A light purple world map serves as a background. A small red heart is located in the southern part of South America, specifically over Argentina. The title text is centered over the map.

Data Manipulation with dplyr

Ruth Chirinos



Ruth Chirinos

R-Ladies
La Paz, Bolivia



I am a System Engineer with a Master Degree in Management of Information Systems. Currently, I work at Mojix as Development Leader in IoT projects with NoSQL databases.

Speaking about communities, I am ambassador of City.AI and R-Ladies La Paz and I am an active member of communities: JUG La Paz, and Google Developer Group (GDG), Women in Data Science (WiDS) and Ocean Protocol. These communities give me the place where I can learn and share my knowledge to other people.

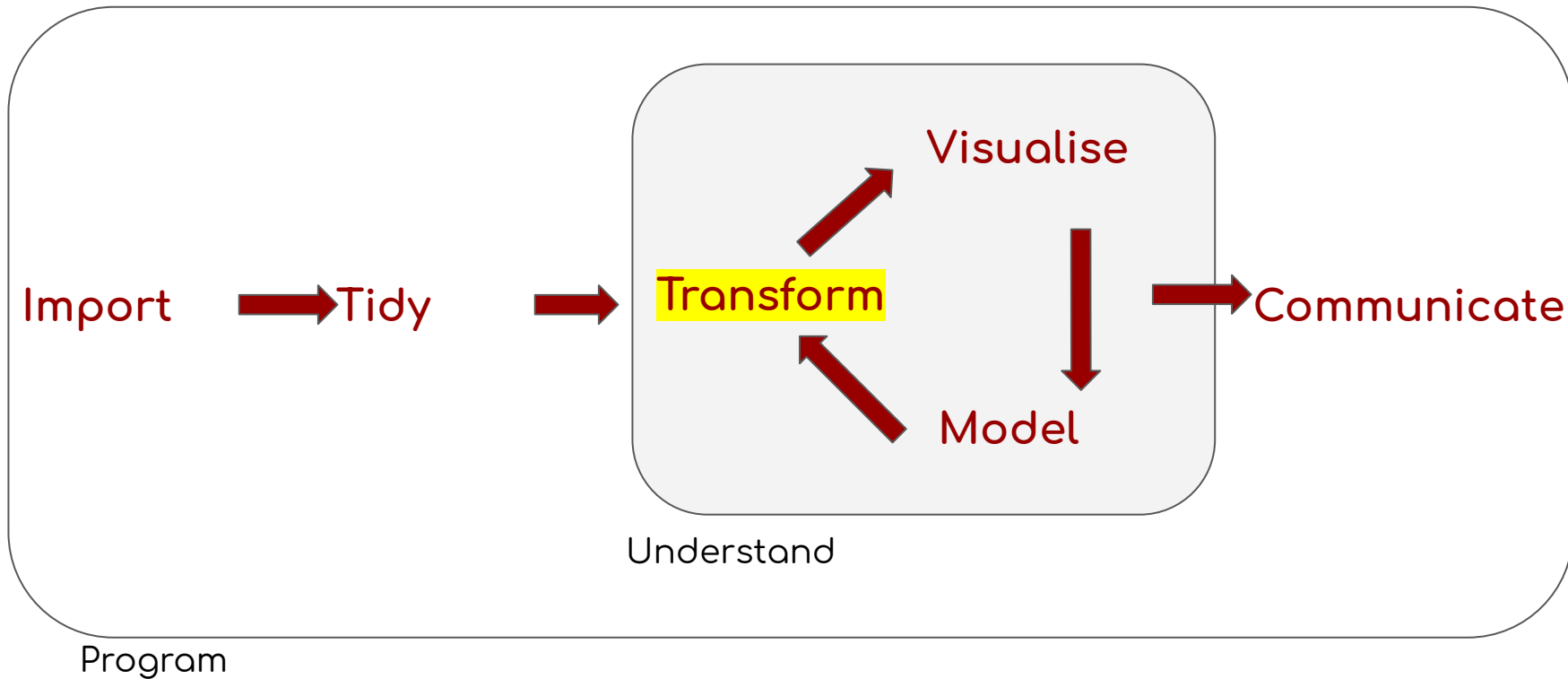
Founder of Akath, enterprise focus on data analysis - data science. So, another important thing for me is AI & Blockchain, this is why I have participated in startups with AI technology and one with Blockchain , so I want to meet with people who want to do great things with these technologies.

Program

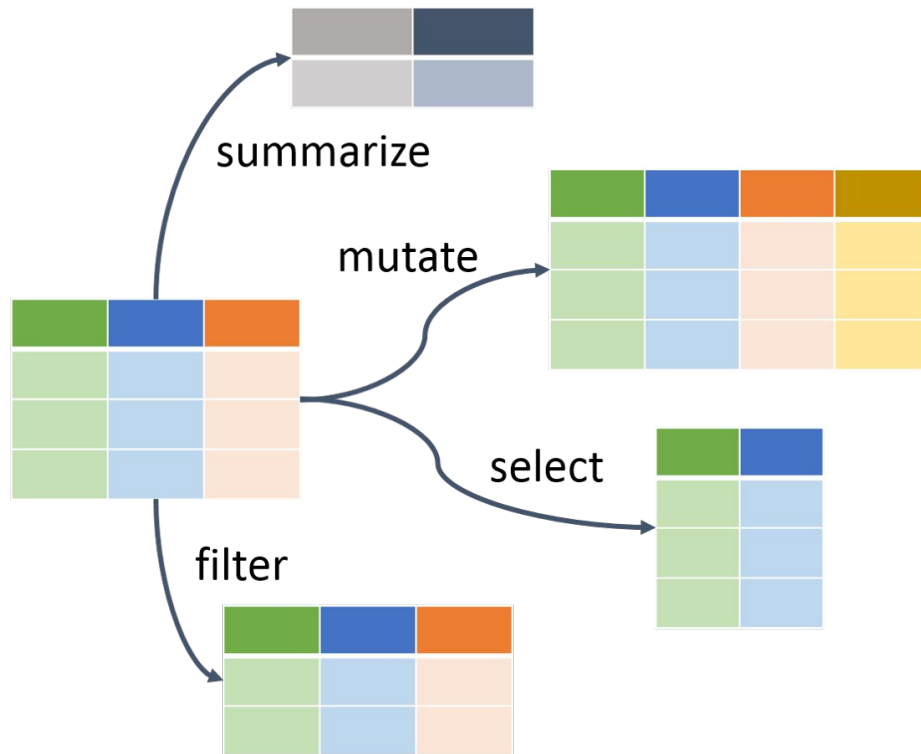
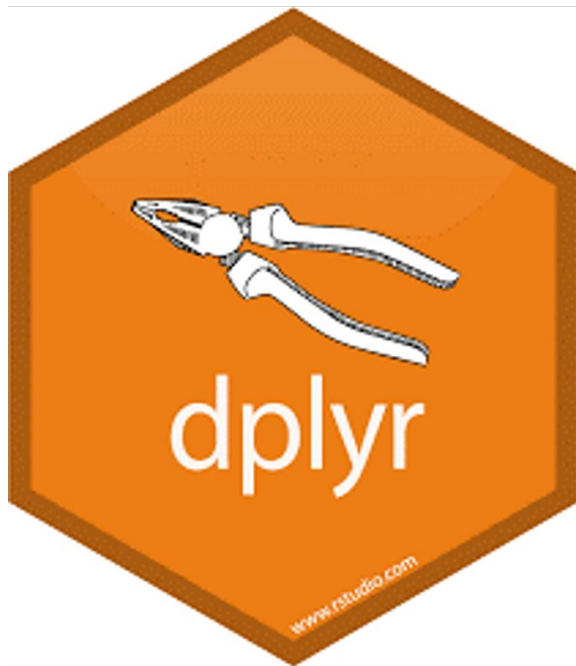
- Fundamentals of **dplyr**
- **filter()**
- **arrange()**
- **select()**
- **mutate()**
- **group_by()**
- **summarise()** with **group_by()**

Fundamentals of dplyr

Tidy Data (Cont.)



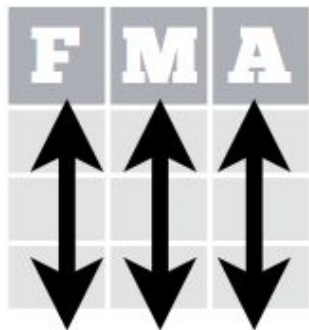
Fundamentals of dplyr



Fundamentals of dplyr

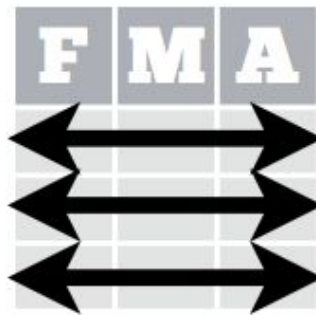
Tidy Data

In a tidy data set:



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

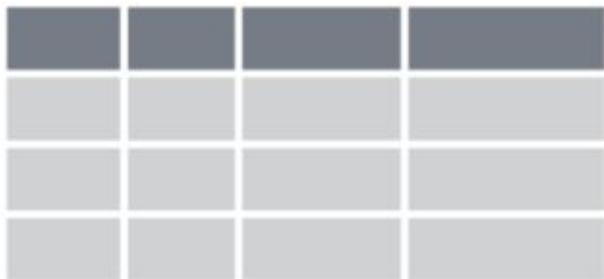
&



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

Fundamentals of dplyr

Tidy Data (Cont.)



region	state	code	park_name	type	visitors	year
PW	CA	CHIS	Channel Islands National Park	National Park	1200	1963
PW	CA	CHIS	Channel Islands National Park	National Park	1500	1964
PW	CA	CHIS	Channel Islands National Park	National Park	1600	1965
PW	CA	CHIS	Channel Islands National Park	National Park	300	1966
PW	CA	CHIS	Channel Islands National Park	National Park	15700	1967
PW	CA	CHIS	Channel Islands National Park	National Park	31000	1968
PW	CA	CHIS	Channel Islands National Park	National Park	33100	1969
PW	CA	CHIS	Channel Islands National Park	National Park	32000	1970

Fundamentals of dplyr

5 main functions in dplyr

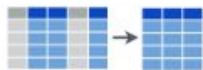
- `filter()` : pick observations by their values

Subset Observations (Rows)



- `select()` : pick variables by their names

Subset Variables (Columns)



- `mutate()` : create new variables with functions of existing variables

Make New Variables

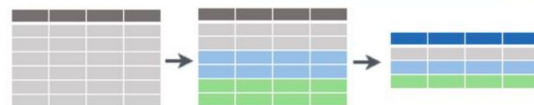


- `summarise()` : collapse many values down to a single summary

Summarise Data



- `arrange()` : reorder the rows



`group_by()`

Fundamentals of dplyr

Similarity among verbs

```
function_verbs( dataframe , what to do with the dataframe)
```

For example:

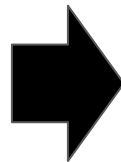
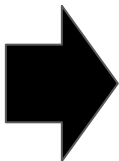
```
filter( flights , month == 1, day == 1)
```

Fundamentals of dplyr

The process

	foo	bar	baz	zoo
0	one	A	1	x
1	one	B	2	y
2	one	C	3	z
3	two	A	4	q
4	two	B	5	w
5	two	C	6	t

INPUT: Data Frame



bar	A	B	C
foo			
one	1	2	3
two	4	5	6

OUTPUT: Data Frame

```
filter()
select()
mutate()
summarise()
arrange()
```

Fundamentals of dplyr

Tidy Data (Cont.)

- **int** stands for integers.
- **dbl** stands for doubles, or real numbers.
- **chr** stands for character vectors, or strings.
- **dtm** stands for date-times (a date + a time).
- **lgl** stands for logical, vectors that contain only TRUE or FALSE.
- **factr** stands for factors, which R uses to represent categorical variables with fixed possible values.
- **date** stands for dates.

Let's analyze our data.....

filter()

Subset Observations (Rows)



```
filter(flights, month == 1, day == 1)
```

Filter

Comparison

- Use ==

```
filter(flights, month = 1)  
#> Error: `month` (`month = 1`) must not be named, you need `==`
```

- Floating Point Numbers

```
sqrt(2) ^ 2 == 2  
#> [1] FALSE
```



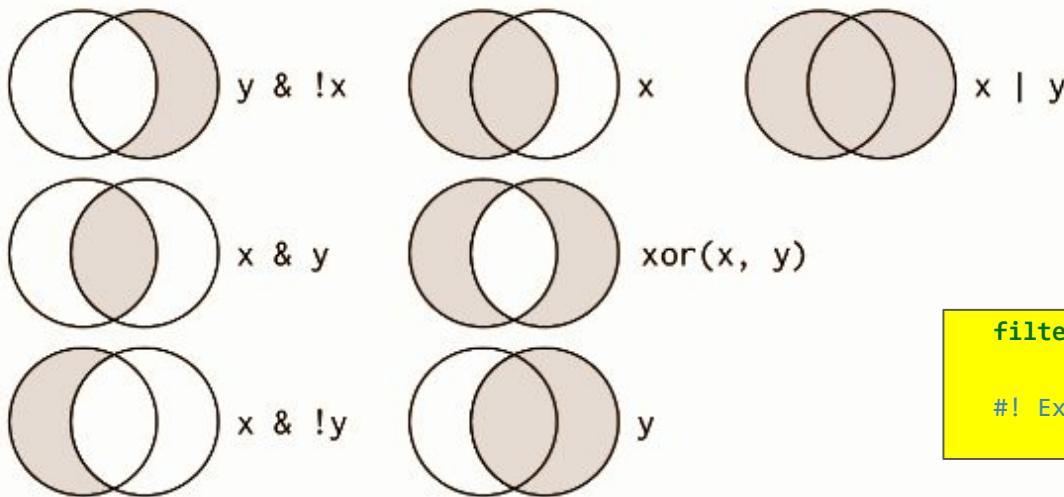
square root of a number
Computers use finite precision arithmetic

```
near(sqrt(2) ^ 2, 2)  
#> [1] TRUE
```



Filter

Logical Operators



```
filter(flights, month %in% c(11, 12))
```

```
#! Exclamation
```

```
filter(flights, month == 11 | month == 12)
```

```
filter(flights, !(arr_delay > 120 | dep_delay > 120))
```

```
filter(flights, arr_delay <= 120, dep_delay <= 120)
```

Filter

Missing Values (NA -> “not availables”)

```
NA > 5  
#> [1] NA
```

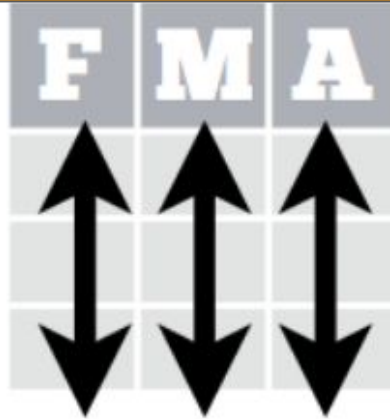
```
10 == NA  
#> [1] NA
```

```
NA + 10  
#> [1] NA
```

```
NA / 2  
#> [1] NA
```

NA represents an unknown value so missing values are “contagious”

arrange()



ASC
DESC

```
arrange(flights, year, month, day)
```


Arrange (Cont.)

```
arrange( flights, year, month, day )
```

- Use desc() to re-order by a column in descending order:

```
arrange( flights, desc(dep_delay) )
```

- Missing values are always sorted at the end:

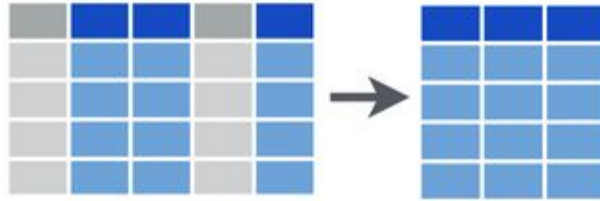
```
df <- tibble( x = c (5, 2, NA) )
```

```
arrange(df, x)
```

```
arrange(df, desc(x))
```

select()

Subset Variables (Columns)



Select (Cont.)

- Select columns by name

```
select(flights, year, month, day)
```

- Select all columns between year and day (inclusive)

```
select(flights, year:day)
```

- Select all columns except those from year to day (inclusive)

```
select(flights, -(year:day))
```

Select (Cont.)

- `starts_with("abc")`: matches names that begin with “abc”.
- `ends_with("xyz")`: matches names that end with “xyz”.
- `contains("ijk")`: matches names that contain “ijk”.
- `matches("(.)\\1")`: selects variables that match a regular expression. This one matches any variables that contain repeated characters.
- `num_range("x", 1:3)`: matches x1, x2 and x3.

Select (Cont.)

- Rename

```
rename(flights, tail_num = tailnum)
```

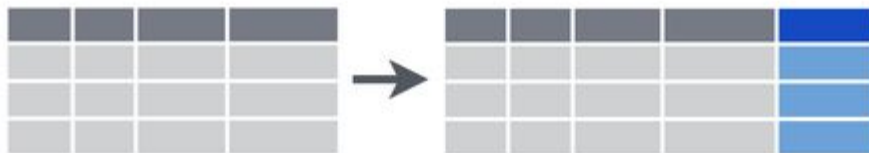
Use: new_name = old_name

- Everything

```
select(flights, time_hour, air_time, everything())
```

mutate()

Make New Variables



Mutate

Besides selecting sets of existing columns, it's often useful to add new columns that are functions of existing columns. That's the job of `mutate()`.

Mutate

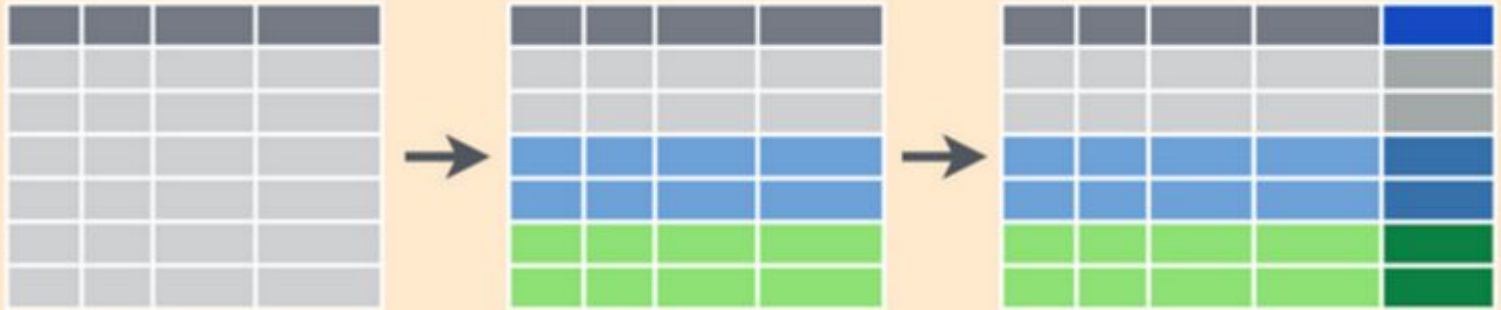
Useful creation functions

- Arithmetic operators: `+`, `-`, `*`, `/`, `^`
- Modular arithmetic: `%/%` (integer division) and `%%` (remainder), where `x == y * (x %/% y) + (x %% y)`
- Logs: `log()`, `log2()`, `log10()`.
- Offsets: `lead()` and `lag()`
- Cumulative and rolling aggregates: `cumsum()`, `cumprod()`, `cummin()`, `cummax()`, `cummean()`
- Logical comparisons, `<`, `<=`, `>`, `>=`, `!=`, and `==`
- Ranking: `min_rank()`, `row_number()`, `dense_rank()`, `percent_rank()`, `cume_dist()`, `ntile()`

group_by()

Group Data

Compute new variables by group.



Group By

- It collapses a data frame to a single row:

```
(by_day <- group_by(flights, year, month, day))
```

summarise() with group_by()

Summarise Data



summarise()

Useful summarise functions

- Group By
- Combining multiple operations with the pipe
- Missing values
- Useful summary functions
- Grouping by multiple variables
- Ungrouping
- Grouped mutates (and filters)

Key Points

- Data manipulation functions in **dplyr** allow you to **filter()** by rows and **select()** by columns, create new columns with **mutate()**, and **group_by()** unique column values to apply **summarize()** for new columns that define aggregate values across groupings.
- The **“then”** operator **%>%** allows you to chain successive operations without needing to define intermediary variables for creating the most parsimonious, easily read analysis.



Thank you!



@ruthy_root



ruth-rosario-chirinos-contreras

