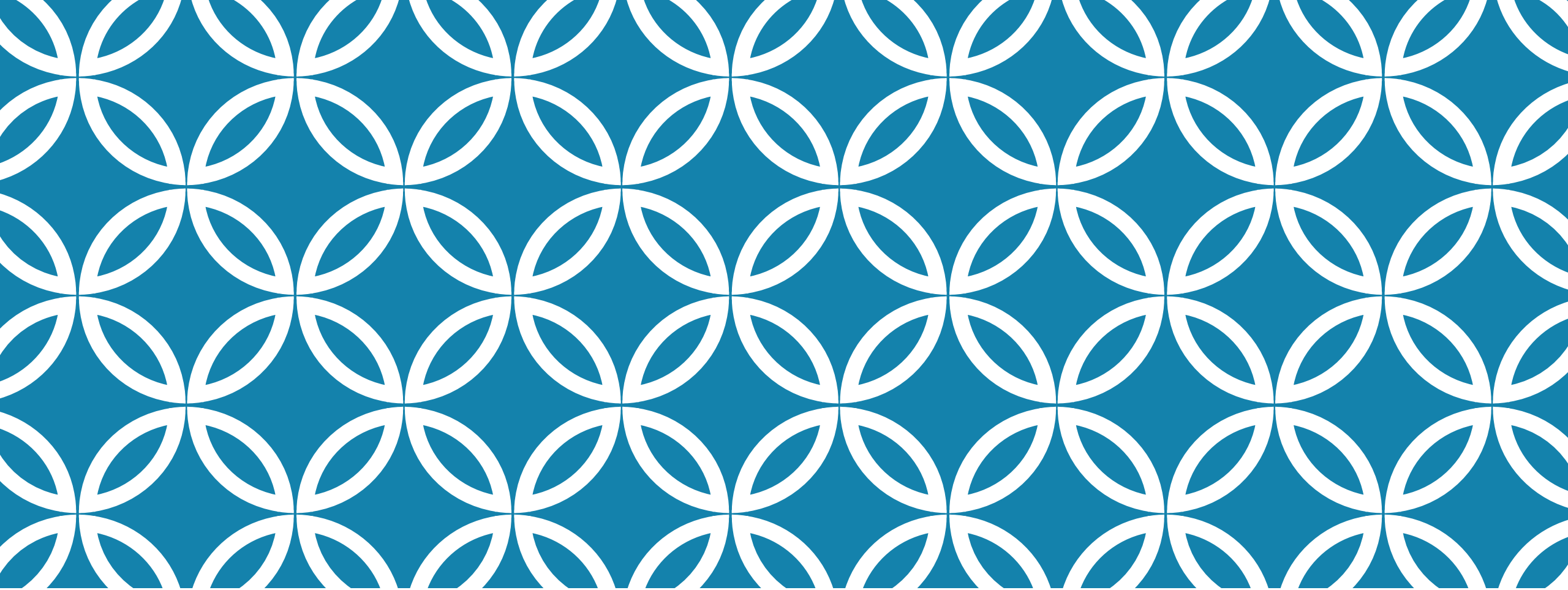


# Data Wrangling in R

Session 3

**Amrom Obstfeld**

Introduction to R Workshop

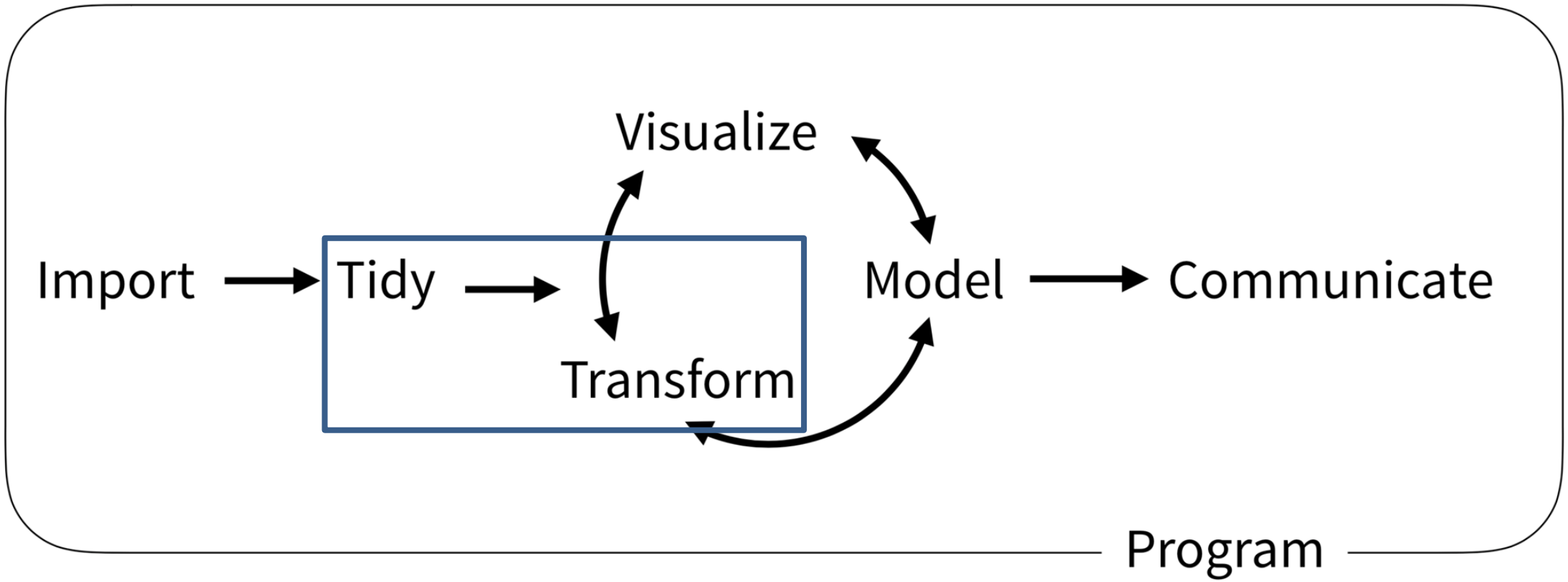
May 6, 2019

| | |
|---|---|
| 7:00 am–8:00 am | BREAKFAST BALLROOM LOBBY 2ND FLOOR |
| 8:00 am–8:10 am | Instructor and Course Introduction |
| 8:10 am–9:50 am | Introduction to R and RStudio for Reproducible Reporting |
| 9:50 am–10:10 am | REFRESHMENT BREAK BALLROOM LOBBY 2ND FLOOR |
| 10:10 am–11:50 am | Data Wrangling |
| 12:00 pm–1:00 pm | LUNCH BALLROOM LOBBY 2ND FLOOR |
| 1:00 pm–2:50 pm | Data Understanding |
| 2:50 pm–3:10 pm | REFRESHMENT BREAK BALLROOM LOBBY 2ND FLOOR |
| 3:10 pm–5:00 pm | Exploratory Data Analysis |

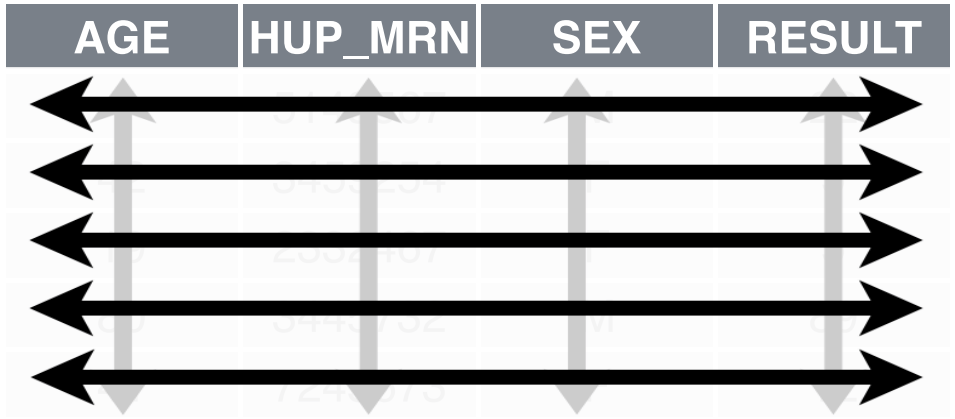
# Goals and Objectives

Learn how to:

- Select and filter data of interest from a large dataset
- Refine data by deriving features of interest from raw data


Visualize
Import
Tidy
Transform
Model
Communicate
Program

# What is a “Tidy” Data Frame




A data set is **tidy** if:

1. Each **variable** is in its own **column**
2. Each **observation** is in its own **row**
3. Each **value** is in its own **cell**

# Transform Data with

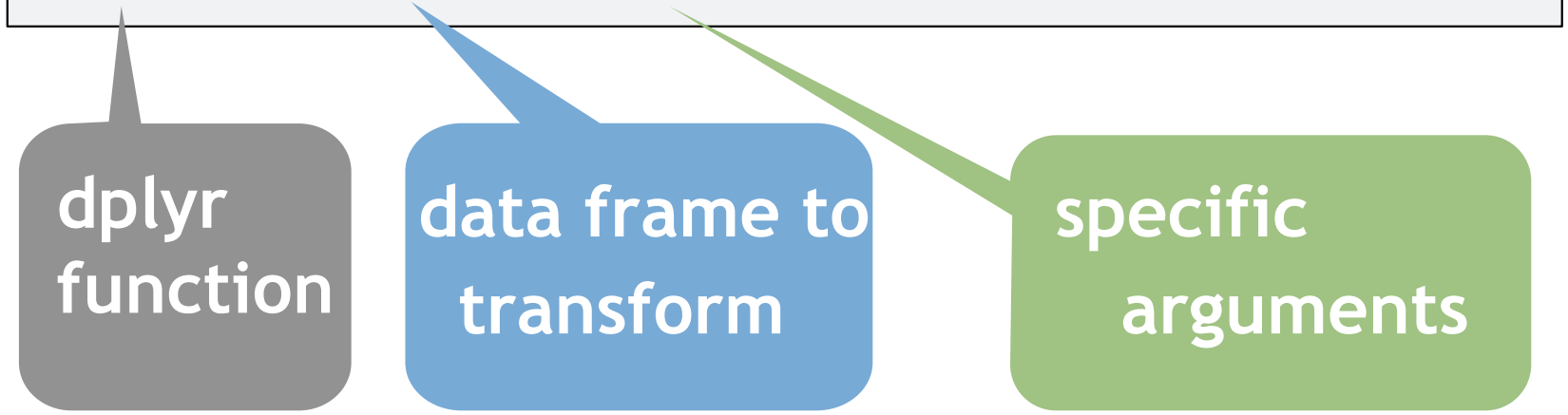
# dplyr


dplyr implements a *grammar* for transforming tabular data.

# common syntax

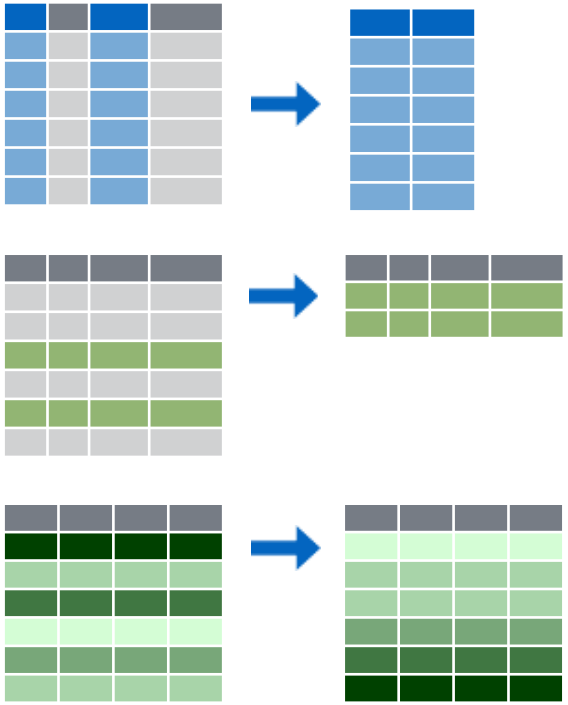
Each function takes a data frame as its first argument and returns a data frame as its output.

```
filter(data, ...)
```



# Isolating data

# Isolating data



Extract columns with **select()**

Extract rows with **filter()**

Arrange rows, with **arrange()**.

# Your Turn 1

## Open "**03-Transform.Rmd**"

## Run the setup chunk

```
```{r setup}
library(tidyverse)  # Provides functions used throughout this session
library(readxl) # Provides function for reading in excel workbooks
orders <- read_excel("data/orders_data_set.xlsx")
```
```

# select()

# select()

Extract columns from a data frame

# select()

Extract columns from a data frame

```
select(data, ... )
```

**dplyr function**

**data frame to transform**

**name(s) of columns to extract**

(or a select helper)

# select()

Extract columns by name.

```
select(orders, description, department)
```

**data frame to transform**

**name(s) of columns to extract**

(or a select helper)

# select()

Extract columns by name.

```
select(orders, description, department)
```

orders

| order_id | patient_id | description | proc_code | department |
|---|---|---|---|---|
| 19766 | 511388 | PROTHROMBIN TIME | PRO | INTERNAL MEDICINE CLINIC |
| 88444 | 511388 | BASIC METABOLIC PANEL | BMP | INTERNAL MEDICINE CLINIC |
| 40477 | 508061 | THYROID STIMULATING HORMONE | TSH | ENDOCRINOLOGY CLINIC |
| 97641 | 508061 | T4, FREE | T4FR | ENDOCRINOLOGY CLINIC |

| description | department |
|---|---|
| PROTHROMBIN TIME | INTERNAL MEDICINE CLINIC |
| BASIC METABOLIC PANEL | INTERNAL MEDICINE CLINIC |
| THYROID STIMULATING HORMONE | ENDOCRINOLOGY CLINIC |
| T4, FREE | ENDOCRINOLOGY CLINIC |

# select()

c() is a little function in R that combines two or more values into a vector

Extract columns by index.

```
select(orders, c(1,4))
```

orders

| order_id | patient_id | description | proc_code | department |
|---|---|---|---|---|
| 19766 | 511388 | PROTHROMBIN TIME | PRO | INTERNAL MEDICINE CLINIC |
| 88444 | 511388 | BASIC METABOLIC PANEL | BMP | INTERNAL MEDICINE CLINIC |
| 40477 | 508061 | THYROID STIMULATING HORMONE | TSH | ENDOCRINOLOGY CLINIC |
| 97641 | 508061 | T4, FREE | T4FR | ENDOCRINOLOGY CLINIC |

| order_id | proc_code |
|---|---|
| 19766 | PRO |
| 88444 | BMP |
| 40477 | TSH |
| 97641 | T4FR |

# Your Turn 2

- Alter the code to select just the **order_status_c** column using (1) column name and (2) column number
- Assign the output to a new object "orders_2"

```
orders_2 <- select(orders, _____________)
```

```
select(orders, order_status_c)
select(orders, 8)
```

| order_status_c <dbl> |
|---|
| 4 |
| 4 |
| 5 |
| 5 |
| 5 |
| 5 |
| 5 |
| 5 |
| 5 |
| 5 |

1-10 of 45,002 rows

# select() helpers

**:** Select range of columns

```
select(orders, order_id:lab_status_c)
```

**–** Select every column but

```
select(orders, -c(description, order_status_c))
```

**starts_with()** Select columns that start with…

```
select(orders, starts_with("order"))
```

**ends_with()** Select columns that end with…

```
select(orders, ends_with(“descr"))
```

# select() helpers

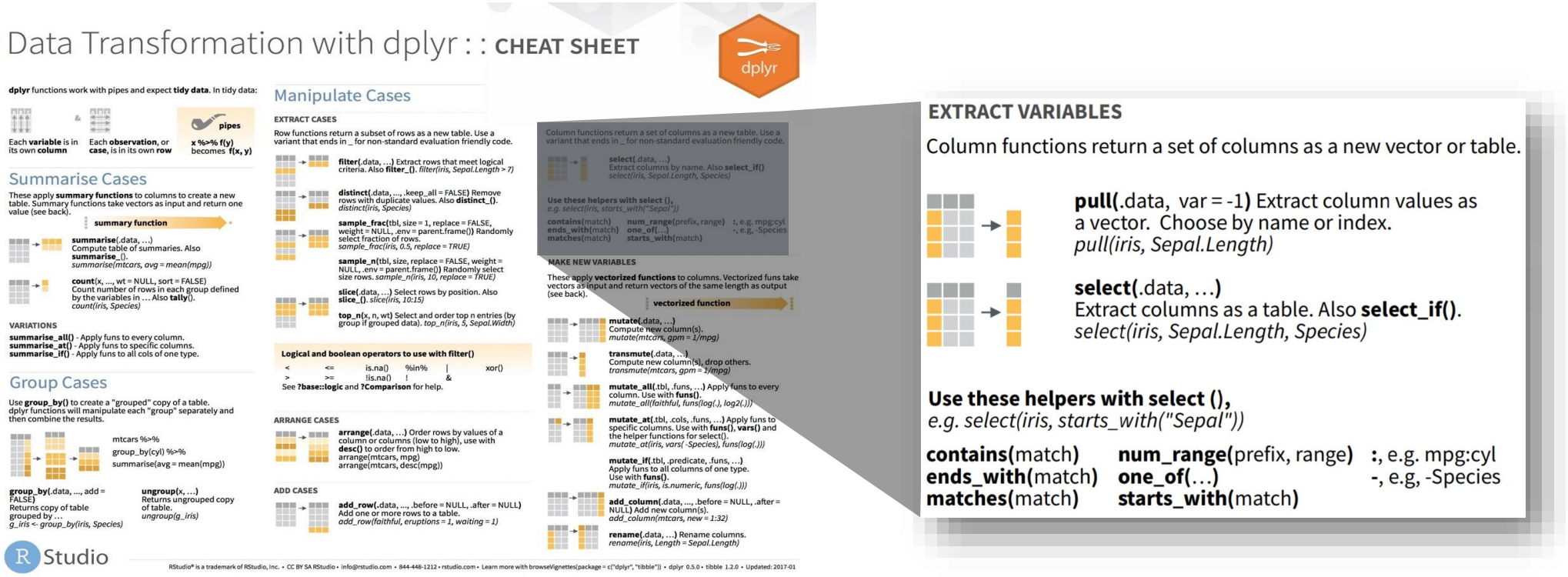
**contains()** Select columns whose names contain…

```
select(orders, contains("time"))
```

**matches()** Select columns whose names match regular expression

```
select(orders, matches("^.{4}$"))
```

# select() helpers



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. Summary functions take vectors as input and return one value (see back).

summary function

**summarise**(.data, …) Compute table of summaries. Also **summarise_()**. *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.
**summarise_at()** - Apply funs to specific columns.
**summarise_if()** - Apply funs to all cols of one type.

## Group Cases

Use **group_by()** to create a "grouped" copy of a table. dplyr functions will manipulate each "group" separately and then combine the results.

```
mtcars %>%
group_by(cyl) %>%
summarise(avg = mean(mpg))
```

**group_by**(.data, ..., add = FALSE) Returns copy of table grouped by … *g_iris <- group_by(iris, Species)*

**ungroup**(x, …) Returns ungrouped copy of table. *ungroup(g_iris)*

## Manipulate Cases

EXTRACT CASES

Row functions return a subset of rows as a new table. Use a variant that ends in _ for non-standard evaluation friendly code.

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

**distinct**(.data, ..., .keep_all = FALSE) Remove rows with duplicate values. Also **distinct_()**. *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. Also **slice_()**. *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)*

**Logical and boolean operators to use with filter()**

| | | | | | |
|---|---|---|---|---|---|
| < | <= | is.na() | %in% | \| | xor() |
| > | >= | !is.na() | ! | & | |

See **?base::logic** and **?Comparison** for help.

ARRANGE CASES

**arrange**(.data, …) Order rows by values of a column or columns (low to high), use with **desc()** to order from high to low.
*arrange(mtcars, mpg)*
*arrange(mtcars, desc(mpg))*

ADD CASES

**add_row**(.data, ..., .before = NULL, .after = NULL) Add one or more rows to a table. *add_row(faithful, eruptions = 1, waiting = 1)*

Column functions return a set of columns as a new table. Use a variant that ends in _ for non-standard evaluation friendly code.

**select**(.data, …) Extract columns by name. Also **select_if()** *select(iris, Sepal.Length, Species)*

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

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

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

**mutate**(.data, …) Compute new column(s). *mutate(mtcars, gpm = 1/mpg)*

**transmute**(.data, …) Compute new column(s), drop others. *transmute(mtcars, gpm = 1/mpg)*

**mutate_all**(.tbl, .funs, …) Apply funs to every column. Use with **funs()**. *mutate_all(faithful, funs(log(.), log2(.)))*

**mutate_at**(.tbl, .cols, .funs, …) Apply funs to specific columns. Use with **funs()**, **vars()** and the helper functions for select(). *mutate_at(iris, vars(-Species), funs(log(.)))*

**mutate_if**(.tbl, .predicate, .funs, …) Apply funs to all columns of one type. Use with **funs()**. *mutate_if(iris, is.numeric, funs(log(.)))*

**add_column**(.data, ..., .before = NULL, .after = NULL) Add new column(s). *add_column(mtcars, new = 1:32)*

**rename**(.data, …) Rename columns. *rename(iris, Length = Sepal.Length)*

RStudio

RStudio® is a trademark of RStudio, Inc. • CC BY SA RStudio • info@rstudio.com • 844-448-1212 • rstudio.com • Learn more with browseVignettes(package = c("dplyr", "tibble")) • dplyr 0.5.0 • tibble 1.2.0 • Updated: 2017-01

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

dplyr

# Consider 1

Which of these is NOT a way to *remove* the columns that represent status codes (i.e. codes for lab status, order status, and cancelation)?

```
select(orders, -ends_with("_c"))

select(orders, -c(lab_status_c, order_status_c, reason_for_canc_c))

select(orders, -c(6,8,10))

select(orders, -contains("status"))
```

# Consider 1

Which of these is NOT a way to *remove* the columns that represent status codes (i.e. codes for lab status, order status, and cancelation)?

```
select(orders, -ends_with("_c"))

select(orders, -c(lab_status_c, order_status_c, reason_for_canc_c))

select(orders, -c(6,8,10))

select(orders, -contains("status"))
```

# select()
# also helpful for renaming

```
select(orders,
       desc = description,
       dept = department)
```

orders

| order_id | patient_id | description | proc_code | department |
|---|---|---|---|---|
| 19766 | 511388 | PROTHROMBIN TIME | PRO | INTERNAL MEDICINE CLINIC |
| 88444 | 511388 | BASIC METABOLIC PANEL | BMP | INTERNAL MEDICINE CLINIC |
| 40477 | 508061 | THYROID STIMULATING HORMONE | TSH | ENDOCRINOLOGY CLINIC |
| 97641 | 508061 | T4, FREE | T4FR | ENDOCRINOLOGY CLINIC |

| desc | dept |
|---|---|
| PROTHROMBIN TIME | INTERNAL MEDICINE CLINIC |
| BASIC METABOLIC PANEL | INTERNAL MEDICINE CLINIC |
| THYROID STIMULATING HORMONE | ENDOCRINOLOGY CLINIC |
| T4, FREE | ENDOCRINOLOGY CLINIC |

# rename()

```
rename(orders,
       desc = description,
       dept = department)
```

orders

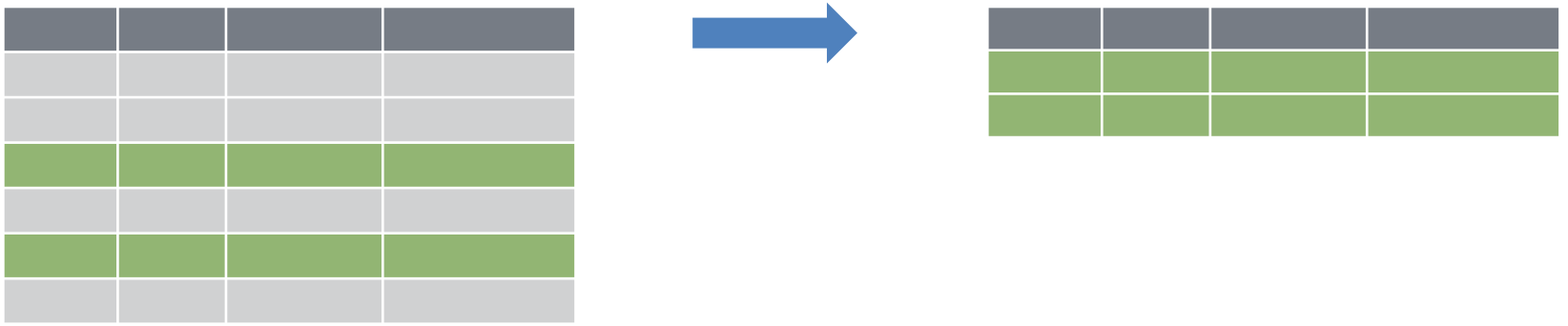
| order_id | patient_id | description | proc_code | department |
|---|---|---|---|---|
| 19766 | 511388 | PROTHROMBIN TIME | PRO | INTERNAL MEDICINE CLINIC |
| 88444 | 511388 | BASIC METABOLIC PANEL | BMP | INTERNAL MEDICINE |
| 40477 | 508061 | THYROID STIMULATING HORMONE | TSH | EN |
| 97641 | 508061 | T4, FREE | T4FR | EN |

| order_id | patient_id | desc | proc_code | dept |
|---|---|---|---|---|
| 19766 | 511388 | PROTHROMBIN TIME | PRO | INTERNAL MEDICINE CLINIC |
| 88444 | 511388 | BASIC METABOLIC PANEL | BMP | INTERNAL MEDICINE CLINIC |
| 40477 | 508061 | THYROID STIMULATING HORMONE | TSH | ENDOCRINOLOGY CLINIC |
| 97641 | 508061 | T4, FREE | T4FR | ENDOCRINOLOGY CLINIC |

# filter()

# filter()

Extract rows that meet logical criteria.



# filter()

Extract rows that meet logical criteria.

```
filter(data, ... )
```

**data frame to transform**

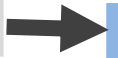
**one or more logical tests** (filter returns each row for which the test is TRUE)

# filter()

Extract rows that meet logical criteria.

```
filter(orders, patient_id==508061)
```

| order_id | patient_id | description | proc_code |
|---|---|---|---|
| 19766 | 511388 | PROTHROMBIN TIME | PRO |
| 88444 | 511388 | BASIC METABOLIC PANEL | BMP |
| 40477 | 508061 | THYROID STIMULATING HORMONE | TSH |
| 97641 | 508061 | T4, FREE | T4FR |



| order_id | patient_id | description | proc_code |
|---|---|---|---|
| 40477 | 508061 | THYROID STIMULATING HORMONE | TSH |
| 97641 | 508061 | T4, FREE | T4FR |

# filter()

Extract rows that meet logical criteria.

```
filter(orders, patient_id == 508061)
```

| order_id | patient_id | description | proc_code |
|---|---|---|---|
| 19766 | 511388 | PROTHROMBIN TIME | PRO |
| 88444 | 511388 | BASIC METABOLIC PANEL | BMP |
| 40477 | 508061 | THYROID STIMULATING HORMONE | TSH |
| 97641 | 508061 | T4, FREE | T4FR |

**= sets**

(returns nothing)

**== tests if equal**

(returns TRUE or FALSE)

# filter()

Values coded as character strings must be surrounded by quotes

Extract rows that meet logical criteria.

```
filter(orders, proc_code=="BMP")
```

| order_id | patient_id | description | proc_code |
|---|---|---|---|
| 19766 | 511388 | PROTHROMBIN TIME | PRO |
| 88444 | 511388 | BASIC METABOLIC PANEL | BMP |
| 40477 | 508061 | THYROID STIMULATING HORMONE | TSH |
| 97641 | 508061 | T4, FREE | T4FR |

| order_id | patient_id | description | proc_code |
|---|---|---|---|
| 88444 | 511388 | BASIC METABOLIC PANEL | BMP |
| 55526 | 511303 | BASIC METABOLIC PANEL | BMP |
| 69809 | 509686 | BASIC METABOLIC PANEL | BMP |
| 24316 | 503847 | BASIC METABOLIC PANEL | BMP |

# Logical tests

| | |
|---|---|
| `x < y` | Less than |
| `x > y` | Greater than |
| `x == y` | Equal to |
| `x <= y` | Less than or equal to |
| `x >= y` | Greater than or equal to |
| `x != y` | Not equal to |
| `x %in% y` | Group membership |
| `is.na(x)` | Is NA |
| `!is.na(x)` | Is not NA |

# Your Turn 3

Use filter() with the logical operators to find:

- Every **order_id** that is greater than 100000
- All of the orders where **lab_status_c_descr** is equal to "Final result"
- CHALLENGE:
  - All of the orders where **reason_for_canc_c_descr** is not NA

```
filter(orders, order_id > 100000)
```

0 rows | 1-4 of 17 columns

```
filter(orders, lab_status_c_descr == "Final result")
```

| order_id <dbl> | patient_id <dbl> | description <chr> |
|---|---|---|
| 61321 | 512524 | HEPATIC FUNCTION PANEL |
| 25091 | 513662 | STOOL C/S AND ENTERIC BATTERY |
| 87734 | 509059 | STOOL C/S AND ENTERIC BATTERY |
| 68531 | 513662 | GIARDIA ANTIGEN (STOOL) |
| 76843 | 504395 | HIV GENOTYPIC RESISTANCE ASSAY |

```
filter(orders, !is.na(reason_for_canc_c_descr))
```

| order_id <dbl> | patient_id <dbl> | description <chr> |
|---|---|---|
| 19766 | 511388 | PROTHROMBIN TIME |
| 88444 | 511388 | BASIC METABOLIC PANEL |
| 34373 | 511303 | IRON, SERUM |
| 79887 | 510902 | CBC (HEMOGRAM) |
| 50728 | 501184 | COMPREHENSIVE METABOLIC PANEL |

# Common mistakes

1. Using `=` instead of `==`

```
filter(orders, proc_code = "BMP")
filter(orders, proc_code == "BMP")
```

2. Forgetting quotes

```
filter(orders, proc_code == COMP)
filter(orders, proc_code == "COMP")
```

# Common mistakes

3. Capitalization matters

```
Filter(Orders, proc_code == "BMP")
filter(orders, proc_code == "BMP")
```

4. Spelling

```
fitler(orders, proc_code == "COMP")
filter(orders, proc_code == "COMP")
```

# filter()

Extract rows that meet *multiple* logical criteria.

```
filter(orders, patient_id == 508061, description=="T4, FREE")
```

orders

| order_id | patient_id | description | proc_code |
|---|---|---|---|
| 19766 | 511388 | PROTHROMBIN TIME | PRO |
| 88444 | 511388 | BASIC METABOLIC PANEL | BMP |
| 40477 | 508061 | THYROID STIMULATING HORMONE | TSH |
| 97641 | 508061 | T4, FREE | T4FR |

→

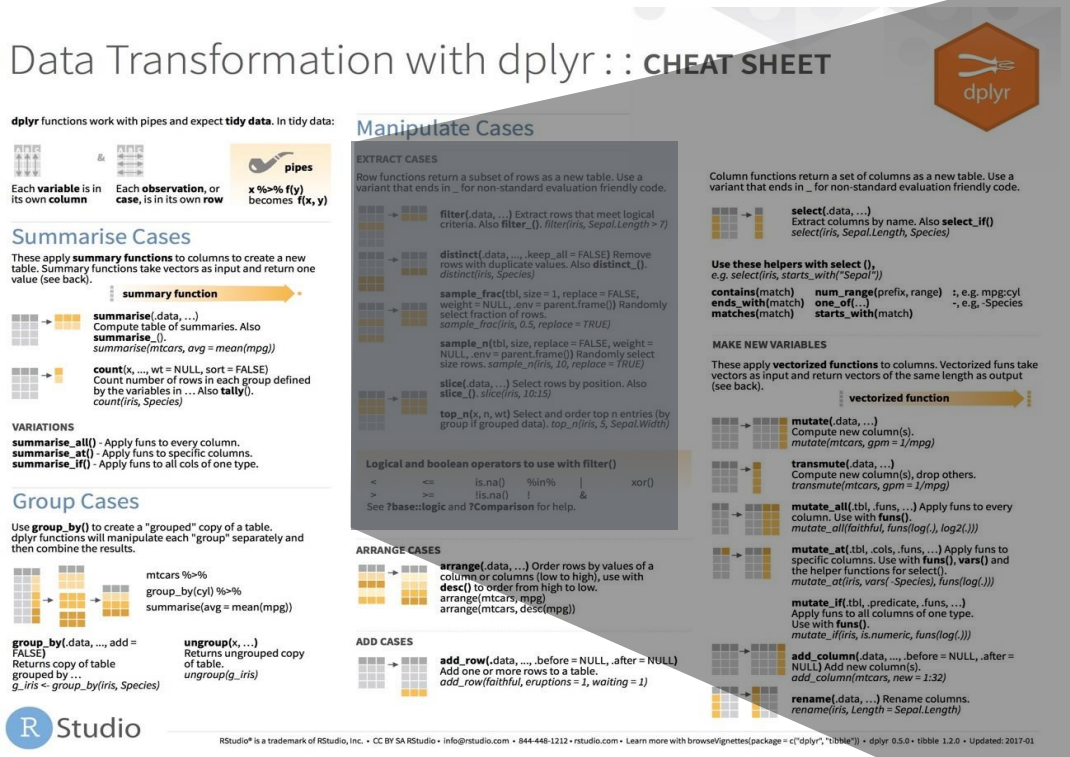
| order_id | patient_id | description | proc_code |
|---|---|---|---|
| 97641 | 508061 | T4, FREE | T4FR |

# Boolean operators

?base::Logic

| | |
|---|---|
| `a & b` | and |
| `a \| b` | or |
| `!a` | not |
| `a %in% c(a, b)` | one of (in) |

# filter() variants




## 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)*

**Logical and boolean operators to use with filter()**

| | | | | | |
|---|---|---|---|---|---|
| < | <= | is.na() | %in% | \| | xor() |
| > | >= | !is.na() | ! | & | |

See **?base::logic** and **?Comparison** for help.

# Your Turn 4

Write 3 commands that use Boolean operators and filter() to return rows that contain:

- Orders for patient number 510909 with **proc_code** TSH
- Orders for any of one of the following **departments**: OB GYN CLINIC, GERIATRIC CLINIC, or PEDIATRIC CLINICS
- Orders for tests that were canceled and originally chosen from a preference list (HINT: These are coded in the **pref_list_type** column)

```
filter(orders, proc_code == "TSH", patient_id == "510909")
```

| order_id <dbl> | patient_id <dbl> | description <chr> | proc_code <chr> |
|---|---|---|---|
| 64333 | 510909 | THYROID STIMULATING HORMONE | TSH |

1 row | 1-5 of 17 columns

```
filter(orders, pref_list_type == "Clinic Preference List" |
               pref_list_type == "Provider Preference List",
               !is.na(reason_for_canc_c))
```

| order_id <dbl> | patient_id <dbl> | description <chr> | proc_code <chr> |
|---|---|---|---|
| 19766 | 511388 | PROTHROMBIN TIME | PRO |
| 88444 | 511388 | BASIC METABOLIC PANEL | BMP |
| 34373 | 511303 | IRON, SERUM | FE |
| 50728 | 501184 | COMPREHENSIVE METABOLIC PANEL | COMP |

```
filter(orders, department %in% c("OB GYN CLINIC","GERIATRIC CLINIC","PEDIATRIC
                                                                  CLINICS"))
```

| order_id <dbl> | patient_id <dbl> | description <chr> | proc_code <chr> |
|---|---|---|---|
| 55347 | 510095 | URINE PREGNANCY TEST HCG, ONSITE | 81025 |
| 27773 | 511000 | URINE PREGNANCY TEST HCG, ONSITE | 81025 |
| 43511 | 511000 | PATHOLOGY, SURGICAL | SURG |
| 80696 | 501931 | WET MOUNTS INCL PREP VAGINAL CERV/SKIN SPECIMENS, ONSITE | Q0111 |
| 21481 | 501931 | R/O YEAST CULT W/DIRECT EXAM | YSTF |

# arrange()

# arrange()

Order rows from smallest to largest values

```
arrange(data, ... )
```

**data frame to transform**

**name(s) of columns to arrange by**

# arrange()

Order rows from smallest to largest values

```
arrange(orders, result_time)
```

| order_id | patient_id | description | result_time |
|---|---|---|---|
| 19766 | 511388 | PROTHROMBIN TIME | 2017-09-20 |
| 88444 | 511388 | BASIC METABOLIC PANEL | 2017-09-01 |
| 40477 | 508061 | THYROID STIMULATING HORMONE | 2017-09-28 |
| 97641 | 508061 | T4, FREE | 2017-09-04 |

| order_id | patient_id | description | result_time |
|---|---|---|---|
| 88444 | 511388 | BASIC METABOLIC PANEL | 2017-09-01 |
| 97641 | 508061 | T4, FREE | 2017-09-04 |
| 19766 | 511388 | PROTHROMBIN TIME | 2017-09-20 |
| 40477 | 508061 | THYROID STIMULATING HORMONE | 2017-09-28 |

# arrange()

Order rows from smallest to largest values

```
arrange(orders, desc(result_time))
```

| order_id | patient_id | description | result_time |
|---|---|---|---|
| 19766 | 511388 | PROTHROMBIN TIME | 2017-09-20 |
| 88444 | 511388 | BASIC METABOLIC PANEL | 2017-09-01 |
| 40477 | 508061 | THYROID STIMULATING HORMONE | 2017-09-28 |
| 97641 | 508061 | T4, FREE | 2017-09-04 |

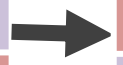
| order_id | patient_id | description | result_time |
|---|---|---|---|
| 40477 | 508061 | THYROID STIMULATING HORMONE | 2017-09-28 |
| 19766 | 511388 | PROTHROMBIN TIME | 2017-09-20 |
| 97641 | 508061 | T4, FREE | 2017-09-04 |
| 88444 | 511388 | BASIC METABOLIC PANEL | 2017-09-01 |

# arrange()

Order rows from smallest to largest values

```
arrange(orders, patient_id, result_time)
```

| order_id | patient_id | description | result_time |
|---|---|---|---|
| 19766 | 511388 | PROTHROMBIN TIME | 2017-09-20 |
| 88444 | 511388 | BASIC METABOLIC PANEL | 2017-09-01 |
| 40477 | 508061 | THYROID STIMULATING HORMONE | 2017-09-28 |
| 97641 | 508061 | T4, FREE | 2017-09-04 |



| order_id | patient_id | description | result_time |
|---|---|---|---|
| 97641 | 508061 | T4, FREE | 2017-09-04 |
| 40477 | 508061 | THYROID STIMULATING HORMONE | 2017-09-28 |
| 88444 | 511388 | BASIC METABOLIC PANEL | 2017-09-01 |
| 19766 | 511388 | PROTHROMBIN TIME | 2017-09-20 |

# Your Turn 5

- Arrange orders by **order_status_c_descr**. What order statuses appear in the top rows of the data frame?

- Add **order_class_c_descr** as a second (tie breaking) column to arrange on, but order it in *reverse* alphabetical order.

- CHALLENGE:

Explore what type of order class appears at the top of the data frame. To what type of lab test does this order class seem to relate to?

```
arrange(orders, order_status_c_descr)
```

| order_id <dbl> | patient_id <dbl> | description <chr> | proc_code <chr> | order_class_c_descr <chr> | lab_status_c <dbl> | lab_status_c_descr <chr> | order_status_c <dbl> | order_status_c_descr <chr> |
|---|---|---|---|---|---|---|---|---|
| 19766 | 511388 | PROTHROMBIN TIME | PRO | Normal | *NA* | *NA* | 4 | Canceled |
| 88444 | 511388 | BASIC METABOLIC PANEL | BMP | Normal | *NA* | *NA* | 4 | Canceled |
| 34373 | 511303 | IRON, SERUM | FE | Normal | 5 | Edited Result - FINAL | 4 | Canceled |
| 79887 | 510902 | CBC (HEMOGRAM) | CBC | Normal | *NA* | *NA* | 4 | Canceled |
| 50728 | 501184 | COMPREHENSIVE METABC | COMP | Normal | *NA* | *NA* | 4 | Canceled |
| 91635 | 501184 | CBC (HEMOGRAM) | CBC | Normal | *NA* | *NA* | 4 | Canceled |

```
arrange(orders, order_status_c_descr,
                desc(order_class_c_descr))
```

| order_id <dbl> | patient_id <dbl> | description <chr> | proc_code <chr> | order_class_c_descr <chr> |
|---|---|---|---|---|
| 33517 | 501790 | U/A NONAUTO DIPSTICK ONLY, ONSITE | 81002 | On Site |
| 90317 | 513135 | GLUCOSE, WHOLE BLOOD, ONSITE | 82962 | On Site |
| 31068 | 502323 | INFLUENZA ASSAY RAPID, ONSITE | 87804 | On Site |
| 78107 | 513364 | GLUCOSE, WHOLE BLOOD, ONSITE | 82962 | On Site |
| 39372 | 500653 | GLUCOSE, WHOLE BLOOD, ONSITE | 82962 | On Site |

%>%

# Data Analysis Steps

```
VITD <- filter(orders, description == "1,25 DIHYDROXY VITAMIN D")

VITD <- select(VITD, department, ordering_route, pref_list_type)

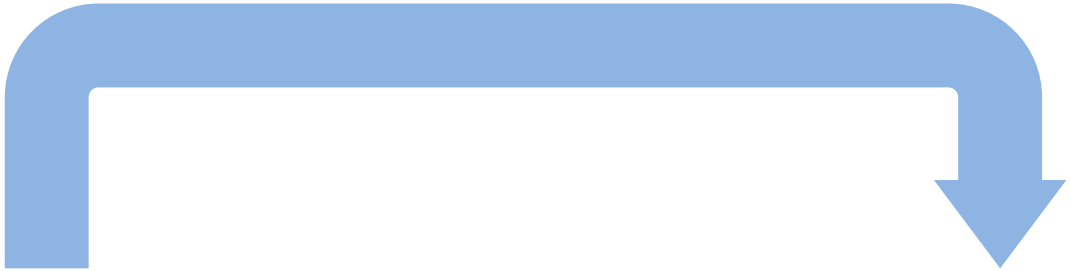
VITD <- arrange(VITD, department)
```

1. Filter orders to 1, 25 Vitamin D
2. Select the relevant columns that contain ordering information
3. Arrange those columns by department

# Data Analysis Steps

```
VITD <- arrange(
            select(
              filter(
                orders,
                description == "1,25 DIHYDROXY VITAMIN D"
              ),
              department,
              ordering_route,
              pref_list_type
            ),
            pref_list_type
          )
```

# The Pipe Operator %>%



```
orders  %>%  filter(____, patient_id == 508061)
```

Passes result on left into first argument of function on right.

```
filter(orders, patient_id == 508061)

orders %>% filter(patient_id == 508061)
```

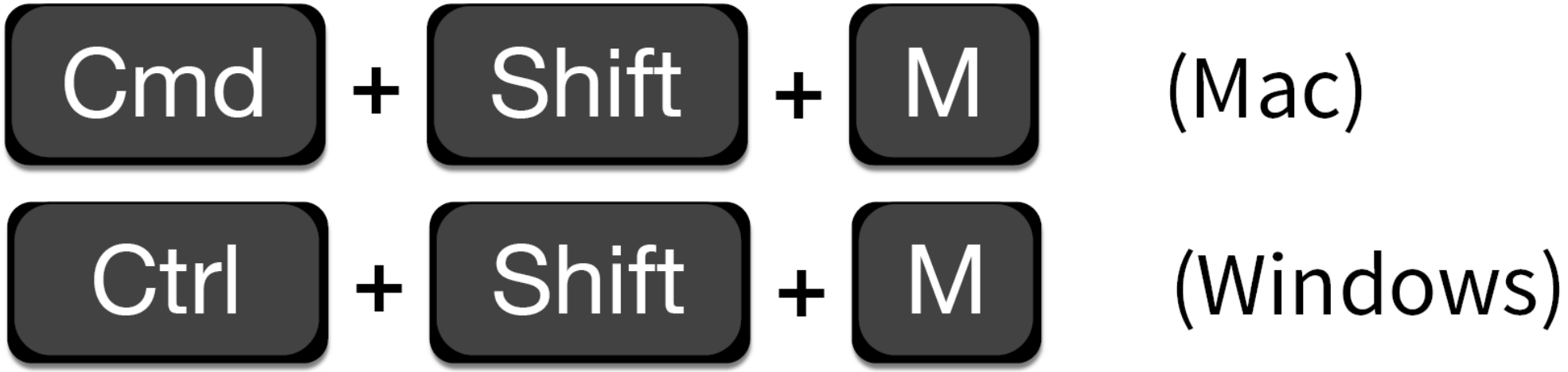
# Data Analysis Steps

```
125VITD <- arrange(
            select(
              filter(
                orders,
                description == "1,25 DIHYDROXY VITAMIN D"
              ),
              department,
              ordering_route,
              pref_list_type
            ),
            pref_list_type
          )
```

# Data Analysis Steps

```
VITD <- orders %>%
            filter(description == "1,25 DIHYDROXY VITAMIN D") %>%
            select(department, ordering_route, pref_list_type) %>%
            arrange(pref_list_type)
```

# Shortcut to type %>%



# Your Turn 6

Use %>% to write a sequence of three functions that:

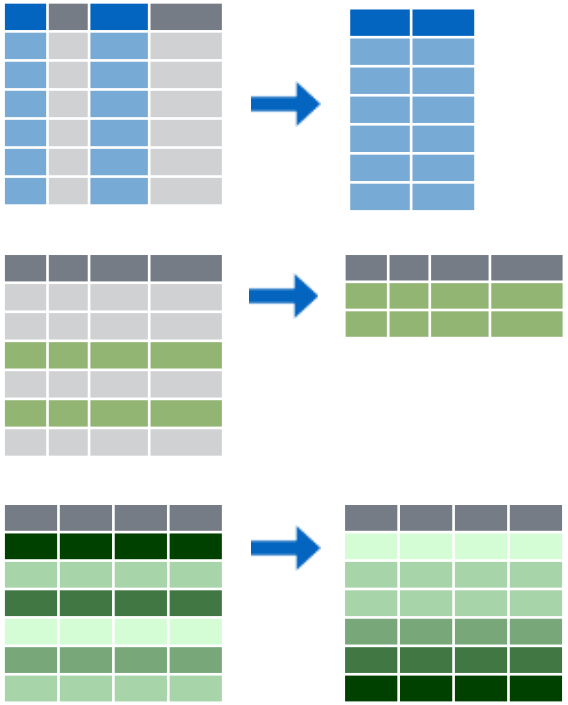
1. Filters to orders coming from the "BEHAVIORAL HEALTH CLINIC"
2. Selects the **description**, **ordering_route**, and **pref_list_type**
3. Arrange the dataset by the **description** and **ordering_route** columns

Using <-, assign the result to a new variable.

4. CHALLENGE- Use your mouse to select the name of the new data frame from the list of data sets in the upper-right pane of Rstudio. Do you notice any ordering patterns?

```
orders %>%
  filter(department =='BEHAVIORAL HEALTH CLINIC') %>%
  select(description,pref_list_type, ordering_route) %>%
  arrange(description,ordering_route)
```

# Isolating data



Extract variables with **select()**

Extract rows with **filter()**

Arrange rows, with **arrange()**.

# Deriving Data

# What are the top 3 tests ordered on weekends?

# Breaking down the analytical question

1. Day of the week for each order
2. Count of each test one weekend
3. Ranking of tests by count

# Deriving data

Make new variables with **mutate()**

Make summaries of data with **summarize()**

# mutate()

# mutate()

Creating new columns

# mutate()

Creating new columns

```
orders %>%
    mutate(new_column = calculation)
```

name for new column

equals

function whose results will populate columns

# mutate()

calculation can involve another column in the data frame

## Creating new columns

```
orders %>%
    mutate(coded_order_id = order_id/2)
```

| order_id | patient_id |
|---|---|
| 19766 | 511388 |
| 88444 | 511388 |
| 40477 | 508061 |
| 97641 | 508061 |

| order_id | patient_id | coded_order_id |
|---|---|---|
| 19766 | 511388 | 9883 |
| 88444 | 511388 | 44222 |
| 40477 | 508061 | 20238 |
| 97641 | 508061 | 48820 |

# Your Turn 7

The weekdays() function will return the weekday for any date.

1. Use the weekdays() function with mutate() to make a new column which contains the day of the week that each order was placed
2. Then select this column and the **order_time** column

```
orders %>%
     mutate(dayofweek = weekdays(order_time)) %>%
     select(dayofweek, order_time)
```

# mutate()

## Replacing columns

Function to "coerce" one type of data into another type of data

```
orders %>%
    mutate(order_id = as.character(order_id))
```

| order_id <chr> | patient_id <dbl> | description <chr> |
|---|---|---|
| 19766 | 511388 | PROTHROMBIN TIME |
| 88444 | 511388 | BASIC METABOLIC PANEL |
| 40477 | 508061 | THYROID STIMULATING HORMONE |
| 97641 | 508061 | T4, FREE |

# data types in R

| Type of Data | Function | Result of Function |
|---|---|---|
| Character | `as.character(1)` | `"1"` |
| Numeric | `as.numeric("123")` | `123` |
| Logical | `as.logical(1)` | `TRUE` |
| Date | `as.Date("05-06-2019")` | `NA` |

# mutate()

Conditionally replace values in a column

```
orders %>%
    mutate(proc_code =
           ifelse(proc_code %in% c("CBC","CBD"),
                  "CBC",
                  proc_code))
```

logical statement

value if true

value if false

# Your Turn: Final Challenge

Conditionally replace values in **pref_list_type:**

The pref_list_type column has values of either:

"Clinic Preference List", "Provider Preference List",  "None"

Conditionally replace these with :

"clinic", "provider", and NA, respectively.

```
orders %>%
   mutate(pref_list_type = ifelse(pref_list_type=="None",
          NA,
          ifelse(pref_list_type=="Clinic Preference List",
                                  "clinic",pref_list_type)))
```