Subsetting Data in R

Data Wrangling in R

UFO data again

```
ufo <- read_csv("../data/ufo/ufo_data_complete.csv", col_ty
                 cols(
                   .default = col_character(),
                   `duration (seconds)` = col_double(),
                   longitude = col_double()
                 ))
head(ufo)
```

```
# A tibble: 6 x 11
  datetime     city  state country shape durat~1 durat~2 co
  <chr>        <chr> <chr> <chr>   <chr>   <dbl> <chr>   <c
1 10/10/1949 ~ san ~ tx    us      cyli~    2700 45 min~ Th
2 10/10/1949 ~ lack~ tx    <NA>    light    7200 1-2 hrs 19
3 10/10/1955 ~ ches~ <NA>  gb      circ~      20 20 sec~ Gr
4 10/10/1956 ~ edna  tx    us      circ~      20 1/2 ho~ My
5 10/10/1960 ~ kane~ hi    us      light     900 15 min~ AS
6 10/10/1961 ~ bris~ tn    us      sphe~     300 5 minu~ My
# ... with 1 more variable: longitude <dbl>, and abbreviate
#   1: `duration (seconds)`, 2: `duration (hours/min)`, 3:
```

Filtering and tibbles

The filter function automatically removes `NA` values.

```
count(ufo, country)
```

```
# A tibble: 6 x 2
  country     n
  <chr>   <int>
1 au        593
2 ca       3266
3 de        112
4 gb       2050
5 us      70293
6 <NA>    12561
```

```
ufo %>%filter(country == "de") %>% count(country)
```

```
# A tibble: 1 x 2
  country     n
  <chr>   <int>
1 de        112
```

Filtering for missing data

Missing value and filter can be powerful

```r
ufo %>%
  filter(is.na(state) & is.na(country)) %>%
  head()
```

```
# A tibble: 6 x 11
  datetime       city  state country shape durat~1 durat~2 co
  <chr>          <chr> <chr> <chr>   <chr>   <dbl> <chr>   <c
1 10/10/1973 ~ berm~ <NA>  <NA>    light      20 20 sec.  sa
2 10/10/1982 ~ gisb~ <NA>  <NA>    disk      120 2min    gi
3 10/10/1993 ~ zlat~ <NA>  <NA>    sphe~    1200 20 min~ I 
4 10/10/1996 ~ lake~ <NA>  <NA>    light     300 5 min   RE
5 10/10/1998 ~ turi~ <NA>  <NA>    disk       15 15 sec  We
6 10/10/1999 ~ dors~ <NA>  <NA>    flash       1 <1 sec~ Wh
# ... with 1 more variable: longitude <dbl>, and abbreviate
#   1: `duration (seconds)`, 2: `duration (hours/min)`, 3:
#   4: `date posted`, 5: latitude
```

Filtering and tibbles

Group logical statements with parentheses

```
ufo %>%
  filter(
    (!is.na(state) & is.na(country)) | city == "seattle") %
  head()
```

```
# A tibble: 6 x 11
  datetime      city  state country shape durat~1 durat~2 co
  <chr>         <chr> <chr> <chr>   <chr>   <dbl> <chr>   <c
1 10/10/1949 ~ lack~ tx    <NA>    light    7200 1-2 hrs 19
2 10/10/1974 ~ will~ az    <NA>    light     120 2 min   Th
3 10/10/1979 ~ sadd~ ab    <NA>    tria~     270 4.5 or~ Li
4 10/10/1986 ~ holm~ ny    <NA>    chev~     180 3 minu~ Fo
5 10/10/1987 ~ mani~ on    <NA>    disk      600 10/mins We
6 10/10/1989 ~ kran~ ky    <NA>    tria~     180 3min    Tr
# ... with 1 more variable: longitude <dbl>, and abbreviate
#   1: `duration (seconds)`, 2: `duration (hours/min)`, 3:
#   4: `date posted`, 5: latitude
```

Renaming Columns

Renaming Columns of a data.frame

To rename columns in dplyr, you use the rename command (NEW=old)

```
ufo <- ufo %>% rename(City = city, duration_s = `duration
glimpse(ufo)
```

```
Rows: 88,875
Columns: 11
$ datetime                   <chr> "10/10/1949 20:30", "10/10/1
$ City                       <chr> "san marcos", "lackland afb"
$ state                      <chr> "tx", "tx", NA, "tx", "hi",
$ country                    <chr> "us", NA, "gb", "us", "us",
$ shape                      <chr> "cylinder", "light", "circle
$ duration_s                 <dbl> 2700, 7200, 20, 20, 900, 300
$ `duration (hours/min)`     <chr> "45 minutes", "1-2 hrs", "20
$ comments                   <chr> "This event took place in ea
$ `date posted`              <chr> "4/27/2004", "12/16/2005", "
$ latitude                   <chr> "29.8830556", "29.38421", "5
$ longitude                  <dbl> -97.941111, -98.581082, -2.9
```

Renaming All Columns of a `data.frame`: dplyr

To rename all columns you use the `rename_with` command (with a function)

```
ufo_upper <- ufo %>% rename_with(toupper)
glimpse(ufo_upper)
```

```
Rows: 88,875
Columns: 11
$ DATETIME                <chr> "10/10/1949 20:30", "10/10/1
$ CITY                    <chr> "san marcos", "lackland afb"
$ STATE                   <chr> "tx", "tx", NA, "tx", "hi",
$ COUNTRY                 <chr> "us", NA, "gb", "us", "us",
$ SHAPE                   <chr> "cylinder", "light", "circle
$ DURATION_S              <dbl> 2700, 7200, 20, 20, 900, 300
$ `DURATION (HOURS/MIN)` <chr> "45 minutes", "1-2 hrs", "20
$ COMMENTS                <chr> "This event took place in ea
$ `DATE POSTED`           <chr> "4/27/2004", "12/16/2005", "
$ LATITUDE                <chr> "29.8830556", "29.38421", "5
$ LONGITUDE               <dbl> -97.941111, -98.581082, -2.9
```

Adding columns to a `data.frame`

`mutate` - allows you to add or replace columns (IMPORTANT: need to reassign for it to stick!)

```
ufo2 <- ufo %>% mutate(State = toupper(state))
ufo2 %>% glimpse()
```

```
Rows: 88,875
Columns: 12
$ datetime               <chr> "10/10/1949 20:30", "10/10/1
$ City                   <chr> "san marcos", "lackland afb"
$ state                  <chr> "tx", "tx", NA, "tx", "hi",
$ country                <chr> "us", NA, "gb", "us", "us",
$ shape                  <chr> "cylinder", "light", "circle
$ duration_s             <dbl> 2700, 7200, 20, 20, 900, 300
$ `duration (hours/min)` <chr> "45 minutes", "1-2 hrs", "20
$ comments               <chr> "This event took place in ea
$ `date posted`          <chr> "4/27/2004", "12/16/2005", "
$ latitude               <chr> "29.8830556", "29.38421", "5
$ longitude              <dbl> -97.941111, -98.581082, -2.9
```

Creating conditional variables

One frequently-used tool is creating variables with conditions.

A general function for creating new variables based on existing variables is the `ifelse()` function, which “returns a value with the same shape as test which is filled with elements selected from either yes or no depending on whether the element of test is `TRUE` or `FALSE`.”

```
ifelse(test, yes, no)

# test: an object which can be coerced
    to logical mode.
# yes: return values for true elements of test.
# no: return values for false elements of test.
```

Recoding to missing

Sometimes people code missing data in weird or inconsistent ways.

```r
ages <-  data.frame(age = c(23,-999,21,44,32,57,65,54))
range(ages$age)
```

```
[1] -999   65
```

Recoding to create new column

Say we want to make a new column about if the age was over 30?

```
pull(ages, age)
```

```
[1]   23 -999   21   44   32   57   65   54
```

```
ages <- ages %>% mutate(over_20 = ifelse(age > 30, "Yes", '
ages
```

```
   age over_20
1   23      No
2 -999      No
3   21      No
4   44     Yes
5   32     Yes
6   57     Yes
7   65     Yes
8   54     Yes
```

Recoding value to missing

How do we change the -999 to be treated as missing for the age column?

```
pull(ages, age)
```

```
[1]   23 -999   21   44   32   57   65   54
```

```
ages <- ages %>% mutate(age = ifelse(age == -999, NA, age))
range(ages$age)
```

```
[1] NA NA
```

```
range(ages$age, na.rm=TRUE)
```

```
[1] 21 65
```

```
pull(ages, age)
```

```
[1] 23 NA 21 44 32 57 65 54
```

Adding columns to a data.frame: dplyr

```r
ufo <- ufo %>% mutate(
              region = ifelse(
              country %in% c("us", "ca"),
              "North America",
              "Not North America")
            )
ufo %>% select(country, region) %>% head()
```

```
# A tibble: 6 x 2
  country region
  <chr>   <chr>
1 us      North America
2 <NA>    Not North America
3 gb      Not North America
4 us      North America
5 us      North America
6 us      North America
```

case_when provides a more general way

```
casewhen(test ~ value if test is true,
         test2 ~ vlue if test2 is true,
         TRUE ~ value if all above tests are not true) # de
```

```r
ufo <- ufo %>% mutate(
            region = case_when(
              country %in% c("us", "ca") ~ "North America",
              country %in% c("de") ~ "Europe",
              country %in% "gb" ~ "Great Britain",
            ))

ufo %>% select(country, region) %>% head()
```

```
# A tibble: 6 x 2
  country region
  <chr>   <chr>
1 us      North America
2 <NA>    <NA>
3 gb      Great Britain
```

case_when defaults to NA when all tests fail

```r
ufo <- ufo %>% mutate(
           region = case_when(
             country %in% c("us", "ca") ~ "North America",
             country %in% c("de") ~ "Europe"
             #country %in% "gb" ~ "Great Britain",
           ))

ufo %>% select(country, region) %>% head()
```

```
# A tibble: 6 x 2
  country region
  <chr>   <chr>
1 us      North America
2 <NA>    <NA>
3 gb      <NA>
4 us      North America
5 us      North America
6 us      North America
```

case_when() with value if all tests fail

```r
ufo <- ufo %>% mutate(
           region = case_when(
             country %in% c("us", "ca") ~ "North America",
             country %in% c("de") ~ "Europe",
             country %in% "gb" ~ "Great Britain",
             TRUE ~ "Other"
           ))
ufo %>% select(country, region) %>% head()
```

```
# A tibble: 6 x 2
  country region
  <chr>   <chr>
1 us      North America
2 <NA>    Other
3 gb      Great Britain
4 us      North America
5 us      North America
6 us      North America
```

case_when() with value if all tests fail (use a variable!)

```
ufo <- ufo %>% mutate(
            region = case_when(
              country %in% "gb" ~ "Great Brit.",
              TRUE ~ region
            ))
ufo %>% select(country, region) %>% head()
```

```
# A tibble: 6 x 2
  country region
  <chr>   <chr>
1 us      North America
2 <NA>    Other
3 gb      Great Brit.
4 us      North America
5 us      North America
6 us      North America
```

Ordering the rows of a data.frame: dplyr

The arrange function can reorder rows By default, arrange orders in ascending order:

```
ufo %>% arrange(duration_s)
```

```
# A tibble: 88,875 x 12
   datetime    City  state country shape durat~1 durat~2 co
   <chr>       <chr> <chr> <chr>   <chr>   <dbl> <chr>   <c
 1 10/10/1995~ puer~ pr    <NA>    <NA>        0 <NA>    Wo
 2 10/10/1999~ ashl~ mo    us      light       0 two se~ We
 3 10/10/2002~ baha~ <NA>  <NA>    egg         0 <NA>    we
 4 10/10/2002~ burn~ <NA>  au      cross       0 12      th
 5 10/10/2005~ edge~ fl    us      <NA>        0 300     or
 6 10/10/2005~ fran~ in    us      disk        0 ?       tw
 7 10/10/2006~ knik  ak    us      tria~       0 5       SI
 8 10/10/2007~ bake~ ca    us      circ~       0 had a ~ UF
 9 10/10/2008~ amar~ tx    us      flash       0 <NA>    we
10 10/10/2009~ gree~ <NA>  <NA>    rect~       0 <NA>    Fo
# ... with 88,865 more rows, 2 more variables: longitude <c
```

Ordering the rows of a data.frame: dplyr

Use the desc to arrange the rows in descending order:

```
ufo %>% arrange(desc(duration_s))
```

```
# A tibble: 88,875 x 12
   datetime     City  state country shape durat~1 durat~2 co
   <chr>        <chr> <chr> <chr>   <chr>   <dbl> <chr>   <c
 1 10/1/1983 ~ birm~ <NA>  gb      sphe~  9.78e7 31 yea~ Fi
 2 6/3/2010 2~ otta~ on    ca      other  8.28e7 23000h~ ((
 3 9/15/1991 ~ gree~ ar    us      light  6.63e7 21 yea~ Or
 4 4/2/1983 2~ dont~ <NA>  <NA>    <NA>   5.26e7 2 mont~ Hi
 5 8/10/2012 ~ finl~ wa    us      light  5.26e7 2 mont~ Th
 6 8/24/2002 ~ engl~ fl    us      light  5.26e7 2 mont~ br
 7 6/30/1969 ~ some~ <NA>  gb      cone   2.52e7 8 years Fi
 8 10/7/2013 ~ okla~ ok    <NA>    circ~  1.05e7 4 mont~ Br
 9 3/1/1994 0~ meni~ ca    us      unkn~  1.05e7 4 mont~ Su
10 8/3/2008 2~ virg~ va    us      fire~  1.05e7 4 mont~ th
# ... with 88,865 more rows, 2 more variables: longitude <c
#   and abbreviated variable names 1: duration_s, 2: `durat
```

Ordering the rows of a data.frame: dplyr

It is a bit more straightforward to mix increasing and decreasing orderings:

```
ufo %>% arrange(country, desc(duration_s))
```

```
# A tibble: 88,875 x 12
   datetime    City  state country shape durat~1 durat~2 co
   <chr>       <chr> <chr> <chr>   <chr>   <dbl> <chr>   <c
 1 11/12/2013~ moun~ <NA>  au      sphe~ 1209600 2 weeks Or
 2 5/12/2004 ~ sydn~ <NA>  au      light  345600 4 days+ In
 3 4/18/2004 ~ sydn~ <NA>  au      light   86400 day     It
 4 4/15/1983 ~ bris~ <NA>  au      chan~   37800 1 1/2 ~ A 
 5 4/18/1996 ~ bris~ <NA>  au      <NA>    18000 5  hou~ Fi
 6 6/9/2005 2~ melb~ <NA>  au      circ~   18000 5 hour~ UF
 7 11/6/2009 ~ pert~ <NA>  au      light   14400 4hrs    Un
 8 3/15/2004 ~ adel~ <NA>  au      form~   10800 1-3 hrs iv
 9 3/2/2014 2~ pert~ <NA>  au      light   10800 2-3 ho~ Co
10 6/20/2001 ~ canb~ <NA>  au      tear~   10800 3 hrs   8 
# ... with 88,865 more rows, 2 more variables: longitude <c
```

Lab

Link to Lab