NC STATE UNIVERSITY

Manipulating Data: Creating New Variables

Justin Post

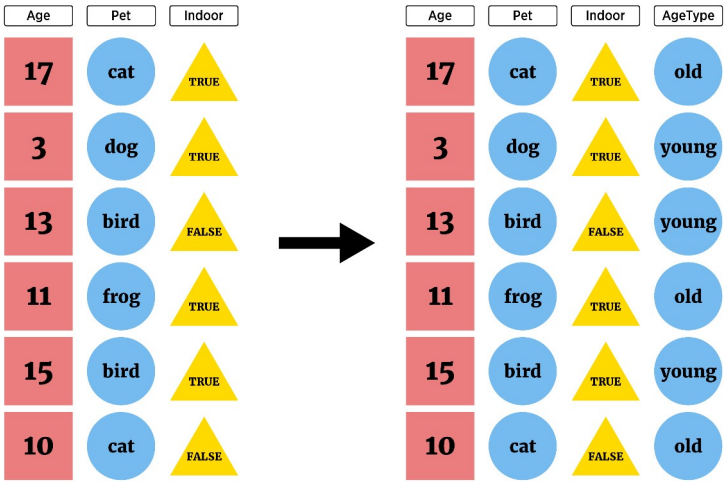
Recap/Next Up!

- Data manipulation idea
- Documenting with Markdown
- Logical statements
- `dplyr`
- Creating new variables
    - Conditional execution (if then)
    - For loops
    - Vectorized functions
- Reshaping data

Data manipulation idea

We may want to subset our full data set or create new data

- Create new variables



Creating New Variables

Given a data frame and an appropriate length vector (a new variable), you can use `cbind` (column bind) to add the variable to the dataframe

```
temp <- cbind(iris, extra = rep("a", 150))
str(temp)
```

```
## 'data.frame':    150 obs. of  6 variables:
##  $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
##  $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
##  $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
##  $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
##  $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ extra       : chr  "a" "a" "a" "a" ...
```

Creating New Variables

Or simply add as a named (list) element!

```
iris$extra <- rep("a", 150)
str(iris)
```

```
## 'data.frame':    150 obs. of  6 variables:
##  $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
##  $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
##  $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
##  $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
##  $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ extra       : chr  "a" "a" "a" "a" ...
```

Creating New Variables

Better method: use `dplyr`!

- `mutate()` - add newly created **column(s)** to current data frame
- `transmute()` - create new data frame with created variable(s)

Creating New Variables

Better method: use `dplyr`!

- `mutate()` - add newly created **column(s)** to current data frame
- `transmute()` - create new data frame with created variable(s)
- Syntax:

```
mutate(data, newVarName = functionOfData, newVarName2 =
functionOfData, ...)
```

Creating New Variables

- Consider a data set on movie ratings

```
library(fivethirtyeight)
fandango
```

```
## # A tibble: 146 x 23
##   film   year rottentomatoes rottentomatoes_… metacritic metacritic_user  imdb
##   <chr> <dbl>          <int>            <int>      <int>           <dbl> <dbl>
## 1 Aven…  2015             74               86         66             7.1   7.8
## 2 Cind…  2015             85               80         67             7.5   7.1
## 3 Ant-…  2015             80               90         64             8.1   7.8
## 4 Do Y…  2015             18               84         22             4.7   5.4
## 5 Hot …  2015             14               28         29             3.4   5.1
## # … with 141 more rows, and 16 more variables: fandango_stars <dbl>,
## #   fandango_ratingvalue <dbl>, rt_norm <dbl>, rt_user_norm <dbl>,
## #   metacritic_norm <dbl>, metacritic_user_nom <dbl>, imdb_norm <dbl>,
## #   rt_norm_round <dbl>, rt_user_norm_round <dbl>, metacritic_norm_round <dbl>,
## #   metacritic_user_norm_round <dbl>, imdb_norm_round <dbl>,
## #   metacritic_user_vote_count <int>, imdb_user_vote_count <int>,
## #   fandango_votes <int>, fandango_difference <dbl>
```

Creating New Variables

- `mutate()` - add newly created **column(s)** to current data frame

```
##Create an average rottentomatoes score variable
fandango %>% mutate(avgRotten = (rottentomatoes + rottentomatoes_user)/2)
```

```
## # A tibble: 146 x 24
##   film   year rottentomatoes rottentomatoes_… metacritic metacritic_user  imdb
##   <chr> <dbl>          <int>            <int>      <int>           <dbl> <dbl>
## 1 Aven…  2015             74               86         66             7.1   7.8
## 2 Cind…  2015             85               80         67             7.5   7.1
## 3 Ant-…  2015             80               90         64             8.1   7.8
## 4 Do Y…  2015             18               84         22             4.7   5.4
## 5 Hot …  2015             14               28         29             3.4   5.1
## # … with 141 more rows, and 17 more variables: fandango_stars <dbl>,
## #   fandango_ratingvalue <dbl>, rt_norm <dbl>, rt_user_norm <dbl>,
## #   metacritic_norm <dbl>, metacritic_user_nom <dbl>, imdb_norm <dbl>,
## #   rt_norm_round <dbl>, rt_user_norm_round <dbl>, metacritic_norm_round <dbl>,
## #   metacritic_user_norm_round <dbl>, imdb_norm_round <dbl>,
## #   metacritic_user_vote_count <int>, imdb_user_vote_count <int>,
## #   fandango_votes <int>, fandango_difference <dbl>, avgRotten <dbl>
```

Creating New Variables

- `mutate()` - add newly created **column(s)** to current data frame

```
#can't see it!
fandango %>% mutate(avgRotten = (rottentomatoes + rottentomatoes_user)/2) %>%
  select(film, year, avgRotten, everything())
```

```
## # A tibble: 146 x 24
##   film   year avgRotten rottentomatoes rottentomatoes_… metacritic
##   <chr> <dbl>     <dbl>          <int>            <int>      <int>
## 1 Aven…  2015      80               74               86         66
## 2 Cind…  2015      82.5             85               80         67
## 3 Ant-…  2015      85               80               90         64
## 4 Do Y…  2015      51               18               84         22
## 5 Hot …  2015      21               14               28         29
## # … with 141 more rows, and 18 more variables: metacritic_user <dbl>,
## #   imdb <dbl>, fandango_stars <dbl>, fandango_ratingvalue <dbl>,
## #   rt_norm <dbl>, rt_user_norm <dbl>, metacritic_norm <dbl>,
## #   metacritic_user_nom <dbl>, imdb_norm <dbl>, rt_norm_round <dbl>,
## #   rt_user_norm_round <dbl>, metacritic_norm_round <dbl>,
## #   metacritic_user_norm_round <dbl>, imdb_norm_round <dbl>,
## #   metacritic_user_vote_count <int>, imdb_user_vote_count <int>,
## #   fandango_votes <int>, fandango_difference <dbl>
```

Creating New Variables

- `mutate()` - add newly created **column(s)** to current data frame
- Add more than one variable

```
fandango %>%
  mutate(avgRotten = (rottentomatoes + rottentomatoes_user)/2,
         avgMeta = (metacritic_norm + metacritic_user_nom)/2) %>%
  select(film, year, avgRotten, avgMeta, everything())
```

```
## # A tibble: 146 x 25
##   film   year avgRotten avgMeta rottentomatoes rottentomatoes_… metacritic
##   <chr> <dbl>     <dbl>   <dbl>          <int>            <int>      <int>
## 1 Aven…  2015      80      3.42             74               86         66
## 2 Cind…  2015      82.5    3.55             85               80         67
## 3 Ant-…  2015      85      3.62             80               90         64
## 4 Do Y…  2015      51      1.72             18               84         22
## 5 Hot …  2015      21      1.58             14               28         29
## # … with 141 more rows, and 18 more variables: metacritic_user <dbl>,
## #   imdb <dbl>, fandango_stars <dbl>, fandango_ratingvalue <dbl>,
## #   rt_norm <dbl>, rt_user_norm <dbl>, metacritic_norm <dbl>,
## #   metacritic_user_nom <dbl>, imdb_norm <dbl>, rt_norm_round <dbl>,
## #   rt_user_norm_round <dbl>, metacritic_norm_round <dbl>,
## #   metacritic_user_norm_round <dbl>, imdb_norm_round <dbl>,
## #   metacritic_user_vote_count <int>, imdb_user_vote_count <int>,
## #   fandango_votes <int>, fandango_difference <dbl>
```

Creating New Variables

- `transmute()` - create new data frame with created variable(s)

```
#transmute will keep the new variable(s) only
fandango %>% transmute(avgRotten = (rottentomatoes + rottentomatoes_user)/2)
```

```
## # A tibble: 146 x 1
##   avgRotten
##       <dbl>
## 1      80
## 2      82.5
## 3      85
## 4      51
## 5      21
## # … with 141 more rows
```

Creating New Variables

- `transmute()` - create new data frame with created variable(s)

```
#transmute will keep the new variable(s) only
fandango %>% transmute(avgRotten = (rottentomatoes + rottentomatoes_user)/2,
                       avgMeta = (metacritic_norm + metacritic_user_nom)/2)
```

```
## # A tibble: 146 x 2
##   avgRotten avgMeta
##       <dbl>   <dbl>
## 1      80      3.42
## 2      82.5    3.55
## 3      85      3.62
## 4      51      1.72
## 5      21      1.58
## # … with 141 more rows
```

Creating New Variables

`mutate` and `transmute` can also use 'window' functions

- Functions that take a vector of values and return another vector of values (see Cheat sheet)

```
fandango %>% select(rottentomatoes) %>% mutate(cumulativeSum = cumsum(rottentomatoes))
```

```
## # A tibble: 146 x 2
##   rottentomatoes cumulativeSum
##            <int>         <int>
## 1             74            74
## 2             85           159
## 3             80           239
## 4             18           257
## 5             14           271
## # … with 141 more rows
```

Creating New Variables

`mutate` and `transmute` can also use some statistical functions

```
fandango %>% select(rottentomatoes) %>%
  mutate(avg = mean(rottentomatoes), sd = sd(rottentomatoes))
```

```
## # A tibble: 146 x 3
##   rottentomatoes   avg    sd
##            <int> <dbl> <dbl>
## 1             74  60.8  30.2
## 2             85  60.8  30.2
## 3             80  60.8  30.2
## 4             18  60.8  30.2
## 5             14  60.8  30.2
## # … with 141 more rows
```

Creating New Variables

`mutate` and `transmute` can also use some statistical functions

- `group_by` to create summaries for groups (more on this later)

```
fandango %>% select(year, rottentomatoes) %>%
  group_by(year) %>% mutate(avg = mean(rottentomatoes), sd = sd(rottentomatoes))
```

```
## # A tibble: 146 x 4
## # Groups:   year [2]
##    year rottentomatoes   avg    sd
##   <dbl>          <int> <dbl> <dbl>
## 1  2015             74  58.4  30.3
## 2  2015             85  58.4  30.3
## 3  2015             80  58.4  30.3
## 4  2015             18  58.4  30.3
## 5  2015             14  58.4  30.3
## # … with 141 more rows
```

Creating New Variables

Conditional Execution with If then, If then else

- Often want to execute statements conditionally to create a variable
- `if then else` syntax

```
if (condition) {
  then execute code
}

#if then else
if (condition) {
  execute this code
} else {
  execute this code
}
```

```
#Or more if statements
if (condition) {
  execute this code
} else if (condition2) {
  execute this code
} else if (condition3) {
  execute this code
} else {
  #if no conditions met
  execute this code
}
```

Creating New Variables

Conditional Execution with If then, If then else

- Consider built-in data set `airquality`
    - daily air quality measurements in New York
    - from May (Day 1) to September (Day 153) in 1973

Creating New Variables

Conditional Execution with If then, If then else

- Consider built-in data set `airquality`

```
airquality <- as_tibble(airquality)
airquality
```

```
## # A tibble: 153 x 6
##   Ozone Solar.R  Wind  Temp Month   Day
##   <int>   <int> <dbl> <int> <int> <int>
## 1    41     190   7.4    67     5     1
## 2    36     118   8      72     5     2
## 3    12     149  12.6    74     5     3
## 4    18     313  11.5    62     5     4
## 5    NA      NA  14.3    56     5     5
## # … with 148 more rows
```

Creating New Variables

Conditional Execution with If then, If then else

Want to code a wind category variable

- high wind days (wind $\geq$ 15mph)
- windy days (10mph $\leq$ wind < 15mph)
- lightwind days (6mph $\leq$ wind < 10mph)
- calm days (wind $\leq$ 6mph)

Creating New Variables

Conditional Execution with If then, If then else

Want to code a wind category variable

Issue: `if(condition)` can only take in a single comparison

```
if(airquality$Wind >= 15) {
  "High Wind"
  }
```

```
## Warning in if (airquality$Wind >= 15) {: the condition has length > 1 and only
## the first element will be used
```

Creating New Variables

Conditional Execution with If then, If then else

Want to code a wind category variable

- high wind days (15mph $\leq$ wind)
- windy days (10mph $\leq$ wind < 15mph)
- lightwind days (6mph $\leq$ wind < 10mph)
- calm days (wind $\leq$ 6mph)

Initial plan

- loop through each observation
- use if then else to determine wind status

Looping in R

- There are a number of ways to do looping in R
  - `for()`
  - `while()`
  - `repeat()`
- Idea:
  - Run code repeatedly changing something each time

Looping in R

- Syntax

```
for(index in values){
  code to be run
}
```

- index defines ‘counter’ or variable that varies
- ‘values’ define which values index takes on

Looping in R

- index defines 'counter' or variable that varies
- 'values' define which values index takes on

```
for (i in 1:10){
  print(i)
}
```

```
## [1] 1
## [1] 2
## [1] 3
## [1] 4
## [1] 5
## [1] 6
## [1] 7
## [1] 8
## [1] 9
## [1] 10
```

Looping in R

- index defines 'counter' or variable that varies
- 'values' define which values index takes on

```
for (index in c("cat","hat","worm")){
  print(index)
}
```

```
## [1] "cat"
## [1] "hat"
## [1] "worm"
```

Creating New Variables

Conditional Execution with If then, If then else

Want to code a wind category variable

```
status<-vector() #initialize vector to save results

for (i in 1:nrow(airquality)){
  if(airquality$Wind[i] >= 15){
    status[i] <- "HighWind"
  } else if (airquality$Wind[i] >= 10){
    status[i] <- "Windy"
  } else if (airquality$Wind[i] >= 6){
    status[i] <- "LightWind"
  } else if (airquality$Wind[i] >= 0){
    status[i] <- "Calm"
  } else {
    status[i] <- "Error"
  }
}
```

Creating New Variables

Conditional Execution with If then, If then else

```
airquality$status <- status
airquality$status
```

```
##   [1] "LightWind" "LightWind" "Windy"     "Windy"     "Windy"     "Windy"
##   [7] "LightWind" "Windy"     "HighWind"  "LightWind" "LightWind" "LightWind"
##  [13] "LightWind" "Windy"     "Windy"     "Windy"     "Windy"     "HighWind"
##  [19] "Windy"     "LightWind" "LightWind" "HighWind"  "LightWind" "Windy"
##  [25] "HighWind"  "Windy"     "LightWind" "Windy"     "Windy"     "Calm"
##  [31] "LightWind" "LightWind" "LightWind" "HighWind"  "LightWind" "LightWind"
##  [37] "Windy"     "LightWind" "LightWind" "Windy"     "Windy"     "Windy"
##  [43] "LightWind" "LightWind" "Windy"     "Windy"     "Windy"     "HighWind"
##  [49] "LightWind" "Windy"     "Windy"     "LightWind" "Calm"      "Calm"
##  [55] "LightWind" "LightWind" "LightWind" "Windy"     "Windy"     "Windy"
##  [61] "LightWind" "Calm"      "LightWind" "LightWind" "Windy"     "Calm"
##  [67] "Windy"     "Calm"      "LightWind" "Calm"      "LightWind" "LightWind"
##  [73] "Windy"     "Windy"     "Windy"     "Windy"     "LightWind" "Windy"
##  [79] "LightWind" "Calm"      "Windy"     "LightWind" "LightWind" "Windy"
##  [85] "LightWind" "LightWind" "LightWind" "Windy"     "LightWind" "LightWind"
##  [91] "LightWind" "LightWind" "LightWind" "Windy"     "LightWind" "LightWind"
##  [97] "LightWind" "Calm"      "Calm"      "Windy"     "LightWind" "LightWind"
## [103] "Windy"     "Windy"     "Windy"     "LightWind" "Windy"     "Windy"
## [109] "LightWind" "LightWind" "Windy"     "Windy"     "HighWind"  "Windy"
## [115] "Windy"     "LightWind" "Calm"      "LightWind" "Calm"      "LightWind"
## [121] "Calm"      "LightWind" "LightWind" "LightWind" "Calm"      "Calm"
## [127] "Calm"      "LightWind" "HighWind"  "Windy"     "Windy"     "Windy"
## [133] "LightWind" "Windy"     "HighWind"  "LightWind" "Windy"     "Windy"
## [139] "LightWind" "Windy"     "Windy"     "Windy"     "LightWind" "Windy"
## [145] "LightWind" "Windy"     "Windy"     "HighWind"  "LightWind" "Windy"
## [151] "Windy"     "LightWind" "Windy"
```

Looping in R

Other things to know

- `break` kicks you out of the loop

```
for (i in 1:5){
    if (i == 3){
      break
      }
  print(i)
}
```

```
## [1] 1
## [1] 2
```

Looping in R

Other things to know

- `next` jumps to the next iteration of the loop

```
for (i in 1:5){
    if (i == 3){
      next
    }
  print(i)
}
```

```
## [1] 1
## [1] 2
## [1] 4
## [1] 5
```

Looping in R

Other things to know

- `while` loop similar

```
while(condition) {
    expression to evaluate
  modify condition?
}
```

Looping in R

Other things to know

For loops inefficient in R

- R interpreted language
- Must figure out how to evaluate code at each iteration of loop
- Slows it down

Looping in R

Other things to know

For loops inefficient in R

- R interpreted language
- Must figure out how to evaluate code at each iteration of loop
- Slows it down

Vectorized functions much faster!

- Vectorized function: works on entire vector at once
- Avoids costly computation time

Creating New Variables

Vectorized ifelse

- `ifelse()` is a vectorized version of `if then else`
- Syntax

```
ifelse(vector_condition, if_true_do_this, if_false_do_this)
```

Creating New Variables

Vectorized if else

```
ifelse(airquality$Wind >= 15, "HighWind",
          ifelse(airquality$Wind >= 10, "Windy",
                 ifelse(airquality$Wind >= 6, "LightWind", "Calm")))
```

```
##   [1] "LightWind" "LightWind" "Windy"     "Windy"     "Windy"     "Windy"
##   [7] "LightWind" "Windy"     "HighWind"  "LightWind" "LightWind" "LightWind"
##  [13] "LightWind" "Windy"     "Windy"     "Windy"     "Windy"     "HighWind"
##  [19] "Windy"     "LightWind" "LightWind" "HighWind"  "LightWind" "Windy"
##  [25] "HighWind"  "Windy"     "LightWind" "Windy"     "Windy"     "Calm"
##  [31] "LightWind" "LightWind" "LightWind" "HighWind"  "LightWind" "LightWind"
##  [37] "Windy"     "LightWind" "LightWind" "Windy"     "Windy"     "Windy"
##  [43] "LightWind" "LightWind" "Windy"     "Windy"     "Windy"     "HighWind"
##  [49] "LightWind" "Windy"     "Windy"     "LightWind" "Calm"      "Calm"
##  [55] "LightWind" "LightWind" "LightWind" "Windy"     "Windy"     "Windy"
##  [61] "LightWind" "Calm"      "LightWind" "LightWind" "Windy"     "Calm"
##  [67] "Windy"     "Calm"      "LightWind" "Calm"      "LightWind" "LightWind"
##  [73] "Windy"     "Windy"     "Windy"     "Windy"     "LightWind" "Windy"
##  [79] "LightWind" "Calm"      "Windy"     "LightWind" "LightWind" "Windy"
##  [85] "LightWind" "LightWind" "LightWind" "Windy"     "LightWind" "LightWind"
##  [91] "LightWind" "LightWind" "LightWind" "Windy"     "LightWind" "LightWind"
##  [97] "LightWind" "Calm"      "Calm"      "Windy"     "LightWind" "LightWind"
## [103] "Windy"     "Windy"     "Windy"     "LightWind" "Windy"     "Windy"
## [109] "LightWind" "LightWind" "Windy"     "Windy"     "HighWind"  "Windy"
## [115] "Windy"     "LightWind" "Calm"      "LightWind" "Calm"      "LightWind"
## [121] "Calm"      "LightWind" "LightWind" "LightWind" "Calm"      "Calm"
## [127] "Calm"      "LightWind" "HighWind"  "Windy"     "Windy"     "Windy"
## [133] "LightWind" "Windy"     "HighWind"  "LightWind" "Windy"     "Windy"
## [139] "LightWind" "Windy"     "Windy"     "Windy"     "LightWind" "Windy"
## [145] "LightWind" "Windy"     "Windy"     "HighWind"  "LightWind" "Windy"
## [151] "Windy"     "LightWind" "Windy"
```

Creating New Variables

Vectorized if else

- Can use with `transmute()` or `mutate()`

```
mutate(airquality, status = ifelse(airquality$Wind >= 15, "HighWind",
                                   ifelse(airquality$Wind >= 10, "Windy",
                                          ifelse(airquality$Wind >= 6, "LightWind", "Calm")))
)
```

```
## # A tibble: 153 x 7
##   Ozone Solar.R  Wind  Temp Month   Day status
##   <int>   <int> <dbl> <int> <int> <int> <chr>
## 1    41     190   7.4    67     5     1 LightWind
## 2    36     118   8      72     5     2 LightWind
## 3    12     149  12.6    74     5     3 Windy
## 4    18     313  11.5    62     5     4 Windy
## 5    NA      NA  14.3    56     5     5 Windy
## # … with 148 more rows
```

Creating New Variables

- Compare speed of for loop and vectorized version
- `microbenchmark` package runs code repeatedly, returns summary stats

```
#install if needed
library(microbenchmark)
```

Creating New Variables

- Compare speed of for loop and vectorized version

```
loopTime <- microbenchmark(
  for (i in 1:nrow(airquality)){
    if(airquality$Wind[i] >= 15){
      status[i] <- "HighWind"
    } else if (airquality$Wind[i] >= 10){
      status[i] <- "Windy"
    } else if (airquality$Wind[i] >= 6){
      status[i] <- "LightWind"
    } else if (airquality$Wind[i] >= 0){
      status[i] <- "Calm"
    } else{
      status[i] <- "Error"
    }
  }
, unit = "us")
```

Creating New Variables

- Compare speed of for loop and vectorized version

```
vectorTime <- microbenchmark(
  ifelse(airquality$Wind >= 15, "HighWind",
           ifelse(airquality$Wind >= 10, "Windy",
                  ifelse(airquality$Wind >= 6, "LightWind", "Calm")))
, unit = "us")
```

Creating New Variables

```
loopTime
```

```
## Unit: microseconds
##
##  for (i in 1:nrow(airquality)) {     if (airquality$Wind[i] >= 15) {         status[i] <- "H
##       min       lq     mean   median       uq      max neval
##  8873.587 9452.457 10925.81 10167.46 12013.24 17050.91   100
```

```
vectorTime
```

```
## Unit: microseconds
##
##  ifelse(airquality$Wind >= 15, "HighWind", ifelse(airquality$Wind >=      10, "Windy", ifels
##     min      lq     mean median      uq     max neval
##  80.929 82.7975 94.15377 86.469 90.2435 209.586   100
```

Creating New Variables Recap!

- Add new column with `$`
- `mutate()` - add newly created **column(s)** to current data frame
- `transmute()` - create new data frame with created variable(s)
    - Can use with window functions
    - Use `ifelse()` to do conditional creation
    - Note: `cut()` can be used to categorize a numeric variable too!