Sane's STATS Academy of Statistics

Tidyverse

Collection of R packages

What is *tidyverse* ?

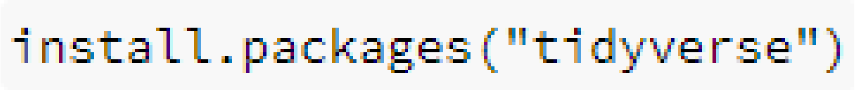
- The tidyverse is a collection of R packages designed for data science.
- All packages share an underlying design philosophy, grammar, and data structures.
- Developed by Hadley Wickham

Components of *tidyverse*

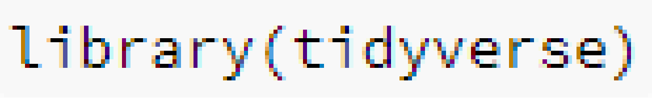
- **ggplot2:** ggplot2 is a system for declaratively creating graphics, based on The Grammar of Graphics.
- **dplyr:** dplyr provides a grammar of data manipulation, providing a consistent set of verbs that solve the most common data manipulation challenges
- **tidyr:** tidyr provides a set of functions that help you get to tidy data.
- readr: readr provides a fast and friendly way to read rectangular data (like csv, tsv, and fwf).
- purrr: purrr enhances R's functional programming (FP) toolkit by providing a complete and consistent set of tools for working with functions and vectors.
- **tibble:** tibble is a modern re-imagining of the data frame, keeping what time has proven to be effective, and throwing out what it has not.
- stringr: stringr provides a cohesive set of functions designed to make working with strings as easy as possible
- forcats: forcats provides a suite of useful tools that solve common problems with factors

Loading *tidyverse*

- All packages in tidyverse can be installed and loaded at one go



```r
install.packages("tidyverse")
```



```r
library(tidyverse)
```

Package *dplyr*

Handling the Data Efficiently

class tbl_df

- We can create an object of class tbl_df. It can be created by function as_tibble

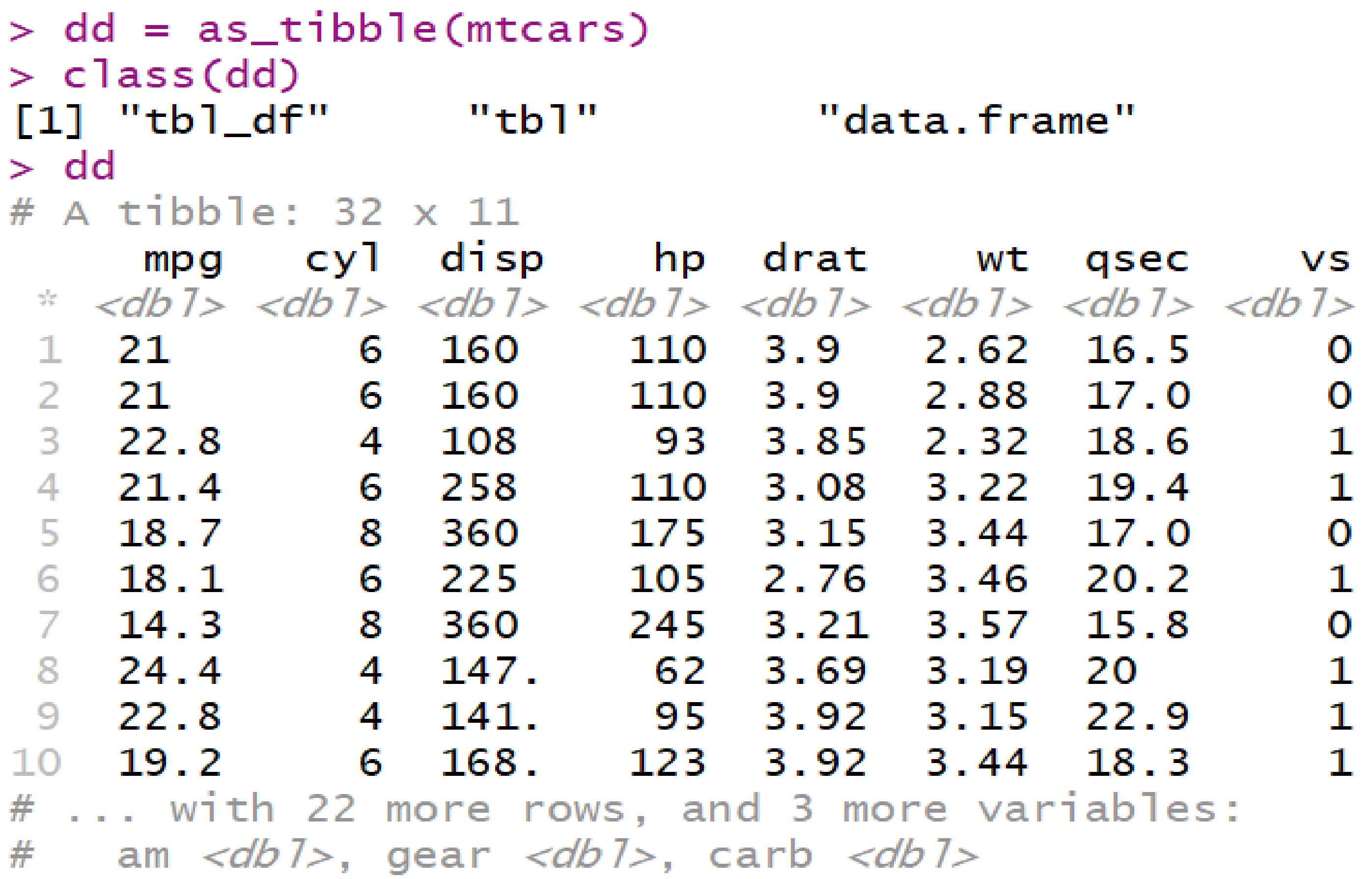
Syntax : as_tibble(objDF)

where

objDF: An object of class data.frame, or a list with each element with same length

tibble Object

- A tibble is a modern class of data frame within R
- It has a convenient print method, will not convert strings to factors, and does not use row names



```
> dd = as_tibble(mtcars)
> class(dd)
[1] "tbl_df"     "tbl"        "data.frame"
> dd
# A tibble: 32 x 11
     mpg   cyl  disp    hp  drat    wt  qsec    vs
 * <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
 1  21       6  160    110  3.9   2.62  16.5     0
 2  21       6  160    110  3.9   2.88  17.0     0
 3  22.8     4  108     93  3.85  2.32  18.6     1
 4  21.4     6  258    110  3.08  3.22  19.4     1
 5  18.7     8  360    175  3.15  3.44  17.0     0
 6  18.1     6  225    105  2.76  3.46  20.2     1
 7  14.3     8  360    245  3.21  3.57  15.8     0
 8  24.4     4  147.    62  3.69  3.19  20       1
 9  22.8     4  141.    95  3.92  3.15  22.9     1
10  19.2     6  168.   123  3.92  3.44  18.3     1
# ... with 22 more rows, and 3 more variables:
#   am <dbl>, gear <dbl>, carb <dbl>
```

Functions in **dplyr**

- **arrange**: reordering rows in the data frame
- **select**: selecting columns / variables
- **filter**: selecting rows / observations
- **rename**: renaming variables
- **mutate**: adding new columns to the data frame
- **summarize** / **summarise**: generating summary statistics of the data frames

Arranging the rows

- The rows in tbl_df object can be arranged using function arrange

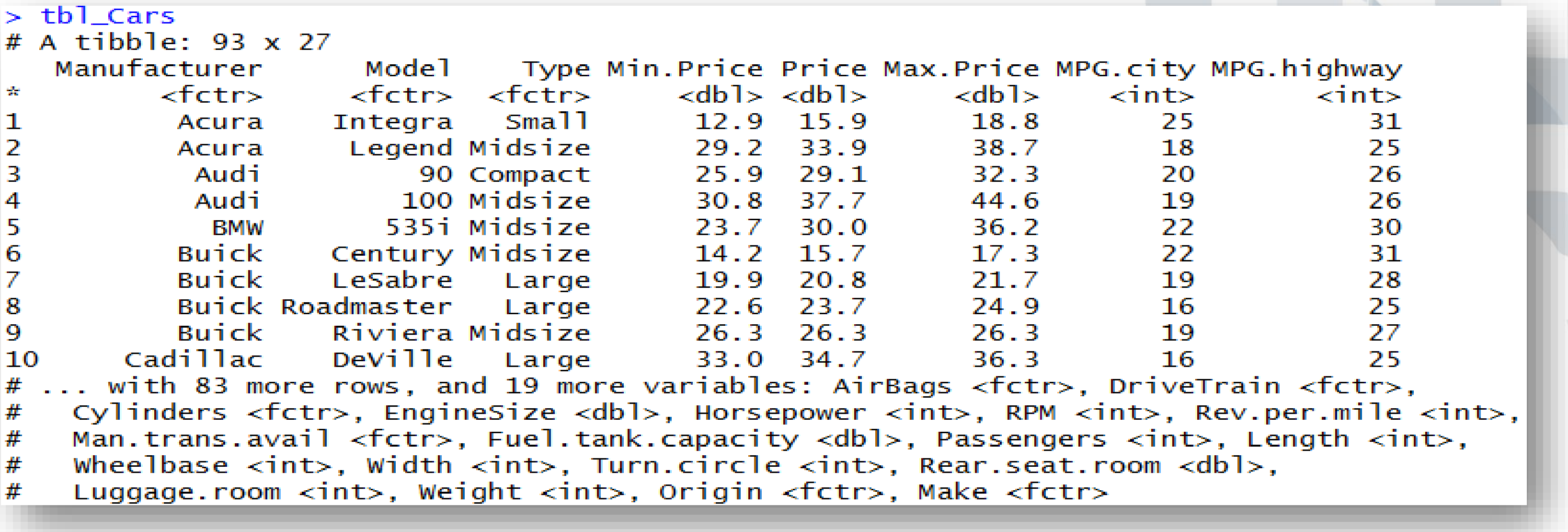
Syntax : arrange(Obj_tbl_df, col1,col2,…)

Where

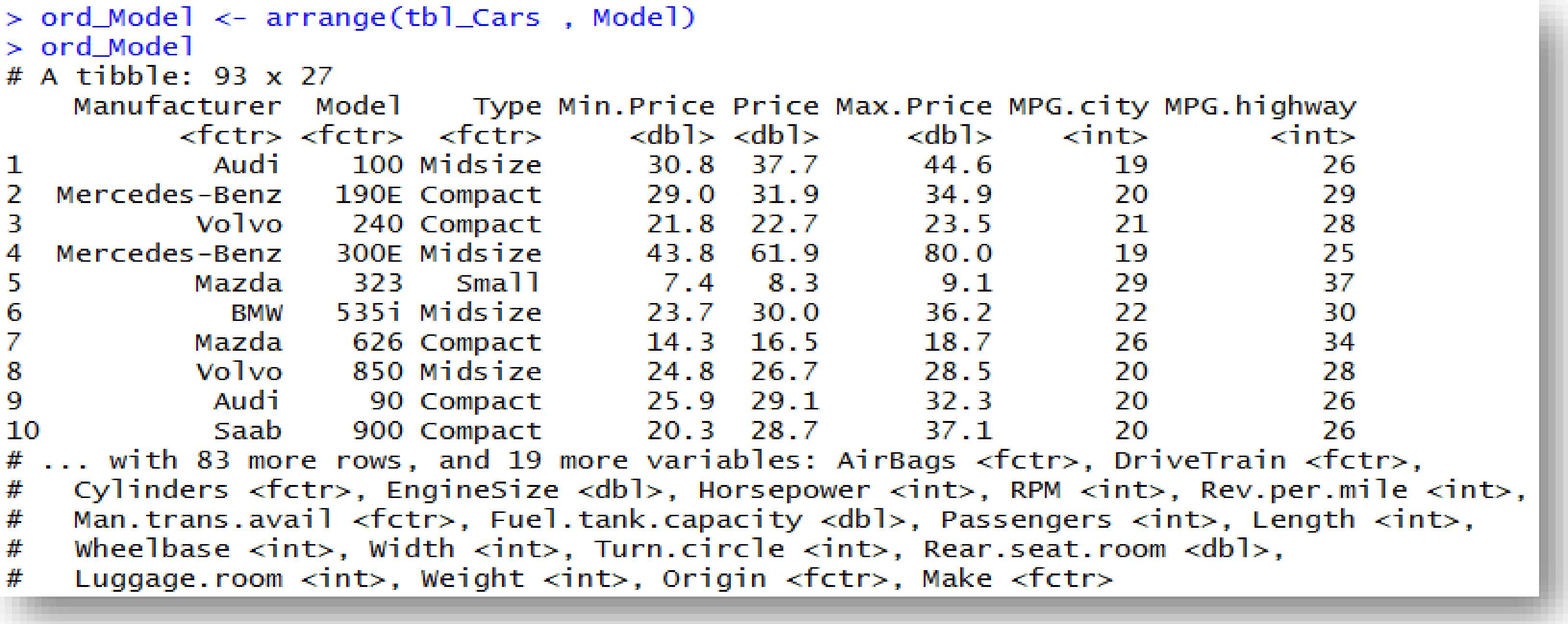
Obj_tbl_df : tbl_df object

col1, col2,… : Columns for sorting the data

Arrange Example

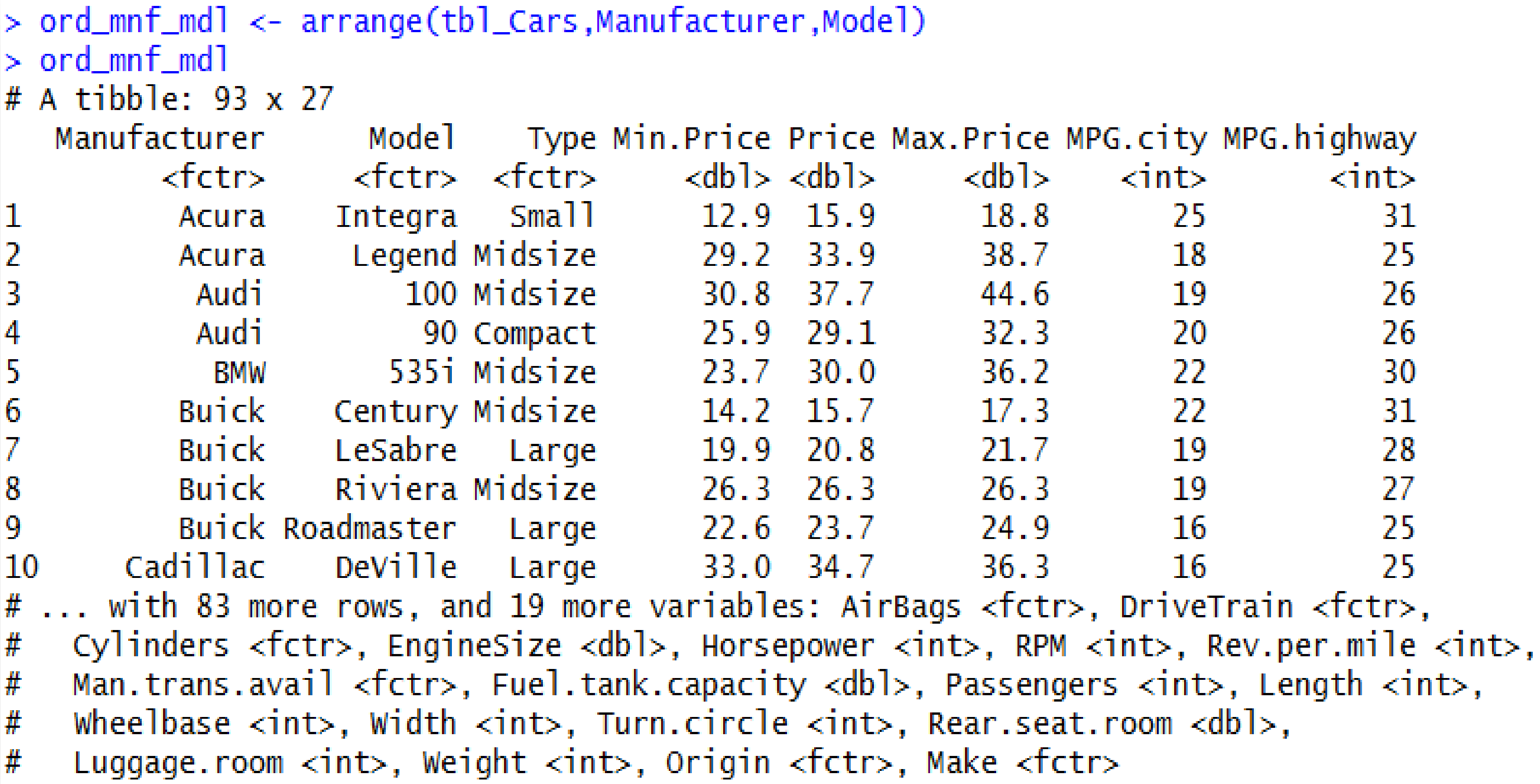


```
> tbl_Cars
# A tibble: 93 x 27
   Manufacturer      Model    Type Min.Price Price Max.Price MPG.city MPG.highway
*        <fctr>     <fctr>  <fctr>     <dbl> <dbl>     <dbl>    <int>       <int>
1         Acura    Integra   Small      12.9  15.9      18.8       25          31
2         Acura     Legend Midsize      29.2  33.9      38.7       18          25
3          Audi         90 Compact      25.9  29.1      32.3       20          26
4          Audi        100 Midsize      30.8  37.7      44.6       19          26
5           BMW       535i Midsize      23.7  30.0      36.2       22          30
6         Buick    Century Midsize      14.2  15.7      17.3       22          31
7         Buick    LeSabre   Large      19.9  20.8      21.7       19          28
8         Buick Roadmaster   Large      22.6  23.7      24.9       16          25
9         Buick    Riviera Midsize      26.3  26.3      26.3       19          27
10     Cadillac    DeVille   Large      33.0  34.7      36.3       16          25
# ... with 83 more rows, and 19 more variables: AirBags <fctr>, DriveTrain <fctr>,
#   Cylinders <fctr>, EngineSize <dbl>, Horsepower <int>, RPM <int>, Rev.per.mile <int>,
#   Man.trans.avail <fctr>, Fuel.tank.capacity <dbl>, Passengers <int>, Length <int>,
#   Wheelbase <int>, Width <int>, Turn.circle <int>, Rear.seat.room <dbl>,
#   Luggage.room <int>, Weight <int>, Origin <fctr>, Make <fctr>
```



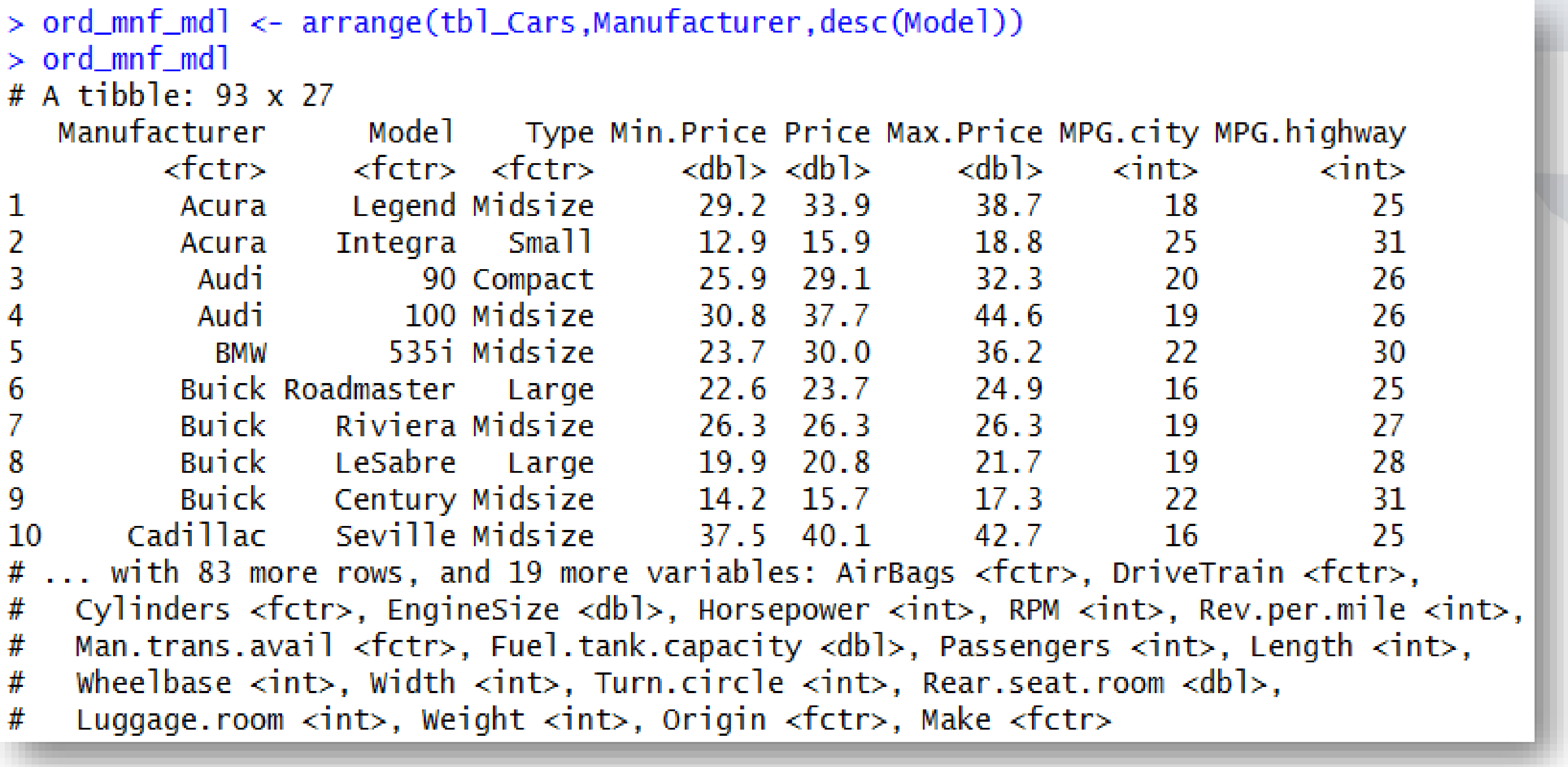
```
> ord_Model <- arrange(tbl_Cars , Model)
> ord_Model
# A tibble: 93 x 27
    Manufacturer  Model    Type Min.Price Price Max.Price MPG.city MPG.highway
          <fctr> <fctr>  <fctr>     <dbl> <dbl>     <dbl>    <int>       <int>
1           Audi    100 Midsize      30.8  37.7      44.6       19          26
2  Mercedes-Benz   190E Compact      29.0  31.9      34.9       20          29
3          Volvo    240 Compact      21.8  22.7      23.5       21          28
4  Mercedes-Benz   300E Midsize      43.8  61.9      80.0       19          25
5          Mazda    323   Small       7.4   8.3       9.1       29          37
6            BMW   535i Midsize      23.7  30.0      36.2       22          30
7          Mazda    626 Compact      14.3  16.5      18.7       26          34
8          Volvo    850 Midsize      24.8  26.7      28.5       20          28
9           Audi     90 Compact      25.9  29.1      32.3       20          26
10          Saab    900 Compact      20.3  28.7      37.1       20          26
# ... with 83 more rows, and 19 more variables: AirBags <fctr>, DriveTrain <fctr>,
#   Cylinders <fctr>, EngineSize <dbl>, Horsepower <int>, RPM <int>, Rev.per.mile <int>,
#   Man.trans.avail <fctr>, Fuel.tank.capacity <dbl>, Passengers <int>, Length <int>,
#   Wheelbase <int>, Width <int>, Turn.circle <int>, Rear.seat.room <dbl>,
#   Luggage.room <int>, Weight <int>, Origin <fctr>, Make <fctr>
```

Arranging Multiple Columns Example



```
> ord_mnf_mdl <- arrange(tbl_Cars,Manufacturer,Model)
> ord_mnf_mdl
# A tibble: 93 x 27
   Manufacturer      Model    Type Min.Price Price Max.Price MPG.city MPG.highway
         <fctr>     <fctr>  <fctr>     <dbl> <dbl>     <dbl>    <int>       <int>
1         Acura    Integra   Small      12.9  15.9      18.8       25          31
2         Acura     Legend Midsize      29.2  33.9      38.7       18          25
3          Audi        100 Midsize      30.8  37.7      44.6       19          26
4          Audi         90 Compact      25.9  29.1      32.3       20          26
5           BMW       535i Midsize      23.7  30.0      36.2       22          30
6         Buick    Century Midsize      14.2  15.7      17.3       22          31
7         Buick    LeSabre   Large      19.9  20.8      21.7       19          28
8         Buick    Riviera Midsize      26.3  26.3      26.3       19          27
9         Buick Roadmaster   Large      22.6  23.7      24.9       16          25
10     Cadillac    DeVille   Large      33.0  34.7      36.3       16          25
# ... with 83 more rows, and 19 more variables: AirBags <fctr>, DriveTrain <fctr>,
#   Cylinders <fctr>, EngineSize <dbl>, Horsepower <int>, RPM <int>, Rev.per.mile <int>,
#   Man.trans.avail <fctr>, Fuel.tank.capacity <dbl>, Passengers <int>, Length <int>,
#   Wheelbase <int>, Width <int>, Turn.circle <int>, Rear.seat.room <dbl>,
#   Luggage.room <int>, Weight <int>, Origin <fctr>, Make <fctr>
```

Arranging Multiple Columns Example



```
> ord_mnf_mdl <- arrange(tbl_Cars,Manufacturer,desc(Model))
> ord_mnf_mdl
# A tibble: 93 x 27
   Manufacturer      Model    Type Min.Price Price Max.Price MPG.city MPG.highway
         <fctr>     <fctr>  <fctr>     <dbl> <dbl>     <dbl>    <int>       <int>
1         Acura     Legend Midsize      29.2  33.9      38.7       18          25
2         Acura    Integra   Small      12.9  15.9      18.8       25          31
3          Audi         90 Compact      25.9  29.1      32.3       20          26
4          Audi        100 Midsize      30.8  37.7      44.6       19          26
5           BMW       535i Midsize      23.7  30.0      36.2       22          30
6         Buick Roadmaster   Large      22.6  23.7      24.9       16          25
7         Buick    Riviera Midsize      26.3  26.3      26.3       19          27
8         Buick    LeSabre   Large      19.9  20.8      21.7       19          28
9         Buick    Century Midsize      14.2  15.7      17.3       22          31
10     Cadillac    Seville Midsize      37.5  40.1      42.7       16          25
# ... with 83 more rows, and 19 more variables: AirBags <fctr>, DriveTrain <fctr>,
#   Cylinders <fctr>, EngineSize <dbl>, Horsepower <int>, RPM <int>, Rev.per.mile <int>,
#   Man.trans.avail <fctr>, Fuel.tank.capacity <dbl>, Passengers <int>, Length <int>,
#   Wheelbase <int>, Width <int>, Turn.circle <int>, Rear.seat.room <dbl>,
#   Luggage.room <int>, Weight <int>, Origin <fctr>, Make <fctr>
```

Selecting Columns

- For selecting specific columns, we can use select function

Syntax : select(objtbl , col 1, col 2, ...)

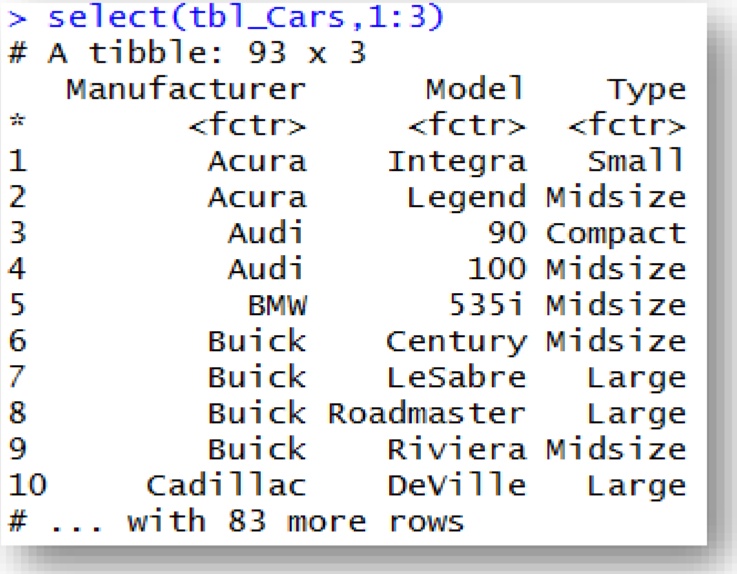
OR

select(objtbl , col 1:col n)

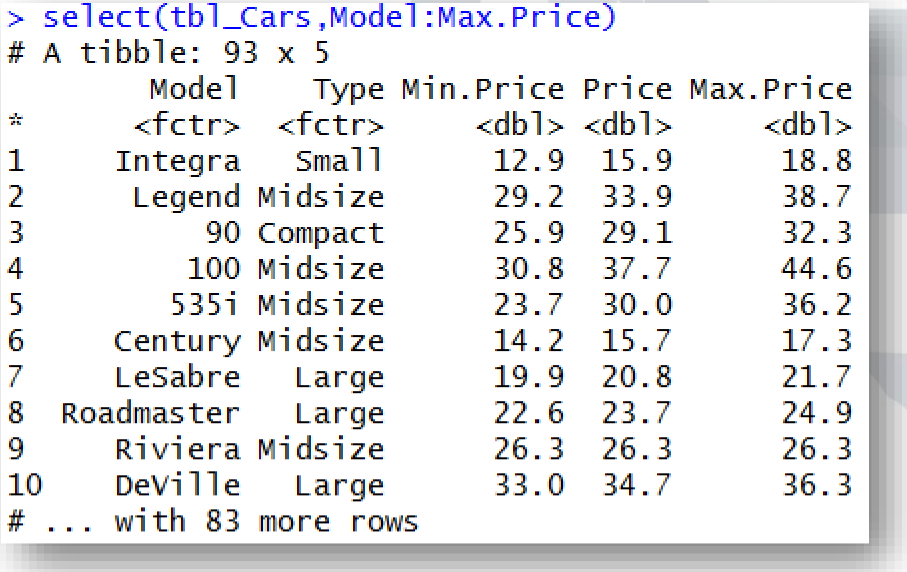
where

objtbl: Object of class tbl_df

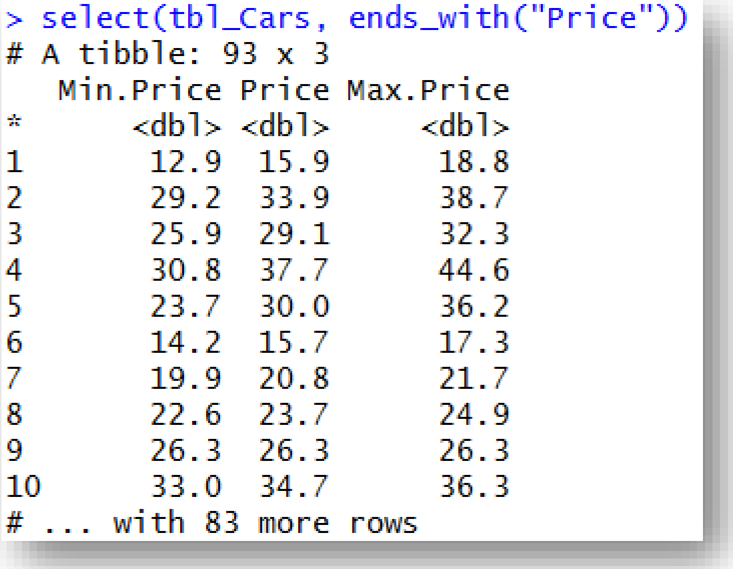
Select Examples



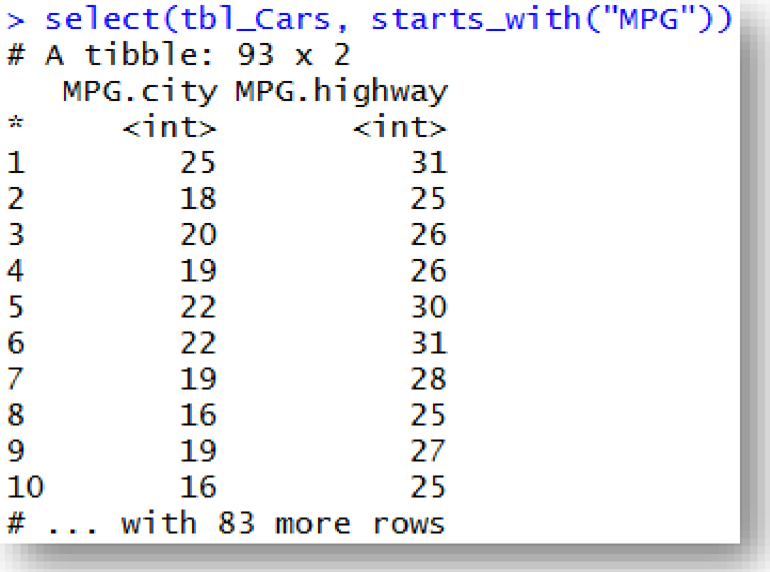
```
> select(tbl_Cars,1:3)
# A tibble: 93 x 3
   Manufacturer      Model    Type
*        <fctr>     <fctr>  <fctr>
1         Acura    Integra   Small
2         Acura     Legend Midsize
3          Audi         90 Compact
4          Audi        100 Midsize
5           BMW       535i Midsize
6         Buick    Century Midsize
7         Buick    LeSabre   Large
8         Buick Roadmaster   Large
9         Buick    Riviera Midsize
10     Cadillac    DeVille   Large
# ... with 83 more rows
```



```
> select(tbl_Cars,Model:Max.Price)
# A tibble: 93 x 5
       Model    Type Min.Price Price Max.Price
*     <fctr>  <fctr>     <dbl> <dbl>     <dbl>
1    Integra   Small      12.9  15.9      18.8
2     Legend Midsize      29.2  33.9      38.7
3         90 Compact      25.9  29.1      32.3
4        100 Midsize      30.8  37.7      44.6
5       535i Midsize      23.7  30.0      36.2
6    Century Midsize      14.2  15.7      17.3
7    LeSabre   Large      19.9  20.8      21.7
8 Roadmaster   Large      22.6  23.7      24.9
9    Riviera Midsize      26.3  26.3      26.3
10   DeVille   Large      33.0  34.7      36.3
# ... with 83 more rows
```



```
> select(tbl_Cars, ends_with("Price"))
# A tibble: 93 x 3
  Min.Price Price Max.Price
*     <dbl> <dbl>     <dbl>
1      12.9  15.9      18.8
2      29.2  33.9      38.7
3      25.9  29.1      32.3
4      30.8  37.7      44.6
5      23.7  30.0      36.2
6      14.2  15.7      17.3
7      19.9  20.8      21.7
8      22.6  23.7      24.9
9      26.3  26.3      26.3
10     33.0  34.7      36.3
# ... with 83 more rows
```



```
> select(tbl_Cars, starts_with("MPG"))
# A tibble: 93 x 2
  MPG.city MPG.highway
*    <int>       <int>
1       25          31
2       18          25
3       20          26
4       19          26
5       22          30
6       22          31
7       19          28
8       16          25
9       19          27
10      16          25
# ... with 83 more rows
```

Subsetting the data

- The data can be subsetted with function filter

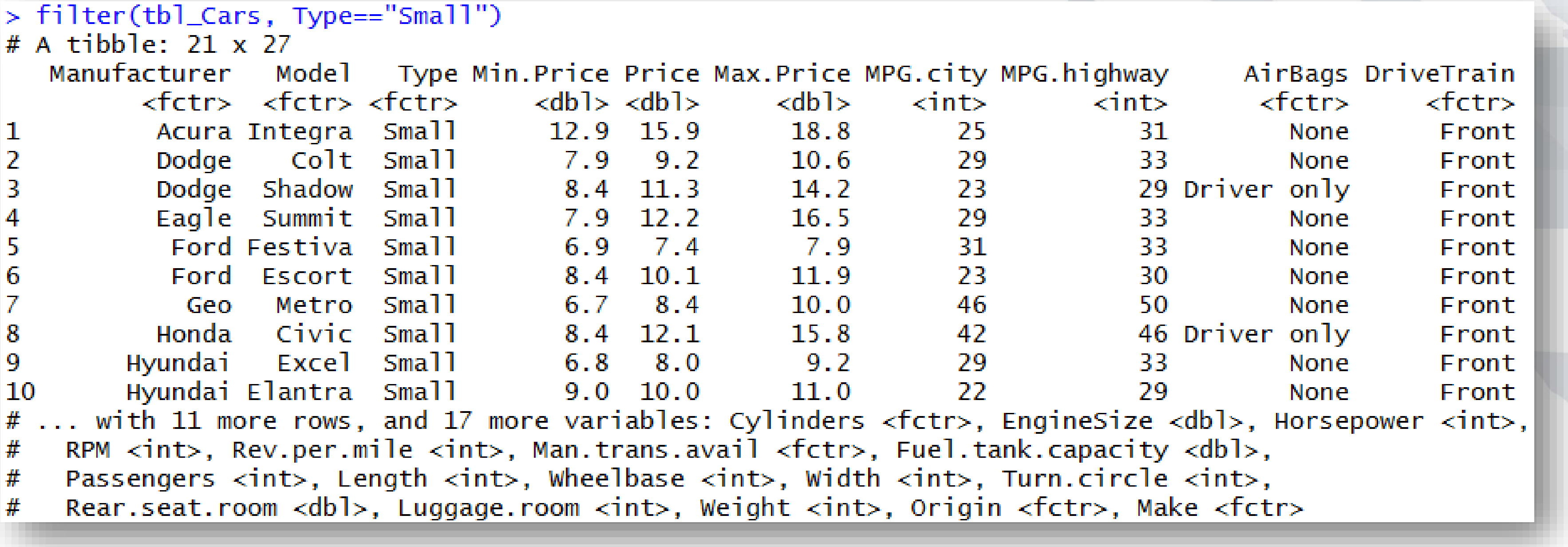
Syntax : filter(objtbl , criteria)

where

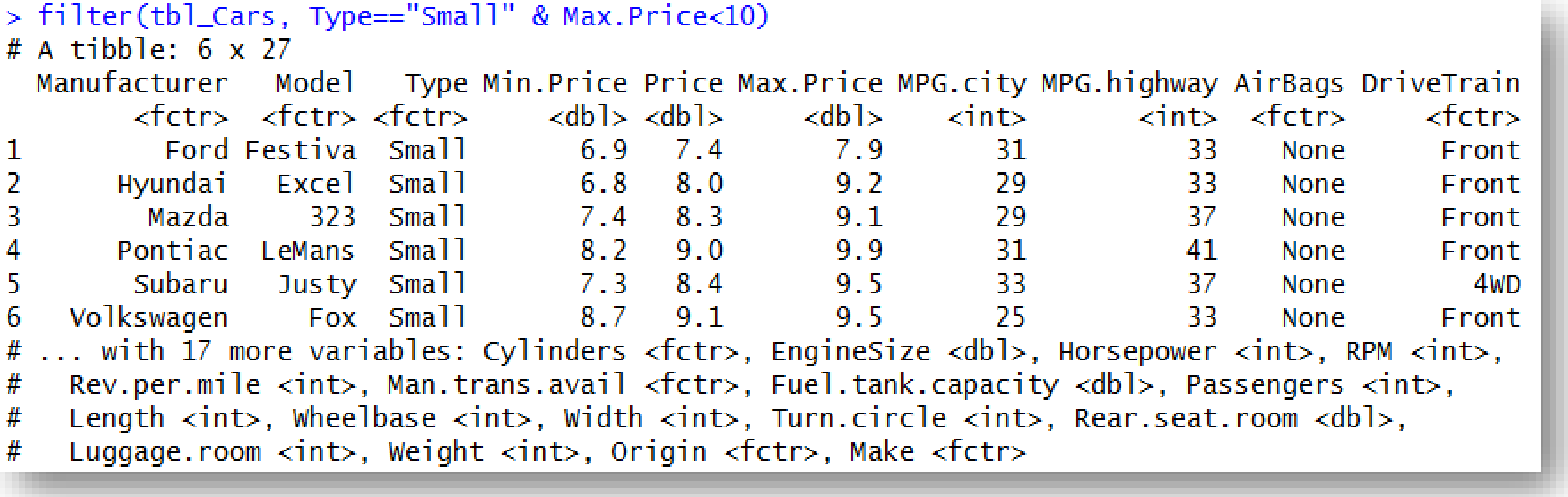
objtbl: Object of class tbl_df

criteria: Condition of filtering

filter examples

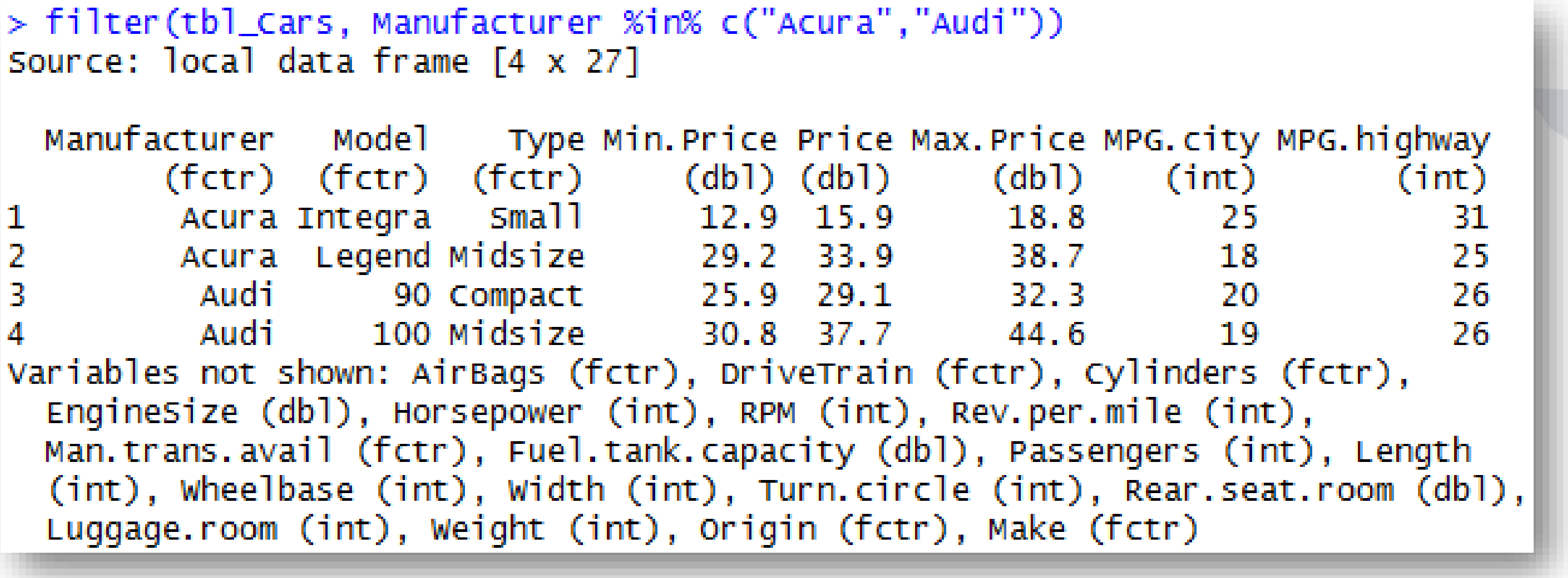


```
> filter(tbl_Cars, Type=="Small")
# A tibble: 21 x 27
   Manufacturer   Model   Type Min.Price Price Max.Price MPG.city MPG.highway     AirBags DriveTrain
         <fctr>  <fctr> <fctr>     <dbl> <dbl>     <dbl>    <int>       <int>      <fctr>     <fctr>
1         Acura Integra  Small      12.9  15.9      18.8       25          31        None      Front
2         Dodge    Colt  Small       7.9   9.2      10.6       29          33        None      Front
3         Dodge  Shadow  Small       8.4  11.3      14.2       23          29 Driver only      Front
4         Eagle  Summit  Small       7.9  12.2      16.5       29          33        None      Front
5          Ford Festiva  Small       6.9   7.4       7.9       31          33        None      Front
6          Ford  Escort  Small       8.4  10.1      11.9       23          30        None      Front
7           Geo   Metro  Small       6.7   8.4      10.0       46          50        None      Front
8         Honda   Civic  Small       8.4  12.1      15.8       42          46 Driver only      Front
9       Hyundai   Excel  Small       6.8   8.0       9.2       29          33        None      Front
10      Hyundai Elantra  Small       9.0  10.0      11.0       22          29        None      Front
# ... with 11 more rows, and 17 more variables: Cylinders <fctr>, EngineSize <dbl>, Horsepower <int>,
#   RPM <int>, Rev.per.mile <int>, Man.trans.avail <fctr>, Fuel.tank.capacity <dbl>,
#   Passengers <int>, Length <int>, Wheelbase <int>, Width <int>, Turn.circle <int>,
#   Rear.seat.room <dbl>, Luggage.room <int>, Weight <int>, Origin <fctr>, Make <fctr>
```



```
> filter(tbl_Cars, Type=="Small" & Max.Price<10)
# A tibble: 6 x 27
  Manufacturer   Model   Type Min.Price Price Max.Price MPG.city MPG.highway AirBags DriveTrain
        <fctr>  <fctr> <fctr>     <dbl> <dbl>     <dbl>    <int>       <int>  <fctr>     <fctr>
1         Ford Festiva  Small       6.9   7.4       7.9       31          33    None      Front
2      Hyundai   Excel  Small       6.8   8.0       9.2       29          33    None      Front
3        Mazda     323  Small       7.4   8.3       9.1       29          37    None      Front
4      Pontiac  LeMans  Small       8.2   9.0       9.9       31          41    None      Front
5       Subaru   Justy  Small       7.3   8.4       9.5       33          37    None        4WD
6   Volkswagen     Fox  Small       8.7   9.1       9.5       25          33    None      Front
# ... with 17 more variables: Cylinders <fctr>, EngineSize <dbl>, Horsepower <int>, RPM <int>,
#   Rev.per.mile <int>, Man.trans.avail <fctr>, Fuel.tank.capacity <dbl>, Passengers <int>,
#   Length <int>, Wheelbase <int>, Width <int>, Turn.circle <int>, Rear.seat.room <dbl>,
#   Luggage.room <int>, Weight <int>, Origin <fctr>, Make <fctr>
```

filter examples



```
> filter(tbl_Cars, Manufacturer %in% c("Acura","Audi"))
Source: local data frame [4 x 27]

  Manufacturer   Model    Type Min.Price Price Max.Price MPG.city MPG.highway
        (fctr)  (fctr)  (fctr)     (dbl) (dbl)     (dbl)    (int)       (int)
1        Acura Integra   Small      12.9  15.9      18.8       25          31
2        Acura  Legend Midsize      29.2  33.9      38.7       18          25
3         Audi      90 Compact      25.9  29.1      32.3       20          26
4         Audi     100 Midsize      30.8  37.7      44.6       19          26
Variables not shown: AirBags (fctr), DriveTrain (fctr), Cylinders (fctr),
  EngineSize (dbl), Horsepower (int), RPM (int), Rev.per.mile (int),
  Man.trans.avail (fctr), Fuel.tank.capacity (dbl), Passengers (int), Length
  (int), Wheelbase (int), Width (int), Turn.circle (int), Rear.seat.room (dbl),
  Luggage.room (int), Weight (int), Origin (fctr), Make (fctr)
```

Renaming Columns

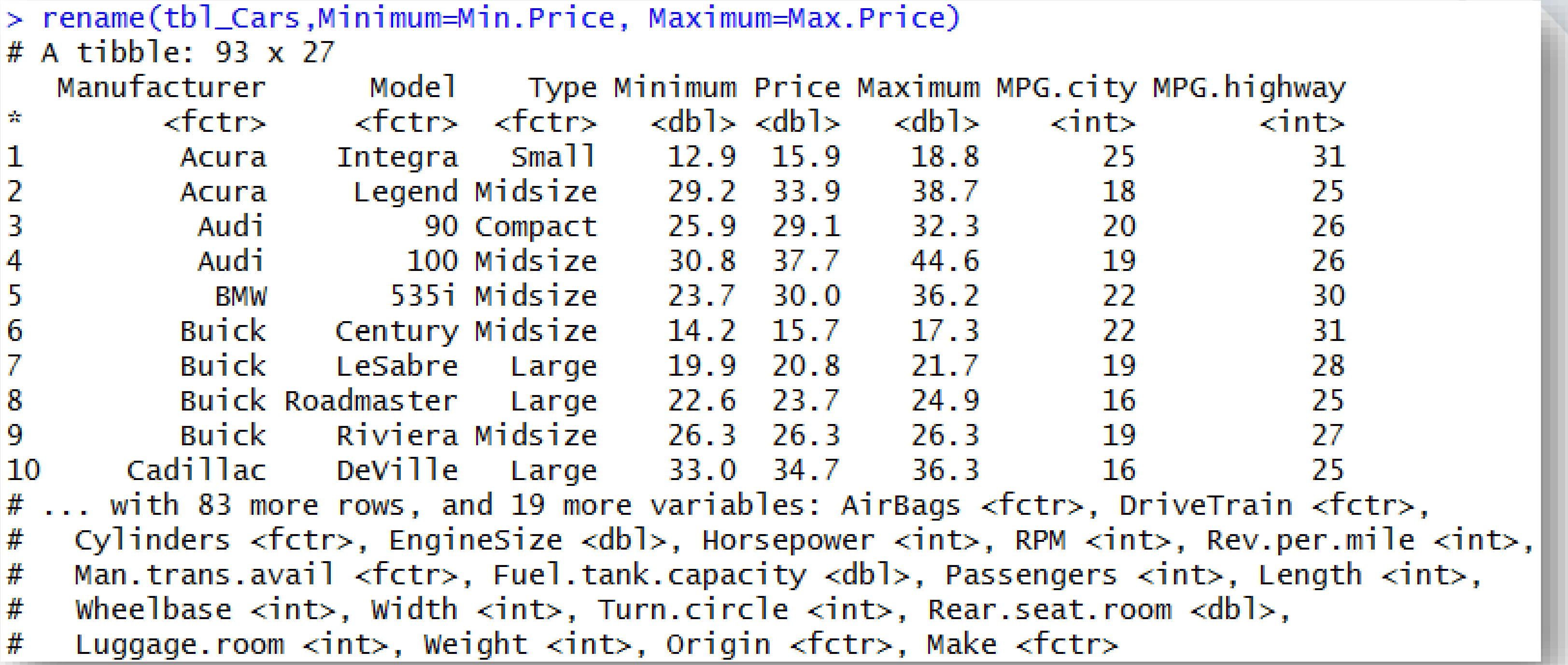
- The columns can renamed with function rename()

Syntax : rename(objtbl, newname1=oldname1, newname2=oldname2,…)

where

objtbl: Object of class tbl_df

rename example



```
> rename(tbl_Cars,Minimum=Min.Price, Maximum=Max.Price)
# A tibble: 93 x 27
   Manufacturer      Model    Type Minimum Price Maximum MPG.city MPG.highway
*        <fctr>     <fctr>  <fctr>   <dbl> <dbl>   <dbl>    <int>       <int>
1         Acura    Integra   Small    12.9  15.9    18.8       25          31
2         Acura     Legend Midsize    29.2  33.9    38.7       18          25
3          Audi         90 Compact    25.9  29.1    32.3       20          26
4          Audi        100 Midsize    30.8  37.7    44.6       19          26
5           BMW       535i Midsize    23.7  30.0    36.2       22          30
6         Buick    Century Midsize    14.2  15.7    17.3       22          31
7         Buick    LeSabre   Large    19.9  20.8    21.7       19          28
8         Buick Roadmaster   Large    22.6  23.7    24.9       16          25
9         Buick    Riviera Midsize    26.3  26.3    26.3       19          27
10     Cadillac    DeVille   Large    33.0  34.7    36.3       16          25
# ... with 83 more rows, and 19 more variables: AirBags <fctr>, DriveTrain <fctr>,
#   Cylinders <fctr>, EngineSize <dbl>, Horsepower <int>, RPM <int>, Rev.per.mile <int>,
#   Man.trans.avail <fctr>, Fuel.tank.capacity <dbl>, Passengers <int>, Length <int>,
#   Wheelbase <int>, Width <int>, Turn.circle <int>, Rear.seat.room <dbl>,
#   Luggage.room <int>, Weight <int>, Origin <fctr>, Make <fctr>
```

Adding new column

- We can create one or more new columns / variables in the data with function mutate

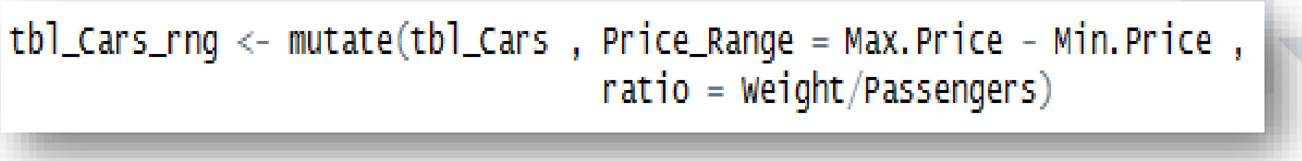
Syntax : mutate(objtbl , assign)

where

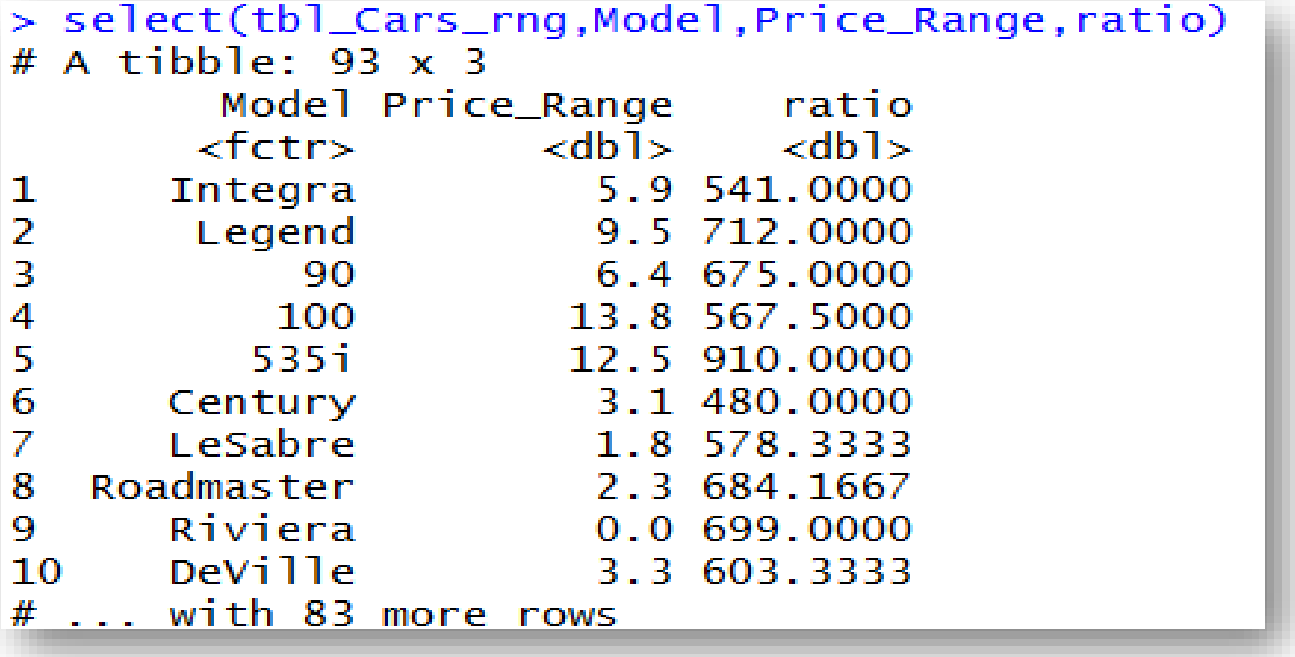
objtbl: Object of class tbl_df

assign: Specification of assignment for new column

mutate example



```
tbl_Cars_rng <- mutate(tbl_Cars , Price_Range = Max.Price - Min.Price ,
                                  ratio = Weight/Passengers)
```



```
> select(tbl_Cars_rng,Model,Price_Range,ratio)
# A tibble: 93 x 3
        Model Price_Range    ratio
       <fctr>       <dbl>    <dbl>
1     Integra         5.9 541.0000
2      Legend         9.5 712.0000
3          90         6.4 675.0000
4         100        13.8 567.5000
5        535i        12.5 910.0000
6     Century         3.1 480.0000
7     LeSabre         1.8 578.3333
8  Roadmaster         2.3 684.1667
9     Riviera         0.0 699.0000
10    DeVille         3.3 603.3333
# ... with 83 more rows
```

Summarizing the data

- The data can be summarized with the function summarize/summarise

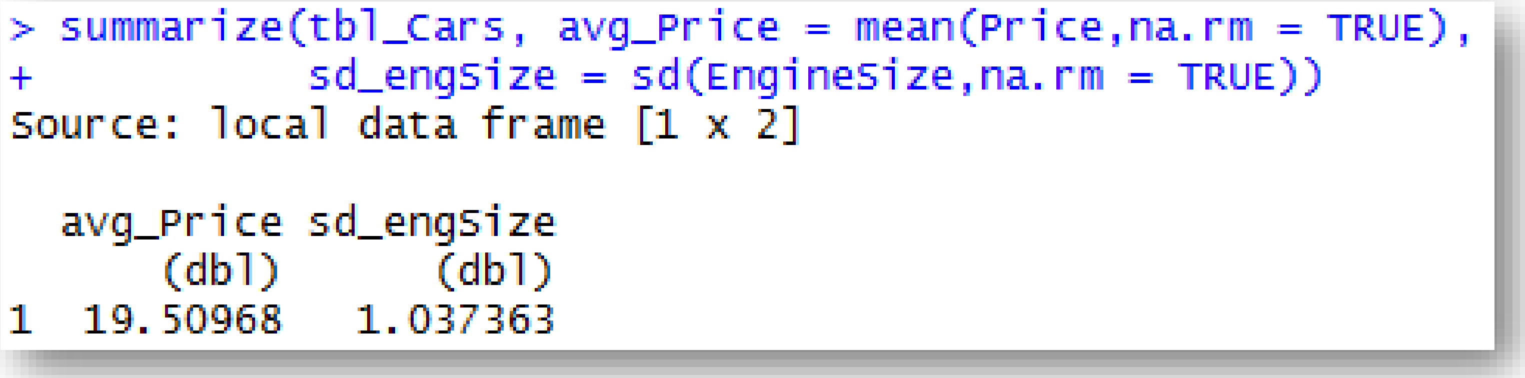
Syntax : summarize(objtbl, assign)

where

objtbl: Object of class tbl_df

assign: Specification of assignment for new column

summarize example



```
> summarize(tbl_Cars, avg_Price = mean(Price,na.rm = TRUE),
+           sd_engSize = sd(EngineSize,na.rm = TRUE))
Source: local data frame [1 x 2]

  avg_Price sd_engSize
      (dbl)      (dbl)
1  19.50968   1.037363
```

Grouping

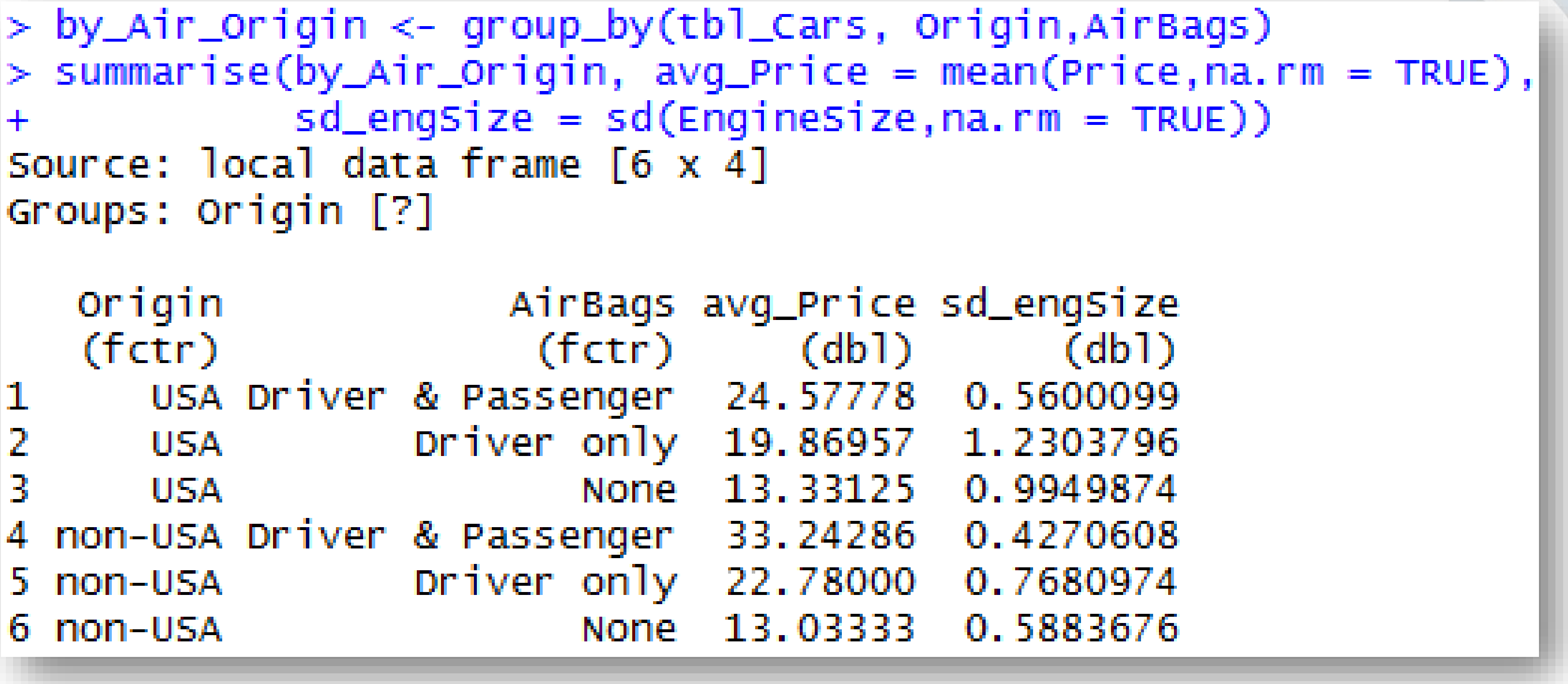
- The group_by function takes an existing tbl and converts it into a grouped tbl where operations are performed "by group".

Syntax : group_by(objtbl)

where

objtbl: Object of class tbl_df

Group by example



```
> by_Air_Origin <- group_by(tbl_Cars, Origin,AirBags)
> summarise(by_Air_Origin, avg_Price = mean(Price,na.rm = TRUE),
+           sd_engSize = sd(EngineSize,na.rm = TRUE))
Source: local data frame [6 x 4]
Groups: Origin [?]

   Origin              AirBags avg_Price sd_engSize
   (fctr)               (fctr)     (dbl)      (dbl)
1     USA Driver & Passenger  24.57778  0.5600099
2     USA        Driver only  19.86957  1.2303796
3     USA               None  13.33125  0.9949874
4 non-USA Driver & Passenger  33.24286  0.4270608
5 non-USA        Driver only  22.78000  0.7680974
6 non-USA               None  13.03333  0.5883676
```

Chaining / Pipelining

- We can pipeline the operations which are consecutive to one tbl object using %>% operator.

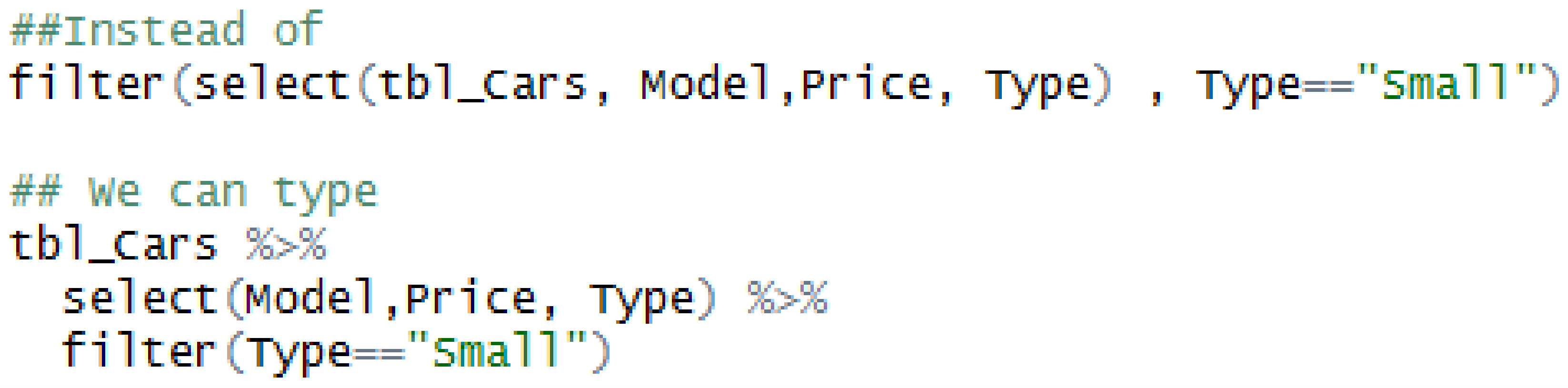
Syntax :

objtbl %>% operations

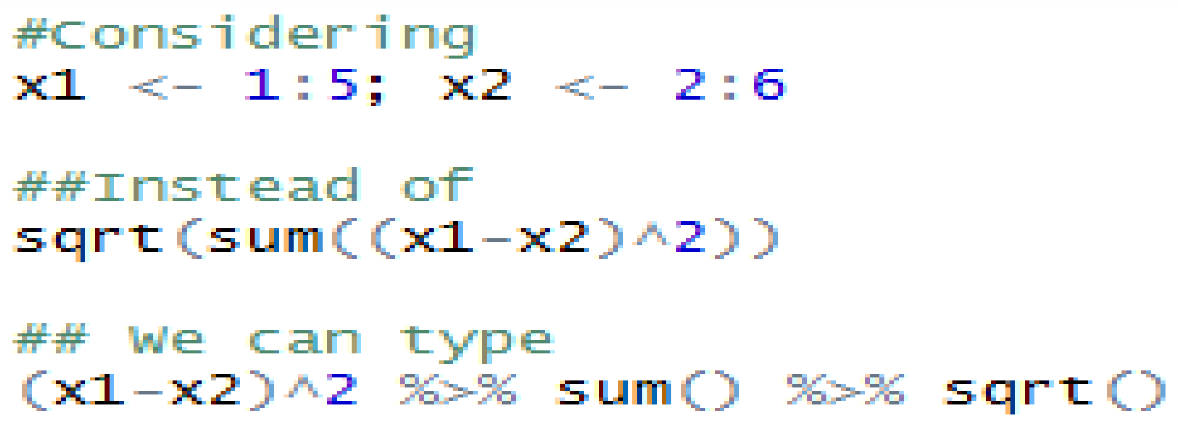
where

objtbl: Object of class tbl_df

Pipelining the data operations



```
##Instead of
filter(select(tbl_Cars, Model,Price, Type) , Type=="Small")

## We can type
tbl_Cars %>%
  select(Model,Price, Type) %>%
  filter(Type=="Small")
```

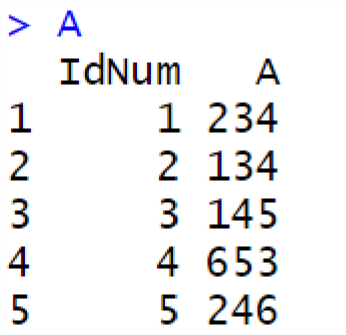


```
#Considering
x1 <- 1:5; x2 <- 2:6

##Instead of
sqrt(sum((x1-x2)^2))

## We can type
(x1-x2)^2 %>% sum() %>% sqrt()
```

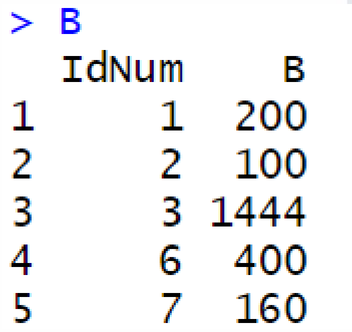
Joins

- In package ***dplyr***, we have all the types of joins like
  - Inner join
  - Left Join
  - Right Join
  - Full Join

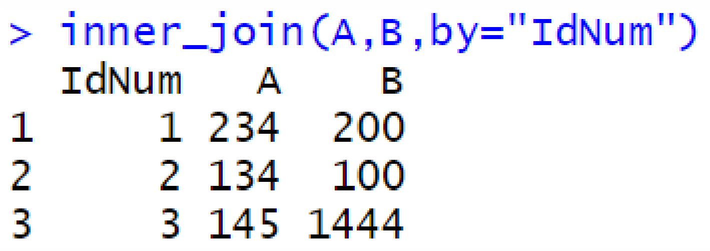
Inner Join



```
> A
  IdNum   A
1     1 234
2     2 134
3     3 145
4     4 653
5     5 246
```

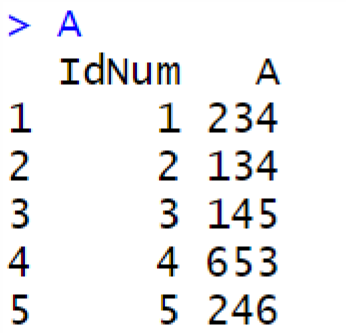


```
> B
  IdNum    B
1     1  200
2     2  100
3     3 1444
4     6  400
5     7  160
```

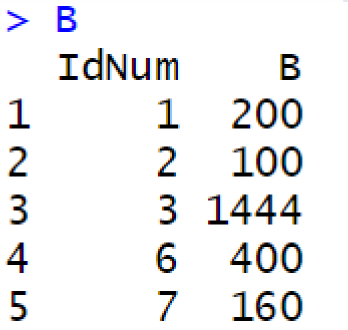


```
> inner_join(A,B,by="IdNum")
  IdNum   A    B
1     1 234  200
2     2 134  100
3     3 145 1444
```

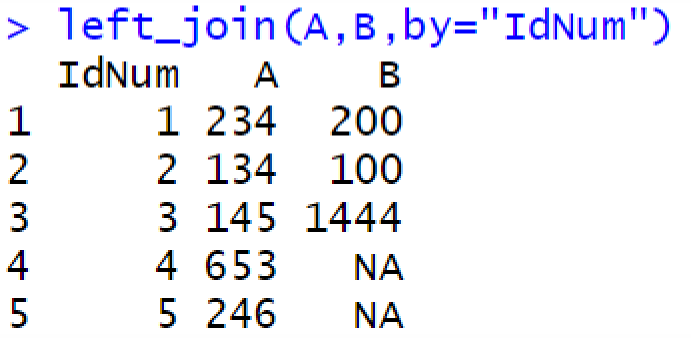
Left Outer Join



```
> A
  IdNum   A
1     1 234
2     2 134
3     3 145
4     4 653
5     5 246
```

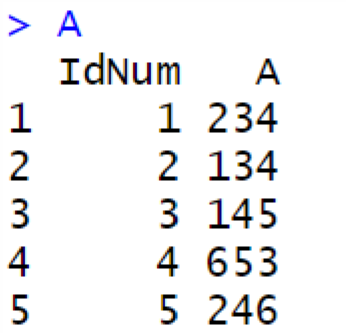


```
> B
  IdNum    B
1     1  200
2     2  100
3     3 1444
4     6  400
5     7  160
```

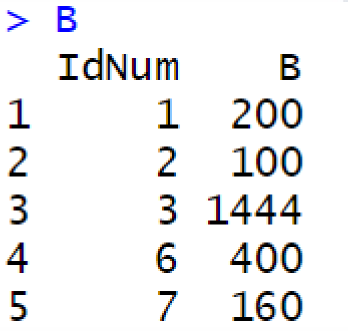


```
> left_join(A,B,by="IdNum")
  IdNum   A    B
1     1 234  200
2     2 134  100
3     3 145 1444
4     4 653   NA
5     5 246   NA
```

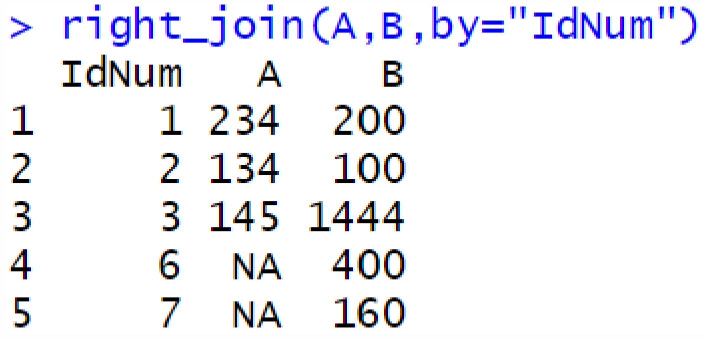
Right Outer Join



```
> A
  IdNum   A
1     1 234
2     2 134
3     3 145
4     4 653
5     5 246
```



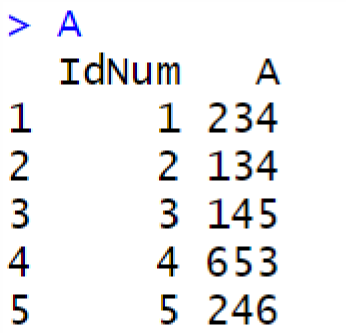
```
> B
  IdNum    B
1     1  200
2     2  100
3     3 1444
4     6  400
5     7  160
```



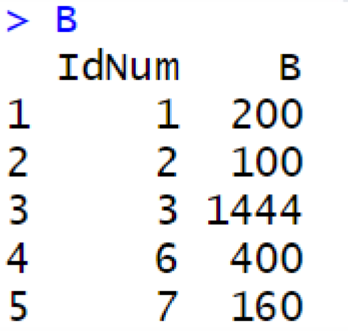
```
> right_join(A,B,by="IdNum")
  IdNum   A    B
1     1 234  200
2     2 134  100
3     3 145 1444
4     6  NA  400
5     7  NA  160
```

Sane's STATS Academy of Statistics

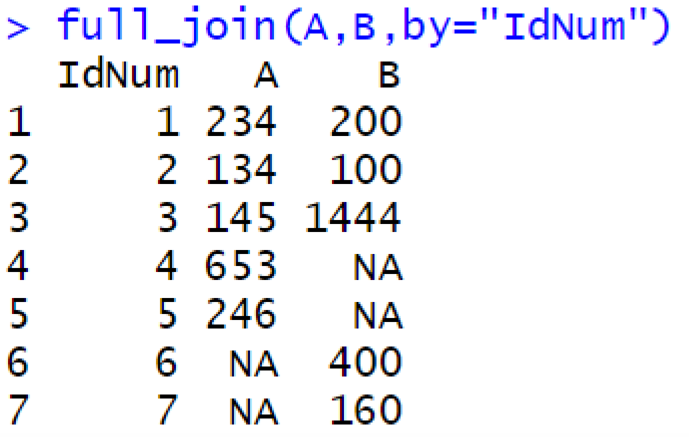
Full Outer Join



```
> A
  IdNum   A
1     1 234
2     2 134
3     3 145
4     4 653
5     5 246
```



```
> B
  IdNum    B
1     1  200
2     2  100
3     3 1444
4     6  400
5     7  160
```



```
> full_join(A,B,by="IdNum")
  IdNum   A    B
1     1 234  200
2     2 134  100
3     3 145 1444
4     4 653   NA
5     5 246   NA
6     6  NA  400
7     7  NA  160
```