2. Data Manipulation with dplyr

Jonathan Hersh, PhD (Chapman Argyros School of Business)

10/24/22

Section 2: Outline

1. **R Projects**
2. **Data Analysis**
   - Loading data
   - Glimpse to view
   - Pipe operator
   - slice() to select rows
   - arrange() to order data frame
   - select() to choose variables
   - rename() to rename variables
   - filter() to select rows matching characteristics
   - mutate() to create new variables
   - group_by() and summarize() to create group

glimpse() to summarize the data

```
> # ------------------------------------------------
> # GLIMPSE to summarize data
> # ------------------------------------------------
> # let's summarize the data using the glimpse function
> glimpse(LFS_2019)
Rows: 2,171
Columns: 48
$ row_number                <chr> "2074", "1831", "21", "380", "691", "1087", "2008",…
$ DISTRICT_STR              <fct> Corozal, Corozal, Belize, Stann Creek, Toledo, Tole…
$ DISTRICT_C                <dbl> 1, 1, 3, 5, 6, 6, 1, 2, 6, 3, 6, 3, 1, 2, 1, 1, 6, …
$ URBANRURAL                <fct> 2, 2, 2, 1, 2, 1, 2, 2, 2, 1, 1, 2, 2, 2, 2, 1, 2, …
$ URBAN_RURAL               <fct> Rural, Rural, Rural, Urban, Rural, Urban, Rural, Ru…
$ mFIPS                     <dbl> 1202209, 1200201, 3200106, 5100104, 6201216, 610210…
$ Weight                    <dbl> 27.40347, 39.65228, 92.18530, 37.06186, 23.96898, 1…
$ any_bank_account          <dbl> 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, …
$ no_bank_account           <dbl> 1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, …
$ no_bank_too_far           <dbl> 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, …
$ no_bank_too_expensive     <dbl> 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, …
$ no_bank_no_documents      <dbl> 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, …
$ no_bank_dont_trust        <dbl> 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, …
$ no_bank_no_money          <dbl> 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, …
$ made_recd_digital_payments <dbl> 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, …
$ send_rec_domestic_remit   <dbl> 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, …
$ used_online_banking       <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, …
$ borrowed_formally         <dbl> 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, …
$ borrowed_any              <dbl> 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, …
$ borrowed_but_not_formally <dbl> 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, …
$ urban                     <fct> 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, …
$ tenureTypeOwn             <dbl> 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, …
```

pipe operator %>%

```
# ---------------------------------------------
# Pipe Operator!
# ---------------------------------------------
# The pipe operator "%>%" is super useful!
# It allows us to execute a series of functions on an object in stages
# The general recipe is Data_Frame %>% function1() %>% function2() etc
# Functions are applied right to left

LFS_2019 %>% glimpse()

# cmd/ctrl + shift as a shortcut create the pipe operator
```

slice() to select rows

```
> LFS_2019 %>% slice(1:10)
# A tibble: 10 × 48
   row_n…¹ DISTR…² DISTR…³ URBAN…⁴ URBAN…⁵  mFIPS Weight any_b…⁶ no_ba…⁷ no_ba…⁸ no_ba…⁹
     <dbl> <fct>     <dbl> <fct>   <fct>    <dbl>  <dbl>   <dbl>   <dbl>   <dbl>   <dbl>
 1       1 Corozal       1 1       Urban   1.10e6   23.0       1       0       0       0
 2       2 Corozal       1 2       Rural   1.20e6   26.5       0       1       0       0
 3       3 Toledo        6 1       Urban   6.10e6   11.4       0       1       0       0
 4       4 Orange…       2 1       Urban   2.10e6   39.6       1       0       0       0
 5       5 Corozal       1 1       Urban   1.10e6   23.3       1       0       0       0
 6       6 Orange…       2 1       Urban   2.10e6   32.5       0       1       0       0
 7       7 Orange…       2 1       Urban   2.10e6   21.6       1       0       0       0
 8       8 Cayo          4 1       Urban   4.10e6   75.9       0       1       0       0
 9       9 Toledo        6 1       Urban   6.10e6   10.5       0       1       0       0
10      10 Cayo          4 1       Urban   4.10e6   80.0       1       0       0       0
# … with 37 more variables: no_bank_no_documents <dbl>, no_bank_dont_trust <dbl>,
#   no_bank_no_money <dbl>, made_recd_digital_payments <dbl>,
#   send_rec_domestic_remit <dbl>, used_online_banking <dbl>, borrowed_formally <dbl>,
#   borrowed_any <dbl>, borrowed_but_not_formally <dbl>, urban <fct>,
#   tenureTypeOwn <dbl>, tenureTypeRent <dbl>, outerWallsPoor <dbl>,
#   floorMatPoor <dbl>, toiletPoor <dbl>, elecGrid <dbl>, bedrooms <dbl>, aircon <dbl>,
#   fridges <dbl>, micros <dbl>, washers <dbl>, stereos <dbl>, DVDplayers <dbl>, …
# ℹ Use `colnames()` to see all variable names
```

arrange() to order dataset

```
> LFS_2019 %>%
+   arrange(desc(Weight))
# A tibble: 2,171 × 48
   row_n…¹ DISTR…² DISTR…³ URBAN…⁴ URBAN…⁵  mFIPS Weight any_b…⁶ no_ba…⁷ no_ba…⁸ no_ba…⁹
     <dbl> <fct>     <dbl> <fct>   <fct>    <dbl>  <dbl>   <dbl>   <dbl>   <dbl>   <dbl>
 1      62 Belize        3 1       Urban   3.11e6   144.       1       0       0       0
 2     130 Belize        3 1       Urban   3.10e6   144.       1       0       0       0
 3     153 Belize        3 1       Urban   3.10e6   144.       1       0       0       0
 4     255 Belize        3 1       Urban   3.10e6   144.       0       1       0       0
 5     285 Belize        3 1       Urban   3.10e6   144.       1       0       0       0
 6     490 Belize        3 1       Urban   3.10e6   144.       0       1       0       1
 7     510 Belize        3 1       Urban   3.10e6   144.       1       0       0       0
 8     528 Belize        3 1       Urban   3.10e6   144.       1       0       0       0
 9     550 Belize        3 1       Urban   3.10e6   144.       1       0       0       0
10     647 Belize        3 1       Urban   3.10e6   144.       0       1       0       0
# … with 2,161 more rows, 37 more variables: no_bank_no_documents <dbl>,
#   no_bank_dont_trust <dbl>, no_bank_no_money <dbl>, made_recd_digital_payments <dbl>,
#   send_rec_domestic_remit <dbl>, used_online_banking <dbl>, borrowed_formally <dbl>,
#   borrowed_any <dbl>, borrowed_but_not_formally <dbl>, urban <fct>,
#   tenureTypeOwn <dbl>, tenureTypeRent <dbl>, outerWallsPoor <dbl>,
#   floorMatPoor <dbl>, toiletPoor <dbl>, elecGrid <dbl>, bedrooms <dbl>, aircon <dbl>,
```

select() to select columns in a dataset

```
> # ---------------------------------------------
> # SELECT columns of the dataset using the 'select' function
> # ---------------------------------------------
> # selecting columns using the select() function
> # here we create a subset of the original dataset that only contains
> # director_name and movie title
> LFS_2019_keys <- LFS_2019 %>%  select(DISTRICT_STR, DISTRICT_C)
> glimpse(LFS_2019_keys)
Rows: 2,171
Columns: 2
$ DISTRICT_STR <fct> Corozal, Corozal, Toledo, Orange Walk, Corozal, Orange Walk, Oran…
$ DISTRICT_C   <dbl> 1, 1, 6, 2, 1, 2, 2, 4, 6, 4, 4, 6, 5, 2, 1, 1, 5, 4, 3, 2, 3, 6,…
```

rename() to rename variables in a dataset

```
> # use the rename function to rename variables
> LFS_2019 <-
+   LFS_2019 %>%
+   rename(eduHead = educ_head_of_hh)
> glimpse(LFS_2019)
Rows: 2,171
Columns: 48
$ row_number                <dbl> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, …
$ DISTRICT_STR              <fct> Corozal, Corozal, Toledo, Orange Walk, Corozal, Ora…
$ DISTRICT_C                <dbl> 1, 1, 6, 2, 1, 2, 2, 4, 6, 4, 4, 6, 5, 2, 1, 1, 5, …
$ URBANRURAL                <fct> 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 1, 1, 1, …
$ URBAN_RURAL               <fct> Urban, Rural, Urban, Urban, Urban, Urban, Urban, Ur…
$ mFIPS                     <dbl> 1100104, 1202207, 6100102, 2102652, 1102104, 210365…
$ Weight                    <dbl> 23.00894, 26.46835, 11.36921, 39.55839, 23.34096, 3…
$ any_bank_account          <dbl> 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 1, …
$ no_bank_account           <dbl> 0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, …
$ no_bank_too_far           <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, …
$ no_bank_too_expensive     <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, …
$ no_bank_no_documents      <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, …
$ no_bank_dont_trust        <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, …
```

filter() to remove rows you don’t want

```
> # ONLY select large budget LFS_2019 and store this as a new data frame
> LFS_2019_big <- LFS_2019 %>% filter(numHHmem > 10)
> nrow(LFS_2019_big)
[1] 13
> # ONLY select Corozal households
> LFS_2019_Corozal <- LFS_2019 %>% filter(DISTRICT_STR  == "Corozal")
> nrow(LFS_2019_Corozal)
[1] 369
> dim(LFS_2019_Corozal)
[1] 369  48
```

mutate() to create new variables

```
# ------------------------------------------
# MUTATE to Transform variables in your dataset
# ------------------------------------------

# adding new variables using mutate()
# let's create new varibles log budget and log gross that are
# budget and gross transformed by logarithm
LFS_2019 <-
  LFS_2019 %>%
  mutate(log_numHHmem = log(numHHmem),
         log_numChildren = log(numChildren))
```

Srvyr package to work with survey weights

```
> # ------------------------------------------------
> # Working with survey weights and summary tables
> # ------------------------------------------------
> # http://gdfe.co/srvyr/
> # install.packages('srvyr')
> library(srvyr, warn.conflicts = FALSE)
> LFS_2019 %>%
+   as_survey(weights = c(Weight)) %>%
+   group_by(DISTRICT_STR) %>%
+   summarize(numChildren = survey_mean(numChildren, na.rm = TRUE))
# A tibble: 6 × 3
  DISTRICT_STR numChildren numChildren_se
  <fct>              <dbl>          <dbl>
1 Corozal             1.46         0.0892
2 Orange Walk         1.47         0.0773
3 Belize              1.01         0.0756
4 Cayo                1.31         0.0705
5 Stann Creek         1.19         0.0891
6 Toledo              2.07         0.105
```

Data Wrangling
with dplyr and tidyr
Cheat Sheet

RStudio

Tidy Data - A foundation for wrangling in R

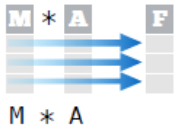
In a tidy data set:

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:

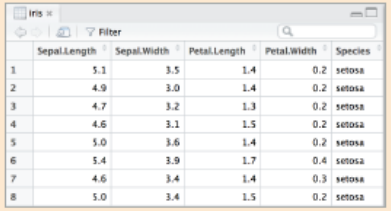
```
Source: local data frame [150 x 5]

   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).



	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5.0	3.6	1.4	0.2	setosa
6	5.4	3.9	1.7	0.4	setosa
7	4.6	3.4	1.4	0.3	setosa
8	5.0	3.4	1.5	0.2	setosa

dplyr::**%>%**

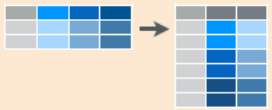
Passes object on left hand side as first argument (or . argument) of function on righthand side.

```
x %>% f(y)  is the same as  f(x, y)
y %>% f(x, ., z)  is the same as  f(x, y, z )
```

"Piping" with %>% makes code more readable, e.g.

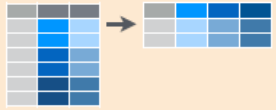
```
iris %>%
  group_by(Species) %>%
  summarise(avg = mean(Sepal.Width)) %>%
  arrange(avg)
```

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.

dplyr::**sample_n(iris, 10, replace = TRUE)**

Randomly select n rows.

dplyr::**slice(iris, 10:15)**

Select rows by position.

dplyr::**top_n(storms, 2, date)**

Select and order top n entries (by group if grouped data).

Logic in R - ?Comparison, ?base::Logic

<	Less than	!=	Not equal to
>	Greater than	%in%	Group membership
==	Equal to	is.na	Is NA
<=	Less than or equal to	!is.na	Is not NA
>=	Greater than or equal to	&, |, !, xor, any, all	Boolean operators

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"))
Select columns whose name ends with a character string.

select(iris, everything())
Select every column.

select(iris, matches(".t."))
Select columns whose name matches a regular expression.

select(iris, num_range("x", 1:5))
Select columns named x1, x2, x3, x4, x5.

select(iris, one_of(c("Species", "Genus")))
Select columns whose names are in a group of names.

select(iris, starts_with("Sepal"))
Select columns whose name starts with a character string.

select(iris, Sepal.Length:Petal.Width)
Select all columns between Sepal.Length and Petal.Width (inclusive).

select(iris, -Species)
Select all columns except Species.

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

https://www.rstudio.com/resources/cheatsheets/

srvyr 1.1.2 | Overview | Reference | Articles | News

srvyr

srvyr brings parts of dplyr's syntax to survey analysis, using the survey package.

srvyr focuses on calculating summary statistics from survey data, such as the mean, total or quantile. It allows for the use of many dplyr verbs, such as `summarize`, `group_by`, and `mutate`, the convenience of pipe-able functions, rlang's style of non-standard evaluation and more consistent return types than the survey package.

You can try it out:

```
install.packages("srvyr")
# or for development version
# remotes::install_github("gergness/srvyr")
```

Example usage

First, describe the variables that define the survey's structure with the function `as_survey()` with the bare column names of the names that you would use in functions from the survey package like `survey::svydesign()`, `survey::svrepdesign()` or `survey::twophase()`.

```
library(srvyr, warn.conflicts = FALSE)
data(api, package = "survey")

dstrata <- apistrat %>%
   as_survey_design(strata = stype, weights = pw)
```

Now many of the dplyr verbs are available.

- `mutate()` adds or modifies a variable.

```
dstrata <- dstrata %>%
  mutate(api_diff = api00 - api99)
```

- `summarise()` calculates summary statistics such as mean, total, quantile or ratio.

Links

Download from CRAN at https://cloud.r-project.org/package=srvyr

Browse source code at https://github.com/gergness/srvyr/

Report a bug at https://github.com/gergness/srvyr/issues

License

GPL-2 | GPL-3

Community

Code of conduct

Developers

Greg Freedman Ellis
Author, maintainer

Ben Schneider
Author, contributor

All authors...

Dev status

CRAN 1.1.2

R-CMD-check failing

documentation click here!

http://gdfe.co/srvyr/