

# Vector Functions

## TO USE WITH MUTATE ()

**mutate()** and **transmute()** apply vectorized functions to columns to create new columns. Vectorized functions take vectors as input and return vectors of the same length as output.

**vectorized function**

## OFFSETS

**dplyr::lag()** - Offset elements by 1  
**dplyr::lead()** - Offset elements by -1

## CUMULATIVE AGGREGATES

**dplyr::cumall()** - Cumulative all()  
**dplyr::cumany()** - Cumulative any()  
**cummax()** - Cumulative max()  
**dplyr::cummean()** - Cumulative mean()  
**cummin()** - Cumulative min()  
**cumprod()** - Cumulative prod()  
**cumsum()** - Cumulative sum()

## RANKINGS

**dplyr::cume\_dist()** - Proportion of all values <=   
**dplyr::dense\_rank()** - rank w ties = min, no gaps  
**dplyr::min\_rank()** - rank with ties = min  
**dplyr::ntile()** - bins into n bins  
**dplyr::percent\_rank()** - min\_rank scaled to [0,1]  
**dplyr::row\_number()** - rank with ties = "first"

## MATH

**+**, **-**, **\***, **/**, **^**, **%/%**, **%%** - arithmetic ops  
**log()**, **log2()**, **log10()** - logs  
**<**, **<=**, **>**, **>=**, **!=**, **==** - logical comparisons  
**dplyr::between()** - x >= left & x <= right  
**dplyr::near()** - safe == for floating point numbers

## MISC

**dplyr::case\_when()** - multi-case if\_else()  
*iris %>% mutate(Species = case\_when(  
Species == "versicolor" ~ "versi",  
Species == "virginica" ~ "virgi",  
TRUE ~ Species))*  
**dplyr::coalesce()** - first non-NA values by element across a set of vectors  
**dplyr::if\_else()** - element-wise if() + else()  
**dplyr::na\_if()** - replace specific values with NA  
**pmax()** - element-wise max()  
**pmin()** - element-wise min()  
**dplyr::recode()** - Vectorized switch()  
**dplyr::recode\_factor()** - Vectorized switch() for factors

# Summary Functions

## TO USE WITH SUMMARISE ()

**summarise()** applies summary functions to columns to create a new table. Summary functions take vectors as input and return single values as output.

**summary function**

## COUNTS

**dplyr::n()** - number of values/rows  
**dplyr::n\_distinct()** - # of uniques  
**sum(!is.na())** - # of non-NA's

## LOCATION

**mean()** - mean, also **mean(!is.na())**  
**median()** - median

## LOGICALS

**mean()** - Proportion of TRUE's  
**sum()** - # of TRUE's

## POSITION/ORDER

**dplyr::first()** - first value  
**dplyr::last()** - last value  
**dplyr::nth()** - value in nth location of vector

## RANK

**quantile()** - nth quantile  
**min()** - minimum value  
**max()** - maximum value

## SPREAD

**IQR()** - Inter-Quartile Range  
**mad()** - median absolute deviation  
**sd()** - standard deviation  
**var()** - variance

# Row Names

Tidy data does not use rownames, which store a variable outside of the columns. To work with the rownames, first move them into a column.

**rownames\_to\_column()**  
Move row names into col.  
*a <- rownames\_to\_column(iris, var = "C")*

**column\_to\_rownames()**  
Move col in row names.  
*column\_to\_rownames(a, var = "C")*

Also **has\_rownames()**, **remove\_rownames()**

# Combine Tables

## COMBINE VARIABLES

**x** + **y** = **A B C A B D**  

|   |   |   |  |  |
|---|---|---|--|--|
| a | t | 1 |  |  |
| b | u | 2 |  |  |
| c | v | 3 |  |  |

 + 

|   |   |   |  |  |
|---|---|---|--|--|
| a | t | 3 |  |  |
| b | u | 2 |  |  |
| d | w | 1 |  |  |

 = 

|   |   |   |   |   |   |
|---|---|---|---|---|---|
| a | t | 1 | a | t | 3 |
| b | u | 2 | b | u | 2 |
| c | v | 3 | d | w | 1 |

Use **bind\_cols()** to paste tables beside each other as they are.

**bind\_cols(...)** Returns tables placed side by side as a single table.  
BE SURE THAT ROWS ALIGN.

Use a "Mutating Join" to join one table to columns from another, matching values with the rows that they correspond to. Each join retains a different combination of values from the tables.

**left\_join(x, y, by = NULL, copy=FALSE, suffix=c(".x",".y"),...)**  
Join matching values from y to x.

**right\_join(x, y, by = NULL, copy = FALSE, suffix=c(".x",".y"),...)**  
Join matching values from x to y.

**inner\_join(x, y, by = NULL, copy = FALSE, suffix=c(".x",".y"),...)**  
Join data. Retain only rows with matches.

**full\_join(x, y, by = NULL, copy=FALSE, suffix=c(".x",".y"),...)**  
Join data. Retain all values, all rows.

Use **by = c("col1", "col2", ...)** to specify one or more common columns to match on.  
*left\_join(x, y, by = "A")*

Use a named vector, **by = c("col1" = "col2")**, to match on columns that have different names in each table.  
*left\_join(x, y, by = c("C" = "D"))*

Use **suffix** to specify the suffix to give to unmatched columns that have the same name in both tables.  
*left\_join(x, y, by = c("C" = "D"), suffix = c("1", "2"))*

## COMBINE CASES

**x** + **y** = **A B C**  

|   |   |   |
|---|---|---|
| a | t | 1 |
| b | u | 2 |
| c | v | 3 |

 + 

|   |   |   |
|---|---|---|
| a | t | 3 |
| c | v | 3 |
| d | w | 4 |

Use **bind\_rows()** to paste tables below each other as they are.

**bind\_rows(..., .id = NULL)**  
Returns tables one on top of the other as a single table. Set .id to a column name to add a column of the original table names (as pictured)

**intersect(x, y, ...)**  
Rows that appear in both x and y.

**setdiff(x, y, ...)**  
Rows that appear in x but not y.

**union(x, y, ...)**  
Rows that appear in x or y. (Duplicates removed). **union\_all()** retains duplicates.

Use **setequal()** to test whether two data sets contain the exact same rows (in any order).

## EXTRACT ROWS

**x** + **y** = **A B C**  

|   |   |   |
|---|---|---|
| a | t | 1 |
| b | u | 2 |
| c | v | 3 |

 + 

|   |   |   |
|---|---|---|
| a | t | 3 |
| b | u | 2 |
| d | w | 1 |

Use a "Filtering Join" to filter one table against the rows of another.

**semi\_join(x, y, by = NULL, ...)**  
Return rows of x that have a match in y. USEFUL TO SEE WHAT WILL BE JOINED.

**anti\_join(x, y, by = NULL, ...)**  
Return rows of x that do not have a match in y. USEFUL TO SEE WHAT WILL NOT BE JOINED.

