

# Untitled

Han Xiao

2/21/2018

## 10.5

1. Tibbles are data frames, but they tweak some older behaviours to make life a little easier. Tibble will only print a limited number of rows and show the class on top of each column.

```
library(tidyverse)

## -- Attaching packages ----- tidyverse 1.2.1 --
## √ ggplot2 2.2.1      √ purrr  0.2.4
## √ tibble  1.4.2      √ dplyr  0.7.4
## √ tidyr   0.8.0      √ stringr 1.2.0
## √ readr   1.1.1      √ forcats 0.3.0

## -- Conflicts ----- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()

mtcars

##           mpg cyl  disp  hp drat   wt  qsec vs am gear carb
## Mazda RX4      21.0   6 160.0 110 3.90 2.620 16.46 0  1   4    4
## Mazda RX4 Wag  21.0   6 160.0 110 3.90 2.875 17.02 0  1   4    4
## Datsun 710      22.8   4 108.0  93 3.85 2.320 18.61 1  1   4    1
## Hornet 4 Drive  21.4   6 258.0 110 3.08 3.215 19.44 1  0   3    1
## Hornet Sportabout 18.7   8 360.0 175 3.15 3.440 17.02 0  0   3    2
## Valiant         18.1   6 225.0 105 2.76 3.460 20.22 1  0   3    1
## Duster 360      14.3   8 360.0 245 3.21 3.570 15.84 0  0   3    4
## Merc 240D       24.4   4 146.7  62 3.69 3.190 20.00 1  0   4    2
## Merc 230        22.8   4 140.8  95 3.92 3.150 22.90 1  0   4    2
## Merc 280        19.2   6 167.6 123 3.92 3.440 18.30 1  0   4    4
## Merc 280C       17.8   6 167.6 123 3.92 3.440 18.90 1  0   4    4
## Merc 450SE      16.4   8 275.8 180 3.07 4.070 17.40 0  0   3    3
## Merc 450SL      17.3   8 275.8 180 3.07 3.730 17.60 0  0   3    3
## Merc 450SLC     15.2   8 275.8 180 3.07 3.780 18.00 0  0   3    3
## Cadillac Fleetwood 10.4   8 472.0 205 2.93 5.250 17.98 0  0   3    4
## Lincoln Continental 10.4   8 460.0 215 3.00 5.424 17.82 0  0   3    4
## Chrysler Imperial 14.7   8 440.0 230 3.23 5.345 17.42 0  0   3    4
## Fiat 128        32.4   4  78.7  66 4.08 2.200 19.47 1  1   4    1
## Honda Civic     30.4   4  75.7  52 4.93 1.615 18.52 1  1   4    2
## Toyota Corolla  33.9   4  71.1  65 4.22 1.835 19.90 1  1   4    1
## Toyota Corona   21.5   4 120.1  97 3.70 2.465 20.01 1  0   3    1
## Dodge Challenger 15.5   8 318.0 150 2.76 3.520 16.87 0  0   3    2
## AMC Javelin     15.2   8 304.0 150 3.15 3.435 17.30 0  0   3    2
## Camaro Z28      13.3   8 350.0 245 3.73 3.840 15.41 0  0   3    4
## Pontiac Firebird 19.2   8 400.0 175 3.08 3.845 17.05 0  0   3    2
## Fiat X1-9       27.3   4  79.0  66 4.08 1.935 18.90 1  1   4    1
```

```
## Porsche 914-2      26.0   4 120.3  91 4.43 2.140 16.70  0  1   5   2
## Lotus Europa      30.4   4  95.1 113 3.77 1.513 16.90  1  1   5   2
## Ford Pantera L    15.8   8 351.0 264 4.22 3.170 14.50  0  1   5   4
## Ferrari Dino      19.7   6 145.0 175 3.62 2.770 15.50  0  1   5   6
## Maserati Bora      15.0   8 301.0 335 3.54 3.570 14.60  0  1   5   8
## Volvo 142E        21.4   4 121.0 109 4.11 2.780 18.60  1  1   4   2
```

## 2.

Using dollar sign a data.frame will partially complete the column. This saves a few keystrokes, but can result in accidentally using a different variable than you thought you were using. With data.frames, with [ the type of object that is returned differs on the number of columns. If it is one column, it won't return a data.frame, but instead will return a vector. With more than one column, then it will return a data.frame. This is fine if you know what you are passing in, but suppose you did `df[, vars]` where `vars` was a variable. Then you what that code does depends on `length(vars)` and you'd have to write code to account for those situations or risk bugs.

```
df <- data.frame(abc = 1, xyz = "a")
df$x
```

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

```
df[, "xyz"]
```

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

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

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

```
tbl <- as_tibble(df)
tbl$x
```

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

```
## NULL
```

```
tbl[, "xyz"]
```

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

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

```
## # A tibble: 1 x 2
##   abc xyz
##   <dbl> <fct>
## 1 1.00 a
```

3.

You can use the double bracket, like `df[[var]]`. You cannot use the dollar sign, because `df$var` would look for a column named `var`.

4.

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

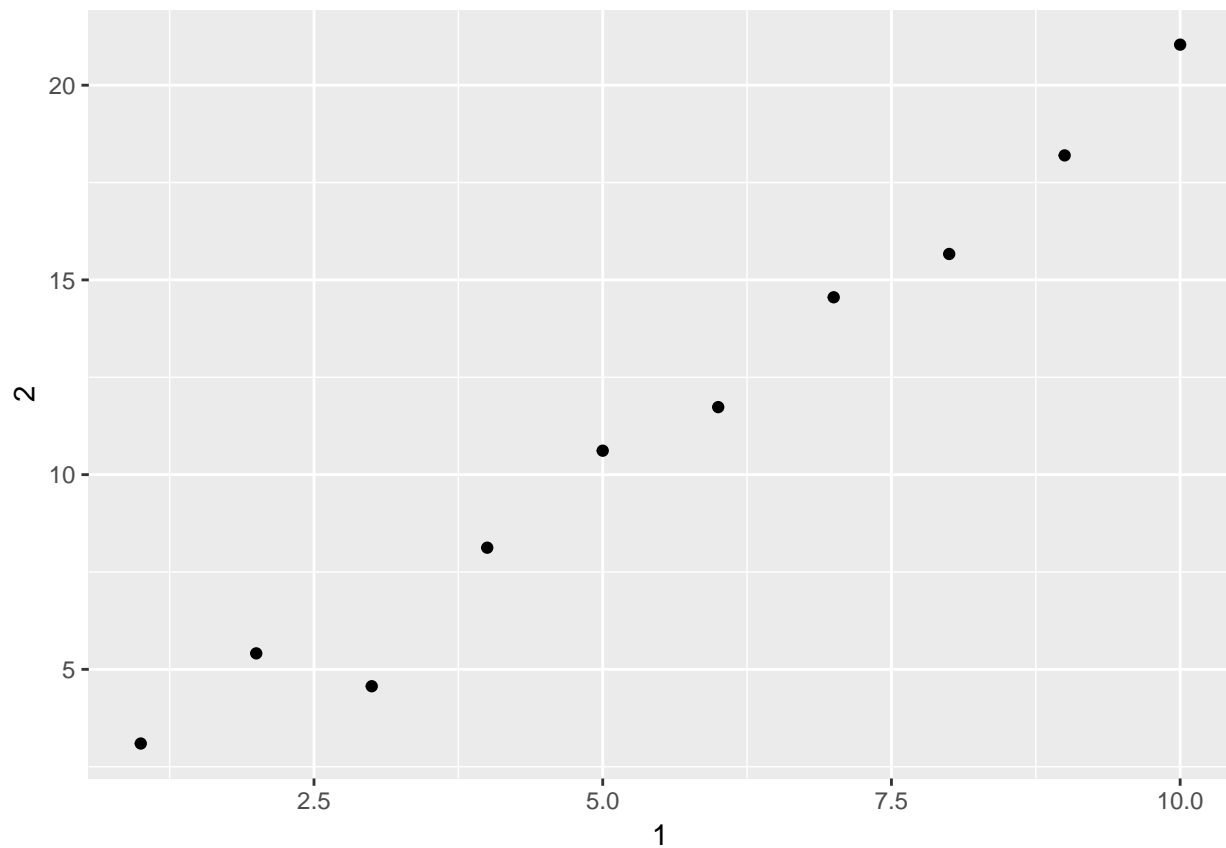
```
annoying[["1"]]
```

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

```
annoying$`1`
```

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

```
ggplot(annoying, aes(x = `1`, y = `2`)) +  
  geom_point()
```



```
annoying[["3"]] <- annoying$`2` / annoying$`1`  
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> 3.095107, 5.409883, 4.566407, 8.123413, 10.614117, 11.73...
## $ three <dbl> 3.095107, 2.704941, 1.522136, 2.030853, 2.122823, 1.9555...
```

## 5.It converts named vectors to a data frame with names and values

```
enframe(c(a = 1, b = 2, c = 3))
```

```
## # A tibble: 3 x 2
##   name  value
##   <chr> <dbl>
## 1 a     1.00
## 2 b     2.00
## 3 c     3.00
```

## 6.The print function for tibbles is in print.tbl\_df

The option `n_extra` determines the number of extra columns to print information for.

### 12.6.1

```
who1 <- who %>%
  gather(new_sp_m014:newrel_f65, key = "key", value = "cases", na.rm = TRUE)
glimpse(who1)
```

```
## Observations: 76,046
## Variables: 6
## $ country <chr> "Afghanistan", "Afghanistan", "Afghanistan", "Afghanis...
## $ iso2     <chr> "AF", "AF", "AF", "AF", "AF", "AF", "AF", "AF", "AF", ...
## $ iso3     <chr> "AFG", "AFG", "AFG", "AFG", "AFG", "AFG", "AFG", "AFG", "AFG"...
## $ year     <int> 1997, 1998, 1999, 2000, 2001, 2002, 2003, 2004, 2005, ...
## $ key      <chr> "new_sp_m014", "new_sp_m014", "new_sp_m014", "new_sp_m...
## $ cases    <int> 0, 30, 8, 52, 129, 90, 127, 139, 151, 193, 186, 187, 2...
```

```
who2 <- who1 %>%
  mutate(key = stringr::str_replace(key, "newrel", "new_rel"))
```

```
who3 <- who2 %>%
  separate(key, c("new", "type", "sexage"), sep = "_")
who3
```

```
## # A tibble: 76,046 x 8
##   country    iso2 iso3  year new   type sexage cases
##   <chr>      <chr> <chr> <int> <chr> <chr> <chr>   <int>
## 1 Afghanistan AF    AFG   1997 new   sp    m014     0
## 2 Afghanistan AF    AFG   1998 new   sp    m014    30
## 3 Afghanistan AF    AFG   1999 new   sp    m014     8
## 4 Afghanistan AF    AFG   2000 new   sp    m014    52
## 5 Afghanistan AF    AFG   2001 new   sp    m014   129
## 6 Afghanistan AF    AFG   2002 new   sp    m014    90
## 7 Afghanistan AF    AFG   2003 new   sp    m014   127
```

```
## 8 Afghanistan AF AFG 2004 new sp m014 139
## 9 Afghanistan AF AFG 2005 new sp m014 151
## 10 Afghanistan AF AFG 2006 new sp m014 193
## # ... with 76,036 more rows
```

```
who3 %>%
  count(new)
```

```
## # A tibble: 1 x 2
##   new      n
##   <chr> <int>
## 1 new  76046
```

```
who4 <- who3 %>%
  select(-new, -iso2, -iso3)
```

```
who5 <- who4 %>%
  separate(sexage, c("sex", "age"), sep = 1)
who5
```

```
## # A tibble: 76,046 x 6
##   country      year type sex age cases
##   <chr>      <int> <chr> <chr> <chr> <int>
## 1 Afghanistan 1997 sp m 014 0
## 2 Afghanistan 1998 sp m 014 30
## 3 Afghanistan 1999 sp m 014 8
## 4 Afghanistan 2000 sp m 014 52
## 5 Afghanistan 2001 sp m 014 129
## 6 Afghanistan 2002 sp m 014 90
## 7 Afghanistan 2003 sp m 014 127
## 8 Afghanistan 2004 sp m 014 139
## 9 Afghanistan 2005 sp m 014 151
## 10 Afghanistan 2006 sp m 014 193
## # ... with 76,036 more rows
```

## 1

So it appears that either a country has all its values in a year as non-missing if the WHO collected data for that country, or all its values are non-missing. So it is okay to treat explicitly and implicitly missing values the same, and we don't lose any information by dropping them.

```
who1 %>%
  filter(cases == 0) %>%
  nrow()
```

```
## [1] 11080
```

```
gather(who, new_sp_m014:newrel_f65, key = "key", value = "cases") %>%
  group_by(country, year) %>%
  mutate(missing = is.na(cases)) %>%
  select(country, year, missing) %>%
  distinct() %>%
  group_by(country, year) %>%
  filter(n() > 1)
```

```
## # A tibble: 6,968 x 3
## # Groups:   country, year [3,484]
```

```
##   country      year missing
##   <chr>        <int> <lg1>
## 1 Afghanistan 1997 F
## 2 Afghanistan 1998 F
## 3 Afghanistan 1999 F
## 4 Afghanistan 2000 F
## 5 Afghanistan 2001 F
## 6 Afghanistan 2002 F
## 7 Afghanistan 2003 F
## 8 Afghanistan 2004 F
## 9 Afghanistan 2005 F
## 10 Afghanistan 2006 F
## # ... with 6,958 more rows
```

## 2

separate emits the warning “too few values”, and if we check the rows for keys beginning with “newrel\_”, we see that sexage is missing, and type = m014

```
who3a <- who1 %>%
  separate(key, c("new", "type", "sexage"), sep = "_")
```

```
## Warning: Expected 3 pieces. Missing pieces filled with `NA` in 2580 rows
## [73467, 73468, 73469, 73470, 73471, 73472, 73473, 73474, 73475, 73476,
## 73477, 73478, 73479, 73480, 73481, 73482, 73483, 73484, 73485, 73486, ...].
```

```
filter(who3a, new == "newrel") %>% head()
```

```
## # A tibble: 6 x 8
##   country      iso2 iso3  year new   type sexage cases
##   <chr>        <chr> <chr> <int> <chr> <chr> <chr> <int>
## 1 Afghanistan AF    AFG   2013 newrel m014 <NA>   1705
## 2 Albania     AL    ALB   2013 newrel m014 <NA>    14
## 3 Algeria     DZ    DZA   2013 newrel m014 <NA>    25
## 4 Andorra     AD    AND   2013 newrel m014 <NA>     0
## 5 Angola      AO    AGO   2013 newrel m014 <NA>   486
## 6 Anguilla     AI    AIA   2013 newrel m014 <NA>     0
```

## 3

```
select(who3, country, iso2, iso3) %>%
  distinct() %>%
  group_by(country) %>%
  filter(n() > 1)
```

```
## # A tibble: 0 x 3
## # Groups:   country [0]
## # ... with 3 variables: country <chr>, iso2 <chr>, iso3 <chr>
```

```
who5 %>%
  group_by(country, year, sex) %>%
  filter(year > 1995) %>%
  summarise(cases = sum(cases)) %>%
  unite(country_sex, country, sex, remove = FALSE) %>%
```

```
ggplot(aes(x = year, y = cases, group = country_sex, colour = sex)) +  
geom_line()
```

