Working with categorical data in R without losing your mind

Amelia McNamara @AmeliaMN
www.amelia.mn
University of St Thomas Department of Computer and Information Sciences
**Practical Data Science for Stats** - a PeerJ Collection

Data Science  |  Statistics  |  Scientific Computing and Simulation  |  Computer Education

Computational Science  |  Science and Medical Education  |  Computational Biology

Social Computing  |  Software Engineering  |  Anthropology  |  Human-Computer Interaction

Programming Languages  |  Visual Analytics  |  Graphics  |  Data Mining and Machine Learning

---

**Data organization in spreadsheets**

September 11, 2018  |  preprint

Data organization in spreadsheets

Karl W Broman, Kara H. Woo

https://doi.org/10.7287/peerj.preprints.3183v2

---

**Packaging data analytical work reproducibly using R (and friends)**

March 20, 2018  |  preprint

Packaging data analytical work reproducibly using R (and friends)

Ben Marwick, Carl Boettiger, Lincoln Mullen

https://doi.org/10.7287/peerj.preprints.3192v2
• Data organization in spreadsheets

• Packaging data analytical work reproducibly using R (and friends)

• Forecasting at scale

• How to share data for collaboration

• Opinionated analysis development

• Wrangling categorical data in R

• Lessons from between the white lines for isolated data scientists

• Teaching stats for data science

• Documenting and evaluating Data Science contributions in academic promotion in Departments of Statistics and Biostatistics

• Modeling offensive player movement in professional basketball

• Excuse me, do you have a moment to talk about version control?

• The democratization of data science education

• Extending R with C++: A Brief Introduction to Rcpp

• How R helps Airbnb make the most of its data

• Infrastructure and tools for teaching computing throughout the statistical curriculum

• Declutter your R workflow with tidy tools

https://peerj.com/collections/50-practicaldatascistats/
• Data organization in spreadsheets

• Packaging data analytical work reproducibly using R (and friends)

• Forecasting at scale

• How to share data for collaboration

• Opinionated analysis development

• Wrangling categorical data in R

• Lessons from between the white lines for isolated data scientists

• Teaching stats for data science

• Documenting and evaluating Data Science contributions in academic promotion in Departments of Statistics and Biostatistics

• Modeling offensive player movement in professional basketball

• Excuse me, do you have a moment to talk about version control?

• The democratization of data science education

• Extending R with C++: A Brief Introduction to Rcpp

• How R helps Airbnb make the most of its data

• Infrastructure and tools for teaching computing throughout the statistical curriculum

• Declutter your R workflow with tidy tools

https://peerj.com/collections/50-practicaldatascistats/
Places to Find Data

Finding the right data to answer your particular question is part of your responsibility for this assignment. Public data sets are available from hundreds of different websites, on virtually any topic. You might not be able to find the exact data that you want, but you should be able to find data that is relevant to your topic. You may also want to refine your research question so that it can be more clearly addressed by the data that you found. But be creative! Go find the data that you want!

Below is a list of places to get started, but this list should be considered grossly non-exhaustive:

- Data is Plural tiny letter and associated spreadsheet
- FiveThirtyEight data archive
- Data.gov 186,000+ datasets!
- Social Explorer is a great interface to Census and American Community Survey data (much more user-friendly than the official government sites). Smith has a site license, but you may need to create an account.
- Gallup Analytics (available through the library databases)
- Data and Story Library (DASL). (This, and more ideas from Robin Lock.)
- Jo Hardin at Pomona College has a nice list of data sources on her website.
- U.S. Bureau of Labor Statistics
- U.S. Census Bureau
- Gapminder, data about the world.
- IRE and NICAR are good resources for the types of data journalists care about. For example, Energy data sources and Chrys Wu's resource page.
- Nathan Yau's (old) guide to finding data on the internet

Keep the following in mind as you select your topic and dataset:

- You need to have enough data to make meaningful inferences. There is no magic number of individuals required for all
The GSS is now accepting proposals for new items and modules on the 2020 General Social Survey. Proposals are due by January 30th.
**factors**

R’s representation of categorical data. Consists of:
1. A set of **values**
2. An ordered set of **valid levels**

```r
eyes <- factor(x = c("blue", "green", "green"),
               levels = c("blue", "brown", "green"))
eyes
## [1] blue  green green
## Levels: blue brown green
```
I'm just coming off of final student projects, so I'm thinking about things that might be useful to new data practitioners in R. Some ideas:

1. A comparison of different ways to express the same action using different syntaxes. Probably I would focus on subsetting in different ways (rows/columns). For example, `mtcars %>% select(wt)` versus `mtcars[,6]` versus `mtcars[,c("wt") or mtcars %>% filter(mpg>30) versus mtcars[mtcars$mpg>30,]` Other than subsetting, I could also look at ways to create new variables, e.g. `mtcars %>% mutate(ratio = gear/carb)` versus `mtcars$ratio <- mtcars$gear/mtcars$carb` This one might be too simplistic and/or too related to #8.

2. Explanation of factors and how to recode them. I might need to talk to @hadley about best practices here, because my current solutions are a bit hacky and I often get warning messages. There are a few different factor issues I/my students often run into.

   a. Starting with the simplest: you want to change the formatting of the factor labels so they all start with a capital letter. When doing this, it is so easy to accidentally ruin your data, so you need a little EDA workflow: look at the `summary()` of the factor and note the numbers in each category, then try your level changes, then look at the `summary()` again.

   b. Another problem is reordering factor levels—maybe because you want ggplot2 to show them in a particular order, or because there is some inherent order to your levels. Again, I often do

   ```r
   SummaryStats <- SummaryStats %>% mutate(Treatment = factor(Treatment, levels=c("Control", "E25", "E50", "E100")))
   ```

   and ruin everything before I remember it's actually

   ```r
   summarystats <- summarystats %>% mutate(treatment = factor(treatment, levels=levels(treatment)[c(3,4,2)]))
   ```

   b. Even easier to mess up is when you have a categorical variable with 10+ categories and want to condense down to 3-4. Again, this is where my hack often runs into errors.
variable names with spaces... and so it begins

53 lines (38 sloc)  1.42 KB

```
1  ```
2  title: "Working with factor variables in R"
3  author: "Amelia McNamara"
4  date: "June 7, 2016"
5  output: html_document
6  ```
7  ```
8  ```
9  ```
10  ```
11  ```
12  ```
13  ```
14  ```
15  ```
16  ```
17  ```
18  ```
```

## Loading the data
We have several options for how to get this data. We could download it in SPSS or Stata formats and use the foreign package to read it in.
```
`
```
Wrangling categorical data in R

Me @askdrstats

Amelia McNamara, Nicholas J Horton

August 30, 2017

Data wrangling is a critical foundation of data science, and wrangling of categorical data is an important component of this process. However,
Article

Wrangling Categorical Data in R

Amelia McNamara & Nicholas J. Horton

Pages 97-104 | Received 01 May 2017, Accepted author version posted online: 27 Jul 2017, Published online: 27 Jul 2017

Download citation https://doi.org/10.1080/00031305.2017.1356375

Amelia McNamara\textsuperscript{a} http://orcid.org/0000-0003-4916-2433 & Nicholas J. Horton\textsuperscript{b} http://orcid.org/0000-0003-3332-4311

\textsuperscript{a} Program in Statistical and Data Sciences, Smith College, Northampton, MA

\textsuperscript{b} Department of Mathematics and Statistics, Amherst College, Amherst, MA

CONTACT Amelia McNamara amcnamara@smith.edu Program in Statistical and Data Sciences,
> x <- c(20, 20, 10, 40, 10)
> x
[1] 20 20 10 40 10
> xf <- factor(x)
> xf
[1] 20 20 10 40 10
Levels: 10 20 40
> as.numeric(xf)
[1] 2 2 1 3 1
>
> factor("a", levels=c("b"))
[1] <NA>
Levels: b
>  |

Data Input

Description

Reads a file in table format and creates a data frame from it, with cases corresponding to lines and variables to fields in the file.

Usage

```r
read.table(file, header = FALSE, sep = " ", quote = "\" ",
  dec = ".", numerals = c("allow.loss", "warn.loss", "no.loss"),
  row.names, col.names, as.is = !stringsAsFactors,
  na.strings = "NA", colClasses = NA, nrow = -1,
  skip = 0, check.names = TRUE, fill = !blank.lines.skip,
  strip.white = FALSE, blank.lines.skip = TRUE,
  comment.char = "#",
  allowEscapes = FALSE, flush = FALSE,
  stringsAsFactors = default.stringsAsFactors(),
  fileEncoding = "", encoding = "unknown", text, skipNu1 = FALSE)
```

```r
read.csv(file, header = TRUE, sep = ",", quote = "\" ",
  dec = ".", fill = TRUE, comment.char = "", ...)
```

```r
read.csv2(file, header = TRUE, sep = ";", quote = "\" ",
  dec = ",", fill = TRUE, comment.char = "", ...)
```

```r
read.delim(file, header = TRUE, sep = "\t", quote = "\" ",
  dec = ".", fill = TRUE, comment.char = "", ...)
```
stringsAsFactors: An unauthorized biography

Roger Peng ️ 2015/07/24

Recently, I was listening in on the conversation of some colleagues who were discussing a bug in their R code. The bug was ultimately traced back to the well-known phenomenon that functions like `read.table()` and `read.csv()` in R convert columns that are detected to be character/strings to be factor variables. This lead to the spontaneous outcry from one colleague of

Why does `stringsAsFactors` not default to `FALSE`???

The argument `stringsAsFactors` is an argument to the `data.frame()` function in R. It is a logical that indicates whether strings in a data frame should be treated as factor variables or as just plain strings. The argument also appears in `read.table()` and related functions because of the role these functions play in reading in table data and converting them to data frames. By default, `stringsAsFactors` is set to `TRUE`.

This argument dates back to May 20, 2006 when it was originally introduced into R as the `charToFactor` argument to `data.frame()`. Soon afterwards, on May 24, 2006, it was changed to `stringsAsFactors` to be compatible with S-PLUS by request from Bill Dunlap.

Most people I talk to today who use R are completely befuddled by the fact that `stringsAsFactors` is set to `TRUE` by default. First of all, it should be noted that before the `stringsAsFactors` argument even existed, the behavior of R was to coerce all character strings to be factors in a data frame. If you didn't want this behavior, you had to manually coerce each column to be character.

So here's the story:

In the old days, when R was primarily being used by statisticians and statistical types, this setting strings to be...
@kwbroman @hspter @_inundata @sgrifter @hadleywickham I'm ready for the mixer
Read a delimited file (including csv & tsv) into a tibble

Description

read_csv() and read_tsv() are special cases of the general read_delim(). They're useful for reading the most common types of flat file data, comma separated values and tab separated values, respectively. read_csv2() uses ; for separators, instead of ,. This is common in European countries which use , as the decimal separator.

Usage

read_delim(file, delim, quote = "\"", escape_backslash = FALSE,
    escape_double = TRUE, col_names = TRUE, col_types = NULL,
    locale = default_locale(), na = c("" , "NA"), quoted_na = TRUE,
    comment = "", trim_ws = FALSE, skip = 0, n_max = Inf,
    guess_max = min(1000, n_max), progress = show_progress())

read_csv(file, col_names = TRUE, col_types = NULL,
    locale = default_locale(), na = c("" , "NA"), quoted_na = TRUE,
    quote = "\"", comment = "", trim_ws = TRUE, skip = 0, n_max = Inf,
    guess_max = min(1000, n_max), progress = show_progress())

read_csv2(file, col_names = TRUE, col_types = NULL,
    locale = default_locale(), na = c("" , "NA"), quoted_na = TRUE,
    quote = "\"", comment = "", trim_ws = TRUE, skip = 0, n_max = Inf,
    guess_max = min(1000, n_max), progress = show_progress())

read_tsv(file, col_names = TRUE, col_types = NULL,
    locale = default_locale(), na = c("" , "NA"), quoted_na = TRUE,
    quote = "\"", comment = "", trim_ws = TRUE, skip = 0, n_max = Inf,
    guess_max = min(1000, n_max), progress = show_progress())
But sometimes, you still need factors…

In particular, for modeling (changing reference levels, etc) and plotting (reordering elements)
races08$race <- factor(races08$race)
levels(races08$race) <- c("Hispanic", "More than one", "Refused", "American Indian "Asian", "Black or African-American", "Native Hawaiian or other Pacific Islander", "White")

kidGroups$neg$Response <- factor(kidGroups$neg$Response,
levels=levels(kidGroups$neg$Response)[c(2,3,1)])
> summary(GSS$BaseOpinionOfIncome)
  Above average    Average Below average Don't know Far above average
        483       1118       666        21           65
Far below average No answer NA's
         179          6           2

> GSS$BaseOpinionOfIncome <-
+  factor(GSS$BaseOpinionOfIncome,
+        levels = c("Far above average", "Above average", "Average ", "Below Average",
+                    "Far below average", "Don't know", "No answer"))

> summary(GSS$BaseOpinionOfIncome)
Far above average Above average Average Below Average Far below average
        65         483       0        0            179
Don't know No answer NA's
        21          6       1786

>
```r
> badApproach <- GSS$OpinionOfIncome
> summary(badApproach)

<table>
<thead>
<tr>
<th></th>
<th>Above average</th>
<th>Average</th>
<th>Below average</th>
<th>Don't know</th>
<th>Far above average</th>
</tr>
</thead>
<tbody>
<tr>
<td>Far below average</td>
<td>483</td>
<td>1118</td>
<td>666</td>
<td>21</td>
<td>65</td>
</tr>
<tr>
<td>Far below average</td>
<td>179</td>
<td>6</td>
<td>NA's</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

> levels(badApproach) <- c("Far above average", "Above average", "Average", "Below Average", "Far below average", "Don't know", "No answer")
> summary(badApproach)

<table>
<thead>
<tr>
<th></th>
<th>Far above average</th>
<th>Above average</th>
<th>Average</th>
<th>Below Average</th>
<th>Far below average</th>
<th>Don’t know</th>
</tr>
</thead>
<tbody>
<tr>
<td>Far above average</td>
<td>483</td>
<td>1118</td>
<td>666</td>
<td>21</td>
<td>65</td>
<td></td>
</tr>
<tr>
<td>Far below average</td>
<td>179</td>
<td>6</td>
<td>NA’s</td>
<td>2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

```r
badApproach <- GSS$OpinionOfIncome

summary(badApproach)

<table>
<thead>
<tr>
<th>Above average</th>
<th>Average</th>
<th>Below average</th>
<th>Don't know</th>
<th>Far above average</th>
</tr>
</thead>
<tbody>
<tr>
<td>483</td>
<td>1118</td>
<td>666</td>
<td>21</td>
<td></td>
</tr>
<tr>
<td>Far below average</td>
<td>No answer</td>
<td>NA's</td>
<td></td>
<td></td>
</tr>
<tr>
<td>179</td>
<td>6</td>
<td>2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

levels(badApproach) <- levels(badApproach)[c(5,1:3,6,4,7)]

summary(badApproach)

<table>
<thead>
<tr>
<th>Far above average</th>
<th>Above average</th>
<th>Average</th>
<th>Below average</th>
<th>Don't know</th>
<th>Far below average</th>
</tr>
</thead>
<tbody>
<tr>
<td>483</td>
<td>1118</td>
<td>666</td>
<td>21</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Don't know</td>
<td>No answer</td>
<td>NA's</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>179</td>
<td>6</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

tools for working with categorical variables (factors)

https://forcats.tidyverse.org

1 commit
3 branches
4 releases
23 contributors

hadley Initial commit

Latest commit 44d039fc9e on Aug 8, 2016

- .Rbuildignore
- .gitignore
- DESCRIPTION
- NAMESPACE
- forcats.Rproj

Initial commit
Initial commit
Initial commit
Initial commit
Initial commit

3 years ago
3 years ago
3 years ago
3 years ago
3 years ago

https://github.com/tidyverse/forcats
Level manipulation functions
Values change to match levels

- `fct_recode()`  Relabel levels "by hand"
- `fct_relevel()` Reorder levels "by hand"
- `fct_reorder()` Reorder levels by another variable
- `fctCollapse()` Collapse levels "by hand"
- `fct_lump()` Lump levels with small counts together
- `fct_other()` Replace levels with "Other"
**SUMMARY STATISTICS:**

- **one continuous variable:**
  - mean ~ mpg, data = mtcars
  - mean(mtcars$mpg)

- **one categorical variable:**
  - tally ~ cyl, data = mtcars
  - table(mtcars$cyl)

- **two categorical variables:**
  - tally(cyl ~ am, data = mtcars)
  - table(mtcars$cyl, mtcars$am)

- **one continuous, one categorical:**
  - mean(mpg ~ cyl, data = mtcars)
  - mean(mtcars$mpg[mtcars$cyl==4])
  - mean(mtcars$mpg[mtcars$cyl==6])
  - mean(mtcars$mpg[mtcars$cyl==8])

**PLOTTING:**

- **one continuous variable:**
  - hist(mtcars$disp)
  - boxplot(mtcars$disp)

- **one categorical variable:**
  - barplot(table(mtcars$cyl))

- **two continuous variables:**
  - plot(mtcars$disp, mtcars$mpg)

- **two categorical variables:**
  - mosaicplot(table(mtcars$am, mtcars$cyl))
  - boxplot(mtcars$disp | mtcars$cyl)

- **one continuous, one categorical:**
  - hist(mtcars$disp | mtcars$cyl)
  - boxplot(mtcars$disp | mtcars$cyl)

**WRANGLING:**

- **subsetting:**
  - mtcars[mtcars$mpg>30,]

- **making a new variable:**
  - mtcars$efficient[mtcars$mpg>30] <- TRUE
  - mtcars$efficient[mtcars$mpg<30] <- FALSE

---

**Dollar sign syntax**

```
goal(data$x, data$y)
```

### SUMMARY STATISTICS:

- one continuous variable:
  - mean(mtcars$mpg)
  - mean(mtcars$cyl)
  - mean(mtcars$am)

- one categorical variable:
  - table(mtcars$cyl)

- two categorical variables:
  - table(mtcars$cyl, mtcars$am)

### PLOTTING:

- one continuous variable:
  - hist(mtcars$disp)
  - boxplot(mtcars$disp)

- one categorical variable:
  - barplot(table(mtcars$cyl))

- two continuous variables:
  - plot(mtcars$disp, mtcars$mpg)

- two categorical variables:
  - mosaicplot(table(mtcars$am, mtcars$cyl))

### WRANGLING:

- subsetting:
  - mtcars[mtcars$mpg>30,]

- making a new variable:
  - mtcars$efficient[mtcars$mpg>30] <- TRUE
  - mtcars$efficient[mtcars$mpg<30] <- FALSE

---

**Formula syntax**

```
goal(y~x|z, data=data, group=mw)
```

### SUMMARY STATISTICS:

- one continuous variable:
  - mosaic::mean(~ mpg, data=mtcars)
  - mosaic::mean(mpg ~ cyl, data=mtcars)
  - mosaic::mean(mpg ~ cyl, data=mtcars)

- one categorical variable:
  - mosaic::tally(cyl ~ am, data=mtcars)
  - mosaic::tally(cyl ~ am, data=mtcars)

- two categorical variables:
  - mosaic::tally(cyl ~ am, data=mtcars)
  - mosaic::tally(cyl ~ am, data=mtcars)

- one continuous, one categorical:
  - mosaic::mean(mpg ~ cyl, data=mtcars)
  - mosaic::mean(mpg ~ cyl, data=mtcars)

### PLOTTING:

- one continuous variable:
  - lattice::histogram(~ disp, data=mtcars)
  - lattice::bwplot(~ disp, data=mtcars)

- one categorical variable:
  - lattice::bargraph(~ cyl, data=mtcars)

- two continuous variables:
  - lattice::xyplot(mpg~disp, data=mtcars)

- two categorical variables:
  - lattice::bargraph(~ am, data=mtcars, group=cyl)

- one continuous, one categorical:
  - lattice::histogram(~ disp | cyl, data=mtcars)
  - lattice::bwplot(cyl~disp, data=mtcars)

### WRANGLING:

- subsetting:
  - mtcars %>% dplyr::filter(mpg>30)

- making a new variable:
  - mtcars %>% dplyr::mutate(efficient = if_else(mpg>30, TRUE, FALSE))

---

**Tidyverse syntax**

```
data %>% goal(x)
```

### SUMMARY STATISTICS:

- one continuous variable:
  - mtcars %>% dplyr::summarize(mean(mpg))

- one categorical variable:
  - mtcars %>% dplyr::group_by(cyl) %>% dplyr::summarize(n())

- two categorical variables:
  - mtcars %>% dplyr::group_by(cyl, am) %>% dplyr::summarize(n())

- one continuous, one categorical:
  - mtcars %>% dplyr::group_by(cyl) %>% dplyr::summarize(mean(mpg))

### PLOTTING:

- one continuous variable:
  - ggplot2::qplot(x=mpg, data=mtcars, geom = "histogram")
  - ggplot2::qplot(y=disp, x=1, data=mtcars, geom="boxplot")

- one categorical variable:
  - ggplot2::qplot(x=cyl, data=mtcars, geom="bar")

- two continuous variables:
  - ggplot2::qplot(x=disp, y=mpg, data=mtcars, geom="point")

- two categorical variables:
  - ggplot2::qplot(x=factor(cyl), data=mtcars, geom="bar") + facet_grid(.~am)

- one continuous, one categorical:
  - ggplot2::qplot(y=disp, x=factor(cyl), data=mtcars, geom="boxplot") + facet_grid(.~cyl)

### WRANGLING:

- subsetting:
  - mtcars %>% dplyr::filter(mpg>30)

- making a new variable:
  - mtcars %>% dplyr::mutate(efficient = if_else(mpg>30, TRUE, FALSE))
Compact but fragile (base R) → Robust but verbose (base R) → Direct and robust (tidyverse)
```r
library(forcats)

# Summary of opinion of income
summary(GSS$OpinionOfIncome)

<table>
<thead>
<tr>
<th>Opinion</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Above average</td>
<td>483</td>
</tr>
<tr>
<td>Average</td>
<td>1118</td>
</tr>
<tr>
<td>Below average</td>
<td>666</td>
</tr>
<tr>
<td>Far above average</td>
<td>65</td>
</tr>
<tr>
<td>Far below average</td>
<td>179</td>
</tr>
<tr>
<td>No answer</td>
<td>6</td>
</tr>
<tr>
<td>NA's</td>
<td>2</td>
</tr>
</tbody>
</table>

GSS <- GSS %>%
  mutate(tidyOpinionOfIncome =
        fct_relevel(OpinionOfIncome,
        "Far above average",
        "Above average",
        "Average",
        "Below average",
        "Far below average"))

# Summary of tidy opinion of income
summary(GSS$tidyOpinionOfIncome)

<table>
<thead>
<tr>
<th>Opinion</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Far above average</td>
<td>65</td>
</tr>
<tr>
<td>Above average</td>
<td>483</td>
</tr>
<tr>
<td>Average</td>
<td>1118</td>
</tr>
<tr>
<td>Below average</td>
<td>666</td>
</tr>
<tr>
<td>Far below average</td>
<td>179</td>
</tr>
<tr>
<td>Don't know</td>
<td>21</td>
</tr>
<tr>
<td>No answer</td>
<td>6</td>
</tr>
<tr>
<td>NA's</td>
<td>2</td>
</tr>
</tbody>
</table>
```
> GSS$BaseMarital <- GSS$MaritalStatus
> summary(GSS$BaseMarital)

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Divorced</td>
<td>411</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Married</td>
<td>1158</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Never married</td>
<td>675</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No answer</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Separated</td>
<td>81</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Widowed</td>
<td>209</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

> levels(GSS$BaseMarital) <- c("Not married", "Married", 
+    "Not married", "No answer", 
+    "Not married", "Not married", NA)

> summary(GSS$BaseMarital)

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Not married</td>
<td>1376</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Married</td>
<td>1158</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No answer</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NA's</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
> summary(GSS$MaritalStatus)
<table>
<thead>
<tr>
<th>Divorced</th>
<th>Married</th>
<th>Never married</th>
<th>No answer</th>
<th>Separated</th>
<th>Widowed</th>
<th>NA's</th>
</tr>
</thead>
<tbody>
<tr>
<td>411</td>
<td>1158</td>
<td>675</td>
<td>4</td>
<td>81</td>
<td>209</td>
<td>2</td>
</tr>
</tbody>
</table>

> GSS <- GSS %>%
+ mutate(tidyMaritalStatus = recode(MaritalStatus,
+ Divorced = "Not married",
+ "Never married" = "Not married",
+ Widowed = "Not married",
+ Separated = "Not married"))

> summary(GSS$tidyMaritalStatus)
<table>
<thead>
<tr>
<th>Not married</th>
<th>Married</th>
<th>No answer</th>
<th>NA's</th>
</tr>
</thead>
<tbody>
<tr>
<td>1376</td>
<td>1158</td>
<td>4</td>
<td>2</td>
</tr>
</tbody>
</table>

>`
Defensive coding
> summary(GSS$tidyOpinionOfIncome)

<table>
<thead>
<tr>
<th>Far above average</th>
<th>Above average</th>
<th>Average</th>
<th>Below average</th>
<th>Far below average</th>
</tr>
</thead>
<tbody>
<tr>
<td>65</td>
<td>483</td>
<td>1118</td>
<td>666</td>
<td>179</td>
</tr>
<tr>
<td>Don't know</td>
<td>No answer</td>
<td>NA's</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>21</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

> summary(GSS$tidyOpinionOfIncome)

<table>
<thead>
<tr>
<th>Far above average</th>
<th>Above average</th>
<th>Average</th>
<th>Below average</th>
<th>Far below average</th>
</tr>
</thead>
<tbody>
<tr>
<td>65</td>
<td>483</td>
<td>1118</td>
<td>666</td>
<td>179</td>
</tr>
<tr>
<td>Don't know</td>
<td>No answer</td>
<td>NA's</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>21</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

>
```r
library(assertthat)
levels(drinkstat)
[1] "abstinent" "highrisk" "moderate"
assert_that(length(levels(drinkstat)) == 3)
[1] TRUE
library(testthat)
levels(GSS$Sex)
[1] "Female" "Male"
expect_equivalent(levels(GSS$Sex), c("Female", "Male"))
expect_equivalent(levels(GSS$Sex), c("Male", "Female"))
Error: levels(GSS$Sex) not equivalent to c("Male", "Female").
2/2 mismatches
x[1]: "Female"
y[1]: "Male"
x[2]: "Male"
y[2]: "Female"
```
Takeaways:
• Use forcats
• Practice defensive coding
• `summary()` is your friend
• `assertthat` and `testthat`
Thank you

Amelia McNamara                   @AmeliaMN
www.amelia.mn

University of St Thomas Department of Computer and Information Sciences