

DEALING WITH DATA IN R

HOW TO USE DPLYR

J. Alexander Branham

Fall 2016

DATA TRANSFORMATION

Next up: data transformation. We'll be working with the `gapminder` data frame from the `gapminder` package, so make sure it's installed then load it:

```
## install.packages(c("gapminder", "dplyr"))  
library(dplyr) # for data transformation  
library(gapminder) # example data to work with
```

THE DATA

```
gapminder
```

```
## Source: local data frame [1,704 x 6]
```

```
##
```

```
##      country continent  year lifeExp      pop gdpPercap
```

```
##      <fctr>      <fctr> <int>   <dbl>      <int>      <dbl>
```

```
## 1  Afghanistan      Asia  1952   28.801   8425333    779.4453
```

```
## 2  Afghanistan      Asia  1957   30.332   9240934    820.8530
```

```
## 3  Afghanistan      Asia  1962   31.997  10267083    853.1007
```

```
## 4  Afghanistan      Asia  1967   34.020  11537966    836.1971
```

```
## 5  Afghanistan      Asia  1972   36.088  13079460    739.9811
```

```
## 6  Afghanistan      Asia  1977   38.438  14880372    786.1134
```

```
## 7  Afghanistan      Asia  1982   39.854  12881816    978.0114
```

- keep only certain observations - `filter`

- keep only certain observations - `filter`
- keep only certain variables - `select`

- keep only certain observations - `filter`
- keep only certain variables - `select`
- reorder rows - `arrange`

- keep only certain observations - `filter`
- keep only certain variables - `select`
- reorder rows - `arrange`
- create new variables - `mutate`

- keep only certain observations - `filter`
- keep only certain variables - `select`
- reorder rows - `arrange`
- create new variables - `mutate`
- collapse data into summary statistics - `summarize`

- keep only certain observations - `filter`
- keep only certain variables - `select`
- reorder rows - `arrange`
- create new variables - `mutate`
- collapse data into summary statistics - `summarize`

- keep only certain observations - `filter`
- keep only certain variables - `select`
- reorder rows - `arrange`
- create new variables - `mutate`
- collapse data into summary statistics - `summarize`

Perform the above actions by groups - `group_by`

- Usage for all of these is the same

- Usage for all of these is the same
 - Name of the data frame

- Usage for all of these is the same
 - Name of the data frame
 - What to do with the data frame

- Usage for all of these is the same
 - Name of the data frame
 - What to do with the data frame
 - Result is always a data frame

How to get only countries in Africa?

How to get only countries in Africa?

```
filter(gapminder, continent == "Africa")
```

```
## Source: local data frame [624 x 6]
```

```
##
```

```
##   country continent  year lifeExp      pop gdpPercap
```

```
##   <fctr>      <fctr> <int>   <dbl>    <int>    <dbl>
```

```
## 1  Algeria    Africa  1952   43.077  9279525  2449.008
```

```
## 2  Algeria    Africa  1957   45.685 10270856  3013.976
```

```
## 3  Algeria    Africa  1962   48.303 11000948  2550.817
```

```
## 4  Algeria    Africa  1967   51.407 12760499  3246.992
```

```
## 5  Algeria    Africa  1972   54.518 14760787  4182.664
```

```
## 6  Algeria    Africa  1977   58.014 17152804  4910.417
```

YOU TRY!

Get a data frame of all the countries in Europe in 1997

YOU TRY (ANSWER)

```
filter(gapminder, continent == "Europe", year == 1997)
```

```
## Source: local data frame [30 x 6]
```

```
##  
##           country continent  year lifeExp      pop gdpPercap  
##           <fctr>    <fctr> <int>   <dbl>    <int>      <dbl>  
## 1      Albania      Europe  1997   72.950  3428038   3193.055  
## 2      Austria      Europe  1997   77.510  8069876  29095.921  
## 3      Belgium      Europe  1997   77.530 10199787  27561.197  
## 4 Bosnia and Herzegovina Europe  1997   73.244  3607000   4766.356  
## 5      Bulgaria      Europe  1997   70.320  8066057   5970.389  
## 6      Croatia      Europe  1997   73.680  4444595   9875.605  
## 7      Czech Republic Europe  1997   74.010 10300707 16048.5148
```

R supports several logical comparisons:

- Equal ==

R supports several logical comparisons:

- Equal ==
- Not equal !=

R supports several logical comparisons:

- Equal ==
- Not equal !=
- Greater than > (or equal to >=)

R supports several logical comparisons:

- Equal ==
- Not equal !=
- Greater than > (or equal to >=)
- Less than < (or equal to <=)

- `filter` automatically joins multiple arguments with `&`

- `filter` automatically joins multiple arguments with `&`
- You can use `|` instead, which means “or”

- `filter` automatically joins multiple arguments with `&`
- You can use `|` instead, which means “or”
- Try to get all the countries in Europe or Africa

```
filter(gapminder, continent == "Europe" | "Africa")
```

```
## Error in eval(expr, envir, enclos): operations are possible only for r
```

```
filter(gapminder, continent %in% c("Europe", "Africa"))
```

```
## Source: local data frame [984 x 6]
```

```
##
```

```
##   country continent  year lifeExp      pop gdpPercap
```

```
##   <fctr>      <fctr> <int>   <dbl>    <int>      <dbl>
```

```
## 1  Albania     Europe  1952   55.230  1282697  1601.056
```

```
## 2  Albania     Europe  1957   59.280  1476505  1942.284
```

```
## 3  Albania     Europe  1962   64.820  1728137  2312.889
```

```
## 4  Albania     Europe  1967   66.220  1984060  2760.197
```

```
## 5  Albania     Europe  1972   67.690  2263554  3313.422
```

```
## 6  Albania     Europe  1977   68.930  2509048  3533.004
```

```
## 7  Albania     Europe  1982   70.420  2780097  3630.881
```

WHAT'S OR USED FOR?

```
filter(gapminder, continent == "Asia" | country == "Turkey")
```

```
## Source: local data frame [408 x 6]
```

```
##
```

```
##      country continent  year lifeExp      pop gdpPercap
```

```
##      <fctr>      <fctr> <int>   <dbl>      <int>      <dbl>
```

```
## 1  Afghanistan      Asia  1952   28.801   8425333    779.4453
```

```
## 2  Afghanistan      Asia  1957   30.332   9240934    820.8530
```

```
## 3  Afghanistan      Asia  1962   31.997  10267083    853.1007
```

```
## 4  Afghanistan      Asia  1967   34.020  11537966    836.1971
```

```
## 5  Afghanistan      Asia  1972   36.088  13079460    739.9811
```

```
## 6  Afghanistan      Asia  1977   38.438  14880372    786.1134
```

```
## 7  Afghanistan      Asia  1982   39.854  12881816    978.0114
```

SELECT

Sometimes you'll want to keep only the columns you're interested in. `select` lets you do that:

```
select(gapminder, country, year, pop)
```

```
## Source: local data frame [1,704 x 3]
```

```
##
```

```
##      country  year      pop
```

```
##      <fctr> <int>    <int>
```

```
## 1  Afghanistan  1952  8425333
```

```
## 2  Afghanistan  1957  9240934
```

```
## 3  Afghanistan  1962 10267083
```

```
## 4  Afghanistan  1967 11537966
```

```
## 5  Afghanistan  1972 13079460
```

SELECT HELPER FUNCTIONS

`select` has some helper functions: `starts_with` and `ends_with` are among the most useful:

```
select(gapminder, starts_with("c"), pop)
```

```
## Source: local data frame [1,704 x 3]
```

```
##
```

```
##      country continent      pop
```

```
##      <fctr>    <fctr>    <int>
```

```
## 1  Afghanistan      Asia  8425333
```

```
## 2  Afghanistan      Asia  9240934
```

```
## 3  Afghanistan      Asia 10267083
```

```
## 4  Afghanistan      Asia 11537966
```

```
## 5  Afghanistan      Asia 13079460
```

RENAME

You can use `select` to rename variables, but since it drops everything that it doesn't return, it oftentimes isn't good at that. `rename` does what you want it to, though:

```
rename(gapminder, population = pop)
```

```
## Source: local data frame [1,704 x 6]
```

```
##
```

```
##      country continent  year lifeExp population gdpPercap
```

```
##      <fctr>      <fctr> <int>   <dbl>         <int>         <dbl>
```

```
## 1  Afghanistan      Asia  1952   28.801     8425333     779.4453
```

```
## 2  Afghanistan      Asia  1957   30.332     9240934     820.8530
```

```
## 3  Afghanistan      Asia  1962   31.997    10267083     853.1007
```

```
## 4  Afghanistan      Asia  1967   34.020    11537966     836.1971
```

```
## 5  Afghanistan      Asia  1972   36.088    13079460     739.9811
```



```
arrange(gapminder, year)
```

```
## Source: local data frame [1,704 x 6]
```

```
##
```

```
##      country continent  year lifeExp      pop  gdpPercap
##      <fctr>    <fctr> <int>   <dbl>    <int>      <dbl>
## 1  Afghanistan      Asia  1952  28.801  8425333   779.4453
## 2    Albania      Europe  1952  55.230  1282697  1601.0561
## 3    Algeria      Africa  1952  43.077  9279525  2449.0082
## 4    Angola      Africa  1952  30.015  4232095  3520.6103
## 5   Argentina  Americas  1952  62.485 17876956  5911.3151
## 6   Australia  Oceania   1952  69.120  8691212 10039.5956
## 7    Austria      Europe  1952  66.800  6927772  6137.0765
```

MUTATE

`mutate` allows you to create new variables:

```
mutate(gapminder, gdp = pop * gdpPercap)
```

```
## Source: local data frame [1,704 x 7]
```

```
##
```

##		country	continent	year	lifeExp	pop	gdpPercap	gdp
##		<fctr>	<fctr>	<int>	<dbl>	<int>	<dbl>	<dbl>
## 1		Afghanistan	Asia	1952	28.801	8425333	779.4453	6567086330
## 2		Afghanistan	Asia	1957	30.332	9240934	820.8530	7585448670
## 3		Afghanistan	Asia	1962	31.997	10267083	853.1007	8758855797
## 4		Afghanistan	Asia	1967	34.020	11537966	836.1971	9648014150
## 5		Afghanistan	Asia	1972	36.088	13079460	739.9811	9678553274
## 6		Afghanistan	Asia	1977	38.438	14880372	786.1134	11697659231

MUTATE

We can create multiple variables at once:

```
mutate(gapminder,  
      gdp = pop * gdpPercap,  
      gdp_in_billions = gdp / 1000000)
```

```
## Source: local data frame [1,704 x 8]
```

```
##
```

```
##      country continent  year lifeExp      pop gdpPercap      gdp  
##      <fctr>      <fctr> <int>   <dbl>    <int>    <dbl>    <dbl>  
## 1  Afghanistan      Asia  1952  28.801  8425333  779.4453 6567086330  
## 2  Afghanistan      Asia  1957  30.332  9240934  820.8530 7585448670  
## 3  Afghanistan      Asia  1962  31.997 10267083  853.1007 8758855797  
## 4  Afghanistan      Asia  1967  34.020 11537966  836.1971 9648014150
```

`summarize` (or `summarise` if you prefer) creates summary statistics:

```
summarize(gapminder, mean_life = mean(lifeExp))
```

```
## Source: local data frame [1 x 1]
```

```
##
```

```
##   mean_life
```

```
##      <dbl>
```

```
## 1  59.47444
```

`group_by` allows us to perform operations by groups:

```
by_year <- group_by(gapminder, year)
summarize(by_year, mean_life = mean(lifeExp))
```

```
## Source: local data frame [12 x 2]
```

```
##
```

```
##      year mean_life
```

```
##      <int>      <dbl>
```

```
## 1    1952    49.05762
```

```
## 2    1957    51.50740
```

```
## 3    1962    53.60925
```

```
## 4    1967    55.67829
```

```
## 5    1972    57.64739
```

PIPING

The pipe operator `%>%` pipes the output of the left side to the first argument of the right side:

```
gapminder %>%  
  group_by(continent, year) %>%  
  summarize(mean_life = mean(lifeExp),  
            n = n())
```

```
## Source: local data frame [60 x 4]  
## Groups: continent [?]  
##  
##   continent  year mean_life    n  
##   <fctr> <int>    <dbl> <int>  
## 1   Africa  1952   39.13550    52
```

YOU TRY!

- What is the mean life expectancy in Europe in 1997?

YOU TRY!

- What is the mean life expectancy in Europe in 1997?
- What is the total population of Asia in 1992?

YOU TRY!

- What is the mean life expectancy in Europe in 1997?
- What is the total population of Asia in 1992?
- Create a plot with year along the x-axis and average life expectancy by continent along the y-axis.

YOU TRY (ANSWERS)

```
gapminder %>%  
  filter(year == 1997, continent == "Europe") %>%  
  summarize(mean_life = mean(lifeExp))
```

```
## Source: local data frame [1 x 1]  
##  
##   mean_life  
##   <dbl>  
## 1  75.50517
```

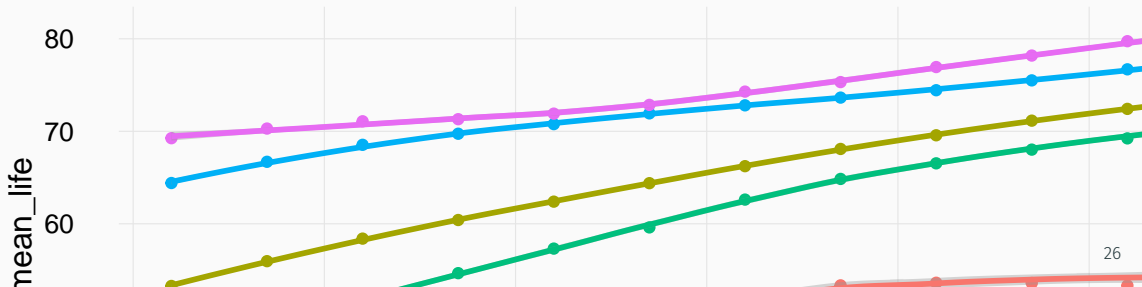
YOU TRY (ANSWERS)

```
gapminder %>%  
  filter(continent == "Asia", year == 1992) %>%  
  summarize(total_pop = sum(as.numeric(pop)))
```

```
## Source: local data frame [1 x 1]  
##  
##   total_pop  
##   <dbl>  
## 1 3133292191
```

YOU TRY (ANSWERS)

```
gapminder %>%  
  group_by(year, continent) %>%  
  summarize(mean_life = mean(lifeExp)) %>%  
  ggplot(aes(year, mean_life, color = continent)) +  
  geom_point() + geom_smooth()
```



SUMMARIZE ALL

We can use `summarize_all` to summarize multiple variables:

```
gapminder %>%  
  group_by(year) %>%  
  summarize_all(mean)  
  
## Warning in mean.default(structure(1:142, .Label = c("Afghanistan",  
## "Albania", : argument is not numeric or logical: returning NA  
  
## Warning in mean.default(structure(1:142, .Label = c("Afghanistan",  
## "Albania", : argument is not numeric or logical: returning NA  
  
## Warning in mean.default(structure(1:142, .Label = c("Afghanistan",  
## "Albania", : argument is not numeric or logical: returning NA
```

SUMMARIZE IF

`summarize_if` allows us to do conditional summaries:

```
gapminder %>%  
  group_by(year) %>%  
  summarize_if(is.numeric, mean)
```

```
## Source: local data frame [12 x 4]  
##  
##      year  lifeExp      pop gdpPercap  
##   <int>    <dbl>    <dbl>    <dbl>  
## 1  1952  49.05762 16950402  3725.276  
## 2  1957  51.50740 18763413  4299.408  
## 3  1962  53.60925 20421007  4725.812  
## 4  1967  55.67829 22658298  5483.653
```