Section A01
CDS 101 *Introduction to Computational and Data Sciences*

Class 7: Data wrangling II

May 30, 2018

General

Annoucements

- Homework 1 due tonight by 11:59pm: http://summer18.cds101.com/assignments/homework-1/
- Reading 7: *Representing distributions* will be posted on website this afternoon

`dplyr` package (continued)

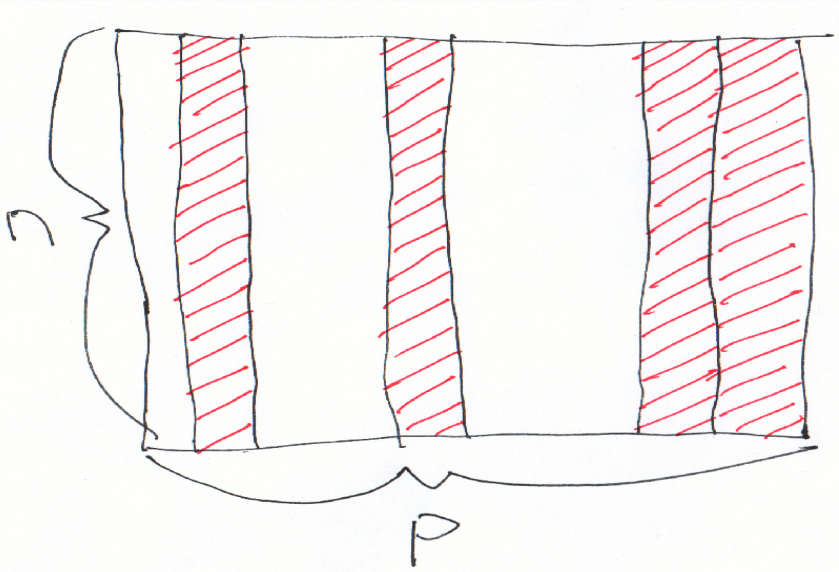
Get copy of `dplyr` demo repository

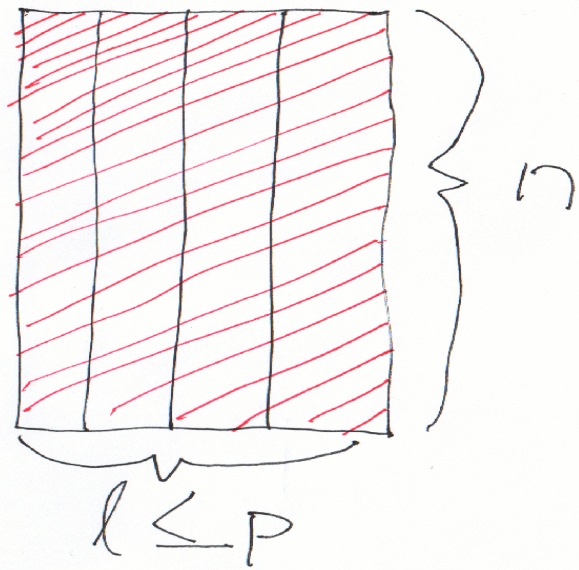
- Open RStudio and reload your **dplyr demo** repository from last class
- If you need to re-clone, find the link to Github repository in the Slack channel `su18-a01-activities`
- Follow along in the demos

`dplyr` so far

In the previous class, we reviewed the following `dplyr` commands

- `select()`

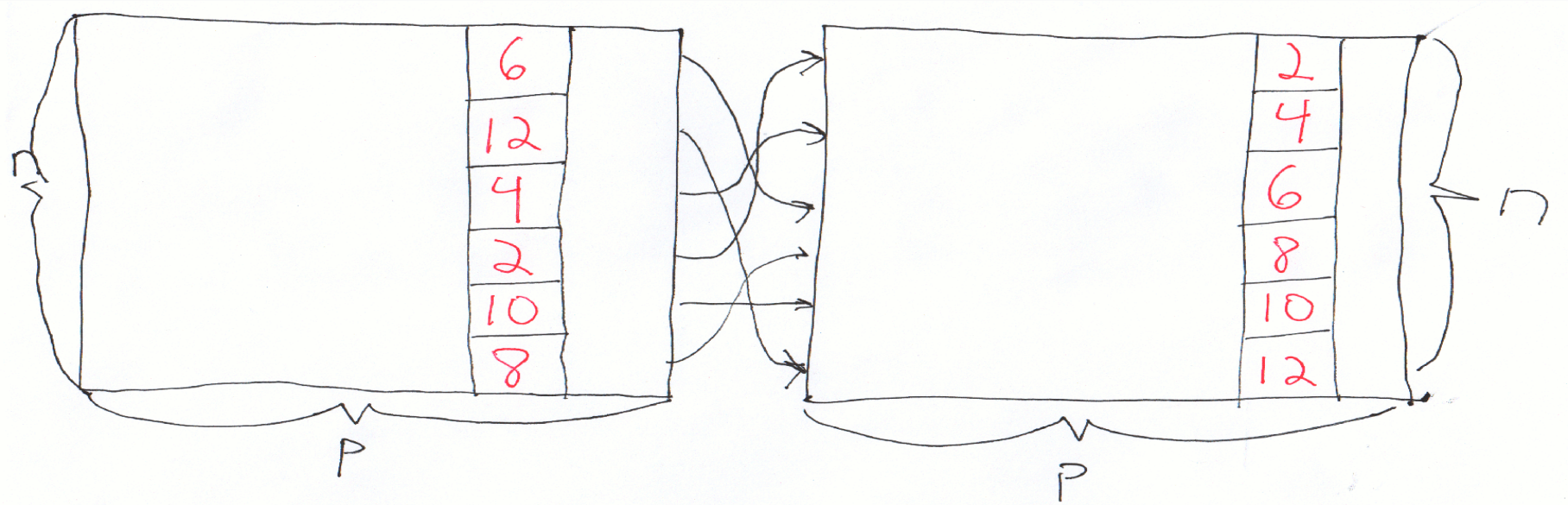




`dplyr` so far

In the previous class, we reviewed the following `dplyr` commands

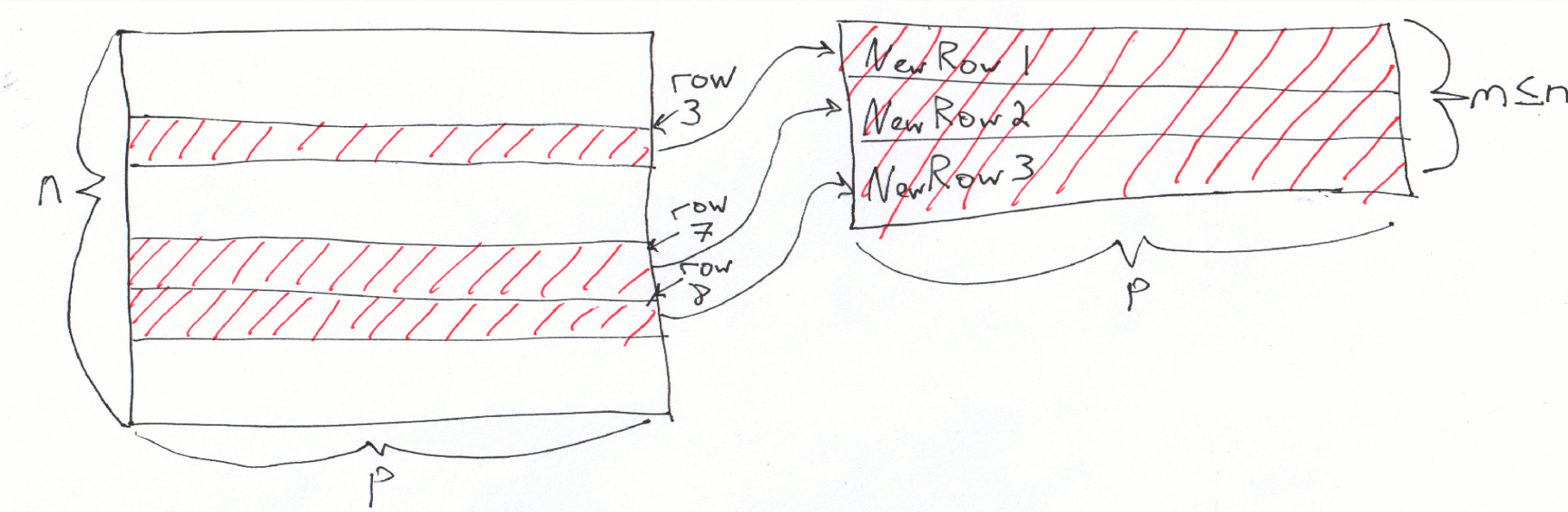
- `select()`
- `arrange()`



`dplyr` so far

In the previous class, we reviewed the following `dplyr` commands

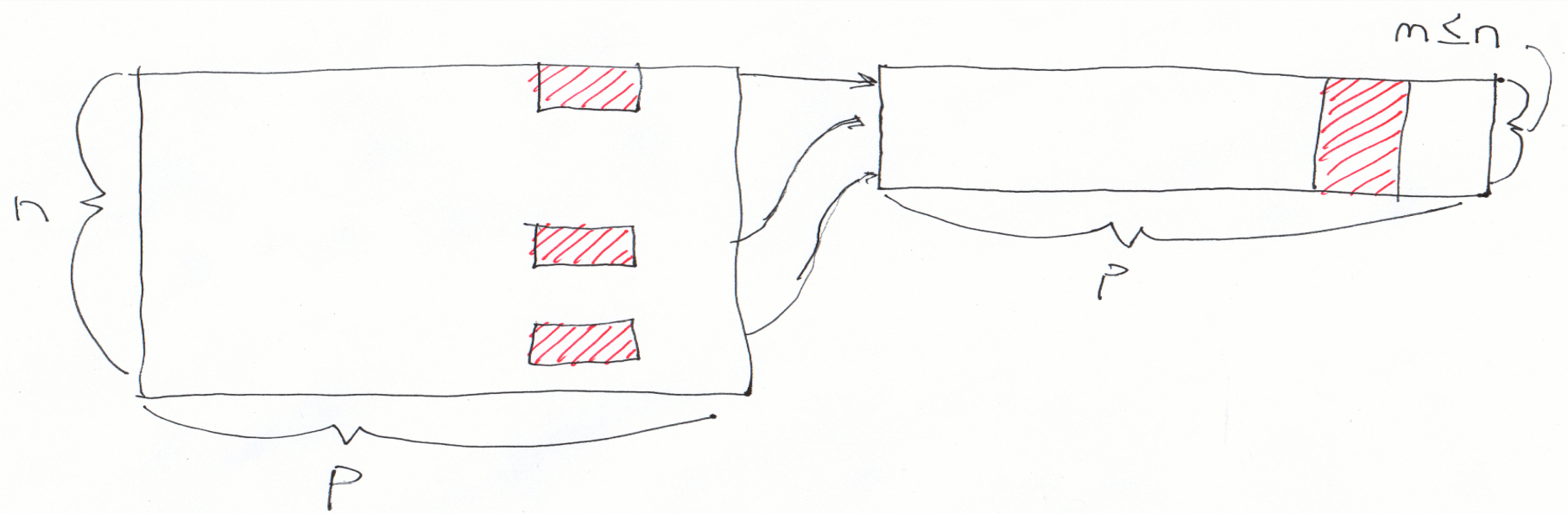
- `select()`
- `arrange()`
- `slice()`



`dplyr` so far

In the previous class, we reviewed the following `dplyr` commands

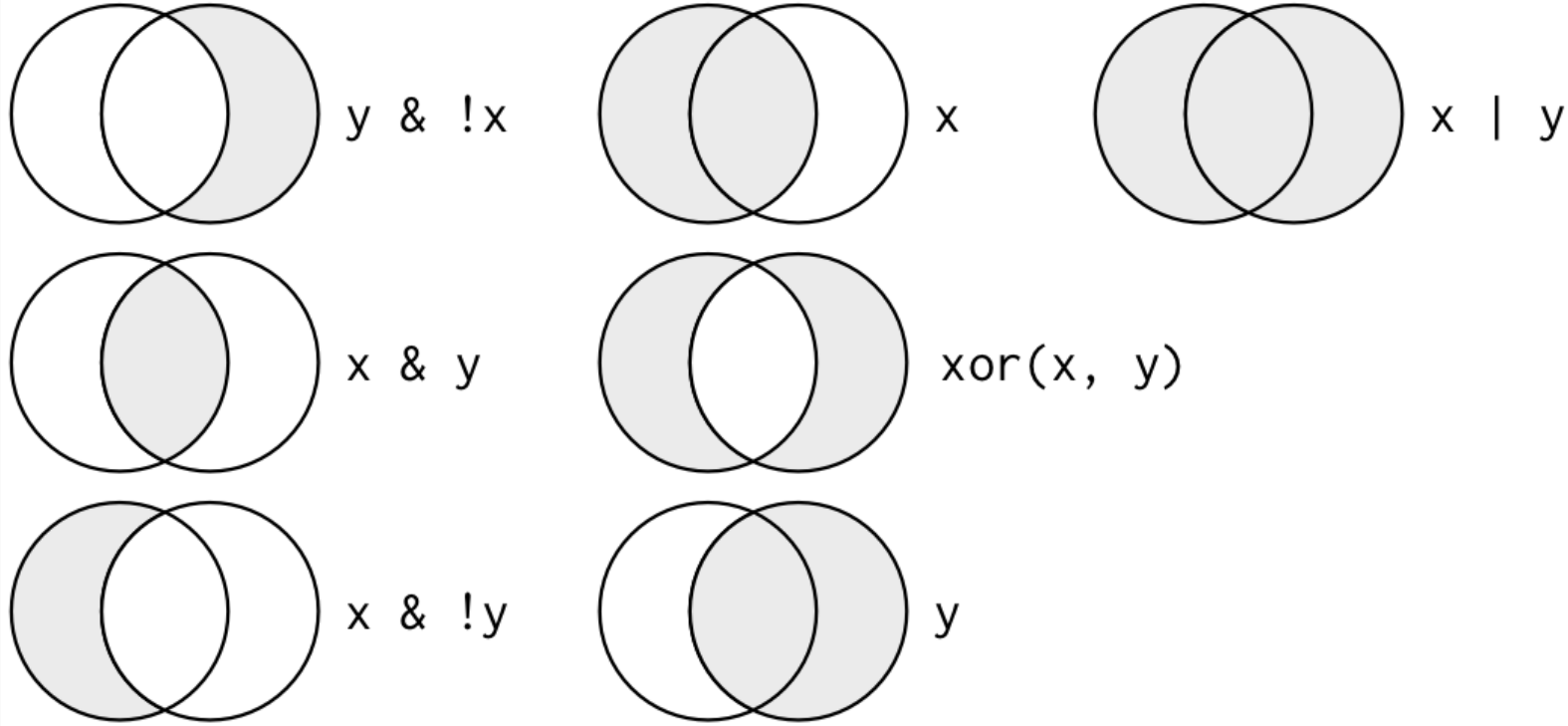
- `select()`
- `arrange()`
- `slice()`
- `filter()`



Use comparisons for filtering

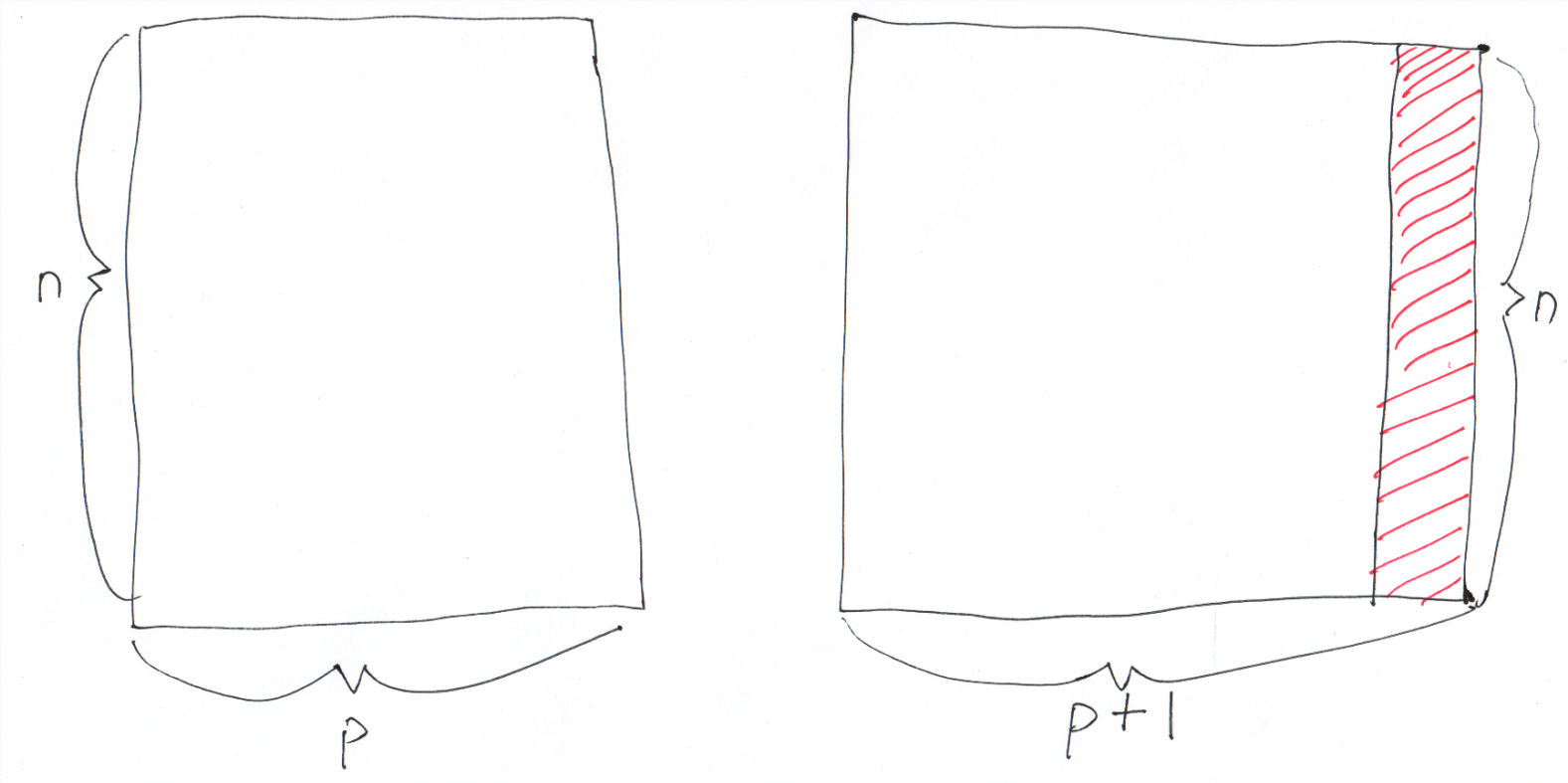
- `>`: greater than
- `>=`: greater than or equal to
- `<`: less than
- `<=`: less than or equal to
- `!=`: not equal
- `==`: equal

Logical operators



Source: Digital image of logical operations, *R for Data Science website*, accessed September 20, 2017, http://r4ds.had.co.nz/transform.html#logical-operators

mutate()



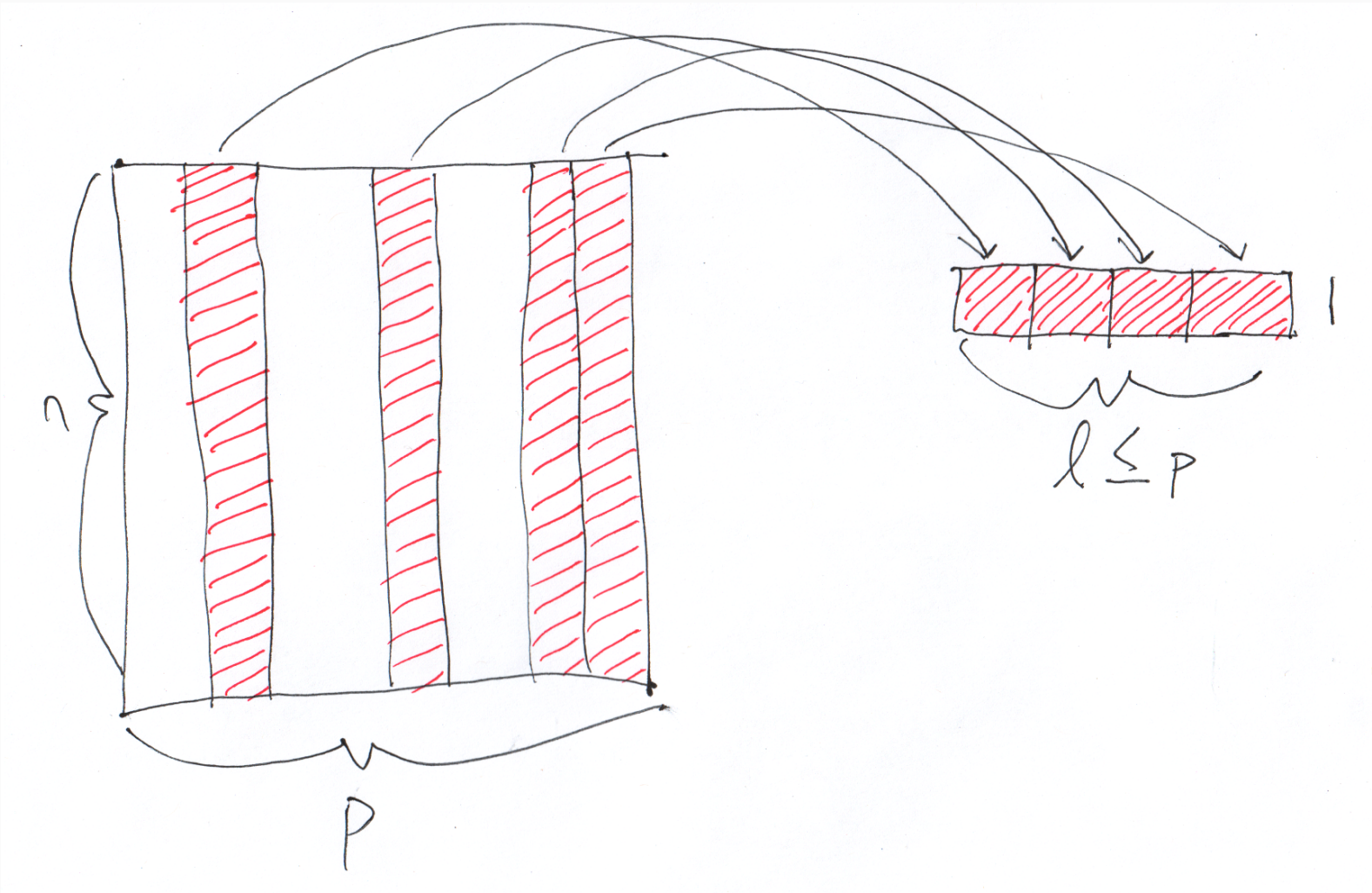
Using `mutate()`

- Many different operators and functions can be used with `mutate()`
- **Arithmetic operators:** `+`, `-`, `*`, `/`, `^`
- **Modular arithmetic**
  - `%/%`: integer division
  - `%%`: remainder
- **Logs:** `log()`
- **Logical comparisons:** `<`, `<=`, `>`, `>=`, `!=`

`mutate()` demo

Follow along in RStudio

group_by() and summarize()



Using `summarize()`

- `n()`: Counts number of rows in a group
- `sum()`: For numerical variables, sums rows within a group
- **statistical**: `mean()`, `median()`, `sd()`, `min()`, `max()`
- Counts and proportions of logical values: `sum(x > 10)`, `mean(y == 0)`

`group_by()` and `summarize()` demo

Follow along in RStudio

Other helpful `dplyr` verbs

- `transmute()`: Like `mutate()`, except the transformed output is placed in a new data frame
- `pull()`: Extract column into the base R `vector` data type
- `rename()`: Convenient way to change the name of a variable (column)
- `distinct()`: Finds unique rows in the dataset
- `count()`: Group by category and count the number of group members

transmute() example

```
presidential %>%
  transmute(term_length = interval(start, end) / dyears(1))
```

```
## # A tibble: 11 x 1
##    term_length
##          <dbl>
##  1        8.01
##  2        2.84
##  3        5.17
##  4        5.55
##  5        2.45
##  6        4.00
##  7        8.01
##  8        4.00
##  9        8.01
## 10        8.01
## 11        8.01
```

pull() example

```
presidential %>%
  pull(name)
```

```
##  [1] "Eisenhower" "Kennedy"    "Johnson"    "Nixon"      "Ford"
##  [6] "Carter"     "Reagan"     "Bush"       "Clinton"    "Bush"
## [11] "Obama"
```

rename() example

```
presidential %>%
  rename(term_begin = start, term_end = end)
```

```
## # A tibble: 11 x 4
##    name       term_begin term_end   party
##    <chr>      <date>     <date>     <chr>
##  1 Eisenhower 1953-01-20 1961-01-20 Republican
##  2 Kennedy    1961-01-20 1963-11-22 Democratic
##  3 Johnson    1963-11-22 1969-01-20 Democratic
##  4 Nixon      1969-01-20 1974-08-09 Republican
##  5 Ford       1974-08-09 1977-01-20 Republican
##  6 Carter     1977-01-20 1981-01-20 Democratic
##  7 Reagan     1981-01-20 1989-01-20 Republican
##  8 Bush       1989-01-20 1993-01-20 Republican
##  9 Clinton    1993-01-20 2001-01-20 Democratic
## 10 Bush       2001-01-20 2009-01-20 Republican
## 11 Obama      2009-01-20 2017-01-20 Democratic
```

distinct() example

```
presidential %>%
  distinct(name)
```

```
## # A tibble: 10 x 1
##    name
##    <chr>
##  1 Eisenhower
##  2 Kennedy
##  3 Johnson
##  4 Nixon
##  5 Ford
##  6 Carter
##  7 Reagan
##  8 Bush
##  9 Clinton
## 10 Obama
```

count() example

```
presidential %>%
  count(party)
```

```
## # A tibble: 2 x 2
##   party          n
##   <chr>      <int>
## 1 Democratic     5
## 2 Republican     6
```

Practicing with real data: Chicago Towed Vehicles dataset

Chicago Towing Data

- US cities are posting data online for citizens to download and analyze

Chicago Towing Data

- US cities are posting data online for citizens to download and analyze
- Chicago posts Towed Vehicle information over the past 90 days that is a good dataset to practice our `dplyr` skills on

Chicago Towing Data

- US cities are posting data online for citizens to download and analyze
- Chicago posts Towed Vehicle information over the past 90 days that is a good dataset to practice our `dplyr` skills on
- Link (also posted in Slack): https://data.cityofchicago.org/Transportation/Towed-Vehicles/ygr5-vcbg/
- Follow along to see how to import this dataset into RStudio and create a compressed version of it

Credits

These slides are based on the following sources:

- Ideas and examples for the **dplyr demos** adapted from *Modern Data Science with R* by Benjamin Baumer, Daniel Kaplan, and Nicholas Horton, chapter 4.