STA 326 2.0 Programming and Data Analysis with R

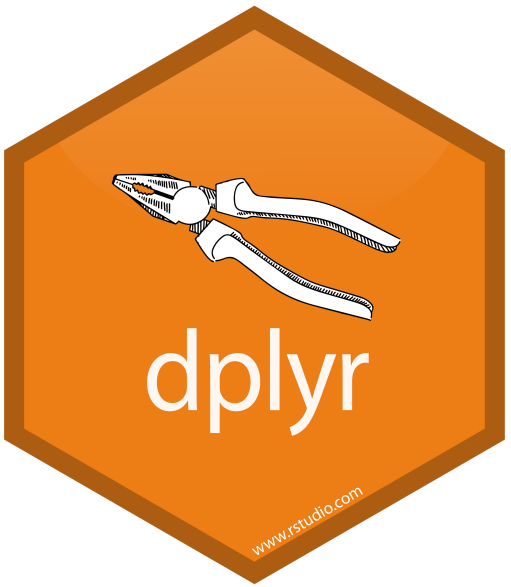
Data Manipulation with dplyr

Dr Thiyanga Talagala

Online distance learning/teaching materials during the COVID-19 outbreak.

packages

```
library(tidyverse) # TO obtain dplyr
library(magrittr)
```



Dataset

```
library(gapminder)
str(gapminder)
```

```
tibble [1,704 × 6] (S3: tbl_df/tbl/data.frame)
 $ country  : Factor w/ 142 levels "Afghanistan",..: 1 1 1 1 1 1 1 1 1 1 ...
 $ continent: Factor w/ 5 levels "Africa","Americas",..: 3 3 3 3 3 3 3 3 3 3 ...
 $ year     : int [1:1704] 1952 1957 1962 1967 1972 1977 1982 1987 1992 1997 ...
 $ lifeExp  : num [1:1704] 28.8 30.3 32 34 36.1 ...
 $ pop      : int [1:1704] 8425333 9240934 10267083 11537966 13079460 14880372 1288
 $ gdpPercap: num [1:1704] 779 821 853 836 740 ...
```

```
head(gapminder)
```

```
# A tibble: 6 x 6
  country     continent  year lifeExp      pop gdpPercap
  <fct>       <fct>     <int>   <dbl>    <int>     <dbl>
1 Afghanistan Asia       1952    28.8  8425333      779.
2 Afghanistan Asia       1957    30.3  9240934      821.
3 Afghanistan Asia       1962    32.0 10267083      853.
4 Afghanistan Asia       1967    34.0 11537966      836.
5 Afghanistan Asia       1972    36.1 13079460      740.
6 Afghanistan Asia       1977    38.4 14880372      786.
```

Dataset (cont.)

```
glimpse(gapminder)
```

```
Rows: 1,704
Columns: 6
$ country   <fct> Afghanistan, Afghanistan, Afghanistan, Afghanistan, Afghani…
$ continent <fct> Asia, Asia, Asia, Asia, Asia, Asia, Asia, Asia, Asia, Asia,…
$ year      <int> 1952, 1957, 1962, 1967, 1972, 1977, 1982, 1987, 1992, 1997,…
$ lifeExp   <dbl> 28.801, 30.332, 31.997, 34.020, 36.088, 38.438, 39.854, 40.…
$ pop       <int> 8425333, 9240934, 10267083, 11537966, 13079460, 14880372, 1…
$ gdpPercap <dbl> 779.4453, 820.8530, 853.1007, 836.1971, 739.9811, 786.1134,…
```

Dataset (cont.)

```
tbl_df(gapminder)
```

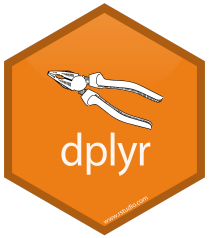
```
# A tibble: 1,704 x 6
   country     continent  year lifeExp      pop gdpPercap
   <fct>       <fct>     <int>   <dbl>    <int>     <dbl>
 1 Afghanistan Asia       1952    28.8  8425333      779.
 2 Afghanistan Asia       1957    30.3  9240934      821.
 3 Afghanistan Asia       1962    32.0 10267083      853.
 4 Afghanistan Asia       1967    34.0 11537966      836.
 5 Afghanistan Asia       1972    36.1 13079460      740.
 6 Afghanistan Asia       1977    38.4 14880372      786.
 7 Afghanistan Asia       1982    39.9 12881816      978.
 8 Afghanistan Asia       1987    40.8 13867957      852.
 9 Afghanistan Asia       1992    41.7 16317921      649.
10 Afghanistan Asia       1997    41.8 22227415      635.
# … with 1,694 more rows
```

Dataset (cont.)

```
library(skimr)
skim(gapminder)
```

```
── Data Summary ────────────────────────
                           Values
Name                       gapminder
Number of rows             1704
Number of columns          6
_______________________
Column type frequency:
  factor                   2
  numeric                  4
________________________
Group variables            None

── Variable type: factor ─────────────────────────────────────────────────────
  skim_variable n_missing complete_rate ordered n_unique
1 country               0             1 FALSE        142
2 continent             0             1 FALSE          5
  top_counts
1 Afg: 12, Alb: 12, Alg: 12, Ang: 12
2 Afr: 624, Asi: 396, Eur: 360, Ame: 300

── Variable type: numeric ────────────────────────────────────────────────────
  skim_variable n_missing complete_rate       mean          sd      p0       p25
1 year                  0             1     1980.         17.3  1952      1966.
2 lifeExp               0             1       59.5        12.9    23.6      48.2
3 pop                   0             1 29601212.  106157897.  60011   2793664
4 gdpPercap             0             1     7215.        9857.    241.     1202.
        p50        p75         p100 hist
1    1980.      1993.        2007   ▇▇▇▇▇
2      60.7       70.8         82.6 ▁▂▆▇▇
3 7023596.  19585222.  1318683096   ▇▁▁▁▁
4    3532.      9325.      113523.  ▇▁▁▁▁
```

dplyr verbs



- filter
- select
- mutate
- summarise
- arrange
- group_by
- rename

filter

- Picks observations by their values.
- Takes logical expressions and returns the rows for which all are `TRUE`.

```
filter(gapminder, lifeExp < 50)
```

```
# A tibble: 491 x 6
   country     continent  year lifeExp      pop gdpPercap
   <fct>       <fct>     <int>   <dbl>    <int>     <dbl>
 1 Afghanistan Asia       1952    28.8  8425333      779.
 2 Afghanistan Asia       1957    30.3  9240934      821.
 3 Afghanistan Asia       1962    32.0 10267083      853.
 4 Afghanistan Asia       1967    34.0 11537966      836.
 5 Afghanistan Asia       1972    36.1 13079460      740.
 6 Afghanistan Asia       1977    38.4 14880372      786.
 7 Afghanistan Asia       1982    39.9 12881816      978.
 8 Afghanistan Asia       1987    40.8 13867957      852.
 9 Afghanistan Asia       1992    41.7 16317921      649.
10 Afghanistan Asia       1997    41.8 22227415      635.
# … with 481 more rows
```

filter (cont)


```
filter(gapminder, country == "Sri Lanka")
```

```
# A tibble: 12 x 6
   country   continent  year lifeExp      pop gdpPercap
   <fct>     <fct>     <int>   <dbl>    <int>     <dbl>
 1 Sri Lanka Asia       1952    57.6  7982342     1084.
 2 Sri Lanka Asia       1957    61.5  9128546     1073.
 3 Sri Lanka Asia       1962    62.2 10421936     1074.
 4 Sri Lanka Asia       1967    64.3 11737396     1136.
 5 Sri Lanka Asia       1972    65.0 13016733     1213.
 6 Sri Lanka Asia       1977    65.9 14116836     1349.
 7 Sri Lanka Asia       1982    68.8 15410151     1648.
 8 Sri Lanka Asia       1987    69.0 16495304     1877.
 9 Sri Lanka Asia       1992    70.4 17587060     2154.
10 Sri Lanka Asia       1997    70.5 18698655     2664.
11 Sri Lanka Asia       2002    70.8 19576783     3015.
12 Sri Lanka Asia       2007    72.4 20378239     3970.
```

```
# gapminder %>% filter(country == "Sri Lanka")
```

filter (cont)


```
filter(gapminder, country %in% c("Sri Lanka", "Australia"))
```

```
# A tibble: 24 x 6
   country   continent  year lifeExp      pop gdpPercap
   <fct>     <fct>     <int>   <dbl>    <int>     <dbl>
 1 Australia Oceania    1952    69.1  8691212    10040.
 2 Australia Oceania    1957    70.3  9712569    10950.
 3 Australia Oceania    1962    70.9 10794968    12217.
 4 Australia Oceania    1967    71.1 11872264    14526.
 5 Australia Oceania    1972    71.9 13177000    16789.
 6 Australia Oceania    1977    73.5 14074100    18334.
 7 Australia Oceania    1982    74.7 15184200    19477.
 8 Australia Oceania    1987    76.3 16257249    21889.
 9 Australia Oceania    1992    77.6 17481977    23425.
10 Australia Oceania    1997    78.8 18565243    26998.
# … with 14 more rows
```

filter (cont)


```
filter(gapminder, country %in% c("Sri Lanka", "Australia")) %>%
  head()
```

```
# A tibble: 6 x 6
  country   continent  year lifeExp      pop gdpPercap
  <fct>     <fct>     <int>   <dbl>    <int>     <dbl>
1 Australia Oceania    1952    69.1  8691212    10040.
2 Australia Oceania    1957    70.3  9712569    10950.
3 Australia Oceania    1962    70.9 10794968    12217.
4 Australia Oceania    1967    71.1 11872264    14526.
5 Australia Oceania    1972    71.9 13177000    16789.
6 Australia Oceania    1977    73.5 14074100    18334.
```

```
filter(gapminder, country %in% c("Sri Lanka", "Australia")) %>%
  tail()
```

```
# A tibble: 6 x 6
  country   continent  year lifeExp      pop gdpPercap
  <fct>     <fct>     <int>   <dbl>    <int>     <dbl>
1 Sri Lanka Asia       1982    68.8 15410151     1648.
2 Sri Lanka Asia       1987    69.0 16495304     1877.
3 Sri Lanka Asia       1992    70.4 17587060     2154.
4 Sri Lanka Asia       1997    70.5 18698655     2664.
5 Sri Lanka Asia       2002    70.8 19576783     3015.
6 Sri Lanka Asia       2007    72.4 20378239     3970.
```

select

- Picks variables by their names.

```
head(gapminder, 3)
```

```
# A tibble: 3 x 6
  country     continent  year lifeExp      pop gdpPercap
  <fct>       <fct>     <int>   <dbl>    <int>     <dbl>
1 Afghanistan Asia       1952    28.8  8425333      779.
2 Afghanistan Asia       1957    30.3  9240934      821.
3 Afghanistan Asia       1962    32.0 10267083      853.
```

```
select(gapminder, year:gdpPercap)
```

```
# A tibble: 1,704 x 4
    year lifeExp      pop gdpPercap
   <int>   <dbl>    <int>     <dbl>
 1  1952    28.8  8425333      779.
 2  1957    30.3  9240934      821.
 3  1962    32.0 10267083      853.
 4  1967    34.0 11537966      836.
 5  1972    36.1 13079460      740.
 6  1977    38.4 14880372      786.
 7  1982    39.9 12881816      978.
 8  1987    40.8 13867957      852.
 9  1992    41.7 16317921      649.
10  1997    41.8 22227415      635.
```

select (cont.)


```
head(gapminder, 3)
```

```
# A tibble: 3 x 6
  country     continent  year lifeExp      pop gdpPercap
  <fct>       <fct>     <int>   <dbl>    <int>     <dbl>
1 Afghanistan Asia       1952    28.8  8425333      779.
2 Afghanistan Asia       1957    30.3  9240934      821.
3 Afghanistan Asia       1962    32.0 10267083      853.
```

```
select(gapminder, year, gdpPercap)
```

```
# A tibble: 1,704 x 2
    year gdpPercap
   <int>     <dbl>
 1  1952      779.
 2  1957      821.
 3  1962      853.
 4  1967      836.
 5  1972      740.
 6  1977      786.
 7  1982      978.
 8  1987      852.
 9  1992      649.
10  1997      635.
# … with 1,694 more rows
```

select (cont.)


```
head(gapminder, 3)
```

```
# A tibble: 3 x 6
  country     continent  year lifeExp      pop gdpPercap
  <fct>       <fct>     <int>   <dbl>    <int>     <dbl>
1 Afghanistan Asia       1952    28.8  8425333      779.
2 Afghanistan Asia       1957    30.3  9240934      821.
3 Afghanistan Asia       1962    32.0 10267083      853.
```

```
select(gapminder, -c(year, gdpPercap))
```

```
# A tibble: 1,704 x 4
   country     continent lifeExp      pop
   <fct>       <fct>       <dbl>    <int>
 1 Afghanistan Asia         28.8  8425333
 2 Afghanistan Asia         30.3  9240934
 3 Afghanistan Asia         32.0 10267083
 4 Afghanistan Asia         34.0 11537966
 5 Afghanistan Asia         36.1 13079460
 6 Afghanistan Asia         38.4 14880372
 7 Afghanistan Asia         39.9 12881816
 8 Afghanistan Asia         40.8 13867957
 9 Afghanistan Asia         41.7 16317921
10 Afghanistan Asia         41.8 22227415
# … with 1,694 more rows
```

select (cont.)


```
head(gapminder, 3)
```

```
# A tibble: 3 x 6
  country     continent  year lifeExp      pop gdpPercap
  <fct>       <fct>     <int>   <dbl>    <int>     <dbl>
1 Afghanistan Asia       1952    28.8  8425333      779.
2 Afghanistan Asia       1957    30.3  9240934      821.
3 Afghanistan Asia       1962    32.0 10267083      853.
```

```
select(gapminder, -(year:gdpPercap))
```

```
# A tibble: 1,704 x 2
   country     continent
   <fct>       <fct>
 1 Afghanistan Asia
 2 Afghanistan Asia
 3 Afghanistan Asia
 4 Afghanistan Asia
 5 Afghanistan Asia
 6 Afghanistan Asia
 7 Afghanistan Asia
 8 Afghanistan Asia
 9 Afghanistan Asia
10 Afghanistan Asia
# … with 1,694 more rows
```

mutate

- Creates new variables with functions of existing variables

```
gapminder %>%
mutate(gdp = pop * gdpPercap)
```

```
# A tibble: 1,704 x 7
   country     continent  year lifeExp      pop gdpPercap          gdp
   <fct>       <fct>     <int>   <dbl>    <int>     <dbl>        <dbl>
 1 Afghanistan Asia       1952    28.8  8425333      779.  6567086330.
 2 Afghanistan Asia       1957    30.3  9240934      821.  7585448670.
 3 Afghanistan Asia       1962    32.0 10267083      853.  8758855797.
 4 Afghanistan Asia       1967    34.0 11537966      836.  9648014150.
 5 Afghanistan Asia       1972    36.1 13079460      740.  9678553274.
 6 Afghanistan Asia       1977    38.4 14880372      786. 11697659231.
 7 Afghanistan Asia       1982    39.9 12881816      978. 12598563401.
 8 Afghanistan Asia       1987    40.8 13867957      852. 11820990309.
 9 Afghanistan Asia       1992    41.7 16317921      649. 10595901589.
10 Afghanistan Asia       1997    41.8 22227415      635. 14121995875.
# … with 1,694 more rows
```

`summarise` (British) or `summarize` (US)

- Collapse many values down to a single summary

```
gapminder %>%
  summarise(
    lifeExp_mean=mean(lifeExp),
    pop_mean=mean(pop),
    gdpPercap_mean=mean(gdpPercap))
```

```
# A tibble: 1 x 3
  lifeExp_mean  pop_mean gdpPercap_mean
         <dbl>     <dbl>          <dbl>
1         59.5 29601212.          7215.
```

arrange

- Reorder the rows

```
arrange(gapminder, desc(lifeExp))
```

```
# A tibble: 1,704 x 6
   country          continent  year lifeExp       pop gdpPercap
   <fct>            <fct>     <int>   <dbl>     <int>     <dbl>
 1 Japan            Asia       2007    82.6 127467972    31656.
 2 Hong Kong, China Asia       2007    82.2   6980412    39725.
 3 Japan            Asia       2002    82   127065841    28605.
 4 Iceland          Europe     2007    81.8    301931    36181.
 5 Switzerland      Europe     2007    81.7   7554661    37506.
 6 Hong Kong, China Asia       2002    81.5   6762476    30209.
 7 Australia        Oceania    2007    81.2  20434176    34435.
 8 Spain            Europe     2007    80.9  40448191    28821.
 9 Sweden           Europe     2007    80.9   9031088    33860.
10 Israel           Asia       2007    80.7   6426679    25523.
# … with 1,694 more rows
```

group_by

- Takes an existing tibble and converts it into a grouped tibble where operations are performed "by group". ungroup() removes grouping.

```
Japan_SL <- filter(gapminder, country %in% c("Japan", "Sri Lanka"))
Japan_SL %>% head()
```

```
# A tibble: 6 x 6
  country continent  year lifeExp       pop gdpPercap
  <fct>   <fct>     <int>   <dbl>     <int>     <dbl>
1 Japan   Asia       1952    63.0  86459025     3217.
2 Japan   Asia       1957    65.5  91563009     4318.
3 Japan   Asia       1962    68.7  95831757     6577.
4 Japan   Asia       1967    71.4 100825279     9848.
5 Japan   Asia       1972    73.4 107188273    14779.
6 Japan   Asia       1977    75.4 113872473    16610.
```

```
Japan_SL_grouped <- Japan_SL %>% group_by(country)
Japan_SL_grouped
```

```
# A tibble: 24 x 6
# Groups:   country [2]
   country continent  year lifeExp       pop gdpPercap
   <fct>   <fct>     <int>   <dbl>     <int>     <dbl>
 1 Japan   Asia       1952    63.0  86459025     3217.
 2 Japan   Asia       1957    65.5  91563009     4318.
```

group_by (cont.)


```
Japan_SL %>% summarise(mean_lifeExp=mean(lifeExp))
```

```
# A tibble: 1 x 1
  mean_lifeExp
         <dbl>
1         70.7
```

```
Japan_SL_grouped %>% summarise(mean_lifeExp=mean(lifeExp))
```

```
# A tibble: 2 x 2
  country   mean_lifeExp
  <fct>            <dbl>
1 Japan             74.8
2 Sri Lanka         66.5
```

rename

- Rename variables

```
head(gapminder, 3)
```

```
# A tibble: 3 x 6
  country     continent  year lifeExp      pop gdpPercap
  <fct>       <fct>     <int>   <dbl>    <int>     <dbl>
1 Afghanistan Asia       1952    28.8  8425333      779.
2 Afghanistan Asia       1957    30.3  9240934      821.
3 Afghanistan Asia       1962    32.0 10267083      853.
```

```
rename(gapminder,
       `life expectancy`=lifeExp,
       population=pop) # new_name = old_name
```

```
# A tibble: 1,704 x 6
   country     continent  year `life expectancy` population gdpPercap
   <fct>       <fct>     <int>             <dbl>      <int>     <dbl>
 1 Afghanistan Asia       1952              28.8    8425333      779.
 2 Afghanistan Asia       1957              30.3    9240934      821.
 3 Afghanistan Asia       1962              32.0   10267083      853.
 4 Afghanistan Asia       1967              34.0   11537966      836.
 5 Afghanistan Asia       1972              36.1   13079460      740.
 6 Afghanistan Asia       1977              38.4   14880372      786.
 7 Afghanistan Asia       1982              39.9   12881816      978.
 8 Afghanistan Asia       1987              40.8   13867957      852.
```

Combine multiple operations


```
gapminder %>%
filter(country == 'China') %>% head(2)
```

```
# A tibble: 2 x 6
  country continent  year lifeExp       pop gdpPercap
  <fct>   <fct>     <int>   <dbl>     <int>     <dbl>
1 China   Asia       1952    44   556263527      400.
2 China   Asia       1957    50.5 637408000      576.
```

```
gapminder %>%
filter(country == 'China') %>% summarise(lifemax=max(lifeExp))
```

```
# A tibble: 1 x 1
  lifemax
    <dbl>
1    73.0
```

```
gapminder %>%
filter(country == 'China') %>%
filter(lifeExp == max(lifeExp))
```

```
# A tibble: 1 x 6
  country continent  year lifeExp        pop gdpPercap
  <fct>   <fct>     <int>   <dbl>      <int>     <dbl>
1 China   Asia       2007    73.0 1318683096     4959.
```

Combine multiple operations


```
gapminder %>%
filter(continent == 'Asia') %>%
group_by(country) %>%
filter(lifeExp == max(lifeExp)) %>%
arrange(desc(year))
```

```
# A tibble: 33 x 6
# Groups:   country [33]
   country          continent  year lifeExp        pop gdpPercap
   <fct>            <fct>     <int>   <dbl>      <int>     <dbl>
 1 Afghanistan      Asia       2007    43.8   31889923      975.
 2 Bahrain          Asia       2007    75.6     708573    29796.
 3 Bangladesh       Asia       2007    64.1  150448339     1391.
 4 Cambodia         Asia       2007    59.7   14131858     1714.
 5 China            Asia       2007    73.0 1318683096     4959.
 6 Hong Kong, China Asia       2007    82.2    6980412    39725.
 7 India            Asia       2007    64.7 1110396331     2452.
 8 Indonesia        Asia       2007    70.6  223547000     3541.
 9 Iran             Asia       2007    71.0   69453570    11606.
10 Israel           Asia       2007    80.7    6426679    25523.
# … with 23 more rows
```

Combine Data Sets

Combine Data Sets

Mutating joins

- `left_join`
- `right_join`
- `inner_join`
- `full_join`

Set operations

- `intersect`
- `union`

Binding

- `bind_rows`
- `bind_cols`

left_join

```
first <- tibble(x1=c("A", "B", "C"), x2=c(1, 2, 3))
second <- tibble(x1=c("A", "B", "D"), x3=c("red", "yellow" , "green"))
```

```
first
```

```
# A tibble: 3 x 2
  x1       x2
  <chr> <dbl>
1 A         1
2 B         2
3 C         3
```

```
second
```

```
# A tibble: 3 x 2
  x1    x3
  <chr> <chr>
1 A     red
2 B     yellow
3 D     green
```

```
left_join(first, second, by="x1")
```

```
# A tibble: 3 x 3
  x1       x2 x3
  <chr> <dbl> <chr>
1 A         1 red
2 B         2 yellow
3 C         3 <NA>
```

right_join

```
first <- tibble(x1=c("A", "B", "C"), x2=c(1, 2, 3))
second <- tibble(x1=c("A", "B", "D"), x3=c("red", "yellow" , "green"))
```

```
first
```

```
# A tibble: 3 x 2
  x1       x2
  <chr> <dbl>
1 A         1
2 B         2
3 C         3
```

```
second
```

```
# A tibble: 3 x 2
  x1    x3
  <chr> <chr>
1 A     red
2 B     yellow
3 D     green
```

```
right_join(first, second, by="x1")
```

```
# A tibble: 3 x 3
  x1       x2 x3
  <chr> <dbl> <chr>
1 A         1 red
2 B         2 yellow
3 D        NA green
```

inner_join

```
first <- tibble(x1=c("A", "B", "C"), x2=c(1, 2, 3))
second <- tibble(x1=c("A", "B", "D"), x3=c("red", "yellow" , "green"))
```

```
first
```

```
# A tibble: 3 x 2
  x1       x2
  <chr> <dbl>
1 A         1
2 B         2
3 C         3
```

```
second
```

```
# A tibble: 3 x 2
  x1    x3
  <chr> <chr>
1 A     red
2 B     yellow
3 D     green
```

```
inner_join(first, second, by="x1")
```

```
# A tibble: 2 x 3
  x1       x2 x3
  <chr> <dbl> <chr>
1 A         1 red
2 B         2 yellow
```

full_join

```
first <- tibble(x1=c("A", "B", "C"), x2=c(1, 2, 3))
second <- tibble(x1=c("A", "B", "D"), x3=c("red", "yellow" , "green"))
```

```
first
```

```
# A tibble: 3 x 2
  x1       x2
  <chr> <dbl>
1 A         1
2 B         2
3 C         3
```

```
second
```

```
# A tibble: 3 x 2
  x1    x3
  <chr> <chr>
1 A     red
2 B     yellow
3 D     green
```

```
full_join(first, second, by="x1")
```

```
# A tibble: 4 x 3
  x1       x2 x3
  <chr> <dbl> <chr>
1 A         1 red
2 B         2 yellow
3 C         3 <NA>
4 D        NA green
```

Set operations

```
first <- tibble(x1=c("A", "B", "C"), x2=c(1, 2, 3))
second <- tibble(x1=c("D", "B", "C"), x2=c(10, 2, 3))
```

Two compatible data sets. Column names are the same.

```
first
```

```
# A tibble: 3 x 2
  x1       x2
  <chr> <dbl>
1 A         1
2 B         2
3 C         3
```

```
second
```

```
# A tibble: 3 x 2
  x1       x2
  <chr> <dbl>
1 D        10
2 B         2
3 C         3
```

intersect

```
intersect(first, second)
```

```
# A tibble: 2 x 2
  x1       x2
  <chr> <dbl>
1 B         2
2 C         3
```

union

```
union(first, second)
```

```
# A tibble: 4 x 2
  x1       x2
  <chr> <dbl>
1 A         1
2 B         2
3 C         3
4 D        10
```

Set operations (cont.)

```
first <- tibble(x1=c("A", "B", "C"), x2=c(1, 2, 3))
second <- tibble(x1=c("D", "B", "C"), x2=c(10, 20, 30))
```

Two compatible data sets. Column names are the same.

```
first
```

```
# A tibble: 3 x 2
  x1       x2
  <chr> <dbl>
1 A         1
2 B         2
3 C         3
```

```
second
```

```
# A tibble: 3 x 2
  x1       x2
  <chr> <dbl>
1 D        10
2 B        20
3 C        30
```

intersect

```
intersect(first, second)
```

```
# A tibble: 0 x 2
# … with 2 variables: x1 <chr>, x2 <dbl>
```

union

```
union(first, second)
```

```
# A tibble: 6 x 2
  x1       x2
  <chr> <dbl>
1 A         1
2 B         2
3 C         3
4 D        10
5 B        20
6 C        30
```

Binding

```
first <- tibble(x1=c("A", "B", "C"), x2=c(1, 2, 3))
second <- tibble(x1=c("D", "B", "C"), x2=c(10, 20, 30))
```

```
first
```

```
# A tibble: 3 x 2
  x1       x2
  <chr> <dbl>
1 A         1
2 B         2
3 C         3
```

```
second
```

```
# A tibble: 3 x 2
  x1       x2
  <chr> <dbl>
1 D        10
2 B        20
3 C        30
```

bind_rows

```
bind_rows(first, second)
```

```
# A tibble: 6 x 2
  x1       x2
  <chr> <dbl>
1 A         1
2 B         2
3 C         3
4 D        10
5 B        20
6 C        30
```

Binding (cont.)

```
first <- tibble(x1=c("A", "B", "C"), x2=c(1, 2, 3))
second <- tibble(x1=c("D", "B", "C"), x2=c(10, 20, 30))
```

```
first
```

```
# A tibble: 3 x 2
  x1       x2
  <chr> <dbl>
1 A         1
2 B         2
3 C         3
```

```
second
```

```
# A tibble: 3 x 2
  x1       x2
  <chr> <dbl>
1 D        10
2 B        20
3 C        30
```

bind_cols

```
bind_cols(first, second)
```

```
# A tibble: 3 x 4
  x1       x2 x11      x21
  <chr> <dbl> <chr> <dbl>
1 A         1 D        10
2 B         2 B        20
3 C         3 C        30
```

Data Wrangling with dplyr and tidyr Cheat Sheet

RStudio

Tidy Data - A foundation for wrangling in R

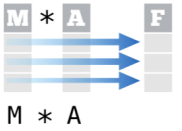
In a tidy data set:

Each **variable** is saved in its own **column**

&

Each **observation** is saved in its own **row**

Tidy data complements R's **vectorized operations**. R will automatically preserve observations as you manipulate variables. No other format works as intuitively with R.



Syntax - Helpful conventions for wrangling

dplyr::**tbl_df(iris)**

Converts data to tbl class. tbl's are easier to examine than data frames. R displays only the data that fits onscreen:

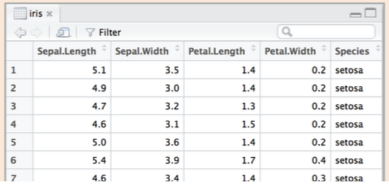
```
Source: local data frame [150 x 5]

   Sepal.Length Sepal.Width Petal.Length
1           5.1         3.5          1.4
2           4.9         3.0          1.4
3           4.7         3.2          1.3
4           4.6         3.1          1.5
5           5.0         3.6          1.4
..          ...         ...          ...
Variables not shown: Petal.Width (dbl),
  Species (fctr)
```

dplyr::**glimpse(iris)**

Information dense summary of tbl data.

utils::**View(iris)**

View data set in spreadsheet-like display (note capital V).



Reshaping Data - Change the layout of a data set

tidyr::**gather(cases, "year", "n", 2:4)**

Gather columns into rows.

tidyr::**spread(pollution, size, amount)**

Spread rows into columns.

tidyr::**separate(storms, date, c("y", "m", "d"))**

Separate one column into several.

tidyr::**unite(data, col, ..., sep)**

Unite several columns into one.

dplyr::**data_frame(a = 1:3, b = 4:6)**

Combine vectors into data frame (optimized).

dplyr::**arrange(mtcars, mpg)**

Order rows by values of a column (low to high).

dplyr::**arrange(mtcars, desc(mpg))**

Order rows by values of a column (high to low).

dplyr::**rename(tb, y = year)**

Rename the columns of a data frame.

Subset Observations (Rows)

dplyr::**filter(iris, Sepal.Length > 7)**

Extract rows that meet logical criteria.

dplyr::**distinct(iris)**

Remove duplicate rows.

dplyr::**sample_frac(iris, 0.5, replace = TRUE)**

Randomly select fraction of rows

Subset Variables (Columns)

dplyr::**select(iris, Sepal.Width, Petal.Length, Species)**

Select columns by name or helper function.

Helper functions for select - ?select

select(iris, contains("."))

Select columns whose name contains a character string.

select(iris, ends_with("Length"))

Summarise Data

dplyr::**summarise(iris, avg = mean(Sepal.Length))**
Summarise data into single row of values.

dplyr::**summarise_each(iris, funs(mean))**
Apply summary function to each column.

dplyr::**count(iris, Species, wt = Sepal.Length)**
Count number of rows with each unique value of variable (with or without weights).

Summarise uses **summary functions**, functions that take a vector of values and return a single value, such as:

dplyr::**first**
First value of a vector.

dplyr::**last**
Last value of a vector.

dplyr::**nth**
Nth value of a vector.

dplyr::**n**
# of values in a vector.

dplyr::**n_distinct**
# of distinct values in a vector.

IQR
IQR of a vector.

min
Minimum value in a vector.

max
Maximum value in a vector.

mean
Mean value of a vector.

median
Median value of a vector.

var
Variance of a vector.

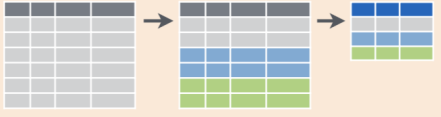
sd
Standard deviation of a vector.

Group Data

dplyr::**group_by(iris, Species)**
Group data into rows with the same value of Species.

dplyr::**ungroup(iris)**
Remove grouping information from data frame.

iris %>% group_by(Species) %>% summarise(…)
Compute separate summary row for each group.



Make New Variables

dplyr::**mutate(iris, sepal = Sepal.Length + Sepal. Width)**
Compute and append one or more new columns.

dplyr::**mutate_each(iris, funs(min_rank))**
Apply window function to each column.

dplyr::**transmute(iris, sepal = Sepal.Length + Sepal. Width)**
Compute one or more new columns. Drop original columns.

Mutate uses **window functions**, functions that take a vector of values and return another vector of values, such as:

dplyr::**lead**
Copy with values shifted by 1.

dplyr::**lag**
Copy with values lagged by 1.

dplyr::**dense_rank**
Ranks with no gaps.

dplyr::**min_rank**
Ranks. Ties get min rank.

dplyr::**percent_rank**
Ranks rescaled to [0, 1].

dplyr::**row_number**
Ranks. Ties got to first value.

dplyr::**ntile**
Bin vector into n buckets.

dplyr::**between**
Are values between a and b?

dplyr::**cume_dist**
Cumulative distribution.

dplyr::**cumall**
Cumulative `all`

dplyr::**cumany**
Cumulative `any`

dplyr::**cummean**
Cumulative `mean`

cumsum
Cumulative `sum`

cummax
Cumulative `max`

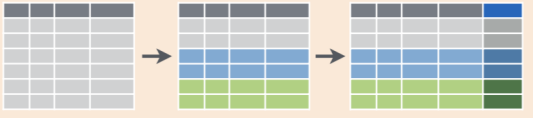
cummin
Cumulative `min`

cumprod
Cumulative `prod`

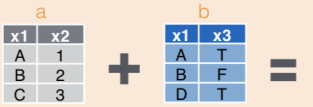
pmax
Element-wise `max`

pmin
Element-wise `min`

iris %>% group_by(Species) %>% mutate(…)
Compute new variables by group.

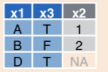


Combine Data Sets



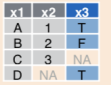
Mutating Joins

dplyr::**left_join(a, b, by = "x1")**
Join matching rows from b to a.



dplyr::**right_join(a, b, by = "x1")**
Join matching rows from a to b.

dplyr::**inner_join(a, b, by = "x1")**
Join data. Retain only rows in both sets.

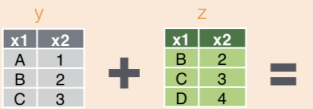


dplyr::**full_join(a, b, by = "x1")**
Join data. Retain all values, all rows.

Filtering Joins

dplyr::**semi_join(a, b, by = "x1")**
All rows in a that have a match in b.

dplyr::**anti_join(a, b, by = "x1")**
All rows in a that do not have a match in b.



Set Operations

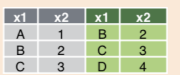
dplyr::**intersect(y, z)**
Rows that appear in both y and z.

dplyr::**union(y, z)**
Rows that appear in either or both y and z.

dplyr::**setdiff(y, z)**
Rows that appear in y but not z.

Binding

dplyr::**bind_rows(y, z)**
Append z to y as new rows.



dplyr::**bind_cols(y, z)**
Append z to y as new columns.
Caution: matches rows by position.

devtools::**install_github("rstudio/EDAWR")** for data sets

Learn more with **browseVignettes(package = c("dplyr", "tidyr"))** • dplyr 0.4.0• tidyr 0.2.0 • Updated: 1/15

Slides available at: hellor.netlify.app

Acknowledgement: R Studio Education Team