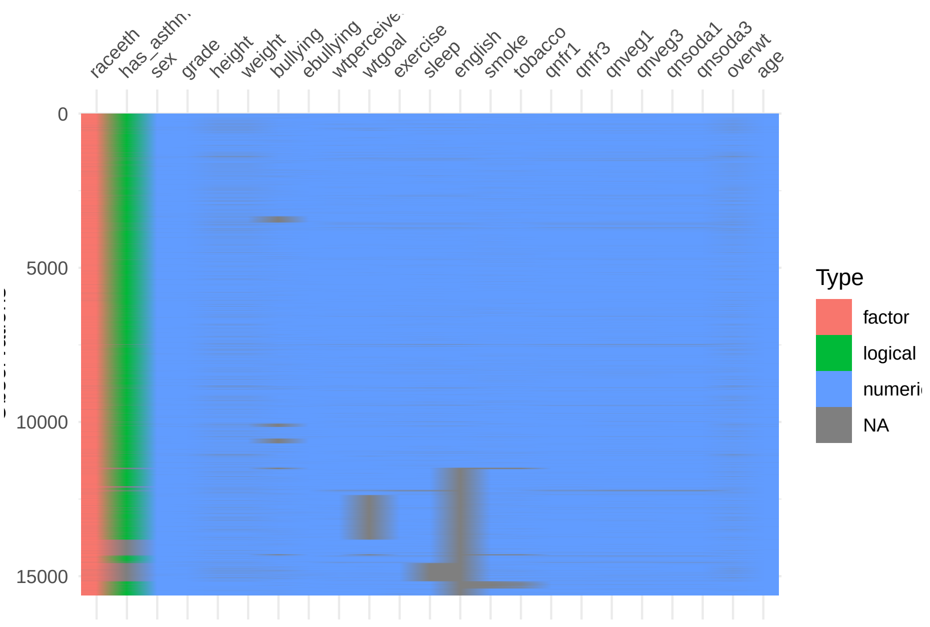
```
##
## Call:
## glm(formula = has_asthma ~ sex * age + I(weight/height^2) + exercise +
##     smoke, family = binomial, data = yrbs15, na.action = na.exclude)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.1500  -0.7544  -0.7130  -0.6387   1.8851
##
## Coefficients:
##                       Estimate Std. Error z value Pr(>|z|)
## (Intercept)          -2.187909   0.903773  -2.421   0.0155 *
## sex                   1.055178   0.565948   1.864   0.0623 .
## age                   0.064770   0.055457   1.168   0.2428
## I(weight/height^2)    0.027639   0.004075   6.782 1.19e-11 ***
## exercise              0.004964   0.008934   0.556   0.5785
## smoke                -0.306338   0.057552  -5.323 1.02e-07 ***
## sex:age              -0.070038   0.035162  -1.992   0.0464 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 13150  on 11829  degrees of freedom
## Residual deviance: 13068  on 11823  degrees of freedom
##   (3794 observations deleted due to missingness)
## AIC: 13082
##
## Number of Fisher Scoring iterations: 4
```

```
cc_pred<-predict(cc_model,type="response")
```

The output says there are 3891 observations deleted because of missing data.

First, look at missing data patterns

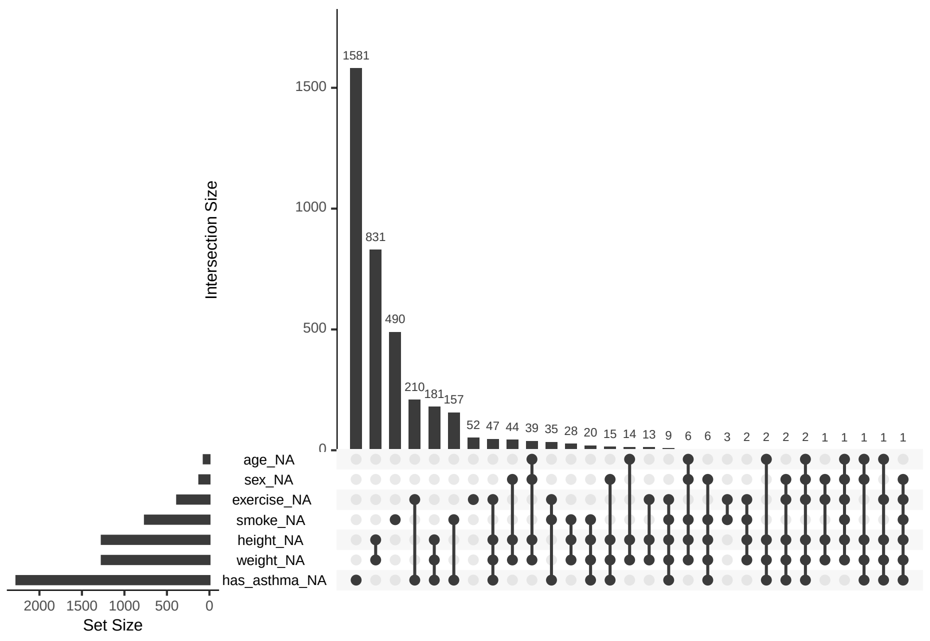
```
library(visdat)
library(naniar)
vis_dat(yrbs15)
```



We can see there is both sporadic missingness and some large blocks of missing values that indicate poor compliance at particular survey sites.

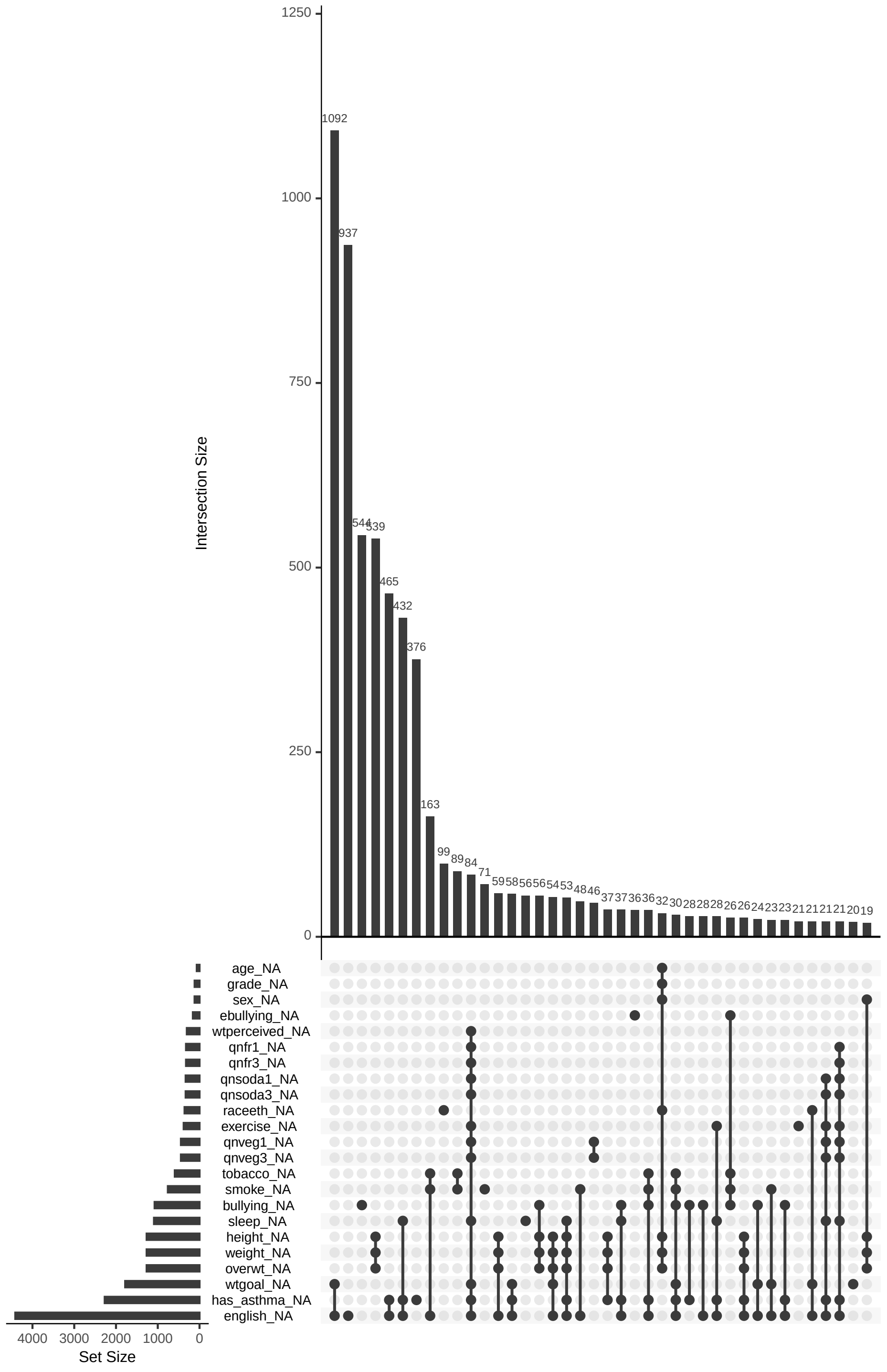
An UpSet plot shows the combinations more clearly. There are many people missing the `has_asthma` variable; there are some who have the asthma measurement but are missing data on height and weight, sleep, smoking or exercise.

```
gg_miss_upset(yrbs15[,c("has_asthma","sex","age","height","weight","exercise","smoke")]
```



The other variables in our reduced data set measure bullying, diet, perception of the teens own weight, diet, and facility with English. The dietary variables are less often missing than height and weight, and in different people, so they could be valuable. Sleep and bullying are also rarely missing

```
gg_miss_upset(yrbs15,nsets=23)
```


Intersection Size
1250
1000
750
500
250
0
1092
937
544
539
465
432
376
163
99
89
84
71
59
58
56
56
54
53
48
46
37
37
36
36
32
30
28
28
28
26
26
24
23
23
21
21
21
21
20
19
age_NA
grade_NA
sex_NA
ebullying_NA
wtperceived_NA
qnfr1_NA
qnfr3_NA
qnsoda1_NA
qnsoda3_NA
raceeth_NA
exercise_NA
qnveg1_NA
qnveg3_NA
tobacco_NA
smoke_NA
bullying_NA
sleep_NA
height_NA
weight_NA
overwt_NA
wtgoal_NA
has_asthma_NA
english_NA
4000
3000
2000
1000
0
Set Size

4.2 Imputation with `mice`

First, we will use `mice`.

The function `mice()` creates imputations. We will use $M = 20$ complete data sets. The literature varies on how large M should be, largely driven by improvements in computing over the three decades that multiple imputation has been in use. Where computationally feasible, I would recommend at least 20, and up to 100; if necessary, it may be possible to get away with as few as $M = 5$.

```
library(mice)
system.time(
  m_imputations <- mice(yrbs15, m=20, maxit=5, printFlag=FALSE)
)
```

```
##     user   system  elapsed
##  937.672  106.035 1044.745
```

We can now perform analyses with the 20 imputed data sets using the `with()` function and pool them using the `pool()` function

```
mice_models <- with(m_imputations,
                    glm(has_asthma~sex*age+I(weight/height^2)+exercise+smoke,
                       family=binomial,na.action=na.exclude)
                    )
pool(mice_models)$pooled[,c("estimate","fmi")]
```

```
##                         estimate       fmi
## (Intercept)         -1.465927041 0.1276096
## sex                  0.730912151 0.1177881
## age                  0.016863127 0.1083691
## I(weight/height^2)   0.029194860 0.2890587
## exercise             0.003309807 0.2548758
## smoke               -0.316716521 0.1585347
## sex:age             -0.048629590 0.1143963
```

```
summary(pool(mice_models))
```

```
##                         estimate   std.error  statistic        df
## (Intercept)         -1.465927041 0.832994421 -1.7598282 1099.8893
## sex                  0.730912151 0.517531995  1.4123033 1272.8830
## age                  0.016863127 0.050715247  0.3325061 1478.9480
## I(weight/height^2)   0.029194860 0.004151449  7.0324505  232.3686
## exercise             0.003309807 0.008849589  0.3740069  296.6139
## smoke               -0.316716521 0.053062390 -5.9687572  734.8548
## sex:age             -0.048629590 0.032166147 -1.5118251 1341.9575
##                          p.value
## (Intercept)         7.871490e-02
## sex                 1.581051e-01
## age                 7.395543e-01
## I(weight/height^2) 2.247269e-11
## exercise            7.086664e-01
## smoke               3.718402e-09
## sex:age             1.308138e-01
```

The imputation has been very successful: the fraction of missing information (`fmi`) about the parameters is small. We can now use the pooled regression coefficients to make predictions, and the pooled standard errors to evaluate the uncertainties in those predictions. The coefficient of body mass index has stayed about the same, but those for `sex`, `age`, and `smoke` have changed noticeably, and the predictions will be impacted.

Alternatively, we could extract predicted values from each model and pool these. Unless we have a strong belief in the accuracy of the predictive model, it should be better (though less convenient) to postpone the pooling as long as possible

```
mice_datasets<-complete(m_imputations, action="all")

mice_predictions<-lapply(mice_datasets,
                    function(dataset){
                      predict(glm(has_asthma~sex*age+I(weight/height^2)+exercise+smoke,
                       family=binomial,data=dataset), type="response")
                    })

length(mice_predictions)
```

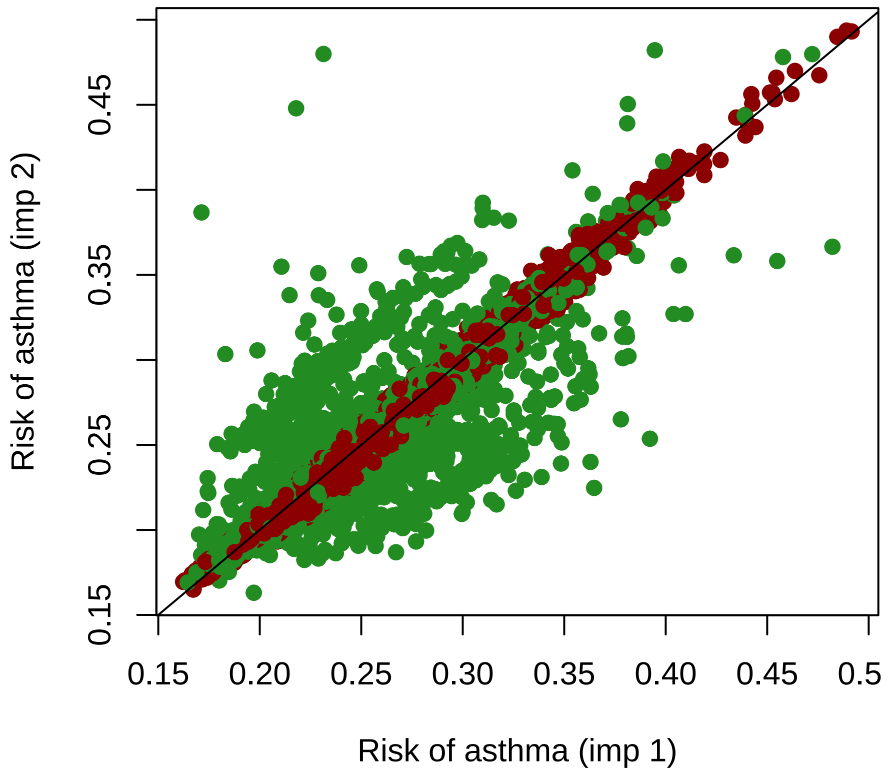
```
## [1] 20
```

```
length(mice_predictions[[1]])
```

```
## [1] 15624
```

Here we look at the relationship between the first two versions of the predicted value. Observations with complete data are in red, and those with missing data are in green

```
plot(mice_predictions[[1]],mice_predictions[[2]],
     col=ifelse(is.na(cc_pred),"forestgreen","darkred"),
     pch=19, xlab="Risk of asthma (imp 1)",ylab="Risk of asthma (imp 2)")
abline(0,1)
```



The green points are spread out around the diagonal line, reflecting that the two imputations have different values for the missing data. The red points are clustered fairly closely around the diagonal line; differences between the two predictions for these points reflect differences in the fitted model.

Our best estimate of the predictions is the average across all the impute values. We compare this to the predictions for the complete-case analysis

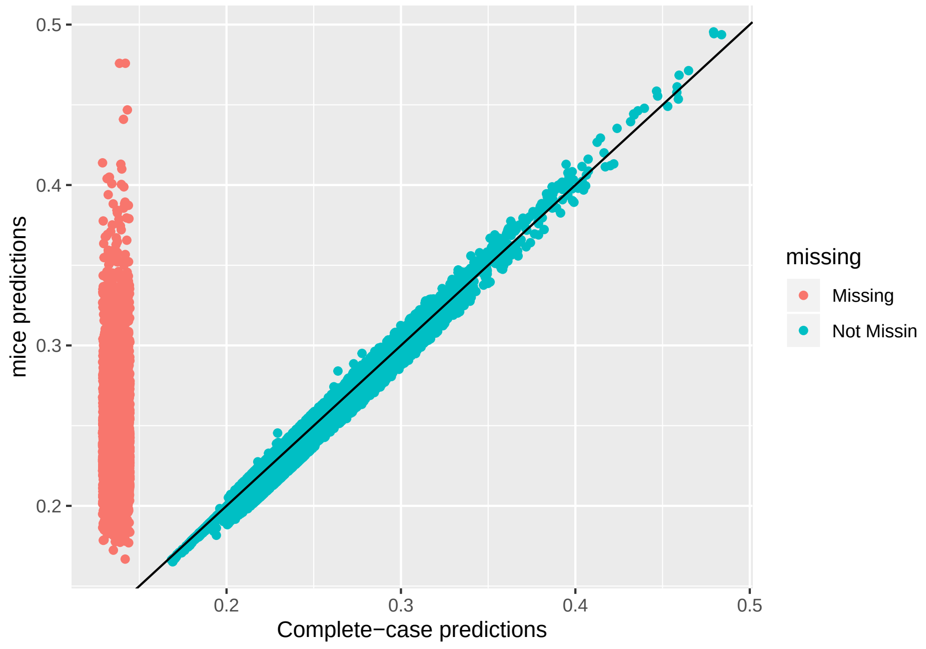
```
library(ggplot2)
mice_final_pred<-Reduce(`+`,mice_predictions)/length(mice_predictions)

summary(mice_final_pred)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.1651  0.2183  0.2345  0.2437  0.2620  0.4955
```

```
summary(cc_pred)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   0.168   0.220   0.235   0.244   0.261   0.484    3794
```

```
ggplot(data.frame(cc=cc_pred,mice=mice_final_pred), aes(x=cc,y=mice))+
  geom_miss_point()+
  xlab("Complete-case predictions")+ylab("mice predictions")+
  geom_abline(intercept=0,slope=1)
```



There are two benefits from imputation visible here. First, and most dramatically, the red observations do not have a prediction in complete-case data; the predictive model fails. Second, the predictions from the imputed data are higher at the high end and lower at the low end; there is a small amount of systematic bias

4.3 Imputation with missRanger

We need to make one data change to use `missRanger`: it does not allow logical (TRUE/FALSE) variables, so `has_asthma` must be recoded as `0/1`. There is also a change in how the imputations are created: `missRanger` only does one imputation each time it is invoked, so we use `replicate()` to invoke it M times. The `pmm.k` option controls the use of predictive mean matching (see @ref{pmm}), which ensures that imputed values only take on values that are already seen in the data

```
library(missRanger)
yrbs15$has_asthma<-as.integer(yrbs15$has_asthma)
system.time(
  r_imputations <- replicate(20, {
        missRanger(yrbs15, maxiter=5, pmm.k=5)
        },simplify = FALSE)
)
```

```
##
## Missing value imputation by random forests
##
##   Variables to impute:       sex, grade, height, weight, bullying, ebullying, wtperc
##   Variables used to impute:  sex, grade, height, weight, bullying, ebullying, wtperc
## iter 1:  .......................
## iter 2:  .......................
## iter 3:  .......................
##
## Missing value imputation by random forests
##
##   Variables to impute:       sex, grade, height, weight, bullying, ebullying, wtperc
##   Variables used to impute:  sex, grade, height, weight, bullying, ebullying, wtperc
## iter 1:  .......................
## iter 2:  .......................
## iter 3:  .......................
##
## Missing value imputation by random forests
##
##   Variables to impute:       sex, grade, height, weight, bullying, ebullying, wtperc
##   Variables used to impute:  sex, grade, height, weight, bullying, ebullying, wtperc
## iter 1:  .......................
## iter 2:  .......................
## iter 3:  .......................
##
## Missing value imputation by random forests
##
##   Variables to impute:       sex, grade, height, weight, bullying, ebullying, wtperc
##   Variables used to impute:  sex, grade, height, weight, bullying, ebullying, wtperc
## iter 1:  .......................
## iter 2:  .......................
## iter 3:  .......................
##
## Missing value imputation by random forests
##
##   Variables to impute:       sex, grade, height, weight, bullying, ebullying, wtperc
##   Variables used to impute:  sex, grade, height, weight, bullying, ebullying, wtperc
## iter 1:  .......................
## iter 2:  .......................
## iter 3:  .......................
## iter 4:  .......................
##
```

```
## Missing value imputation by random forests
##
##   Variables to impute:       sex, grade, height, weight, bullying, ebullying, wtperceived, wtg
##   Variables used to impute:  sex, grade, height, weight, bullying, ebullying, wtperceived, wtg
## iter 1:  .......................
## iter 2:  .......................
## iter 3:  .......................
##
## Missing value imputation by random forests
##
##   Variables to impute:       sex, grade, height, weight, bullying, ebullying, wtperceived, wtg
##   Variables used to impute:  sex, grade, height, weight, bullying, ebullying, wtperceived, wtg
## iter 1:  .......................
## iter 2:  .......................
## iter 3:  .......................
## iter 4:  .......................
##
## Missing value imputation by random forests
##
##   Variables to impute:       sex, grade, height, weight, bullying, ebullying, wtperceived, wtg
##   Variables used to impute:  sex, grade, height, weight, bullying, ebullying, wtperceived, wtg
## iter 1:  .......................
## iter 2:  .......................
## iter 3:  .......................
##
## Missing value imputation by random forests
##
##   Variables to impute:       sex, grade, height, weight, bullying, ebullying, wtperceived, wtg
##   Variables used to impute:  sex, grade, height, weight, bullying, ebullying, wtperceived, wtg
## iter 1:  .......................
## iter 2:  .......................
## iter 3:  .......................
##
## Missing value imputation by random forests
##
##   Variables to impute:       sex, grade, height, weight, bullying, ebullying, wtperceived, wtg
##   Variables used to impute:  sex, grade, height, weight, bullying, ebullying, wtperceived, wtg
## iter 1:  .......................
## iter 2:  .......................
## iter 3:  .......................
##
## Missing value imputation by random forests
##
##   Variables to impute:       sex, grade, height, weight, bullying, ebullying, wtperceived, wtg
##   Variables used to impute:  sex, grade, height, weight, bullying, ebullying, wtperceived, wtg
## iter 1:  .......................
## iter 2:  ......................
## iter 3:  ......................
##
## Missing value imputation by random forests
##
##   Variables to impute:       sex, grade, height, weight, bullying, ebullying, wtperc
##   Variables used to impute:  sex, grade, height, weight, bullying, ebullying, wtperc
## iter 1:  ......................
## iter 2:  ......................
## iter 3:  ......................
##
## Missing value imputation by random forests
##
##   Variables to impute:       sex, grade, height, weight, bullying, ebullying, wtperc
##   Variables used to impute:  sex, grade, height, weight, bullying, ebullying, wtperc
## iter 1:  ......................
## iter 2:  ......................
## iter 3:  ......................
## iter 4:  ......................
##
## Missing value imputation by random forests
##
##   Variables to impute:       sex, grade, height, weight, bullying, ebullying, wtperc
##   Variables used to impute:  sex, grade, height, weight, bullying, ebullying, wtperc
## iter 1:  ......................
## iter 2:  ......................
## iter 3:  ......................
## iter 4:  ......................
##
## Missing value imputation by random forests
##
##   Variables to impute:       sex, grade, height, weight, bullying, ebullying, wtperc
##   Variables used to impute:  sex, grade, height, weight, bullying, ebullying, wtperc
## iter 1:  ......................
## iter 2:  ......................
## iter 3:  ......................
## iter 4:  ......................
## iter 5:  ......................
##
## Missing value imputation by random forests
##
##   Variables to impute:       sex, grade, height, weight, bullying, ebullying, wtperc
##   Variables used to impute:  sex, grade, height, weight, bullying, ebullying, wtperc
## iter 1:  ......................
## iter 2:  ......................
## iter 3:  ......................
```

```
##
## Missing value imputation by random forests
##
##   Variables to impute:       sex, grade, height, weight, bullying, ebullying, wtperceived, wtg
##   Variables used to impute:  sex, grade, height, weight, bullying, ebullying, wtperceived, wtg
## iter 1:  ......................
## iter 2:  ......................
## iter 3:  ......................
## iter 4:  ......................
## iter 5:  ......................
##
## Missing value imputation by random forests
##
##   Variables to impute:       sex, grade, height, weight, bullying, ebullying, wtperceived, wtg
##   Variables used to impute:  sex, grade, height, weight, bullying, ebullying, wtperceived, wtg
## iter 1:  ......................
## iter 2:  ......................
## iter 3:  ......................
##
## Missing value imputation by random forests
##
##   Variables to impute:       sex, grade, height, weight, bullying, ebullying, wtperceived, wtg
##   Variables used to impute:  sex, grade, height, weight, bullying, ebullying, wtperceived, wtg
## iter 1:  ......................
## iter 2:  ......................
## iter 3:  ......................
##
## Missing value imputation by random forests
##
##   Variables to impute:       sex, grade, height, weight, bullying, ebullying, wtperceived, wtg
##   Variables used to impute:  sex, grade, height, weight, bullying, ebullying, wtperceived, wtg
## iter 1:  ......................
## iter 2:  ......................
## iter 3:  ......................

##      user    system   elapsed
## 13240.713   300.419  3976.111
```

The output from this code is like the output from `complete()` in `mice`, a list of data sets. We can combine them using imputation-combining functions from the `mitools` package

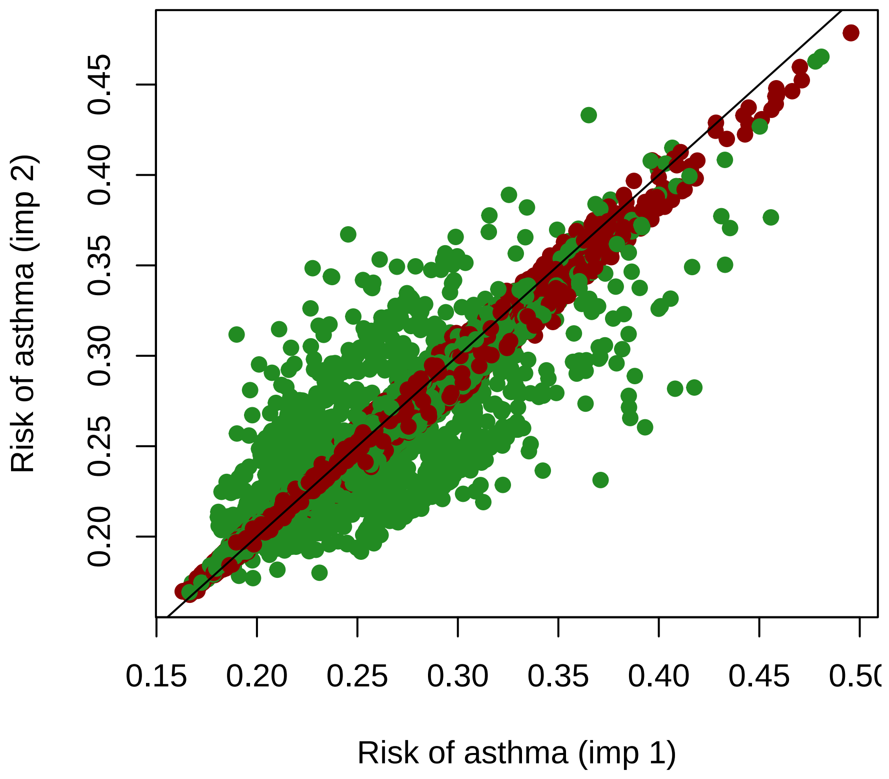
```
library(mitools)
ranger_imps<-imputationList(r_imputations)
ranger_models<- with(ranger_imps,
                    glm(has_asthma~sex*age+I(weight/height^2)+exercise+smoke,
                       family=binomial,na.action=na.exclude)
                    )
summary(MIcombine(ranger_models))
```

```
## Multiple imputation results:
##       with(ranger_imps, glm(has_asthma ~ sex * age + I(weight/height^2) +
##     exercise + smoke, family = binomial, na.action = na.exclude))
##       MIcombine.default(ranger_models)
##                          results          se      (lower       upper)
## (Intercept)        -1.853724781 0.835663183 -3.49329541 -0.214154152
## sex                 0.891706608 0.535771825 -0.16041311  1.943826325
## age                 0.037947080 0.051786897 -0.06368223  0.139576392
## I(weight/height^2)  0.027932408 0.003841455  0.02039265  0.035472162
## exercise            0.002305394 0.008236694 -0.01385667  0.018467457
## smoke              -0.278113393 0.055418525 -0.38706296 -0.169163827
## sex:age            -0.058270843 0.033210005 -0.12347648  0.006934792
##                    missInfo
## (Intercept)            13 %
## sex                    18 %
## age                    14 %
## I(weight/height^2)     15 %
## exercise               14 %
## smoke                  22 %
## sex:age                17 %
```

Next, we find the predicted values for each set of imputations and average the multiple predictions for each variable. Again, the red indicates predicted values where there is no missing data, so that the differences are due to differences in the trained model; the green indicates differences where there is missing data and shows the uncertainty in individual imputations.

The scatter of green points is less structured than with `mice`, because the predictions are averaged over many trees. The lack of structure shows that many variables have contributed small amounts to predicting the missing values. Whether this is good or bad depends on the setting. If there is actually strong predictive information in just a few variables it's bad; if the informative really is diffusely spread across many variables it's good. In this case I think it's bad.

```
ranger_predictions<-with(ranger_imps,{
    predict(glm(has_asthma~sex*age+I(weight/height^2)+exercise+smoke,
                       family=binomial), type="response")
})

plot(ranger_predictions[[1]],ranger_predictions[[2]],
     col=ifelse(is.na(cc_pred),"forestgreen","darkred"),
     pch=19, xlab="Risk of asthma (imp 1)",ylab="Risk of asthma (imp 2)")
abline(0,1)
```



The plot comparing complete-case to `missRanger` predictions shows a different pattern from that for `mice`. We now are seeing lower predictions at the high end, not higher.

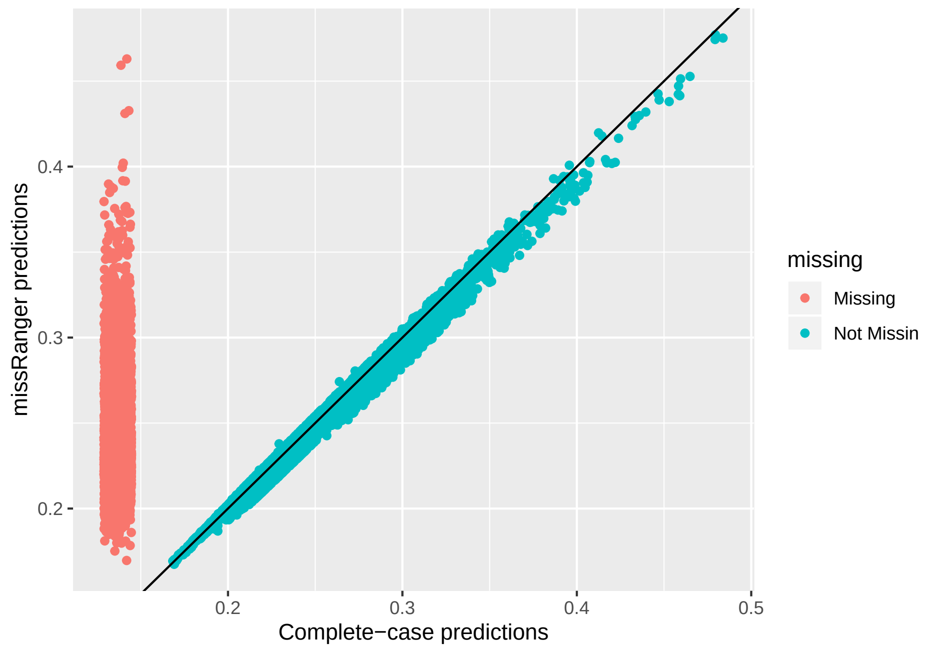
```
library(ggplot2)
ranger_final_pred<-Reduce(`+`,ranger_predictions)/length(ranger_predictions)

summary(ranger_final_pred)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.1674  0.2192  0.2342  0.2427  0.2596  0.4771
```

```
summary(cc_pred)
```

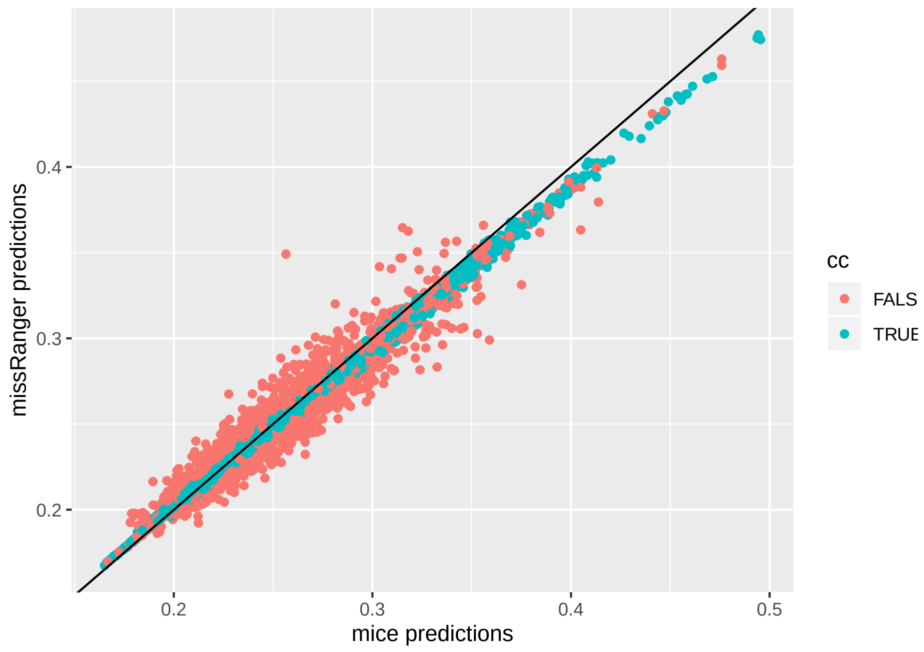
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   0.168   0.220   0.235   0.244   0.261   0.484    3794
```

```
ggplot(data.frame(cc=cc_pred,ranger=ranger_final_pred), aes(x=cc,y=ranger))+
  geom_miss_point()+
  xlab("Complete-case predictions")+ylab("missRanger predictions")+
  geom_abline(intercept=0,slope=1)
```



Comparing the predictions from `missRanger` and `mice` confirms that `missRanger` is predicting lower values at the high end, and shows considerable scatter. Because `mice` has been more thoroughly tested, and because we expect the information to come mostly from a small number of variables, we would trust `mice` more here. It's still useful to know how they compare.

```
ggplot(data.frame(mice=mice_final_pred,ranger=ranger_final_pred,cc=!is.na(cc_pred)), ae
  geom_point()+
  xlab("mice predictions")+ylab("missRanger predictions")+
  geom_abline(intercept=0,slope=1)
```



missRanger predictions
0.4
0.3
0.2
0.2
0.3
0.4
0.5
mice predictions
cc
FALS
TRUE

Chapter 5

Some technical details

In this section M is the number of imputed datasets, which are indexed by $m = 1, \ldots, M$.

5.1 Formulas for combining multiply-imputed datasets

Suppose we are interested in estimating a parameter β. Given M coefficient estimates $\hat{\beta}^*_1, \ldots \hat{\beta}^*_M$ based on a collection of imputed datasets, the overall estimate is

$$\bar{\beta} = \frac{1}{M} \sum_{m=1}^{M} \hat{\beta}^*_m$$

If in addition we have M variance estimates $\hat{\sigma}^2_1, \ldots, \hat{\sigma}^2_M$ of the variance of $\hat{\beta}_1, \ldots, \hat{\beta}_M$ the estimated variance is

$$\bar{\sigma}^2 = \frac{1}{M} \sum_{m=1}^{M} \hat{\sigma}^2_m + \frac{1}{M-1} \sum_{m=1}^{M} \left(\hat{\beta}^*_m - \bar{\beta} \right)^2$$

The first term in this equation estimates the variance we would have with complete data; the second term is the additional variance due to having missing data. If the missing data could be imputed perfectly, the second term would be zero; if the missing data cannot be predicted at all, the second term will be large.

The *fraction of missing information* estimates how much of the information about β has been lost due to missing data. This can be much smaller than the proportion of cases with missing information, because imputation makes use of the partial information present in incomplete observations. Heuristically, the fraction of missing information just the second term in the variance formula divided by the whole variance; in practice we use a more complicated formula that behaves better when M is small.

5.2 Chained equations

An obvious difficulty in creating a model to predict missing values is that there will be missing values in more than one variable – where do you start? The chained-equations strategy is to start with random noise and to iterate: predict the first variable from the rest, the second variable from the new values of the first and the old values of the rest, the third from the new values of the first two and the old values of the rest, and so on through all the variable and for many iterations.

This iterative process is a *Markov chain*, a class of random processes that is very thoroughly studied in statistics and probability. Under fairly weak assumptions (basically, that it doesn't get stuck in a loop) a Markov chain will forget its starting values and sample randomly from a unique *stationary distribution* defined by the models predicting each variable from all the others.

Both `mice` and the random-forest techniques use chained equations to construct imputations. The random-forest packages construct M separate Markov chains; `mice` runs one Markov chain long enough to extract M datasets from it.

5.3 Predictive Mean Matching

One of the complications with imputed data is that the imputation model may not know about all the constraints on the true variable. Age may be recorded to the nearest year but be imputed as a continuous variable. Salary will be non-negative in the observed data but might be imputed as negative. There are two reasons this can be problematic. First, it is bad for face validity. Second, while it's typically not a problem for traditional statistical analyses, it presents concern for deep neural networks, which are known for using unintended data features to distort predictions.

Predictive mean matching is a way to ensure that imputations do not introduce imputed values that are not present anywhere in the original data for the variable. First, predicted mean values are generated for every observation, missing or not. For each missing observation, we find the observed value with the closest predicted mean (or choose randomly from the few with the closest means). We use this observed value as the imputed value.

5.4 Random forests

Classification and regression tree predictors work by splitting the data into two groups, then splitting each of those into two groups, and so on recursively. Single trees are usually not very good predictive models, but collections of trees can be.

A random forest predictive model fits a large collection of trees, with each tree using a randomly chosen subset of variables. Each tree gives a prediction, and the prediction from the forest is the average (for a continuous variable) or a majority vote (for a categorical variable).

Random forests give good-quality 'black box' predictions. They can straightforwardly be implemented to run on large data sets and take advantage of parallel computing, since each tree is constructed independently and each split results in a smaller data set.

5.5 Autoencoders

All the imputation approaches express the data as a model plus noise, estimating the model part and sampling the noise at random. The previous approaches create the model one variable at a time. Autoencoders, by contrast, create a multivariate model for combinations of the variables, and sample the noise for combinations of the variables.

The idea of an autoencoder is to have a deep neural network with a narrow bottleneck in the middle layer and with (mostly) symmetric layers before and after. The low-dimensional bottleneck layer represents the 'model' part of the data; the layers before the bottleneck learn the encoding down to the low-dimensional representation; the layers after the bottleneck expand to the full distribution. If the neural network used linear transformations rather than the nonlinear transformations it actually uses, an autoencoder would perform principal components analysis, with the first few principal components used as the model and the remainder treated as noise.

Chapter 6

Acknowledgments

This report is part of a project funded by Precision Driven Health. The report builds on work by Jiunn Howe Lee and Stephen Connor that was first presented in their dissertations for the Master of Professional Studies in Data Science at the University of Auckland.

References

Bibliography

Connor, S. (2018). Multiple imputation applied to synthetic datasets for the imputation of missing health data. Technical report.

External Data Quality Panel (2019). Initial report of the 2018 census external data quality panel. Technical report.

Gelman, A. and Hill, J. (2011). Opening windows to the black box. *Journal of Statistical Software*, 40.

Lee, J. H. (2018). Multiple imputation applied to real world datasets for the imputation of missing health data. Technical report.

Lumley, T. (2019). *mitools: Tools for Multiple Imputation of Missing Data.* R package version 2.4.

Mayer, M. (2019). *missRanger: Fast Imputation of Missing Values.* R package version 1.0.4.

Oracen (2018). Multiple imputation utilising denoising autoencoder for approximate Bayesian inference.

Rubin, D. B. (1987). *Multiple Imputation for Nonresponse in Surveys.* John Wiley & Sons.

Stekhoven, D. J. and Buehlmann, P. (2012). Missforest - non-parametric missing value imputation for mixed-type data. *Bioinformatics*, 28(1):112–118.

Tierney, N., Cook, D., McBain, M., and Fay, C. (2019). *naniar: Data Structures, Summaries, and Visualisations for Missing Data.* R package version 0.4.2.

van Buuren, S. and Groothuis-Oudshoorn, K. (2011). mice: Multivariate imputation by chained equations in r. *Journal of Statistical Software*, 45(3):1–67.