

Programming for Professional Research Using R

Session 3

November 5, 2025

Today

- Learn how to:
 - Filter, mutate, group, and summarize data using Tidyverse functions
 - Reshape data using Tidyverse functions
- Be introduced to:
 - The concept of a "tidy" dataset
- Practice the above!

Data Wrangling

Tidyverse Introduction

Base R Layout

Tidyverse Layout

```
names(iris)
```

```
## [1] "Sepal.Length" "Sepal.Width"  "Petal.Length" "Petal.Width"  "Species"
```

```
str_replace(str_to_lower(names(iris)), "\\.", "_")
```

```
## [1] "sepal_length" "sepal_width"  "petal_length" "petal_width"  "species"
```

Tidyverse Introduction

Base R Layout

Tidyverse Layout

Tidyverse functions introduce a 'cleaner' method to write code out, using what is called the 'pipe operator': `%>%`. It's almost like writing a recipe, step by step.

```
names(iris)
```

```
## [1] "Sepal.Length" "Sepal.Width"  "Petal.Length" "Petal.Width"  "Species"
```

```
iris %>%  
  names() %>%  
  str_to_lower() %>%  
  str_replace("\\.", "_")
```

```
## [1] "sepal_length" "sepal_width"  "petal_length" "petal_width"  "species"
```

(Note — you can also write the pipe operator as `|>`)

Basic Wrangling Functions

filter() select() mutate()

`filter()` is used to **extract** rows (a.k.a. observations) from a dataset. It does so using a logical condition.

```
filter_example <- mtcars %>%  
  filter(wt > 3)  
  
filter_example %>% head()
```

##		mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
##	Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	1
##	Hornet Sportabout	18.7	8	360.0	175	3.15	3.440	17.02	0	0	3	2
##	Valiant	18.1	6	225.0	105	2.76	3.460	20.22	1	0	3	1
##	Duster 360	14.3	8	360.0	245	3.21	3.570	15.84	0	0	3	4
##	Merc 240D	24.4	4	146.7	62	3.69	3.190	20.00	1	0	4	2
##	Merc 230	22.8	4	140.8	95	3.92	3.150	22.90	1	0	4	2

Basic Wrangling Functions

`filter()` `select()` `mutate()`

`select()` is used to **extract** columns (a.k.a variables) from a dataset. It does so using the variable(s)'s name.

```
select_example <- mtcars %>%  
  select(  
    mpg, carb  
  )  
select_example %>% head()
```

```
##           mpg carb  
## Mazda RX4      21.0    4  
## Mazda RX4 Wag  21.0    4  
## Datsun 710     22.8    1  
## Hornet 4 Drive  21.4    1  
## Hornet Sportabout 18.7    2  
## Valiant        18.1    1
```

Basic Wrangling Functions

filter()

select()

mutate()

`mutate()` can be used to either **create** a new column (a.k.a. variable) or to **modify** an existing column (a.k.a. variable).

```
mutate_example <- mtcars %>%  
  mutate(  
    heavy = case_when(  
      wt > 3 ~ "Yes",  
      TRUE  ~ "No"  
    )  
  )  
mutate_example %>% select(wt, heavy) %>% head()
```

##	wt	heavy
## Mazda RX4	2.620	No
## Mazda RX4 Wag	2.875	No
## Datsun 710	2.320	No
## Hornet 4 Drive	3.215	Yes
## Hornet Sportabout	3.440	Yes
## Valiant	3.460	Yes

Basic Wrangling Functions

group_by() and summarize()

pivot_longer()

pivot_longer() result

pivot_wider()

pivot_wider() result

group_by() and summarize() are used to **aggregate** data, i.e. to summarize information to a different level of observation.

```
group_by_summarize_example <- mtcars %>%  
  group_by(cyl) %>%  
  summarize(  
    mpg = mean(mpg, na.rm = TRUE)  
  )
```

```
group_by_summarize_example
```

```
## # A tibble: 3 × 2  
##   cyl  mpg  
##   <dbl> <dbl>  
## 1     4  26.7  
## 2     6  19.7  
## 3     8  15.1
```

Basic Wrangling Functions

group_by() and summarize()

pivot_longer()

pivot_longer() result

pivot_wider()

pivot_wider() result

```
relig_income[1:6] %>% head(n = 2)
```

```
## # A tibble: 2 × 6
##   religion `<$10k` `<$10-20k` `<$20-30k` `<$30-40k` `<$40-50k`
##   <chr>      <dbl>      <dbl>      <dbl>      <dbl>      <dbl>
## 1 Agnostic    27         34         60         81         76
## 2 Atheist     12         27         37         52         35
```

```
relig_income_long ← relig_income %>%
  pivot_longer(
    cols      = !religion, # Everything but religion
    names_to  = "levels",
    values_to = "num"
  )
```

Basic Wrangling Functions

group_by() and summarize()

pivot_longer()

pivot_longer() result

pivot_wider()

pivot_wider() result

```
relig_income[1:6] %>% head(n = 2)
```

```
## # A tibble: 2 × 6
##   religion `<$10k` ` $10-20k` ` $20-30k` ` $30-40k` ` $40-50k`
##   <chr>      <dbl>      <dbl>      <dbl>      <dbl>      <dbl>
## 1 Agnostic      27          34          60          81          76
## 2 Atheist       12          27          37          52          35
```

```
relig_income_long %>% head(n = 4)
```

```
## # A tibble: 4 × 3
##   religion levels      num
##   <chr>      <chr>    <dbl>
## 1 Agnostic <$10k      27
## 2 Agnostic $10-20k    34
## 3 Agnostic $20-30k    60
## 4 Agnostic $30-40k    81
```

Basic Wrangling Functions

group_by() and summarize()

pivot_longer()

pivot_longer() result

pivot_wider()

pivot_wider() result

```
fish_encounters %>% head(n = 4)
```

```
## # A tibble: 4 × 3
##   fish station seen
##   <fct> <fct>   <int>
## 1 4842 Release     1
## 2 4842 I80_1      1
## 3 4842 Lisbon     1
## 4 4842 Rstr       1
```

```
fish_encounters_wide ← fish_encounters %>%
  pivot_wider(
    names_from = station,
    values_from = seen
  )
```

Basic Wrangling Functions

group_by() and summarize() pivot_longer() pivot_longer() result

pivot_wider() pivot_wider() result

```
fish_encounters %>% head(n = 4)
```

```
## # A tibble: 4 × 3
##   fish station seen
##   <fct> <fct>   <int>
## 1 4842 Release     1
## 2 4842 I80_1      1
## 3 4842 Lisbon     1
## 4 4842 Rstr       1
```

```
fish_encounters_wide[1:6] %>% head(n = 2)
```

```
## # A tibble: 2 × 6
##   fish Release I80_1 Lisbon Rstr Base_TD
##   <fct>   <int> <int>   <int> <int>   <int>
## 1 4842     1     1     1     1     1
## 2 4843     1     1     1     1     1
```

Working with 'Tidy' Datasets

table1

```
#> # A tibble: 6 x 4
#>   country      year  cases population
#>   <chr>      <int> <int>      <int>
#> 1 Afghanistan 1999     745    19987071
#> 2 Afghanistan 2000    2666    20595360
#> 3 Brazil      1999   37737   172006362
#> 4 Brazil      2000   80488   174504898
#> 5 China       1999  212258  1272915272
#> 6 China       2000  213766  1280428583
```

table2

```
#> # A tibble: 12 x 4
#>   country      year type          count
#>   <chr>      <int> <chr>      <int>
#> 1 Afghanistan 1999 cases          745
#> 2 Afghanistan 1999 population 19987071
#> 3 Afghanistan 2000 cases          2666
#> 4 Afghanistan 2000 population 20595360
#> 5 Brazil      1999 cases          37737
#> 6 Brazil      1999 population 172006362
#> # ... with 6 more rows
```

```
table3
#> # A tibble: 6 x 3
#>   country      year rate
#> * <chr>      <int> <chr>
#> 1 Afghanistan 1999 745/19987071
#> 2 Afghanistan 2000 2666/20595360
#> 3 Brazil       1999 37737/172006362
#> 4 Brazil       2000 80488/174504898
#> 5 China        1999 212258/1272915272
#> 6 China        2000 213766/1280428583
```

Spread across two tibbles

```
table4a # cases
#> # A tibble: 3 x 3
#>   country      `1999` `2000`
#> * <chr>      <int>  <int>
#> 1 Afghanistan    745    2666
#> 2 Brazil        37737   80488
#> 3 China          212258  213766
```

```
table4b # population
#> # A tibble: 3 x 3
#>   country      `1999`      `2000`
#> * <chr>      <int>      <int>
#> 1 Afghanistan 19987071  20595360
#> 2 Brazil      172006362  174504898
#> 3 China       1272915272 1280428583
```


These are all useable versions of the same data. Only one of them, however, is 'tidy'.

What makes a dataset 'tidy'? From Hadley Wickham, Mine Çetinkaya-Rundel, and Garrett Grolemund, [R for Data Science \(2e\) Chapter 5 — Tidy Tidying](#):

1. Each **variable** must have **its own column**.
2. Each **observation** must have **its own row**.
3. Each **value** must have **its own cell**.

Easier to think about when these conditions are *not* met:

- When one variable is spread across multiple columns.
- When one observation is scattered across multiple rows.

`table1` is the tidy version of this dataset. How can we convert the other versions to be tidy?

table2

table3

table4

table4 results

```
table2 %>% head(n = 2)
```

```
## # A tibble: 2 × 4
##   country      year type      count
##   <chr>      <dbl> <chr>    <dbl>
## 1 Afghanistan 1999 cases      745
## 2 Afghanistan 1999 population 19987071
```

```
table2 %>%
  pivot_wider(
    names_from = type,
    values_from = count
  ) %>% head(n = 4)
```

```
## # A tibble: 4 × 4
##   country      year cases population
##   <chr>      <dbl> <dbl>    <dbl>
## 1 Afghanistan 1999    745  19987071
## 2 Afghanistan 2000   2666  20595360
## 3 Brazil      1999  37737  172006362
## 4 Brazil      2000  80488  174504898
```

table2

table3

table4

table4 results

```
table3 %>% head(n = 2)
```

```
## # A tibble: 2 × 3
##   country      year rate
##   <chr>      <dbl> <chr>
## 1 Afghanistan 1999 745/19987071
## 2 Afghanistan 2000 2666/20595360
```

```
table3 %>%
  mutate(
    cases      = as.numeric(str_extract(rate, ".*(?=\\/)")),
    population = as.numeric(str_extract(rate, "(?<=\\/).*"))
  ) %>% select(-rate) %>% head(n = 4)
```

```
## # A tibble: 4 × 4
##   country      year cases population
##   <chr>      <dbl> <dbl>      <dbl>
## 1 Afghanistan 1999    745   19987071
## 2 Afghanistan 2000   2666   20595360
## 3 Brazil      1999  37737  172006362
## 4 Brazil      2000  80488  174504898
```

table2

table3

table4

table4 results

```
table4a %>% head(n = 2)
```

```
## # A tibble: 2 × 3
##   country      `1999` `2000`
##   <chr>        <dbl> <dbl>
## 1 Afghanistan    745   2666
## 2 Brazil        37737  80488
```

```
table4b %>% head(n = 2)
```

```
## # A tibble: 2 × 3
##   country      `1999`      `2000`
##   <chr>        <dbl>        <dbl>
## 1 Afghanistan 19987071  20595360
## 2 Brazil      172006362 174504898
```

table2

table3

table4

table4 results

```
table4a %>%
  pivot_longer(
    cols      = c(`1999`, `2000`),
    names_to  = "year",
    values_to = "cases"
  ) %>%
  left_join(
    table4b %>%
      pivot_longer(
        cols      = c(`1999`, `2000`),
        names_to  = "year",
        values_to = "population"
      )
  ) %>% head(n = 4)
```

```
## Joining with `by = join_by(country, year)`
```

```
## # A tibble: 4 × 4
```

```
##   country      year  cases population
##   <chr>        <chr> <dbl>      <dbl>
## 1 Afghanistan 1999     745   19987071
## 2 Afghanistan 2000    2666   20595360
## 3 Brazil       1999   37737   172006362
## 4 Brazil       2000   80488   174504898
```

Practical Exercise — Using the World Values Survey Dataset

World Values Survey

Background

"The survey, which started in 1981, seeks to use the most rigorous, high-quality research designs in each country. The WVS consists of nationally representative surveys conducted in almost 100 countries which contain almost 90 percent of the world's population, using a common questionnaire. [...] WVS seeks to help scientists and policy makers understand changes in the beliefs, values and motivations of people throughout the world."

Survey Contents

- Social values, attitudes & stereotypes
- Societal well-being
- Social capital, trust and organizational membership
- Economic values
- Corruption
- Migration
- Post-materialist index
- Science & technology
- Religious values
- Security
- Ethical values & norms
- Political interest and political participation
- Political culture and political regimes
- Demography

Today's practical component

1. Successfully run the code in the `session_3.R` script
2. Create your own script and do the following:
 - Find mean values for 'importance in life' variables (Q1-6) for countries in another region than Europe
 - Calculate average 'enthusiasm' for these life subjects in countries in another region than Europe
 - Perform the same analysis, either on European countries or other countries, for another group of indicators in the dataset:
 - Important child qualities: Q7-17
 - Neighbors: Q18-26
 - Statements to agree with: Q27-41

NOTE — You should refer to documentation for the dataset, which can be found at https://mfiorina.github.io/sais_r_course/

Links

Dominic Royé, [“A very short introduction to Tidyverse”](#)

tidyr, [“Pivoting”](#)

Hadley Wickham, Mine Çetinkaya-Rundel & Garrett Grolemund, [R for Data Science, 2e](#)

RStudio, [RStudio Cheatsheets](#)