# Wrangling with Dataframes

TRSM R Bootcamp

## Introduction

Being able to manipulate data is an important skill needed for any kind of analysis. In this section, various functions are introduced to demonstrate several techniques and common practices for working with data. While exploring dataframes, various techniques related to plotting will be introduced as well.

Some useful functions will be introduced to wrangle data. Additionally, pipes are introduced to make it easier to manipulate data.

### Table of content



**Setup**: Importing libraries and datasets needed through the session.

```
# Set up:
library(ggplot2)
library(tidyverse)

# Creating dataframes using data.frame() object
courses <- c("ECN340", "MTH108", "CPS109", "ECN702", "ITM320")
rates <- c(9.7, 5.4, 7.8, 6.5, 8.9)
# Course Rating dataframe
d <- data.frame(courses, rates)

# Importing the housing dataframe
housing_df <- read_csv("https://www.dropbox.com/s/tvvtf9dwjufo7os/housing_train.csv?dl=1")

# Subset of housing_df to make it easier to run in console
df <- housing_df[1:6, c("SalePrice", "MSSubClass", "MSZoning", "LotFrontage")]

playground <- read_csv("https://www.dropbox.com/s/1zz4a82514avdeq/playground_injuries.csv?dl=1")
```

## 1. Select

- Selecting specific columns using either names or index columns
- **select(df, A, B ,C)**: Get columns A, B, and C from the dataframe df

```
# Selecting "SalePrice" and "MSSubClass" columns
select(df, SalePrice, MSSubClass)
```

```
## # A tibble: 6 x 2
##   SalePrice MSSubClass
##       <dbl>      <dbl>
## 1    208500         60
## 2    181500         20
## 3    223500         60
## 4    140000         70
## 5    250000         60
## 6    143000         50
```

- **select(df, A:C)**: Select all variables from A to C from df dataset.

```
# Selecting columns from MSSubClass to LotFrontage
select(df, MSSubClass:LotFrontage)
```

```
## # A tibble: 6 x 3
##   MSSubClass MSZoning LotFrontage
##        <dbl> <chr>          <dbl>
## 1         60 RL                65
## 2         20 RL                80
## 3         60 RL                68
## 4         70 RL                60
## 5         60 RL                84
## 6         50 RL                85
```

- **select(df, A:F, -C)**: All columns from to A to F excluding C

```
# Selecting columns from MSSubClass to LotFrontage excluding MSZoning
select(df, MSSubClass:LotFrontage, -MSZoning)
```

```
## # A tibble: 6 x 2
##   MSSubClass LotFrontage
##        <dbl>       <dbl>
## 1         60          65
## 2         20          80
## 3         60          68
## 4         70          60
## 5         60          84
## 6         50          85
```

- **select(df, contains(str))**: Columns that contain str in their name

```
# Selecting all the columns where there contain bsmt in their name
select(housing_df[1:5,], contains("bsmt"))
```

```
## # A tibble: 5 x 11
##   BsmtQual BsmtCond BsmtExposure BsmtFinType1 BsmtFinSF1 BsmtFinType2 BsmtFinSF2
##   <chr>    <chr>    <chr>        <chr>             <dbl> <chr>             <dbl>
## 1 Gd       TA       No           GLQ                 706 Unf                   0
## 2 Gd       TA       Gd           ALQ                 978 Unf                   0
## 3 Gd       TA       Mn           GLQ                 486 Unf                   0
## 4 TA       Gd       No           ALQ                 216 Unf                   0
## 5 Gd       TA       Av           GLQ                 655 Unf                   0
## # ... with 4 more variables: BsmtUnfSF <dbl>, TotalBsmtSF <dbl>,
## #   BsmtFullBath <dbl>, BsmtHalfBath <dbl>
```

**Exercise**: Use selection to get the following subsets

1. 7th to 12th Column

2. 7th to 12th Column excluding columns 8 and Alley
3. Columns GarageType to GarageCond
4. Columns GarageType to GarageCond exclusive (excluding GarageType and GarageCond)
5. Columns with bsmt in their names

## 2. Filter

- **filter(df, condition)**:
  - df: dataset used to filter the data
  - condition: Condition used to filter the data
- **multiple conditions**:
  - filter(df, condition1 | condition2)
  - filter(df, condition1 & condition2)

**Note**: Negation is also possible within this notation.

```
# Filter for houses with over a 3 in overall quality
filter(select(housing_df[1:5,], BldgType:YearBuilt), OverallQual > 3 )
```

```
## # A tibble: 5 x 5
##   BldgType HouseStyle OverallQual OverallCond YearBuilt
##   <chr>    <chr>            <dbl>       <dbl>     <dbl>
## 1 1Fam     2Story               7           5      2003
## 2 1Fam     1Story               6           8      1976
## 3 1Fam     2Story               7           5      2001
## 4 1Fam     2Story               7           5      1915
## 5 1Fam     2Story               8           5      2000
```

```
# Filter for houses with above median sale prices
filter(df, SalePrice > median(SalePrice))
```

```
## # A tibble: 3 x 4
##   SalePrice MSSubClass MSZoning LotFrontage
##       <dbl>      <dbl> <chr>          <dbl>
## 1    208500         60 RL                65
## 2    223500         60 RL                68
## 3    250000         60 RL                84
```

**Exercises**:

1. Choose rows with MSSubclass of higher than 20
2. MSSubclass of higher than 20 and SalePrice below average
3. MSSubclass anything but 60, not a Low LandContour
4. Entries where SalePrice is higher than average or Neighborhood is BrDale

## 3. Mutate:

- Mutating entries within the dataframe
- **General Structure**: mutate(df, new_column=operation(column1, column2, . . . ))

```
# Adding log_sale_price column which is the log of the SalePrice columns
housing_df <- mutate(housing_df, log_sale_price = log(SalePrice))
# Create a new column in housing_df called **total_sf** which adds the square feet the basement, first,
housing_df <- mutate(housing_df, total_sf =  TotalBsmtSF + `1stFlrSF`  + `2ndFlrSF` )
```

# 4. Pipes

Run the series of commands below to obtain a dataframe containing columns SalePrice ,log_SalePrice, and MSZoning Where the SalePrice values are greater than the average of SalePrice value.

```
# 3 steps:
# Select SalePrice and MSZoning columns
step1 <- select(df, SalePrice, MSZoning)
# Keep rows where SalePrice is higher than the average
step2 <- filter(step1, SalePrice > mean(SalePrice))
# Make a new column called log_SalePrice where it is the log of SalePrice column
step3 <- mutate(step2, log_SalePrice=log(SalePrice))

# Notice: Each time you are passing the preceding dataframe (first df, then step1...)

step3
```

```
## # A tibble: 3 x 3
##   SalePrice MSZoning log_SalePrice
##       <dbl> <chr>            <dbl>
## 1    208500 RL                12.2
## 2    223500 RL                12.3
## 3    250000 RL                12.4
```

- The method above contains several steps and it might get hard to follow
- Pipes are an alternative to make it easier to pass data and manipulate it with a better format

```
# Doing the same procedure using pipes
pipe_way <- housing_df %>%
  select(SalePrice, MSZoning) %>%
  filter(SalePrice > mean(SalePrice)) %>%
  mutate(log_SalePrice=log(SalePrice))

# Note: Notice that we do not need to pass the data!

pipe_way
```

```
## # A tibble: 560 x 3
##    SalePrice MSZoning log_SalePrice
##        <dbl> <chr>            <dbl>
##  1    208500 RL                12.2
##  2    181500 RL                12.1
##  3    223500 RL                12.3
##  4    250000 RL                12.4
##  5    307000 RL                12.6
##  6    200000 RL                12.2
##  7    345000 RL                12.8
##  8    279500 RL                12.5
##  9    325300 RL                12.7
## 10    230000 RL                12.3
## # ... with 550 more rows
```

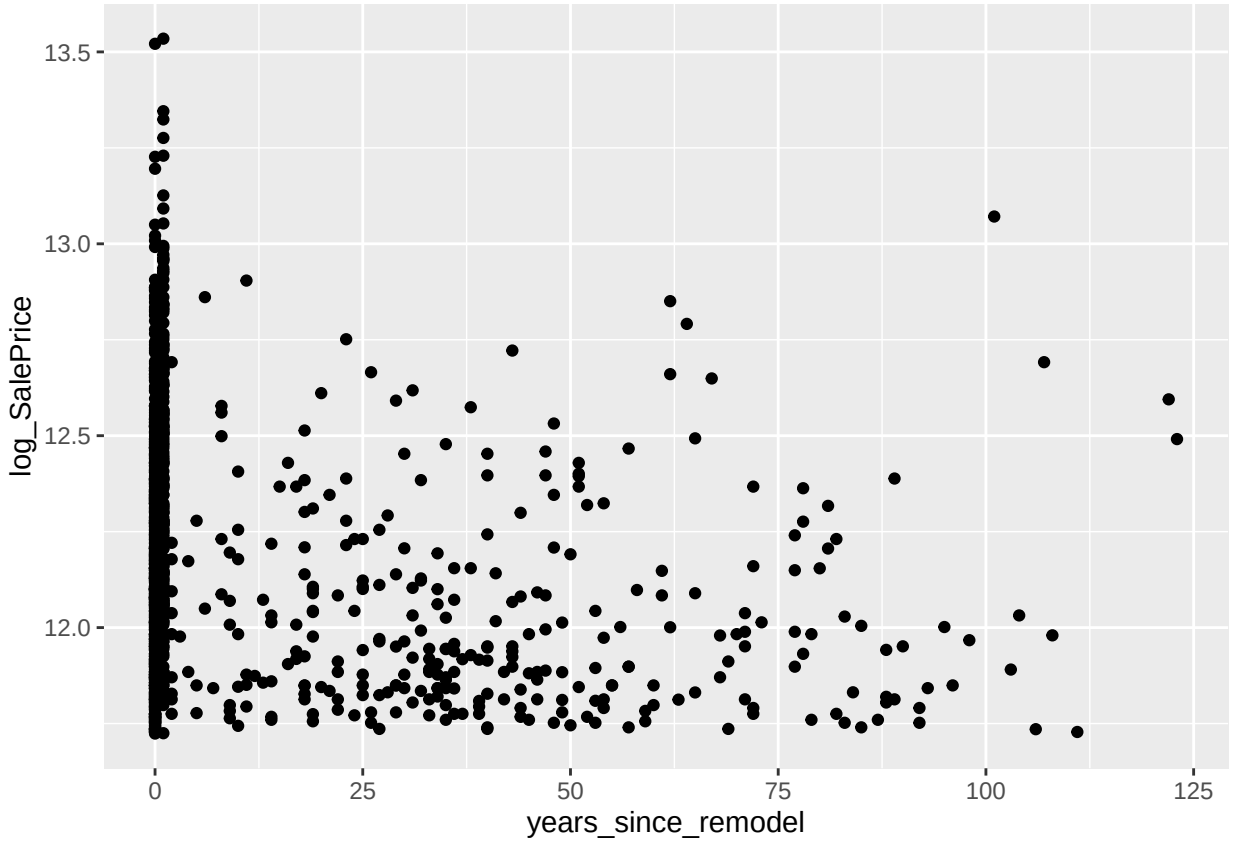
- Notice that data is being passed on in each line via %>% (pipe operator).
- We do not need to pass the data in each step.
- It is convenient to use the pipe operator when there are more than one step.

### Case Study

We are interested to know the effect of remodeling the house on the houses higher than 123, 000 price value:

1. Getting the rows with price higher than 123e3
2. Add new column called years_since_remodel
3. Plot two variables

```
housing_df %>%
    filter(SalePrice > 123e3) %>% # 1
    mutate(years_since_remodel = YearRemodAdd - YearBuilt, log_SalePrice=log(SalePrice)) %>% # 2
    ggplot(aes(x=years_since_remodel, y=log_SalePrice)) + geom_point() # 2
```




## 5. Arrange

- Sort dataframe based on given columns
- **arrange(A)**: Ascending sort of variable A
- **arrange(A, B)**: Ascending sort of variable A and B
- **arrange(desc(A), B)/arrange(-A, B)**: Descending sort of variable A and ascending sort of B

```
# Ascending sort of housing dataframe based on sale_price
df %>% arrange(SalePrice)
```

```
## # A tibble: 6 x 4
##   SalePrice MSSubClass MSZoning LotFrontage
##       <dbl>      <dbl> <chr>          <dbl>
## 1    140000         70 RL                60
## 2    143000         50 RL                85
## 3    181500         20 RL                80
```

```
## 4    208500         60 RL                65
## 5    223500         60 RL                68
## 6    250000         60 RL                84
```

```
# Descending sort of housing dataframe based on sale_price
df %>% arrange(-SalePrice)
```

```
## # A tibble: 6 x 4
##   SalePrice MSSubClass MSZoning LotFrontage
##       <dbl>      <dbl> <chr>           <dbl>
## 1    250000         60 RL                84
## 2    223500         60 RL                68
## 3    208500         60 RL                65
## 4    181500         20 RL                80
## 5    143000         50 RL                85
## 6    140000         70 RL                60
```