Assignment

Method 1	`new_df <- colleges %>% filter(State=="IA")`	Store the filtered dataset in new_df.
Method 2	`new_df = colleges %>% filter(State=="IA")`	Store the filtered dataset in new_df.
No assignment	`colleges %>% filter(State=="IA")`	display the filtered dataset, but it is not stored

Without assignment, you will not be able to use the new dataframe you created later.

dplyr

filter()

Pick rows that meets the conditions

Format	`dataframe_name %>%`
	`   filter(Condition1, Condition2, …)`

Example	`colleges %>%`
	`   filter(State=="IA", ACT_median > 25)`
	`      # Column name    Column name`

arrange()

Reorder the rows

	Ascending Order	**Descending Order**

Format	`dataframe_name %>%`
	`   arrange(column_name)`
		`dataframe_name%>%`
		`  arrange(desc(column_name))`

Example	`colleges %>%`
	`  arrange(ACT_median)`
	# Rows with smaller ACT_median goes in front
		`colleges %>%`
		`  arrange(desc(ACT_median))`
		# Rows with bigger ACT_median goes in front

`select()`

pick columns by their names

	Select columns you **want** to display	Select columns you **don't want** to display
Format	`dataframe_name %>%`	`dataframe_name %>%`
	`  select(column_name, column_name,`	`  select(-column_name, -column_name, …)`
	`…)`	
Example	`colleges %>%`	`colleges %>%`
	`  select(Name, ACT_median, Cost)`	`  select(-State, -City)`
		#All columns except "State" and "City"
	`Output:`	
	`##    Name     ACT_median   Cost`	`Output:`
	`## 1 Cornell     27        55817`	`##    Name     Enrollment Private Region …`
	`## 2 Drake       27        53507`	`## 1 Cornell   1022      Private Plains …`
	`## 3 Grinnell    32        65814`	`## 2 Drake     2952      Private Plains …`
	`## 4 Luther      26        54045`	`## 3 Grinnell  1683      Private Plains …`
	`...`	`## 4 Luther    1974      Private Plains …`
		`## 5 UIowa     23410     Public  Plains …`
		`...`
		# This is just a portion of the output

`mutate()`

Add new derived columns to a data frame

Format	`dataframe_name%>%`
	`  mutate(new_column_name = operations on existing columns)`
Example	`colleges %>%`
	`  mutate(Expected_Discount = (Cost_Net_Tuition) / Cost) %>%`
	# New Column Name Existing column names
	`  select(Name, Cost, Net_Tuition, Expected_Discount)`
	# For each row, "Cost_Net_Tuition" in that row is divided by "Cost" in the same row
	# The result is stored in the new column called "Expected_Discont"
	Output:

```
##   Name      Cost   Net_Tuition  Expected_Discount
## 1 Cornell   55817    16457          0.7051615
## 2 Drake     53507    21160          0.6045377
## 3 Grinnell  65814    20369          0.6905066
## 4 Luther    54045    16779          0.6895365
## 5 UIowa     22607    14547          0.3565267
```

summarize()

Aggregate many rows into a summary measure

Format

```
dataframe_name %>%
  summarize(new_column_name = function(existing_column_name))
```

Example

```
colleges %>%
  summarize(min_Cost = min(Cost),
            Ten_Cost = quantile(Cost, 0.1),
            median_Cost = median(Cost),
            Ninety_Cost = quantile(Cost, 0.9),
            max_Cost = max(Cost)
            mean_Cost = mean(Cost))
```

New column name   function   Existing columns name

```
# Find information(mean, median…) about all the data in a column
# Store it in the new column created
```

Output:

```
##   minCost Ten_Cost  medianCost  Ninety_Cost  maxCost
## 1  20476   22368.2    43520       54256.2     65814
```

group_by()

Internally add grouping tags to the rows of your data.
You will not see these tags, but R can see them and use them.

Format

```
dataframe_name %>%
  group_by(existing_column_name) %>%
  # The selected column should be categorical data.
  summarize(new_column_name = function(existing_column_name))
  mutate(new_column_name = operations on existing_columns)
```

Example

```
colleges %>%
```

	`group_by(State) %>%`
	`# Column Name`
	`summarize(Median_Cost = median(Cost))`
	# For each category(i.e. IA, KS, MN) in the State, find the median cost
	
	`Output:`
	`##   State  Median_Cost`
	`## 1  IA      43520`
	`## 2  KS      38832`
	`## 3  MN      35887`
	`## 4  MO      30279`
	`## 5  ND      19299`
	`## 6  NE      29258`
	`## 7  SD      22609`
	
	Note: `group_by()` must be followed by `summarize()` or `mutate()`, otherwise it does nothing