**TOOL**

# Manipulating Data With the Tidyverse

The tidyverse allows you to manipulate your data set by filtering rows, selecting columns, creating new variables, and calculating summary statistics. Use this tool as a guide to manipulate data with the tidyverse.

For each of the examples below, we will begin with the Credit data set. To set the data up for use with this tool, use the following code:

```
library(tidyverse)

Credit <- read.csv("credit_cards.csv")
```

| ID <int> | Income <dbl> | Limit <int> | Rating <int> | Cards <int> | Age <int> | Education <int> | Student <fctr> | Married <fctr> | Balance <int> |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 14.891 | 3606 | 283 | 2 | 34 | 11 | No | Yes | 333 |
| 2 | 106.025 | 6645 | 483 | 3 | 82 | 15 | Yes | Yes | 903 |
| 3 | 104.593 | 7075 | 514 | 4 | 71 | 11 | No | No | 580 |
| 4 | 148.924 | 9504 | 681 | 3 | 36 | 11 | No | No | 964 |
| 5 | 55.882 | 4897 | 357 | 2 | 68 | 16 | No | Yes | 331 |
| 6 | 80.180 | 8047 | 569 | 4 | 77 | 10 | No | No | 1151 |
| 7 | 20.996 | 3388 | 259 | 2 | 37 | 12 | No | No | 203 |

## filter()

Use filter() to keep only observations (rows) that satisfy certain criteria.

This function takes two inputs:

1) The name of the data set

2) The criteria by which we're filtering

### Filter based on a single criterion:

```
filter(Credit, Student == "Yes")
```

### Filter based on multiple criteria, often with the AND operator (&) or the OR operator (|):

```
filter(Credit, Student == "Yes" & Balance > 0)
```

| ID <int> | Income <dbl> | Limit <int> | Rating <int> | Cards <int> | Age <int> | Education <int> | Student <fctr> | Married <fctr> | Balance <int> |
|---|---|---|---|---|---|---|---|---|---|
| 2 | 106.025 | 6645 | 483 | 3 | 82 | 15 | Yes | Yes | 903 |
| 10 | 71.061 | 6819 | 491 | 3 | 41 | 19 | Yes | Yes | 1350 |
| 27 | 42.471 | 3625 | 289 | 6 | 44 | 12 | Yes | No | 654 |
| 36 | 23.350 | 2558 | 220 | 3 | 49 | 12 | Yes | No | 419 |
| 42 | 113.659 | 7659 | 538 | 2 | 66 | 15 | Yes | Yes | 1155 |
| 47 | 19.531 | 5043 | 376 | 2 | 64 | 16 | Yes | Yes | 1241 |
| 48 | 44.646 | 4431 | 320 | 2 | 49 | 15 | Yes | Yes | 797 |


© 2021 Cornell University

## Beginning data set:

```
library(tidyverse)

Credit <- read.csv("credit_cards.csv")
```

| ID <int> | Income <dbl> | Limit <int> | Rating <int> | Cards <int> | Age <int> | Education <int> | Student <fctr> | Married <fctr> | Balance <int> |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 14.891 | 3606 | 283 | 2 | 34 | 11 | No | Yes | 333 |
| 2 | 106.025 | 6645 | 483 | 3 | 82 | 15 | Yes | Yes | 903 |
| 3 | 104.593 | 7075 | 514 | 4 | 71 | 11 | No | No | 580 |
| 4 | 148.924 | 9504 | 681 | 3 | 36 | 11 | No | No | 964 |
| 5 | 55.882 | 4897 | 357 | 2 | 68 | 16 | No | Yes | 331 |
| 6 | 80.180 | 8047 | 569 | 4 | 77 | 10 | No | No | 1151 |
| 7 | 20.996 | 3388 | 259 | 2 | 37 | 12 | No | No | 203 |

## select()

The select() function keeps only certain variables (columns) from a data set. This can be done by either specifying the columns to *retain* or to *discard*.

There are two inputs to the `select()` function:

1. The name of the data set
2. The name(s) of the variable(s) that you are keeping/discarding

### To specify which variables to keep, put their names in a vector:

```
select(Credit, c(ID, Income))
```

| ID <int> | Income <dbl> |
|---|---|
| 1 | 14.891 |
| 2 | 106.025 |
| 3 | 104.593 |
| 4 | 148.924 |
| 5 | 55.882 |
| 6 | 80.180 |
| 7 | 20.996 |

### To specify which variables to remove, put their names in a vector and include a minus sign in front of the vector:

```
select(Credit, -c(ID, Income))
```

| Limit <int> | Rating <int> | Cards <int> | Age <int> | Education <int> | Student <fctr> | Married <fctr> | Balance <int> |
|---|---|---|---|---|---|---|---|
| 3606 | 283 | 2 | 34 | 11 | No | Yes | 333 |
| 6645 | 483 | 3 | 82 | 15 | Yes | Yes | 903 |
| 7075 | 514 | 4 | 71 | 11 | No | No | 580 |
| 9504 | 681 | 3 | 36 | 11 | No | No | 964 |
| 4897 | 357 | 2 | 68 | 16 | No | Yes | 331 |
| 8047 | 569 | 4 | 77 | 10 | No | No | 1151 |
| 3388 | 259 | 2 | 37 | 12 | No | No | 203 |


© 2021 Cornell University

## Beginning data set:

```
library(tidyverse)

Credit <- read.csv("credit_cards.csv")
```

| ID <int> | Income <dbl> | Limit <int> | Rating <int> | Cards <int> | Age <int> | Education <int> | Student <fctr> | Married <fctr> | Balance <int> |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 14.891 | 3606 | 283 | 2 | 34 | 11 | No | Yes | 333 |
| 2 | 106.025 | 6645 | 483 | 3 | 82 | 15 | Yes | Yes | 903 |
| 3 | 104.593 | 7075 | 514 | 4 | 71 | 11 | No | No | 580 |
| 4 | 148.924 | 9504 | 681 | 3 | 36 | 11 | No | No | 964 |
| 5 | 55.882 | 4897 | 357 | 2 | 68 | 16 | No | Yes | 331 |
| 6 | 80.180 | 8047 | 569 | 4 | 77 | 10 | No | No | 1151 |
| 7 | 20.996 | 3388 | 259 | 2 | 37 | 12 | No | No | 203 |

## mutate()

The `mutate()` function is used to create new variables (i.e., to add new columns to the data set) by manipulating the current ones.

There are two inputs:

1. The name of the data set
2. The variable you're creating: its name, followed by an equals sign, followed by the definition of the new variable

### Create a new variable called SqrtLimit that is the square root of the Limit variable:

```
mutate(Credit, SqrtLimit = sqrt(Limit))
```

| ID <int> | Income <dbl> | Limit <int> | Rating <int> | Cards <int> | Age <int> | Education <int> | Student <fctr> | Married <fctr> | Balance <int> | SqrtLimit <dbl> |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 14.891 | 3606 | 283 | 2 | 34 | 11 | No | Yes | 333 | 60.04998 |
| 2 | 106.025 | 6645 | 483 | 3 | 82 | 15 | Yes | Yes | 903 | 81.51687 |
| 3 | 104.593 | 7075 | 514 | 4 | 71 | 11 | No | No | 580 | 84.11302 |
| 4 | 148.924 | 9504 | 681 | 3 | 36 | 11 | No | No | 964 | 97.48846 |
| 5 | 55.882 | 4897 | 357 | 2 | 68 | 16 | No | Yes | 331 | 69.97857 |
| 6 | 80.180 | 8047 | 569 | 4 | 77 | 10 | No | No | 1151 | 89.70507 |
| 7 | 20.996 | 3388 | 259 | 2 | 37 | 12 | No | No | 203 | 58.20653 |

### Create more than one new variable at once by separating the variables with commas:

```
# HigherEd will be TRUE if the value of the
# Education variable is greater than 13 and
# FALSE otherwise

mutate(Credit, HigherEd = Education > 13,
SqrtLimit = sqrt(Limit))
```

| ID <int> | Income <dbl> | Limit <int> | Rating <int> | Cards <int> | Age <int> | Education <int> | Student <fctr> | Married <fctr> | Balance <int> | HigherEd <lgl> | SqrtLimit <dbl> |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 14.891 | 3606 | 283 | 2 | 34 | 11 | No | Yes | 333 | FALSE | 60.04998 |
| 2 | 106.025 | 6645 | 483 | 3 | 82 | 15 | Yes | Yes | 903 | TRUE | 81.51687 |
| 3 | 104.593 | 7075 | 514 | 4 | 71 | 11 | No | No | 580 | FALSE | 84.11302 |
| 4 | 148.924 | 9504 | 681 | 3 | 36 | 11 | No | No | 964 | FALSE | 97.48846 |
| 5 | 55.882 | 4897 | 357 | 2 | 68 | 16 | No | Yes | 331 | TRUE | 69.97857 |
| 6 | 80.180 | 8047 | 569 | 4 | 77 | 10 | No | No | 1151 | FALSE | 89.70507 |
| 7 | 20.996 | 3388 | 259 | 2 | 37 | 12 | No | No | 203 | FALSE | 58.20653 |


© 2021 Cornell University

## Beginning data set:

```
library(tidyverse)

Credit <- read.csv("credit_cards.csv")
```

| ID <int> | Income <dbl> | Limit <int> | Rating <int> | Cards <int> | Age <int> | Education <int> | Student <fctr> | Married <fctr> | Balance <int> |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 14.891 | 3606 | 283 | 2 | 34 | 11 | No | Yes | 333 |
| 2 | 106.025 | 6645 | 483 | 3 | 82 | 15 | Yes | Yes | 903 |
| 3 | 104.593 | 7075 | 514 | 4 | 71 | 11 | No | No | 580 |
| 4 | 148.924 | 9504 | 681 | 3 | 36 | 11 | No | No | 964 |
| 5 | 55.882 | 4897 | 357 | 2 | 68 | 16 | No | Yes | 331 |
| 6 | 80.180 | 8047 | 569 | 4 | 77 | 10 | No | No | 1151 |
| 7 | 20.996 | 3388 | 259 | 2 | 37 | 12 | No | No | 203 |

## group_by() summarise()

The `group_by()` function separates the data set into groups, after which the `summarise()` function is used to calculate summary statistics for each group.

The input(s) to the `group_by()` function are the name(s) of the variables to group by. The input to the `summarise()` function is the name of the summary statistic you're creating, followed by an equal sign and the definition of the summary statistic. If you're calculating more than one summary statistic, separate them with commas.

```
# Group by the Student and Married variables and
# calculate the maximum credit limit and income
# range for each group:

Credit %>%
  group_by(Student, Married) %>%
  summarise(MaxLimit = max(Limit),
    IncomeRange = max(Income) - min(Income))
```

| Student <fctr> | Married <fctr> | MaxLimit <int> | IncomeRange <dbl> |
|---|---|---|---|
| No | No | 10748 | 149.489 |
| No | Yes | 13913 | 176.280 |
| Yes | No | 9560 | 109.978 |
| Yes | Yes | 9310 | 169.752 |

4 rows

**Notice that here we use the pipe operator, %>%, which can be read as "and then." The pipe operator takes a data frame and sends it as input to the function that follows.**


© 2021 Cornell University