Tibbles and Data Transformation

Samuel Lippl

01 November 2018

Agenda

- Tibbles and data frames
- Tidy data
- The tidyverse
- Data transformation

Tibbles and data frames

First look

```
library(dplyr)
starwars
```

First look

```
library(dplyr)
starwars
```

```
## # A tibble: 87 x 13
##    name  height  mass hair_color skin_color eye_color birth_year gender
##    <chr>  <int> <dbl> <chr>      <chr>      <chr>          <dbl> <chr>
##  1 Luke…    172    77 blond      fair       blue            19   male
##  2 C-3PO    167    75 <NA>       gold       yellow         112   <NA>
##  3 R2-D2     96    32 <NA>       white, bl… red             33   <NA>
##  4 Dart…    202   136 none       white      yellow          41.9 male
##  5 Leia…    150    49 brown      light      brown           19   female
##  6 Owen…    178   120 brown, gr… light      blue            52   male
##  7 Beru…    165    75 brown      light      blue            47   female
##  8 R5-D4     97    32 <NA>       white, red red             NA   <NA>
##  9 Bigg…    183    84 black      light      brown           24   male
## 10 Obi-…    182    77 auburn, w… fair       blue-gray       57   male
## # ... with 77 more rows, and 5 more variables: homeworld <chr>,
## #   species <chr>, films <list>, vehicles <list>, starships <list>
```

First look

```
library(dplyr)
as.data.frame(starwars)
```

```
##                     name height   mass    hair_color          skin_color
## 1         Luke Skywalker    172   77.0         blond                fair
## 2                  C-3PO    167   75.0          <NA>                gold
## 3                  R2-D2     96   32.0          <NA>         white, blue
## 4            Darth Vader    202  136.0          none               white
## 5            Leia Organa    150   49.0         brown               light
## 6              Owen Lars    178  120.0   brown, grey               light
## 7     Beru Whitesun lars    165   75.0         brown               light
## 8                  R5-D4     97   32.0          <NA>          white, red
## 9      Biggs Darklighter    183   84.0         black               light
## 10        Obi-Wan Kenobi    182   77.0 auburn, white                fair
## 11      Anakin Skywalker    188   84.0         blond                fair
## 12        Wilhuff Tarkin    180     NA  auburn, grey                fair
## 13             Chewbacca    228  112.0         brown             unknown
## 14              Han Solo    180   80.0         brown                fair
## 15                Greedo    173   74.0          <NA>               green
```

Tibbles

```
vignette("tibble")
```

Tibbles are a modern take on data frames. They keep the features that have stood the test of time, and drop the features that used to be convenient but are now frustrating.

- You should always use tibbles

Data frames

Data frames consist of:

- rows representing observations
- columns representing variables
- every column has one type

Column types

- Numbers:
    - `dbl`: real numbers
    - `int`: integers
- `lgl`: boolean (true/false)
- `chr`: Characters
- `fct`: Factors
- many other types

Factors

- represent categorical variables

```
colleges <- c("St Edmund", "Exeter", "Queen's", "St John's")
x <- c("St Edmund", "Exeter", "St Edmund", "Queen's",
       "St Edmund", "Queen's", "Exeter", "Exeter")
fct <- factor(x, levels = colleges)
fct
```

```
## [1] St Edmund Exeter    St Edmund Queen's   St Edmund Queen's   Exeter   
## [8] Exeter   
## Levels: St Edmund Exeter Queen's St John's
```

```
str(fct)
```

```
##  Factor w/ 4 levels "St Edmund","Exeter",..: 1 2 1 3 1 3 2 2
```

Structure of factors

```
attributes(fct)
```

```
## $levels
## [1] "St Edmund" "Exeter"    "Queen's"   "St John's"
## 
## $class
## [1] "factor"
```

```
levels(fct)
```

```
## [1] "St Edmund" "Exeter"    "Queen's"   "St John's"
```

Exercise 1

1. Create a tibble with three rows (different students) and the two variables `name` (as a character) and `college` (as a character). (If someone does not have a college, type in "NA" (not applicable).)
2. Turn the character column into a factor.
3. Dangers with factors: what is happening in the following two lines of code?

```
as.integer(c("1", "2"))
```

```
## [1] 1 2
```

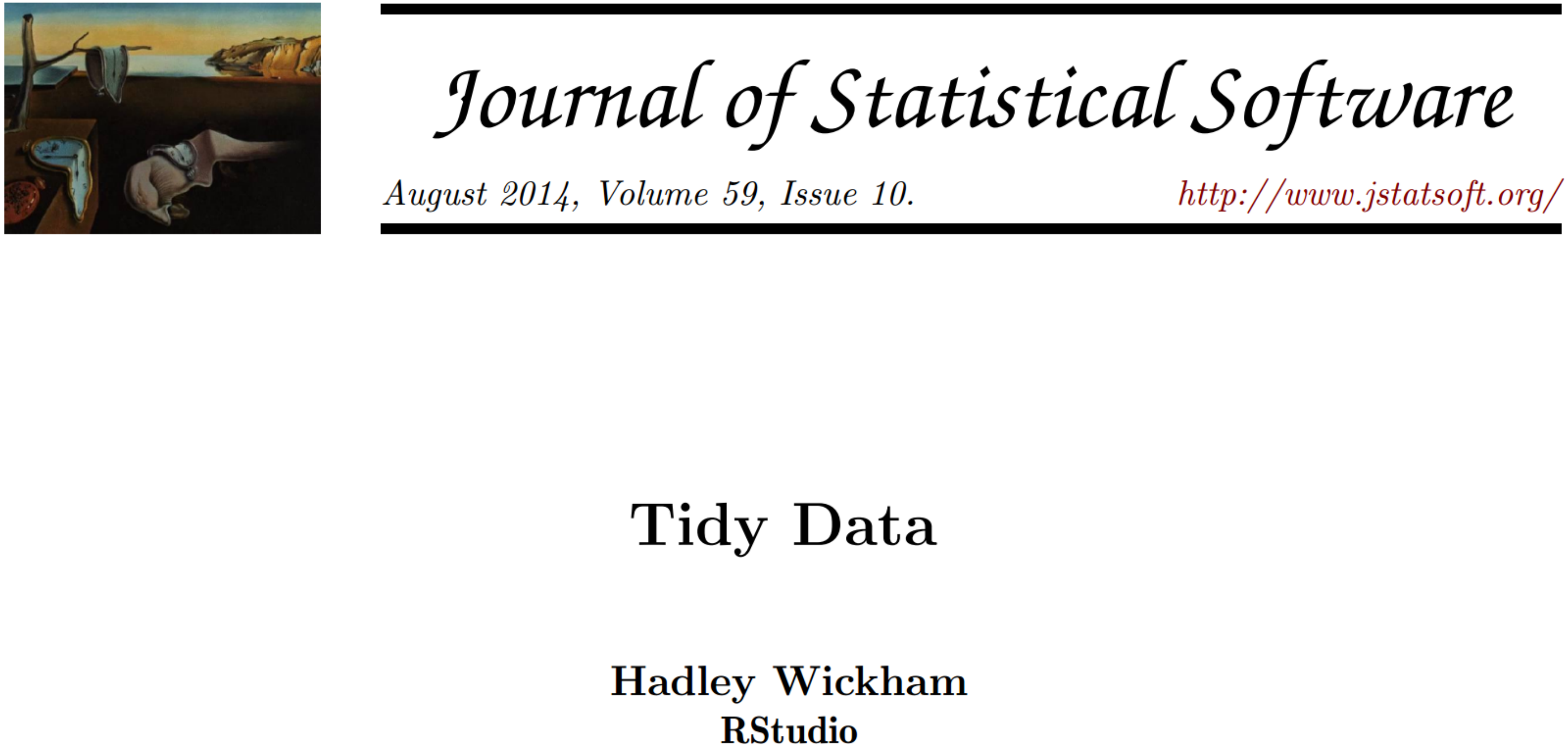
```
as.integer(factor(c("2", "1")))
```

```
## [1] 2 1
```

Tidy data

Tidy data

http://www.jstatsoft.org/v59/i10/paper



Journal of Statistical Software

August 2014, Volume 59, Issue 10. http://www.jstatsoft.org/

Tidy Data

Hadley Wickham
RStudio

Tolstoy in statistics

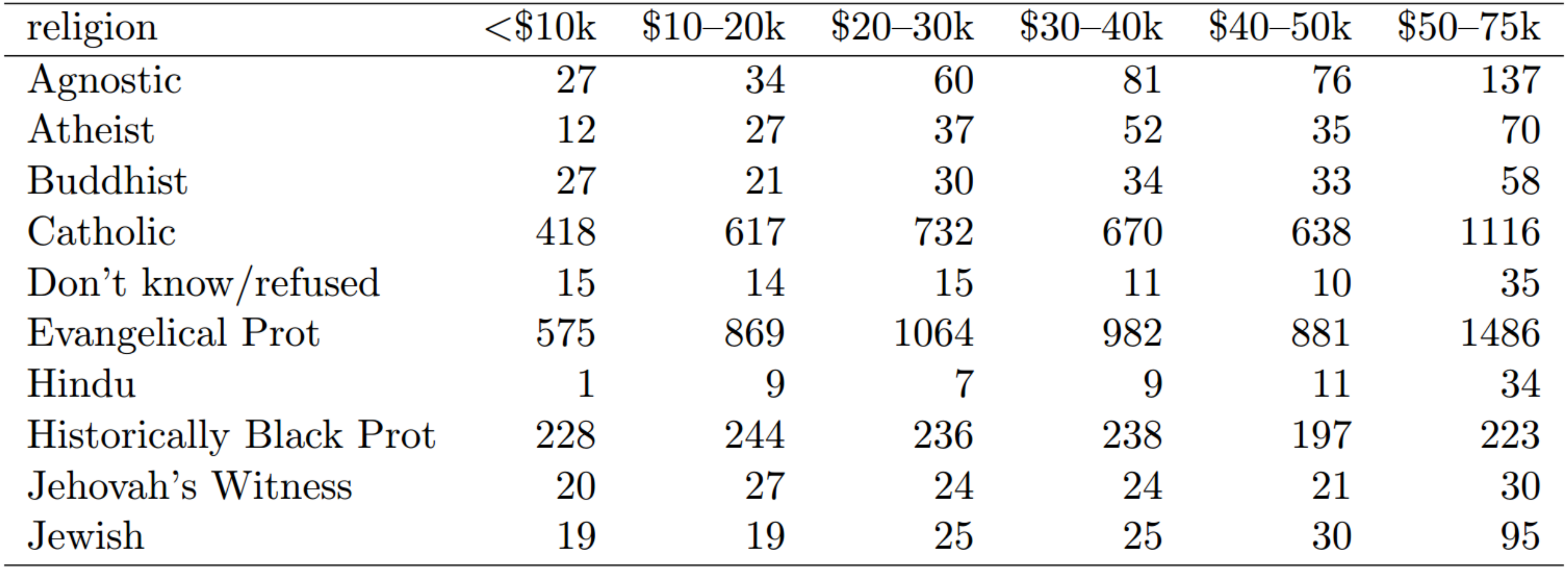
"Happy families are all alike; every unhappy family is unhappy in its own way." - Leo Tolstoy

"Tidy datasets are alike but every messy dataset is messy in its own way." - Hadley Wickham

- A standardized format for datasets makes their manipulation easier.

1. Each variable forms a column.
2. Each observation forms a row.
3. Each type of observational unit forms a table.

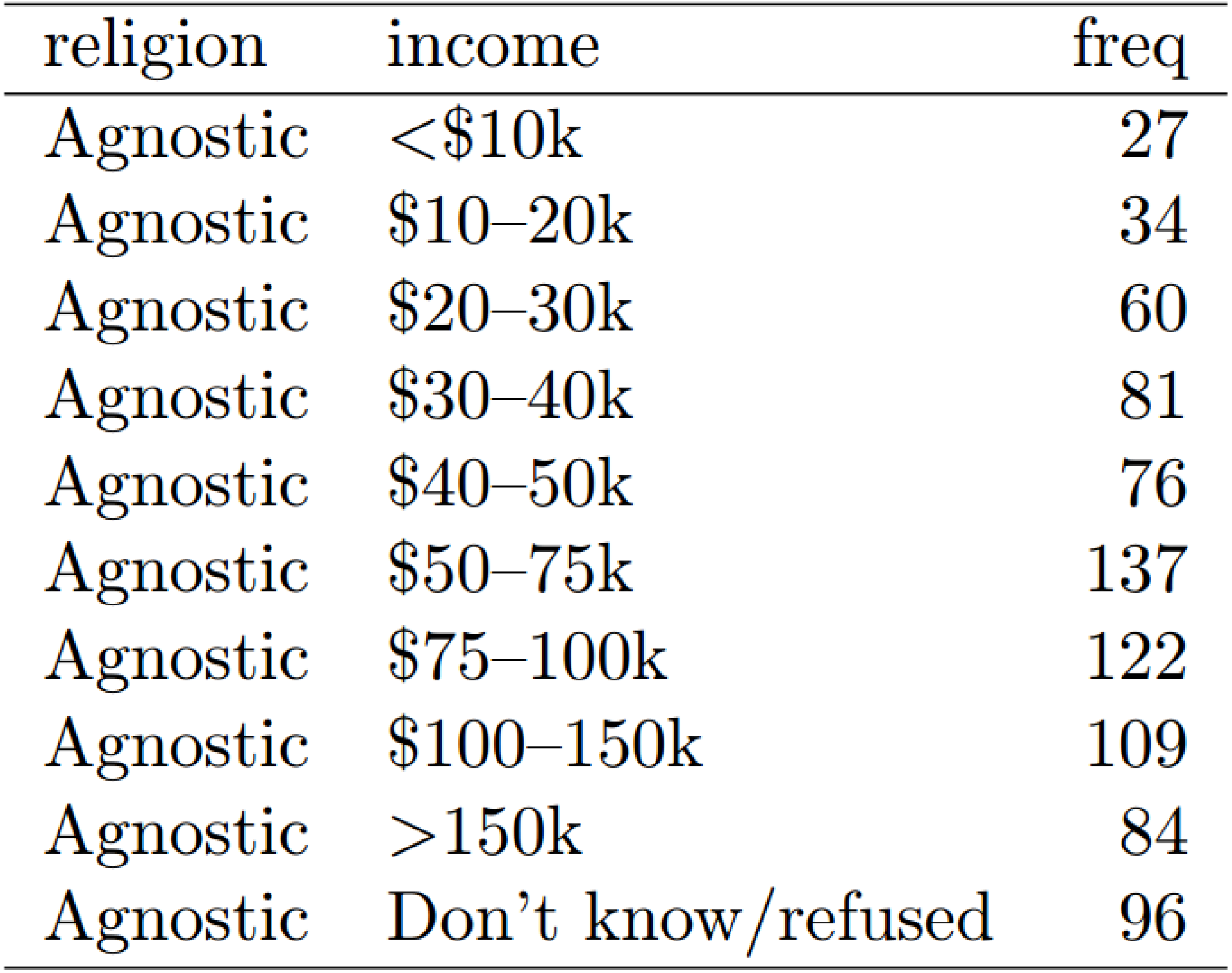
Messy data: example



religion	<$10k	$10–20k	$20–30k	$30–40k	$40–50k	$50–75k
Agnostic	27	34	60	81	76	137
Atheist	12	27	37	52	35	70
Buddhist	27	21	30	34	33	58
Catholic	418	617	732	670	638	1116
Don’t know/refused	15	14	15	11	10	35
Evangelical Prot	575	869	1064	982	881	1486
Hindu	1	9	7	9	11	34
Historically Black Prot	228	244	236	238	197	223
Jehovah’s Witness	20	27	24	24	21	30
Jewish	19	19	25	25	30	95

- Problem: Variable headers are values

Messy data: solution



religion	income	freq
Agnostic	<$10k	27
Agnostic	$10–20k	34
Agnostic	$20–30k	60
Agnostic	$30–40k	81
Agnostic	$40–50k	76
Agnostic	$50–75k	137
Agnostic	$75–100k	122
Agnostic	$100–150k	109
Agnostic	>150k	84
Agnostic	Don't know/refused	96

Remarks on tidy data

- the principles of tidy data might seem trivial but they are not
- functions expect tidy input and give tidy output (exceptions are e. g. visualizations)

The tidyverse

The tidyverse

- the tidyverse implements a tidy approach towards data analysis

```
install.packages("tidyverse")
library(tidyverse)
```

Coding style

- a variable is an object
- a function is a verb
- the pipe `%>%` connects objects and verbs

```
diamonds %>%
  filter(cut == "Ideal") %>%
  select(carat, clarity)
```

Coding style

```
## # A tibble: 21,551 x 2
##    carat clarity
##    <dbl> <ord>  
##  1  0.23 SI2    
##  2  0.23 VS1    
##  3  0.31 SI2    
##  4  0.3  SI2    
##  5  0.33 SI2    
##  6  0.33 SI2    
##  7  0.33 SI1    
##  8  0.23 VS1    
##  9  0.32 SI1    
## 10  0.3  SI2    
## # ... with 21,541 more rows
```

The pipe

```
diamonds %>%
  filter(cut == "Ideal")
```

is actually

```
filter(diamonds, cut == "Ideal")
```

The pipe

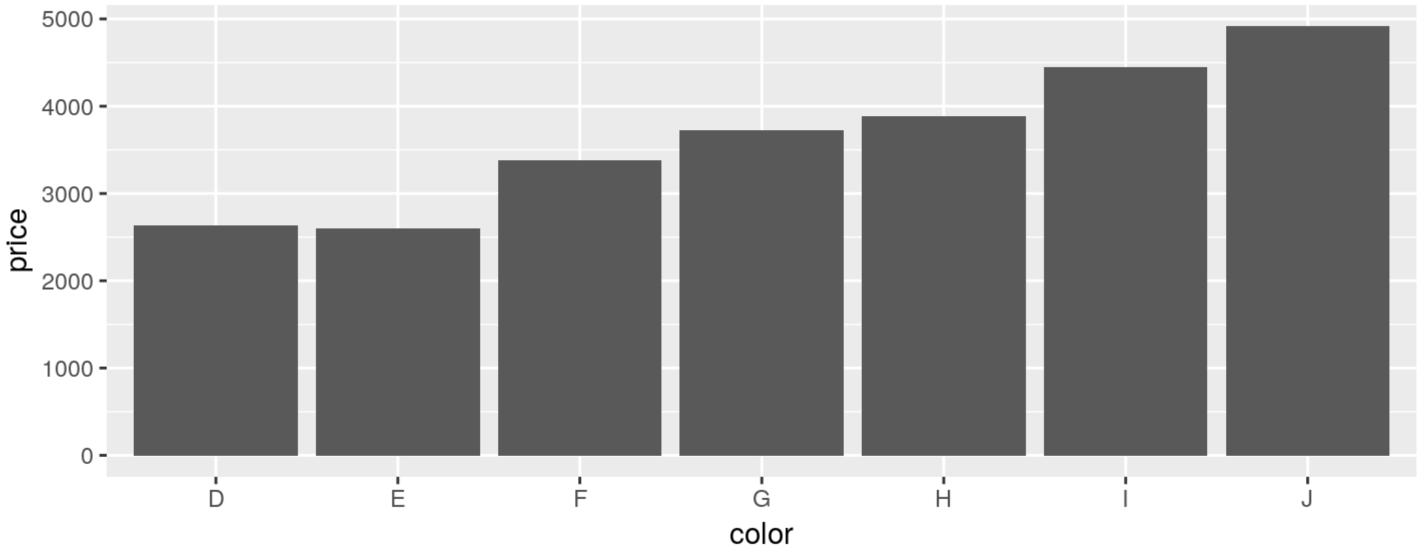
```r
diamonds %>%
  filter(cut == "Ideal") %>%
  select(carat, clarity)
```

is actually

```r
select(
  filter(diamonds, cut == "Ideal"),
  carat, clarity
)
```

The pipe makes code more readable

```
diamonds %>%
  filter(cut == "Ideal") %>%
  group_by(color) %>%
  summarise(price = mean(price)) %>%
  ggplot(aes(x = color, y = price)) +
  geom_bar(stat = "identity")
```



Exercise 2

Split the following chunks of code up using the pipe:

```
select(
  diamonds,
  color, cut, clarity
)
```

```
filter(
  select(
    diamonds,
    color, cut, clarity, depth
  ),
  depth >= 60
)
```

Data transformation

Select variables

- `select` selects all given, unquoted variables
- there are special functions to help with selection

```
diamonds
```

```
## # A tibble: 53,940 x 10
##    carat cut       color clarity depth table price     x     y     z
##    <dbl> <ord>     <ord> <ord>   <dbl> <dbl> <int> <dbl> <dbl> <dbl>
##  1 0.23  Ideal     E     SI2      61.5    55   326  3.95  3.98  2.43
##  2 0.21  Premium   E     SI1      59.8    61   326  3.89  3.84  2.31
##  3 0.23  Good      E     VS1      56.9    65   327  4.05  4.07  2.31
##  4 0.290 Premium   I     VS2      62.4    58   334  4.2   4.23  2.63
##  5 0.31  Good      J     SI2      63.3    58   335  4.34  4.35  2.75
##  6 0.24  Very Good J     VVS2     62.8    57   336  3.94  3.96  2.48
##  7 0.24  Very Good I     VVS1     62.3    57   336  3.95  3.98  2.47
##  8 0.26  Very Good H     SI1      61.9    55   337  4.07  4.11  2.53
##  9 0.22  Fair      E     VS2      65.1    61   337  3.87  3.78  2.49
```

`select`: drop variables

```
diamonds %>%
  select(-x, -y, -z)
```

```
## # A tibble: 53,940 x 7
##    carat cut       color clarity depth table price
##    <dbl> <ord>     <ord> <ord>   <dbl> <dbl> <int>
##  1 0.23  Ideal     E     SI2      61.5    55   326
##  2 0.21  Premium   E     SI1      59.8    61   326
##  3 0.23  Good      E     VS1      56.9    65   327
##  4 0.290 Premium   I     VS2      62.4    58   334
##  5 0.31  Good      J     SI2      63.3    58   335
##  6 0.24  Very Good J     VVS2     62.8    57   336
##  7 0.24  Very Good I     VVS1     62.3    57   336
##  8 0.26  Very Good H     SI1      61.9    55   337
##  9 0.22  Fair      E     VS2      65.1    61   337
## 10 0.23  Very Good H     VS1      59.4    61   338
## # ... with 53,930 more rows
```

`select`: `:` picks adjacent variables

```
diamonds %>%
  select(carat:price)
```

```
## # A tibble: 53,940 x 7
##    carat cut       color clarity depth table price
##    <dbl> <ord>     <ord> <ord>   <dbl> <dbl> <int>
##  1 0.23  Ideal     E     SI2      61.5    55   326
##  2 0.21  Premium   E     SI1      59.8    61   326
##  3 0.23  Good      E     VS1      56.9    65   327
##  4 0.290 Premium   I     VS2      62.4    58   334
##  5 0.31  Good      J     SI2      63.3    58   335
##  6 0.24  Very Good J     VVS2     62.8    57   336
##  7 0.24  Very Good I     VVS1     62.3    57   336
##  8 0.26  Very Good H     SI1      61.9    55   337
##  9 0.22  Fair      E     VS2      65.1    61   337
## 10 0.23  Very Good H     VS1      59.4    61   338
## # ... with 53,930 more rows
```

`select`: special functions

- see manual

```
diamonds %>%
  select(carat, ends_with("e"))
```

```
## # A tibble: 53,940 x 3
##    carat table price
##    <dbl> <dbl> <int>
##  1 0.23     55   326
##  2 0.21     61   326
##  3 0.23     65   327
##  4 0.290    58   334
##  5 0.31     58   335
##  6 0.24     57   336
##  7 0.24     57   336
##  8 0.26     55   337
##  9 0.22     61   337
## 10 0.23     61   338
## # ... with 53,930 more rows
```

Exercises: Preliminary remarks

For our exercises, we will look at the `nycflights13` dataset:

```r
install.packages("nycflights13")
library(nycflights13)
```

The `dplyr` cheatsheet might be helpful:
https://www.rstudio.org/links/data_transformation_cheat_sheet

Exercise 3

Solve the exercises in: https://r4ds.had.co.nz/transform.html#exercises-9

Filter the data frame

- use logical criteria to filter rows
- you can use the data frame's variables

```r
diamonds %>%
  filter(cut == "Ideal")
```

```r
diamonds %>%
  filter(cut == "Ideal", depth >= 60)
```

Logical operators

- Compare values:
    - `x == y`: Are x and y equal?
    - `x != y`: Are x and y not equal?
- Modify logical values:
    - `!x`: not x
    - `x | y`: x or y
    - `x & y`: x and y

Exercise 4

Solve exercises 1 and 2 in: https://r4ds.had.co.nz/transform.html#exercises-7

Mutate the data frame

- `mutate` adds new variables
- you can use the values of other variables within the variables
- uses

```
diamonds %>%
  mutate(price_per_carat = price / carat)
```

Vectorized functions

- Vectorized functions return one value for each entry
  - examples: `x + y`, `log(x)`, `==`

Exercise 5

Solve exercises 2 and 3 in: https://r4ds.had.co.nz/transform.html#exercises-10

Group the data frame

```
diamonds %>%
  group_by(cut)
```

```
## # A tibble: 53,940 x 10
## # Groups:   cut [5]
##    carat cut       color clarity depth table price     x     y     z
##    <dbl> <ord>     <ord> <ord>   <dbl> <dbl> <int> <dbl> <dbl> <dbl>
##  1 0.23  Ideal     E     SI2      61.5    55   326  3.95  3.98  2.43
##  2 0.21  Premium   E     SI1      59.8    61   326  3.89  3.84  2.31
##  3 0.23  Good      E     VS1      56.9    65   327  4.05  4.07  2.31
##  4 0.290 Premium   I     VS2      62.4    58   334  4.2   4.23  2.63
##  5 0.31  Good      J     SI2      63.3    58   335  4.34  4.35  2.75
##  6 0.24  Very Good J     VVS2     62.8    57   336  3.94  3.96  2.48
##  7 0.24  Very Good I     VVS1     62.3    57   336  3.95  3.98  2.47
##  8 0.26  Very Good H     SI1      61.9    55   337  4.07  4.11  2.53
##  9 0.22  Fair      E     VS2      65.1    61   337  3.87  3.78  2.49
## 10 0.23  Very Good H     VS1      59.4    61   338  4     4.05  2.39
## # ... with 53,930 more rows
```

Summarise the groups

```
diamonds %>%
  group_by(color) %>%
  summarise(mean_price = mean(price))
```

```
## # A tibble: 7 x 2
##   color mean_price
##   <ord>      <dbl>
## 1 D          3170.
## 2 E          3077.
## 3 F          3725.
## 4 G          3999.
## 5 H          4487.
## 6 I          5092.
## 7 J          5324.
```

Exercise 6

1. Determine the mean price for each cut
2. Determine the number of occurrences for each cut (hint: look at `n()`)
3. Determine the maximal price and the maximal carat for each color
4. Determine the mean price for each combination of cut and color

Further reading

- R4DS, ch. 5, 9, 10, 12
- "Tidy data" by Hadley Wickham: http://www.jstatsoft.org/v59/i10/paper
- Introduction to the tidyverse: tidyverse.org
- `dplyr` cheatsheet: https://www.rstudio.org/links/data_transformation_cheat_sheet