School of Computer Science and Software Engineering

CITS4009 Introduction to Data Science

SEMESTER 2, 2017: CHAPTER 4 MANAGING DATA

Chapter Objectives

- Fixing data quality problems
- Organizing your data for the modeling process

Cleaning Data

- This is where you fix issues you faced during the data exploration (discussed in chapter 3 chapter).
- Treating missing values (NAs):
  - To Drop or not to Drop?

  If the missing data represents a fairly small fraction of the dataset, it's probably safe just to drop these customers from your analysis.

  - Missing data in categorical variables:

  The most straightforward solution is just to create a new category for the variable, called missing.

- Why so many customers are missing this information?
  1. It could just be bad record-keeping
  2. but it could be semantic.
- Why a new variable?

Make a new variable from the original to differentiate the original record from the fix in case you second-guess or redo data cleaning.

Missing values in numeric data:

```
> summary(custdata$Income)
```

```
Min. 1st Qu.  Median    Mean  3rd Qu.    Max.     NA's
0    25000     45000     66200 82000    615000     328
```

You believe that income is still an important predictor of the probability of health insurance coverage, so you still want to use the variable. What do you do?

When values are missing randomly

you can replace the missing values with the expected, or mean, income

```
> meanIncome <- mean(custdata$Income, na.rm=T)
> Income.fix <- ifelse(is.na(custdata$Income),
                       meanIncome,
                       custdata$Income)
> summary(Income.fix)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
      0   35000   66200   66200   66200  615000
```

Don't forget the argument "na.rm=T"! Otherwise, the mean() function will include the NAs by default, and meanIncome will be NA.

- The estimate can be improved when you remember that income is related to other variables in your data—for instance, you know from your data exploration in the previous chapter that there's a relationship between age and income.
- It's possible that the customers with missing income data are systematically different from the others.
- The customers with missing income information truly have no. If this is so, then "filling in" their income information by using one of the preceding methods is the wrong thing to do.

When values are missing systematically

- Convert the numeric data into categorical data
- Then use methods such as `cut()`

- You could also replace all the NAs with zero income

Select some income ranges of interest. To use the cut() function, the upper and lower bounds should encompass the full income range of the data.

```
> breaks <-c(0, 10000, 50000, 100000, 250000, 1000000)
```

Cut the data into income ranges. The include.lowest=T argument makes sure that zero income data is included in the lowest income range category. By default it would be excluded.

```
> Income.groups <-
      cut(custdata$Income,
                  breaks=breaks, include.lowest=T)
```

```
> summary(Income.groups)
 [0,1e+04] (1e+04,5e+04] (5e+04,1e+05] (1e+05,2.5e+05] (2.5e+05,1e+06]
        63           312           178              98              21
      NA's
       328
```

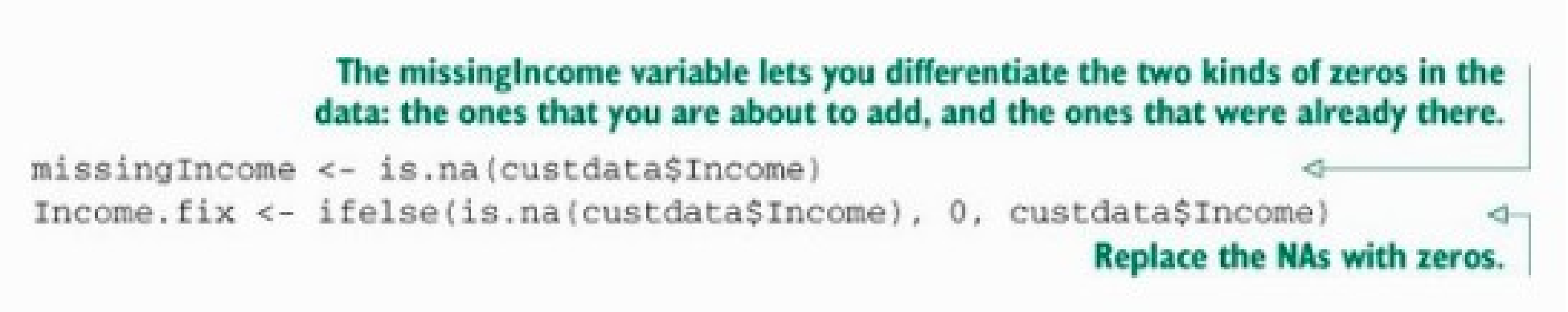
The cut() function produces factor variables. Note the NAs are preserved.

Add the "no income" category to replace the NAs.

```
> Income.groups <- as.character(Income.groups)
> Income.groups <- ifelse(is.na(Income.groups),
                          "no income", Income.groups)
```

To preserve the category names before adding a new category, convert the variables to strings.

```
> summary(as.factor(Income.groups))
(1e+04,5e+04] (1e+05,2.5e+05] (2.5e+05,1e+06]  (5e+04,1e+05]   [0,1e+04]
          312              98              21             178          63
    no income
          328
```



Tracking original NAs with an extra categorical variable

- You could also replace all the NAs with zero income and add an additional variable (we call it a *masking variable*) to keep track of which data points have been altered.
- Note that if the missing values really are missing randomly, then the masking variable will basically pick up the variable's mean value (at least in regression models).

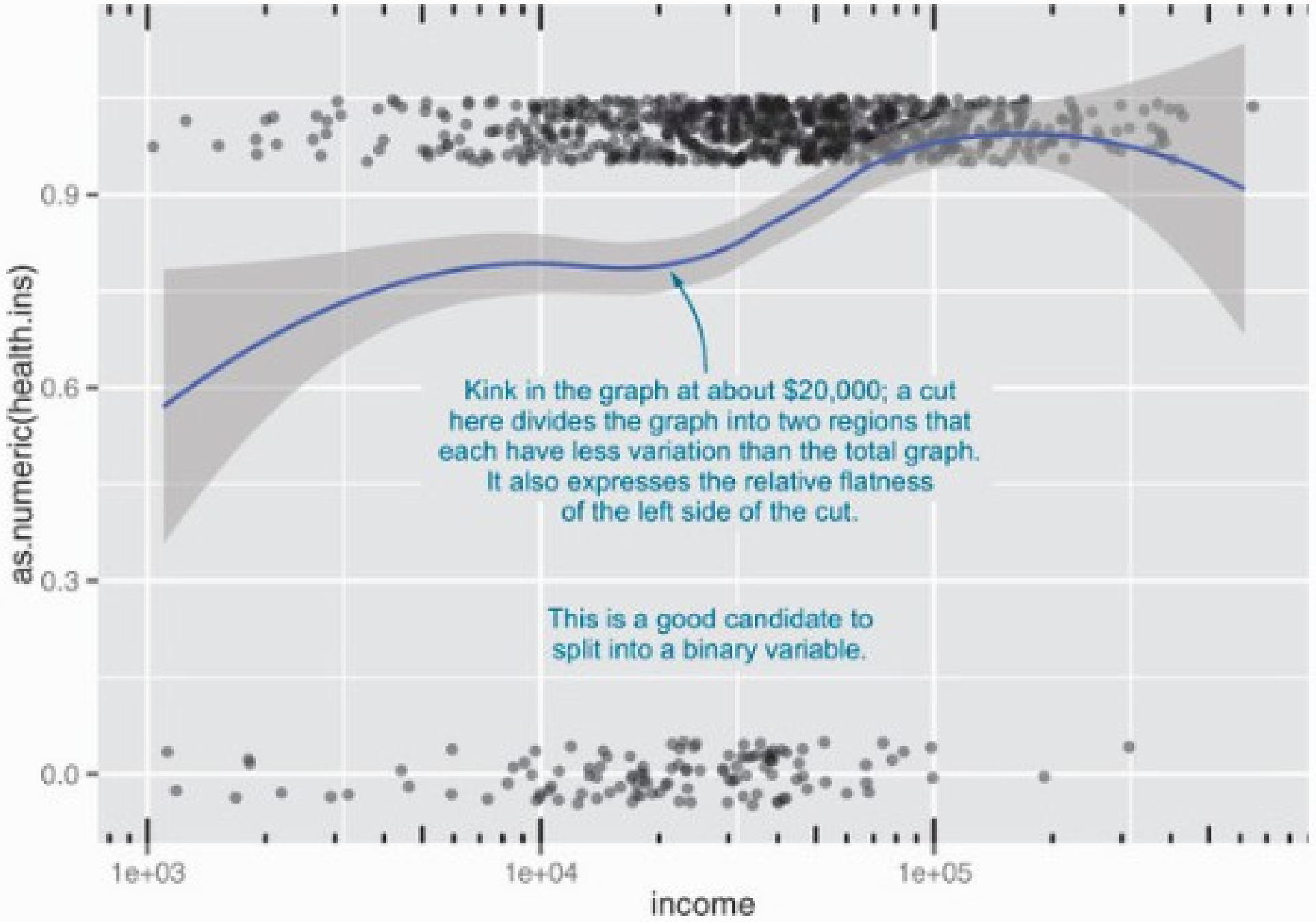
Data transformations

- The purpose of data transformation is to make data easier to model—and easier to understand
- The need for data transformation can also depend on which modeling method you plan to use
- For linear and logistic regression, for example, you ideally want to make sure that the relationship between input variables and output variable is approximately linear, and that the output variable is constant variance

Converting continuous variables to discrete

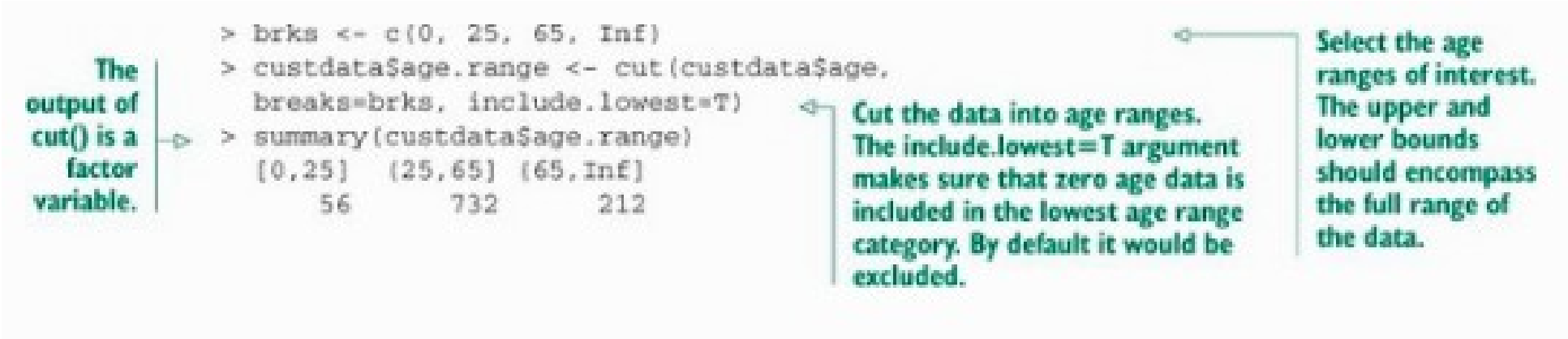
- Discretizing continuous variables is useful when the relationship between input and output isn’t linear, but you’re using a modeling technique that assumes it is, like regression.

you can replace the income variable with a Boolean variable that indicates whether income is less than $20,000



```
> custdata$income.lt.20K <- custdata$income < 20000
> summary(custdata$income.lt.20K)
   Mode      FALSE     TRUE     NA's
   logical     678      322        0
```

For more than a simple threshold use the **`cut()`** function.



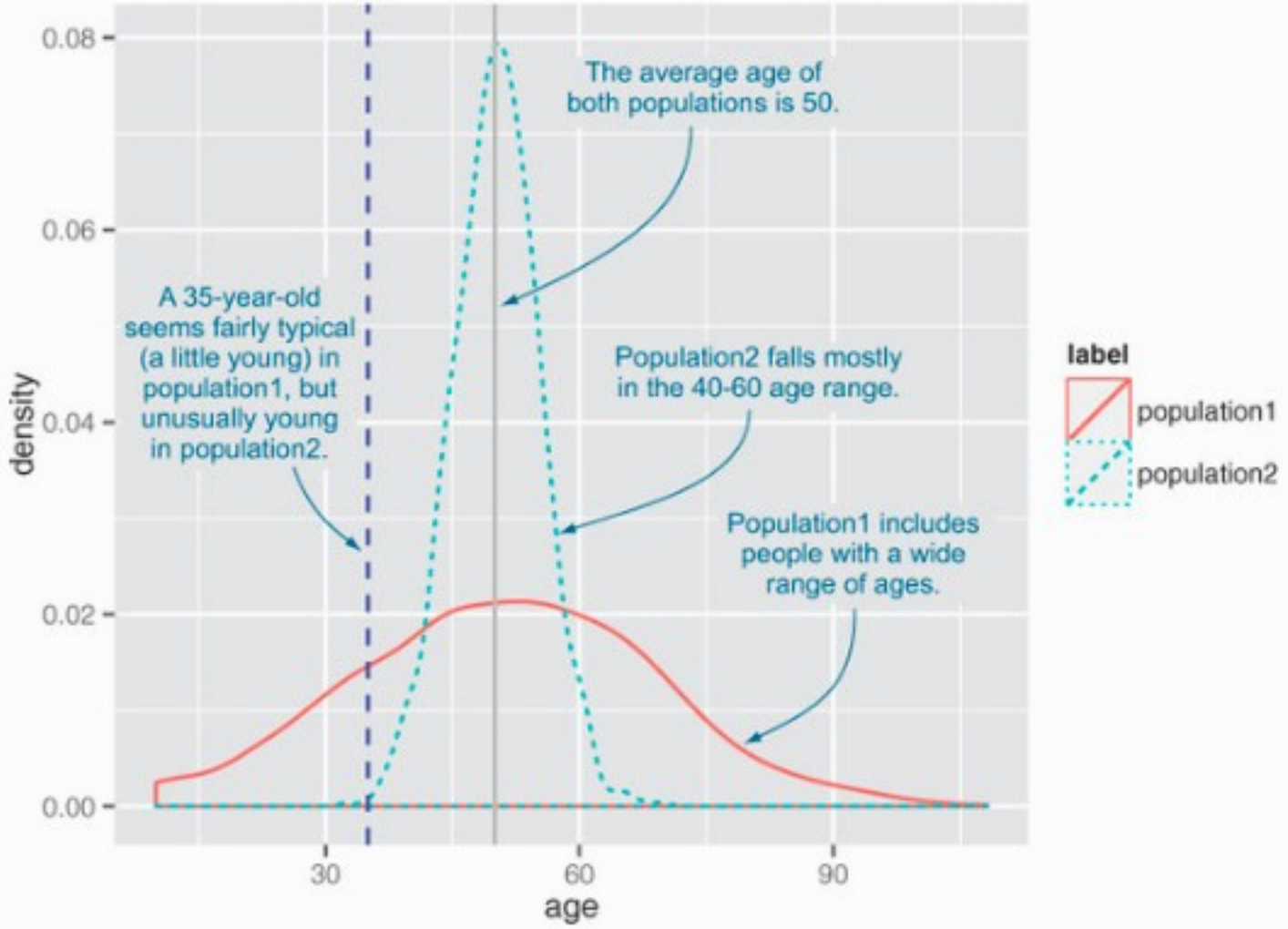
Converting age into ranges

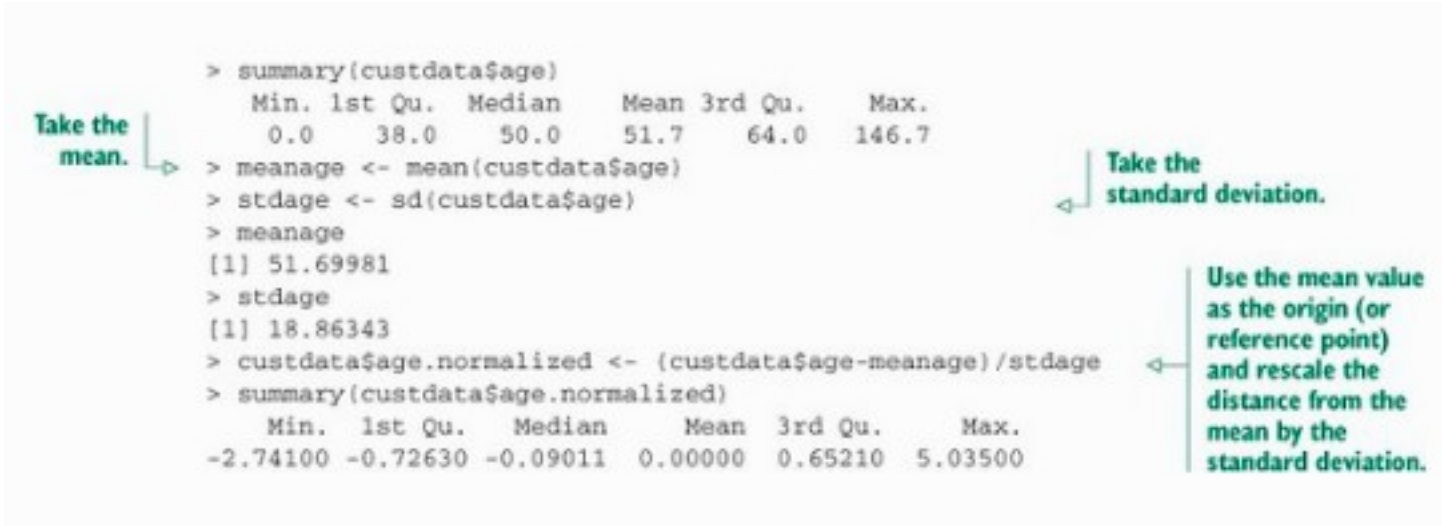
Normalization and rescaling

- Normalization is useful when absolute quantities are less meaningful than relative ones.

```
> summary(custdata$age)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
    0.0    38.0    50.0    51.7    64.0   146.7
> meanage <- mean(custdata$age)
> custdata$age.normalized <- custdata$age/meanage
> summary(custdata$age.normalized)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.0000  0.7350  0.9671  1.0000  1.2380  2.8370
```

- The typical age spread of your customers is summarized in the standard deviation.
- *rescale* your data by using the standard deviation as a unit of distance.
- A customer who is within one standard deviation of the mean is not much older or younger than typical. A customer who is more than one or two standard deviations from the mean can be considered much older, or much younger.



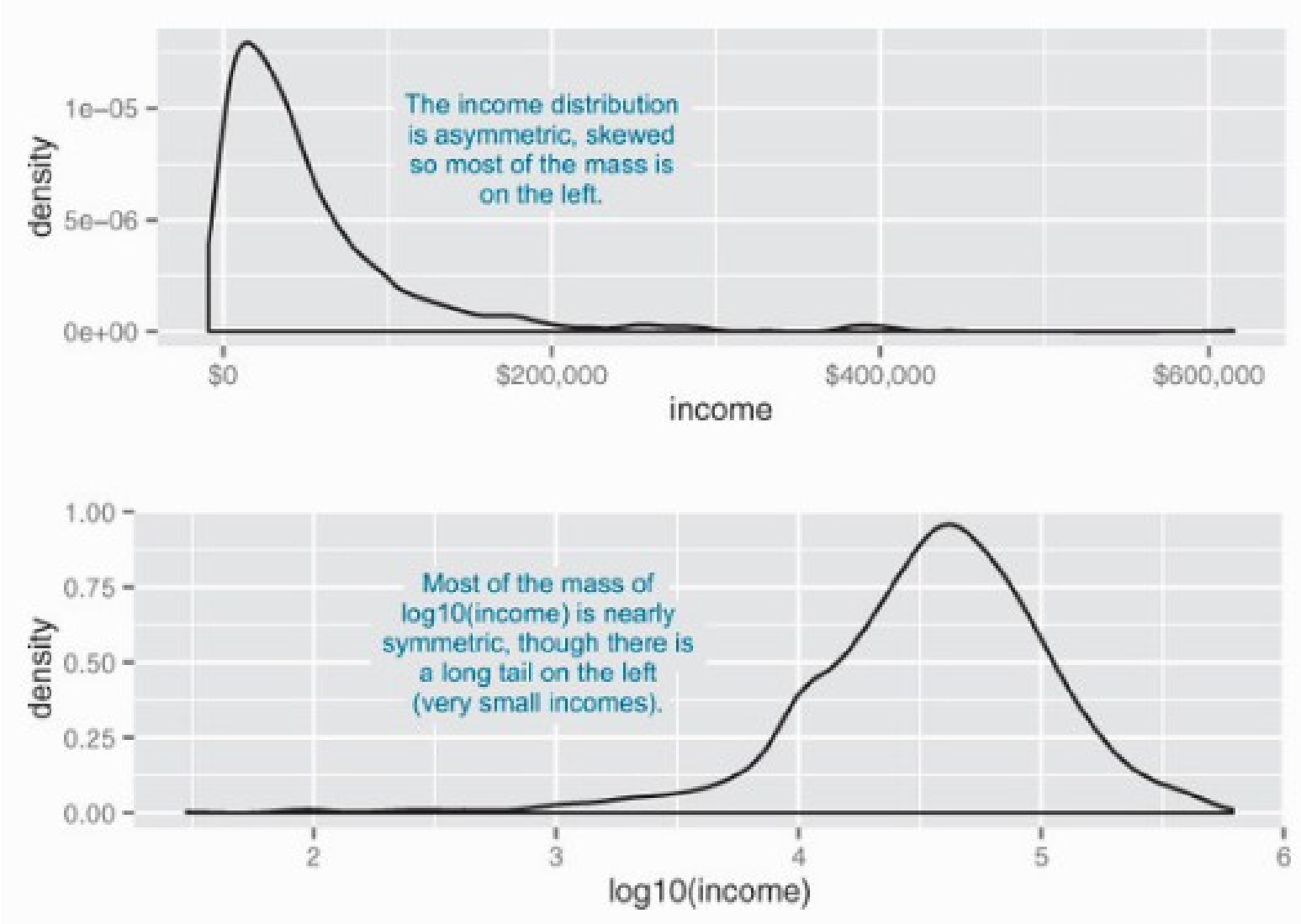


- Values less than -1 signify customers younger than typical; values greater than 1 signify customers older than typical.

- Normalizing by mean and standard deviation is most meaningful when the data distribution is roughly symmetric. Next, we'll look at a transformation that can make some distributions more symmetric.

Log transformations for skewed and wide distributions

- Monetary amounts—incomes, customer value, account, or purchase sizes—are some of the most commonly encountered sources of skewed distributions in data science applications
- Taking the log of the data can restore symmetry to it.



A nearly lognormal distribution and its log

A technicality

- For the purposes of modeling, which logarithm you use—natural logarithm, log base 10, or log base 2—is generally not critical.
- In regression, for example, the choice of logarithm affects the magnitude of the coefficient that corresponds to the logged variable, but it doesn't affect the value of the outcome.
- We like to use log base 10 for monetary amounts, because orders of ten seem natural for money: $100, $1000, $10,000, and so on.

A technicality- Cont.

- Generally a good idea to log transform data with values that range over several orders of magnitude:
- first, because modeling techniques often have a difficult time with very wide data ranges; and
- second, because such data often comes from multiplicative processes, so log units are in some sense more natural.

Additive Process

when you're studying weight loss, the natural unit is often pounds or kilograms. If you weigh 150 pounds and your friend weighs 200, you're both equally active, and you both go on the exact same restricted-calorie diet, then you'll probably both lose about the same number of pounds—in other words, how much weight you lose doesn't (to first order) depend on how much you weighed in the first place, only on calorie intake.

Multiplicative Process

- If management gives everyone in the department a raise, it probably isn't giving everyone $5,000 extra. Instead, everyone gets a 2% raise: how much extra money ends up in your paycheck depends on your initial salary.
- The natural unit of measurement is percentage, not absolute dollars.

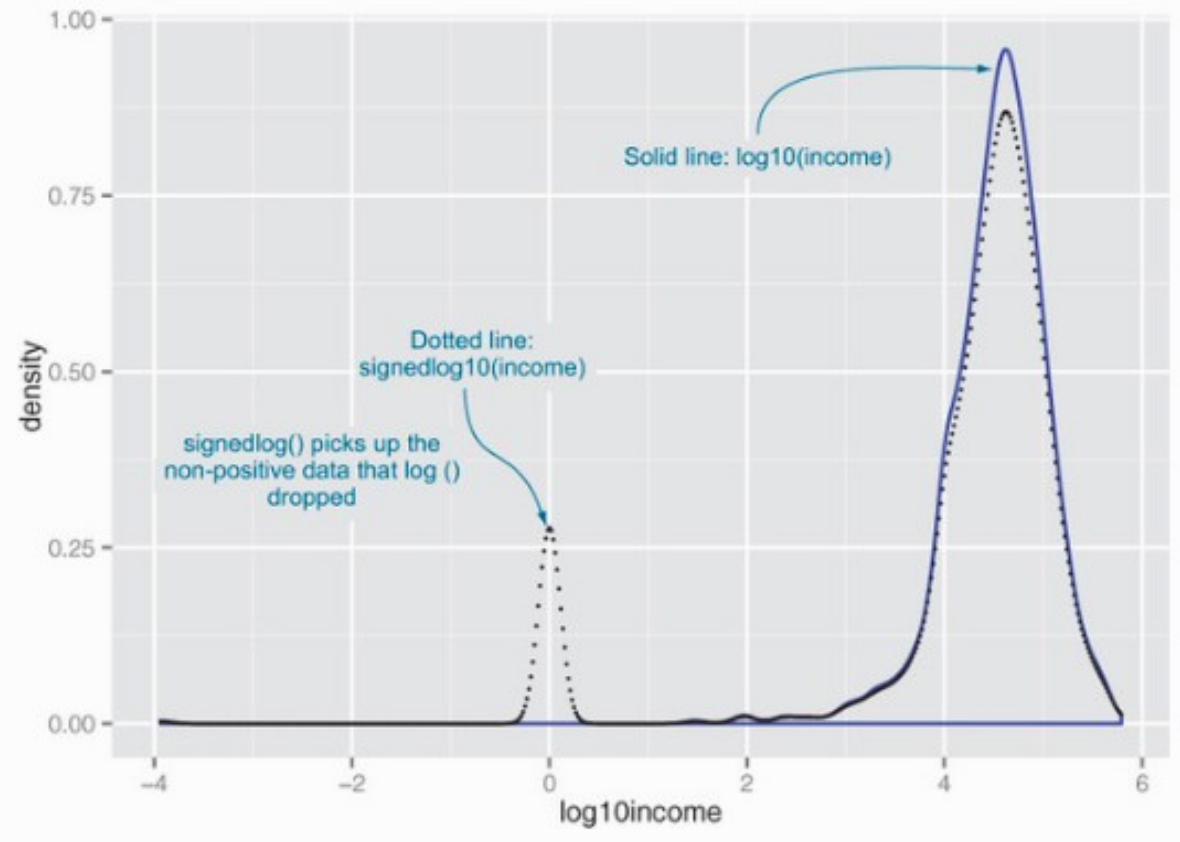
A technicality

- The logarithm only works if the data is non-negative.
- There are other transforms, such as ***arcsinh***, that you can use to decrease data range if you have zero or negative values.
- In applications where the skewed data is monetary (like account balances or customer value), we instead use what we call a signed logarithm. A signed logarithm takes the logarithm of the absolute value of the variable and multiplies by the appropriate sign. Values strictly between -1 and 1 are mapped to zero.

The difference between log and signed log

Here's how to calculate signed log base 10, in R:

```
signedlog10 <- function(x) {
   ifelse(abs(x) <= 1, 0,
sign(x)*log10(abs(x)))
}
```



Sampling for modeling and validation

- Sampling is the process of selecting a subset of a population to represent the whole, during analysis and modeling.
- In the current era of big datasets, some people argue that computational power and modern algorithms let us analyze the entire large dataset without the need to sample.

Why Sampling?

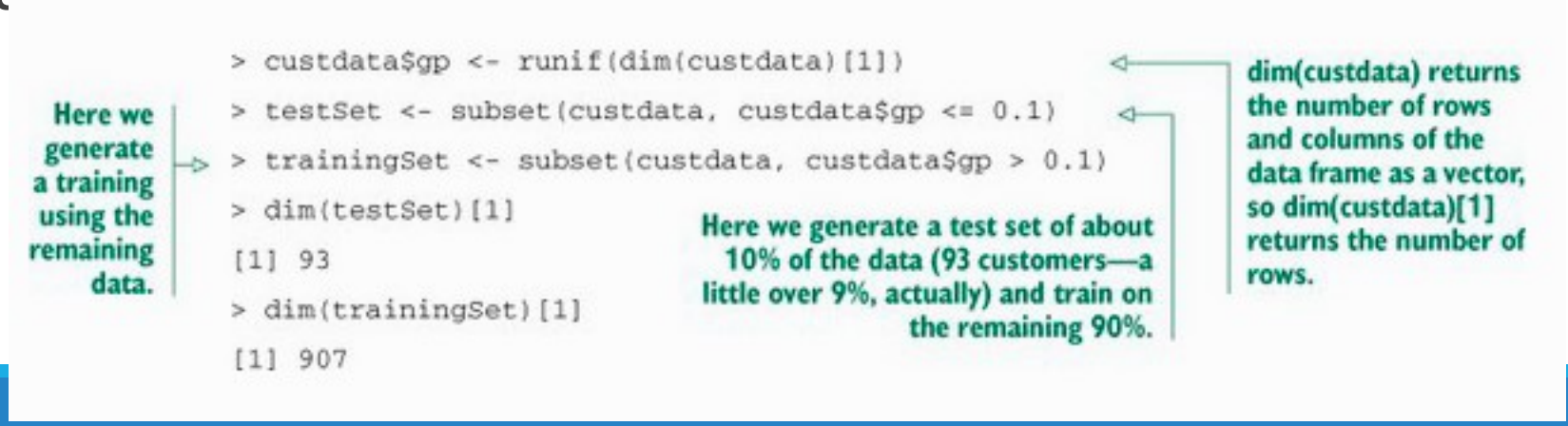
- sampling is a necessary task for other reasons.:
  1. When you're in the middle of developing or refining a modeling procedure, it's easier to test and debug the code on small subsamples before training the model on the entire dataset.
  2. Visualization can be easier with a subsample of the data; `ggplot` runs faster on smaller datasets, and too much data will often obscure the patterns in a graph
  3. And often it's not feasible to use your entire customer base to train a model.
  4. It's important that the dataset that you do use is an accurate representation of your population as a whole
  5. The other reason to sample your data is to create test and training splits.

Test and training splits

- For prediction models:
  - Need data to build model (training set)
  - Need data to test model (test set)
- **The training set** is the data that you feed to the model-building algorithm—regression, decision trees, and so on—so that the algorithm can set the correct parameters to best predict the outcome variable.
- **The test set** is the data that you feed into the resulting model, to verify that the model's predictions are accurate.

Creating a sample group column

- A convenient way to manage random sampling is to add a sample group column to the data frame. The sample group column contains a number generated uniformly from zero to one, using the **`runif`** function.
- You can draw a random sample of arbitrary size from the data frame by using the appropriate threshold on the sample group column.



Reproducible sampling is not just a trick for R

If your data is in a database or other external store, and you only want to pull a subset of the data into R for analysis, you can draw a reproducible random sample by generating a sample group column in an appropriate table in the database, using the SQL command **`RAND`** .

Record grouping

- If you're modeling a question at the *household level* rather than the *customer level*, then every member of a household should be in the same group (*test or training*). In other words, the random sampling also has to be at the household level.
- Suppose your customers are marked both by a household ID and a customer ID (so the unique ID for a customer is the combination **`(household_id, cust_id)`**.(This is shown next figure).
- We want to split the households into a training set and a test set.

1- Example of dataset with customers and households:

	household_id	cust_id	income
Household 1	hh1	cust1	30200
Household 2	hh2	cust1	24800
	hh2	cust2	134800
Household 3	hh3	cust1	299000
	hh3	cust2	65000
	hh3	cust3	95000
Household 4	hh4	cust1	38800
	hh4	cust2	0
Household 5	hh5	cust1	100300
	hh5	cust2	27000

3- Example of dataset with customers and households:

	household_id	cust_id	income	gp
Household 1	hh1	cust1	30200	0.8625189
Household 2	hh2	cust1	24800	0.8880607
	hh2	cust2	134800	0.8880607
Household 3	hh3	cust1	299000	0.9130094
	hh3	cust2	65000	0.9130094
	hh3	cust3	95000	0.9130094
Household 4	hh4	cust1	38800	0.5244124
	hh4	cust2	0	0.5244124
Household 5	hh5	cust1	100300	0.5388283
	hh5	cust2	27000	0.5388283

Note that each member of a household has the same group number

2- Ensuring test/train split doesn't split inside a household:

```
hh <- unique(hhdata$household_id)

households <- data.frame(household_id = hh, gp = runif(length(hh)))

hhdata <- merge(hhdata, households, by="household_id")
```

Get all unique household IDs from your data frame.

Create a temporary data frame of household IDs and a uniformly random number from 0 to 1.

Merge new random sample group column back into original data frame.

Data provenance

- You'll want to add a column (or columns) to record data provenance
- when your dataset was collected, perhaps what version of your data cleaning procedure was used on the data before modeling, and so on.
- This is akin to version control for data.
- It's handy information to have, to make sure that you're comparing apples to apples when you're in the process of improving your model, or comparing different models or different versions of a model.

Key Takeaways

- What you do with missing values depends on how many there are, and whether they're missing randomly or systematically.
- When in doubt, assume that missing values are missing systematically.
- Appropriate data transformations can make the data easier to understand and easier to model.
- Normalization and rescaling are important when relative changes are more important than absolute ones.
- Data provenance records help reduce errors as you iterate over data collection, data treatment, and modeling.