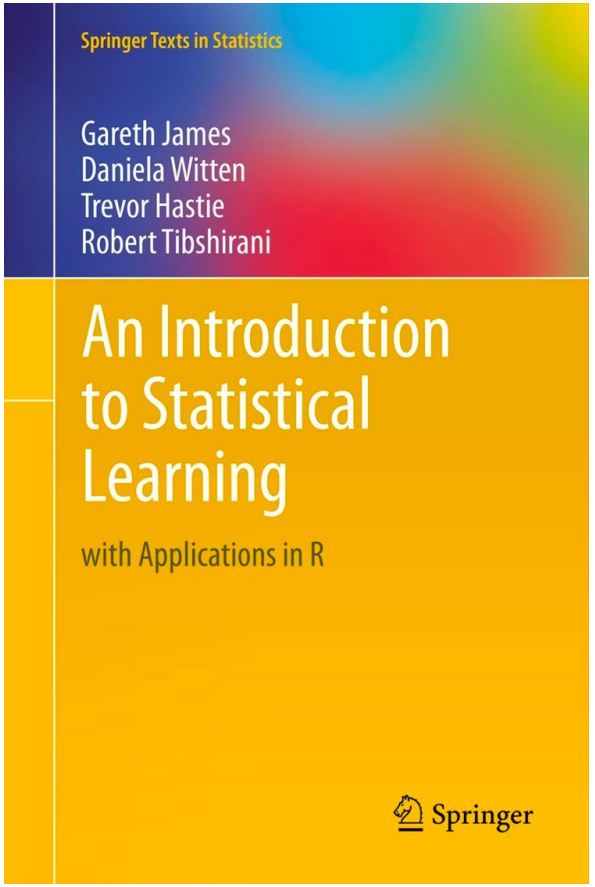
Topic07.Exploring/managing data!

資料科學 Data Science
張家銘 Jia-Ming Chang
政治大學資訊科學系

Copyright declaration 版權說明

- Some of the figures in this presentation are taken from "An Introduction to Statistical Learning, with applications in R" (Springer, 2013) with permission from the authors: G. James, D. Witten, T. Hastie and R. Tibshirani.
- The web site of the book
- The credit of individual is indicated in the bottom part of the slide.
  - ie.,

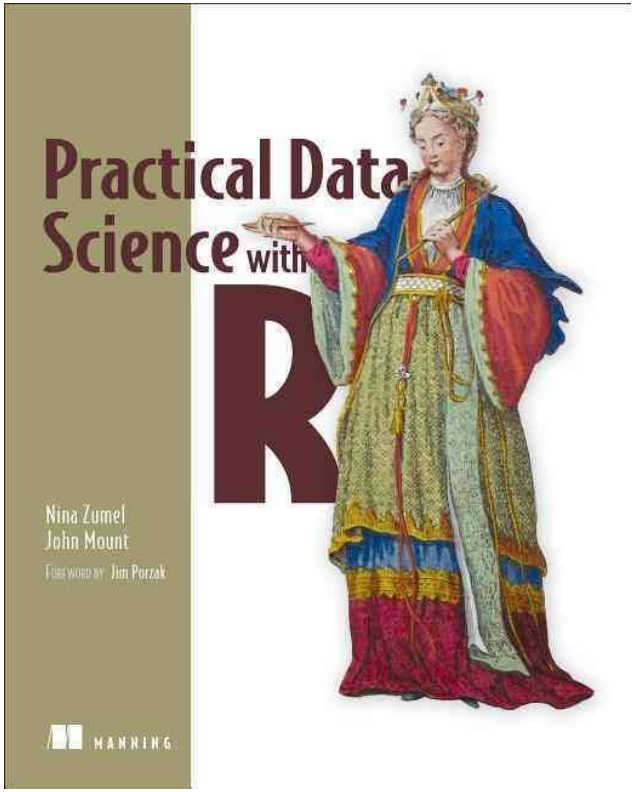
Figure 3.18, *An Introduction to Statistical Learning with Applications in R,* by Gareth James, Daniela Witten, Trevor Hastie and Robert Tibshirani



Copyright declaration 版權說明

- Some of the figures in this presentation are taken from "Practical Data Science with R" (Manning, 2019)
- The web site of the book
- The credit of individual is indicated in the bottom part of the slide.
  - ie.,

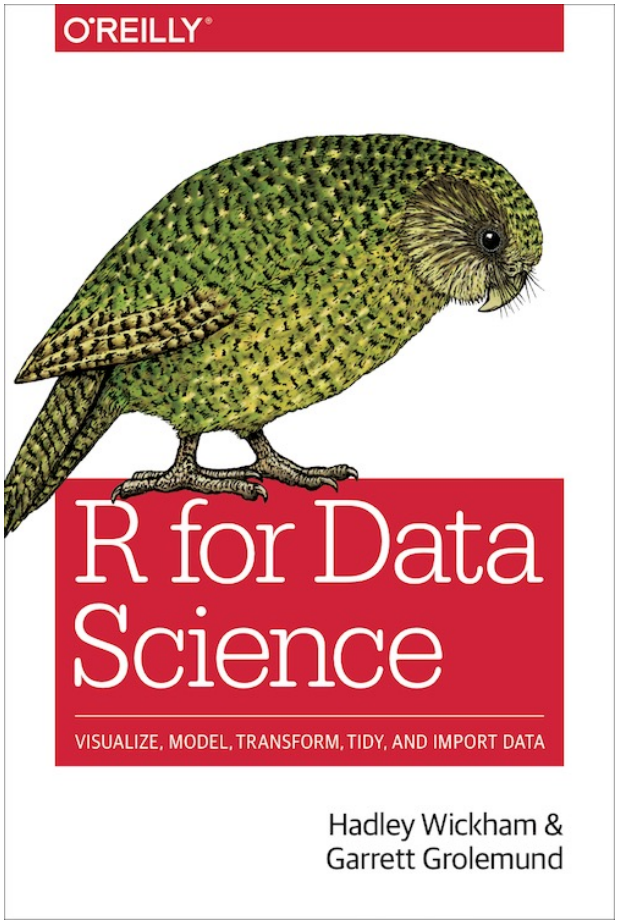
Figure 7.6, *Practical Data Science with R* by Nina Zumel and John Mount



Copyright declaration 版權說明

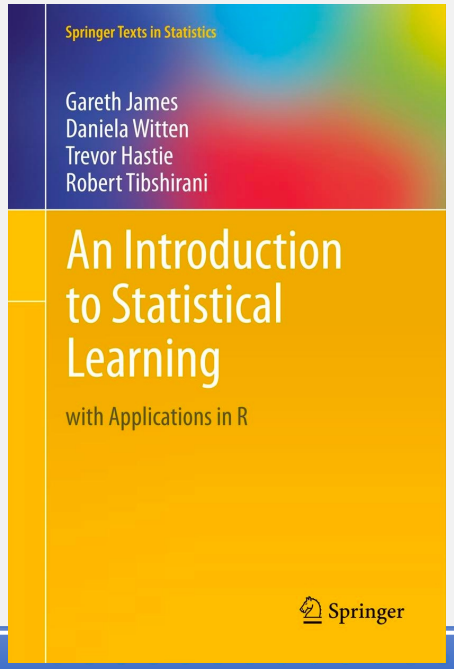
- Some of the figures in this presentation are taken from "R for Data Science" under the Creative Commons Attribution-NonCommercial-NoDerivs 3.0 License.
- The web site of the book
- The credit of individual is indicated in the bottom part.
  - ie.,

R for Data Science by Garrett Grolemund, Hadley Wickham



Data Science this week

- In-depth introduction to machine learning in 15 hours of expert videos
- https://www.r-bloggers.com/in-depth-introduction-to-machine-learning-in-15-hours-of-expert-videos/



Recap for the last week

10 DATA DESIGN DOS AND DON'TS

Designing your data doesn't have to be overwhelming. With a basic understanding of how different data sets should be visualized, along with a few fundamental design tips and best practices, you can create more accurate, more effective data visualizations. Follow these 10 tips to ensure your design does your data justice.

1 | DO USE ONE COLOR TO REPRESENT EACH CATEGORY.

2 | DO ORDER DATA SETS USING LOGICAL HEIRARCHY.

3 | DO USE CALLOUTS TO HIGHLIGHT IMPORTANT OR INTERESTING INFORMATION.

4 | DO VISUALIZE DATA IN A WAY THAT IS EASY FOR READERS TO COMPARE VALUES.

5 | DO USE ICONS TO ENHANCE COMPREHENSION AND REDUCE UNNECESSARY LABELING.

6 | DON'T USE HIGH CONTRAST COLOR COMBINATIONS SUCH AS RED/GREEN OR BLUE/YELLOW.

7 | DON'T USE 3D CHARTS. THEY CAN SKEW PERCEPTION OF THE VISUALIZATION.

8 | DON'T ADD CHART JUNK. UNNECESSARY ILLUSTRATIONS, DROP SHADOWS, OR ORNAMENTATIONS DISTRACT FROM THE DATA.

9 | DON'T USE MORE THAN 6 COLORS IN A SINGLE LAYOUT.

10 | DON'T USE DISTRACTING FONTS OR ELEMENTS (SUCH AS BOLD, ITALIC, OR UNDERLINED TEXT).

Credit by *Data Visualization 101: How To Design Charts and Graphs, by HubSpot and Visage*

Interaction

```
mtcars %>% ggvis(~mpg, ~disp, size = ~vs) %>%
layer_points()
mtcars %>% ggvis(~wt, ~mpg, size := 300,
opacity := 0.4) %>% layer_points()
=>
mtcars %>% ggvis(~wt, ~mpg, size :=
input_slider(10, 100), opacity := input_slider(0,
1) ) %>% layer_points()
=>
slider <- input_slider(10, 1000)
mtcars %>% ggvis(~wt, ~mpg) %>%
layer_points(fill := "red", opacity := 0.5,
size := slider)
```

Interaction, input_*

- `input_checkbox()`: a check-box
- `input_checkboxgroup()`: a group of check boxes
- `input_numeric()`: a spin box
- `input_radiobuttons()`: pick one from a set options
- `input_select()`: create a drop-down text box
- `input_text()`: arbitrary text input

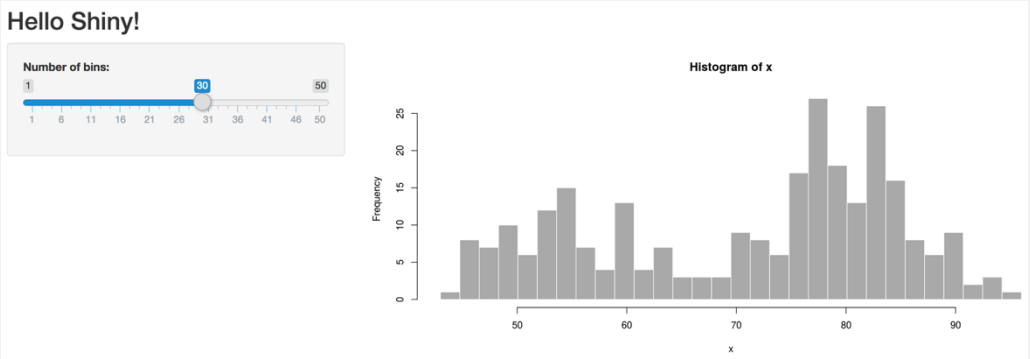
Shiny

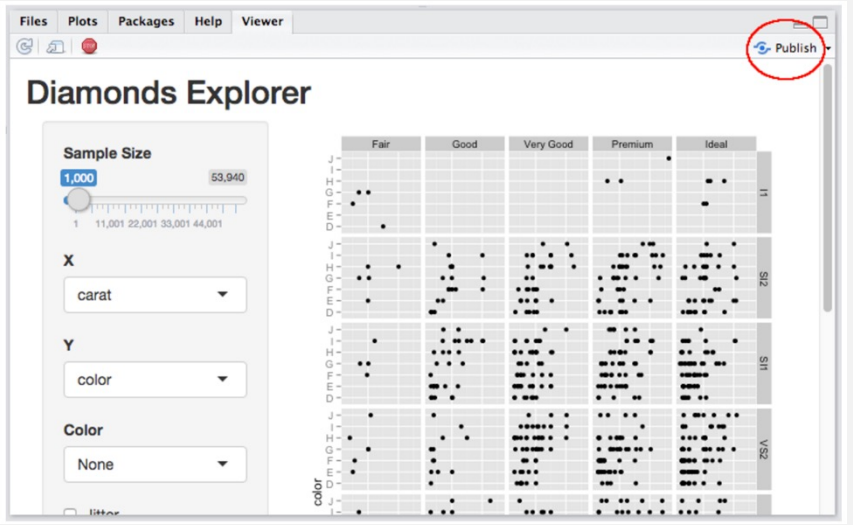
- an R package that makes it easy to build interactive web applications (apps) straight from R.
- http://shiny.rstudio.com/

Credit by http://shiny.rstudio.com/tutorial/lesson1/

Share by Shinyapps.io

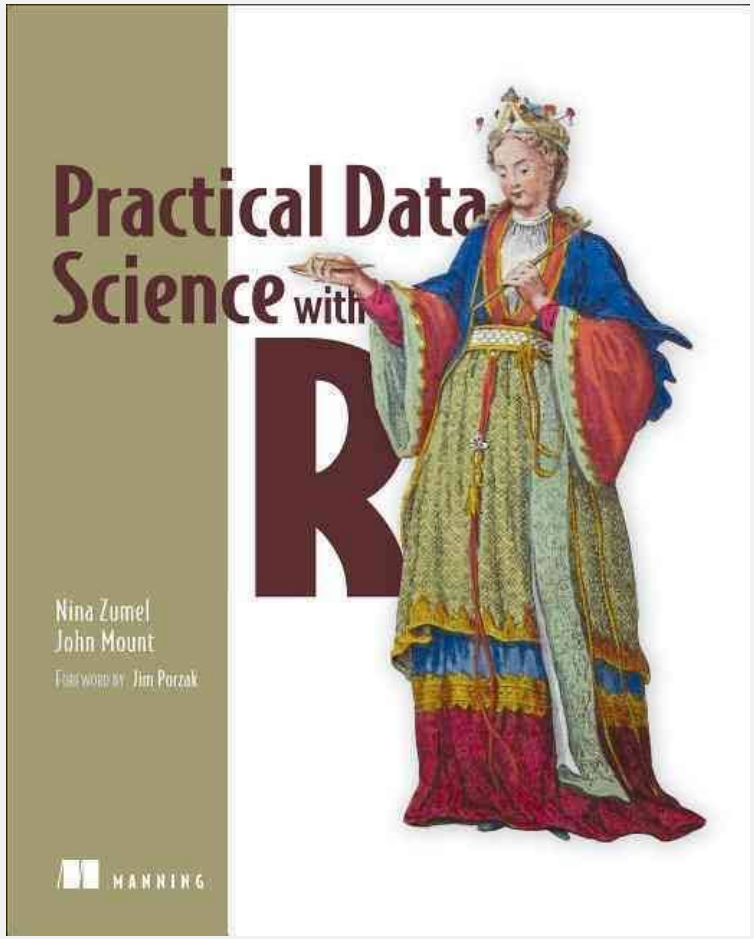
- https://changlab.shinyapps.io/app1/





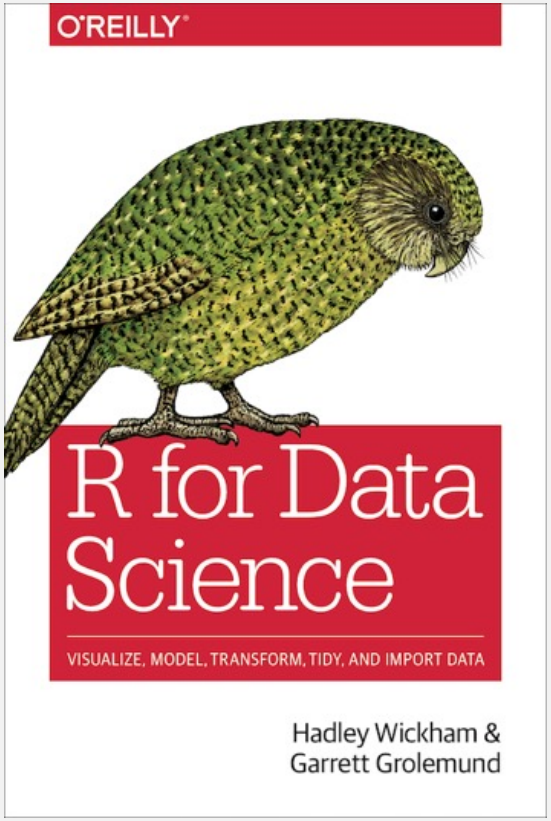
Today

- Chp. 3: Exploring data
- Chp. 4: Managing data



Today

- Cha 7. Exploratory Data Analysis
- Cha 11. Data import



Data Science project

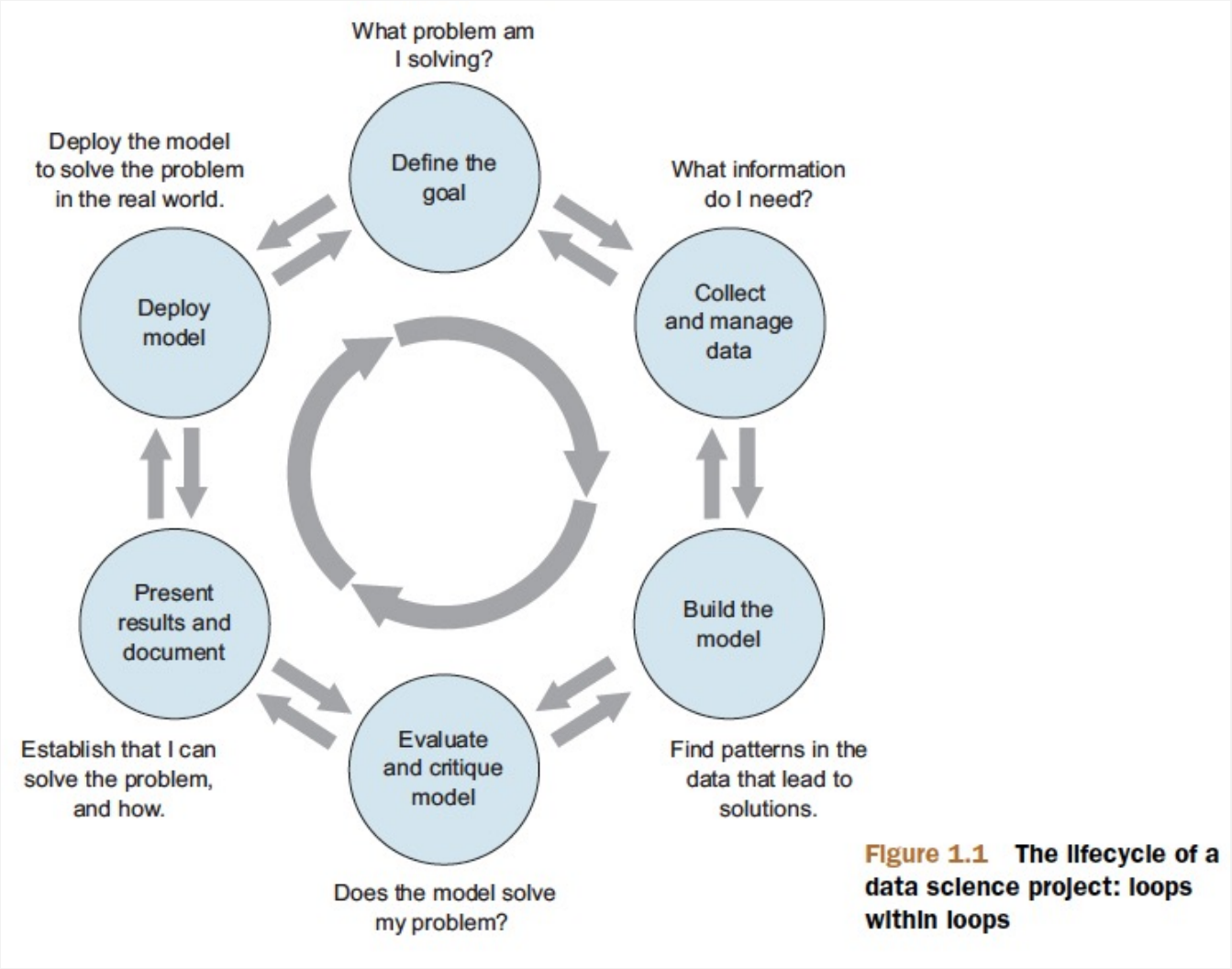


Figure 1.1 The lifecycle of a data science project: loops within loops

Figure1.1, *Practical Data Science with R* by Nina Zumel, John Mount

Explore & Manage data

- The data exploration and data cleaning stages are two of the more time-consuming—and also the most important—stages of the process.
- Without good data, you can't build good models.
- Time you spend here is time you don't waste elsewhere.

GIGO (Garbage In, Garbage Out)

Data Import by tidyverse package

http://tidyverse.org/

- a collection of R packages that share common philosophies and are designed to work together.
- This site is a work-in-progress guide to the tidyverse and its packages.
- install.packages("tidyverse")

Components

Import functions

- `library(tidyverse)`
  - `read_csv()` reads comma delimited files
  - `ead_csv2()` reads semicolon separated files
  - `read_tsv()` reads tab delimited files
  - `read_delim()` reads in files with any delimiter.
  - `read_fwf()` reads fixed width files
  - `read_log()` reads Apache style log files.

read_csv

- `heights <- read_csv("heights.csv")`
  - What is different with read.csv()?

in readr package in base R

`read_csv` vs. `read.csv`

read_csv

- They are typically much faster (~10x) than their base equivalents. Long running jobs have a progress bar, so you can see what's happening.
- If you're looking for raw speed, try `data.table::fread()`. It doesn't fit quite so well into the `tidyverse`, but it can be quite a bit faster.

read_csv vs. read.csv

essentially a modern version of data frame

- They produce `tibbles`, they don't convert character vectors to factors, use row names, or munge the column names.
- These are common sources of frustration with the base *R* functions.

read_csv vs. read.csv

- They are more reproducible. Base *R* functions inherit some behavior from your operating system and environment variables, so import code that works on your computer might not work on someone else's.

Writing to a file

- writeData.R

```
library(tidyverse)

challenge <-
read_csv(readr_example("challenge.csv"))

write_csv(challenge, "challenge.csv")
```

write_rds & read_rds

- # store data in R's custom binary format called RDS

```
write_rds(challenge, "challenge.rds")
read_rds("challenge.rds")
```

save() vs. saveRDS()

```
mod <- lm(mpg ~ wt, data = mtcars) # linear regression model
mod
save(mod, file = "mymodel.rda")
ls()
load(file = "mymodel.rda")
ls()
saveRDS(mod, "mymodel.rds")
mod2 <- readRDS("mymodel.rds")
ls()
identical(mod, mod2, ignore.environment = TRUE)
```

`save()` vs. `saveRDS()`

- `save()`
  - save many objects to a file in a single call
  - maintaining the original name
- `saveRDS()`
  - being a lower-level function, works with a single object at a time
  - forgets the original name of the object

The functions `save()`, `saveRDS()`, and `write_rds()` (from the `readr` package) are used to save R objects to disk, but they differ in their behavior and file formats:

1. `save()`:
   - Allows you to save one or more objects into a single file.
   - Creates a binary file in `.RData` or `.rda` format.
   - On loading, all objects saved will be loaded into the environment.
   - Syntax:

```r
save(object1, object2, ..., file = "filename.rda")
```

   - Usage example:

```r
save(mod, df, file = "analysis.rda")
```

2. `saveRDS()`:
   - Saves a single R object to disk in RDS format.
   - The output file will have an `.rds` extension.
   - When reading, you assign the loaded data to an object explicitly.
   - Syntax:

```r
saveRDS(object, file = "filename.rds")
```

   - Usage example:

```r
saveRDS(mod, file = "mymodel.rds")
```

3. `write_rds()` (from the `readr` package):
   - Equivalent to `saveRDS()` but provided by the `readr` package.
   - Offers some additional options, such as compressing the output file.
   - Syntax:

```r
write_rds(object, file = "filename.rds", compress = "none")
```

   - Usage example:

```r
write_rds(mod, file = "mymodel_compressed.rds", compress = "gz")
```

Key Differences:

- `save()` can store multiple objects but requires all objects to be loaded when the file is read.
- `saveRDS()` and `write_rds()` only save one object per file and require explicit assignment during loading.
- `write_rds()` is more flexible than `saveRDS()` with compression and other options.

Choosing Between Them:

- Use `save()` if you need to store multiple objects at once.
- Use `saveRDS()` or `write_rds()` for storing individual objects with more control over how they are loaded later.

1. **`.rds` Files**:
   - This format is specifically associated with the **`saveRDS()`** and **`readRDS()`** functions.
   - It is designed to store a single R object at a time.
   - When loading an **`.rds`** file with **`readRDS()`**, the object must be assigned explicitly to a variable.
   - Example usage:

```r
saveRDS(object, "file.rds")
loaded_object <- readRDS("file.rds")
```

2. **`.rda` (or `.RData`) Files**:
   - These files are used by the **`save()`** and **`load()`** functions.
   - Can store multiple R objects in a single file.
   - When loading an **`.rda`** or **`.RData`** file with **`load()`**, all objects saved in the file are directly loaded into the environment with their original names.
   - Example usage:

```r
save(object1, object2, file = "file.rda")
load("file.rda")  # object1 and object2 are loaded with their original names
```

Exploring data

Exploring data

1. Using **summary** statistics to explore data
2. Exploring data using visualization
3. Finding problems and issues during data exploration

Data exploration

- *summary statistics*: means and medians, variances, and counts
- *Visualization*: graphs of the data

Custdata

- expData.R
- custdata.tsv
  - Click “raw” to download it
- Load data

```
custdata <-
read.table('custdata.tsv',header=TRUE,sep='\t')
```

Data

- exampleData.rData
- Load data

```
load("exampleData.rData")
```

Setting expectations

- Is data you have available is good enough?
  - Very important in data science!!!

Using *summary* statistics to spot problems

- Which gender has higher employed rate?
- Any problem? too many missing values

```
summary(custdata)
```

Spotting problems using graphics and visualization

- A graphic should display as much information as it can, with the lowest possible cognitive strain to the viewer.
- Visualization is an iterative process. Its purpose is to answer questions about the data.

Spotting problems using graphics and visualization

- Make the data stand out. Specific tips for increasing clarity include
  - Avoid too many superimposed elements, such as too many curves in the same graphing space.
  - Find the right aspect ratio and scaling to properly bring out the details of the data.
  - Avoid having the data all skewed to one side or the other of your graph.

Some information is easier to read from a graph, and some from a summary

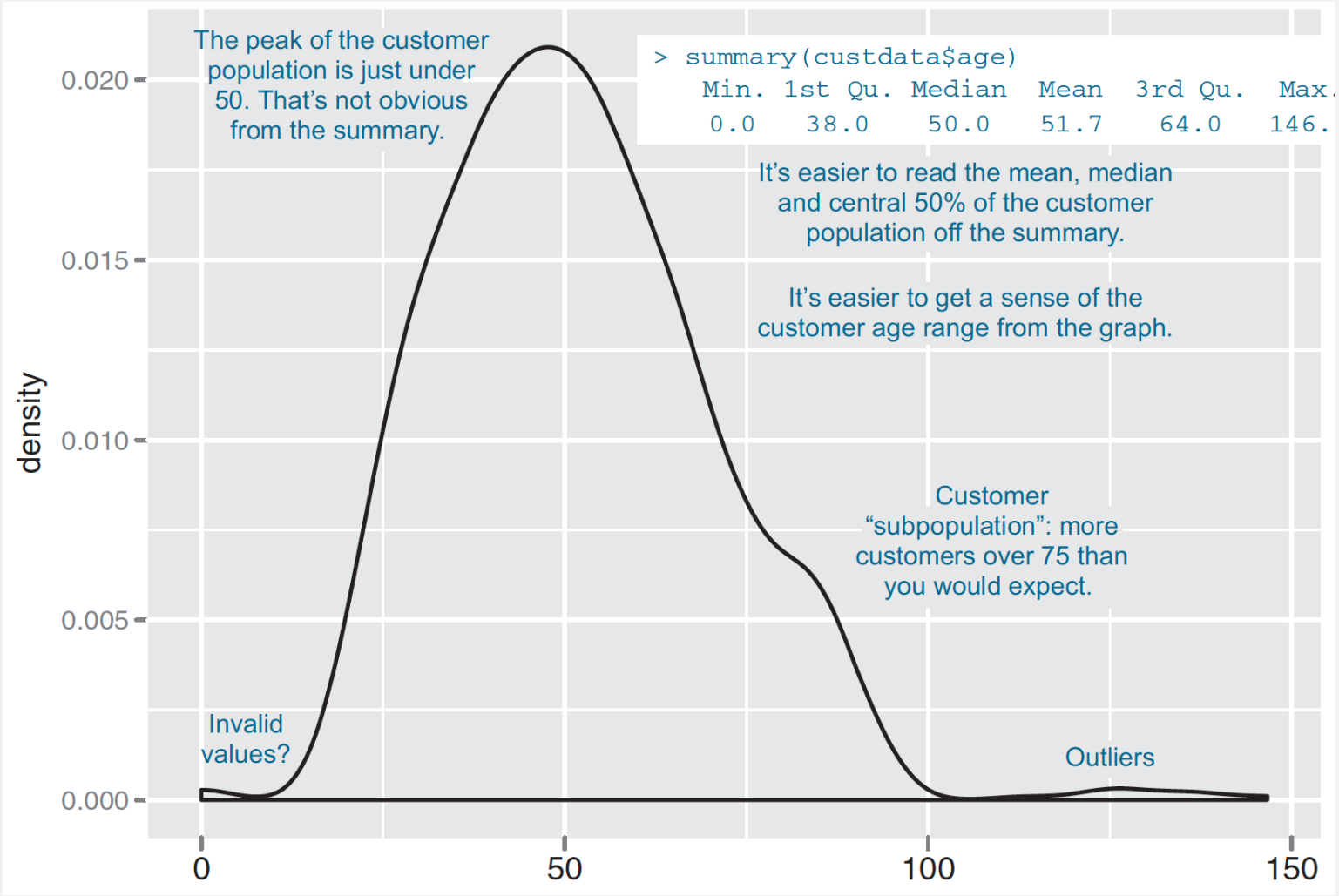


Figure 3.1, *Practical Data Science with R* by Nina Zumel, John Mount

Visualizations for one variable

Graph type	Uses
Histogram or density plot	Examines data range Checks number of modes Checks if distribution is normal/lognormal Checks for anomalies and outliers
Bar chart	Compares relative or absolute frequencies of the values of a categorical variable

Visually checking distributions for a single variable

- What is the peak value of the distribution?
- How many peaks are there in the distribution (unimodality vs bimodality)?
- How normal (or lognormal) is the data?
- How much does the data vary? Is it concentrated in a certain interval or in a certain category?

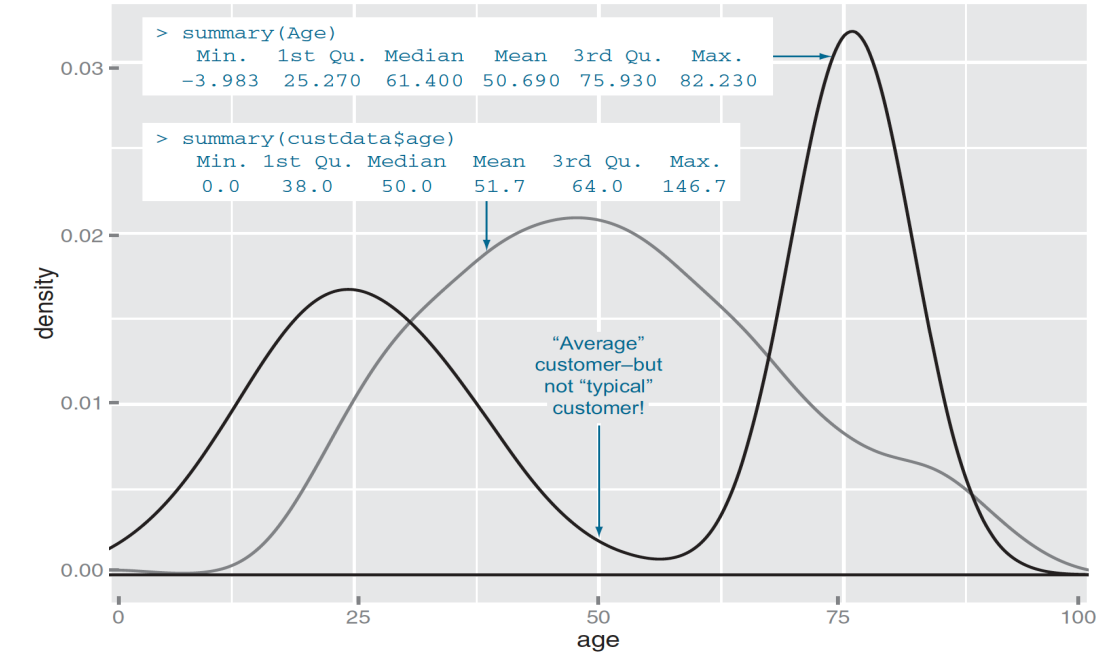


Figure3.2, *Practical Data Science with R* by Nina Zumel, John Mount

Visually checking distributions for a single variable

- Unimodal, bimodal, multimodal
- It wouldn't be a bad idea to model the two populations separately - especially if you're using linear or logistic regression

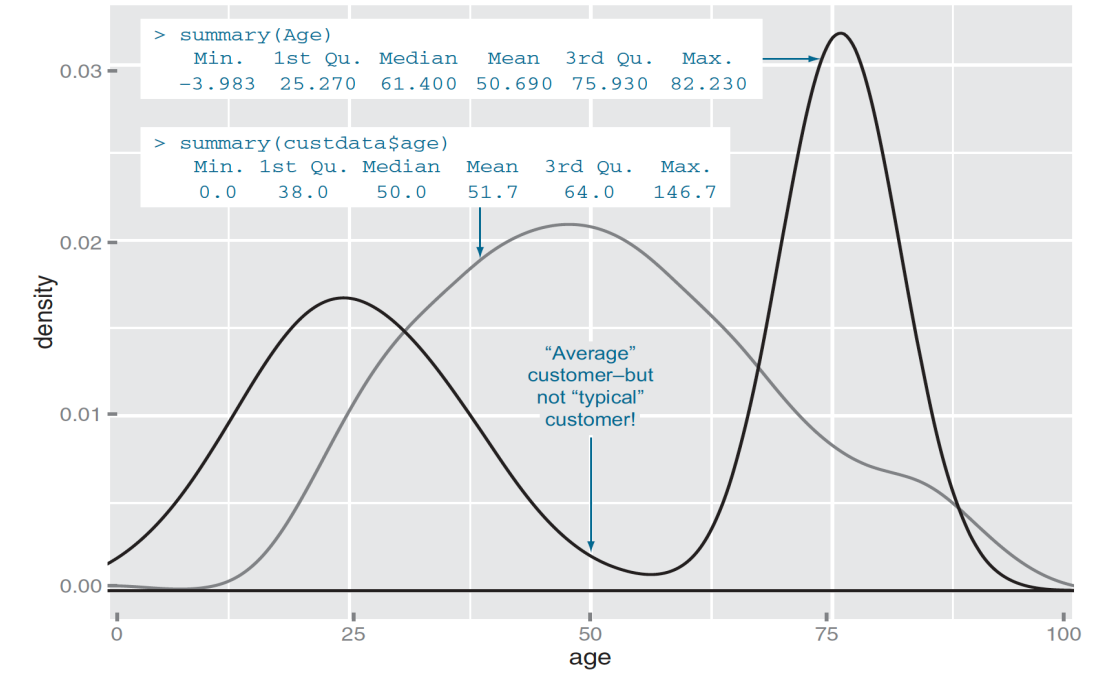


Figure3.2, *Practical Data Science with R* by Nina Zumel, John Mount

Bar Charts

```
statesums <- table(custdata$state.of.res)

statef <- as.data.frame(statesums)

colnames(statef)<-c("state.of.res", "count")

statef <- transform(statef, state.of.res=reorder(state.of.res, count))

ggplot(statef)+ geom_bar(aes(x=state.of.res,y=count),

   stat="identity", fill="gray") + coord_flip() +  theme(axis.text.y=element_text(size=rel(0.8)))
```

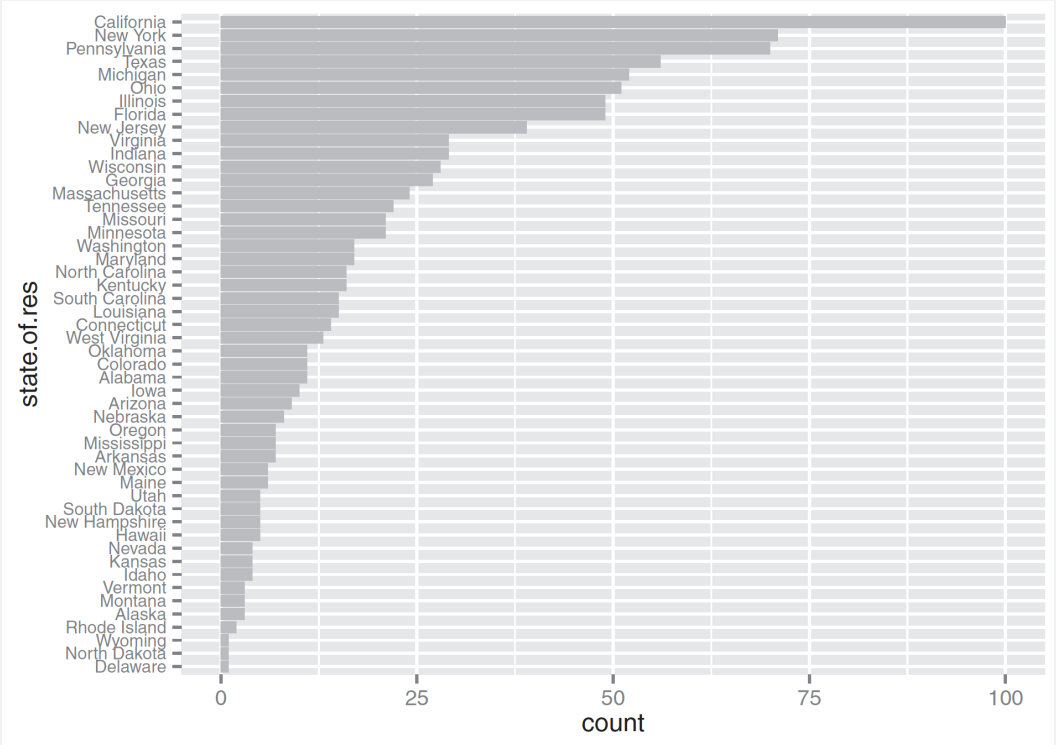


Figure3.8, *Practical Data Science with R* by Nina Zumel, John Mount

Typical problems revealed by data summaries

- Missing Values
- Invalid Values and Outliers
- Units
- Data Range => Managing data

1.Missing value

Missing Values

- Sometimes the fact that a value is missing is informative in and of itself
  - Is employment status unknown?
  - Did the company start collecting employment data only recently?
  - Does NA mean “not in the active workforce”

```
    custid          sex       is.employed         income                marital.stat health.ins                                housing.type recent.move
Min.   :   2068   F:440   Mode :logical   Min.   : -8700   Divorced/Separated:155   Mode :logical   Homeowner free and clear    :157   Mode :logical
1st Qu.: 345667   M:560   FALSE:73        1st Qu.: 14600   Married           :516   FALSE:159       Homeowner with mortgage/loan:412   FALSE:820
Median : 693403           TRUE :599       Median : 35000   Never Married     :233   TRUE :841       Occupied with no rent       : 11   TRUE :124
Mean   : 698500           NA's :328       Mean   : 53505   Widowed           : 96   NA's :0         Rented                      :364   NA's :56
3rd Qu.:1044606                           3rd Qu.: 67000                                             NA's                        : 56
Max.   :1414286                           Max.   :615000

 num.vehicles         age             state.of.res
Min.   : 0.000   Min.   :  0.0   California   :100
1st Qu.: 1.000   1st Qu.: 38.0   New York     : 71
Median : 2.000   Median : 50.0   Pennsylvania: 70
Mean   : 1.916   Mean   : 51.7   Texas        : 56
3rd Qu.: 2.000   3rd Qu.: 64.0   Michigan     : 52
Max.   : 6.000   Max.   :146.7   Ohio         : 51
NA's   :56.000                   (Other)      :600
```

Missing Values

- Do you include a variable with missing values in your model, or not?
  - Proportion matter

```
    custid          sex      is.employed          income                      marital.stat  health.ins                        housing.type recent.move
 Min.   :   2068   F:440   Mode :logical    Min.   : -8700   Divorced/Separated:155   Mode :logical   Homeowner free and clear    :157   Mode :logical
 1st Qu.: 345667   M:560   FALSE:73         1st Qu.: 14600   Married           :516   FALSE:159       Homeowner with mortgage/loan:412   FALSE:820
 Median : 693403           TRUE :599        Median : 35000   Never Married     :233   TRUE :841       Occupied with no rent       : 11   TRUE :124
 Mean   : 698500           NA's :328        Mean   : 53505   Widowed           : 96   NA's :0         Rented                      :364   NA's :56
 3rd Qu.:1044606                            3rd Qu.: 67000                                            NA's                        : 56
 Max.   :1414286                            Max.   :615000

  num.vehicles        age              state.of.res
 Min.   : 0.000   Min.   :  0.0   California  :100
 1st Qu.: 1.000   1st Qu.: 38.0   New York    : 71
 Median : 2.000   Median : 50.0   Pennsylvania: 70
 Mean   : 1.916   Mean   : 51.7   Texas       : 56
 3rd Qu.: 2.000   3rd Qu.: 64.0   Michigan    : 52
 Max.   : 6.000   Max.   :146.7   Ohio        : 51
 NA's   :56.000                   (Other)     :600
```

How to treat missing data?

- Drop all the rows where this field is missing
- Convert the missing values to o or to an additional category

Treating missing values (NAs)

- To Drop Or Not To Drop?

```
summary(custdata[is.na(custdata$housing.type),
c("recent.move","num.vehicles")])
```

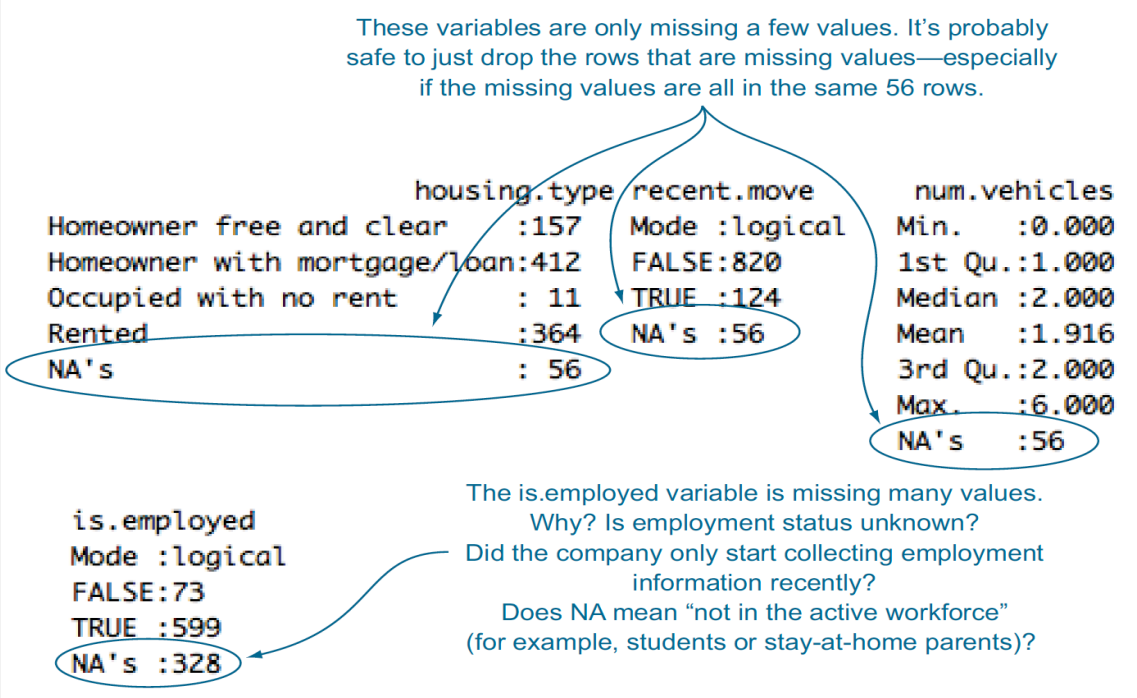


Figure 4.1, *Practical Data Science with R* by Nina Zumel, John Mount

Missing Data In Categorical Variables

```
custdata$is.employed.fix <- ifelse(is.na(custdata$is.employed),
                                   "missing",
                            ifelse(custdata$is.employed==T,
                                   "employed",
                                   "not employed"))
summary(as.factor(custdata$is.employed.fix))
custdata$is.employed.fix <- ifelse(is.na(custdata$is.employed),
                    "not in active workforce",
                     ifelse(custdata$is.employed==T,
                                   "employed",
                                    "not employed"))
```

Missing Data In Categorical Variables

- NA =>“missing” : prevent that most analysis functions in *R* will, by default, drop rows with missing data.
- Why a new variable?
  - a new variable called *is.employed.fix*, rather than simply replacing *is.employed*.
  - have the original variable on hand, in case we second-guess our data cleaning and want to redo it.

Missing values in numeric data

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

- Two types
  - When values are missing randomly
  - When values are missing systematically

When values are missing randomly

ignoring missing values

```
meanIncome <- mean(custdata$Income, na.rm=T)
Income.fix <- ifelse(is.na(custdata$Income),
                     meanIncome,
                     custdata$Income)
summary(Income.fix)
summary(custdata$Income)
```

When values are missing systematically

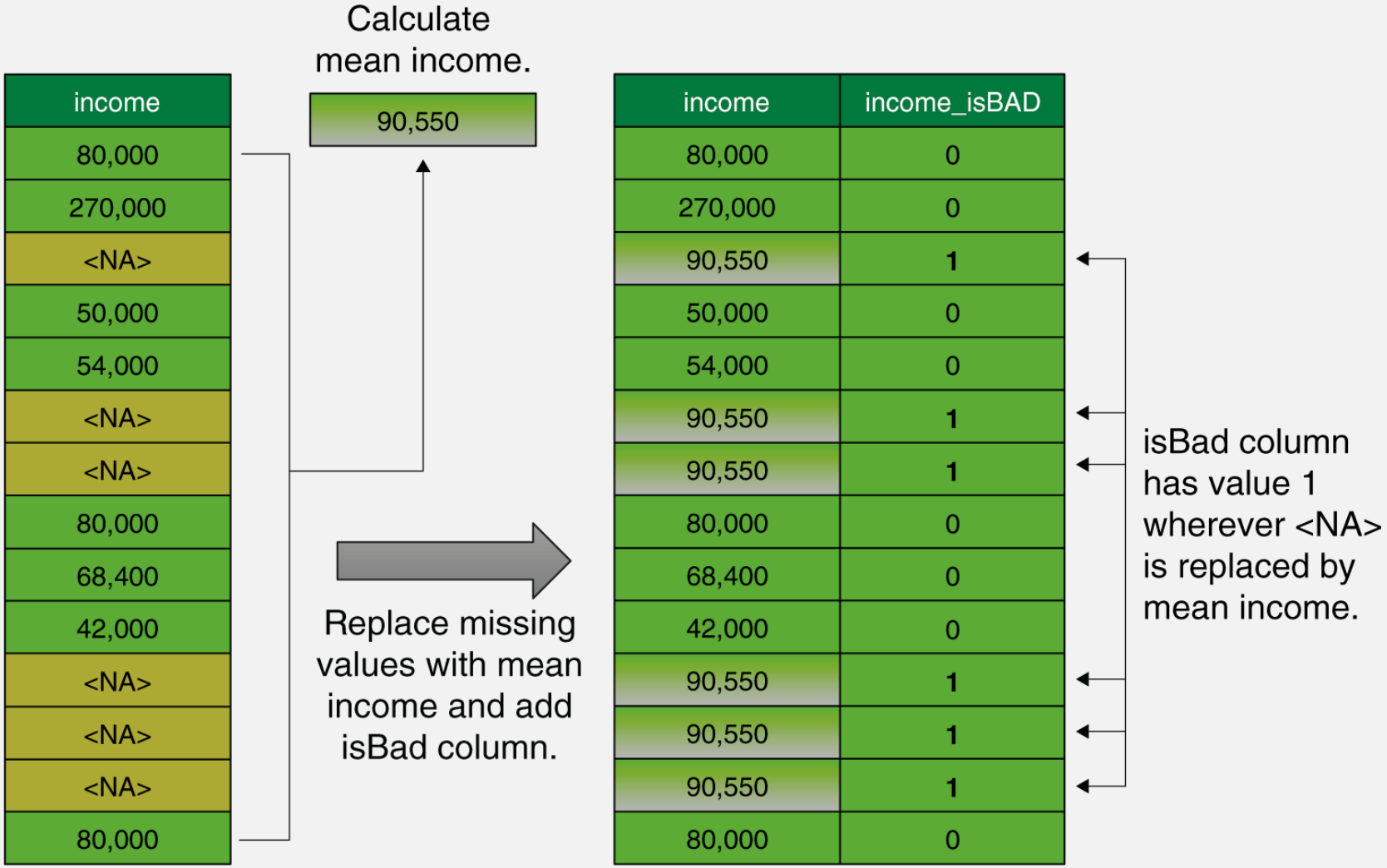
- One thing you can do is to convert the numeric data into categorical data

```
breaks <-c(0, 10000, 50000, 100000, 250000, 1000000)
Income.groups <- cut(custdata$income, breaks=breaks,
include.lowest=T)
summary(Income.groups)

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

Tracking original NAs with an extra categorical variable

```
missingIncome <- is.na(custdata$Income)

Income.fix <- ifelse(is.na(custdata$Income), 0,
custdata$Income)
```

Replacing missing values with the mean and adding an indicator column to track the altered values



Credit by Figure 4.6 *Practical Data Science with R* by Nina Zumel, John Mount

2.Invalid Values and Outliers

Examples of invalid values and outliers

- Negative values for income
- Outliers: customers of
  - age zero
  - an age greater than about 110

```
    custid          sex       is.employed         income                marital.stat health.ins                          housing.type recent.move
Min.   :   2068   F:440   Mode :logical   Min.   : -8700   Divorced/Separated:155   Mode :logical   Homeowner free and clear    :157   Mode :logical
1st Qu.: 345667   M:560   FALSE:73        1st Qu.: 14600   Married           :516   FALSE:159       Homeowner with mortgage/loan:412   FALSE:820
Median : 693403           TRUE :599       Median : 35000   Never Married     :233   TRUE :841       Occupied with no rent       : 11   TRUE :124
Mean   : 698500           NA's :328       Mean   : 53505   Widowed           : 96   NA's :0         Rented                      :364   NA's :56
3rd Qu.:1044606                           3rd Qu.: 67000                                             NA's                        : 56
Max.   :1414286                           Max.   :615000

 num.vehicles         age             state.of.res
Min.   : 0.000   Min.   :  0.0   California   :100
1st Qu.: 1.000   1st Qu.: 38.0   New York     : 71
Median : 2.000   Median : 50.0   Pennsylvania: 70
Mean   : 1.916   Mean   : 51.7   Texas        : 56
3rd Qu.: 2.000   3rd Qu.: 64.0   Michigan     : 52
Max.   : 6.000   Max.   :146.7   Ohio         : 51
NA's   :56.000                   (Other)      :600
```

Variation

- Visualising distributions
- visVariation.R

```
ggplot(data = diamonds) + geom_bar(mapping = aes(x = cut))
```

- What is the largest category?

```
library(tidyverse)
diamonds %>% count(cut)
```

Continuous variable

```
ggplot(data = diamonds) +
geom_histogram(mapping = aes(x = carat),
binwidth = 0.5)

diamonds %>% count(cut_width(carat, 0.5))
```

Overlay multiple histograms in the same plot

```
smaller <- diamonds %>% filter(carat < 3)

ggplot(data = smaller, mapping = aes(x =
carat, colour = cut)) +
geom_freqpoly(binwidth = 0.1)
```

Unusual values

- unusual.R

```
ggplot(diamonds) +
geom_histogram(mapping = aes(x = y),
binwidth = 0.5)
```

  - Any outlier?

Unusual values

- unusual.R

```
ggplot(diamonds) + geom_histogram(mapping =
aes(x = y), binwidth = 0.5) +
coord_cartesian(ylim = c(0, 50))
unusual <- diamonds %>% filter(y < 3 | y > 20)
%>% arrange(y)
unusual
```

Handle unusual values

1. drop them

```
diamonds2 <- diamonds %>% filter(between(y, 3, 20))
```

2. mutate, what is advantage?

```
diamonds2 <- diamonds %>% mutate(y = ifelse(y < 3 | y > 20,
NA, y))
ggplot(data = diamonds2, mapping = aes(x = x, y = y)) +
geom_point()
```

Covariation

- covairation.R
- How the price of a diamond varies with its quality?
- Which one is better way to answer the above question?

```
ggplot(data = diamonds, mapping = aes(x = price)) +
    geom_freqpoly(mapping = aes(colour = cut), binwidth = 500)
ggplot(data = diamonds, mapping = aes(x = price, y = ..density..)) +
    geom_freqpoly(mapping = aes(colour = cut), binwidth = 500)
```

3.Units

Looking at the data range of a variable

- Too narrow
  - wouldn't be a very good predictor if not vary much at all => standard deviation / the mean
  - Unit also matter

```
    custid           sex      is.employed          income                          marital.stat  health.ins                            housing.type  recent.move
 Min.   :   2068   F:440   Mode :logical   Min.    : -8700   Divorced/Separated:155   Mode :logical   Homeowner free and clear    :157   Mode :logical
 1st Qu.: 345667   M:560   FALSE:73        1st Qu.: 14600   Married           :516   FALSE:159       Homeowner with mortgage/loan:412   FALSE:820
 Median : 693403           TRUE :599       Median : 35000   Never Married     :233   TRUE :841       Occupied with no rent       : 11   TRUE :124
 Mean   : 698500           NA's :328       Mean   : 53505   Widowed           : 96   NA's :0         Rented                      :364   NA's :56
 3rd Qu.:1044606                           3rd Qu.: 67000                                            NA's                        : 56
 Max.   :1414286                           Max.   :615000

  num.vehicles         age               state.of.res
 Min.   : 0.000   Min.   :  0.0   California  :100
 1st Qu.: 1.000   1st Qu.: 38.0   New York    : 71
 Median : 2.000   Median : 50.0   Pennsylvania: 70
 Mean   : 1.916   Mean   : 51.7   Texas       : 56
 3rd Qu.: 2.000   3rd Qu.: 64.0   Michigan    : 52
 Max.   : 6.000   Max.   :146.7   Ohio        : 51
 NA's   :56.000                   (Other)     :600
```

Checking units can prevent inaccurate results later

```
Income = custdata$income/1000
summary(Income)
```

4.Data ranging => manage data

Managing data

1. Fixing data quality problems
2. Organizing your data for the modeling process

Data transformations

- make data easier to model—and easier to understand
  - converting continuous variables to discrete
  - normalizing variables
  - Log transformations

Converting continuous variables to discrete

- Their exact value matters less than whether they fall into a certain range
- 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

Converting continuous variables to discrete

- manageData.R

```
custdata$income.lt.20K <- custdata$income < 20000
summary(custdata$income.lt.20K)
```

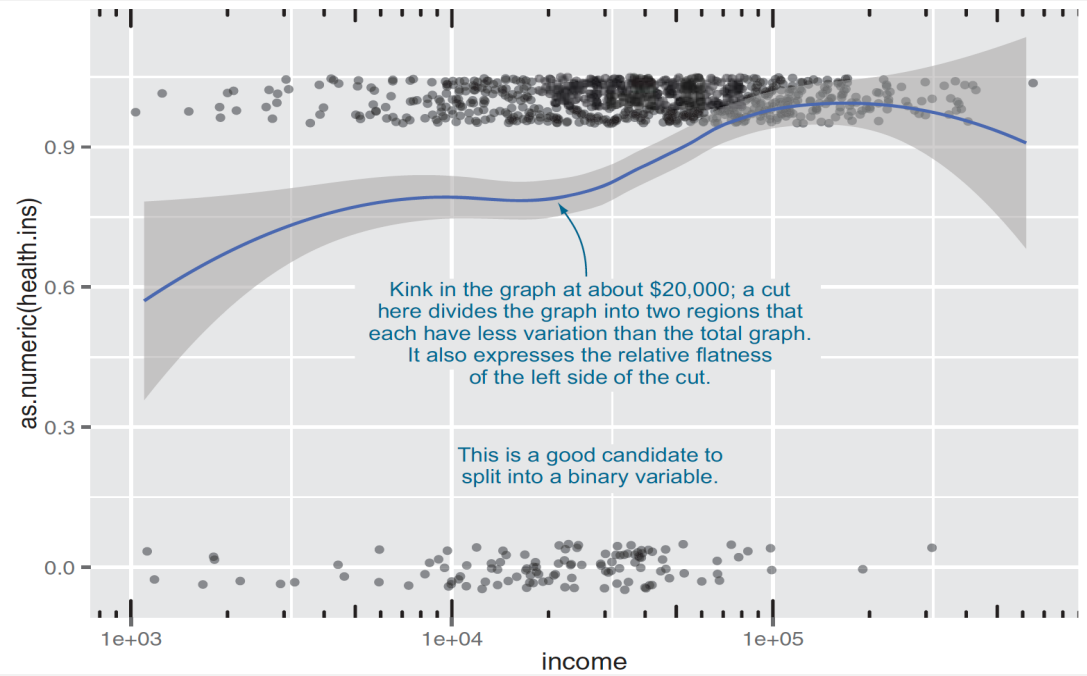


Figure 4.2, *Practical Data Science with R* by Nina Zumel, John Mount

Converting age into ranges

```
brks <- c(0, 25, 65, Inf)
custdata$age.range <- cut(custdata$age,
    breaks=brks, include.lowest=T)
summary(custdata$age.range)
```

Normalization and rescaling

- absolute quantities are less meaningful than relative ones
- Code1

```
summary(custdata$age)
meanage <- mean(custdata$age)
custdata$age.normalized <- custdata$age/meanage
```

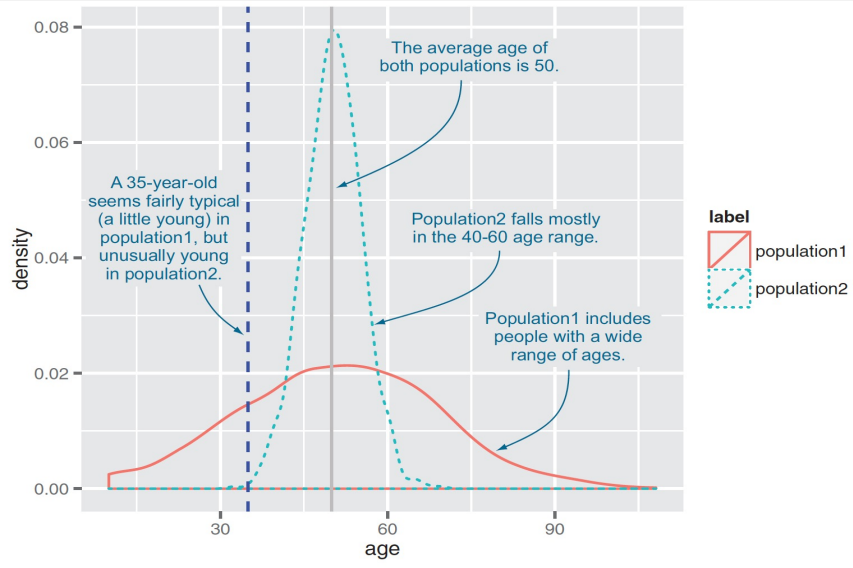


Figure4.3, *Practical Data Science with R* by Nina Zumel, John Mount

Normalization and rescaling

- absolute quantities are less meaningful than relative ones
- Code2

```
meanage <- mean(custdata$age)
stdage <- sd(custdata$age)
custdata$age.normalized <- (custdata$age-
meanage)/stdage
```

SD is most meaningful as a unit of distance

- the data is unimodal
- roughly symmetric around the mean.

Figure4.3, *Practical Data Science with R* by Nina Zumel, John Mount

Looking at the data range of a variable

- Too wide
  - Data that ranges over several orders of magnitude like this can be a problem for some modeling methods => might use *logarithmic* transformations

```
    custid          sex       is.employed         income                marital.stat health.ins                                housing.type recent.move
Min.   :   2068   F:440   Mode :logical   Min.   : -8700   Divorced/Separated:155   Mode :logical   Homeowner free and clear    :157   Mode :logical
1st Qu.: 345667   M:560   FALSE:73        1st Qu.: 14600   Married           :516   FALSE:159       Homeowner with mortgage/loan:412   FALSE:820
Median : 693403           TRUE :599       Median : 35000   Never Married     :233   TRUE :841       Occupied with no rent       : 11   TRUE :124
Mean   : 698500           NA's :328       Mean   : 53505   Widowed           : 96   NA's :0         Rented                      :364   NA's :56
3rd Qu.:1044606                           3rd Qu.: 67000                                            NA's                        : 56
Max.   :1414286                           Max.   :615000

 num.vehicles         age             state.of.res
Min.   : 0.000   Min.   :  0.0   California  :100
1st Qu.: 1.000   1st Qu.: 38.0   New York    : 71
Median : 2.000   Median : 50.0   Pennsylvania: 70
Mean   : 1.916   Mean   : 51.7   Texas       : 56
3rd Qu.: 2.000   3rd Qu.: 64.0   Michigan    : 52
Max.   : 6.000   Max.   :146.7   Ohio        : 51
NA's   :56.000                   (Other)     :600
```

Plotting a histogram

```
ggplot(custdata) + geom_histogram(aes(x=age),
    binwidth=5,fill="gray")
```

- *binwidth* : geom_histogram make bins of five-year intervals
  - Default: datarange/30
- *fill* : specifies the color of the histogram bars (default: black)

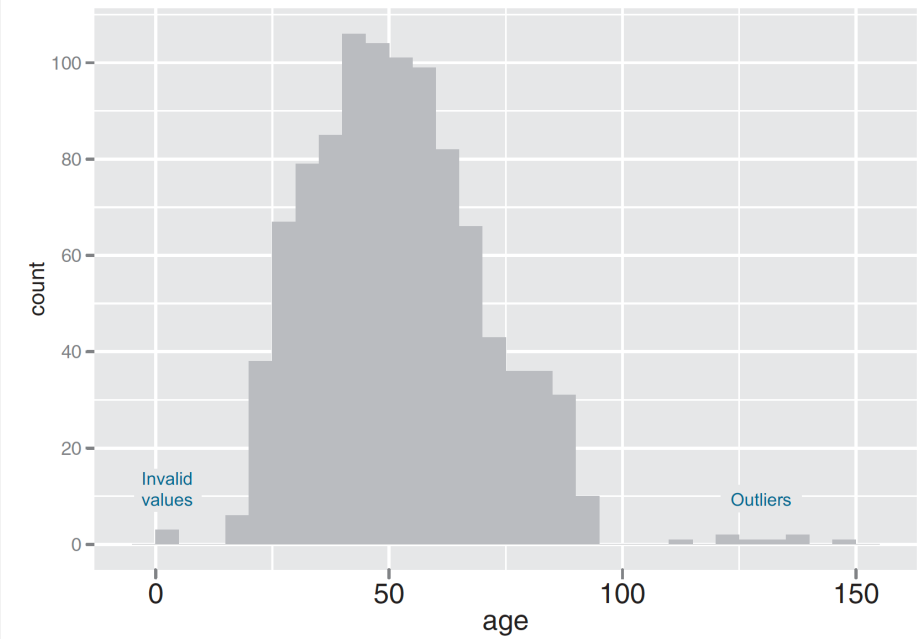


Figure3.3, *Practical Data Science with R* by Nina Zumel, John Mount

Density Plots

- When the data range
  - very wide
  - the mass of the distribution is heavily concentrated to one side (difficult to see the details of its shape)
  - non-negative

```
library(scales)
ggplot(custdata) +
geom_density(aes(x=income)) +
scale_x_continuous(labels=dollar)
```

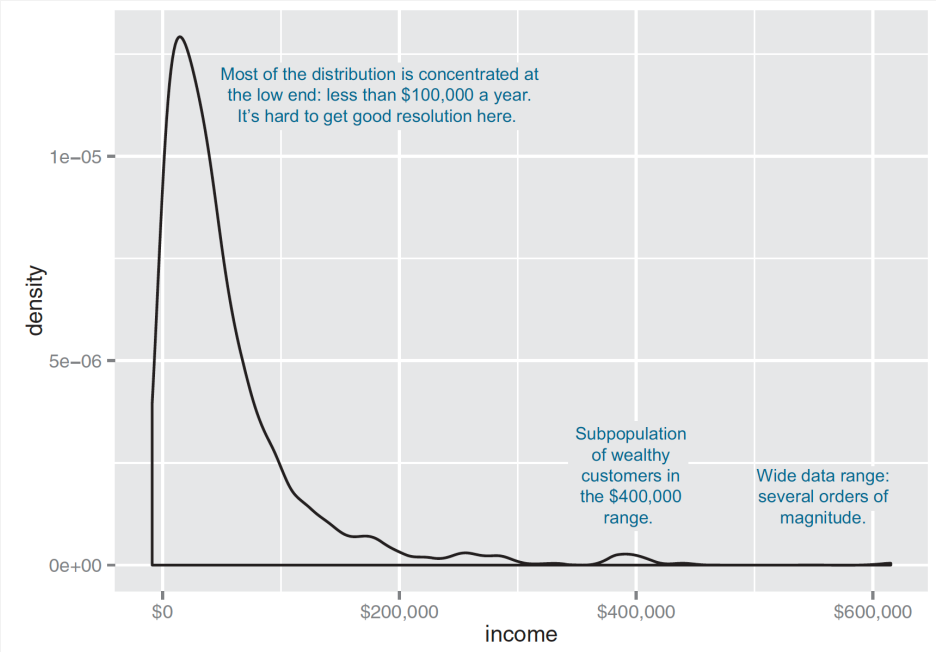
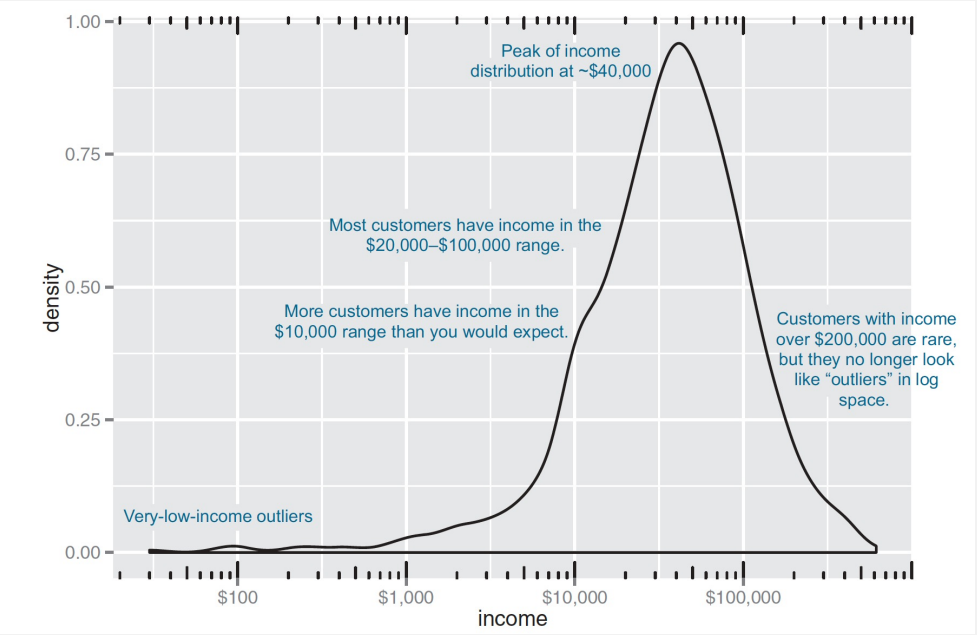


Figure3.4,*Practical Data Science with R* by Nina Zumel, John Mount

Density Plots

- bring out more detail is to plot the distribution on a *logarithmic* scale



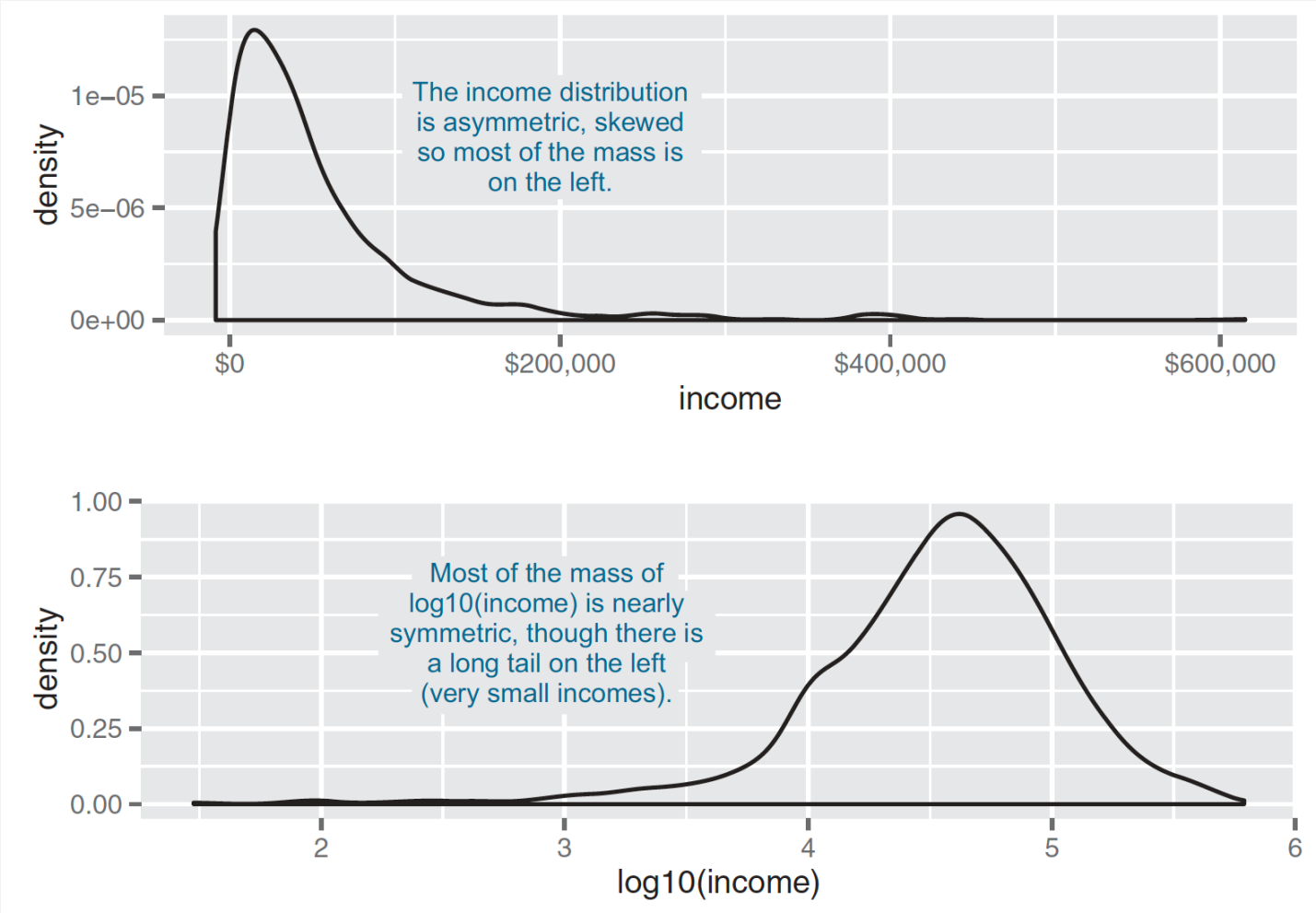

```
ggplot(custdata) +
geom_density(aes(x=income)) +
scale_x_log10(breaks=c(100,1000,10
000,100000), labels=dollar) +
annotation_logticks(sides="bt")
```

Warning messages:
1: In self$trans$transform(x) : NaNs produced
2: Transformation introduced infinite values in continuous x-axis
3: Removed 79 rows containing non-finite values (stat_density).

Log transformations for skewed and wide distributions

- data with values that range over several orders of magnitude
  - modeling techniques often have a difficult time with very wide data ranges
  - data often comes from multiplicative processes, so log units are in some sense more natural
- additive vs multiplicative:
  - a change to a restaurant menu increases patronage every night by 5%

A nearly lognormal distribution and its log



Fiure4.4, *Practical Data Science with R* by Nina Zumel, John Mount

If the data is negative?

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

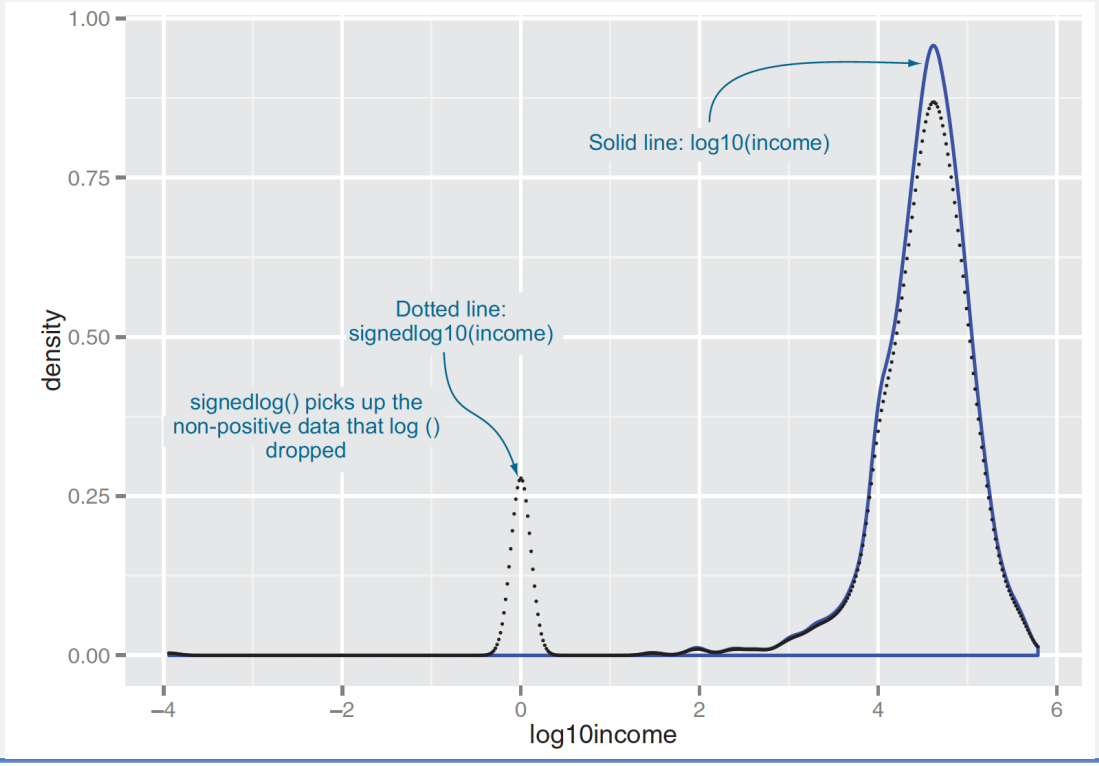


Figure4.5, *Practical Data Science with R* by Nina Zumel, John Mount

Intentional view of variables

- Explanatory variables : variables you can control
- Nuisance variables : important variables you can’t control
- Omitted variables : important variables you don’t know

Sampling bias

```
veryHighIncome <- subset(d, EarnedIncome+CapitalGains>=500000)

print(with(veryHighIncome,cor.test(EarnedIncome,CapitalGains,
method='spearman')))
```

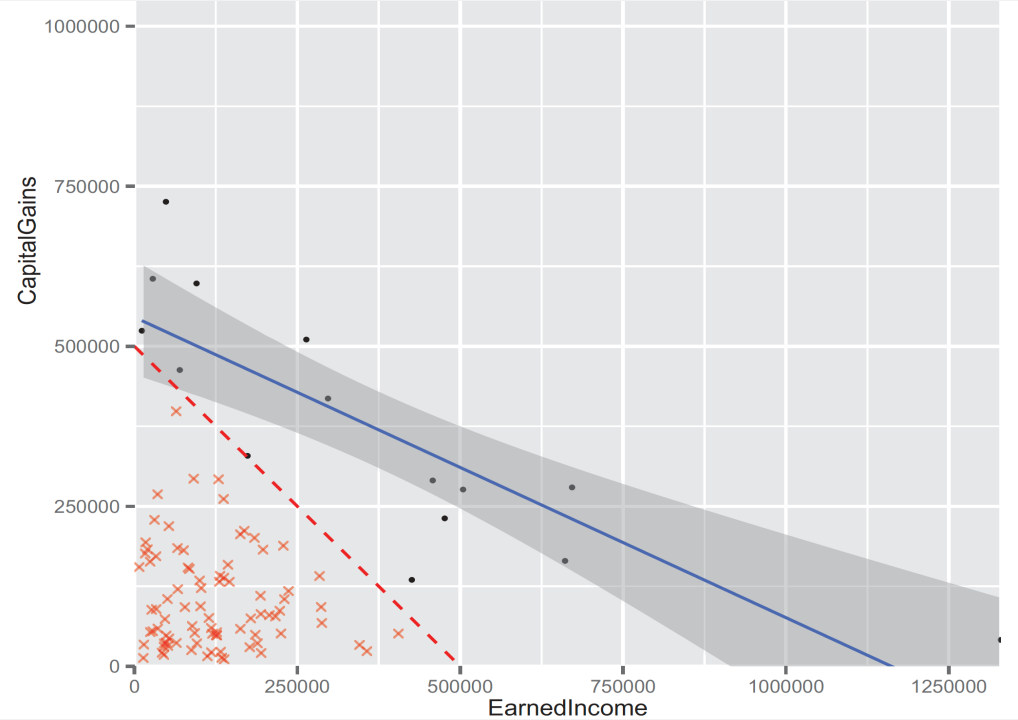


Figure B.9, *Practical Data Science with R* by Nina Zumel, John Mount

Sample Bias

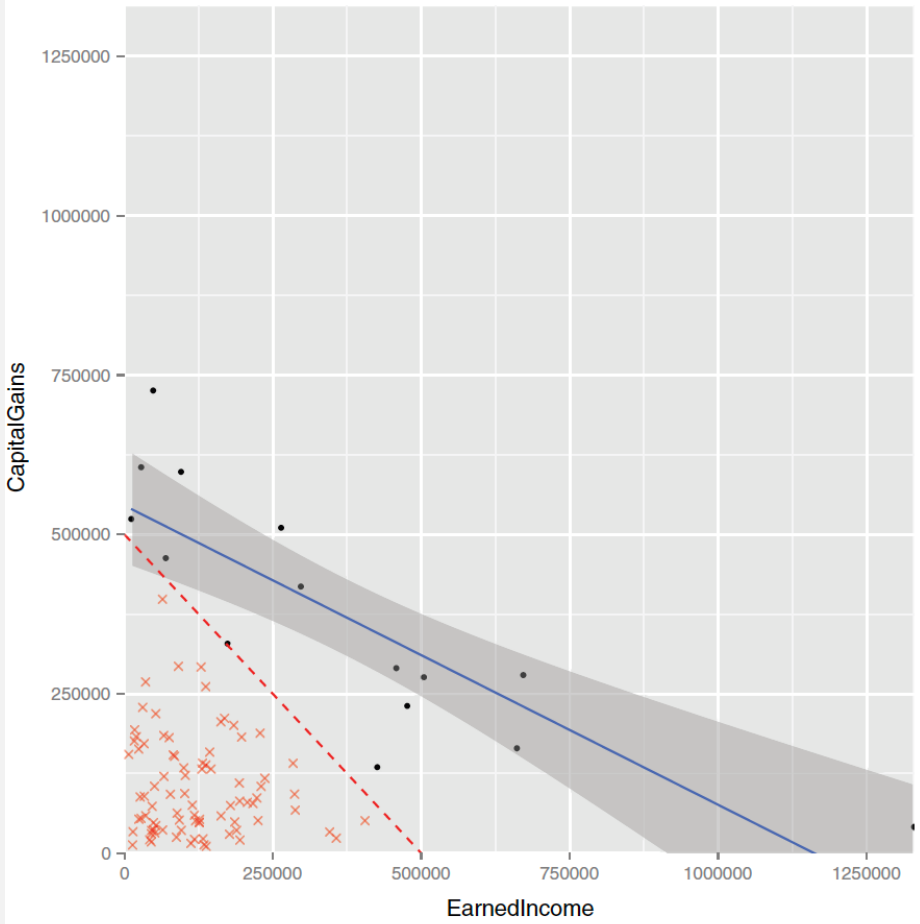


Figure B.9 Biased earned income vs. capital gains

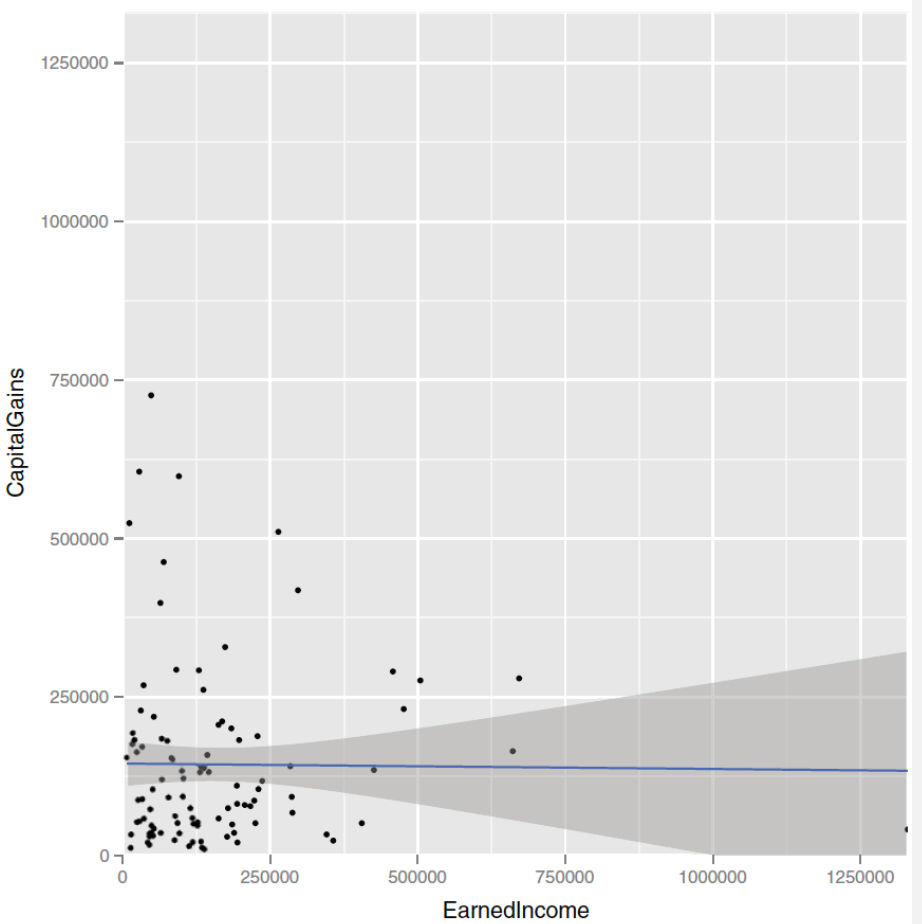


Figure B.8 Earned income versus capital gains

Summary

- 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.

How to handle imbalance data set?

Oversampling Techniques

- Random Oversampling
- Synthetic Minority Oversampling Technique (SMOTE)
- Borderline-SMOTE
- Borderline Oversampling with SVM
- Adaptive Synthetic Sampling (ADASYN)

Undersampling Techniques

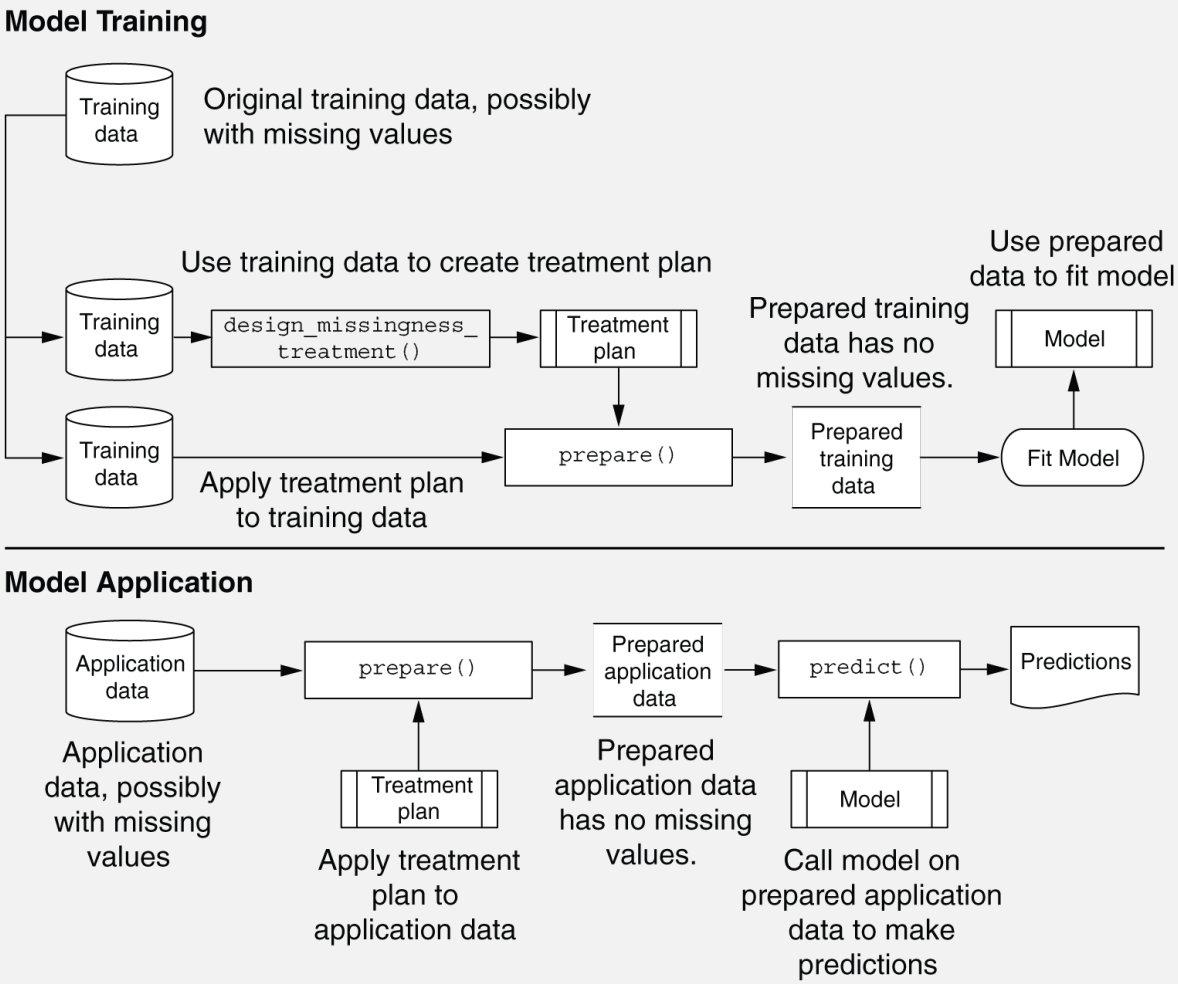
- Random Undersampling
- Condensed Nearest Neighbor Rule (CNN)
- Near Miss Undersampling
- Tomek Links Undersampling
- Edited Nearest Neighbors Rule (ENN)
- One-Sided Selection (OSS)
- Neighborhood Cleaning Rule (NCR)

vtreat package

vtreat package

- an automatic and repeatable process for dealing with missing data
- The vtreat process creates a *treatment plan* that records all the information needed to repeat the data treatment process
- use this treatment plan to “prepare” or treat your training data before you fit a model, and then again to treat new data before feeding it into the model.

Creating and applying a simple treatment plan



Credit by Figure 4.7, *Practical Data Science with R* by Nina Zumel, John Mount

Summary

- Take the time to examine your data before diving into the modeling.
- The **summary** command helps you spot issues with data range, units, data type, and missing or invalid values.
- Visualization additionally gives you a sense of data distribution and relationships among variables.
- Visualization is an iterative process and helps answer questions about the data. Time spent here is time not wasted during the modeling process.

Supervised & Unsupervised learning

supervised vs. unsupervised learning

- supervised learning
  - Input: a set of p features $X_1, X_2, \ldots, X_p$, measured on n observations, and a response Y also measured on those same n observations
  - Goal: predict Y using $X_1, X_2, \ldots, X_p$
- unsupervised learning
  - Input: $X_1, X_2, \ldots, X_p$ measured on n observations
  - Output: discover interesting things about the measurements

Mapping problems to machine learning tasks

- Predicting what customers might buy, based on past transactions
- Identifying fraudulent transactions
- Determining price elasticity (the rate at which a price increase will decrease sales, and vice versa) of various products or product classes
- Determining the best way to present product listings when a customer searches for an item

Mapping problems to machine learning tasks

- Customer segmentation: grouping customers with similar purchasing behavior
- AdWord valuation: how much the company should spend to buy certain AdWords on search engines
- Evaluating marketing campaigns
- Organizing new products into a product catalog

Solving classification problems

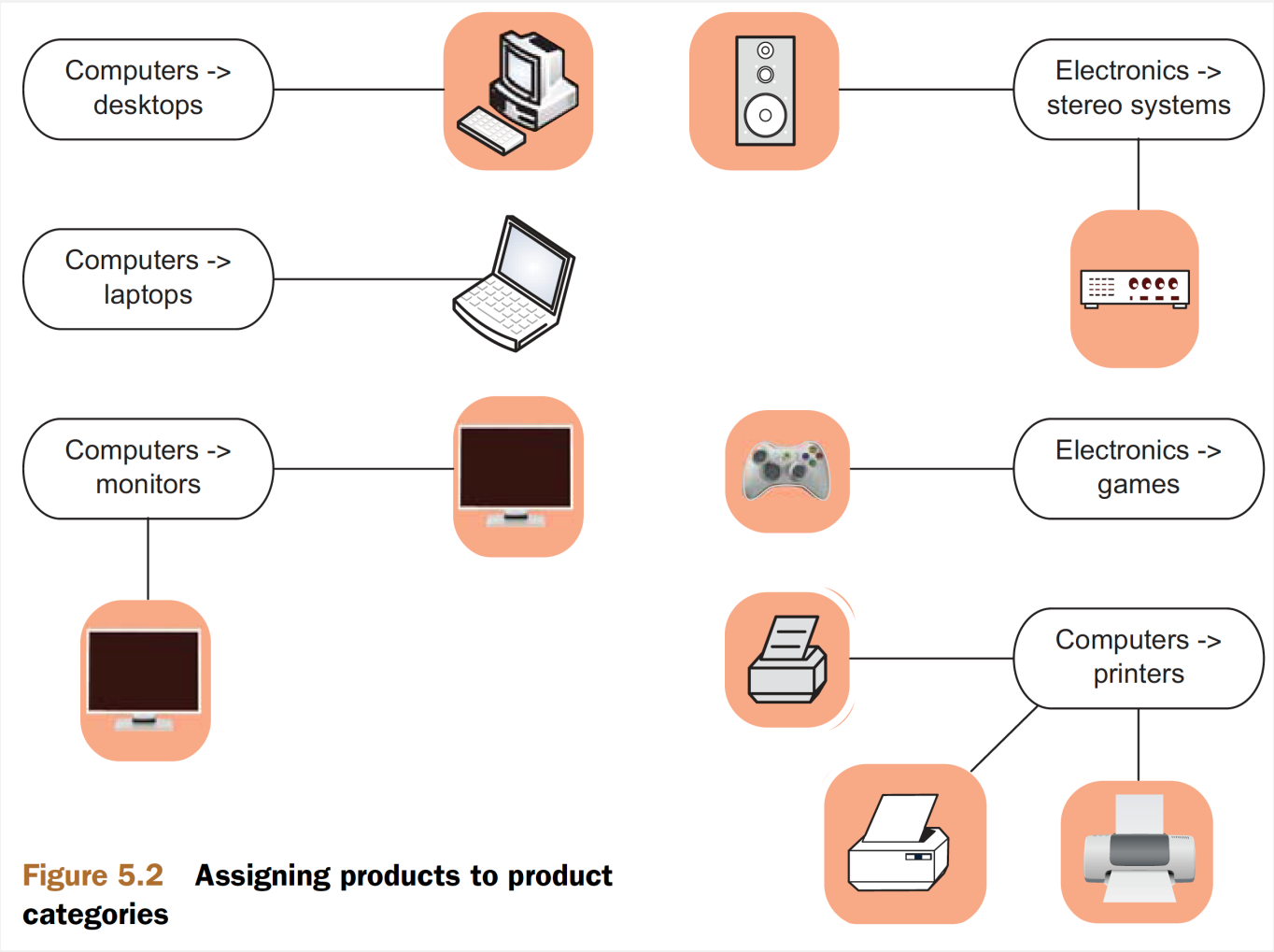


Figure 5.2 **Assigning products to product categories**

Solving classification problems

- Classification itself is an example of what is called *supervised learning*.
- Multicategory vs. two-category classification
  - using binary classifiers to solve multicategory problems : building one classifier for each category (a *1-vs-rest* classifier)
  - in most cases a suitable multiple-category implementation better than multiple binary classifiers
    - Use the package *mlogit* instead of the base method *glm* for logistic regression

Solving classification problems

- Some common classification methods
  - *Naive Bayes* : a good first attempt at solving the product categorization problem.
  - *Decision trees* : an important extension of decision trees - random forests
  - *Logistic regression* : estimate class probabilities (the probability that an object is in a given class) in addition to class assignments
  - *Support vector machines* : SVMs make fewer assumptions about variable distribution than do many other methods

Solving scoring problems

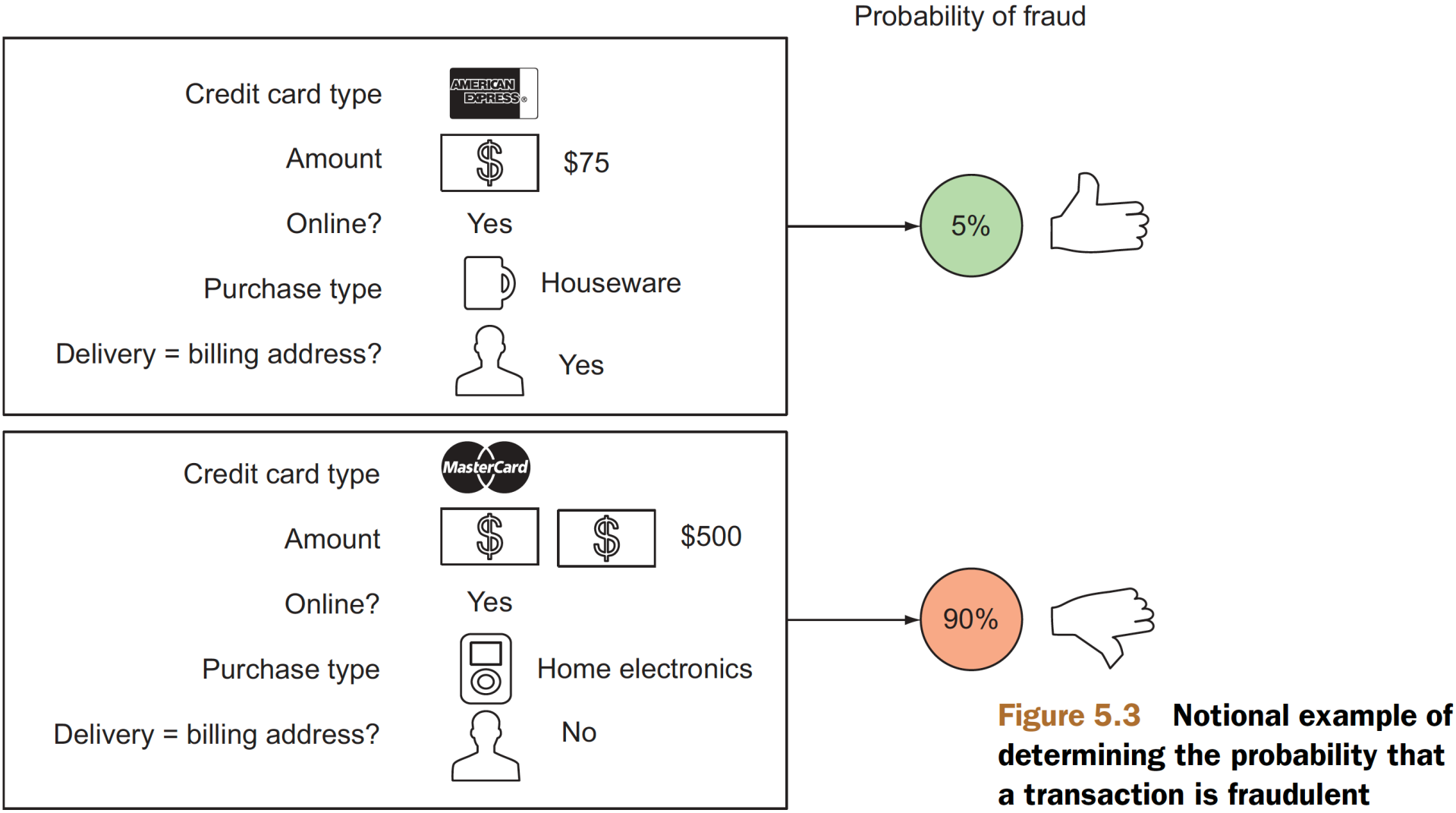


Figure 5.3 Notional example of determining the probability that a transaction is fraudulent

Solving scoring problems

- Linear regression
  - builds a model such that the predicted numerical output is a linear additive function of the inputs.
  - a good first model to try when trying to predict a numeric value
- Logistic regression
  - always predicts a value between 0 and 1
  - if what you want to estimate is the probability that a given transaction is fraudulent or legitimate.

Working without known targets

- *unsupervised learning*
  - there's not (yet) a specific outcome that you want to predict
  - discover similarities and relationships in the data. rather than predicting outputs based on inputs
- common clustering methods
  - *K-means* clustering
  - Nearest neighbor
  - *apriori* algorithm for finding association rules

When to use basic clustering

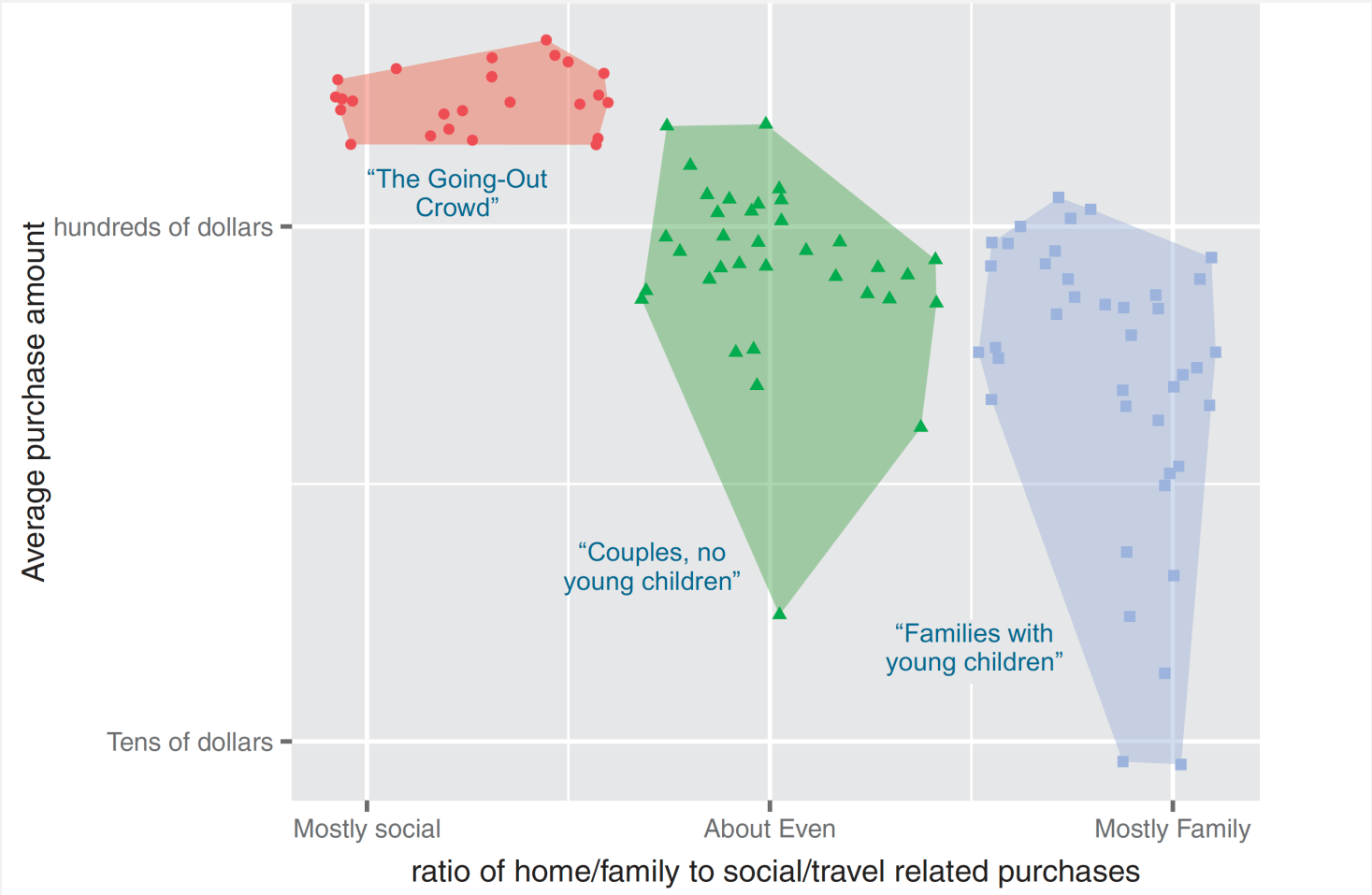


Figure 5.4 **Notional example of clustering your customers by purchase pattern and purchase amount**

When to use association rules

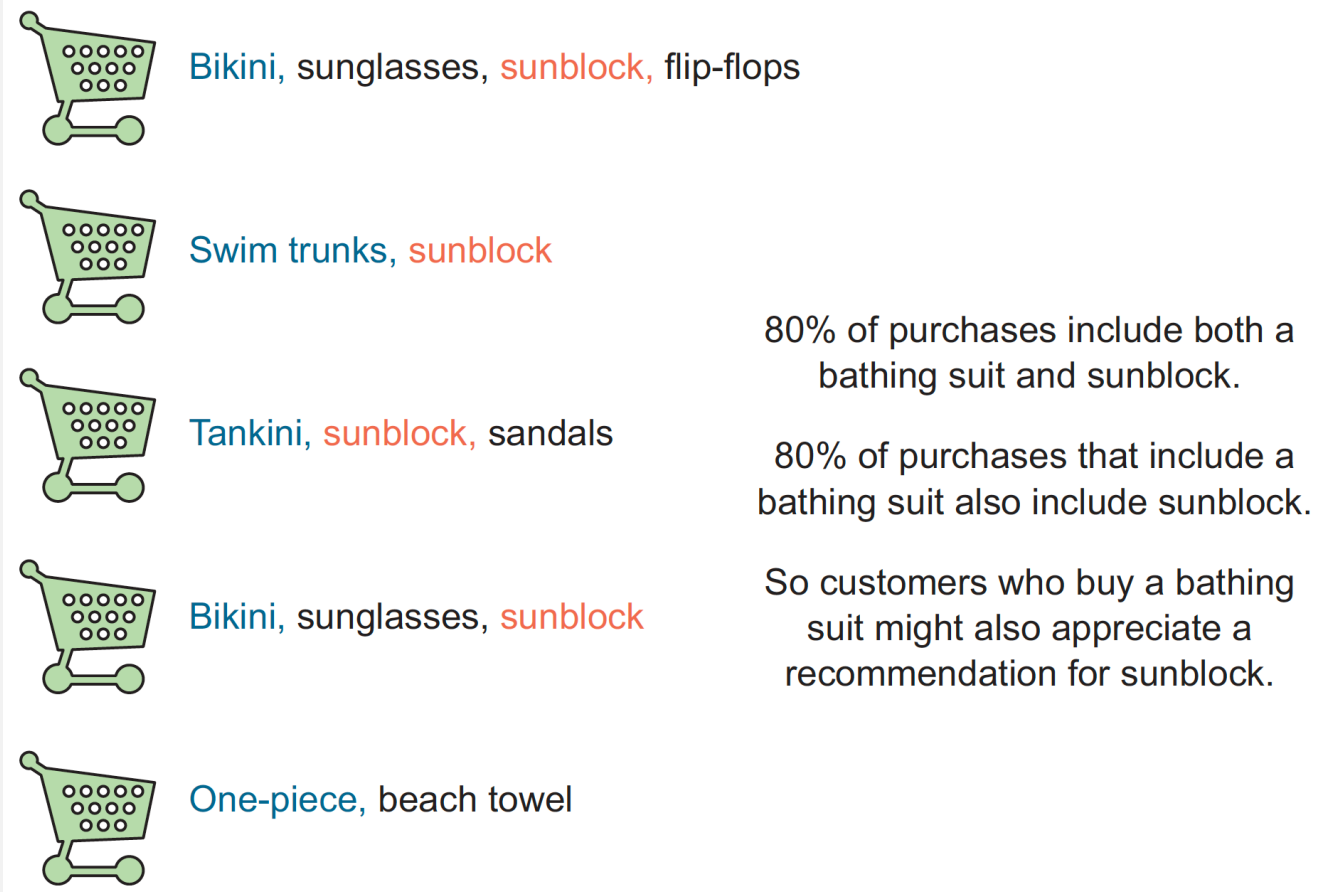


Figure 5.5 Notional example of finding purchase patterns in your data

When to use nearest neighbor methods

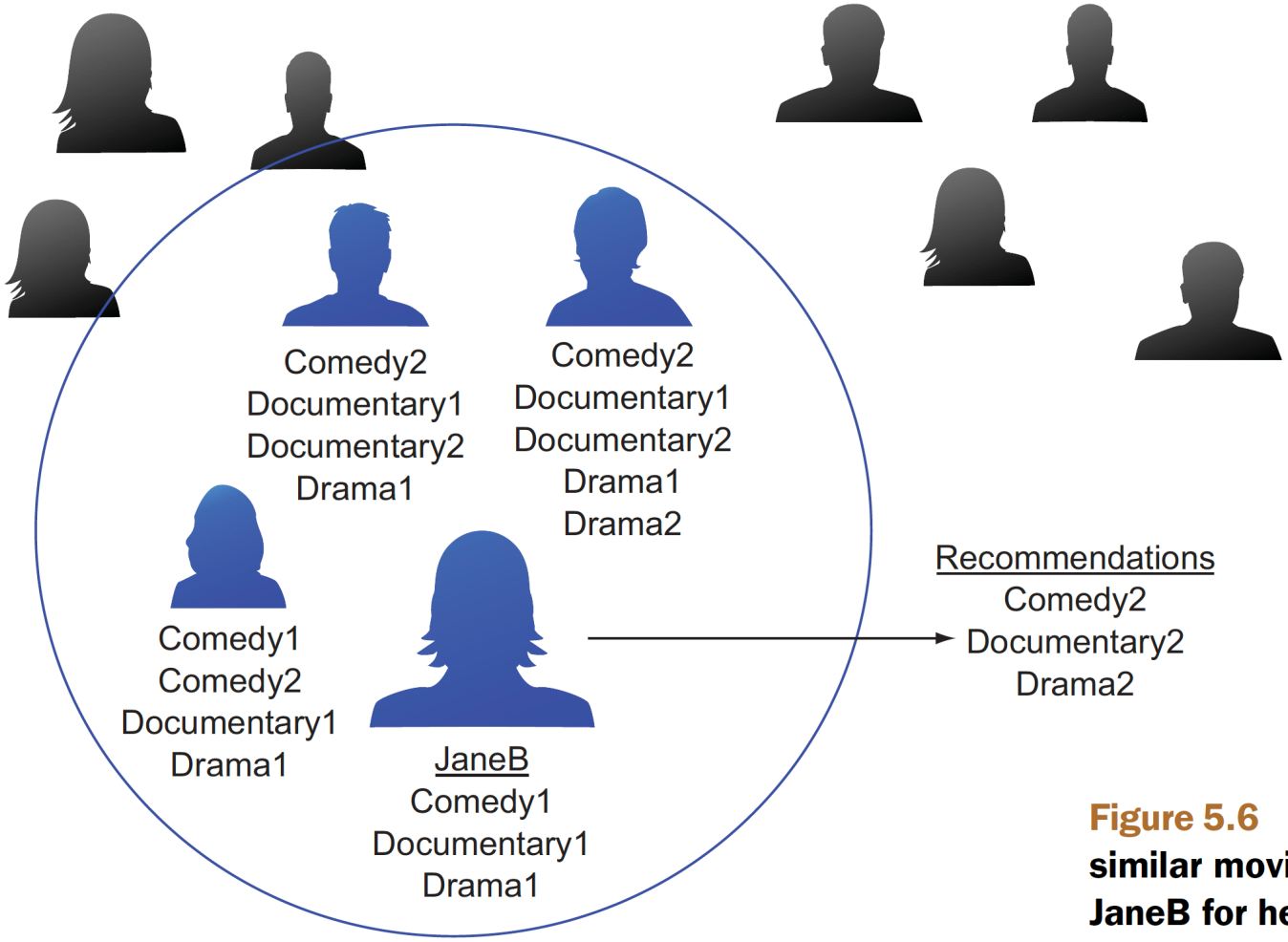


Figure 5.6 **Look to the customers with similar movie-watching patterns as JaneB for her movie recommendations.**

AITC
教育部人工智慧技術及應用人才培育計畫
Artificial Intelligence Talent Cultivation Program