Intro to R for Data Science

Dr. Laurie Baker
Office for National Statistics/Department for International Development
(updated: 2020-05-19)

Learning objectives: Part 2

- Introduction to data wrangling using the `tidyverse` set of metapackages.
- Use the tidyverse verbs to explore the gapminder data set.
- Learn to merge datasets using `left_join`.
- Create meaningful visualisations of the data using `ggplot2`.
- Learn where to go for help.

Subsetting and summarising using dplyr:

`dplyr`, has made a lot of data manipulation easier and clearer using **verbs** to filter and select different elements.

- `select()` subsets columns based on their names.
- `filter()` subsets rows based on their values.
- `summarise()` calculates summary statistics.
- `group_by()` groups variable for summarising.
- `mutate()` adds new columns that are functions of existing variables.

These verbs can be combined in powerful ways to do some really interesting data manipulation tasks!

select

```
select(gapminder, lifeExp, pop)
```

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

The pipe operator

- `%>%` = **"and then"**

```
gapminder %>%
  select(lifeExp, country)
```

```
## # A tibble: 1,704 x 2
##    lifeExp country
##      <dbl> <chr>
##  1    28.8 Afghanistan
##  2    30.3 Afghanistan
##  3    32.0 Afghanistan
##  4    34.0 Afghanistan
##  5    36.1 Afghanistan
##  6    38.4 Afghanistan
##  7    39.9 Afghanistan
##  8    40.8 Afghanistan
##  9    41.7 Afghanistan
## 10    41.8 Afghanistan
## # ... with 1,694 more rows
```

Assign your output to a new data frame

```
lifeExp_by_country ← gapminder %>%
  select(lifeExp, country)

head(lifeExp_by_country)
```

```
## # A tibble: 6 x 2
##   lifeExp country
##     <dbl> <chr>
## 1    28.8 Afghanistan
## 2    30.3 Afghanistan
## 3    32.0 Afghanistan
## 4    34.0 Afghanistan
## 5    36.1 Afghanistan
## 6    38.4 Afghanistan
```

Exercises Part I

1. Run the following line of code, what does the minus do?

```
gapminder %>%
  select(-c(lifeExp, country))
```

```
## # A tibble: 1,704 x 6
##    continent  year      pop gdpPercap infant_mortality fertility
##    <chr>     <dbl>    <dbl>     <dbl>             <dbl>     <dbl>
##  1 Asia       1952  8425333      779.               NA        NA
##  2 Asia       1957  9240934      821.               NA        NA
##  3 Asia       1962 10267083      853.               NA        NA
##  4 Asia       1967 11537966      836.               NA        NA
##  5 Asia       1972 13079460      740.               NA        NA
##  6 Asia       1977 14880372      786.               NA        NA
##  7 Asia       1982 12881816      978.               NA        NA
##  8 Asia       1987 13867957      852.               NA        NA
##  9 Asia       1992 16317921      649.               NA        NA
## 10 Asia       1997 22227415      635.               NA        NA
## #  ... with 1,694 more rows
```

2. Select the columns `country`, `continent` and `gdpPercap` from the data frame.

```
gapminder %>%
  select("country", "continent", "gdpPercap")
```

```
## # A tibble: 1,704 x 3
##    country     continent gdpPercap
##    <chr>       <chr>         <dbl>
##  1 Afghanistan Asia           779.
##  2 Afghanistan Asia           821.
##  3 Afghanistan Asia           853.
##  4 Afghanistan Asia           836.
##  5 Afghanistan Asia           740.
##  6 Afghanistan Asia           786.
##  7 Afghanistan Asia           978.
##  8 Afghanistan Asia           852.
##  9 Afghanistan Asia           649.
## 10 Afghanistan Asia           635.
## # ... with 1,694 more rows
```

3. Write 2 ways to select all the columns except for year.

Option 1

```
gapminder %>%
  select(c("country", "continent", "lifeExp",
           "pop", "gdpPercap", "infant_mortality",
           "fertility"))
```

Option 2

```
gapminder %>%
  select(-year)
```

filter

- `filter`: subsetting rows

Logical Operator	Description
<	Less Than
<=	Less Than or Equal To
>	Greater Than
>=	Greater Than or Equal To
==	Equal To
!=	Not Equal To
&	And
%in% c(....)	Membership **one in** a list of elements

Filter using double "="

- Filter a particular country:

```
gapminder %>%
  filter(country == "Yemen, Rep.") %>%
  head()
```

```
## # A tibble: 6 x 8
##   country     continent  year lifeExp     pop gdpPercap infant_mortality fertility
##   <chr>       <chr>     <dbl>   <dbl>   <dbl>     <dbl>             <dbl>     <dbl>
## 1 Yemen, Rep. Asia       1952    32.5 4963829      782.               NA        NA
## 2 Yemen, Rep. Asia       1957    34.0 5498090      805.               NA        NA
## 3 Yemen, Rep. Asia       1962    35.2 6120081      826.               NA        NA
## 4 Yemen, Rep. Asia       1967    37.0 6740785      862.               NA        NA
## 5 Yemen, Rep. Asia       1972    39.8 7407075     1265.               NA        NA
## 6 Yemen, Rep. Asia       1977    44.2 8403990     1830.               NA        NA
```

Filter rows from a set of matches

```
gapminder %>%
  filter(country %in% c("Jordan", "Oman",
                        "Syria", "Yemen, Rep."))
```

```
## # A tibble: 48 x 8
##    country continent  year lifeExp     pop gdpPercap infant_mortality fertility
##    <chr>   <chr>     <dbl>   <dbl>   <dbl>     <dbl>             <dbl>     <dbl>
##  1 Jordan  Asia       1952    43.2  607914     1547.              NA       NA
##  2 Jordan  Asia       1957    45.7  746559     1886.              NA       NA
##  3 Jordan  Asia       1962    48.1  933559     2348.              96.6      7.9
##  4 Jordan  Asia       1967    51.6 1255058     2742.              75.2      8.03
##  5 Jordan  Asia       1972    56.5 1613551     2111.              60.2      7.82
##  6 Jordan  Asia       1977    61.1 1937652     2852.              49.1      7.46
##  7 Jordan  Asia       1982    63.7 2347031     4161.              40.2      7.07
##  8 Jordan  Asia       1987    65.9 2820042     4449.              33.1      6.16
##  9 Jordan  Asia       1992    68.0 3867409     3432.              28.3      5.17
## 10 Jordan  Asia       1997    69.8 4526235     3645.              24.9      4.4
## # ... with 38 more rows
```

Combining multiple filters

- You can add multiple filters with a comma.

```
gapminder %>%
  filter(country == "Yemen, Rep.", year >= 1960 & year <= 1985)
```

```
## # A tibble: 5 x 8
##   country     continent  year lifeExp     pop gdpPercap infant_mortality fertility
##   <chr>       <chr>     <dbl>   <dbl>   <dbl>     <dbl>             <dbl>     <dbl>
## 1 Yemen, Rep. Asia       1962    35.2 6120081      826.               NA        NA
## 2 Yemen, Rep. Asia       1967    37.0 6740785      862.               NA        NA
## 3 Yemen, Rep. Asia       1972    39.8 7407075     1265.               NA        NA
## 4 Yemen, Rep. Asia       1977    44.2 8403990     1830.               NA        NA
## 5 Yemen, Rep. Asia       1982    49.1 9657618     1978.               NA        NA
```

Exercises Part 2

1. What do these lines of code filter the data for?

```
gapminder %>%
  filter(continent == "Europe", lifeExp > 70)
```

```
## # A tibble: 257 x 8
##    country continent  year lifeExp     pop gdpPercap infant_mortality fertility
##    <chr>   <chr>     <dbl>   <dbl>   <dbl>     <dbl>             <dbl>     <dbl>
##  1 Albania Europe     1982    70.4 2780097     3631.              56.1      3.46
##  2 Albania Europe     1987    72   3075321     3739.              40.8      3.13
##  3 Albania Europe     1992    71.6 3326498     2497.              32.5      2.87
##  4 Albania Europe     1997    73.0 3428038     3193.              26.8      2.61
##  5 Albania Europe     2002    75.7 3508512     4604.              21        2.2
##  6 Albania Europe     2007    76.4 3600523     5937.              16.7      1.8
##  7 Austria Europe     1967    70.1 7376998    12835.              26.4      2.62
##  8 Austria Europe     1972    70.6 7544201    16662.              24.1      2.09
##  9 Austria Europe     1977    72.2 7568430    19749.              16.6      1.63
## 10 Austria Europe     1982    73.2 7574613    21597.              12.6      1.66
## #  ... with 247 more rows
```

2. Filter the data for countries in "Asia" where the "lifeExp" was below 35

```
gapminder %>%
  filter(continent == "Asia", lifeExp < 35)
```

```
## # A tibble: 7 x 8
##   country     continent  year lifeExp      pop gdpPercap infant_mortality fertility
##   <chr>       <chr>     <dbl>   <dbl>    <dbl>     <dbl>             <dbl>     <dbl>
## 1 Afghanistan Asia       1952    28.8  8425333      779.              NA       NA
## 2 Afghanistan Asia       1957    30.3  9240934      821.              NA       NA
## 3 Afghanistan Asia       1962    32.0 10267083      853.              NA       NA
## 4 Afghanistan Asia       1967    34.0 11537966      836.              NA       NA
## 5 Cambodia    Asia       1977    31.2  6978607      525.             155.       5.44
## 6 Yemen, Rep. Asia       1952    32.5  4963829      782.              NA       NA
## 7 Yemen, Rep. Asia       1957    34.0  5498090      805.              NA       NA
```

3. Filter the data for observations where the gdpPercap was equal to 1000 or less.

```
gapminder %>%
  filter(gdpPercap ⩽ 1000)
```

```
## # A tibble: 351 x 8
##    country     continent  year lifeExp      pop gdpPercap infant_mortality fertility
##    <chr>       <chr>     <dbl>   <dbl>    <dbl>     <dbl>             <dbl>     <dbl>
##  1 Afghanistan Asia       1952    28.8  8425333      779.               NA        NA
##  2 Afghanistan Asia       1957    30.3  9240934      821.               NA        NA
##  3 Afghanistan Asia       1962    32.0 10267083      853.               NA        NA
##  4 Afghanistan Asia       1967    34.0 11537966      836.               NA        NA
##  5 Afghanistan Asia       1972    36.1 13079460      740.               NA        NA
##  6 Afghanistan Asia       1977    38.4 14880372      786.               NA        NA
##  7 Afghanistan Asia       1982    39.9 12881816      978.               NA        NA
##  8 Afghanistan Asia       1987    40.8 13867957      852.               NA        NA
##  9 Afghanistan Asia       1992    41.7 16317921      649.               NA        NA
## 10 Afghanistan Asia       1997    41.8 22227415      635.               NA        NA
## # ... with 341 more rows
```

4. Filter using `%in%` for countries "Chile", "Argentina", "Uruguay", and "Peru" and year >= to 1992.

```
gapminder %>%
  filter(country %in% c("Chile", "Argentina", "Uruguay", "Peru") & year >= 1992)
```

```
## # A tibble: 16 x 8
##    country   continent  year lifeExp      pop gdpPercap infant_mortality fertility
##    <chr>     <chr>     <dbl>   <dbl>    <dbl>     <dbl>            <dbl>     <dbl>
##  1 Argentina Americas   1992    71.9 33958947     9308.             22.8      2.92
##  2 Argentina Americas   1997    73.3 36203463    10967.             19.6      2.66
##  3 Argentina Americas   2002    74.3 38331121     8798.             17.1      2.38
##  4 Argentina Americas   2007    75.3 40301927    12779.             14.1      2.25
##  5 Chile     Americas   1992    74.1 13572994     7596.             13.7      2.56
##  6 Chile     Americas   1997    75.8 14599929    10118.             10.2      2.25
##  7 Chile     Americas   2002    77.9 15497046    10779.              8.3      2.01
##  8 Chile     Americas   2007    78.6 16284741    13172.              7.6      1.9
##  9 Peru      Americas   1992    66.5 22430449     4446.             51        3.62
## 10 Peru      Americas   1997    68.4 24748122     5838.             36.9      3.14
## 11 Peru      Americas   2002    69.9 26769436     5909.             25.7      2.82
```

5. Filter the data using $\neq$ to include the data from all continents apart from Europe.

```
gapminder %>%
  filter(continent ≠ "Europe")
```

```
## # A tibble: 1,344 x 8
##    country     continent  year lifeExp      pop gdpPercap infant_mortality fertility
##    <chr>       <chr>     <dbl>   <dbl>    <dbl>     <dbl>             <dbl>     <dbl>
##  1 Afghanistan Asia       1952    28.8  8425333      779.               NA        NA
##  2 Afghanistan Asia       1957    30.3  9240934      821.               NA        NA
##  3 Afghanistan Asia       1962    32.0 10267083      853.               NA        NA
##  4 Afghanistan Asia       1967    34.0 11537966      836.               NA        NA
##  5 Afghanistan Asia       1972    36.1 13079460      740.               NA        NA
##  6 Afghanistan Asia       1977    38.4 14880372      786.               NA        NA
##  7 Afghanistan Asia       1982    39.9 12881816      978.               NA        NA
##  8 Afghanistan Asia       1987    40.8 13867957      852.               NA        NA
##  9 Afghanistan Asia       1992    41.7 16317921      649.               NA        NA
## 10 Afghanistan Asia       1997    41.8 22227415      635.               NA        NA
## # ... with 1,334 more rows
```

summarise()

- `summarise()` uses existing R functions to calculate summary statistics.
- mean lifeExp for all countries:

```
(lifeExp_stats ← gapminder %>%
                  summarise(
                    mean_lifeExp = mean(lifeExp)
                    )
 )
```

```
## # A tibble: 1 x 1
##   mean_lifeExp
##          <dbl>
## 1         59.5
```

summarise() multiple summary statistics

- Calculate multiple summary statistics by separating using ","

```
(lifeExp_stats ← gapminder %>%
                  summarise(
                    mean_lifeExp = mean(lifeExp), # mean
                    min_lifeExp = min(lifeExp), # min
                    max_lifeExp = max(lifeExp)) # max
                    )
```

```
## # A tibble: 1 x 3
##   mean_lifeExp min_lifeExp max_lifeExp
##          <dbl>       <dbl>       <dbl>
## 1         59.5        23.6        82.6
```

group_by()

- `group_by()` used to group variables.
- summarise min, mean, and max `lifeExp` per country

```
(lifeExp_stats_country ← gapminder %>%
                            group_by(country) %>%
                            summarise(
                               mean_lifeExp = mean(lifeExp),
                               min_lifeExp = min(lifeExp),
                               max_lifeExp = max(lifeExp)
                               ))
```

```
## # A tibble: 142 x 4
##    country     mean_lifeExp min_lifeExp max_lifeExp
##    <chr>              <dbl>       <dbl>       <dbl>
##  1 Afghanistan         37.5        28.8        43.8
##  2 Albania             68.4        55.2        76.4
##  3 Algeria             59.0        43.1        72.3
##  4 Angola              37.9        30.0        42.7
##  5 Argentina           69.1        62.5        75.3
##  6 Australia           74.7        69.1        81.2
##  7 Austria             73.1        66.8        79.8
##  8 Bahrain             65.6        50.9        75.6
##  9 Bangladesh          49.8        37.5        64.1
```

Exercises Part 3

1. What does the following bit of code do?

```
gapminder %>%
      group_by(continent, year) %>%
      summarise(mean_gdpPercap = mean(gdpPercap))
```

```
## # A tibble: 60 x 3
## # Groups:   continent [5]
##    continent  year mean_gdpPercap
##    <chr>     <dbl>          <dbl>
##  1 Africa     1952          1253.
##  2 Africa     1957          1385.
##  3 Africa     1962          1598.
##  4 Africa     1967          2050.
##  5 Africa     1972          2340.
##  6 Africa     1977          2586.
##  7 Africa     1982          2482.
##  8 Africa     1987          2283.
##  9 Africa     1992          2282.
## 10 Africa     1997          2379.
## #  ... with 50 more rows
```

2. Group the data by country and summarise the minimum and maximum population sizes.

```
gapminder %>%
  group_by(country) %>%
  summarise(min_pop = min(pop),
            max_pop = max(pop)) %>%
            head(8)
```

```
## # A tibble: 8 x 3
##   country      min_pop  max_pop
##   <chr>          <dbl>    <dbl>
## 1 Afghanistan  8425333 31889923
## 2 Albania      1282697  3600523
## 3 Algeria      9279525 33333216
## 4 Angola       4232095 12420476
## 5 Argentina   17876956 40301927
## 6 Australia    8691212 20434176
## 7 Austria      6927772  8199783
## 8 Bahrain       120447   708573
```

3. Group the data by continent and year. Summarise the max and min population.

```
gapminder %>%
  group_by(continent, year) %>%
  summarise(min_pop = min(pop),
            max_pop = max(pop)) %>%
            head(8)
```

```
## # A tibble: 8 x 4
## # Groups:   continent [1]
##   continent  year min_pop  max_pop
##   <chr>     <dbl>   <dbl>    <dbl>
## 1 Africa     1952   60011 33119096
## 2 Africa     1957   61325 37173340
## 3 Africa     1962   65345 41871351
## 4 Africa     1967   70787 47287752
## 5 Africa     1972   76595 53740085
## 6 Africa     1977   86796 62209173
## 7 Africa     1982   98593 73039376
```

The pipe function %>%

The pipe function

- The pipe function `%>%` allows you to combine multiple data wrangling steps

The pipe function in action!

Let's calculate life expectancy in Yemen pre 1980. First we take the `gapminder` data

```
yemen_pre1980_mean_lifeExp ← gapminder %>%
    filter(country == "Yemen, Rep.", year <1980) %>%
    select(lifeExp) %>%
    summarise(meanlifeExp = mean(lifeExp))
```

The pipe function in action!

And then, we filter for Yemen and years less than 1980.

```
yemen_pre1980_mean_lifeExp ← gapminder %>%
    filter(country == "Yemen, Rep.", year <1980) %>%
    select(lifeExp) %>%
    summarise(meanlifeExp = mean(lifeExp))
```

The pipe function in action!

And then we can select our column of interest `lifeExp`

```
yemen_pre1980_mean_lifeExp ← gapminder %>%
    filter(country == "Yemen, Rep.", year <1980) %>%
    select(lifeExp) %>%
    summarise(meanlifeExp = mean(lifeExp))
```

The pipe function in action!

And then we can use summarise to calculate the mean `lifeExp`

```
yemen_pre1980_mean_lifeExp ← gapminder %>%
    filter(country == "Yemen, Rep.", year <1980) %>%
    select(lifeExp) %>%
    summarise(mean_lifeExp = mean(lifeExp))


yemen_pre1980_mean_lifeExp
```

mean_lifeExp
37.1175

Looking at a slice()

`slice()` chooses rows by their position within the group, e.g. minimum `lifeExp`.

```
gapminder %>%
  group_by(year) %>%
  slice(which.min(lifeExp))
```

country	continent	year	lifeExp	pop	gdpPercap	infant_mortality	fertility
Afghanistan	Asia	1952	28.801	8425333	779.4453	NA	NA
Afghanistan	Asia	1957	30.332	9240934	820.8530	NA	NA
Afghanistan	Asia	1962	31.997	10267083	853.1007	NA	NA
Afghanistan	Asia	1967	34.020	11537966	836.1971	NA	NA
Sierra Leone	Africa	1972	35.400	2879013	1353.7598	185.2	6.80
Cambodia	Asia	1977	31.220	6978607	524.9722	155.4	5.44
Sierra Leone	Africa	1982	38.445	3464522	1465.0108	164.1	7.03
Angola	Africa	1987	39.906	7874230	2430.2083	134.1	7.20

Looking at a slice()

- Which country had the highest life Expectancy in each year?

```
gapminder %>%
  group_by(year) %>%
  slice(which.max(lifeExp))
```

country	continent	year	lifeExp	pop	gdpPercap	infant_mortality	fertility
Norway	Europe	1952	72.670	3327728	10095.422	NA	NA
Iceland	Europe	1957	73.470	165110	9244.001	NA	NA
Iceland	Europe	1962	73.680	182053	10350.159	16.9	3.98
Sweden	Europe	1967	74.160	7867931	15258.297	12.6	2.27
Sweden	Europe	1972	74.720	8122293	17832.025	10.4	1.91
Iceland	Europe	1977	76.110	221823	19654.962	9.2	2.49
Japan	Asia	1982	77.110	118454974	19384.106	6.5	1.75
Japan	Asia	1987	78.670	122091325	22375.942	5.0	1.67

Mutate

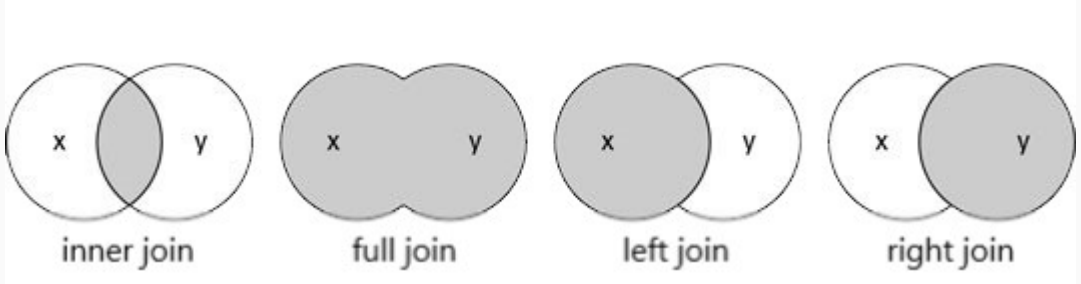
- `mutate()` adds new columns that are functions of existing variables.

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

country	continent	year	lifeExp	pop	gdpPercap	infant_mortality	fertility	gdp
Afghanistan	Asia	1952	28.80100	8425333	779.4453	NA	NA	6.567086e+09
Afghanistan	Asia	1957	30.33200	9240934	820.8530	NA	NA	7.585449e+09
Afghanistan	Asia	1962	31.99700	10267083	853.1007	NA	NA	8.758856e+09
Afghanistan	Asia	1967	34.02000	11537966	836.1971	NA	NA	9.648014e+09
Afghanistan	Asia	1972	36.08800	13079460	739.9811	NA	NA	9.678553e+09
Afghanistan	Asia	1977	38.43800	14880372	786.1134	NA	NA	1.169766e+10
Afghanistan	Asia	1982	39.85400	12881816	978.0114	NA	NA	1.259856e+10
Afghanistan	Asia	1987	40.82200	13867957	852.3959	NA	NA	1.182099e+10
Afghanistan	Asia	1992	41.67400	16317921	649.3414	NA	NA	1.059590e+10

Joining data frames: when one data frame is not

- Our data is often spread out over several data frames!
- We can **join** these data frames together using a variety of `join` functions from the `dplyr` package.



Joining data

Person Table

```
(person_table ← data.frame(Person_ID = c("Person1", "Person2"),
                           Name = c("Jane Doe", "John Smith"),
                           Job_ID = c("Job_1", "NA")))
```

```
##   Person_ID       Name Job_ID
## 1   Person1   Jane Doe  Job_1
## 2   Person2 John Smith     NA
```

Job Table

```
(job_table ← data.frame(Job_ID = c("Job_1", "Job_2"),
                        Job_Name = c("Programmer", "Statistician")))
```

```
##   Job_ID     Job_Name
## 1  Job_1   Programmer
## 2  Job_2 Statistician
```

Inner join:

```
person_table
```

```
##   Person_ID       Name Job_ID
## 1   Person1   Jane Doe  Job_1
## 2   Person2 John Smith     NA
```

```
job_table
```

```
##   Job_ID     Job_Name
## 1  Job_1   Programmer
## 2  Job_2 Statistician
```

- `inner_join:` only rows where there is a match.

```
inner_join(x = person_table, y = job_table, by = "Job_ID")
```

```
##   Person_ID     Name Job_ID   Job_Name
## 1   Person1 Jane Doe  Job_1 Programmer
```

Left join:

```
person_table
```

```
##   Person_ID       Name Job_ID
## 1   Person1   Jane Doe  Job_1
## 2   Person2 John Smith     NA
```

```
job_table
```

```
##   Job_ID     Job_Name
## 1  Job_1   Programmer
## 2  Job_2 Statistician
```

- `left_join:` All rows on the left side of join.
- Only rows from the right side where there's a match on the left.

```
left_join(x = person_table, y = job_table, by = "Job_ID")
```

```
##   Person_ID       Name Job_ID   Job_Name
## 1   Person1   Jane Doe  Job_1 Programmer
## 2   Person2 John Smith     NA       <NA>
```

Right join:

```
person_table
```

```
##   Person_ID       Name Job_ID
## 1   Person1   Jane Doe  Job_1
## 2   Person2 John Smith     NA
```

```
job_table
```

```
##   Job_ID     Job_Name
## 1  Job_1   Programmer
## 2  Job_2 Statistician
```

- `right_join:` Rows from the left side only where there is a match on the right.
- All rows from the right side of the join.

```
right_join(x = person_table, y = job_table, by = "Job_ID")
```

```
##   Person_ID     Name Job_ID     Job_Name
## 1   Person1 Jane Doe  Job_1   Programmer
## 2      <NA>     <NA>  Job_2 Statistician
```

Full join

```
person_table
```

```
##   Person_ID       Name Job_ID
## 1   Person1   Jane Doe  Job_1
## 2   Person2 John Smith     NA
```

```
job_table
```

```
##   Job_ID     Job_Name
## 1  Job_1   Programmer
## 2  Job_2 Statistician
```

- `full_join:` All rows from the left.
- All rows from the right, **Joined** where the criteria matches. NAs where no matches.

```
full_join(x = person_table, y = job_table, by = "Job_ID")
```

```
##   Person_ID       Name Job_ID     Job_Name
## 1   Person1   Jane Doe  Job_1   Programmer
## 2   Person2 John Smith     NA         <NA>
## 3      <NA>       <NA>  Job_2 Statistician
```

Creating a new data frame uk_gdpPercap_df

- Let's look at the per capita GDP in a way that's more meaningful.
- Let's create `gdpPercap_rel`, that is the `gdpPercap` of the country **relative** to the UK's `gdpPercap` of that same year.
- How? Country A: `gdpPercap` divided by UK's `gdpPercap` in the same year.

Creating a new data frame uk_gdpPercap_df

- Create a new dataframe `uk_gdpPercap_df`

```
uk_gdpPercap_df ← gapminder %>%
  filter(country == "United Kingdom") %>%
  select(gdpPercap, year) %>%
  rename(uk_gdpPercap = gdpPercap)
```

Creating a new data frame uk_gdpPercap_df

- Filter the rows for `country == "United Kingdom"`.

```
uk_gdpPercap_df ← gapminder %>%
  filter(country == "United Kingdom") %>%
  select(gdpPercap, year) %>%
  rename(uk_gdpPercap = gdpPercap)
```

Creating a new data frame uk_gdpPercap_df

- Select the columns `gdpPercap` and `year`.

```
uk_gdpPercap_df ← gapminder %>%
  filter(country == "United Kingdom") %>%
  select(gdpPercap, year) %>%
  rename(uk_gdpPercap = gdpPercap)
```

Creating a new data frame uk_gdpPercap_df

- Rename the variable `gdpPercap` to `uk_gdpPercap`.

```
uk_gdpPercap_df ← gapminder %>%
  filter(country == "United Kingdom") %>%
  select(gdpPercap, year) %>%
  rename(uk_gdpPercap = gdpPercap)
```

Creating a new data frame uk_gdpPercap_df

- Inspect the data frame.

```
uk_gdpPercap_df ← gapminder %>%
  filter(country == "United Kingdom") %>%
  select(gdpPercap, year) %>%
  rename(uk_gdpPercap = gdpPercap)

head(uk_gdpPercap_df)
```

```
## # A tibble: 6 x 2
##   uk_gdpPercap  year
##          <dbl> <dbl>
## 1        9980.  1952
## 2       11283.  1957
## 3       12477.  1962
## 4       14143.  1967
## 5       15895.  1972
## 6       17429.  1977
```

Joining `uk_gdpPercap_df` to the `gapminder` data frame

- We want to divide all the other country's `gdpPercap` by the UK `gdpPercap` in that same year.
- We can match the two data frames using a `left_join` on the common variable, `year`.

```
gapminder ← left_join(gapminder, uk_gdpPercap_df, by = "year")

names(gapminder)
```

```
##  [1] "country"          "continent"        "year"             "lifeExp"
##  [5] "pop"              "gdpPercap"        "infant_mortality" "fertility"
##  [9] "gdp"              "uk_gdpPercap"
```

Calculating the relative GDP per capita

```
gapminder ← gapminder %>%
  mutate(gdpPercap_rel = gdpPercap/uk_gdpPercap)
```

- Doublechecking our calculations... Is the United Kindom relative gdp always 1?

```
gapminder %>%
  filter(country == "United Kingdom") %>%
  select(gdpPercap_rel) %>%
  head()
```

```
## # A tibble: 6 x 1
##   gdpPercap_rel
##           <dbl>
## 1             1
## 2             1
## 3             1
## 4             1
## 5             1
## 6             1
```

How many countries had a smaller gdp per capita than the UK each year?

```
gapminder %>%
  group_by(year) %>%
  filter(gdpPercap_rel < 1) %>%
  summarise(count = n())
```

```
## # A tibble: 12 x 2
##     year count
##    <dbl> <int>
##  1  1952   134
##  2  1957   134
##  3  1962   131
##  4  1967   127
##  5  1972   124
##  6  1977   123
##  7  1982   123
##  8  1987   126
##  9  1992   123
```

Exercises Part 4

1. What does the following bit of code do?

```
gapminder %>%
  select(country, gdpPercap_rel) %>%
  filter(country %in% c("Argentina", "Chile", "Peru", "Brazil")) %>%
  group_by(country) %>%
  summarise(
            max_gdp = max(gdpPercap_rel),
            min_gdp = min(gdpPercap_rel),
            mean_gdp = mean(gdpPercap_rel)
            )
```

```
## # A tibble: 4 x 4
##   country   max_gdp min_gdp mean_gdp
##   <chr>       <dbl>   <dbl>    <dbl>
## 1 Argentina   0.608   0.298    0.495
## 2 Brazil      0.386   0.211    0.295
## 3 Chile       0.397   0.256    0.345
## 4 Peru        0.409   0.196    0.315
```

2. How many countries had a higher relative gdp per capita than the United Kindom per year?

```
gapminder %>%
  group_by() %>%
  filter(gdpPercap_rel > 1) %>%
  summarise(count = n())
```

count
170

3. **Which countries** had a higher relative gdp per capita than the UK in 2007? Fill in the blanks

```
gapminder %>%
  filter(gdpPercap_rel > 1, year == 2007) %>%
  select(country) %>%
  unique()
```

country
Australia
Austria
Belgium
Canada
Denmark
Finland

Answers to our poll

Using what we've learned so far, let's go back to our original comparisons.

Which country had a higher infant mortality rate in 2007?

- Sri Lanka or Turkey

```
gapminder %>%
  filter(year == 2007, country %in% c("Sri Lanka", "Turkey")) %>%
  select(country, infant_mortality)
```

country	infant_mortality
Sri Lanka	10.7
Turkey	20.2

- **Turkey**, difference = 9.5.

Which country had a higher infant mortality rate in 2007?

- Poland or Malaysia

```
gapminder %>%
  filter(year == 2007, country %in% c("Poland", "Malaysia")) %>%
  select(country, infant_mortality)
```

country	infant_mortality
Malaysia	6.9
Poland	6.0

- **Malaysia**, difference = 0.9

Which country had a higher infant mortality rate in 2007?

- Pakistan or Vietnam

```
gapminder %>%
  filter(year == 2007, country %in% c("Pakistan", "Vietnam")) %>%
  select(country, infant_mortality)
```

country	infant_mortality
Pakistan	77.4
Vietnam	21.4

- **Pakistan**, difference = 56

Which country had a higher life Expectancy in 2007?

- South Africa or Yemen

```
gapminder %>%
  filter(year == 2007, country %in% c("South Africa", "Yemen, Rep.")) %>%
  select(country, lifeExp)
```

country	lifeExp
South Africa	49.339
Yemen, Rep.	62.698

- **Yemen**, difference ~ 13.4 years

Which country had a higher life Expectancy in 2007?

- Chile or Hungary

```
gapminder %>%
  filter(year == 2007, country %in% c("Chile", "Hungary")) %>%
  select(country, lifeExp)
```

country	lifeExp
Chile	78.553
Hungary	73.338

- **Chile**, difference ~ 5.2 years

Which country do you think had the highest gdpPercap in 2007?

- Switzerland or Kuwait

```
gapminder %>%
  filter(year == 2007, country %in% c("Switzerland", "Kuwait")) %>%
  select(country, gdpPercap)
```

country	gdpPercap
Kuwait	47306.99
Switzerland	37506.42

- **Kuwait**, difference ~ 10,000 USD, about 20-25% more

Which country do you think had the highest gdpPercap in 2007?

- Colombia or Nepal

```
gapminder %>%
  filter(year == 2007, country %in% c("Colombia", "Nepal")) %>%
  select(country, gdpPercap)
```

country	gdpPercap
Colombia	7006.58
Nepal	1091.36

- **Colombia**, difference ~ 5,915, more than 6x

Which results did you find the most surprising?

Intro to Data Visualisation Using ggplot2

- See the data visualisation presentation and exercises accompanying these materials.

Getting Help

1. **Help and Vignette** Check the function or the documentation of the package you're working with using the help function `?` or `vignette` respectively.

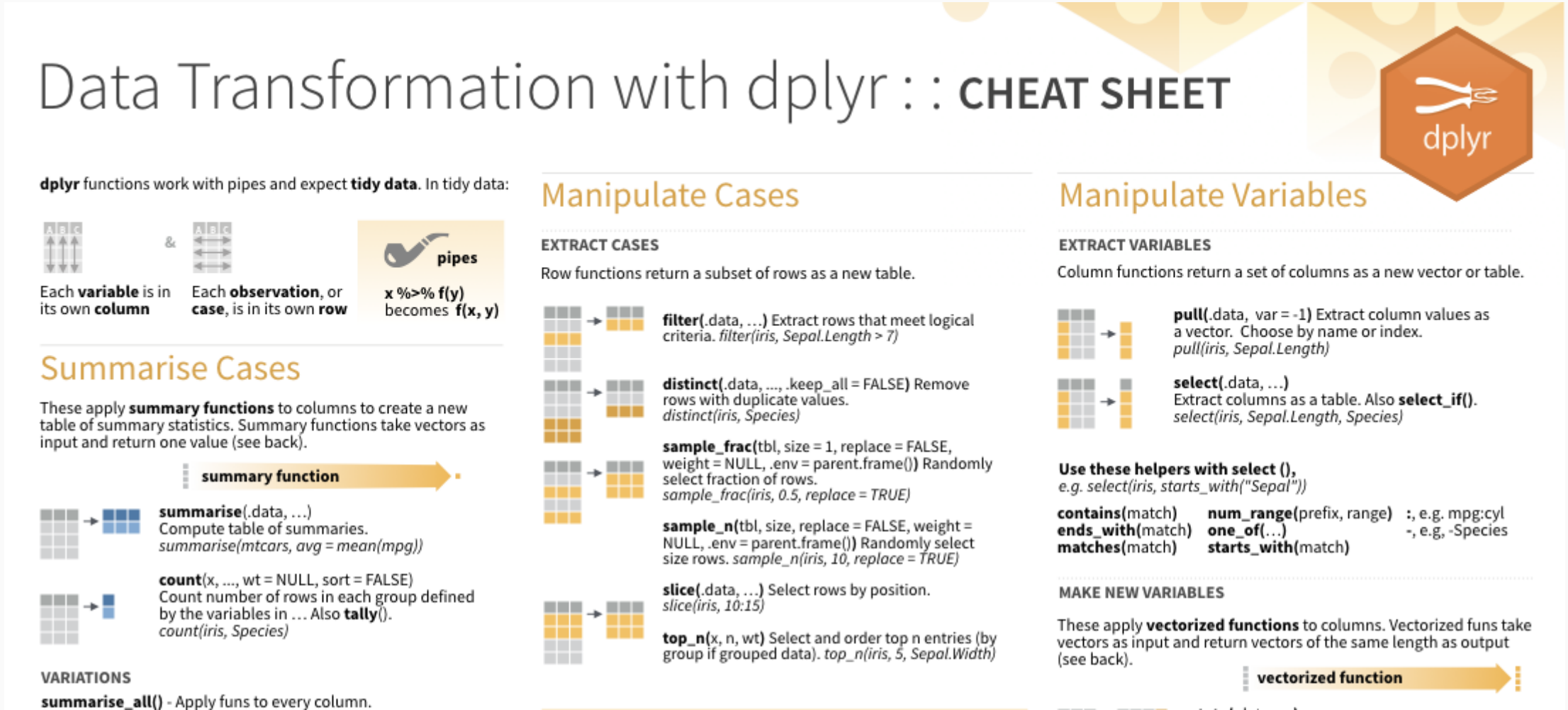
```
?filter

vignette("dplyr")
```

1. **CRAN Task View** Looking for a package to carry out a particular analysis? Check out CRAN Task View
2. **Stack Overflow** Stack Overflow Check out Stack Overflow. This is one of the first calls where members from the R Community will help you answer questions.

Getting Help

1. **Cheatsheets** Many of the tidyverse packages come with their own cheatsheets, which are a quick reference on how to use various functions. It also gives a good overview of what functions are available.



Data Transformation with dplyr : : **CHEAT SHEET**

dplyr

dplyr functions work with pipes and expect **tidy data**. In tidy data:

Each **variable** is in its own **column** & Each **observation**, or **case**, is in its own **row**

pipes

x %>% f(y) becomes **f(x, y)**

Summarise Cases

These apply **summary functions** to columns to create a new table of summary statistics. Summary functions take vectors as input and return one value (see back).

summary function

summarise(.data, …)
Compute table of summaries.
summarise(mtcars, avg = mean(mpg))

count(x, ..., wt = NULL, sort = FALSE)
Count number of rows in each group defined by the variables in … Also **tally**().
count(iris, Species)

VARIATIONS

summarise_all() - Apply funs to every column.

Manipulate Cases

EXTRACT CASES

Row functions return a subset of rows as a new table.

filter(.data, …) Extract rows that meet logical criteria. *filter(iris, Sepal.Length > 7)*

distinct(.data, ..., .keep_all = FALSE) Remove rows with duplicate values.
distinct(iris, Species)

sample_frac(tbl, size = 1, replace = FALSE, weight = NULL, .env = parent.frame()) Randomly select fraction of rows.
sample_frac(iris, 0.5, replace = TRUE)

sample_n(tbl, size, replace = FALSE, weight = NULL, .env = parent.frame()) Randomly select size rows. *sample_n(iris, 10, replace = TRUE)*

slice(.data, …) Select rows by position.
slice(iris, 10:15)

top_n(x, n, wt) Select and order top n entries (by group if grouped data). *top_n(iris, 5, Sepal.Width)*

Manipulate Variables

EXTRACT VARIABLES

Column functions return a set of columns as a new vector or table.

pull(.data, var = -1) Extract column values as a vector. Choose by name or index.
pull(iris, Sepal.Length)

select(.data, …)
Extract columns as a table. Also **select_if()**.
select(iris, Sepal.Length, Species)

Use these helpers with select (),
e.g. select(iris, starts_with("Sepal"))

contains(match)
ends_with(match)
matches(match)
num_range(prefix, range)
one_of(…)
starts_with(match)
:, e.g. mpg:cyl
-, e.g, -Species

MAKE NEW VARIABLES

These apply **vectorized functions** to columns. Vectorized funs take vectors as input and return vectors of the same length as output (see back).

vectorized function

Getting Help

1. **Google.** Google is your friend! Type "R help" followed by the warning or error message you received and I guarantee there will be someone who has had this problem before.

2. **Meet ups and coding clubs** Join a meet up or coffee and code group. Check out R-Ladies.

3. **Further resources** Looking to develop your learning further? Check out my trello board on R Resources for Data Science. This is still a work in progress, but I'm continually updating it with useful resources.

References

- Changing R Studio Settings and Overview of RStudio Panels Sydney R Ladies
- Showcasing RStudio features, overview of functions using `seq()` as an example. Stat 545 University of British Columbia Blog by Jenny Bryan
- Introduction to ggplot2 and the grammar of graphics. R for Data Science by Hadley Wickham and Garret Gromelund.

Thanks!

Slides created via the R package **xaringan**.

The chakra comes from remark.js, **knitr**, and R Markdown.