ch7

Michael Rose

```
# coerce a data frame to a tibble

as_tibble(iris)
```

```
## # A tibble: 150 x 5
##    Sepal.Length Sepal.Width Petal.Length Petal.Width Species
##           <dbl>       <dbl>        <dbl>       <dbl> <fct>
##  1          5.1         3.5          1.4         0.2 setosa
##  2          4.9         3            1.4         0.2 setosa
##  3          4.7         3.2          1.3         0.2 setosa
##  4          4.6         3.1          1.5         0.2 setosa
##  5          5           3.6          1.4         0.2 setosa
##  6          5.4         3.9          1.7         0.4 setosa
##  7          4.6         3.4          1.4         0.3 setosa
##  8          5           3.4          1.5         0.2 setosa
##  9          4.4         2.9          1.4         0.2 setosa
## 10          4.9         3.1          1.5         0.1 setosa
## # ... with 140 more rows
```

```
# create a new tibble

tibble(
  x = 1:5,
  y = 1,
  z = x^2 + y
)
```

```
## # A tibble: 5 x 3
##       x     y     z
##   <int> <dbl> <dbl>
## 1     1     1     2
## 2     2     1     5
## 3     3     1    10
## 4     4     1    17
## 5     5     1    26
```

```
# tibble never
# 1. changes the type of inputs
# 2. changes the name of variables
# 3. creates row names

# create a tibble with a nonsyntactic name

tb <- tibble(
  `:)` = "smile",
  ` ` = "space",
  `2000` = "number"
)

tb
```

```
## # A tibble: 1 x 3
##   `:)`  ` `   `2000`
##   <chr> <chr> <chr>
## 1 smile space number
```

```
# tribble is a transposed tibble. It is customized for data entry

tribble(
  ~x, ~y, ~z,
  #--/--/----
  "a", 2, 3.6,
  "b", 1, 8.5
)
```

```
## # A tibble: 2 x 3
##   x         y     z
##   <chr> <dbl> <dbl>
## 1 a         2   3.6
## 2 b         1   8.5
```

```
# Tibbles print method shows only the first 10 rows

tibble(
  a = lubridate::now() + runif(1e3) * 86400,
  b = lubridate::now() + runif(1e3) * 30,
  c = 1:1e3,
  d = runif(1e3),
  e = sample(letters, 1e3, replace = TRUE)
)
```

```
## # A tibble: 1,000 x 5
##    a                   b                       c     d e
##    <dttm>              <dttm>              <int> <dbl> <chr>
##  1 2018-05-05 01:43:41 2018-05-04 18:08:54     1 0.109 p
##  2 2018-05-05 13:42:25 2018-05-04 18:09:13     2 0.177 u
##  3 2018-05-04 23:37:56 2018-05-04 18:09:05     3 0.977 q
##  4 2018-05-05 08:01:55 2018-05-04 18:09:17     4 0.137 e
##  5 2018-05-05 06:35:39 2018-05-04 18:09:12     5 0.767 e
##  6 2018-05-04 23:30:59 2018-05-04 18:09:09     6 0.833 i
##  7 2018-05-05 13:09:27 2018-05-04 18:09:08     7 0.371 g
##  8 2018-05-05 07:03:07 2018-05-04 18:09:22     8 0.739 b
##  9 2018-05-04 23:29:26 2018-05-04 18:08:53     9 0.474 j
## 10 2018-05-05 12:47:05 2018-05-04 18:09:06    10 0.210 x
## # ... with 990 more rows
```

```
# when more of the data frame needs to be shown. width = Inf shows all columns

nycflights13::flights %>%
  print(n = 10, width = Inf)
```

```
## # A tibble: 336,776 x 19
##     year month   day dep_time sched_dep_time dep_delay arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>
##  1  2013     1     1      517            515         2      830
##  2  2013     1     1      533            529         4      850
##  3  2013     1     1      542            540         2      923
##  4  2013     1     1      544            545        -1     1004
```

```
##  5  2013     1     1      554            600        -6      812
##  6  2013     1     1      554            558        -4      740
##  7  2013     1     1      555            600        -5      913
##  8  2013     1     1      557            600        -3      709
##  9  2013     1     1      557            600        -3      838
## 10  2013     1     1      558            600        -2      753
##    sched_arr_time arr_delay carrier flight tailnum origin dest  air_time
##             <int>     <dbl> <chr>    <int> <chr>   <chr>  <chr>    <dbl>
##  1            819        11 UA        1545 N14228  EWR    IAH        227
##  2            830        20 UA        1714 N24211  LGA    IAH        227
##  3            850        33 AA        1141 N619AA  JFK    MIA        160
##  4           1022       -18 B6         725 N804JB  JFK    BQN        183
##  5            837       -25 DL         461 N668DN  LGA    ATL        116
##  6            728        12 UA        1696 N39463  EWR    ORD        150
##  7            854        19 B6         507 N516JB  EWR    FLL        158
##  8            723       -14 EV        5708 N829AS  LGA    IAD         53
##  9            846        -8 B6          79 N593JB  JFK    MCO        140
## 10            745         8 AA         301 N3ALAA  LGA    ORD        138
##    distance  hour minute time_hour
##       <dbl> <dbl>  <dbl> <dttm>
##  1     1400     5     15 2013-01-01 05:00:00
##  2     1416     5     29 2013-01-01 05:00:00
##  3     1089     5     40 2013-01-01 05:00:00
##  4     1576     5     45 2013-01-01 05:00:00
##  5      762     6      0 2013-01-01 06:00:00
##  6      719     5     58 2013-01-01 05:00:00
##  7     1065     6      0 2013-01-01 06:00:00
##  8      229     6      0 2013-01-01 06:00:00
##  9      944     6      0 2013-01-01 06:00:00
## 10      733     6      0 2013-01-01 06:00:00
## # ... with 3.368e+05 more rows
```

```
# view the whole data set

nycflights13::flights %>%
  View()
```

Subsetting

```
df <- tibble(
  x = runif(5),
  y = rnorm(5)
)

# extract by name

df$x
```

```
## [1] 0.1745560 0.3107221 0.9376954 0.1648328 0.4242488
```

```
df[["x"]]
```

```
## [1] 0.1745560 0.3107221 0.9376954 0.1648328 0.4242488
```

```
# extract by position

df[[1]]
```

```
## [1] 0.1745560 0.3107221 0.9376954 0.1648328 0.4242488
```

```
# to use these in a pipe, we need the placeholder .

df %>% .$x
```

```
## [1] 0.1745560 0.3107221 0.9376954 0.1648328 0.4242488
```

```
df %>% .[["x"]]
```

```
## [1] 0.1745560 0.3107221 0.9376954 0.1648328 0.4242488
```

```
# if an older function that is incompatible is encountered, use as.data.frame

class(as.data.frame(tb))
```

```
## [1] "data.frame"
```

```
# How can you tell if an object is a tibble

class(mtcars)
```

```
## [1] "data.frame"
```

```
class(as.tibble(mtcars))
```

```
## [1] "tbl_df"     "tbl"        "data.frame"
```

```
# compare and contrast

qf <- data.frame(abc = 1, xyz = "a")

qf$x
```

```
## [1] a
## Levels: a
```

```
qf[, "xyz"]
```

```
## [1] a
## Levels: a
```

```
qf[, c("abc", "xyz")]
```

```
##   abc xyz
## 1   1   a
```

```
rf <- as.tibble(qf)
rf$x
```

```
## Warning: Unknown or uninitialised column: 'x'.
```

```
## NULL
```

```
rf[, "xyz"]
```

```
## # A tibble: 1 x 1
##   xyz
##   <fct>
```

```
## 1 a
```

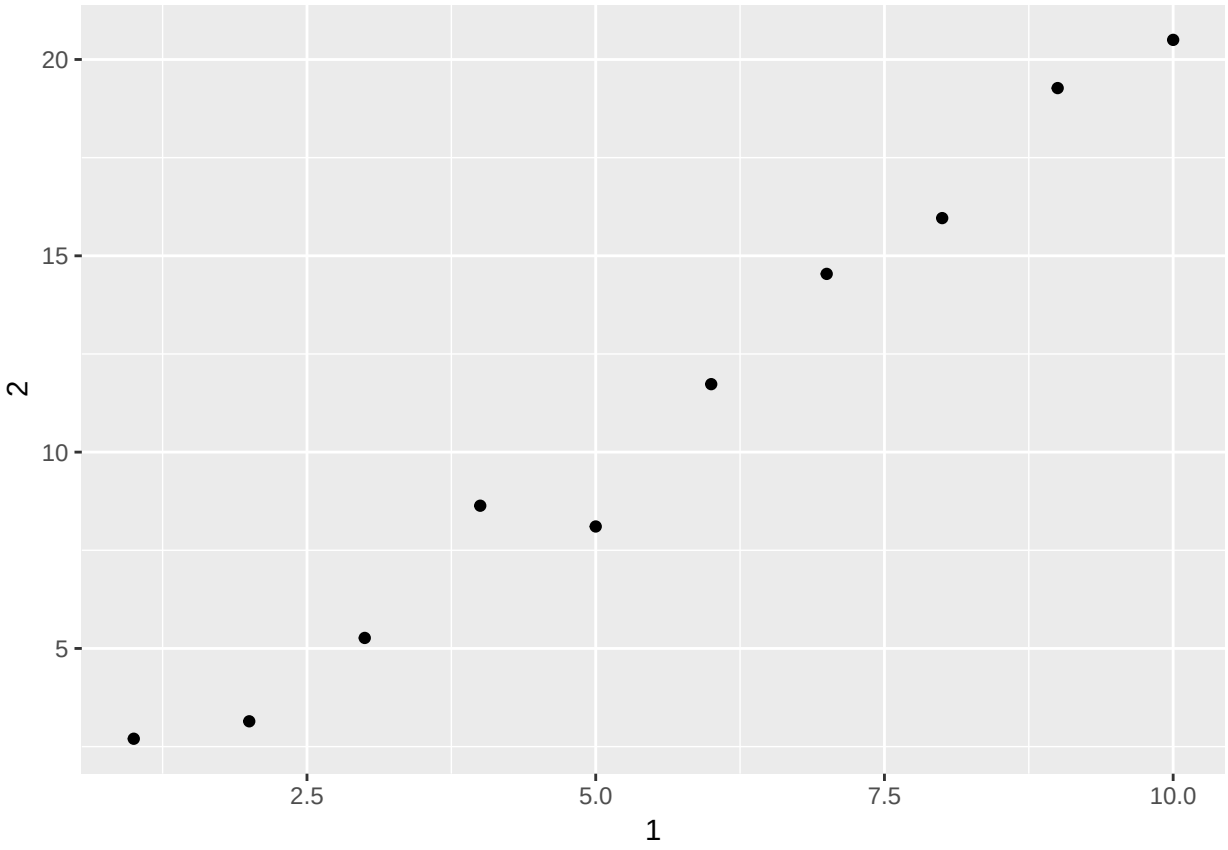
```
rf[, c("abc", "xyz")]
```

```
## # A tibble: 1 x 2
##     abc xyz
##   <dbl> <fct>
## 1     1 a
```

```
# If you have the name of a variable stored in an object, e.g. var <- "mpg" how can you extract the ref
# you can use df[["mpg"]]

# practice with nonsyntactics

annoying <- tibble(
  `1` = 1:10,
  `2` = `1` * 2 + rnorm(length(`1`))
)

# a
annoying$`1`
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

```
# b
ggplot(data = annoying) + geom_point(mapping = aes(x = `1`, y = `2`))
```




```
# c
annoying <- annoying %>%
```

```
  mutate(`3` = `2` / `1`)

# d
annoying <- rename(annoying, one = `1`, two = `2`, three = `3`)
glimpse(annoying)
```

```
## Observations: 10
## Variables: 3
## $ one   <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10
## $ two   <dbl> 2.700294, 3.144495, 5.267522, 8.635074, 8.105897, 11.731...
## $ three <dbl> 2.700294, 1.572248, 1.755841, 2.158769, 1.621179, 1.9551...
```

```
# What does tibble::enframe do? When might you use it?
# It converts atomic vectors into data frames with an extra column for names

enframe(c(1, 2, 3))
```

```
## # A tibble: 3 x 2
##    name value
##   <int> <dbl>
## 1     1     1
## 2     2     2
## 3     3     3
```

```
# What option controls how many additional column names are printed at the footer of a tibble?
# ?print.tbl_df
# n_extra
```