# STAT 451 - Visualizing Data - Autumn 2025

Ariane Ducellier

09/30/2025

## Tutorial Tidyverse part 1

Today, we are going to review useful R functions for reading and exploring data sets. The objective is to make sure that you have all the tools you need to explore your data set before starting to visualize it. For more on data exploration with R, I recommend the two books:

Data Wrangling with R Load, Explore, Transform and Visualize Data for Modeling with Tidyverse Libraries. Santos, Gustavo R.
2023; Birmingham : Packt Publishing, Limited

R for data science : import, tidy, transform, visualize, and model data Wickham, Hadley, author.
2023; Sebastopol, CA : O'Reilly Media, Inc.

We will need the following R libraries:

```
library(tidyverse)
library(skimr)
```

### 1. Read data (csv format)

Read csv files with basic R.

```
df <- read.csv("../../data/gapminder-data.csv")
print(class(df))
```

```
## [1] "data.frame"
```

Read csv files with tidyverse.

```
df_t <- read_csv("../../data/gapminder-data.csv")
```

```
## New names:
## Rows: 1512 Columns: 10
## -- Column specification
## -------------------------------------------------------- Delimiter: "," chr
## (1): Country dbl (9): ...1, Year, gdp_per_capita,
## Electricity_consumption_per_capita, und...
## i Use `spec()` to retrieve the full column specification for this data. i
## Specify the column types or set `show_col_types = FALSE` to quiet this message.
## * `` -> `...1`
```

```
print(class(df_t))
```

```
## [1] "spec_tbl_df" "tbl_df"      "tbl"         "data.frame"
```

Read select columns, specify the encoding of the missing values.

```
df_t_sub <- read_csv("../../data/gapminder-data.csv",
  col_select=c("Country", "Year", "gdp_per_capita"),
  na=c("", "NA"))
```

```
## New names:
## Rows: 1512 Columns: 3
## -- Column specification
## -------------------------------------------------------- Delimiter: "," chr
## (1): Country dbl (2): Year, gdp_per_capita
## i Use `spec()` to retrieve the full column specification for this data. i
## Specify the column types or set `show_col_types = FALSE` to quiet this message.
## * `` -> `...1`
```

## 2. Get basic information on the data.

Reads the first part of a data frame.

```
df_t <- read_csv("../../data/gapminder-data.csv")
```

```
## New names:
## Rows: 1512 Columns: 10
## -- Column specification
## -------------------------------------------------------- Delimiter: "," chr
## (1): Country dbl (9): ...1, Year, gdp_per_capita,
## Electricity_consumption_per_capita, und...
## i Use `spec()` to retrieve the full column specification for this data. i
## Specify the column types or set `show_col_types = FALSE` to quiet this message.
## * `` -> `...1`
```

```
head(df_t, 3)
```

```
## # A tibble: 3 x 10
##    ...1 Country  Year gdp_per_capita Electricity_consumption_p~1 under5mortality
##   <dbl> <chr>   <dbl>          <dbl>                       <dbl>           <dbl>
## 1     0 Brazil   1800           1109                          NA            417.
## 2     1 Brazil   1801           1109                          NA            417.
## 3     2 Brazil   1802           1109                          NA            417.
## # i abbreviated name: 1: Electricity_consumption_per_capita
## # i 4 more variables: Poverty <dbl>, BMI_male <dbl>, BMI_female <dbl>,
## #   population <dbl>
```

Reads the last part of a data frame.

```
tail(df_t, 3)
```

```
## # A tibble: 3 x 10
##    ...1 Country       Year gdp_per_capita Electricity_consumpt~1 under5mortality
##   <dbl> <chr>        <dbl>          <dbl>                  <dbl>           <dbl>
## 1  1509 United Stat~  2013          51282                     NA             6.9
## 2  1510 United Stat~  2014          52118                     NA             6.7
## 3  1511 United Stat~  2015          53354                     NA             6.5
## # i abbreviated name: 1: Electricity_consumption_per_capita
## # i 4 more variables: Poverty <dbl>, BMI_male <dbl>, BMI_female <dbl>,
## #   population <dbl>
```

Gets column specifications of a tibble.

```
spec(df_t)
```

```
## cols(
##   ...1 = col_double(),
##   Country = col_character(),
##   Year = col_double(),
##   gdp_per_capita = col_double(),
##   Electricity_consumption_per_capita = col_double(),
##   under5mortality = col_double(),
##   Poverty = col_double(),
##   BMI_male = col_double(),
##   BMI_female = col_double(),
##   population = col_double()
## )
```

Prints the data: number of rows and columns, type of columns, and first rows.

```
glimpse(df_t)
```

```
## Rows: 1,512
## Columns: 10
## $ ...1                                <dbl> 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 1~
## $ Country                             <chr> "Brazil", "Brazil", "Brazil", "Braz~
## $ Year                                <dbl> 1800, 1801, 1802, 1803, 1804, 1805,~
## $ gdp_per_capita                      <dbl> 1109, 1109, 1109, 1109, 1109, 1110,~
## $ Electricity_consumption_per_capita <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA,~
## $ under5mortality                     <dbl> 417.44, 417.44, 417.44, 417.44, 417~
## $ Poverty                             <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA,~
## $ BMI_male                            <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA,~
## $ BMI_female                          <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA,~
## $ population                          <dbl> 3639636, NA, NA, NA, NA, NA, NA, NA~
```

Returns descriptive statistics on each column of a data frame.

```
summary(df_t)
```

```
##       ...1          Country               Year      gdp_per_capita
##  Min.   :   0.0   Length:1512        Min.   :1800   Min.   :  529
##  1st Qu.: 377.8   Class :character   1st Qu.:1854   1st Qu.: 1124
##  Median : 755.5   Mode  :character   Median :1908   Median : 2496
##  Mean   : 755.5                      Mean   :1908   Mean   : 7234
##  3rd Qu.:1133.2                      3rd Qu.:1961   3rd Qu.: 8219
##  Max.   :1511.0                      Max.   :2015   Max.   :53354
##
##  Electricity_consumption_per_capita under5mortality     Poverty
##  Min.   :   97.78                   Min.   :  2.70   Min.   : 0.000
##  1st Qu.: 1062.24                   1st Qu.: 77.59   1st Qu.: 0.920
##  Median : 4310.62                   Median :306.66   Median : 9.385
##  Mean   : 4386.74                   Mean   :260.02   Mean   :15.338
##  3rd Qu.: 6495.64                   3rd Qu.:417.44   3rd Qu.:15.960
##  Max.   :13704.58                   Max.   :539.16   Max.   :84.270
##  NA's   :1181                                        NA's   :1440
##     BMI_male       BMI_female      population
##  Min.   :20.62   Min.   :20.48   Min.   :3.640e+06
##  1st Qu.:22.22   1st Qu.:21.90   1st Qu.:6.740e+07
##  Median :24.04   Median :24.57   Median :1.250e+08
```

```
##  Mean   :24.16   Mean   :23.91   Mean   :2.996e+08
##  3rd Qu.:26.17   3rd Qu.:25.56   3rd Qu.:3.767e+08
##  Max.   :28.46   Max.   :28.34   Max.   :1.376e+09
##  NA's   :1309    NA's   :1309    NA's   :945
```

Provides a broad overview of a data frame, handles data of all types, dispatching a different set of summary functions based on the types of columns in the data frame.

```
# Comment this line for knitting with pdf
#skim(df_t)
```

## 3. The pipe operator

The magrittr package provides the %>% operator as a shortcut for modifying an object in place. It is installed by default when installing tidyverse.

```
data(iris)
df_iris <- iris %>%
  group_by(Species) %>%
  summarize_if(is.numeric, mean) %>%
  ungroup() %>%
  gather(measure, value, -Species) %>%
  arrange(value)
```

is the same as:

```
data(iris)
df_iris_alt <- group_by(iris, Species)
df_iris_alt <- summarize_if(df_iris_alt, is.numeric, mean)
df_iris_alt <- ungroup(df_iris_alt)
df_iris_alt <- gather(df_iris_alt, measure, value, -Species)
df_iris_alt <- arrange(df_iris_alt, value)
```

## 4. Transform the data

We will review some basic transformations.

```
header <- c("age", "workclass", "fnlwgt", "education",
  "education_num", "marital_status", "occupation",
  "relationship", "race", "sex", "capital_gain",
  "capital_loss", "hours_per_week", "native_country", "target")
df <- read_csv("https://archive.ics.uci.edu/ml/machine-learning-databases/adult/adult.data",
  col_names=header, trim_ws=TRUE)
```

```
## Rows: 32561 Columns: 15
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (9): workclass, education, marital_status, occupation, relationship, rac...
## dbl (6): age, fnlwgt, education_num, capital_gain, capital_loss, hours_per_week
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

### 4.1 Slicing

Cuts unwanted parts of the data set.

```r
df %>% slice_head(n=5)
```

```
## # A tibble: 5 x 15
##     age workclass       fnlwgt education education_num marital_status occupation
##   <dbl> <chr>            <dbl> <chr>             <dbl> <chr>          <chr>
## 1    39 State-gov        77516 Bachelors            13 Never-married  Adm-cleri~
## 2    50 Self-emp-not-i~  83311 Bachelors            13 Married-civ-s~ Exec-mana~
## 3    38 Private         215646 HS-grad               9 Divorced       Handlers-~
## 4    53 Private         234721 11th                  7 Married-civ-s~ Handlers-~
## 5    28 Private         338409 Bachelors            13 Married-civ-s~ Prof-spec~
## # i 8 more variables: relationship <chr>, race <chr>, sex <chr>,
## #   capital_gain <dbl>, capital_loss <dbl>, hours_per_week <dbl>,
## #   native_country <chr>, target <chr>
```

```r
df %>% slice_tail(n=5)
```

```
## # A tibble: 5 x 15
##     age workclass    fnlwgt education  education_num marital_status   occupation
##   <dbl> <chr>         <dbl> <chr>              <dbl> <chr>            <chr>
## 1    27 Private      257302 Assoc-acdm            12 Married-civ-spo~ Tech-supp~
## 2    40 Private      154374 HS-grad                9 Married-civ-spo~ Machine-o~
## 3    58 Private      151910 HS-grad                9 Widowed          Adm-cleri~
## 4    22 Private      201490 HS-grad                9 Never-married    Adm-cleri~
## 5    52 Self-emp-inc 287927 HS-grad                9 Married-civ-spo~ Exec-mana~
## # i 8 more variables: relationship <chr>, race <chr>, sex <chr>,
## #   capital_gain <dbl>, capital_loss <dbl>, hours_per_week <dbl>,
## #   native_country <chr>, target <chr>
```

```r
df %>% slice_min(age, prop=0.10)
```

```
## # A tibble: 3,895 x 15
##      age workclass fnlwgt education education_num marital_status occupation
##    <dbl> <chr>      <dbl> <chr>              <dbl> <chr>          <chr>
##  1    17 ?         304873 10th                   6 Never-married  ?
##  2    17 Private    65368 11th                   7 Never-married  Sales
##  3    17 Private   245918 11th                   7 Never-married  Other-service
##  4    17 Private   191260 9th                    5 Never-married  Other-service
##  5    17 Private   270942 5th-6th                3 Never-married  Other-service
##  6    17 Private    89821 11th                   7 Never-married  Other-service
##  7    17 Private   175024 11th                   7 Never-married  Handlers-clean~
##  8    17 ?         202521 11th                   7 Never-married  ?
##  9    17 ?         258872 11th                   7 Never-married  ?
## 10    17 Private   211870 9th                    5 Never-married  Other-service
## # i 3,885 more rows
## # i 8 more variables: relationship <chr>, race <chr>, sex <chr>,
## #   capital_gain <dbl>, capital_loss <dbl>, hours_per_week <dbl>,
## #   native_country <chr>, target <chr>
```

```r
df %>% slice_max(age, prop=0.30)
```

```
## # A tibble: 10,361 x 15
##      age workclass fnlwgt education    education_num marital_status   occupation
##    <dbl> <chr>      <dbl> <chr>                <dbl> <chr>            <chr>
##  1    90 Private    51744 HS-grad                  9 Never-married    Other-ser~
##  2    90 Private   137018 HS-grad                  9 Never-married    Other-ser~
##  3    90 Private   221832 Bachelors               13 Married-civ-spo~ Exec-mana~
```

```
##  4    90 Private    52386 Some-college            10 Never-married    Other-ser~
##  5    90 Private   171956 Some-college            10 Separated        Adm-cleri~
##  6    90 Private   313986 11th                     7 Never-married    Handlers-~
##  7    90 ?         256514 Bachelors               13 Widowed          ?
##  8    90 Private    52386 Some-college            10 Never-married    Other-ser~
##  9    90 Private   141758 9th                      5 Never-married    Adm-cleri~
## 10    90 Local-gov 227796 Masters                 14 Married-civ-spo~ Exec-mana~
## # i 10,351 more rows
## # i 8 more variables: relationship <chr>, race <chr>, sex <chr>,
## #   capital_gain <dbl>, capital_loss <dbl>, hours_per_week <dbl>,
## #   native_country <chr>, target <chr>
```

```
df %>% slice_sample(n=10, replace=TRUE)
```

```
## # A tibble: 10 x 15
##      age workclass fnlwgt education    education_num marital_status   occupation
##    <dbl> <chr>      <dbl> <chr>                <dbl> <chr>            <chr>
##  1    20 Private   157541 Some-college            10 Never-married    Other-ser~
##  2    30 Private   182177 HS-grad                  9 Never-married    Other-ser~
##  3    49 Private   134797 Some-college            10 Divorced         Adm-cleri~
##  4    25 ?          12285 Some-college            10 Never-married    ?
##  5    33 Private   223212 HS-grad                  9 Divorced         Craft-rep~
##  6    17 ?          41643 11th                     7 Never-married    ?
##  7    50 Private   302372 Bachelors               13 Married-civ-spo~ Prof-spec~
##  8    35 Private   267891 HS-grad                  9 Married-civ-spo~ Other-ser~
##  9    41 Private    56795 Masters                 14 Married-civ-spo~ Prof-spec~
## 10    37 Private   122493 HS-grad                  9 Married-civ-spo~ Transport~
## # i 8 more variables: relationship <chr>, race <chr>, sex <chr>,
## #   capital_gain <dbl>, capital_loss <dbl>, hours_per_week <dbl>,
## #   native_country <chr>, target <chr>
```

### 4.2 Filtering

Apply a condition to one of the variables to filter unwanted rows of the data.

```
df %>% filter(age > 30)
```

```
## # A tibble: 21,989 x 15
##      age workclass      fnlwgt education education_num marital_status occupation
##    <dbl> <chr>           <dbl> <chr>             <dbl> <chr>          <chr>
##  1    39 State-gov       77516 Bachelors            13 Never-married  Adm-cleri~
##  2    50 Self-emp-not-~  83311 Bachelors            13 Married-civ-s~ Exec-mana~
##  3    38 Private        215646 HS-grad               9 Divorced       Handlers-~
##  4    53 Private        234721 11th                  7 Married-civ-s~ Handlers-~
##  5    37 Private        284582 Masters              14 Married-civ-s~ Exec-mana~
##  6    49 Private        160187 9th                   5 Married-spous~ Other-ser~
##  7    52 Self-emp-not-~ 209642 HS-grad               9 Married-civ-s~ Exec-mana~
##  8    31 Private         45781 Masters              14 Never-married  Prof-spec~
##  9    42 Private        159449 Bachelors            13 Married-civ-s~ Exec-mana~
## 10    37 Private        280464 Some-col~            10 Married-civ-s~ Exec-mana~
## # i 21,979 more rows
## # i 8 more variables: relationship <chr>, race <chr>, sex <chr>,
## #   capital_gain <dbl>, capital_loss <dbl>, hours_per_week <dbl>,
## #   native_country <chr>, target <chr>
```

### 4.3 Selecting

Select specific columns.

```
df %>% select(marital_status, age)
```

```
## # A tibble: 32,561 x 2
##    marital_status          age
##    <chr>                 <dbl>
##  1 Never-married            39
##  2 Married-civ-spouse       50
##  3 Divorced                 38
##  4 Married-civ-spouse       53
##  5 Married-civ-spouse       28
##  6 Married-civ-spouse       37
##  7 Married-spouse-absent    49
##  8 Married-civ-spouse       52
##  9 Never-married            31
## 10 Married-civ-spouse       42
## # i 32,551 more rows
```

### 4.4 Unique values

Get unique entries for categorical variables.

```
df %>% distinct(sex)
```

```
## # A tibble: 2 x 1
##   sex
##   <chr>
## 1 Male
## 2 Female
```

### 4.5 Grouping

Group by column and summarize.

```
df %>%
  group_by(workclass) %>%
  summarize(age_avg=mean(age))
```

```
## # A tibble: 9 x 2
##   workclass        age_avg
##   <chr>              <dbl>
## 1 ?                   41.0
## 2 Federal-gov         42.6
## 3 Local-gov           41.8
## 4 Never-worked        20.6
## 5 Private             36.8
## 6 Self-emp-inc        46.0
## 7 Self-emp-not-inc    45.0
## 8 State-gov           39.4
## 9 Without-pay         47.8
```

### 4.6 Summarizing

The summary may be: - counting observations - counting available observations (i.e. not NA) - getting first or last value - compute statistics on each group (mean, standard deviation, quantile)

```
df %>% group_by(workclass) %>% summarize(n())
```

```
## # A tibble: 9 x 2
##   workclass        `n()`
##   <chr>            <int>
## 1 ?                 1836
## 2 Federal-gov        960
## 3 Local-gov         2093
## 4 Never-worked         7
## 5 Private          22696
## 6 Self-emp-inc      1116
## 7 Self-emp-not-inc  2541
## 8 State-gov         1298
## 9 Without-pay         14
```

```
df %>% summarize(sum(!is.na(workclass)))
```

```
## # A tibble: 1 x 1
##   `sum(!is.na(workclass))`
##                      <int>
## 1                    32561
```

```
df %>% group_by(workclass) %>% summarize(first(age))
```

```
## # A tibble: 9 x 2
##   workclass        `first(age)`
##   <chr>                   <dbl>
## 1 ?                          54
## 2 Federal-gov                35
## 3 Local-gov                  56
## 4 Never-worked               18
## 5 Private                    38
## 6 Self-emp-inc               47
## 7 Self-emp-not-inc           50
## 8 State-gov                  39
## 9 Without-pay                65
```

```
df %>% group_by(workclass) %>% summarize(sd(capital_gain))
```

```
## # A tibble: 9 x 2
##   workclass        `sd(capital_gain)`
##   <chr>                         <dbl>
## 1 ?                             5147.
## 2 Federal-gov                   4102.
## 3 Local-gov                     5775.
## 4 Never-worked                     0
## 5 Private                       6424.
## 6 Self-emp-inc                 17977.
## 7 Self-emp-not-inc             10986.
## 8 State-gov                     3778.
## 9 Without-pay                   1301.
```

```
df %>% group_by(workclass) %>% summarize(quantile(age, 0.5))
```

```
## # A tibble: 9 x 2
##   workclass        `quantile(age, 0.5)`
##   <chr>                           <dbl>
```

```
## 1 ?                                35
## 2 Federal-gov                      43
## 3 Local-gov                        41
## 4 Never-worked                     18
## 5 Private                          35
## 6 Self-emp-inc                     45
## 7 Self-emp-not-inc                 44
## 8 State-gov                        39
## 9 Without-pay                      57
```

We can also apply the summary over selected columns.

```
df %>% select(1, 3, 5, 11, 12, 13) %>% summarize(across(everything(), mean))
```

```
## # A tibble: 1 x 6
##     age  fnlwgt education_num capital_gain capital_loss hours_per_week
##   <dbl>   <dbl>         <dbl>        <dbl>        <dbl>          <dbl>
## 1  38.6 189778.          10.1        1078.         87.3           40.4
```

### 4.7 Arranging

To sort the data set.

```
df %>% arrange(native_country)
```

```
## # A tibble: 32,561 x 15
##      age workclass fnlwgt education    education_num marital_status   occupation
##    <dbl> <chr>      <dbl> <chr>                <dbl> <chr>            <chr>
##  1    40 Private   121772 Assoc-voc               11 Married-civ-spo~ Craft-rep~
##  2    31 Private    84154 Some-college            10 Married-civ-spo~ Sales
##  3    18 Private   226956 HS-grad                 9 Never-married    Other-ser~
##  4    32 ?         293936 7th-8th                 4 Married-spouse-~ ?
##  5    30 Private   117747 HS-grad                 9 Married-civ-spo~ Sales
##  6    56 Private   203580 HS-grad                 9 Married-civ-spo~ Adm-cleri~
##  7    45 Private   153141 HS-grad                 9 Married-civ-spo~ Adm-cleri~
##  8    39 ?         157443 Masters                14 Married-civ-spo~ ?
##  9    34 State-gov  98101 Bachelors              13 Married-civ-spo~ Exec-mana~
## 10    42 Private   197583 Assoc-acdm             12 Married-civ-spo~ Exec-mana~
## # i 32,551 more rows
## # i 8 more variables: relationship <chr>, race <chr>, sex <chr>,
## #   capital_gain <dbl>, capital_loss <dbl>, hours_per_week <dbl>,
## #   native_country <chr>, target <chr>
```

It is most useful to sort the data set after grouping and summarizing.

```
df %>%
  group_by(education) %>%
  summarize(count=n(),
            avg_net_gain = mean(capital_gain - capital_loss)) %>%
  arrange(desc(avg_net_gain))
```

```
## # A tibble: 16 x 3
##    education    count avg_net_gain
##    <chr>        <int>        <dbl>
##  1 Prof-school    576      10183.
##  2 Doctorate      413       4507.
##  3 Masters       1723       2396.
```

```
##  4 Bachelors     5355       1638.
##  5 Preschool       51        832.
##  6 Assoc-voc     1382        642.
##  7 Assoc-acdm    1067        547.
##  8 Some-college  7291        527.
##  9 HS-grad      10501        506.
## 10 10th           933        348.
## 11 9th            514        313.
## 12 12th           433        252.
## 13 7th-8th        646        168.
## 14 11th          1175        165.
## 15 5th-6th        333        108.
## 16 1st-4th        168         77.5
```

### 4.8 Separating and uniting

This is often useful to create new columns.

```
df %>% separate_wider_regex(cols=target, patterns=c(sign="(?:<=|>)", amount="\\d+", unit="."))
```

```
## # A tibble: 32,561 x 17
##      age workclass      fnlwgt education education_num marital_status occupation
##    <dbl> <chr>           <dbl> <chr>             <dbl> <chr>          <chr>
##  1    39 State-gov       77516 Bachelors            13 Never-married  Adm-cleri~
##  2    50 Self-emp-not-~  83311 Bachelors            13 Married-civ-s~ Exec-mana~
##  3    38 Private        215646 HS-grad               9 Divorced       Handlers-~
##  4    53 Private        234721 11th                  7 Married-civ-s~ Handlers-~
##  5    28 Private        338409 Bachelors            13 Married-civ-s~ Prof-spec~
##  6    37 Private        284582 Masters              14 Married-civ-s~ Exec-mana~
##  7    49 Private        160187 9th                   5 Married-spous~ Other-ser~
##  8    52 Self-emp-not-~ 209642 HS-grad               9 Married-civ-s~ Exec-mana~
##  9    31 Private         45781 Masters              14 Never-married  Prof-spec~
## 10    42 Private        159449 Bachelors            13 Married-civ-s~ Exec-mana~
## # i 32,551 more rows
## # i 10 more variables: relationship <chr>, race <chr>, sex <chr>,
## #   capital_gain <dbl>, capital_loss <dbl>, hours_per_week <dbl>,
## #   native_country <chr>, sign <chr>, amount <chr>, unit <chr>
```

```
df %>% unite(sex, race, age, col="description", sep="_", remove=FALSE)
```

```
## # A tibble: 32,561 x 16
##    description       age workclass fnlwgt education education_num marital_status
##    <chr>           <dbl> <chr>      <dbl> <chr>             <dbl> <chr>
##  1 Male_White_39      39 State-gov  77516 Bachelors            13 Never-married
##  2 Male_White_50      50 Self-emp~  83311 Bachelors            13 Married-civ-s~
##  3 Male_White_38      38 Private   215646 HS-grad               9 Divorced
##  4 Male_Black_53      53 Private   234721 11th                  7 Married-civ-s~
##  5 Female_Black_28    28 Private   338409 Bachelors            13 Married-civ-s~
##  6 Female_White_37    37 Private   284582 Masters              14 Married-civ-s~
##  7 Female_Black_49    49 Private   160187 9th                   5 Married-spous~
##  8 Male_White_52      52 Self-emp~ 209642 HS-grad               9 Married-civ-s~
##  9 Female_White_31    31 Private    45781 Masters              14 Never-married
## 10 Male_White_42      42 Private   159449 Bachelors            13 Married-civ-s~
## # i 32,551 more rows
## # i 9 more variables: occupation <chr>, relationship <chr>, race <chr>,
## #   sex <chr>, capital_gain <dbl>, capital_loss <dbl>, hours_per_week <dbl>,
```

```
## #   native_country <chr>, target <chr>
```

### 4.8 Mutate function

This function can be used for many purposes. You can create new variables:

```
df %>%
  mutate(total_gain = capital_gain - capital_loss,
                tax = ifelse(total_gain >= 15000,
                             total_gain * 0.1,
                             0)
  ) %>%
  arrange(desc(tax))
```

```
## # A tibble: 32,561 x 17
##      age workclass      fnlwgt education education_num marital_status occupation
##    <dbl> <chr>           <dbl> <chr>              <dbl> <chr>          <chr>
##  1    54 Self-emp-inc   166459 Prof-sch~             15 Married-civ-s~ Prof-spec~
##  2    52 Private        152234 HS-grad                9 Married-civ-s~ Exec-mana~
##  3    53 Self-emp-inc   263925 HS-grad                9 Married-civ-s~ Sales
##  4    52 Private        118025 Bachelors             13 Married-civ-s~ Exec-mana~
##  5    46 Private        370119 Prof-sch~             15 Married-civ-s~ Prof-spec~
##  6    43 Private        176270 Bachelors             13 Married-civ-s~ Exec-mana~
##  7    49 Private        159816 Bachelors             13 Married-civ-s~ Prof-spec~
##  8    50 Private        171338 Some-col~             10 Married-civ-s~ Exec-mana~
##  9    22 Self-emp-not-~ 202920 HS-grad                9 Never-married  Prof-spec~
## 10    43 Self-emp-inc   172826 Some-col~             10 Married-civ-s~ Sales
## # i 32,551 more rows
## # i 10 more variables: relationship <chr>, race <chr>, sex <chr>,
## #   capital_gain <dbl>, capital_loss <dbl>, hours_per_week <dbl>,
## #   native_country <chr>, target <chr>, total_gain <dbl>, tax <dbl>
```

We can use the mutate function to replace values (e.g. "?" by NA). Let see how many "?" we have in this data set.

```
for (variable in colnames(df)) {
  print (
    paste (variable, dim(drop_na(df[df[variable]=="?", variable]))[1])
  )
}
```

```
## [1] "age 0"
## [1] "workclass 1836"
## [1] "fnlwgt 0"
## [1] "education 0"
## [1] "education_num 0"
## [1] "marital_status 0"
## [1] "occupation 1843"
## [1] "relationship 0"
## [1] "race 0"
## [1] "sex 0"
## [1] "capital_gain 0"
## [1] "capital_loss 0"
## [1] "hours_per_week 0"
## [1] "native_country 583"
## [1] "target 0"
```

We now replace the "?" by NA in the three columns workclass, occupation and native_country.

```
df_replaced <- df %>%
  mutate(workclass = replace(workclass, workclass=="?", NA),
         occupation = replace(occupation, occupation=="?", NA),
         native_country =replace(native_country, native_country=="?", NA)
  )
```

Let us check that all the "?" have been replaced.

```
for (variable in colnames(df_replaced)) {
  print (
    paste (variable, dim(drop_na(df_replaced[df_replaced[variable]=="?", variable]))[1])
  )
}
```

```
## [1] "age 0"
## [1] "workclass 0"
## [1] "fnlwgt 0"
## [1] "education 0"
## [1] "education_num 0"
## [1] "marital_status 0"
## [1] "occupation 0"
## [1] "relationship 0"
## [1] "race 0"
## [1] "sex 0"
## [1] "capital_gain 0"
## [1] "capital_loss 0"
## [1] "hours_per_week 0"
## [1] "native_country 0"
## [1] "target 0"
```

An alternative is to use the na_if function.

```
df %>% mutate(workclass = na_if(workclass, "?"),
              occupation = na_if(occupation, "?"),
              native_country = na_if(native_country, "?"))
```

```
## # A tibble: 32,561 x 15
##      age workclass      fnlwgt education education_num marital_status occupation
##    <dbl> <chr>           <dbl> <chr>             <dbl> <chr>          <chr>
##  1    39 State-gov       77516 Bachelors            13 Never-married  Adm-cleri~
##  2    50 Self-emp-not-~  83311 Bachelors            13 Married-civ-s~ Exec-mana~
##  3    38 Private        215646 HS-grad               9 Divorced       Handlers-~
##  4    53 Private        234721 11th                  7 Married-civ-s~ Handlers-~
##  5    28 Private        338409 Bachelors            13 Married-civ-s~ Prof-spec~
##  6    37 Private        284582 Masters              14 Married-civ-s~ Exec-mana~
##  7    49 Private        160187 9th                   5 Married-spous~ Other-ser~
##  8    52 Self-emp-not-~ 209642 HS-grad               9 Married-civ-s~ Exec-mana~
##  9    31 Private         45781 Masters              14 Never-married  Prof-spec~
## 10    42 Private        159449 Bachelors            13 Married-civ-s~ Exec-mana~
## # i 32,551 more rows
## # i 8 more variables: relationship <chr>, race <chr>, sex <chr>,
## #   capital_gain <dbl>, capital_loss <dbl>, hours_per_week <dbl>,
## #   native_country <chr>, target <chr>
```

Here are additional examples of the use of the mutate function. We can create a new column based on the

values of another column:

```
df %>%
  mutate(over_under = case_match(target,
                                 "<=50K" ~ "under",
                                 ">50K" ~ "over")) %>%
  select(target, over_under)
```

```
## # A tibble: 32,561 x 2
##    target over_under
##    <chr>  <chr>
##  1 <=50K  under
##  2 <=50K  under
##  3 <=50K  under
##  4 <=50K  under
##  5 <=50K  under
##  6 <=50K  under
##  7 <=50K  under
##  8 >50K   over
##  9 >50K   over
## 10 >50K   over
## # i 32,551 more rows
```

or cut the range of a variable into intervals:

```
df %>%
  mutate(age_avg = mean(age),
         over_under_age_avg = cut(age,
                                  c(0, mean(age), max(age)),
                                  c("Lower than avg", "Above the avg"))
  ) %>%
  select(age, age_avg, over_under_age_avg)
```

```
## # A tibble: 32,561 x 3
##      age age_avg over_under_age_avg
##    <dbl>   <dbl> <fct>
##  1    39    38.6 Above the avg
##  2    50    38.6 Above the avg
##  3    38    38.6 Lower than avg
##  4    53    38.6 Above the avg
##  5    28    38.6 Lower than avg
##  6    37    38.6 Lower than avg
##  7    49    38.6 Above the avg
##  8    52    38.6 Above the avg
##  9    31    38.6 Lower than avg
## 10    42    38.6 Above the avg
## # i 32,551 more rows
```

### 4.9 Joining tibbles

Joining two data sets is a very useful and common function. We are going to use these 3 small data sets as an example.

```
sales <- data.frame(
  date = c("2022-01-01", "2022-01-02", "2022-01-03", "2022-01-04", "2022-01-05"),
  store_cd= c(1, 2, 3, 4, 5),
  product_cd= c(1, 2, 3, 4, 5),
```

```
  qty= c(10, 12, 9, 12, 8),
  sales= c(30, 60, 45, 24, 32)
)

stores <- data.frame(
  store_cd= c(1, 2, 3, 4, 6),
  address= c("1 main st", "20 side st", "19 square blvd", "101 first st", "1002 retail ave"),
  city= c("Main", "East", "West", "North", "South"),
  open_hours= c("7-23", "7-23", "9-21", "9-21", "9-21")
  )

products <- data.frame(
  product_cd= c(1, 2, 3, 4, 6),
  description= c("Soft drink", "Frozen snack", "Fruit", "Water", "Fruit 2"),
  unit_price= c(3.0, 5.0, 5.0, 2.0, 4.0),
  unit_measure= c("each", "each", "kg", "each", "kg")
  )
```

**4.9.1 Left join** All the rows from sales and matched rows from products.

```
sales %>% left_join(products, by="product_cd")
```

```
##         date store_cd product_cd qty sales  description unit_price unit_measure
## 1 2022-01-01        1          1  10    30   Soft drink          3         each
## 2 2022-01-02        2          2  12    60 Frozen snack          5         each
## 3 2022-01-03        3          3   9    45        Fruit          5           kg
## 4 2022-01-04        4          4  12    24        Water          2         each
## 5 2022-01-05        5          5   8    32         <NA>         NA         <NA>
```

**4.9.2 Right join** All the rows from stores and matched rows from sales.

```
sales %>% right_join(stores, by="store_cd")
```

```
##         date store_cd product_cd qty sales         address  city open_hours
## 1 2022-01-01        1          1  10    30       1 main st  Main       7-23
## 2 2022-01-02        2          2  12    60      20 side st  East       7-23
## 3 2022-01-03        3          3   9    45  19 square blvd  West       9-21
## 4 2022-01-04        4          4  12    24    101 first st North       9-21
## 5       <NA>        6         NA  NA    NA 1002 retail ave South       9-21
```

**4.9.3 Inner join** All the rows common to sales and stores.

```
sales %>% inner_join(stores, by="store_cd")
```

```
##         date store_cd product_cd qty sales        address  city open_hours
## 1 2022-01-01        1          1  10    30      1 main st  Main       7-23
## 2 2022-01-02        2          2  12    60     20 side st  East       7-23
## 3 2022-01-03        3          3   9    45 19 square blvd  West       9-21
## 4 2022-01-04        4          4  12    24   101 first st North       9-21
```

**4.9.4 Full join** All the rows from sales and stores.

```
sales %>% full_join(stores)
```

```
## Joining with `by = join_by(store_cd)`
```

```
##         date store_cd product_cd qty sales         address  city open_hours
## 1 2022-01-01        1          1  10    30       1 main st  Main       7-23
## 2 2022-01-02        2          2  12    60      20 side st  East       7-23
## 3 2022-01-03        3          3   9    45  19 square blvd  West       9-21
## 4 2022-01-04        4          4  12    24    101 first st North       9-21
## 5 2022-01-05        5          5   8    32            <NA>  <NA>       <NA>
## 6       <NA>        6         NA  NA    NA 1002 retail ave South       9-21
```

**4.9.5 Anti-join** Only rows that are in sales but not in products.

```
sales %>% anti_join(products)
```

```
## Joining with `by = join_by(product_cd)`
```

```
##         date store_cd product_cd qty sales
## 1 2022-01-05        5          5   8    32
```

### 4.10 Reshaping tables

This is useful when you do not have a tidy data set. A tidy data set if often required by plotting functions and many data exploration functions. Let us look at this example:

```
df_wide <- data.frame(
  project = c("project1", "project2", "project3"),
  Jan= sample(1000:2000, 3),
  Feb= sample(1000:2000, 3),
  Mar= sample(1000:2000, 3)
)
```

This is not a tidy data set (there is more than 1 observation per row). We can convert it to a tidy data set using the pivot function.

```
df_long <- df_wide %>%
  pivot_longer(cols= 2:4,
               names_to = "months",
               values_to = "expenses")
```

If we need a smaller table for visualization or for a presentation, we can pivot it back to the original format that is more compact and easier to read for your audience.

```
df_wide_2 <- df_long %>%
  pivot_wider(names_from = "months",
              values_from = "expenses")
```