Applications of Data Science

Tidy Data Wrangling - Part A

Applications of Data Science - Class 2

Giora Simchoni

`gsimchoni@gmail.com and add #dsapps in subject`

Stat. and OR Department, TAU

2020-03-17

APPLICATIONS

OF DATA SCIENCE

`dplyr`: Basic Data Verbs

Applications

Of Data Science

Basic Data Verbs

- `filter()` rows based on one or more conditions
- `mutate()` one or more columns, usually based on existing columns
- `select()` the column(s) you want
- `arrange()` rows by one or more columns order
- `summarize()` or `summarise()` that single quantity off a column
- `pull()` a column as a vector, don't want it as a column no more

And the much beloved `group_by()`: do *whatever* by groups of one or more variables.

Read in the data

```r
library(tidyverse)

okcupid <- read_csv("../data/okcupid.csv.zip")
```

Reminder:

```r
dim(okcupid)
```

```
## [1] 59946    31
```

```r
colnames(okcupid)
```

```
##  [1] "age"         "body_type"   "diet"        "drinks"      "drugs"
##  [6] "education"   "essay0"      "essay1"      "essay2"      "essay3"
## [11] "essay4"      "essay5"      "essay6"      "essay7"      "essay8"
## [16] "essay9"      "ethnicity"   "height"      "income"      "job"
## [21] "last_online" "location"    "offspring"   "orientation" "pets"
## [26] "religion"    "sex"         "sign"        "smokes"      "speaks"
## [31] "status"
```

`mutate()`

Add a column `height_cm`, the `height` in centimeters:

```
okcupid <- okcupid %>%
  mutate(height_cm = 2.54 * height)
```

💡 if you also load the `magrittr` package you could do:

```
okcupid %<>% mutate(height_cm = 2.54 * height)
```

filter() and select()

Filter only women, select only age and height:

```
okcupid %>%
  filter(sex == "f") %>%
  select(age, height)
```

```
## # A tibble: 24,117 x 2
##       age height
##     <dbl>  <dbl>
##  1     32     65
##  2     31     65
##  3     24     67
##  4     30     66
##  5     29     62
##  6     39     65
##  7     26     64
##  8     27     67
##  9     22     67
## 10     27     64
## # ... with 24,107 more rows
```

Same but income over 100K, and select all essay questions:

```
okcupid %>%
  filter(sex == "f", income > 100000) %>%
  select(starts_with("essay"))
```

```
## # A tibble: 208 x 10
##    essay0  essay1  essay2  essay3 essay4 essay5 essay6 essay7 essay8 ess
##    <chr>   <chr>   <chr>   <chr>  <chr>  <chr>  <chr>  <chr>  <chr>  <ch
##  1 "i lov~ "being~ "scraw~ "my b~ "musi~ "vege~ "maki~ "kick~ "wow,~ "if
##  2 i'm si~ curren~ eating~ my po~ pride~ nothi~ my ne~ eatin~ i'm n~ you
##  3 "welco~ "piano~ "singi~ "my h~ "book~ "touc~ diffe~ <NA>   <NA>   you
##  4 "pureb~ "by da~ "being~ my ha~ "to s~ "- wa~ my ne~ "i tr~ ummmm~ "yo
##  5 "i was~ "chick~ "using~ lips ~ "arma~ lust,~ enter~ makin~ <NA>   "yo
##  6 "hello~ "i tal~ "anyth~ "my a~ "book~ "my g~ "ever~ "i wo~ <NA>   "if
##  7 "life'~ "i'm j~ "getti~ its b~ "otis~ "1. s~ "the ~ "oh m~ i don~ if
##  8 "every~ "livin~ being ~ my ey~ "dubs~ "dirt~ "how ~ "reco~ i lov~ "yo
##  9 love t~ daily ~ "i am ~ my sm~ love ~ masca~ if i ~ <NA>   "i am~ <NA
## 10 "<b>ph~ "i am ~ "pissi~ my sm~ "book~ "my d~ who p~ "tota~ "my d~ "if
## # ... with 198 more rows
```

Same but using a range of columns:

```
okcupid %>%
  filter(sex == "f", income > 100000) %>%
  select(essay0:essay9)
```

```
## # A tibble: 208 x 10
##    essay0  essay1  essay2  essay3 essay4 essay5 essay6 essay7 essay8 ess
##    <chr>   <chr>   <chr>   <chr>  <chr>  <chr>  <chr>  <chr>  <chr>  <ch
##  1 "i lov~ "being~ "scraw~ "my b~ "musi~ "vege~ "maki~ "kick~ "wow,~ "if
##  2 i'm si~ curren~ eating~ my po~ pride~ nothi~ my ne~ eatin~ i'm n~ you
##  3 "welco~ "piano~ "singi~ "my h~ "book~ "touc~ diffe~ <NA>   <NA>   you
##  4 "pureb~ "by da~ "being~ my ha~ "to s~ "- wa~ my ne~ "i tr~ ummmm~ "yo
##  5 "i was~ "chick~ "using~ lips ~ "arma~ lust,~ enter~ makin~ <NA>   "yo
##  6 "hello~ "i tal~ "anyth~ "my a~ "book~ "my g~ "ever~ "i wo~ <NA>   "if
##  7 "life'~ "i'm j~ "getti~ its b~ "otis~ "1. s~ "the ~ "oh m~ i don~ if
##  8 "every~ "livin~ being ~ my ey~ "dubs~ "dirt~ "how ~ "reco~ i lov~ "yo
##  9 love t~ daily ~ "i am ~ my sm~ love ~ masca~ if i ~ <NA>   "i am~ <NA
## 10 "<b>ph~ "i am ~ "pissi~ my sm~ "book~ "my d~ who p~ "tota~ "my d~ "if
## # ... with 198 more rows
```

Many, many such gifts, see tidyselect

`summarize()`

Find the average height of women

```
okcupid %>%
  filter(sex == "f") %>%
  summarize(avg_height = mean(height_cm, na.rm = TRUE))
```

```
## # A tibble: 1 x 1
##   avg_height
##        <dbl>
## 1       165.
```

Notice we got a `tibble`. We could either `pull` this single number:

```
okcupid %>%
  filter(sex == "f") %>%
  summarize(avg_height = mean(height_cm, na.rm = TRUE)) %>%
  pull()
```

```
## [1] 165.3638
```

Or pull the vector of heights first, then calculate their mean:

```
okcupid %>%
  filter(sex == "f") %>%
  pull(height_cm) %>%
  mean(na.rm = TRUE)
```

```
## [1] 165.3638
```

Amazingly, this would also work:

```
mean(pull(filter(okcupid, sex == "f"), height_cm), na.rm = TRUE)
```

```
## [1] 165.3638
```

`group_by()`

But why settle for women only?

```
okcupid %>%
  group_by(sex) %>%
  summarize(avg_height = mean(height_cm, na.rm = TRUE))
```

```
## # A tibble: 2 x 2
##   sex   avg_height
##   <chr>      <dbl>
## 1 f           165.
## 2 m           179.
```

And you might want to consider `rename()`ing sex!

```
okcupid %>%
  group_by(sex) %>%
  summarize(avg_height = mean(height_cm, na.rm = TRUE)) %>%
  rename(gender = sex)
```

Group by multiple variables, get more summaries, arrange by descending average height:

```
okcupid %>%
  group_by(sex, status) %>%
  summarize(avg_height = mean(height_cm, na.rm = TRUE),
            med_height = median(height_cm, na.rm = TRUE),
            n = n()) %>%
  arrange(-med_height)
```

```
## # A tibble: 10 x 5
## # Groups:   sex [2]
##    sex   status         avg_height med_height     n
##    <chr> <chr>               <dbl>      <dbl> <int>
##  1 m     available            179.       180.  1209
##  2 m     married              179.       180.   175
##  3 m     seeing someone       179.       178.  1061
##  4 m     single               179.       178. 33378
##  5 m     unknown              177.       177.     6
##  6 f     available            166.       166.   656
##  7 f     married              166.       165.   135
##  8 f     seeing someone       165.       165.  1003
##  9 f     single               165.       165. 22319
## 10 f     unknown              161.       159.     4
```

Pro tip: `count()`

When all you want is, well, `count`, no need to `group_by`:

```
okcupid %>% count(body_type, sort = TRUE)
```

```
## # A tibble: 13 x 2
##    body_type          n
##    <chr>          <int>
##  1 average        14652
##  2 fit            12711
##  3 athletic       11819
##  4 <NA>            5296
##  5 thin            4711
##  6 curvy           3924
##  7 a little extra  2629
##  8 skinny          1777
##  9 full figured    1009
## 10 overweight       444
## 11 jacked           421
## 12 used up          355
## 13 rather not say   198
```

Pro tip: `add_count()`

Add count without first creating an initial table, joining etc.:

```
okcupid %>%
  mutate(id = row_number()) %>%
  select(id, body_type, sex) %>%
  add_count(body_type, name = "n_bt") %>%
  filter(n_bt > 10000) %>%
  head(5)
```

```
## # A tibble: 5 x 4
##      id body_type sex    n_bt
##   <int> <chr>     <chr> <int>
## 1     2 average   m     14652
## 2     5 athletic  m     11819
## 3     6 average   m     14652
## 4     7 fit       f     12711
## 5     8 average   f     14652
```

Beyond Basics

Applications

Of Data Science

A simple answer to the religion question?

```
okcupid %>% count(religion)
```

```
## # A tibble: 46 x 2
##    religion                                        n
##    <chr>                                       <int>
##  1 agnosticism                                  2724
##  2 agnosticism and laughing about it            2496
##  3 agnosticism and somewhat serious about it     642
##  4 agnosticism and very serious about it         314
##  5 agnosticism but not too serious about it     2636
##  6 atheism                                      2175
##  7 atheism and laughing about it                2074
##  8 atheism and somewhat serious about it         848
##  9 atheism and very serious about it             570
## 10 atheism but not too serious about it         1318
## # ... with 36 more rows
```

Recoding with `case_when()`

```
okcupid <- okcupid %>% mutate(religion2 = case_when(
  str_detect(religion, "agnosticism") | str_detect(religion, "athe
  str_detect(religion, "buddhism") ~ "buddhist",
  str_detect(religion, "christianity") | str_detect(religion, "cat
  str_detect(religion, "judaism") ~ "jewish",
  str_detect(religion, "hinduism") ~ "hindu",
  str_detect(religion, "islam") ~ "muslim",
  TRUE ~ "NA"))

okcupid %>% count(religion2, sort = TRUE)
```

```
## # A tibble: 7 x 2
##   religion2     n
##   <chr>     <int>
## 1 NA        27969
## 2 atheist   15797
## 3 christian 10545
## 4 jewish     3098
## 5 buddhist   1948
## 6 hindu       450
## 7 muslim      139
```

Getting extreme observations with `top_n()` and `top_frac()`

```r
okcupid %>%
  select(sex, age) %>%
  group_by(sex) %>%
  top_n(3, age)
```

```
## # A tibble: 33 x 2
## # Groups:   sex [2]
##    sex     age
##    <chr> <dbl>
##  1 f       110
##  2 m        69
##  3 m        69
##  4 m        69
##  5 m        69
##  6 m        69
##  7 m        69
##  8 m        69
##  9 m        69
## 10 m        69
## # ... with 23 more rows
```

Unfortunately `top_n` does not deal with ties, could use `slice` for that:

```
okcupid %>%
  select(sex, age) %>%
  group_by(sex) %>%
  top_n(3, age) %>%
  slice(1:3)
```

```
## # A tibble: 6 x 2
## # Groups:   sex [2]
##   sex     age
##   <chr> <dbl>
## 1 f       110
## 2 f        69
## 3 f        69
## 4 m        69
## 5 m        69
## 6 m        69
```

Or use `rank()`

Remove duplicates with `distinct()`

```r
okcupid %>%
  filter(diet == "kosher") %>%
  distinct(body_type, drugs)
```

```
## # A tibble: 7 x 2
##   body_type      drugs
##   <chr>          <chr>
## 1 fit            <NA>
## 2 <NA>           never
## 3 used up        <NA>
## 4 fit            never
## 5 skinny         never
## 6 a little extra never
## 7 jacked         never
```

`distinct()` is much more powerful than `unique()`, see `?distinct`.

To count number of distinct obs look at `n_distinct()`

The `_at()`, `_if()` and `_all()` families

Many of the verbs we've seen come with these suffixes:

```
okcupid %>%
  select_if(is.numeric)
```

```
## # A tibble: 59,946 x 4
##       age height income height_cm
##     <dbl>  <dbl>  <dbl>     <dbl>
##  1     22     75     -1      190.
##  2     35     70  80000      178.
##  3     38     68     -1      173.
##  4     23     71  20000      180.
##  5     29     66     -1      168.
##  6     29     67     -1      170.
##  7     32     65     -1      165.
##  8     31     65     -1      165.
##  9     24     67     -1      170.
## 10     37     65     -1      165.
## # ... with 59,936 more rows
```

Do you see something strange?

Take care of those missing observations for me without breaking the pipe:

```r
okcupid %>%
  na_if(-1) %>%
  select_if(is.numeric)
```

```
## # A tibble: 59,946 x 4
##      age height income height_cm
##    <dbl>  <dbl>  <dbl>     <dbl>
##  1    22     75     NA      190.
##  2    35     70  80000      178.
##  3    38     68     NA      173.
##  4    23     71  20000      180.
##  5    29     66     NA      168.
##  6    29     67     NA      170.
##  7    32     65     NA      165.
##  8    31     65     NA      165.
##  9    24     67     NA      170.
## 10    37     65     NA      165.
## # ... with 59,936 more rows
```

Transform all my numeric columns with `log`:

```
okcupid %>%
  na_if(-1) %>%
  select_if(is.numeric) %>%
  mutate_all(log)
```

```
## # A tibble: 59,946 x 4
##       age height income height_cm
##     <dbl>  <dbl>  <dbl>     <dbl>
##  1  3.09   4.32  NA          5.25
##  2  3.56   4.25  11.3        5.18
##  3  3.64   4.22  NA          5.15
##  4  3.14   4.26   9.90       5.19
##  5  3.37   4.19  NA          5.12
##  6  3.37   4.20  NA          5.14
##  7  3.47   4.17  NA          5.11
##  8  3.43   4.17  NA          5.11
##  9  3.18   4.20  NA          5.14
## 10  3.61   4.17  NA          5.11
## # ... with 59,936 more rows
```

Also see `mutate_if()`

Same but add `sqrt` and keep original columns:

```
okcupid %>%
  na_if(-1) %>%
  select_if(is.numeric) %>%
  mutate_all(list(logged = log, sqrted = sqrt))
```

```
## # A tibble: 59,946 x 12
##      age height income height_cm age_logged height_logged income_logged
##    <dbl>  <dbl>  <dbl>     <dbl>      <dbl>         <dbl>         <dbl>
##  1    22     75     NA      190.       3.09          4.32         NA
##  2    35     70  80000      178.       3.56          4.25         11.3
##  3    38     68     NA      173.       3.64          4.22         NA
##  4    23     71  20000      180.       3.14          4.26          9.90
##  5    29     66     NA      168.       3.37          4.19         NA
##  6    29     67     NA      170.       3.37          4.20         NA
##  7    32     65     NA      165.       3.47          4.17         NA
##  8    31     65     NA      165.       3.43          4.17         NA
##  9    24     67     NA      170.       3.18          4.20         NA
## 10    37     65     NA      165.       3.61          4.17         NA
## # ... with 59,936 more rows, and 5 more variables: height_cm_logged <dbl
## #   age_sqrted <dbl>, height_sqrted <dbl>, income_sqrted <dbl>,
## #   height_cm_sqrted <dbl>
```

Same but take care of zeros under `log`:

```
okcupid %>%
  na_if(-1) %>%
  select_if(is.numeric) %>%
  mutate_all(list(logged = function(x) log(x + 1), sqrted = sqrt))
```

```
## # A tibble: 59,946 x 12
##      age height income height_cm age_logged height_logged income_logged
##    <dbl>  <dbl>  <dbl>     <dbl>      <dbl>         <dbl>         <dbl>
##  1    22     75     NA      190.       3.14          4.33         NA
##  2    35     70  80000      178.       3.58          4.26         11.3
##  3    38     68     NA      173.       3.66          4.23         NA
##  4    23     71  20000      180.       3.18          4.28          9.90
##  5    29     66     NA      168.       3.40          4.20         NA
##  6    29     67     NA      170.       3.40          4.22         NA
##  7    32     65     NA      165.       3.50          4.19         NA
##  8    31     65     NA      165.       3.47          4.19         NA
##  9    24     67     NA      170.       3.22          4.22         NA
## 10    37     65     NA      165.       3.64          4.19         NA
## # ... with 59,936 more rows, and 5 more variables: height_cm_logged <dbl
## #   age_sqrted <dbl>, height_sqrted <dbl>, income_sqrted <dbl>,
## #   height_cm_sqrted <dbl>
```

Same but select only non-negative columns:

```r
is_non_negative <- function(x) is.numeric(x) && (is.na(x) || x >=

okcupid %>%
  na_if(-1) %>%
  select_if(is_non_negative) %>%
  mutate_all(list(logged = function(x) log(x + 1), sqrted = sqrt))
```

```
## # A tibble: 59,946 x 12
##      age height income height_cm age_logged height_logged income_logged
##    <dbl>  <dbl>  <dbl>     <dbl>      <dbl>         <dbl>         <dbl>
##  1    22     75     NA      190.       3.14          4.33         NA
##  2    35     70  80000      178.       3.58          4.26         11.3
##  3    38     68     NA      173.       3.66          4.23         NA
##  4    23     71  20000      180.       3.18          4.28          9.90
##  5    29     66     NA      168.       3.40          4.20         NA
##  6    29     67     NA      170.       3.40          4.22         NA
##  7    32     65     NA      165.       3.50          4.19         NA
##  8    31     65     NA      165.       3.47          4.19         NA
##  9    24     67     NA      170.       3.22          4.22         NA
## 10    37     65     NA      165.       3.64          4.19         NA
## # ... with 59,936 more rows, and 5 more variables: height_cm_logged <dbl
## #   age_sqrted <dbl>, height_sqrted <dbl>, income_sqrted <dbl>,
## #   height_cm_sqrted <dbl>
```

On second thought `log` would probably be appropriate just for `income` and `height_cm` (not really, just for demo):

```
okcupid %>%
  na_if(-1) %>%
  mutate_at(c("income", "height_cm"),
            list(logged = function(x) log(x + 1), sqrted = sqrt)) %>%
  select(ends_with("logged"), ends_with("sqrted"))
```

```
## # A tibble: 59,946 x 4
##    income_logged height_cm_logged income_sqrted height_cm_sqrted
##            <dbl>            <dbl>         <dbl>            <dbl>
##  1         NA                5.25           NA              13.8
##  2         11.3              5.19          283.             13.3
##  3         NA                5.16           NA              13.1
##  4          9.90             5.20          141.             13.4
##  5         NA                5.13           NA              12.9
##  6         NA                5.14           NA              13.0
##  7         NA                5.11           NA              12.8
##  8         NA                5.11           NA              12.8
##  9         NA                5.14           NA              13.0
## 10         NA                5.11           NA              12.8
## # ... with 59,936 more rows
```

Dealing with NAs

You've already seen `na_if()`. We could simply, always, keep those `NA`s in income:

```
okcupid <- okcupid %>%
  mutate(income = ifelse(income == -1, NA, income))
```

Or:

```
okcupid <- okcupid %>%
  mutate(income = na_if(income, -1))
```

Dropping `NA`s with, well, `drop_na()`:

```
okcupid_no_nas <- okcupid %>% drop_na()
```

Replacing `NA`s with, well, `replace_na()`:

```
okcupid_back_to_minus1 <- okcupid %>% replace_na(list(income = -1)
```

Could be useful for imputing `NA`s, say the median:

```
okcupid_na_income_imputed <- okcupid %>%
  replace_na(list(income = median(.$income, na.rm = TRUE)))
```

Sampling with `sample_n()` and `sample_frac()`

```
okcupid %>% select(drugs, age, income, sex) %>%
  group_by(drugs) %>%
  sample_n(3, replace = TRUE)
```

```
## # A tibble: 12 x 4
## # Groups:   drugs [4]
##    drugs       age income sex  
##    <chr>     <dbl>  <dbl> <chr>
##  1 never        24     NA f    
##  2 never        26     NA m    
##  3 never        29     NA m    
##  4 often        34     NA m    
##  5 often        19     NA f    
##  6 often        23     NA m    
##  7 sometimes    22     NA m    
##  8 sometimes    29 100000 m    
##  9 sometimes    32     NA m    
## 10 <NA>         25     NA f    
## 11 <NA>         27     NA m    
## 12 <NA>         40 250000 m    
```

Put it in a function

Applications

Of Data Science

Compose a function which would accept an unquoted variable

```r
count_var_for_gender <- function(var, gender) {
  okcupid %>%
    filter(sex == gender) %>%
    count({{var}}, sort = TRUE)
}

count_var_for_gender(body_type, "f") %>% head(9)
```

```
## # A tibble: 9 x 2
##   body_type          n
##   <chr>          <int>
## 1 average         5620
## 2 fit             4431
## 3 curvy           3811
## 4 <NA>            2703
## 5 thin            2469
## 6 athletic        2309
## 7 full figured     870
## 8 a little extra   821
## 9 skinny           601
```

Making a `data.frame` function pipeable

```
transform_all_my_numerics <- function(df, transformation) {
  df %>% mutate_if(is.numeric, transformation)
}

okcupid %>%
  transform_all_my_numerics(log) %>%
  select_if(is.numeric)
```

```
## # A tibble: 59,946 x 4
##      age height income height_cm
##    <dbl>  <dbl>  <dbl>     <dbl>
##  1  3.09   4.32  NA         5.25
##  2  3.56   4.25  11.3       5.18
##  3  3.64   4.22  NA         5.15
##  4  3.14   4.26   9.90      5.19
##  5  3.37   4.19  NA         5.12
##  6  3.37   4.20  NA         5.14
##  7  3.47   4.17  NA         5.11
##  8  3.43   4.17  NA         5.11
##  9  3.18   4.20  NA         5.14
## 10  3.61   4.17  NA         5.11
## # ... with 59,936 more rows
```

`invisible()`

If your function does not return a `data.frame` make it!

```r
print_n_rows <- function(df) {
  cat("number of rows: ", nrow(df), "\n")
  invisible(df)
}

okcupid %>%
  filter(sex == "m", body_type %in% c("fit", "thin", "skinny")) %>
  print_n_rows() %>%
  summarise(mean_height = mean(height_cm, trim = 0.025))
```

```
## number of rows:  11698

## # A tibble: 1 x 1
##   mean_height
##         <dbl>
## 1        179.
```

Or even better:

```r
filter_and_print <- function(df, ...) {
  df_filtered <- df %>% filter(...)
  cat("number of rows: ", nrow(df_filtered), "\n")
  df_filtered
}

okcupid %>%
  filter_and_print(sex == "m", body_type %in% c("fit", "thin", "sk
  summarise(mean_height = mean(height_cm, trim = 0.025))
```

```
## number of rows:  11698

## # A tibble: 1 x 1
##   mean_height
##         <dbl>
## 1        179.
```

for better living see `glue::glue("number of rows: {nrow(df)}")`