FILLING IN THE BLANKS: MULTIPLY IMPUTING MISSING DATA

R-Ladies RTP

Tuesday 9th November, 2021

Marissa Ashner, PhD Candidate
Sarah Lotspeich, PhD

University of North Carolina at Chapel Hill

Topics

- R packages.
- Example dataset.
- Definitions and describing missing data.
- Handling missing data: what not to do, what to do, and why.

Choose Your Own Adventure

Materials for this tutorial can be found on GitHub:
`https://github.com/sarahlotspeich/filling-in-blanks`.

You can find an outline of the code, as well as a completed "key" there.

R packages

Install/Load Packages

To follow along with our code as written, you will need to install and load the following `R packages:`

- `magrittr*: we will use pipes (%>%) instead of nested parentheses ((···))`
- `dplyr*: we will use mutate(), group_by(), and lag() for some data manipulation`

Install/Load Packages

To follow along with our code as written, you will need to install and load the following `R packages`:

- `magrittr*: we will use pipes (%>%) instead of nested parentheses ((···))`
- `dplyr*: we will use mutate(), group_by(), and lag() for some data manipulation`
- `geepack: we will use geese() to model correlated data`
- `MASS: we will use mvnorm() in multiple imputation`

`* You'll need these from the start.`

Example Dataset

Netflix Audience Ratings

Suppose Netflix is interested in predicting audience `rating` for their movies and tv series based on...

1. `votes`,
2. `runtime`,
3. `is_comedy`, and
4. `is_drama`.

The sample has 97 movies and 5405 episodes from tv series. The data were obtained from data.world.

Read in `movies` Data

The Netflix `movies` data can be read directly into `R` from the GitHub:

```
movies <- read.csv("https://raw.githubusercontent.com/sarahlotspeich/filling-in-blanks/main/data/MCAR/movies.csv")
head(movies)
        series_name rating votes runtime is_comedy is_drama rating_miss
1      An Easy Girl    5.5  1519      92         1        1          NA
2       The Week Of    5.1 17594     116         1        0         5.1
3    Murder Mystery    6.0 94014      97         1        0         6.0
4        Sextuplets    4.4  6784      97         1        0         4.4
5 The Kissing Booth    6.0 60140     105         1        0         6.0
6      #REALITYHIGH    5.2  5332      99         1        1         5.2
```

We have added missingness in the `rating` variable (the outcome of interest) - we call this variable `rating_miss`. We include `rating` just as a comparator (pretend we can't use it).

Modeling movies Ratings

- Interested in predicting the rating_miss of a movie based on votes, runtime, is_comedy, and is_drama
- The effects of these predictors on the rating are β from the linear model:

$$\begin{aligned} E[\texttt{RATING}|\cdot] = &\beta_0 + \beta_1 \log(\texttt{votes}) \\ &+ \beta_2 \texttt{runtime} + \beta_3 \texttt{is\_comedy} + \beta_4 \texttt{is\_drama} \end{aligned}$$

True (No Missingness) `movies` Model

```
summary(lm(formula = rating ~ log(votes) + runtime + is_comedy + is_drama,
           data = movies))

Call:
lm(formula = rating ~ log(votes) + runtime + is_comedy + is_drama,
    data = movies)

Residuals:
     Min       1Q   Median       3Q      Max
-2.15013 -0.63526 -0.07633  0.53881  2.54139

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  6.132787   0.503494  12.180  < 2e-16 ***
log(votes)   0.227211   0.058889   3.858 0.000212 ***
runtime     -0.018322   0.004515  -4.058 0.000104 ***
is_comedy   -0.327467   0.234864  -1.394 0.166589
is_drama     0.187928   0.228869   0.821 0.413701
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.8971 on 92 degrees of freedom
Multiple R-squared:  0.1967,    Adjusted R-squared:  0.1618
F-statistic: 5.633 on 4 and 92 DF,  p-value: 0.0004225
```

Read in `series` Data

The Netflix `series` data can be read directly into `R` from the GitHub:

```
series <- read.csv("https://raw.githubusercontent.com/sarahlotspeich/filling-in-blanks/main/data/MCAR/series.csv")
head(series)
     series_name series_num season episode rating votes runtime is_comedy is_drama rating_miss
1 13 Reasons Why          1      1       1    8.3  6952      54         0        1         8.3
2 13 Reasons Why          1      1       2    8.0  5803      52         0        1         8.0
3 13 Reasons Why          1      1       3    7.9  5453      57         0        1          NA
4 13 Reasons Why          1      1       4    8.1  5260      57         0        1         8.1
5 13 Reasons Why          1      1       5    8.2  5202      59         0        1         8.2
6 13 Reasons Why          1      1       6    8.0  5015      52         0        1         8.0
```

Modeling `series` Ratings

- Again, interested in predicting the `rating_miss` of a TV series episode based on `votes`, `runtime`, `is_comedy`, and `is_drama`
- However, we cannot use a simple linear model as we did in the movie dataset, since these data are correlated.
  - That is, we expect ratings within a TV series to be related closely to each other
- To capture that correlation, we will use a generalized estimating equation (GEE), and the model will be excluded here for simplicity

True (No Missingness) `series` Model

```
summary(geese(formula = rating ~ log(votes) + runtime + is_comedy + is_drama,
              data = series,
              id = series_num))

Call:
geese(formula = rating ~ log(votes) + runtime + is_comedy + is_drama,
    id = series_num, data = series)

Mean Model:
 Mean Link:                 identity
 Variance to Mean Relation: gaussian

 Coefficients:
              estimate      san.se        wald            p
(Intercept) 6.30605556 0.305648149 425.6690237 0.0000000000
log(votes)  0.12387523 0.034366026  12.9930251 0.0003126534
runtime     0.01096477 0.003011369  13.2577836 0.0002714504
is_comedy   0.10306925 0.143166353   0.5182941 0.4715702700
is_drama    0.44508262 0.176422884   6.3646047 0.0116419028

Scale Model:
 Scale Link:                identity

 Estimated Scale Parameters:
             estimate     san.se     wald p
(Intercept) 0.6942684 0.05755642 145.5014 0

Correlation Model:
 Correlation Structure:     independence

Returned Error Value:    0
Number of clusters:   279   Maximum cluster size: 1011
```

Describing and Defining Missing Data

Describing Missingness

It is always a good idea to describe the patterns of missingness:

- Which variables are missing?
- How many observations are missing them?
- Are they missing together on the same subjects?
- Who has missing data?

You should be able to answer these questions when presenting your final analysis.

There are many ways to do this.

Percent Missing (Usually Just Single Variable)

Since we only have missingness in `rating_miss`, we can summarize this with a simple **percent missing**.

```
# Missingness in movie ratings
mean(is.na(movies$rating_miss))
[1] 0.2061856

# Missingness in series ratings
mean(is.na(series$rating_miss))
[1] 0.1970006
```

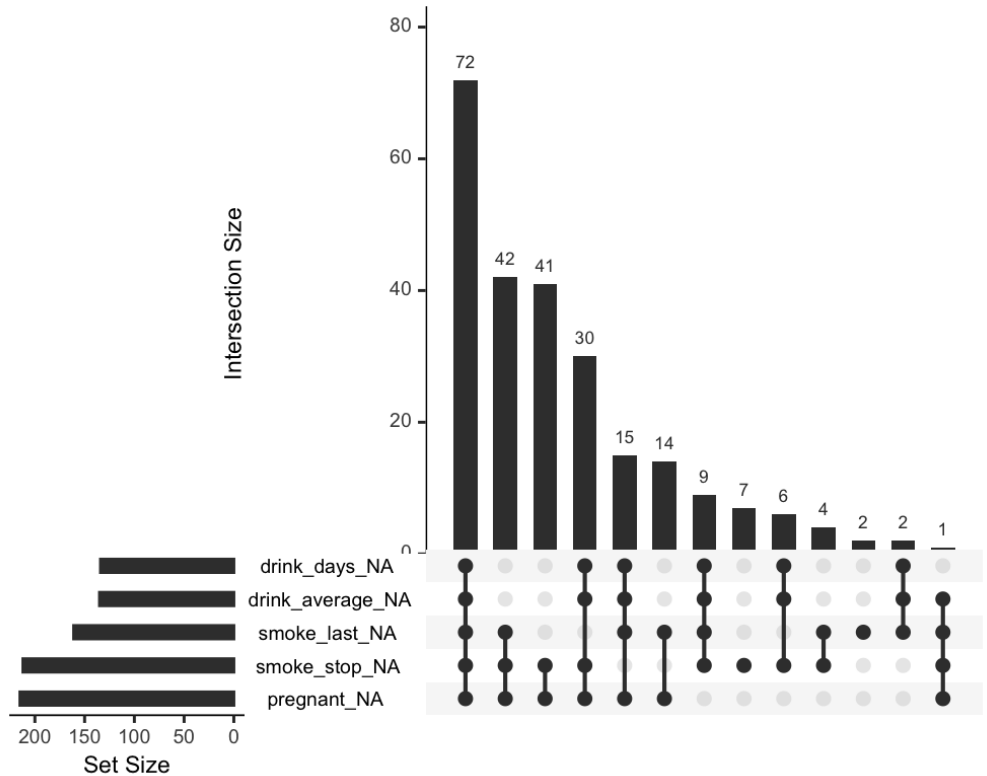
Could also make a bar graph with the `count` of missing values per variable.

Patterns in Missingness (Multiple Variables)

If we had missingness across two or more variables, we are interested in more than just the percent of observations with missing data. We are interested in the **patterns of missingness** across variables.

`naniar::gg_miss_upset()` is a nice ready-made option for this!

Source: `https://naniar.njtierney.com/articles/naniar-visualisation.html`

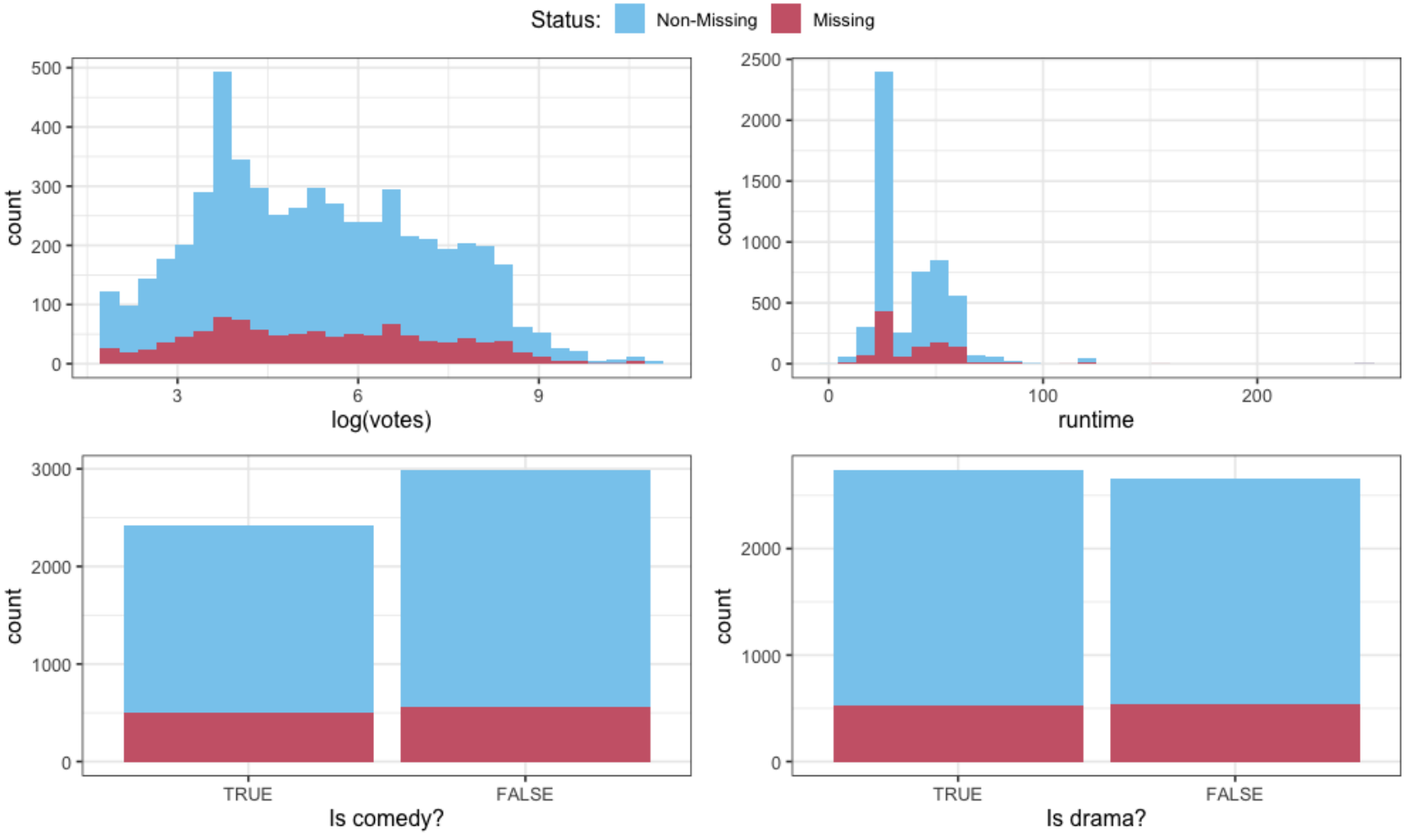


Types of Missing Data:

1. Missing Completely at Random (MCAR)

- Missingness does not depend on other observed or unobserved variables.
- Subjects with non-missing variables are a random subsample.
- A common assumption for many approaches (this will come back). But it's a strong one, and there will be trouble if you're wrong.
- Ex: Data manager accidentally deleted some responses (yikes).
- In our MCAR datasets, about 20% of the ratings were deleted

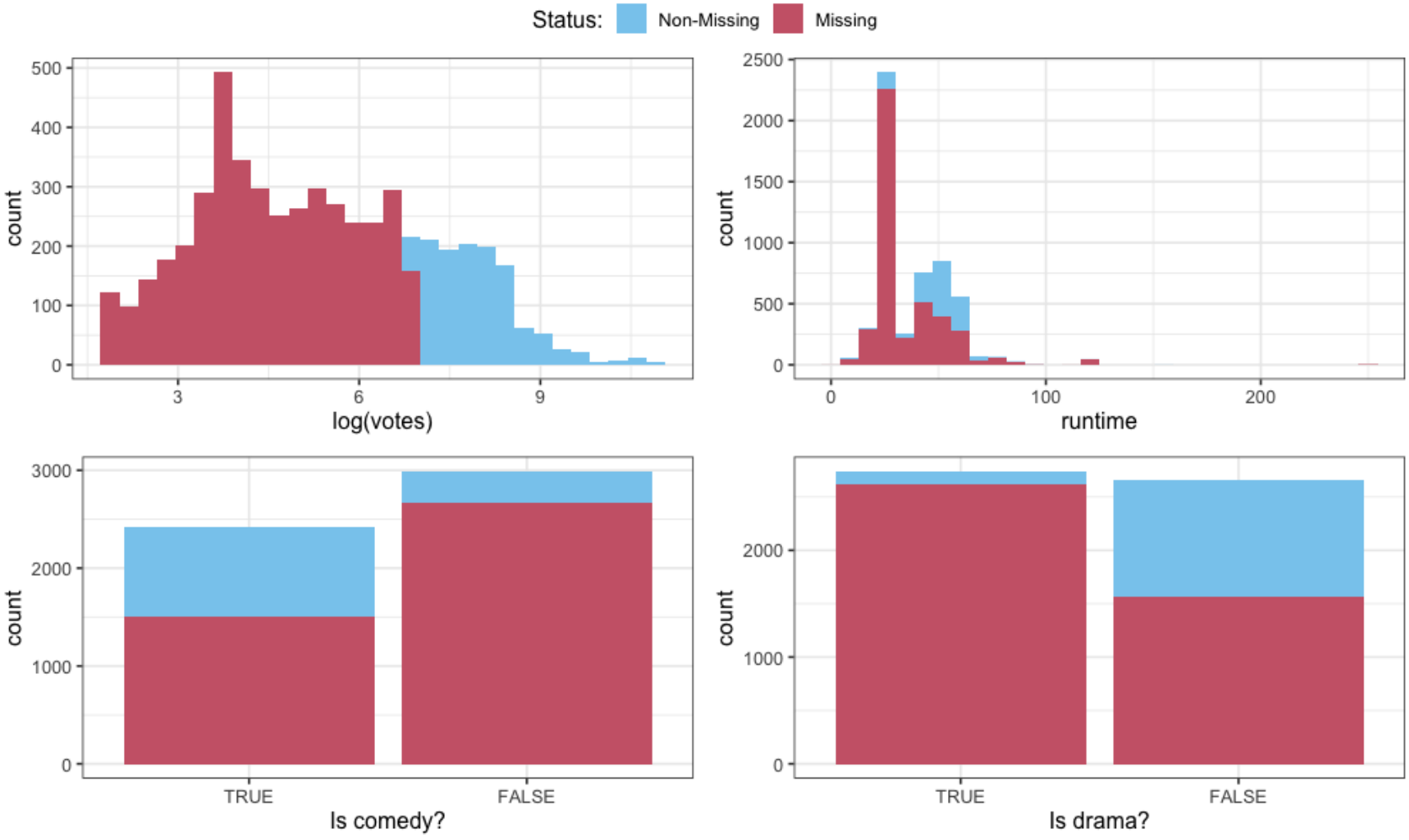
1. MCAR: Observed Variables in the `series` Dataset



2. Missing at Random (MAR)

- Missingness depends only on other variables that are fully observed.
- Good news: we have all the information we need to explain the missingness.
- Ex: Movies and TV series episodes with small amounts of votes (i.e. < 1000) may have their ratings unavailable in order to retain anonymity. Since the number of `votes` is fully observed, this is MAR.
- In our data, $\sim 18\%$ of `movies` and $\sim 77\%$ of `series` episodes had a small amount of votes.

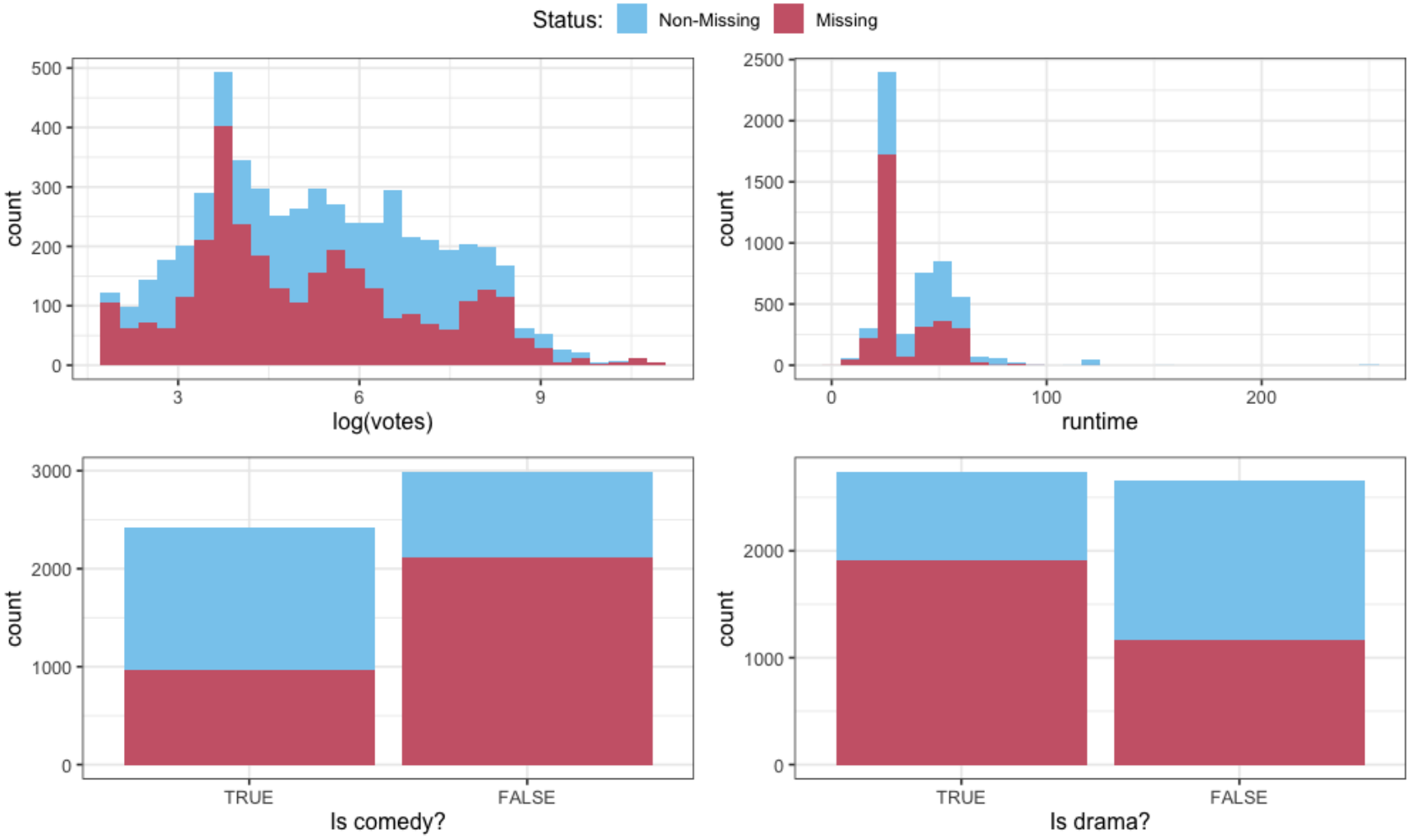
2. MAR: Observed Variables in the `series` Dataset



3. Missing Not at Random (MNAR) or Non-Ignorable Missingness (NIM)

- Missingness depends on other variables that are unobserved.
- Bad news: we cannot obtain the information needed to explain this missingness.
- Ex: Netflix may not have started to collect ratings until 2017, so the missing `rating_miss` depend on the release year. If you don't have the release year available, this is MNAR (*we do understand the release year for any of these could be found on the internet, but let's pretend it's lost for learning purposes*).
- In our data, $\sim 10\%$ of `movies` and $\sim 57\%$ of `series` episodes were released prior to 2017.

3. MNAR: Observed variables in the series dataset



Summary: Types of Missing Data

Consider a study with outcome Y and covariates X_1, X_2.

There is missingness in Y:

- if the occurrence of missingness does not depend on Y, X_1, or X_2 it is missing completely at random (MCAR),
- if the occurrence of missingness depends on X_1 and/ or X_2 it is missing at random (MAR),
- if the occurrence of missingness depends on Y it is missing not at random (MNAR).

Handling Missing Data (the "Easy" Way Out)

Simple Approaches

For independent or dependent data:

1. Complete-case analysis
2. Single imputation (mean, median, etc.)

Simple Approaches

For independent or dependent data:

1. Complete-case analysis
2. Single imputation (mean, median, etc.)

Specific to dependent data:

3. Within-subject single imputation (mean, median, etc.)
4. Carry-forward imputation

Simple Approaches

For independent or dependent data:

1. Complete-case analysis
2. Single imputation (mean, median, etc.)

Specific to dependent data:

3. Within-subject single imputation (mean, median, etc.)
4. Carry-forward imputation

Each of these approaches is used to create a complete dataset on which the model of interest is fit once. Approaches 2-4 ignore uncertainty around imputed values, causing standard errors to be underestimated. Approach 1 has its own problems, though.

Simple Approaches

1. Complete-case analysis
   - Fit the model and exclude subjects with missing data.
   - Can be biased (depends on the type of missingness).
   - Even if it isn't biased, you're throwing people (and thus information) away.
   - Smaller sample size $\rightarrow$ larger standard errors/ loss of efficiency.
   - **Note:** This is the default in `R` for nearly all modeling functions!

Visualizing Complete-Case Analysis for `movies`

series_name	rating_miss	votes	runtime	is_comedy	is_drama
An Easy Girl	5.5	1519	92	1	1
The Week Of	5.1	17594	116	1	0
Murder Mystery	6	94014	97	1	0
Sextuplets	4.4	6784	97	1	0
The Kissing Booth	6	60140	105	1	0
#REALITYHIGH	×	5332	99	1	1
Dragons: Rescue Riders: ...	6.7	56	46	1	0
Jeff Dunham: Relative Disaster	×	1082	70	1	0

Visualizing Complete-Case Analysis for `movies`

series_name	rating_miss	votes	runtime	is_comedy	is_drama
An Easy Girl	5.5	1519	92	1	1
The Week Of	5.1	17594	116	1	0
Murder Mystery	6	94014	97	1	0
Sextuplets	4.4	6784	97	1	0
The Kissing Booth	6	60140	105	1	0
–	–	–	–	–	–
Dragons: Rescue Riders: ...	6.7	56	46	1	0
–	–	–	–	–	–

Comparing the Simple Approaches (Independent Data)

0. True Parameter Estimates (`movies`)
   - i.e., if no data were missing and we ran the regression model

R Code:

```
# Fit the *true* (i.e., no missing data) movie ratings model using lm()
summary(lm(formula = rating ~ log(votes) + runtime + is_comedy + is_drama,
           data = movies))
```

Results:

	Truth	
	Est	SE
(Intercept)	6.133	0.503
log(votes)	0.227	0.059
runtime	-0.018	0.005
is_comedy	-0.327	0.235
is_drama	0.188	0.229

Comparing the Simple Approaches (Independent Data)

1. Complete-case analysis
   - Remember: MCAR - 20% missing, MAR - 18% missing, MNAR - 10% missing (these subjects are all being deleted).

R Code:

```
# Fit the complete case model for movie ratings using lm()
summary(lm(formula = rating_miss ~ log(votes) + runtime + is_comedy + is_drama,
           data = movies))
```

Results:

	Truth		Complete Case Analysis					
			MCAR		MAR		MNAR	
	Est	SE	Est	SE	Est	SE	Est	SE
(Intercept)	6.133	0.503	6.418	0.556	6.631	0.722	5.851	0.531
log(votes)	0.227	0.059	0.174	0.069	0.173	0.080	0.242	0.063
runtime	-0.018	0.005	-0.016	0.005	-0.017	0.005	-0.017	0.005
is_comedy	-0.327	0.235	-0.331	0.260	-0.453	0.248	-0.253	0.248
is_drama	0.188	0.229	0.139	0.267	0.087	0.232	0.232	0.243

Simple Approaches

2. Single imputation
   - Replace missing values with summary measure (e.g. mean, median, mode) of non-missing values from other subjects.
   - Ignores relationship between missingness and other variables.
   - Can take it a step farther and do single conditional mean imputation using other non-missing information.

Visualizing Single Mean Imputation for `movies`

series_name	rating_miss	votes	runtime	is_comedy	is_drama
An Easy Girl	5.5	1519	92	1	1
The Week Of	5.1	17594	116	1	0
Murder Mystery	6	94014	97	1	0
Sextuplets	4.4	6784	97	1	0
The Kissing Booth	6	60140	105	1	0
#REALITYHIGH	×	5332	99	1	1
Dragons: Rescue Riders: ...	6.7	56	46	1	0
Jeff Dunham: Relative Disaster	×	1082	70	1	0

Visualizing Single Mean Imputation for `movies`

series_name	rating_miss	votes	runtime	is_comedy	is_drama
An Easy Girl	5.5	1519	92	1	1
The Week Of	5.1	17594	116	1	0
Murder Mystery	6	94014	97	1	0
Sextuplets	4.4	6784	97	1	0
The Kissing Booth	6	60140	105	1	0
#REALITYHIGH	$\overline{\texttt{RATING}}$	5332	99	1	1
Dragons: Rescue Riders: ...	6.7	56	46	1	0
Jeff Dunham: Relative Disaster	$\overline{\texttt{RATING}}$	1082	70	1	0

where $\overline{\texttt{RATING}}$ is the mean of all non-missing movie ratings.

Visualizing Single Mean Imputation for `movies`

series_name	rating_miss	votes	runtime	is_comedy	is_drama
An Easy Girl	5.5	1519	92	1	1
The Week Of	5.1	17594	116	1	0
Murder Mystery	6	94014	97	1	0
Sextuplets	4.4	6784	97	1	0
The Kissing Booth	6	60140	105	1	0
#REALITYHIGH	$\overline{\texttt{RATING}}$	5332	99	1	1
Dragons: Rescue Riders: ...	6.7	56	46	1	0
Jeff Dunham: Relative Disaster	$\overline{\texttt{RATING}}$	1082	70	1	0

where $\overline{\texttt{RATING}}$ is the mean of all non-missing movie ratings.

R Code:

```
# Replace missing rating_miss values with the mean of the non-missing values
movies %>%
  dplyr::mutate(rating_imp = ifelse(is.na(rating_miss), mean(rating_miss, na.rm = TRUE), rating_miss)) -> movies
head(movies)
        series_name rating votes runtime is_comedy is_drama rating_miss rating_imp
1      An Easy Girl    5.5  1519      92         1        1          NA   6.180519
2       The Week Of    5.1 17594     116         1        0         5.1   5.100000
3    Murder Mystery    6.0 94014      97         1        0         6.0   6.000000
4        Sextuplets    4.4  6784      97         1        0         4.4   4.400000
5 The Kissing Booth    6.0 60140     105         1        0         6.0   6.000000
6      #REALITYHIGH    5.2  5332      99         1        1         5.2   5.200000
```

2. Single mean imputation
   - Remember: MCAR - 20% missing, MAR - 18% missing, MNAR - 10% missing (these subjects assumed to be "average")

R Code:

```
# Fit the single imputation model (i.e., using your singly imputed rating value from the previous chunk)
summary(lm(formula = rating_imp ~ log(votes) + runtime + is_comedy + is_drama,
        data = movies))
```

Results:

	Truth		Complete Case Analysis						Single Mean Imputation					
			MCAR		MAR		MNAR		MCAR		MAR		MNAR	
	Est	SE	Est	SE	Est	SE	Est	SE	Est	SE	Est	SE	Est	SE
(Intercept)	6.133	0.503	6.418	0.556	6.631	0.722	5.851	0.531	6.298	0.457	6.739	0.457	5.745	0.484
log(votes)	0.227	0.059	0.174	0.069	0.173	0.080	0.242	0.063	0.119	0.053	0.140	0.053	0.205	0.057
runtime	-0.018	0.005	-0.016	0.005	-0.017	0.005	-0.017	0.005	-0.011	0.004	-0.015	0.004	-0.013	0.004
is_comedy	-0.327	0.235	-0.331	0.260	-0.453	0.248	-0.253	0.248	-0.219	0.213	-0.436	0.213	-0.206	0.226
is_drama	0.188	0.229	0.139	0.267	0.087	0.232	0.232	0.243	0.059	0.208	0.075	0.208	0.156	0.220

Simple Approaches

3. Within-subject single imputation
   - Replace missing time points with some summary measure (e.g. mean, median, mode) of the non-missing time points from same subject.
   - Like regular single imputation, this ignores relationship between missingness and other variables.
   - Ignores temporality in these dependent data.

Visualizing Within-Series Single Mean Imputation

season	episode	rating_miss	votes	runtime	is_comedy	is_drama
1	1	×	58	58	0	0
1	2	8	38	58	0	0
1	3	8.1	33	58	0	0
1	4	8	32	52	0	0
1	5	×	33	58	0	0
1	6	8.2	30	58	0	0
2	1	8.3	27	59	0	0
2	2	8.5	21	58	0	0
2	3	8.5	21	58	0	0

Visualizing Within-Series Single Mean Imputation

season	episode	rating_miss	votes	runtime	is_comedy	is_drama
1	1	$\overline{\texttt{RATING}}_{GBBS}$	58	58	0	0
1	2	8	38	58	0	0
1	3	8.1	33	58	0	0
1	4	8	32	52	0	0
1	5	$\overline{\texttt{RATING}}_{GBBS}$	33	58	0	0
1	6	8.2	30	58	0	0
2	1	8.3	27	59	0	0
2	2	8.5	21	58	0	0
2	3	8.5	21	58	0	0

where $\overline{\texttt{RATING}}_{GBBS}$ is the mean of all non-missing episodes ratings for our favorite series, the Great British Baking Show (GBBS).

Visualizing Within-Series Single Mean Imputation

season	episode	rating_miss	votes	runtime	is_comedy	is_drama
1	1	$\overline{\text{RATING}}_{GBBS}$	58	58	0	0
1	2	8	38	58	0	0
1	3	8.1	33	58	0	0
1	4	8	32	52	0	0
1	5	$\overline{\text{RATING}}_{GBBS}$	33	58	0	0
1	6	8.2	30	58	0	0
2	1	8.3	27	59	0	0
2	2	8.5	21	58	0	0
2	3	8.5	21	58	0	0

where $\overline{\text{RATING}}_{GBBS}$ is the mean of all non-missing episodes ratings for our favorite series, the Great British Baking Show (GBBS).

R Code:

```
# Replace missing rating_miss values with the mean of the non-missing values *for the same series*
series %>%
  dplyr::group_by(series_name) %>%
  dplyr::mutate(rating_imp = ifelse(is.na(rating_miss), mean(rating_miss, na.rm = TRUE), rating_miss)) -> series
series %>%
  dplyr::filter(series_name == "The Great British Baking Show") %>%
  head()
    series_name                  series_num season episode rating votes runtime is_comedy is_drama rating_miss rating_imp
1 The Great British Baking Show        217      1       1    8.2    58      58         0         0        NA         8.21
2 The Great British Baking Show        217      1       2    8      38      58         0         0         8         8
3 The Great British Baking Show        217      1       3    8.1    33      58         0         0         8.1       8.1
4 The Great British Baking Show        217      1       4    8      32      52         0         0         8         8
5 The Great British Baking Show        217      1       5    8      33      58         0         0        NA         8.21
6 The Great British Baking Show        217      1       6    8.2    30      58         0         0         8.2       8.2
```

Comparing the Simple Approaches (Dependent Data)

0. True Parameter Estimates (`series`)
   - i.e., if no data were missing and we ran the regression model

R Code:

```
# Fit the *true* (i.e., no missing data) series episode ratings model using geese()
summary(geese(formula = rating ~ log(votes) + runtime + is_comedy + is_drama,
              data = series,
              id = series_num))
```

Results:

	Truth	
	Est	**SE**
`(Intercept)`	6.306	0.305
`log(votes)`	0.124	0.034
`runtime`	0.011	0.003
`is_comedy`	0.103	0.143
`is_drama`	0.444	0.176

Comparing the Simple Approaches (Dependent Data)

3. Within-series single imputation
   - Remember: MCAR - 20% missing, MAR - 77% missing, MNAR - 57% missing (these episodes assumed to be "average" for the show overall)

R Code:

```
# Fit the single imputation model (i.e., using your singly imputed rating value from the previous chunk)
summary(geepack::geese(formula = rating_imp ~ log(votes) + runtime + is_comedy + is_drama,
               data = series,
               id = series_num))
```

Results:

	Truth		Single Mean Imputation within Series					
			MCAR		MAR		MNAR	
	Est	SE	Est	SE	Est	SE	Est	SE
(Intercept)	6.306	0.305	6.309	0.307	8.525	0.363	6.497	0.278
log(votes)	0.124	0.034	0.124	0.034	-0.047	0.056	0.071	0.059
runtime	0.011	0.003	0.011	0.003	0.005	0.005	0.010	0.003
is_comedy	0.103	0.143	0.089	0.144	0.029	0.160	0.345	0.168
is_drama	0.444	0.176	0.437	0.179	-0.170	0.229	0.372	0.150

Simple approaches

4. Last observation carried forward (LOCF)
   - Carry forward value for that subject at their last visit.
   - This assumes that there was no change in the variable between these visits. Is that reasonable?
   - For intermittent missingness, value at the next non-missing visit is ignored.
   - Standard errors ignoring uncertainty in these observations will be under-estimated.

Visualizing Last Observation Carried Forward Imputation for `series`

SEASON	EPSIODE	RATING	VOTES	RUNTIME	IS_COMEDY	IS_DRAMA
1	2	8	38	58	0	0
1	3	8.1	33	58	0	0
1	4	8	32	52	0	0
1	5	×	33	58	0	0
1	6	8.2	30	58	0	0
2	1	8.3	27	59	0	0
2	2	8.5	21	58	0	0
2	3	8.5	21	58	0	0

Visualizing Last Observation Carried Forward Imputation for `series`

SEASON	EPSIODE	RATING	VOTES	RUNTIME	IS_COMEDY	IS_DRAMA
1	2	8	38	58	0	0
1	3	8.1	33	58	0	0
1	4	8	32	52	0	0
1	5	8	33	58	0	0
1	6	8.2	30	58	0	0
2	1	8.3	27	59	0	0
2	2	8.5	21	58	0	0
2	3	8.5	21	58	0	0

Visualizing Last Observation Carried Forward Imputation for `series`

SEASON	EPSIODE	RATING	VOTES	RUNTIME	IS_COMEDY	IS_DRAMA
1	2	8	38	58	0	0
1	3	8.1	33	58	0	0
1	4	8	32	52	0	0
1	5	8	33	58	0	0
1	6	8.2	30	58	0	0
2	1	8.3	27	59	0	0
2	2	8.5	21	58	0	0
2	3	8.5	21	58	0	0

R Code:

```
# Replace missing rating_miss values with the preceding non-missing value *for the same series*
series %>%
  dplyr::group_by(series_num) %>%
  dplyr::mutate(rating_imp = ifelse(is.na(rating_miss),
                                    dplyr::lag(x = rating_miss, n = 1, default = NA),
                                    rating_miss)) -> series
# Check your imputations for the Great British Baking Show
series %>%
  dplyr::filter(series_name == "The Great British Baking Show") %>%
  head()
  series_name                   series_num season episode rating votes runtime is_comedy is_drama rating_miss rating_imp
1 The Great British Baking Show        217      1       1    8.2    58      58         0        0        NA         NA
2 The Great British Baking Show        217      1       2    8      38      58         0        0         8          8
3 The Great British Baking Show        217      1       3    8.1    33      58         0        0         8.1        8.1
4 The Great British Baking Show        217      1       4    8      32      52         0        0         8          8
5 The Great British Baking Show        217      1       5    8      33      58         0        0        NA          8
6 The Great British Baking Show        217      1       6    8.2    30      58         0        0         8.2        8.2
```

Simple approaches

4. Last observation carried forward (LOCF)
   - Remember: MCAR - 20% missing, MAR - 77% missing, MNAR - 57% missing (these episodes are assumed to have the same rating as the one before them)

R Code:

```
# Fit the single imputation model (i.e., using your singly imputed rating value
## from the previous chunk)
summary(geepack::geese(formula = rating_imp ~ log(votes) + runtime + is_comedy + is_drama,
               data = series,
               id = series_num))
```

Results:

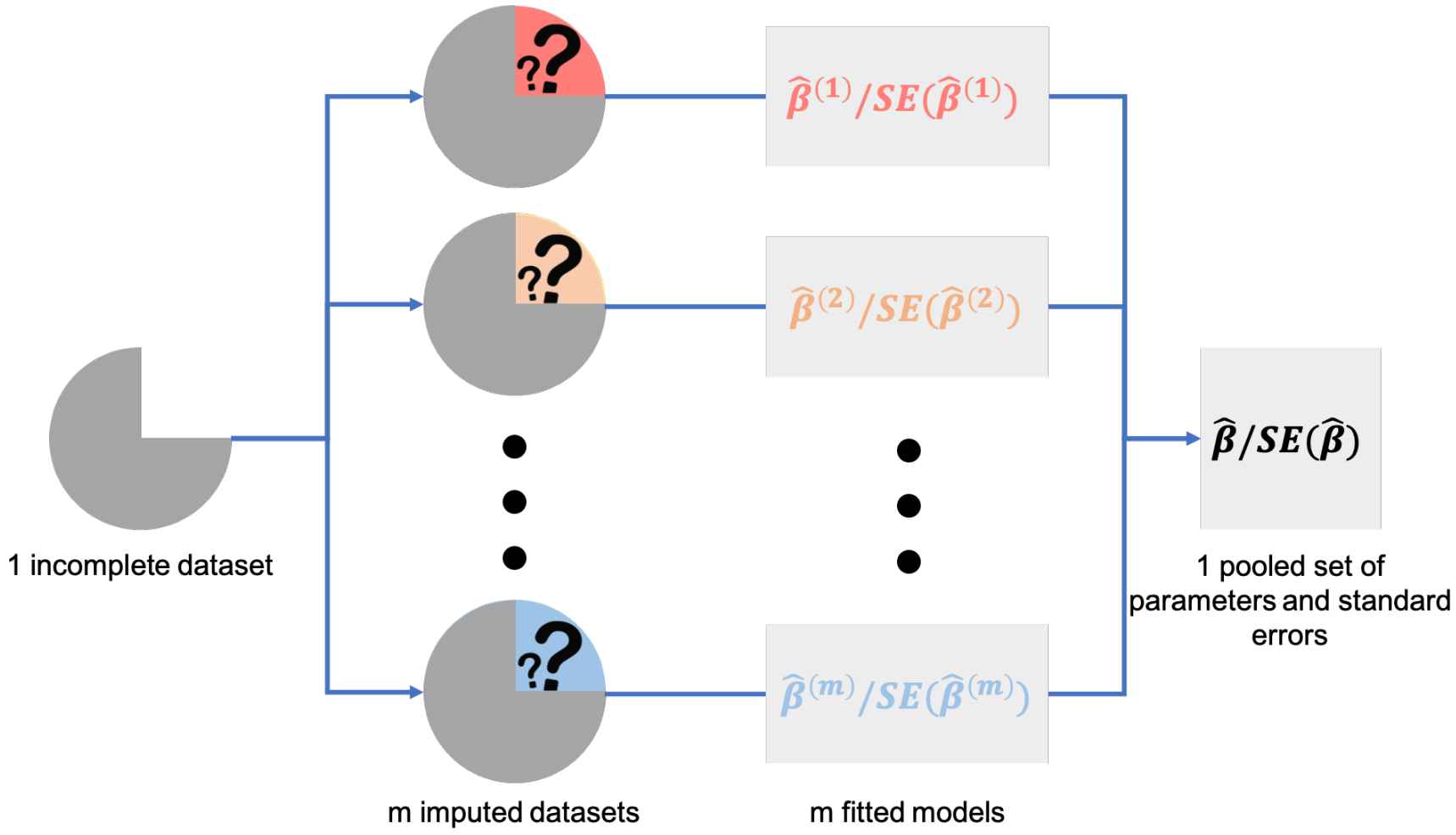
	Truth		Single Mean Imputation within Series						Last Observation Carried Forward					
			MCAR		MAR		MNAR		MCAR		MAR		MNAR	
	Est	SE	Est	SE	Est	SE	Est	SE	Est	SE	Est	SE	Est	SE
(Intercept)	6.306	0.305	6.309	0.307	8.525	0.363	6.497	0.278	6.309	0.307	5.603	0.984	6.373	0.258
log(votes)	0.124	0.034	0.124	0.034	-0.047	0.056	0.071	0.059	0.124	0.034	0.249	0.120	0.087	0.034
runtime	0.011	0.003	0.011	0.003	0.005	0.005	0.010	0.003	0.011	0.003	0.003	0.004	0.012	0.003
is_comedy	0.103	0.143	0.089	0.144	0.029	0.160	0.345	0.168	0.089	0.144	0.041	0.111	0.413	0.127
is_drama	0.444	0.176	0.437	0.179	-0.170	0.229	0.372	0.150	0.437	0.179	0.592	0.188	0.370	0.133

Handling Missing Data (the Way We Want You to from Now on)

Multiple imputation (MI)

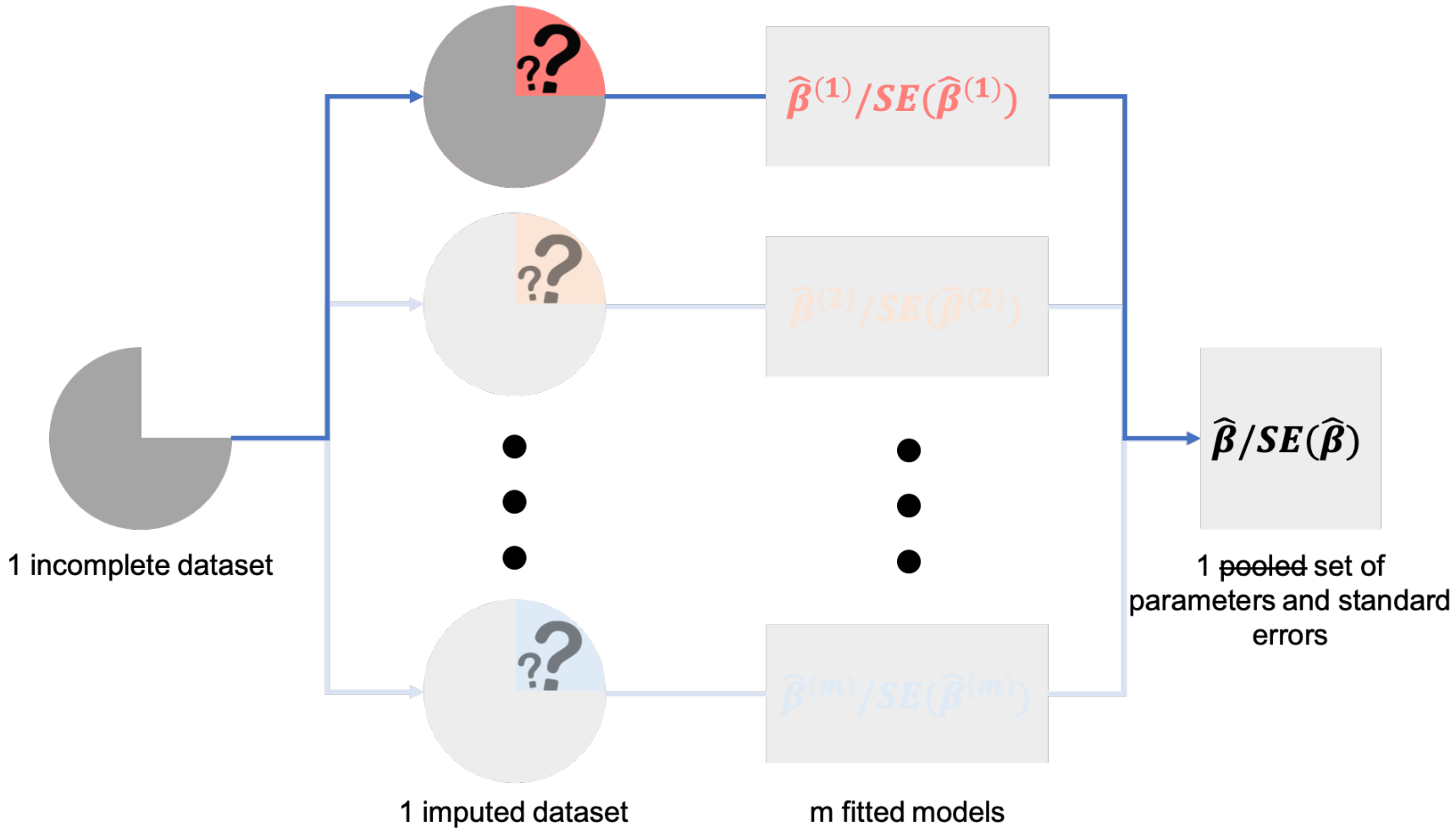
- Simple imputation approaches ignore uncertainty and treat imputed values with as much trust as observed ones.
- They create one imputed/ complete dataset and fit the model.
- Standard errors are underestimated, so p-values are too small/confidence intervals incorrect.
- Advantages of multiple imputation:
  1. Preserves relationships between variables.
  2. Standard errors are correct, so inference is valid.

Multiple imputation: Overview of the steps



Adapted from Figure 1.6. in Buuren, S. (2012). Flexible imputation of missing data. Boca Raton, FL: CRC Press.

Multiple imputation versus single



Adapted from Figure 1.6. in Buuren, S. . (2012). Flexible imputation of missing data. Boca Raton, FL: CRC Press.

Multiple imputation: Steps

Multiple imputation is broken down into three steps:

1. imputation
2. analysis
3. pooling

By repeating steps 1 - 2 m times (for $m > 1$), the resulting pooled result from step 3 correctly captures the uncertainty in the imputed values and yields valid estimates and standard errors.

Step 1: Imputation

Often we can use the subjects with complete observation to fit an **imputation model**.

- Predict the missing values from non-missing information with a regression model.
- Preserves relationships between the variables rather than using an unconditional approach.
- Can include additional information that can better predict the missing variable (even it isn't of interest in your analysis model). Sometimes these are called auxiliary variables.

Step 1: Imputation (`movies` Dataset)

Use subjects without missing data to fit the **imputation model**:

$$E[\texttt{rating\_miss}|\cdot] = \gamma_0 + \gamma_1 \log(\texttt{votes}) + \gamma_2 \texttt{runtime} + \gamma_3 \texttt{is\_comedy} + \gamma_4 \texttt{is\_drama}$$

- Since `rating_miss` is continuous, this is an ordinary linear model.
- Fit the imputation model and obtain estimates $\hat{\gamma}$ and $Cov(\hat{\gamma})$.

R Code:

```
# Fit the imputation model for movies rating (complete-case)
imp_mod <- lm(formula = rating_miss ~ log(votes) + runtime + is_comedy + is_drama,
              data = movies)
# Save the imputation model coefficients
imp_coeff <- imp_mod$coefficients
# Save the imputation model covariance matrix
imp_cov <- vcov(imp_mod)
```

Step 1: Imputation

We need to add randomness to this model to obtain
m different sets of reasonable imputed values for each missing one.

- Draw a random vector of parameters $\hat{\gamma}^{(k)}$ from a multivariate Normal distribution with mean $\hat{\gamma}$ and covariance $Cov(\hat{\gamma})$ (from the imputation model).

Step 1: Imputation

We need to add randomness to this model to obtain
m different sets of reasonable imputed values for each missing one.

- Draw a random vector of parameters $\hat{\gamma}^{(k)}$ from a multivariate Normal distribution with mean $\hat{\gamma}$ and covariance $Cov(\hat{\gamma})$ (from the imputation model).

R Code:

```
# Draw coefficients from the multivariate normal distribution based on `imp_mod`
coeff_m <- mvrnorm(n = 1, mu = imp_coeff, Sigma = imp_cov)
```

Step 1: Imputation

We need to add randomness to this model to obtain
m different sets of reasonable imputed values for each missing one.

- Draw a random vector of parameters $\hat{\gamma}^{(k)}$ from a multivariate Normal distribution with mean $\hat{\gamma}$ and covariance $Cov(\hat{\gamma})$ (from the imputation model).

R Code:

```
# Draw coefficients from the multivariate normal distribution based on `imp_mod`
coeff_m <- mvrnorm(n = 1, mu = imp_coeff, Sigma = imp_cov)
```

- Use $\hat{\gamma}^{(k)}$ to predict missing values (let's call them $\hat{p}^{(k)}$).

Step 1: Imputation

We need to add randomness to this model to obtain m different sets of reasonable imputed values for each missing one.

- Draw a random vector of parameters $\hat{\gamma}^{(k)}$ from a multivariate Normal distribution with mean $\hat{\gamma}$ and covariance $Cov(\hat{\gamma})$ (from the imputation model).

R Code:

```
# Draw coefficients from the multivariate normal distribution based on `imp_mod`
coeff_m <- mvrnorm(n = 1, mu = imp_coeff, Sigma = imp_cov)
```

- Use $\hat{\gamma}^{(k)}$ to predict missing values (let's call them $\hat{p}^{(k)}$).

R Code:

```
# Use the drawn coefficients to calculate imputed values
imp_m <- coeff_m[1] + coeff_m[2] * log(movies$votes) + coeff_m[3] * movies$runtime + coeff_m[4] * movies$is_comedy
            + coeff_m[5] * movies$is_drama

# Replace with the non-missing ratings
imp_m[!is.na(movies$rating_miss)] <- movies$rating_miss[!is.na(movies$rating_miss)]
```

Step 1: Imputation (`movies` Dataset)

Each $\left(\hat{\gamma}^{(k)}, \hat{p}^{(k)}\right)$ produces an imputed dataset. In our example:

series_name	rating_miss	votes	runtime	is_comedy	is_drama
An Easy Girl	5.5	1519	92	1	1
The Week Of	5.1	17594	116	1	0
Murder Mystery	6	94014	97	1	0
Sextuplets	4.4	6784	97	1	0
The Kissing Booth	6	60140	105	1	0
#REALITYHIGH	×	5332	99	1	1
Dragons: Rescue Riders: ...	6.7	56	46	1	0
Jeff Dunham: Relative Disaster	×	1082	70	1	0

Step 1: Imputation (movies Dataset)

Each $\left(\hat{\gamma}^{(k)}, \hat{p}^{(k)}\right)$ produces an imputed dataset. In our example:

series_name	rating_miss	votes	runtime	is_comedy	is_drama
An Easy Girl	5.5	1519	92	1	1
The Week Of	5.1	17594	116	1	0
Murder Mystery	6	94014	97	1	0
Sextuplets	4.4	6784	97	1	0
The Kissing Booth	6	60140	105	1	0
#REALITYHIGH	$\widehat{\texttt{RATING}}^{(k)}_{RH}$	5332	99	1	1
Dragons: Rescue Riders: ...	6.7	56	46	1	0
Jeff Dunham: Relative Disaster	$\widehat{\texttt{RATING}}^{(k)}_{JD}$	1082	70	1	0

where $\widehat{\texttt{rating\_miss}}_i^{(k)} =$
$\hat{\gamma}_0^{(k)} + \hat{\gamma}_1^{(k)} \log(\texttt{votes}_i) + \hat{\gamma}_2^{(k)} \texttt{runtime}_i + \hat{\gamma}_3^{(k)} \texttt{is\_comedy}_i + \hat{\gamma}_4^{(k)} \texttt{is\_drama}_i.$

Step 2: Analysis

The analysis model of interest is fit to each of the m imputed datasets.

- Save estimates $\hat{\beta}^{(k)}$ and $Var(\hat{\beta}^{(k)})$.

Step 2: Analysis

The analysis model of interest is fit to each of the m imputed datasets.

- Save estimates $\hat{\beta}^{(k)}$ and $Var(\hat{\beta}^{(k)})$.

R Code:

```
## Fit the analysis model to the imputed dataset
fit_m <- lm(formula = imp_m ~ log(votes) + runtime + is_comedy + is_drama,
            data = movies)

## Save the analysis model coefficients
fit_m_coeff <- fit_m$coefficients

## Save the analysis  model covariance matrix
fit_m_cov <- vcov(fit_m)
```

Iterating Between Steps 1–2

R Code:

```
# We use 20 imputations
m <- 20

# Create a list to store model fits from each round of imputation
all_imp <- list()

# Repeat Imputation & Analysis steps m times
for (k in 1:m) {
  # Step 1: Imputation
  ## Draw coefficients from the multivariate normal distribution based on `imp_mod`
  coeff_m <- mvrnorm(n = 1, mu = imp_coeff, Sigma = imp_cov)
  ## Use the drawn coefficients to calculate imputed values
  imp_m <- coeff_m[1] + coeff_m[2] * log(movies$votes) + coeff_m[3] * movies$runtime + coeff_m[4] * movies$is_comedy +
            coeff_m[5] * movies$is_drama
  ## Replace with the non-missing ratings
  imp_m[!is.na(movies$rating_miss)] <- movies$rating_miss[!is.na(movies$rating_miss)]
  # Step 2: Analysis
  ## Fit the analysis model to the imputed dataset
  fit_m <- lm(formula = imp_m ~ log(votes) + runtime + is_comedy + is_drama,
                data = movies)
  ## Save the analysis model coefficients
  fit_m_coeff <- fit_m$coefficients
  ## Save the analysis  model covariance matrix
  fit_m_cov <- vcov(fit_m)
  # Save fit_m in all_imp list
  all_imp[[length(all_imp) + 1]] <- fit_m
}
```

Step 3: Pooling

We have m pairs $\left(\hat{\beta}^{(k)}, Var(\hat{\beta}^{(k)})\right)$. What do we do with them?

Rubin's Rules

RR1. Pooled parameter estimate

$$\hat{\beta} = \frac{1}{m}\sum_{k=1}^{m}\hat{\beta}^{(k)}$$

Rubin, D.B. (1987). Multiple Imputation for Nonresponse in Surveys. New York: John Wiley and Sons.

Step 3: Pooling

We have m pairs $\left(\hat{\beta}^{(k)}, Var(\hat{\beta}^{(k)})\right)$. What do we do with them?

Rubin's Rules

RR1. Pooled parameter estimate

$$\hat{\beta} = \frac{1}{m}\sum_{k=1}^{m}\hat{\beta}^{(k)}$$

This formula should look familiar! The pooled estimate of the regression model parameters is the arithmetic mean of the m model estimates $\hat{\beta}^{(1)}, \hat{\beta}^{(2)}, \ldots, \hat{\beta}^{(m)}$.

Rubin, D.B. (1987). Multiple Imputation for Nonresponse in Surveys. New York: John Wiley and Sons.

Step 3: Pooling

We have m pairs $\left(\hat{\beta}^{(k)}, Var(\hat{\beta}^{(k)})\right)$. What do we do with them?

Rubin's Rules

RR2. Pooled standard error

$$Var(\hat{\beta}) = \frac{1}{m}\sum_{k=1}^{m} Var(\hat{\beta}^{(k)}) + \frac{m}{m+1}\sum_{k=1}^{m}\left(\hat{\beta}^{(k)} - \hat{\beta}\right)^2$$

Rubin, D.B. (1987). Multiple Imputation for Nonresponse in Surveys. New York: John Wiley and Sons.

Step 3: Pooling

We have m pairs $\left(\hat{\beta}^{(k)}, Var(\hat{\beta}^{(k)})\right)$. What do we do with them?

Rubin's Rules

RR2. Pooled standard error

$$Var(\hat{\beta}) = \frac{1}{m}\sum_{k=1}^{m} Var(\hat{\beta}^{(k)}) + \frac{m}{m+1}\sum_{k=1}^{m}\left(\hat{\beta}^{(k)} - \hat{\beta}\right)^2$$

Here $\hat{\beta}$ is the pooled parameter estimate from RR1. The pooled variance is the sum of the average within-imputation variance (left) and the between-imputation variance (right).

Rubin, D.B. (1987). Multiple Imputation for Nonresponse in Surveys. New York: John Wiley and Sons.

Implementing Rubin's Rules in R: Coefficients

Recall that the m coefficient/variance pairs from the imputations are stored in the list, `all_imp`. First, we want $\hat{\beta}$ - the average coefficients.

R Code:

```
# Step 3: Pool
## Extract the coefficient estimates from each model
all_imp_coeff <- do.call(rbind, lapply(X = all_imp, FUN = coefficients))
head(all_imp_coeff)
     (Intercept) log(votes)     runtime  is_comedy   is_drama
[1,]    6.441845  0.1864583 -0.01711640 -0.4077372 0.13169405
[2,]    6.264469  0.1427084 -0.01257280 -0.1759375 0.08324976
[3,]    6.680133  0.1725428 -0.01881188 -0.3627855 0.25052812
[4,]    6.315815  0.2063566 -0.01777213 -0.3546369 0.08847459
[5,]    6.406188  0.1842048 -0.01704377 -0.3704962 0.12012270
[6,]    6.465431  0.1871892 -0.01765386 -0.3715542 0.19141473

## Calculate the mean coefficient for each variable
(beta_hat <- colMeans(all_imp_coeff))
(Intercept)  log(votes)     runtime   is_comedy    is_drama
 6.44014194  0.17390227 -0.01651444 -0.33668693  0.15367396
```

Implementing Rubin's Rules in R: Covariance

Then, we want $Var\left(\hat{\beta}\right)$ - the variance of the average coefficients.

R Code:

```
 ## Extract the coefficients' variance estimates from each model
all_imp_var <- do.call(rbind, lapply(X = all_imp, FUN = function(f) diag(vcov(f))))

## Calculate the within-imputation variance for each variable
within_var <- colMeans(all_imp_var)

## Calculate the between-imputation variance for each variable
### Subtract beta_hat from each imputation's coefficients
d <- all_imp_coeff - matrix(data = beta_hat, nrow = nrow(all_imp_coeff), ncol = ncol(all_imp_coeff), byrow = TRUE)

### Sum over the squared values of d (within each coefficient)
s <- colSums(d ^ 2)

### Pre-multiply by m / (m + 1)
between_var <- m / (m + 1) * s

## Calculate the pooled variance = within + between
(var_beta_hat <- within_var + between_var)

 (Intercept)   log(votes)      runtime    is_comedy     is_drama
0.3898924963 0.0091151170 0.0000679755 0.1131340931 0.1196680400
```

Comparing the Multiply Imputed Model to the Truth

Results:

	Truth		Multiple Imputation Analysis					
			MCAR		MAR		MNAR	
	Est	SE	Est	SE	Est	SE	Est	SE
(Intercept)	6.133	0.503	6.440	0.624	6.837	2.146	5.864	0.551
log(votes)	0.227	0.059	0.174	0.095	0.154	0.222	0.243	0.071
runtime	-0.018	0.005	-0.017	0.008	-0.017	0.005	-0.018	0.007
is_comedy	-0.327	0.235	-0.337	0.336	-0.447	0.234	-0.263	0.253
is_drama	0.188	0.229	0.154	0.346	0.080	0.231	0.235	0.251

Multivariate Imputation by Chained Equations (MICE)

Multiple imputation can also be applied in settings where more than one variable is missing.

A popular way to handle this is with **Multivariate Imputation by Chained Equations (MICE)**.

- Separate imputation models for each incomplete variable.
- Imputes variable-by-variable iteratively.
- Accommodates mixture of variable types (continuous, binary, ordinal).
- For now, just trust that it works and is a fairly straightforward extension of univariate multiple imputation as discussed.

For the mouse: https://www.jax.org/why-the-mouse.
For the knowledge: Buuren, S. . (2012). Flexible imputation of missing data. Boca Raton, FL: CRC Press.

Final thoughts on multiple imputation

Strengths:

- Imputed values are informed by other variables.
- Can add complexity to imputation model (splines, interactions, additional covariates).
- Pooled estimates/ standard errors straightforward via Rubin's Rules.
- Statistical inference based on multiple imputation is valid (based on simple imputation it is not).
- It's fairly common, so collaborators may be familiar.

Limitation:

- Misspecification of imputation model will yield incorrect results.