GEAVANCEERDE DATA TECHNIEKEN

Gerko Vink
g.vink@uu.nl
Methodology & Statistics @ Utrecht University

10 Jun 2025

DISCLAIMER

I owe a debt of gratitude to many people as the thoughts and code in these slides are the process of years-long development cycles and discussions with my team, friends, colleagues and peers. When someone has contributed to the content of the slides, I have credited their authorship.

Images are either directly linked, or generated with StableDiffusion or DALL-E. That said, there is no information in this presentation that exceeds legal use of copyright materials in academic settings, or that should not be part of the public domain.

> ⚠ **Warning**
>
> You **may use** any and all content in this presentation - including my name - and submit it as input to generative AI tools, with the following **exception**:
>
> - You must ensure that the content is not used for further training of the model

SLIDE MATERIALS AND SOURCE CODE

Materials

- lecture slides on Moodle
- course page: www.gerkovink.com/sur
- source: github.com/gerkovink/sur

RECAP

Gisteren hebben we deze onderwerpen behandeld:

- Zelf functies ontwikkelen, gebruiken en debuggen
- Map / Reduce workflows
- Binaire operators
- Trekken uit verdelingen
- Random number generation

TODAY

Vandaag behandelen we de volgende onderwerpen:

- Ontbrekende waarden
- Synthetische imputaties maken

PACKAGES WE USE

```
library(purrr)    # for functional programming
library(dplyr)    # for data manipulation
library(magrittr) # for the pipe operator
library(mice)     # for multiple imputation
library(ggmice)   # for visualizing mice objects

set.seed(123) # for reproducibilitysqrt
```

ANATOMY OF AN ANSWER

AT THE START

Let’s start with the core:

> **Statistical inference**
>
> Statistical inference is the process of drawing conclusions from **truths**

Truths are boring, but they are convenient.

- however, for most problems truths require a lot of calculations, tallying or a complete census.
- therefore, a proxy of the truth is in most cases sufficient
- An example for such a proxy is a **sample**
- Samples are widely used and have been for a long time

DO WE NEED DATA?

Without any data we can still come up with a statistically valid answer.

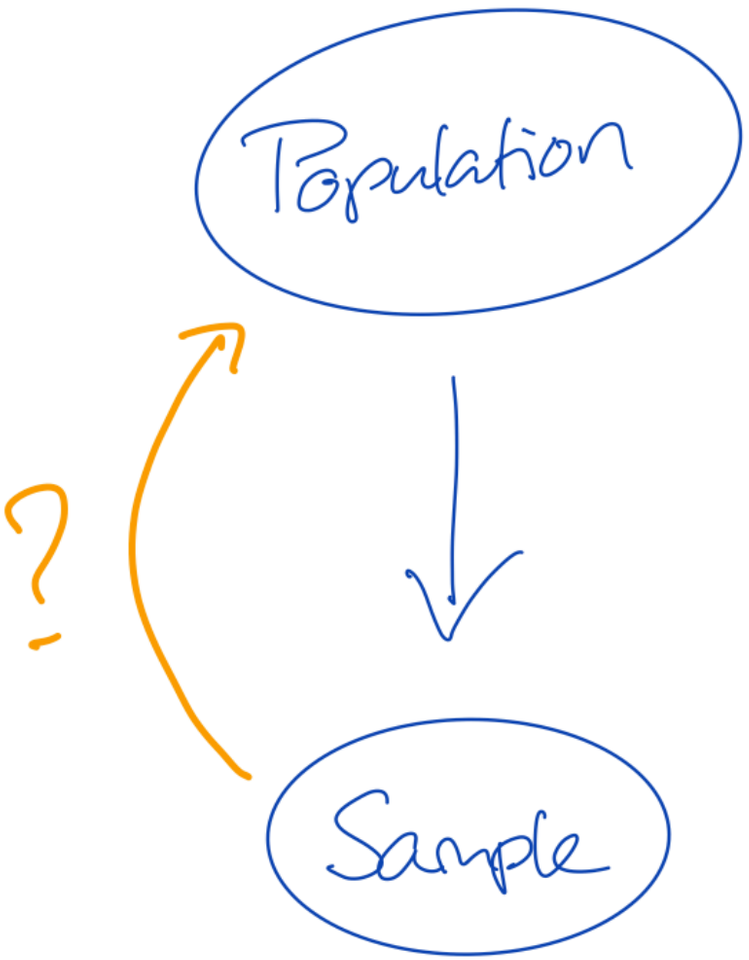
- The answer will not be very *informative*.
- In order for our answer to be more informative, we need more **information**

Some sources of information can already tremendously guide the precision of our answer.

In Short

Information bridges the answer to the truth. Too little information may lead you to a *false truth*.

BEING WRONG ABOUT THE TRUTH

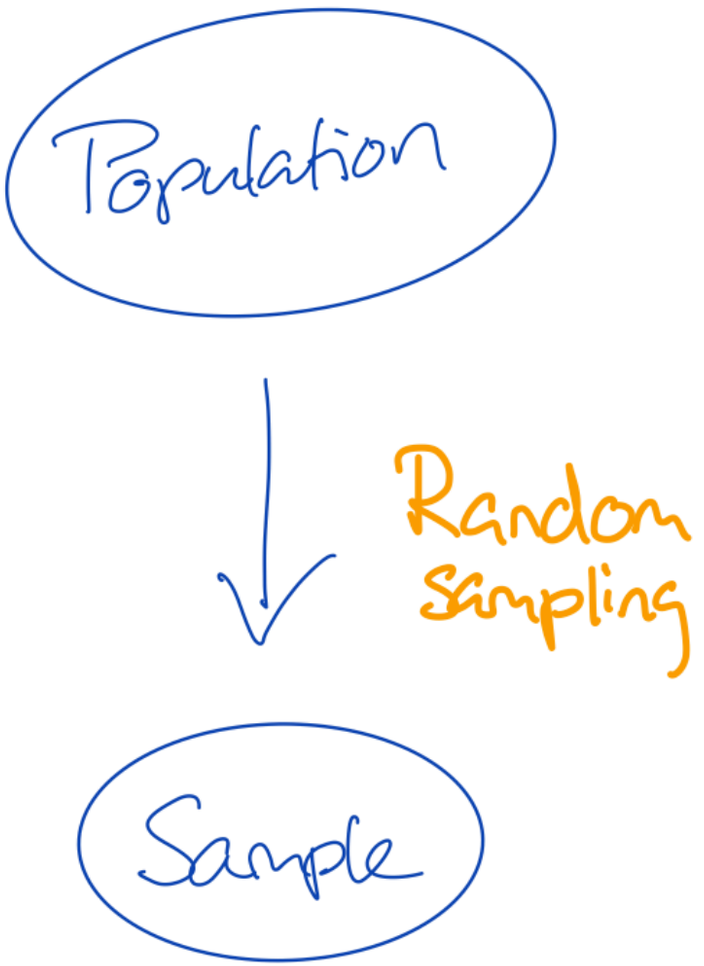


- The population is the truth
- The sample comes from the population, but is generally smaller in size
- This means that not all cases from the population can be in our sample
- If not all information from the population is in the sample, then our sample may be *wrong*

Good questions to ask yourself

1. Why is it important that our sample is not wrong?
2. How do we know that our sample is not wrong?

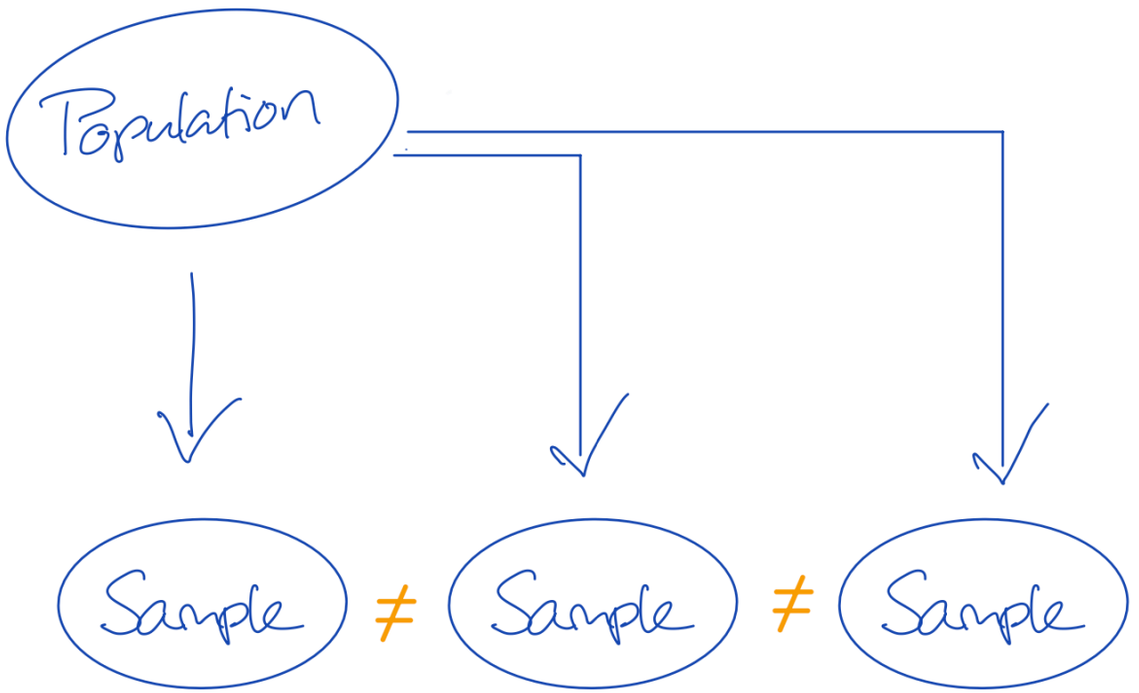
SOLVING THE MISSINGNESS PROBLEM



- There are many flavours of sampling
- If we give every unit in the population the same probability to be sampled, we do **random sampling**
- The convenience with random sampling is that the missingness problem can be ignored
- The missingness problem would in this case be: **not every unit in the population has been observed in the sample**

> ⚠ **Hmmm…**
>
> Would that mean that if we simply observe every potential unit, we would be unbiased about the truth?

HOW DO WE KNOW THAT OUR SAMPLE IS NOT....

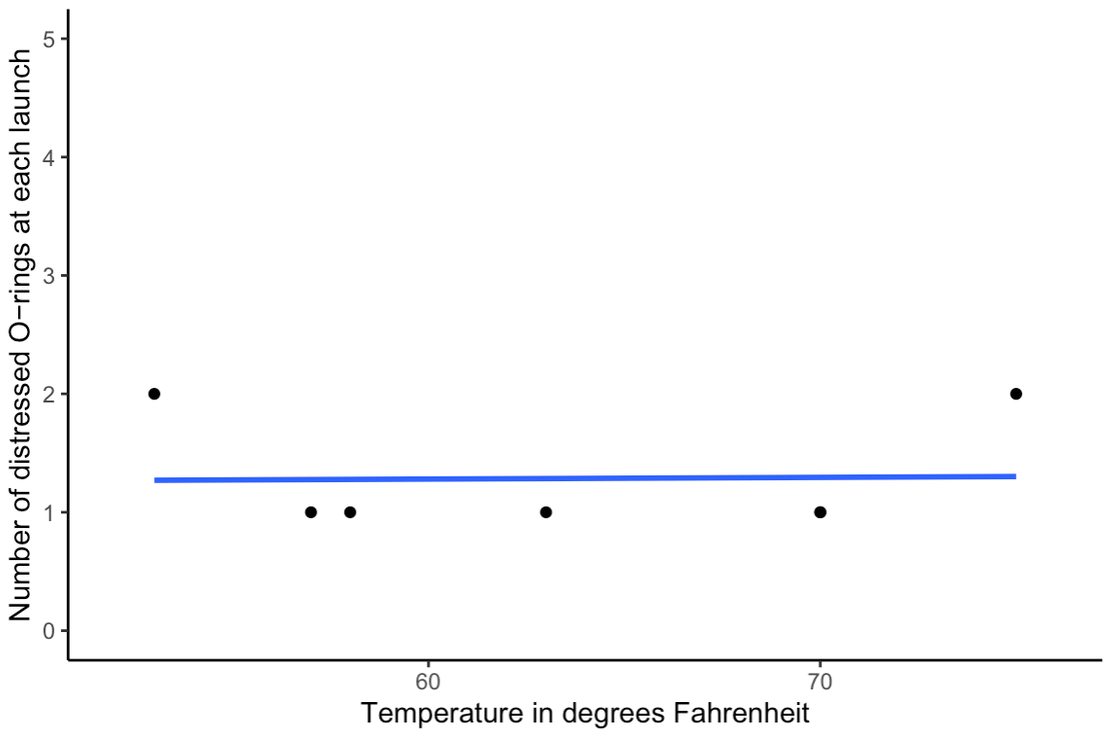


We can replicate our sample.

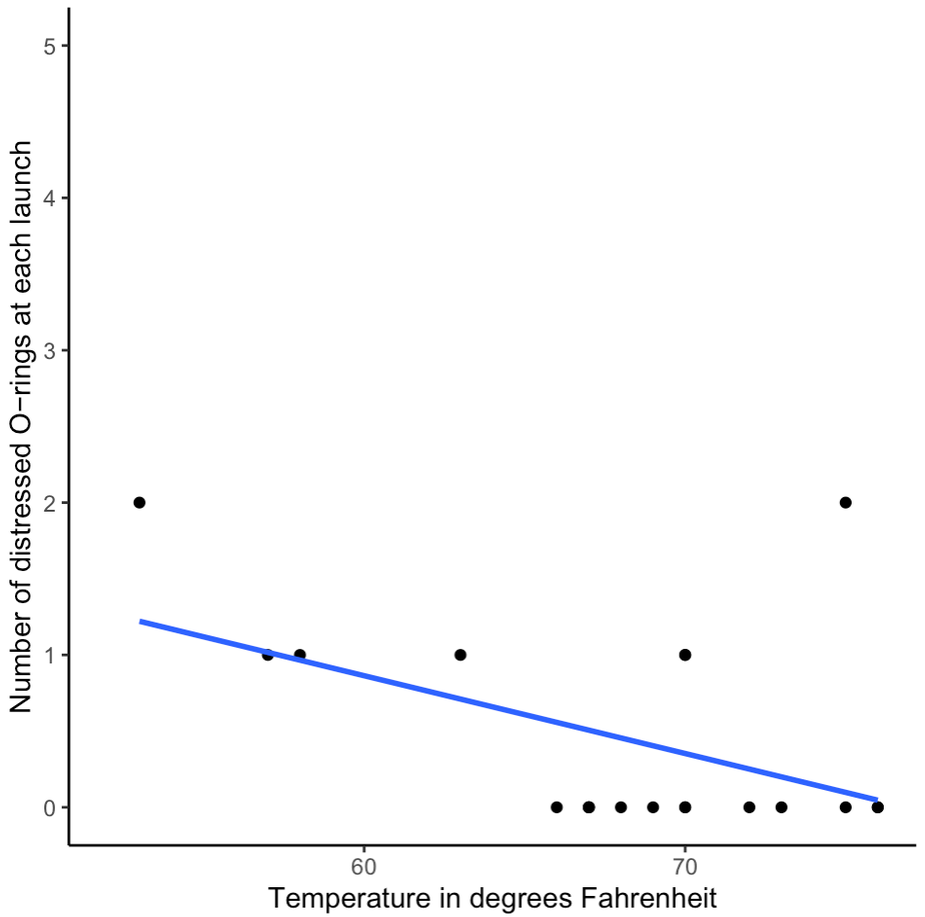
- A replication would be a new sample from the same population or true data generating model obtained by the same data generating process.
- If we would sample 100 times, we would get 100 different samples
- If we would estimate 100 times, we would get 100 different estimates with 100 different confidence intervals (e.g. 95% CI)
- Out of these 100 different intervals, we would expect a nominal coverage. For a 95% CI we'd expect 95 of them to cover the true population value.

CASE: SPACESHUTTLE CHALLENGER

36 years ago, on 28 January 1986, 73 seconds into its flight and at an altitude of 9 miles, the space shuttle Challenger experienced an enormous fireball caused by one of its two booster rockets and broke up. The crew compartment continued its trajectory, reaching an altitude of 12 miles, before falling into the Atlantic. All seven crew members, consisting of five astronauts and two payload specialists, were killed.



NOTHING HAPPENED, SO WE IGNORED IT



In the decision to proceed with the launch, there was a presence of dark data. And no-one noticed!

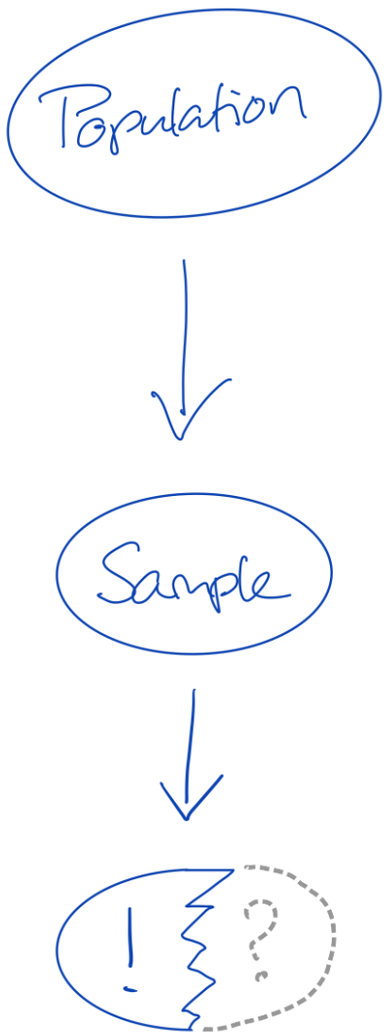
Dark data

> Information that is not available but necessary to arrive at the correct answer.

This missing information has the potential to mislead people. The notion that we can be misled is essential because it also implies that artificial intelligence can be misled!

> ⚠ If you don't have all the information, there is always the possibility of drawing an incorrect conclusion or making a wrong decision.

IN PRACTICE

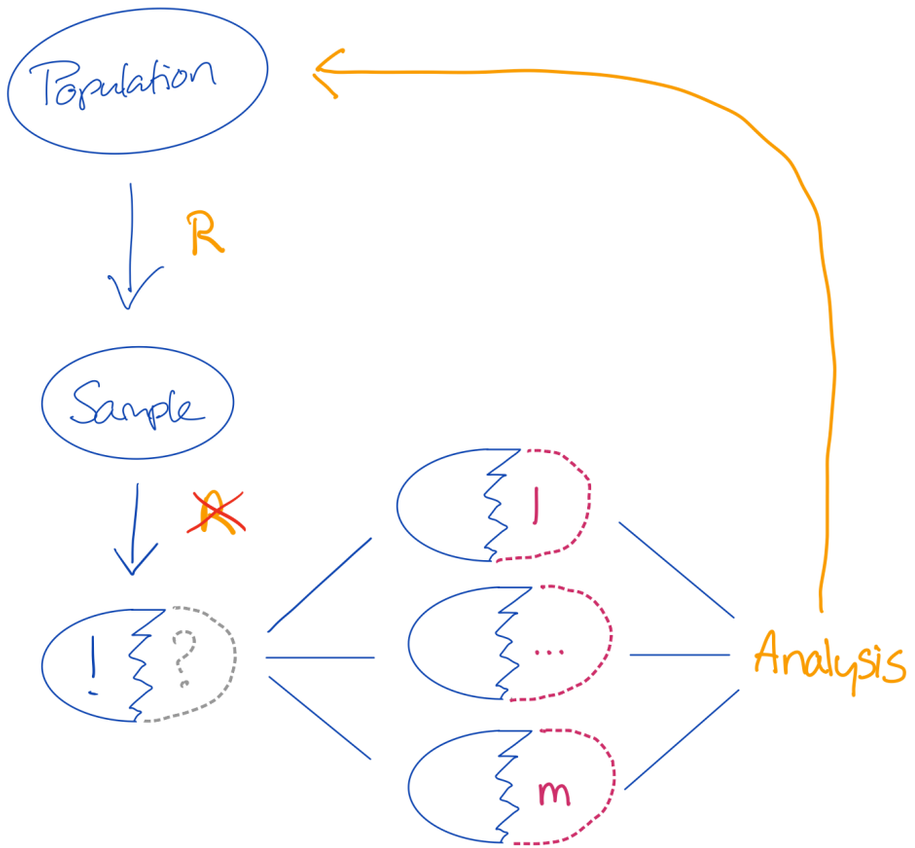


We now have a new problem:

- we do not have the whole truth; but merely a sample of the truth
- we do not even have the whole sample, but merely a sample of the sample of the truth.

> What would be a simple solution to allowing for valid inferences on the incomplete sample? Would that solution work in practice?

HOW TO FIX THE MISSINGNESS PROBLEM



There are two sources of uncertainty that we need to cover when analyzing incomplete data:

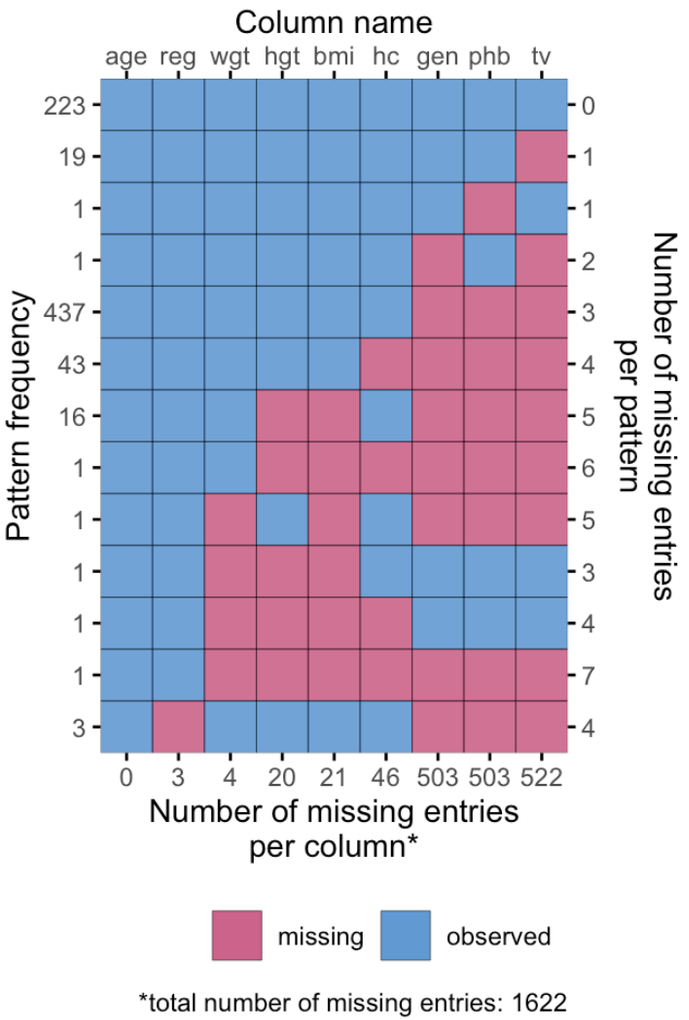
1. **Uncertainty about the data values we don't have**: when we don't know what the true observed value should be, we must create a distribution of values with proper variance (uncertainty).
2. **Uncertainty about the process that generated the values we do have**: nothing can guarantee that our sample is the one true sample. So it is reasonable to assume that the parameters obtained on our sample are biased.

A straightforward and intuitive solution for analyzing incomplete data in such scenarios is *multiple imputation* (Rubin, 1987).

MULTIPLE IMPUTATION WITH `mice`

INSPECT THE MISSINGNESS

```
1 plot_pattern(boys)
```



IMPUTE BOYS

```
1 imp <- mice(boys)
```

```
iter imp variable
 1   1  hgt  wgt  bmi  hc  gen  phb  tv  reg
 1   2  hgt  wgt  bmi  hc  gen  phb  tv  reg
 1   3  hgt  wgt  bmi  hc  gen  phb  tv  reg
 1   4  hgt  wgt  bmi  hc  gen  phb  tv  reg
 1   5  hgt  wgt  bmi  hc  gen  phb  tv  reg
 2   1  hgt  wgt  bmi  hc  gen  phb  tv  reg
 2   2  hgt  wgt  bmi  hc  gen  phb  tv  reg
 2   3  hgt  wgt  bmi  hc  gen  phb  tv  reg
 2   4  hgt  wgt  bmi  hc  gen  phb  tv  reg
 2   5  hgt  wgt  bmi  hc  gen  phb  tv  reg
 3   1  hgt  wgt  bmi  hc  gen  phb  tv  reg
 3   2  hgt  wgt  bmi  hc  gen  phb  tv  reg
 3   3  hgt  wgt  bmi  hc  gen  phb  tv  reg
 3   4  hgt  wgt  bmi  hc  gen  phb  tv  reg
 3   5  hgt  wgt  bmi  hc  gen  phb  tv  reg
 4   1  hgt  wgt  bmi  hc  gen  phb  tv  reg
 4   2  hgt  wgt  bmi  hc  gen  phb  tv  reg
```

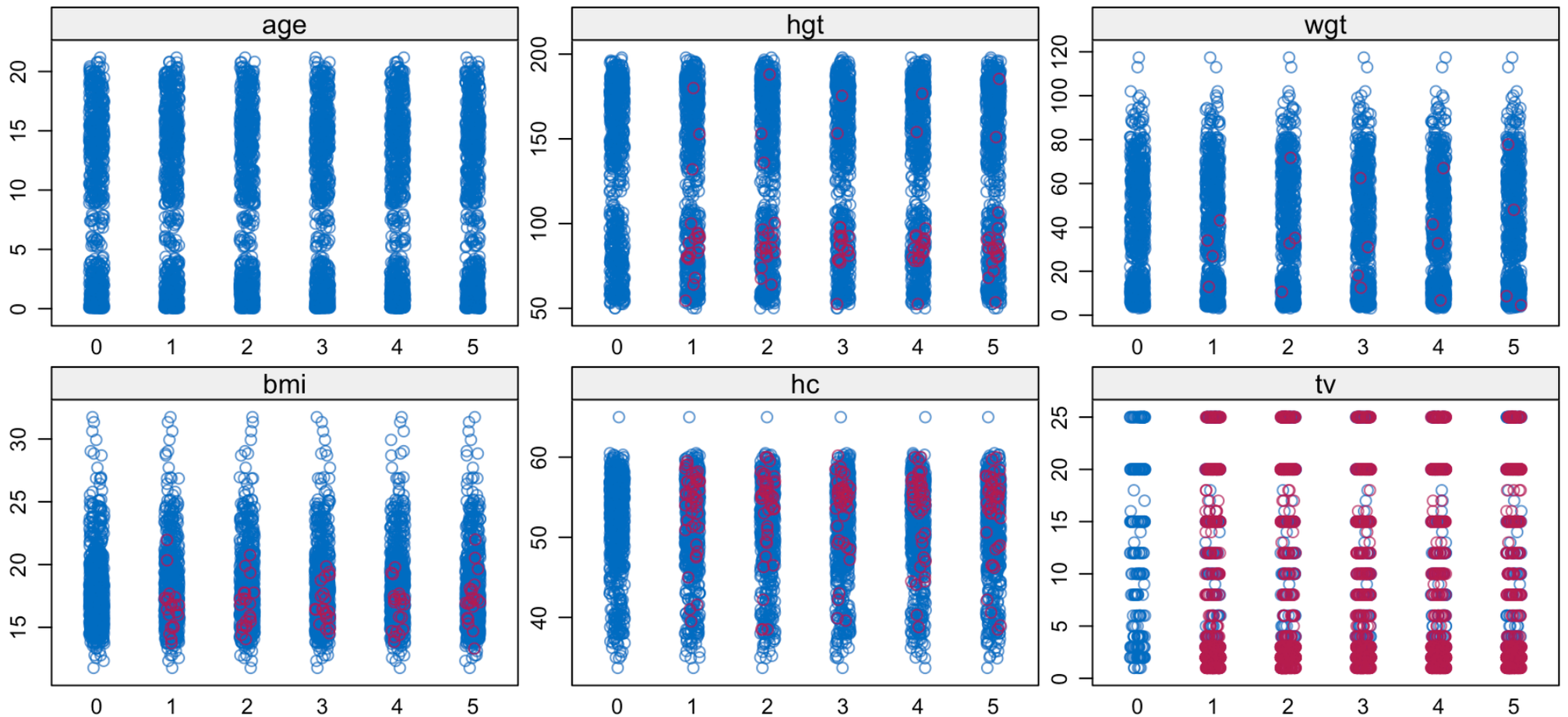

DEFAULT ARGUMENTS

```
mice(
  data,
  m = 5,
  method = NULL,
  predictorMatrix,
  ignore = NULL,
  where = NULL,
  blocks,
  visitSequence = NULL,
  formulas,
  calltype = NULL,
  blots = NULL,
  post = NULL,
  defaultMethod = c("pmm", "logreg", "polyreg", "polr"),
  maxit = 5,
  printFlag = TRUE,
  seed = NA,
  data.init = NULL,
  ...
)
```

For this course we do not go beyond the default imputation methods in `mice`.

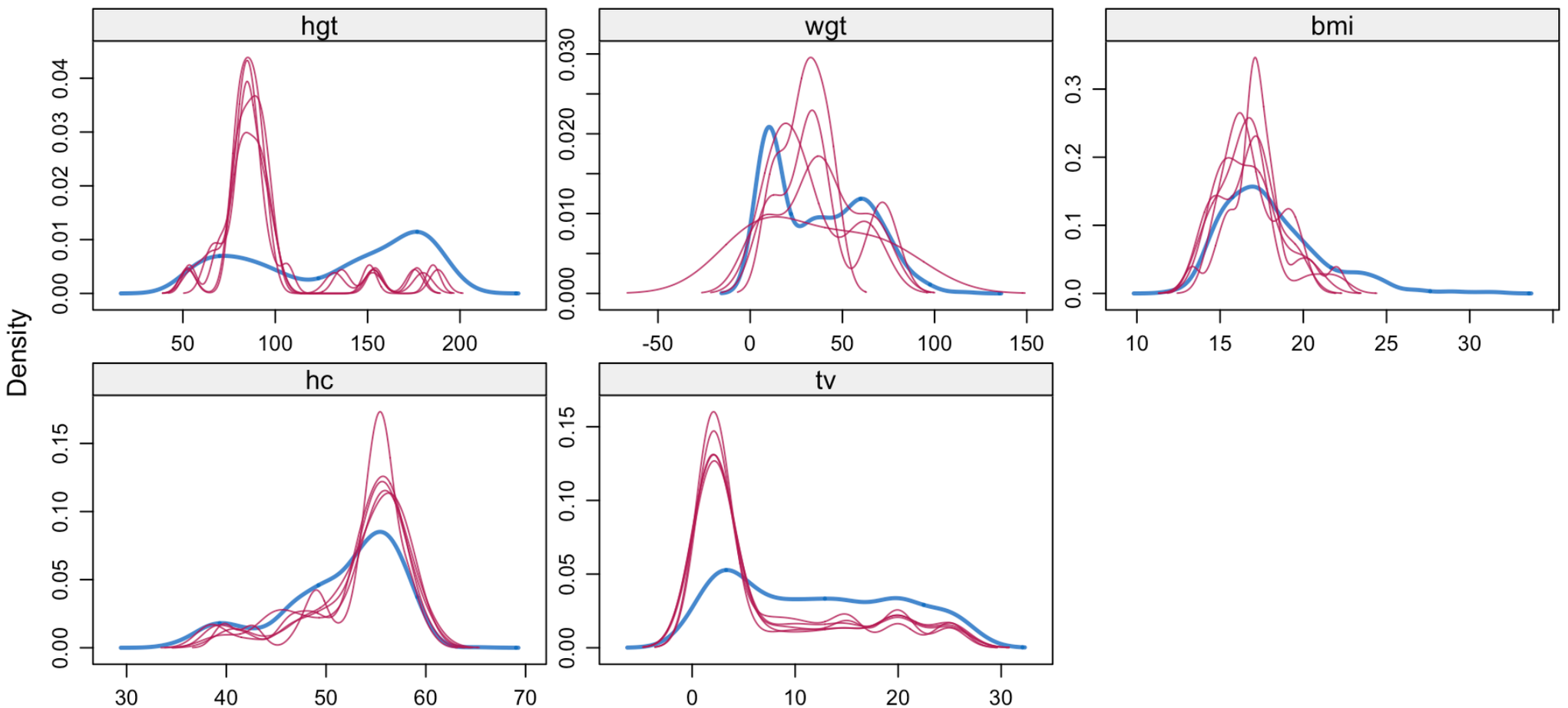
INSPECT THE IMPUTED DATA

```
1  mice::stripplot(imp)
```



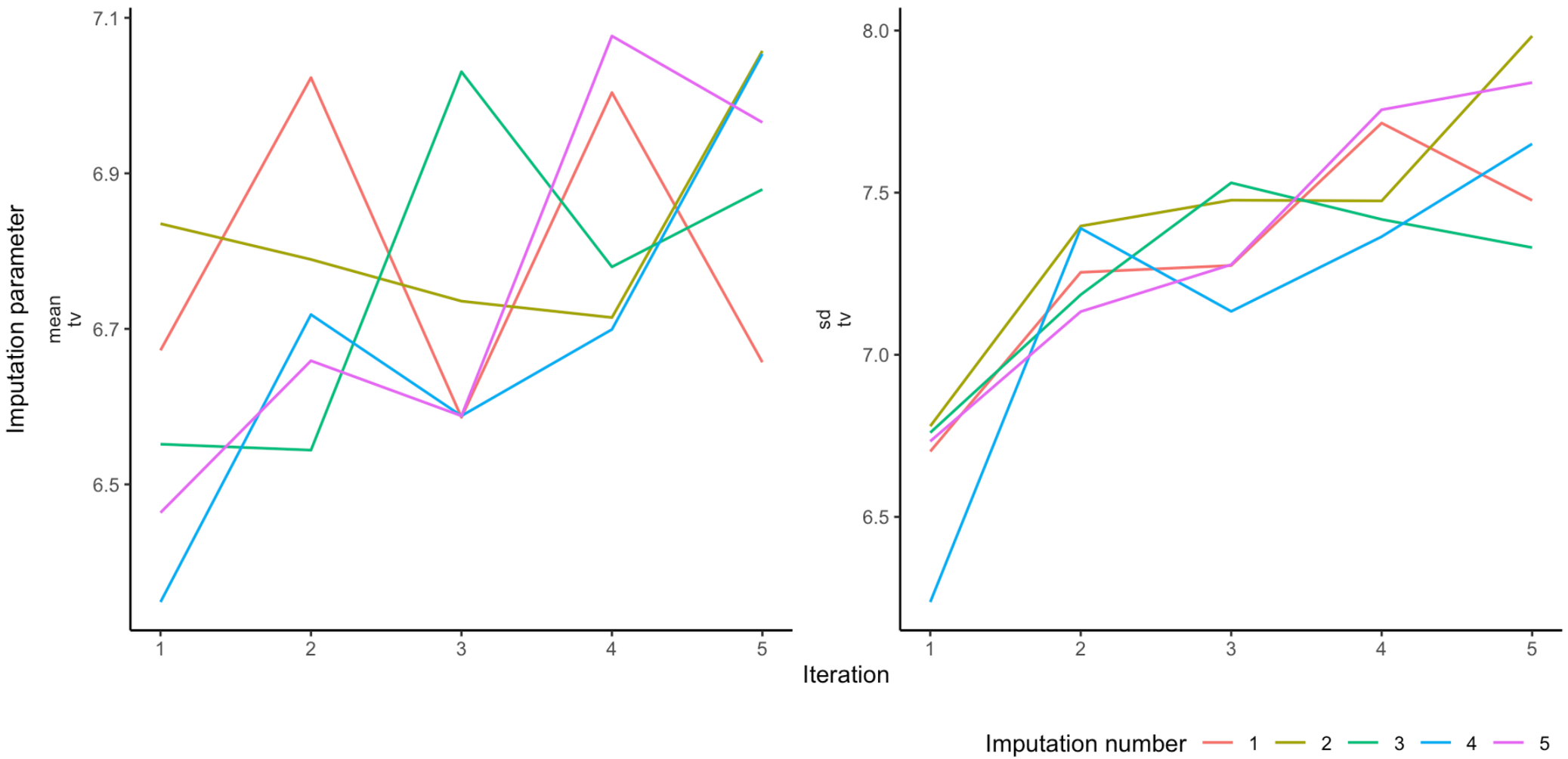
INSPECT THE IMPUTED DATA

```
1  ggmice::densityplot(imp)
```



INSPECT THE CONVERGENCE

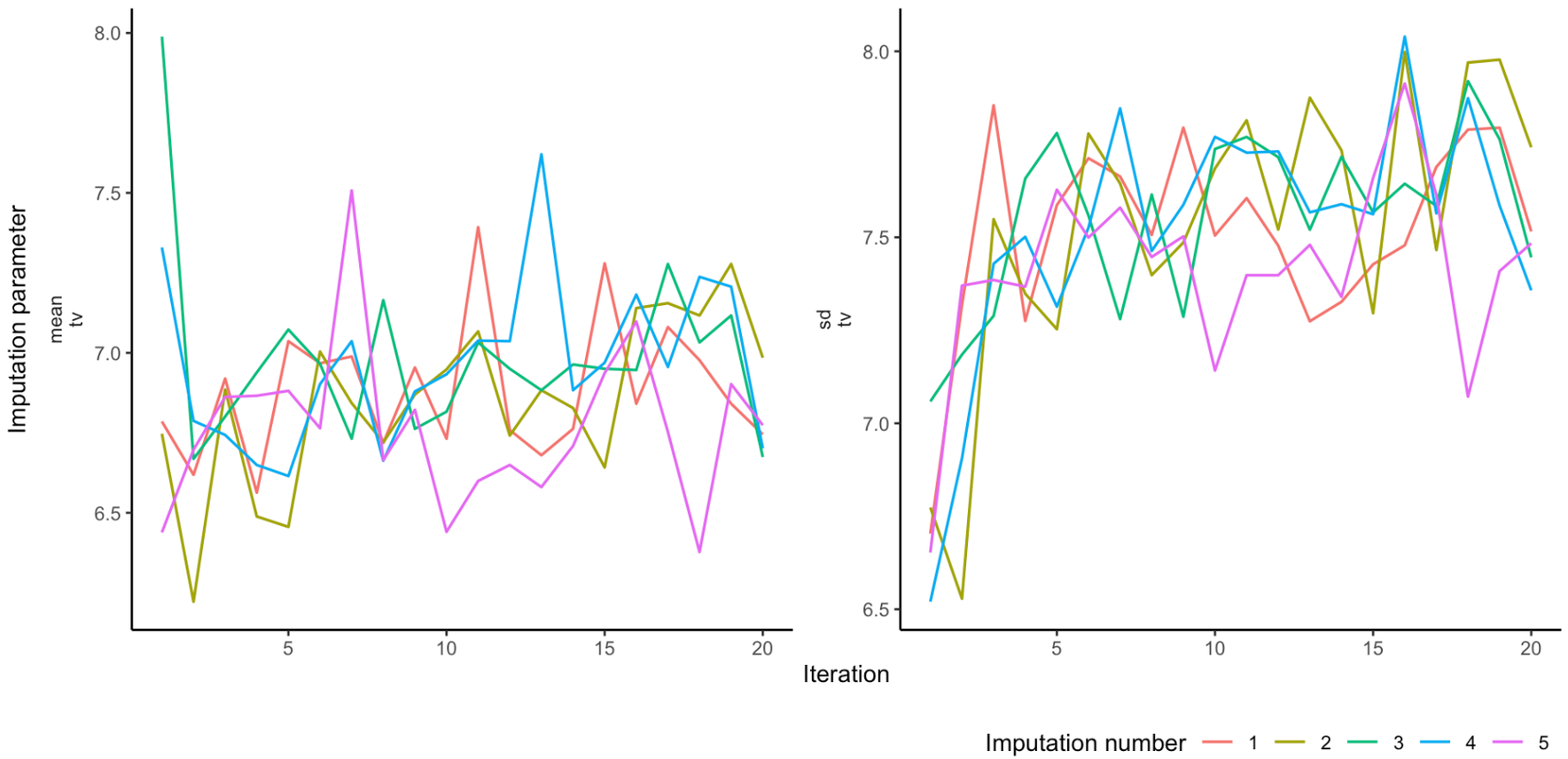
```
1 plot_trace(imp, vrb = "tv")
```



RUN MORE ITERATIONS

```
imp <- mice(boys,
            maxit = 20, # increase the number of iterations to 20
            printFlag = FALSE) # do not print the iteration history
```

```
plot_trace(imp, vrb = "tv")
```



CALCULATE ESTIMATES: CREATE A LIST OF IMPUTED DATA SETS

```
imp %>%
  complete("all") # creates a list of the imputed data sets
```

```
$`1`
        age   hgt     wgt   bmi   hc gen phb tv   reg
3     0.035  50.1   3.650 14.54 33.7  G1  P1  2 south
4     0.038  53.5   3.370 11.77 35.0  G5  P5 10 south
18    0.057  50.0   3.140 12.56 35.2  G3  P4  2 south
23    0.060  54.5   4.270 14.37 36.7  G2  P3  1 south
28    0.062  57.5   5.030 15.21 37.3  G2  P2  2 south
36    0.068  55.5   4.655 15.11 37.0  G1  P1  2 south
37    0.068  52.5   3.810 13.82 34.9  G3  P5  1 south
38    0.071  53.0   3.890 13.84 35.8  G1  P1  2  west
39    0.071  55.1   3.880 12.77 36.8  G2  P2  1  west
43    0.073  54.5   4.200 14.14 38.0  G1  P2  2  east
53    0.076  58.5   5.920 17.29 40.5  G1  P2  3  west
60    0.079  55.0   4.430 14.64 38.0  G1  P1  2  east
62    0.079  58.5   5.745 16.78 38.5  G1  P1  1  city
75    0.082  59.5   5.100 14.40 39.1  G1  P2  2  west
85    0.084  52.5   4.120 14.94 37.3  G1  P1  3 south
93    0.084  54.0   4.500 15.43 37.0  G1  P1  2 south
99    0.087  59.5   5.130 14.49 38.0  G1  P2  2  west
```

CALCULATE ESTIMATES: MEAN OF `tv`

```
imp %>%
  complete("all") %>% # list of imputed data sets
  map(~.x %$% mean(tv)) # calculate the means for every completed data set
```

```
$`1`
[1] 8.300802

$`2`
[1] 8.467914

$`3`
[1] 8.251337

$`4`
[1] 8.270053

$`5`
[1] 8.320856
```

POOLING THE MEANS

```
imp %>%
  complete("all") %>% # list of completed data sets
  map(~.x %$% mean(tv)) %>% # calculate the means for every completed data set
  reduce(`+`) / imp$m # divide the sum by the number of imputations `m`
```

```
[1] 8.322193
```

The mean in the observed data was:

```
boys %>% # start with the boys data
  select(tv) %>% # select only tv
  colMeans(na.rm = TRUE) # calculate the mean, excluding NAs
```

```
      tv
11.89381
```

POOLING A REGRESSION MODEL

```
imp %>%
  complete("all") %>% # list of completed data sets
  map(~.x %$% # start the map with data so we can use the %$% pipe
        lm(tv ~ age)) %>% # fit a linear model to each imputed data set
  pool() # mice internal pooling function for model objects
```

```
Class: mipo    m = 5
         term m   estimate         ubar             b           t dfcom       df
1 (Intercept) 5 -0.4557985 0.0757727041 2.385484e-03 0.078635285   746 579.3246
2         age 5  0.9584146 0.0005768796 3.004535e-05 0.000612934   746 436.0906
         riv     lambda        fmi
1 0.03777853 0.03640326 0.03971275
2 0.06249903 0.05882267 0.06310961
```

The `pool()` functions calculates the average over the estimates and pools the variances of the estimates according to Rubin's rules. Rubin's rules are a set of rules for combining estimates and variances from multiple imputed datasets to obtain a single estimate and variance that accounts for the uncertainty introduced by the missing data process.

POOLING A REGRESSION MODEL: SUMMARY

```
imp %>%
  complete("all") %>% # list of completed data sets
  map(~.x %$% # start the map with data so we can use the %$% pipe
        lm(tv ~ age)) %>% # fit a linear model to each imputed data set
  pool() %>% # mice internal pooling function for model objects
  summary(conf.int = TRUE) # print the summary of the pooled model
```

```
         term   estimate std.error statistic       df       p.value      2.5 %
1 (Intercept) -0.4557985 0.2804198 -1.625415 579.3246  1.046181e-01 -1.0065620
2         age  0.9584146 0.0247575 38.712084 436.0906 3.572171e-143  0.9097557
      97.5 %   conf.low  conf.high
1 0.09496491 -1.0065620 0.09496491
2 1.00707345  0.9097557 1.00707345
```


PRACTICAL