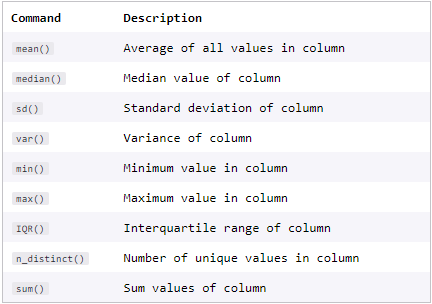
Data

Aggregates:

df %>% group\_by(column\_1) %>% aggregate

**group\_by()** = groups data by a column

**summarize()** = reduces multiple values to a single value

* name\_of\_new\_column\_result = command(column\_name\_of\_df)
* 
* n() Count of the rows within a group, doesn’t require a column argument
  + *\*add na.rm = TRUE to commands*

**filter(**df, filter**)** = filter rows of df based on per-group metrics

* column\_name == values\_you\_want
* is.na(column) --- to only include rows where column has NA values

**mutate()** = add new columns to df that involve per-group metrics

* new\_column\_name =

**select(**df, column**)** = isolate 1 column from df

**pull(**df, column**)** = isolate 1 column from df as a vector (to prepare it for statistics)

Joining Tables = Creating a df by matching common columns of dfs

**inner\_join()** = only joins matching rows between df’s… drops unmatched rows between dfs

*looks for columns that are common between dfs 🡪 finds rows where those columns’ values are   
the same 🡪 combines the matching rows into a single row in a new table*

inner\_join(df1, df2, by = c(‘\_’ = ‘\_’), suffix = c(‘\_’ = ‘\_’))

* 2 df: df\_1 %>% inner\_join(df\_2) = matches up all df\_2 information to the df\_1 information
* 2+ df: df\_1 %>% inner\_join(df\_2) %>% inner\_join(df\_3) … = matches up info between all dfs
* **by =** c(‘column in df1’ = ‘column in df2’) --- to specify which columns to join between df’s
* **suffix =** c(‘\_suffix1’, ‘\_suffix2’) --- to add

Note: Can rename column names before joining to make sure it is joining the correct columns

* df %>% **rename**(new\_column\_name = old\_column\_name)
* inner join(df\_1, df\_2)

**full\_join(**df1, df2**)** = combines data from both dfs (*all rows from both tables*)

**left\_join(**df1, df2**)** = includes all rows from df1 (left), but only rows from df2 (right) that match df1

**right\_join(**df1, df2**)** = includes all rows from df2 (right), but only rows from df1 (left) that match df2

**bind\_rows(**df1, df2**)** = to reconstruct a single df from multiple smaller dfs w/ the same columns

* \*only if all columns are the same in all of the dfs

Data Frames:

**head ( df, #)** = shows the first # (6 default) rows for all columns of the df

**print()** =

**colnames(**df**)** = returns (or sets) the names of the columns in the df

* To set new column names: colnames(df) <- c() vector containing column names
  + vector's length must = # columns in the original df

**ncol(**df**)** = returns the # of columns in the df

**nrow(**df**)** = returns the # of rows in the df

**subset(**df, expression**)** = returns a subset of a df that matches the specified conditions

* df = df to retrieve the subset from
* expression = specifies the subset conditions (ex. cost>10)

**df$column** = to get a single column from a df

Descriptive Statistics:

Mean, Median, Mode – using example data = c(n1, n2, n3, n4)

**mean(**data**)** = Total / # observations

**median(**data**)** = Middle value

* 1. Order the values in the dataset from smallest to largest
* 2. Identify the number(s) that fall(s) in the middle

**Mode(**data**)** = Most common value \*\*capitalized “M”

* 1. Find the frequency of each value
* 2. Determine the value with the largest frequency

**variance(**data**)** = Measure of spread

* + difference = (datapoint Xi – mean µ) ^ 2

**sd(**data**)** = standard deviation = variance ^ 0.5

**quantile(**data, n**)** = find the quartile

* n = number between 0 and 1
  + n = 0.25 (first), 0.5 (second), 0.75 (third)
  + n = c(n1, n2, n3, …)
    - Q1 = median of data lower than Q2 (either include or exclude Q2)
    - Q2 = median
    - Q3 = median of data higher than Q2 (either include or exclude Q2)
* Quartiles = data split into 4 groups of equal size (.25, .5, .75)
* Quintiles = data split into 5 groups of equal size (.2, .4, .6, .8)
* Deciles = data split into 10 groups of equal size (.1, .2, .3, .4, .5, … .9)
* Percentiles = data split into 100 groups of equal size (.01, .02, … .99)

**max(**data**)** = maximum

**min(**data**)** = minimum

**range(**data**)** = max - min

**IQR(**data**)** = Q3 – Q1

* interquartile range = range that ignores the tails of the dataset (ignores outliers on tails)

Hypothesis Testing:

**mean()** – differs based on sample or population

**intersect(**vector1, vector2**)** = takes 2 vectors as arguments and returns a vector containing common elements between them

Type I Error = false positive; occurs when a hypothesis test finds a correlation between things that are not related; null hypothesis is rejected even though it is true

Type II Error = false negative; failing to find a correlation between things that are actually related; null hypothesis is accepted even though it is false

**t.test(**sample\_distribution, mu = expected\_mean**)** – *one sample t-test*

* inputs:
  + sample\_distribution = sample of values collected
  + mu = argument indicating the desired mean of the hypothetical population
  + expected\_mean = value of the desired mean
* One sample t-test: compares a sample mean to a hypothetical population mean
  + p-value = probability the sample of values came from a distribution with the specified mean

**t.test(**distribution1, distribution2**)** – *two sample t-test*

* Two Sample t-test: compares two sets of data, which are both approximately normally distributed
  + p-value = probability that the difference in the means happened by chance (sampling error; prob that the null hypothesis is true)
  + probability error = 1 – (0.95\*\*number\_of\_t-tests)

**aov(**v1 ~ v2 \* v3 + v4, data = df**)** - *anova*

* inputs:
  + v2(\*v3) 🡪 v1
  + +v4 controls for v4
* ANOVA: tests the null hypothesis that all of the datasets have the same mean
  + p-value = probability that at least 1 of the datasets has a different mean (but don’t know which one)
  + \*use **summary()** to look at the p-value of the test