AE 02: Wrangling college education metrics

Suggested answers

APPLICATION EXERCISE ANSWERS

MODIFIED
September 13, 2024

> **Important**
>
> These are suggested answers. This document should be used as reference only, it's not designed to be an exhaustive key.

To demonstrate data wrangling we will use data from College Scorecard.[1] The subset we will analyze contains a small number of metrics for all four-year colleges and universities in the United States for the 2022-23 academic year. [2]

[1] College Scorecard is a product of the U.S. Department of Education and compiles detailed information about student completion, debt and repayment, earnings, and more for all degree-granting institutions across the country.

[2] The full database contains thousands of variables from 1996-2023.

```
library(tidyverse)
```

The data is stored in `scorecard.csv`. The variables are:

- `unit_id` - Unit ID for institution
- `name` - Name of the college
- `state` - State abbreviation
- `type` - Type of college (Public; Private, nonprofit; Private, for-profit)
- `adm_rate` - Undergraduate admissions rate (from 0-100%)
- `sat_avg` - Average SAT equivalent score of students admitted
- `cost` - The average annual total cost of attendance, including tuition and fees, books and supplies, and living expenses
- `net_cost` - The average annual net cost of attendance (annual cost of attendance minus the average grant/scholarship aid)
- `avg_fac_sal` - Average faculty salary (9 month)
- `pct_pell` - Percentage of undergraduates who receive a Pell Grant
- `comp_rate` - Rate of first-time, full-time students at four-year institutions who complete their degree within six years
- `first_gen` - Share of first-generation students
- `debt` - Median debt of students after leaving school
- `locale` - Locale of institution

```
scorecard <- read_csv("data/scorecard.csv")
```

The data frame has over 1700 observations (rows), 1721 observations to be exact, so we will **not** view the entire data frame. Instead we'll use the commands below to help us explore the data.

```
glimpse(scorecard)
```

```
Rows: 1,721
Columns: 14
$ unit_id     <dbl> 100654, 100663, 100706, 100724, 100751, 100830, 100858, 10…
$ name        <chr> "Alabama A & M University", "University of Alabama at Birm…
$ state       <chr> "AL", "AL", "AL", "AL", "AL", "AL", "AL", "AL", "AL", "AL"…
$ type        <chr> "Public", "Public", "Public", "Public", "Public", "Public"…
$ adm_rate    <dbl> 0.6840, 0.8668, 0.7810, 0.9660, 0.8006, 0.9223, 0.4374, 0.…
$ sat_avg     <dbl> 920, 1291, 1259, 963, 1304, 1051, 1292, 1218, 1021, NA, 10…
$ cost        <dbl> 23167, 26257, 25777, 21900, 31024, 19771, 33650, 35495, 36…
$ net_cost    <dbl> 14982, 16755, 18240, 13527, 20888, 12630, 24297, 19723, 19…
$ avg_fac_sal <dbl> 77859, 106533, 92403, 72639, 96993, 75294, 104472, 63261, …
$ pct_pell    <dbl> 0.6536, 0.3308, 0.2173, 0.6976, 0.1788, 0.4589, 0.1254, 0.…
$ comp_rate   <dbl> 0.2678, 0.6442, 0.6295, 0.2773, 0.7276, 0.3584, 0.8075, 0.…
$ first_gen   <dbl> 0.3658281, 0.3412237, 0.3101322, 0.3434343, 0.2257127, 0.3…
$ debt        <dbl> 16600, 15832, 13905, 17500, 17986, 13119, 17750, 16000, 15…
$ locale      <chr> "City", "City", "City", "City", "City", "City", "City", "C…
```

```
names(scorecard)
```

```
 [1] "unit_id"     "name"        "state"       "type"        "adm_rate"
 [6] "sat_avg"     "cost"        "net_cost"    "avg_fac_sal" "pct_pell"
[11] "comp_rate"   "first_gen"   "debt"        "locale"
```

```
head(scorecard)
```

```
# A tibble: 6 × 14
  unit_id name  state type  adm_rate sat_avg  cost net_cost avg_fac_sal pct_pell
    <dbl> <chr> <chr> <chr>    <dbl>   <dbl> <dbl>    <dbl>       <dbl>    <dbl>
1  100654 Alab… AL    Publ…    0.684     920 23167    14982       77859    0.654
2  100663 Univ… AL    Publ…    0.867    1291 26257    16755      106533    0.331
3  100706 Univ… AL    Publ…    0.781    1259 25777    18240       92403    0.217
4  100724 Alab… AL    Publ…    0.966     963 21900    13527       72639    0.698
5  100751 The … AL    Publ…    0.801    1304 31024    20888       96993    0.179
6  100830 Aubu… AL    Publ…    0.922    1051 19771    12630       75294    0.459
# ℹ 4 more variables: comp_rate <dbl>, first_gen <dbl>, debt <dbl>,
#   locale <chr>
```

The `head()` function returns "A tibble: 6 x 14" and then the first six rows of the `scorecard` data.

Tibble vs. data frame

A **tibble** is an opinionated version of the `R` data frame. In other words, all tibbles are data frames, but not all data frames are tibbles!

There are two main differences between a tibble and a data frame:

1. When you print a tibble, the first ten rows and all of the columns that fit on the screen will display, along with the type of each column.

   Let's look at the differences in the output when we type `scorecard` (tibble) in the console versus typing `cars` (data frame) in the console.

2. Second, tibbles are somewhat more strict than data frames when it comes to subsetting data. You will get a warning message if you try to access a variable that doesn't exist in a tibble. You will get `NULL` if you try to access a variable that doesn't exist in a data frame.

```
scorecard$apple
```

```
Warning: Unknown or uninitialised column: `apple`.
```

```
NULL
```

```
cars$apple
```

```
NULL
```

Data wrangling with dplyr

dplyr is the primary package in the **tidyverse** for data wrangling.

> **Helpful data wrangling resources**
>
> - **dplyr** reference page
> - Data transformation cheatsheet

Quick summary of key dplyr functions[3]

Rows:

- `filter()` :chooses rows based on column values.
- `slice()` : chooses rows based on location.
- `arrange()` : changes the order of the rows

- `sample_n()` : take a random subset of the rows

Columns:

- `select()` : changes whether or not a column is included.
- `rename()` : changes the name of columns.
- `mutate()` : changes the values of columns and creates new columns.

Groups of rows:

- `summarize()` : collapses a group into a single row.
- `count()` : count unique values of one or more variables.
- `group_by()` : perform calculations separately for each value of a variable

[3] From dplyr vignette

Operators

In order to make comparisons, we will use **logical operators**. These should be familiar from other programming languages. See below for a reference table for how to use these operators in R.

operator	definition
`<`	is less than?
`<=`	is less than or equal to?
`>`	is greater than?
`>=`	is greater than or equal to?
`==`	is exactly equal to?
`!=`	is not equal to?
`x & y`	is x AND y?
`x | y`	is x OR y?
`is.na(x)`	is x NA?
`!is.na(x)`	is x not NA?
`x %in% y`	is x in y?
`!(x %in% y)`	is x not in y?
`!x`	is not x?

The final operator only makes sense if `x` is logical (TRUE / FALSE).

The pipe

Before working with data wrangling functions, let's formally introduce the pipe. The **pipe**, `|>`, is an operator (a tool) for passing information from one process to another. We will use `|>` mainly in data pipelines to pass the output of the previous line of code as the first input of the next line of code.

When reading code "in English", say "and then" whenever you see a pipe.

- **Your turn (3 minutes):** Run the following chunk and observe its output. Then, come up with a different way of obtaining the same output.

```
scorecard |>
  select(name, type) |>
  head()
```

```
# A tibble: 6 × 2
  name                              type
  <chr>                             <chr>
1 Alabama A & M University          Public
2 University of Alabama at Birmingham Public
3 University of Alabama in Huntsville Public
4 Alabama State University          Public
5 The University of Alabama         Public
6 Auburn University at Montgomery   Public
```

Exercises

Single function transformations

Demo: Select the `name` column.

```
select(.data = scorecard, name)
```

```
# A tibble: 1,721 × 1
   name
   <chr>
 1 Alabama A & M University
 2 University of Alabama at Birmingham
 3 University of Alabama in Huntsville
 4 Alabama State University
 5 The University of Alabama
 6 Auburn University at Montgomery
 7 Auburn University
```

```
 8 Birmingham-Southern College
 9 Faulkner University
10 Herzing University-Birmingham
# ℹ 1,711 more rows
```

Demo: Select all columns except `unit_id`.

```
select(.data = scorecard, -unit_id)
```

```
# A tibble: 1,721 × 13
   name         state type  adm_rate sat_avg  cost net_cost avg_fac_sal pct_pell
   <chr>        <chr> <chr>    <dbl>   <dbl> <dbl>    <dbl>       <dbl>    <dbl>
 1 Alabama A &… AL    Publ…    0.684     920 23167    14982       77859    0.654
 2 University … AL    Publ…    0.867    1291 26257    16755      106533    0.331
 3 University … AL    Publ…    0.781    1259 25777    18240       92403    0.217
 4 Alabama Sta… AL    Publ…    0.966     963 21900    13527       72639    0.698
 5 The Univers… AL    Publ…    0.801    1304 31024    20888       96993    0.179
 6 Auburn Univ… AL    Publ…    0.922    1051 19771    12630       75294    0.459
 7 Auburn Univ… AL    Publ…    0.437    1292 33650    24297      104472    0.125
 8 Birmingham-… AL    Priv…    0.572    1218 35495    19723       63261    0.227
 9 Faulkner Un… AL    Priv…    0.824    1021 36169    19478       58374    0.461
10 Herzing Uni… AL    Priv…    0.941      NA 28152    21275       59625    0.640
# ℹ 1,711 more rows
# ℹ 4 more variables: comp_rate <dbl>, first_gen <dbl>, debt <dbl>,
#   locale <chr>
```

Demo: Filter the data frame to keep only schools with a greater than 40% share of first-generation students.

```
filter(.data = scorecard, first_gen > .40)
```

```
# A tibble: 347 × 14
   unit_id name          state type  adm_rate sat_avg  cost net_cost avg_fac_sal
     <dbl> <chr>         <chr> <chr>    <dbl>   <dbl> <dbl>    <dbl>       <dbl>
 1  101189 Faulkner Uni… AL    Priv…    0.824    1021 36169    19478       58374
 2  101365 Herzing Univ… AL    Priv…    0.941      NA 28152    21275       59625
 3  101587 University o… AL    Publ…    0.689    1015 22456    14006       62226
 4  102270 Stillman Col… AL    Priv…    0.645      NA 25678    16085       46260
 5  104717 Grand Canyon… AZ    Priv…    0.779      NA 31440    21798       63747
 6  106467 Arkansas Tec… AR    Publ…    0.944    1090 20919    13765       62505
 7  107983 Southern Ark… AR    Publ…    0.636    1088 24242    16307       65637
 8  110361 California B… CA    Priv…    0.799      NA 49531    26538       91179
 9  110486 California S… CA    Publ…    0.866      NA 18410     7191       91530
10  110495 California S… CA    Publ…    0.966      NA 16968     5752       93537
# ℹ 337 more rows
# ℹ 5 more variables: pct_pell <dbl>, comp_rate <dbl>, first_gen <dbl>,
#   debt <dbl>, locale <chr>
```

Your turn: Filter the data frame to keep only public schools with a net cost of attendance below $12,000.

```
filter(.data = scorecard, type == "Public", net_cost < 12000)
```

```
# A tibble: 156 × 14
   unit_id name          state type  adm_rate sat_avg  cost net_cost avg_fac_sal
     <dbl> <chr>         <chr> <chr>    <dbl>   <dbl> <dbl>    <dbl>       <dbl>
 1  101879 University o… AL    Publ…    0.957      NA 21621    10527       78363
 2  102553 University o… AK    Publ…    0.653      NA 23461    10978       85653
 3  102632 University o… AK    Publ…    0.627      NA 17471     7056       74817
 4  106412 University o… AR    Publ…    0.693     878 19968     9607       54117
 5  106458 Arkansas Sta… AR    Publ…    0.695    1119 21176    11857       67644
 6  110486 California S… CA    Publ…    0.866      NA 18410     7191       91530
 7  110495 California S… CA    Publ…    0.966      NA 16968     5752       93537
 8  110510 California S… CA    Publ…    0.911      NA 18750     8215       94716
 9  110529 California S… CA    Publ…    0.554      NA 21655    11902      103113
10  110547 California S… CA    Publ…    0.891      NA 14958     4058       95364
# ℹ 146 more rows
# ℹ 5 more variables: pct_pell <dbl>, comp_rate <dbl>, first_gen <dbl>,
#   debt <dbl>, locale <chr>
```

```
filter(.data = scorecard, type == "Public" & net_cost < 12000)
```


```
# A tibble: 156 × 14
   unit_id name          state type  adm_rate sat_avg  cost net_cost avg_fac_sal
     <dbl> <chr>         <chr> <chr>    <dbl>   <dbl> <dbl>    <dbl>       <dbl>
 1  101879 University o… AL    Publ…    0.957      NA 21621    10527       78363
 2  102553 University o… AK    Publ…    0.653      NA 23461    10978       85653
 3  102632 University o… AK    Publ…    0.627      NA 17471     7056       74817
 4  106412 University o… AR    Publ…    0.693     878 19968     9607       54117
 5  106458 Arkansas Sta… AR    Publ…    0.695    1119 21176    11857       67644
 6  110486 California S… CA    Publ…    0.866      NA 18410     7191       91530
 7  110495 California S… CA    Publ…    0.966      NA 16968     5752       93537
 8  110510 California S… CA    Publ…    0.911      NA 18750     8215       94716
 9  110529 California S… CA    Publ…    0.554      NA 21655    11902      103113
10  110547 California S… CA    Publ…    0.891      NA 14958     4058       95364
# ℹ 146 more rows
# ℹ 5 more variables: pct_pell <dbl>, comp_rate <dbl>, first_gen <dbl>,
#   debt <dbl>, locale <chr>
```


Multiple function transformations

Your turn: How many public colleges and universities in each state have a net cost of attendance below $12,000?

```
# using group_by() and summarize()
scorecard |>
  filter(type == "Public", net_cost < 12000) |>
  group_by(state) |>
  summarize(n = n())
```

```
# A tibble: 42 × 2
   state     n
   <chr> <int>
 1 AK        2
 2 AL        1
 3 AR        2
 4 AZ        1
 5 CA       14
 6 CO        1
 7 CT        3
 8 FL       13
 9 FM        1
10 GA        5
# ℹ 32 more rows
```

```
# using count()
scorecard |>
  filter(type == "Public", net_cost < 12000) |>
  count(state)
```

```
# A tibble: 42 × 2
   state     n
   <chr> <int>
 1 AK        2
 2 AL        1
 3 AR        2
 4 AZ        1
 5 CA       14
 6 CO        1
 7 CT        3
 8 FL       13
 9 FM        1
10 GA        5
# ℹ 32 more rows
```

Your turn: Generate a data frame with the 10 most expensive colleges in 2022-23 based on net cost of attendance.

We could use a combination of `arrange()` and `slice()` to sort the data frame from most to least expensive, then keep the first 10 rows:

```
# using desc()
arrange(.data = scorecard, desc(net_cost)) |>
  slice(1:10)
```

```
# A tibble: 10 × 14
   unit_id name          state type  adm_rate sat_avg  cost net_cost avg_fac_sal
     <dbl> <chr>         <chr> <chr>    <dbl>   <dbl> <dbl>    <dbl>       <dbl>
 1  197151 School of Vi… NY    Priv…    0.874    1298 72488    56457       48213
 2  214971 Pennsylvania… PA    Priv…    0.913      NA 63997    56164       55071
```

```
 3  136774 Ringling Col… FL    Priv…    0.647      NA 73962    54319       82413
 4  111081 California I… CA    Priv…    0.248      NA 75865    51386       87039
 5  192712 Manhattan Sc… NY    Priv…    0.550      NA 72477    51067       72999
 6  119775 Newschool of… CA    Priv…    0.450      NA 56958    50126       62262
 7  165662 Emerson Coll… MA    Priv…    0.428    1373 75777    49466       93033
 8  164748 Berklee Coll… MA    Priv…    0.542      NA 66950    49230       96093
 9  193654 The New Scho… NY    Priv…    0.572      NA 75533    49086      113310
10  164368 Hult Interna… MA    Priv…    0.477      NA 74000    49047      101826
# ℹ 5 more variables: pct_pell <dbl>, comp_rate <dbl>, first_gen <dbl>,
#   debt <dbl>, locale <chr>
```

```
# using -
arrange(.data = scorecard, -net_cost) |>
  slice(1:10)
```

```
# A tibble: 10 × 14
   unit_id name          state type  adm_rate sat_avg  cost net_cost avg_fac_sal
     <dbl> <chr>         <chr> <chr>    <dbl>   <dbl> <dbl>    <dbl>       <dbl>
 1  197151 School of Vi… NY    Priv…    0.874    1298 72488    56457       48213
 2  214971 Pennsylvania… PA    Priv…    0.913      NA 63997    56164       55071
 3  136774 Ringling Col… FL    Priv…    0.647      NA 73962    54319       82413
 4  111081 California I… CA    Priv…    0.248      NA 75865    51386       87039
 5  192712 Manhattan Sc… NY    Priv…    0.550      NA 72477    51067       72999
 6  119775 Newschool of… CA    Priv…    0.450      NA 56958    50126       62262
 7  165662 Emerson Coll… MA    Priv…    0.428    1373 75777    49466       93033
 8  164748 Berklee Coll… MA    Priv…    0.542      NA 66950    49230       96093
 9  193654 The New Scho… NY    Priv…    0.572      NA 75533    49086      113310
10  164368 Hult Interna… MA    Priv…    0.477      NA 74000    49047      101826
# ℹ 5 more variables: pct_pell <dbl>, comp_rate <dbl>, first_gen <dbl>,
#   debt <dbl>, locale <chr>
```

We can also use the `slice_max()` function in **dplyr** to accomplish the same thing with a single function.

```
slice_max(.data = scorecard, order_by = net_cost, n = 10)
```

```
# A tibble: 10 × 14
   unit_id name          state type  adm_rate sat_avg  cost net_cost avg_fac_sal
     <dbl> <chr>         <chr> <chr>    <dbl>   <dbl> <dbl>    <dbl>       <dbl>
 1  197151 School of Vi… NY    Priv…    0.874    1298 72488    56457       48213
 2  214971 Pennsylvania… PA    Priv…    0.913      NA 63997    56164       55071
 3  136774 Ringling Col… FL    Priv…    0.647      NA 73962    54319       82413
 4  111081 California I… CA    Priv…    0.248      NA 75865    51386       87039
 5  192712 Manhattan Sc… NY    Priv…    0.550      NA 72477    51067       72999
 6  119775 Newschool of… CA    Priv…    0.450      NA 56958    50126       62262
 7  165662 Emerson Coll… MA    Priv…    0.428    1373 75777    49466       93033
 8  164748 Berklee Coll… MA    Priv…    0.542      NA 66950    49230       96093
 9  193654 The New Scho… NY    Priv…    0.572      NA 75533    49086      113310
10  164368 Hult Interna… MA    Priv…    0.477      NA 74000    49047      101826
# ℹ 5 more variables: pct_pell <dbl>, comp_rate <dbl>, first_gen <dbl>,
#   debt <dbl>, locale <chr>
```

Your turn: Generate a data frame with the average SAT score for each type of college.

Note that since the `sat_avg` column contains `NA`s (missing values), we need to explicitly exclude them from our mean calculation. Otherwise the resulting data frame contains `NA`s.

```
# incorrect - ignores NAs
scorecard |>
  group_by(type) |>
  summarize(mean_sat = mean(sat_avg))
```

```
# A tibble: 3 × 2
  type                mean_sat
  <chr>                  <dbl>
1 Private, for-profit       NA
2 Private, nonprofit        NA
3 Public                    NA
```

```
# exclude NAs using mean()
scorecard |>
  group_by(type) |>
  summarize(mean_sat = mean(sat_avg, na.rm = TRUE))
```

```
# A tibble: 3 × 2
  type                mean_sat
  <chr>                  <dbl>
1 Private, for-profit    1174.
2 Private, nonprofit     1199.
3 Public                 1132.
```

```
# exclude NAs using drop_na() to remove the rows prior to summarizing
scorecard |>
  drop_na(sat_avg) |>
  group_by(type) |>
  summarize(mean_sat = mean(sat_avg))
```

```
# A tibble: 3 × 2
  type                mean_sat
  <chr>                  <dbl>
1 Private, for-profit    1174.
2 Private, nonprofit     1199.
3 Public                 1132.
```

Your turn: Calculate for each school how many students it takes to pay the average faculty member's salary and generate a data frame with the school's name, net cost of attendance, average faculty salary, and the calculated value. How many Cornell and Ithaca College students does it take to pay their average faculty member's salary?

> **Note**

> You should use the net cost of attendance measure, not the sticker price.

```
scorecard |>
  # mutate() to create a column with the ratio
  mutate(ratio = avg_fac_sal / net_cost) |>
  # select() to keep only the name and ratio columns
  select(name, net_cost, avg_fac_sal, ratio) |>
  # filter() to keep only Cornell and Ithaca College
  filter(name == "Cornell University" | name == "Ithaca College")
```

```
# A tibble: 2 × 4
  name               net_cost avg_fac_sal ratio
  <chr>                 <dbl>       <dbl> <dbl>
1 Cornell University    29651      146826  4.95
2 Ithaca College        35552       83610  2.35
```

Your turn: Calculate how many private, nonprofit schools have a smaller net cost than Cornell University.

You will need to create a new column that ranks the schools by net cost of attendance. Look at the back of the **dplyr** cheatsheet for functions that can be used to calculate rankings.

Reported as the number as the total number of schools:

```
scorecard |>
  # keep only private schools and sort by net cost in increasing order
  filter(type == "Private, nonprofit") |>
  arrange(net_cost) |>
  # use row_number() to rank each school by net cost but subtract 1
  # since Cornell is not cheaper than itself
  mutate(net_cost_rank = row_number() - 1) |>
  # examine output for Cornell
  filter(name == "Cornell University") |>
  select(name, net_cost, net_cost_rank)
```

```
# A tibble: 1 × 3
  name               net_cost net_cost_rank
  <chr>                 <dbl>         <dbl>
1 Cornell University    29651           889
```

Reported as the number as the percentage of schools:

```
scorecard |>
  # keep only private schools
  filter(type == "Private, nonprofit") |>
  # use percent_rank() to rank each school by net cost in percentiles
  mutate(net_cost_rank = percent_rank(net_cost)) |>
  # examine output for Cornell
```

```
  filter(name == "Cornell University") |>
  select(name, net_cost, net_cost_rank)
```

```
# A tibble: 1 × 3
  name               net_cost net_cost_rank
  <chr>                 <dbl>         <dbl>
1 Cornell University    29651         0.813
```

Session information