# TEXT ANALYSIS: MATCHING AND LINKING FOR JOINING DATA

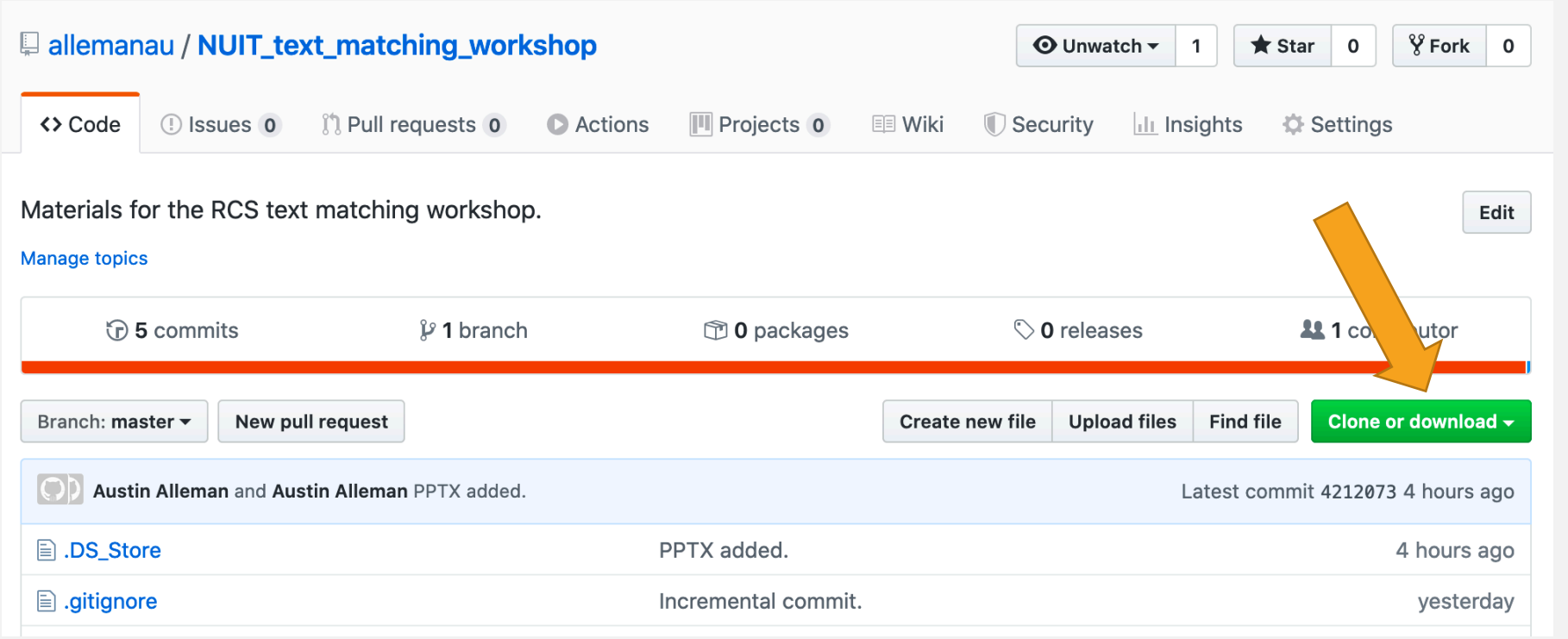
Austin Alleman

November 20, 2019

# WORKSHOP RESOURCES

All materials, including these slides and R scripts are available here:

https://github.com/allemanau/NUIT_text_matching_workshop



# INTRODUCTIONS

**Who am I?**

- Data science research consultant with RCS,  stats PhD candidate
- Fuzzy logic evangelist and practitioner

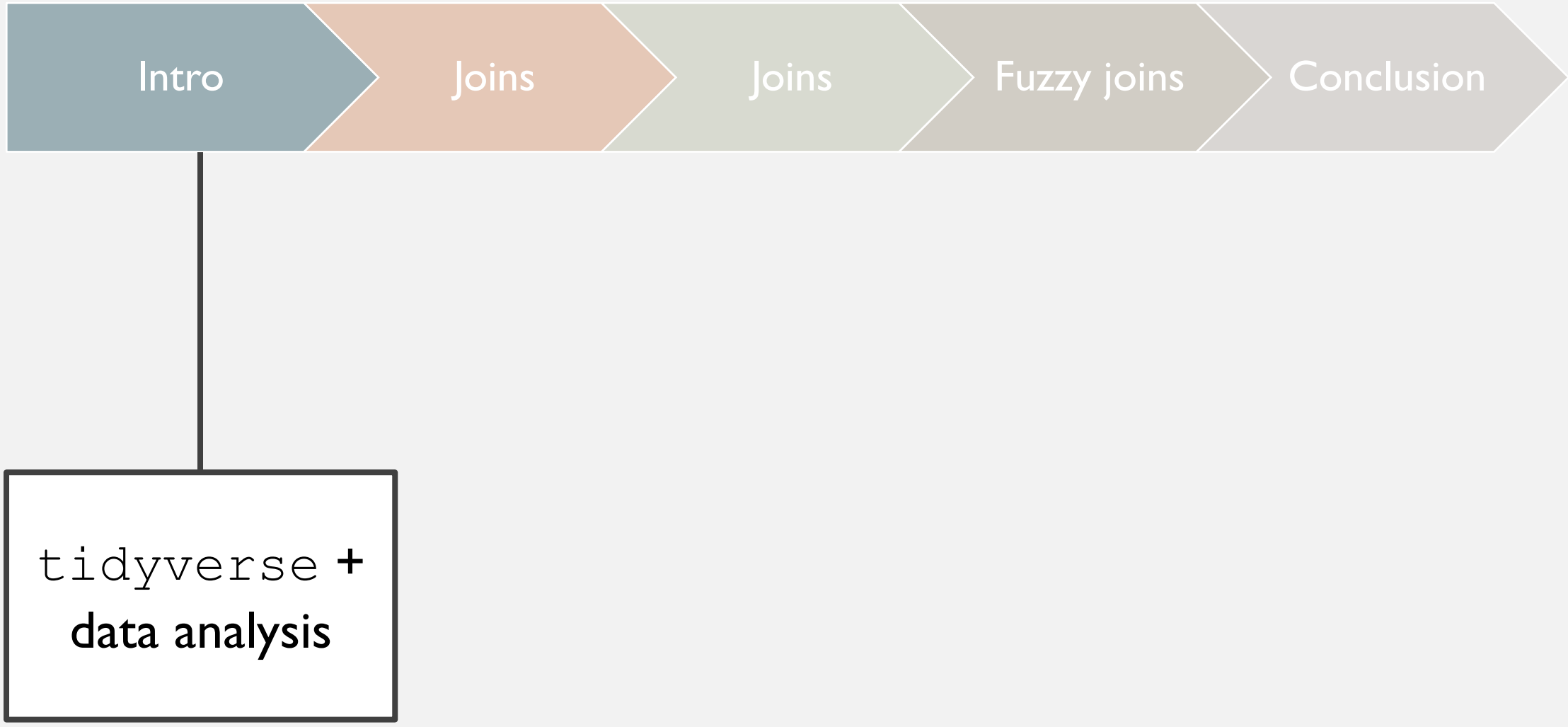
**Who are you?**

- Introduce yourself to your neighbors!
- What do you work on, and what brings you here?

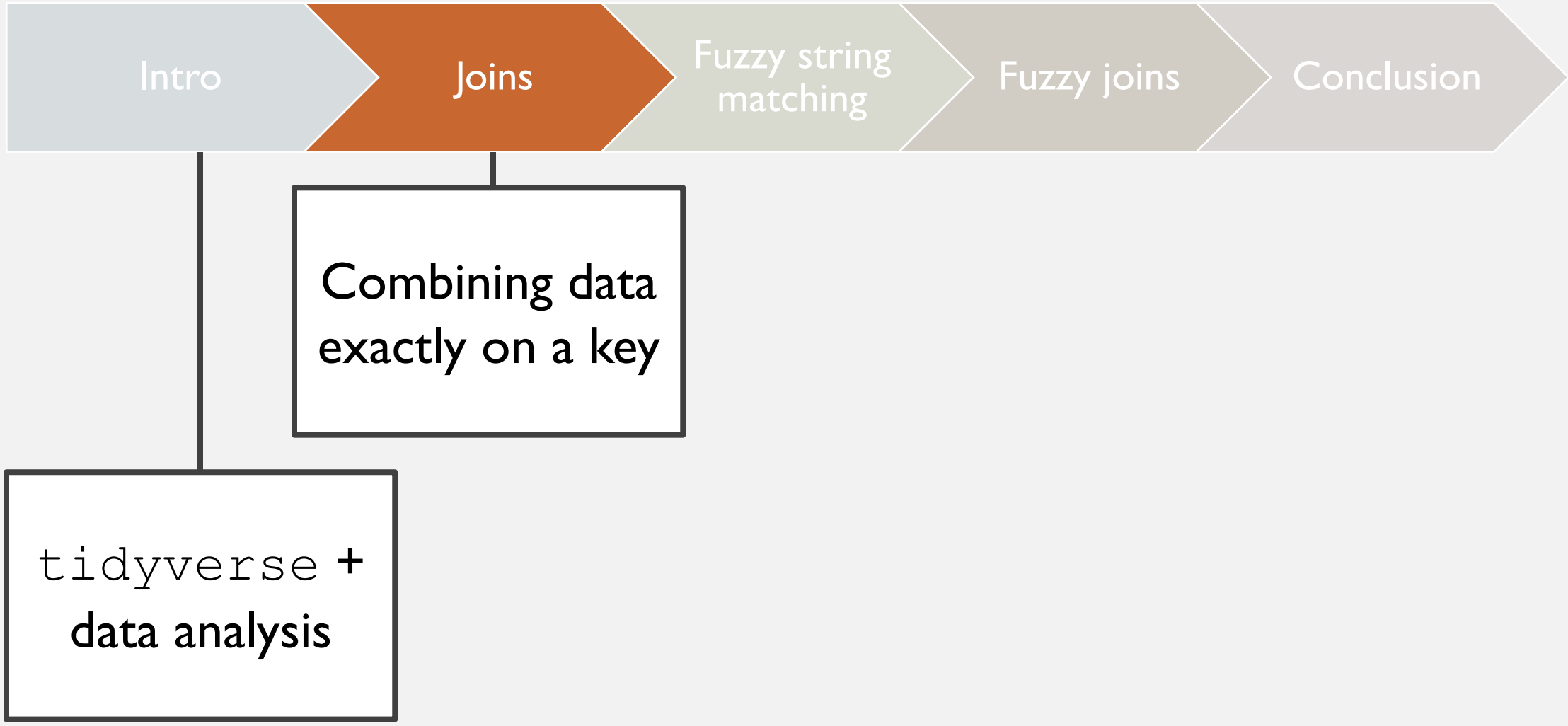
# WORKSHOP GOALS

- review joins for combining data sets and highlight cases where exact matching will fail
- discuss some of the mechanics underlying fuzzy matching techniques and parameter tuning
- use R's `tidyverse`, `stringdist`, and `fuzzyjoin` libraries to work through an example on real-world data

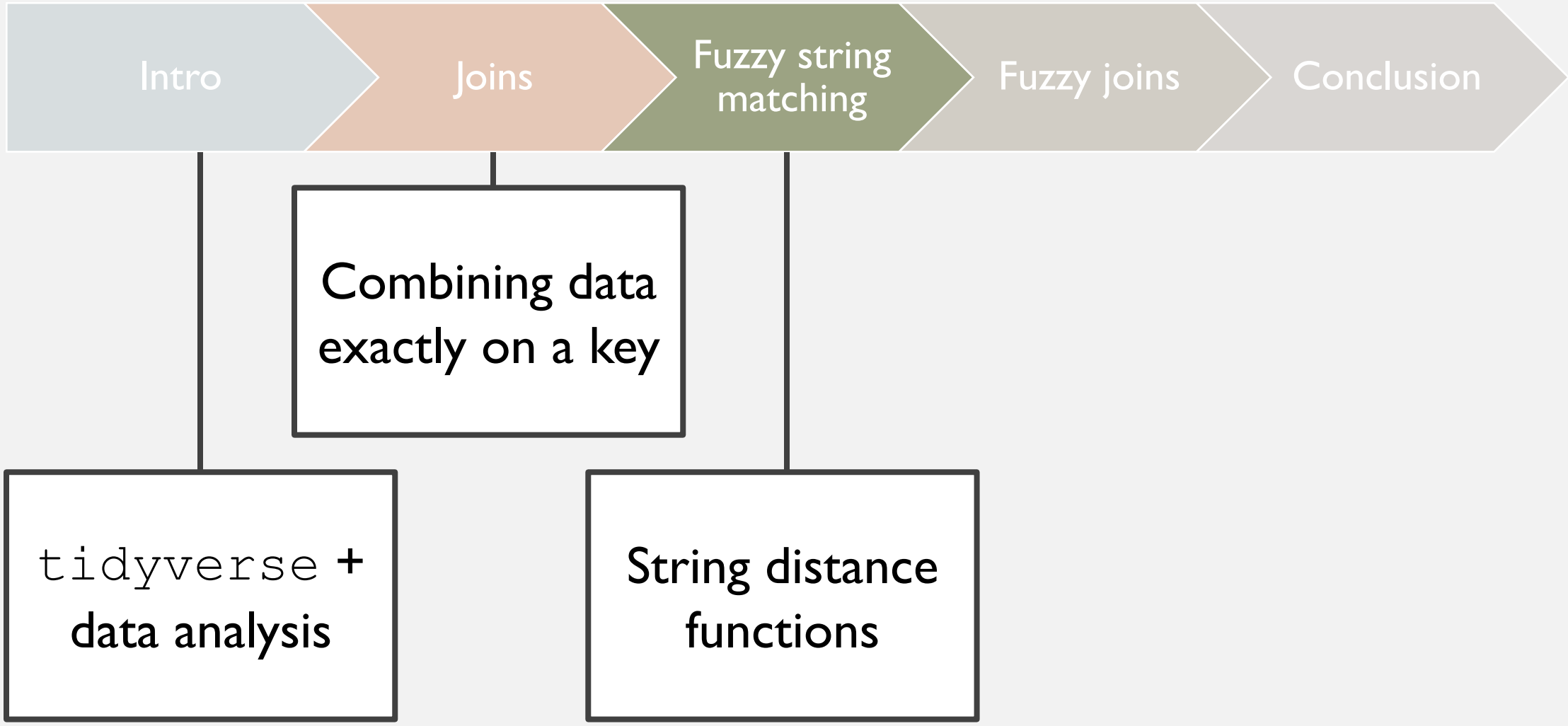
# ORDER OF OPERATIONS



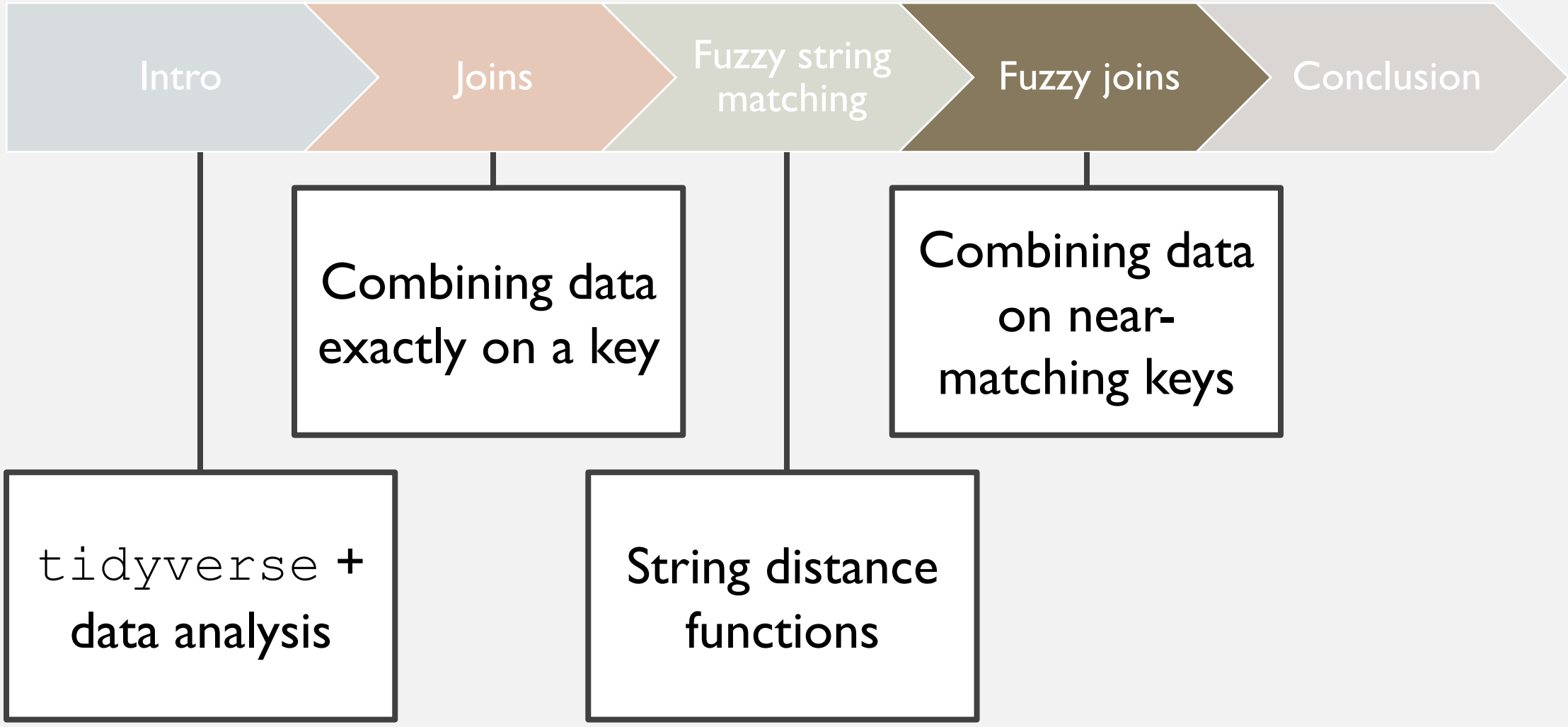
# ORDER OF OPERATIONS



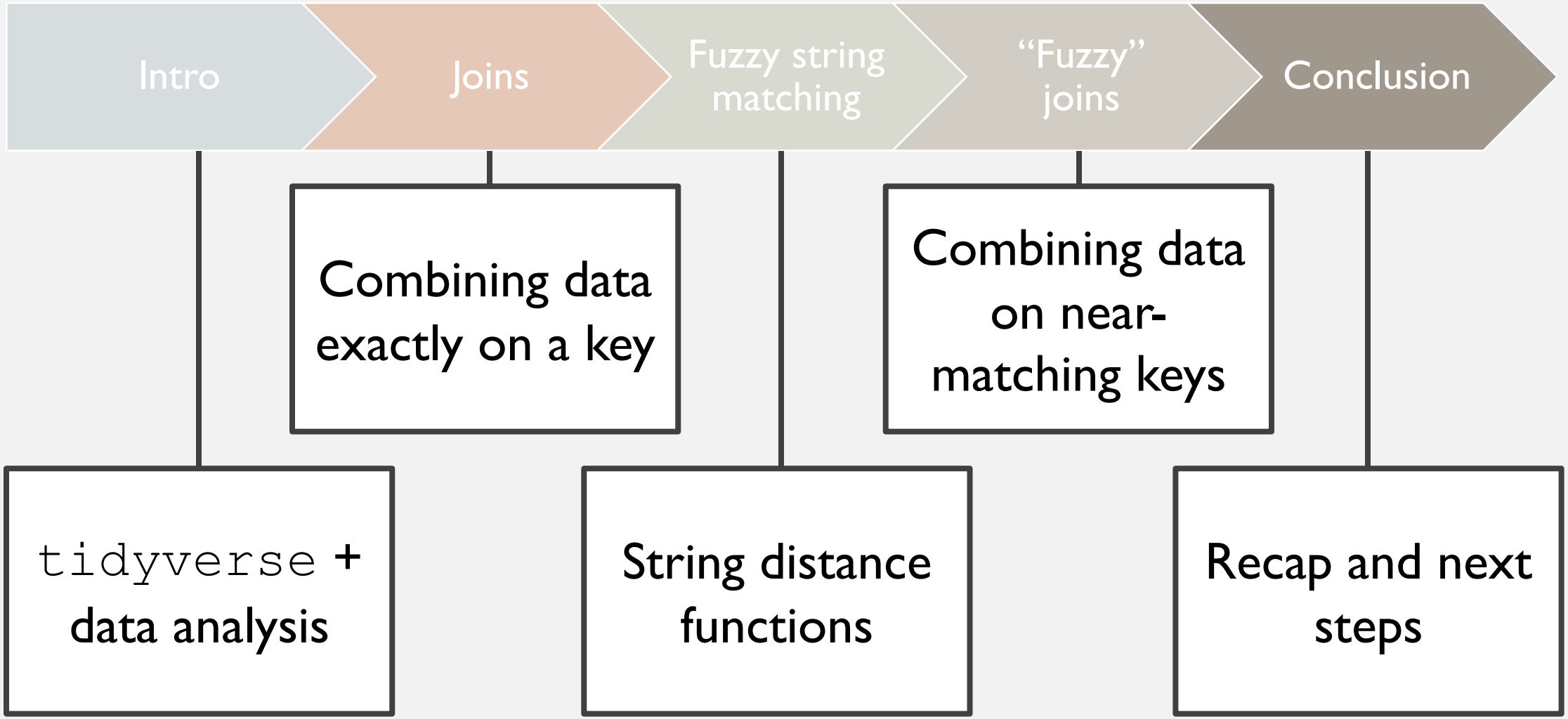
# ORDER OF OPERATIONS



# ORDER OF OPERATIONS



# ORDER OF OPERATIONS



# ANALYZING SCRAPED DATA

Today, we'll analyze a web-scraped manual entry data set.  Some of the challenges associated with this kind of data include:

Today, we'll analyze a web-scraped manual entry data set. Some of the challenges associated with this kind of data include:

Spelling/keystroke errors


Today, we'll analyze a web-scraped manual entry data set. Some of the challenges associated with this kind of data include:

Spelling/keystroke errors



Multiple names per entity


Today, we'll analyze a web-scraped manual entry data set. Some of the challenges associated with this kind of data include:

Spelling/keystroke errors

Multiple names/**keys** per entity

Extra characters

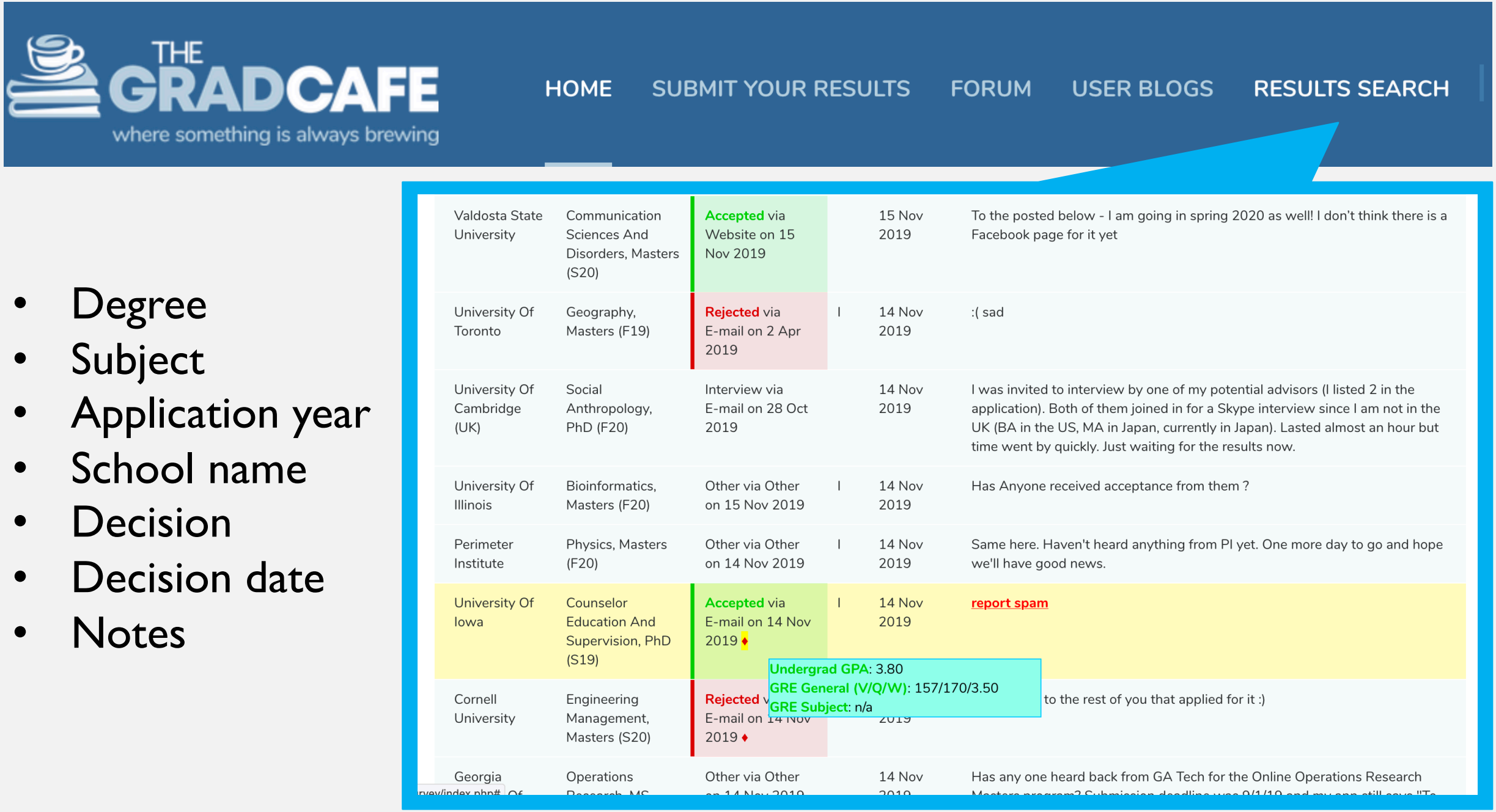





| | | |
|---|---|---|
| Northwestern Universitytern | Ibis, PhD (F18) | **Rejected** via E-mail on 11 Apr 2019 ♦ |

| | | |
|---|---|---|
| Northwestern Universitythwestern | Mechanical Engineeringchanical, PhD (F19) | **Accepted** via E-mail on 29 Mar 2019 |

| | | |
|---|---|---|
| Nortwestern | Music Composition (DMA), Other (F17) | Interview via E-mail on 20 Jan 2017 |




| | | | | | |
|---|---|---|---|---|---|
| Valdosta State University | Communication Sciences And Disorders, Masters (S20) | Accepted via Website on 15 Nov 2019 | | 15 Nov 2019 | To the posted below - I am going in spring 2020 as well! I don't think there is a Facebook page for it yet |
| University Of Toronto | Geography, Masters (F19) | Rejected via E-mail on 2 Apr 2019 | I | 14 Nov 2019 | :( sad |
| University Of Cambridge (UK) | Social Anthropology, PhD (F20) | Interview via E-mail on 28 Oct 2019 | | 14 Nov 2019 | I was invited to interview by one of my potential advisors (I listed 2 in the application). Both of them joined in for a Skype interview since I am not in the UK (BA in the US, MA in Japan, currently in Japan). Lasted almost an hour but time went by quickly. Just waiting for the results now. |
| University Of Illinois | Bioinformatics, Masters (F20) | Other via Other on 15 Nov 2019 | I | 14 Nov 2019 | Has Anyone received acceptance from them ? |
| Perimeter Institute | Physics, Masters (F20) | Other via Other on 14 Nov 2019 | I | 14 Nov 2019 | Same here. Haven't heard anything from PI yet. One more day to go and hope we'll have good news. |
| University Of Iowa | Counselor Education And Supervision, PhD (S19) | Accepted via E-mail on 14 Nov 2019 ♦ | I | 14 Nov 2019 | report spam |
| Cornell University | Engineering Management, Masters (S20) | Rejected v E-mail on 14 Nov 2019 ♦ | | 2019 | to the rest of you that applied for it :) |
| Georgia | Operations | Other via Other | | 14 Nov | Has any one heard back from GA Tech for the Online Operations Research |

- Degree
- Subject
- Application year
- School name
- Decision
- Decision date
- Notes

# (Jump to script)

```r
# Load libraries and functions.
library(tidyverse)
library(stringdist)
library(fuzzyjoin)
source("scripts/user_defined_functions.R")




##### 1: ANALYZING SCRAPED DATA
#####
#####

# Table 2: admission results table
admissions_results <- read_csv("data/gradcafe_cs_results.csv")

admissions_results
```

```
> admissions_results
# A tibble: 42,613 x 11
   institution        date_added decision decision_date decision_day program   degree   gpa gre_verbal gre_quant gre_awa
   <chr>              <chr>      <chr>    <date>        <chr>        <chr>     <chr>  <dbl>      <dbl>     <dbl>   <dbl>
 1 University Of Sou… 11 Nov 20… Accepted 2019-11-11    Monday       Computer… Maste… NA            NA        NA      NA
 2 University Of Mas… 11 Nov 20… Rejected 2019-11-11    Monday       Computer… Maste… NA            NA        NA      NA
 3 UMass-Amherst      10 Nov 20… Rejected 2019-11-11    Monday       Computer… Maste… NA            NA        NA      NA
 4 Columbia Universi… 10 Nov 20… Other    2019-11-11    Monday       Computer… Maste… NA            NA        NA      NA
 5 University Of Mas… 10 Nov 20… Rejected 2019-11-09    Saturday     Computer… Maste… NA            NA        NA      NA
```

**Exercise:** use `tidyverse`-compliant functions to investigate the admissions results data. **(Jump to script)**

- Compute the mean GPA (0 < gpa <= 4) for PhD applicants only
- Compute the mean *total* GRE scores (130 <= gre_* <= 170) for PhD applicants only
- Print the 50 schools with the most results overall in descending order

# COMBINING DATA SETS

# Best Computer Science Schools

Ranked in 2018, part of Best Science Schools

Earning a graduate degree in computer science can lead to positions in research institutions, government agencies, technology companies and colleges and universities. These are the top computer science schools. Each school's score reflects its average rating on a scale from 1 (marginal) to 5 (outstanding), based on a survey of academics at peer institutions. Read the methodology »

SUMMARY

188 schools | Computer Science Schools × | CLEAR ALL

SORT BY: Rank (high to low)

PROGRAM RANKINGS

Sciences

Computer Science

All Specialties

SCHOOL NAME

School Name

LOCATION

| NAME/RANK | PEER ASSESSMENT SCORE |
|---|---|
| **Carnegie Mellon University**<br>Pittsburgh, PA<br>#1 in Computer Science (tie) | 5.0 |
| **Massachusetts Institute of Technology**<br>Cambridge, MA<br>#1 in Computer Science (tie) | 5.0 |
| **Stanford University**<br>Stanford, CA | 5.0 |

**(Jump to script)**

```r
# Table 1: rankings table
rankings <- read_csv("data/rankings_table.csv")

rankings
```

```
> rankings
# A tibble: 10 x 4
   institution                          usnwr_rank csr_rank nrc_rank
   <chr>                                     <dbl>    <dbl>    <dbl>
 1 Carnegie Mellon University                    1        1        4
 2 Massachusetts Institute of Technology         2        2        2
 3 Stanford University                           3        4        1
 4 University of California, Berkeley            4        5        5
 5 University of Illinois, Urbana-Champaign      5        3        6
 6 Cornell University                            6        7        7
 7 University of Washington                      7        6       24
 8 Georgia Institute of Technology               8       11       12
 9 Princeton University                          9       20        3
10 University of Texas, Austin                  10       17       14
```

**Goal:** we want to combine the rankings from the first table with the admissions results from the second table where the institution name matches into a single table – otherwise known as a **join** or **inner join**.

```
inner_join(x, y, by = NULL, copy = FALSE, suffix = c(".x", ".y"),
  ...)
```

`x`: name of first table

`y`: name of first table

`by`: the name(s) of the **key(s)**, or column with values to be matched between `x` and `y`

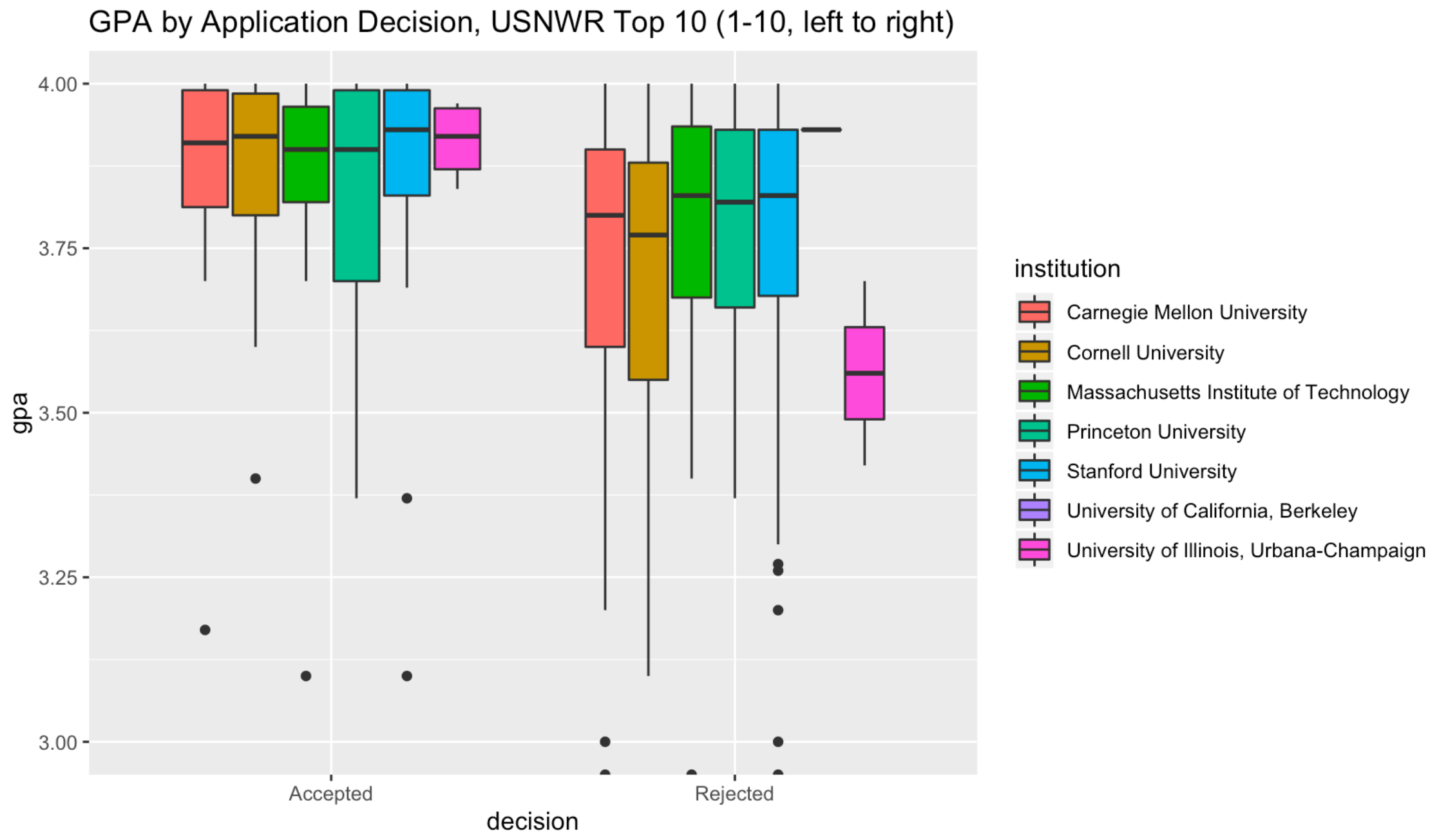
`suffix`: suffix to distinguish non-key columns with names in both `x` and `y`

**(Jump to script)**

```
# INNER JOIN
join_results <- inner_join(rankings, admissions_results,
                           by = "institution")

join_results
```

```
> join_results <- inner_join(rankings, admissions_results,
+            by = "institution")
> join_results
# A tibble: 3,268 x 14
   institution unswr_rank csr_rank nrc_rank date_added decision decision_date decision_day program degree   gpa
   <chr>            <dbl>    <dbl>    <dbl> <chr>      <chr>    <date>        <chr>        <chr>   <chr>  <dbl>
 1 Carnegie M…          1        1        4 3 May 2019 Rejected 2019-02-28    Thursday     Comput… PhD     3.88
 2 Carnegie M…          1        1        4 4 Apr 2019 Rejected 2019-04-04    Thursday     Comput… Maste… NA
 3 Carnegie M…          1        1        4 3 Apr 2019 Accepted 2019-04-03    Wednesday    Comput… Maste… NA
 4 Carnegie M…          1        1        4 3 Apr 2019 Accepted 2019-04-03    Wednesday    Comput… Maste…  3.98
 5 Carnegie M…          1        1        4 2 Apr 2019 Accepted 2019-04-02    Tuesday      Comput… Maste…  3.84
 6 Carnegie M…          1        1        4 2 Apr 2019 Accepted 2019-04-02    Tuesday      Comput… Maste… NA
 7 Carnegie M…          1        1        4 1 Apr 2019 Accepted 2019-04-01    Monday       Comput… Maste… NA
 8 Carnegie M…          1        1        4 28 Mar 20… Other    2019-03-28    Thursday     Comput… Maste… NA
 9 Carnegie M…          1        1        4 27 Mar 20… Other    2019-03-28    Thursday     Comput… Maste… NA
10 Carnegie M…          1        1        4 27 Mar 20… Other    2019-03-27    Wednesday    Comput… Maste… NA
# … with 3,258 more rows, and 3 more variables: gre_verbal <dbl>, gre_quant <dbl>, gre_awa <dbl>
```

# Visualizing the join, we see only 7 institutions matched. What gives?



Let's look at keys containing "Berkeley" in `admissions_results`
**(Jump to script)**:

```r
# Print all institutions in admission_results with substring "Berkeley"
admissions_results %>%
  filter(str_detect(institution, "Berkeley")) %>%
  distinct(institution) %>%
  print(n = nrow(.))
```

We find 38 different keys referencing "Berkeley"! (And these are just the ones that spelled "Berkeley" right *somewhere*.) Here are the first 12:

```
# A tibble: 38 x 1
   institution
   <chr>
 1 University Of California, Berkeley
 2 UC Berkeley
 3 Berkeley
 4 University Of California, Berkeleyley
 5 University Of California Berkeley
 6 Univeristy Of California, Berkeley
 7 UC Berkeley (UCB)
 8 University Of California Berkeley (UCB)
 9 University Of California Berkeley ( UCB )
10 University Of California Berkeley (UC Berkeley)
11 Computer Science Berkeley
12 University Of California, Berkeley (UCB)
```

**Standardization:** we can fix many of the problems with the keys by…

- conversion to lower case
- stripping punctuation
- cleanup of leading/trailing/multiple whitespaces
- removing relatively uninformative words (e.g. words like **university**, **institute**, **school**; also **at** and **of**)
  - why? common/structural words artificially reduce distance of fuzzy mismatches and offer little extra info for true fuzzy matches

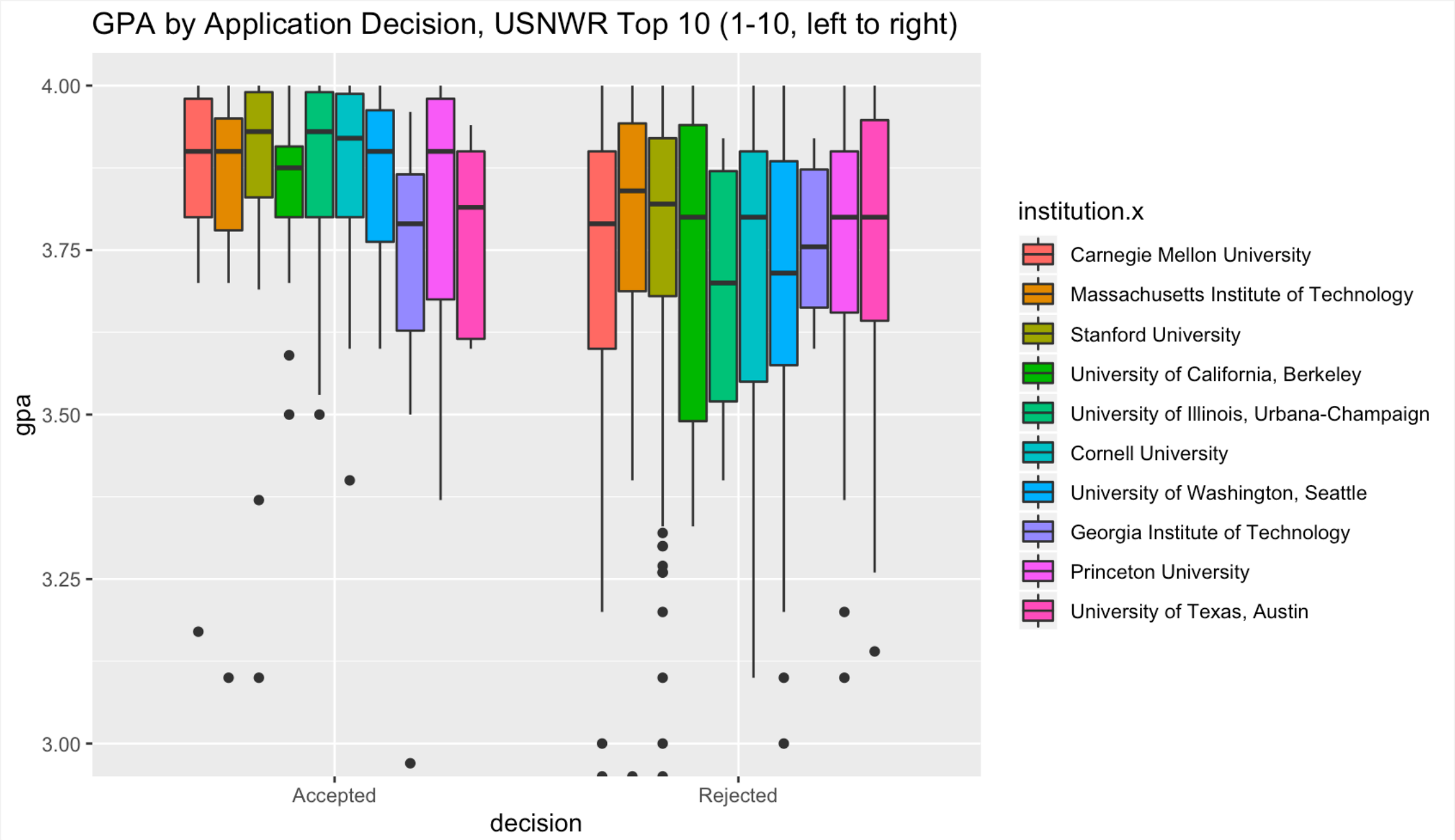
A user-defined function loaded by `source()` does all of this for us.

Adding standardized keys and redoing the join **(Jump to script):**

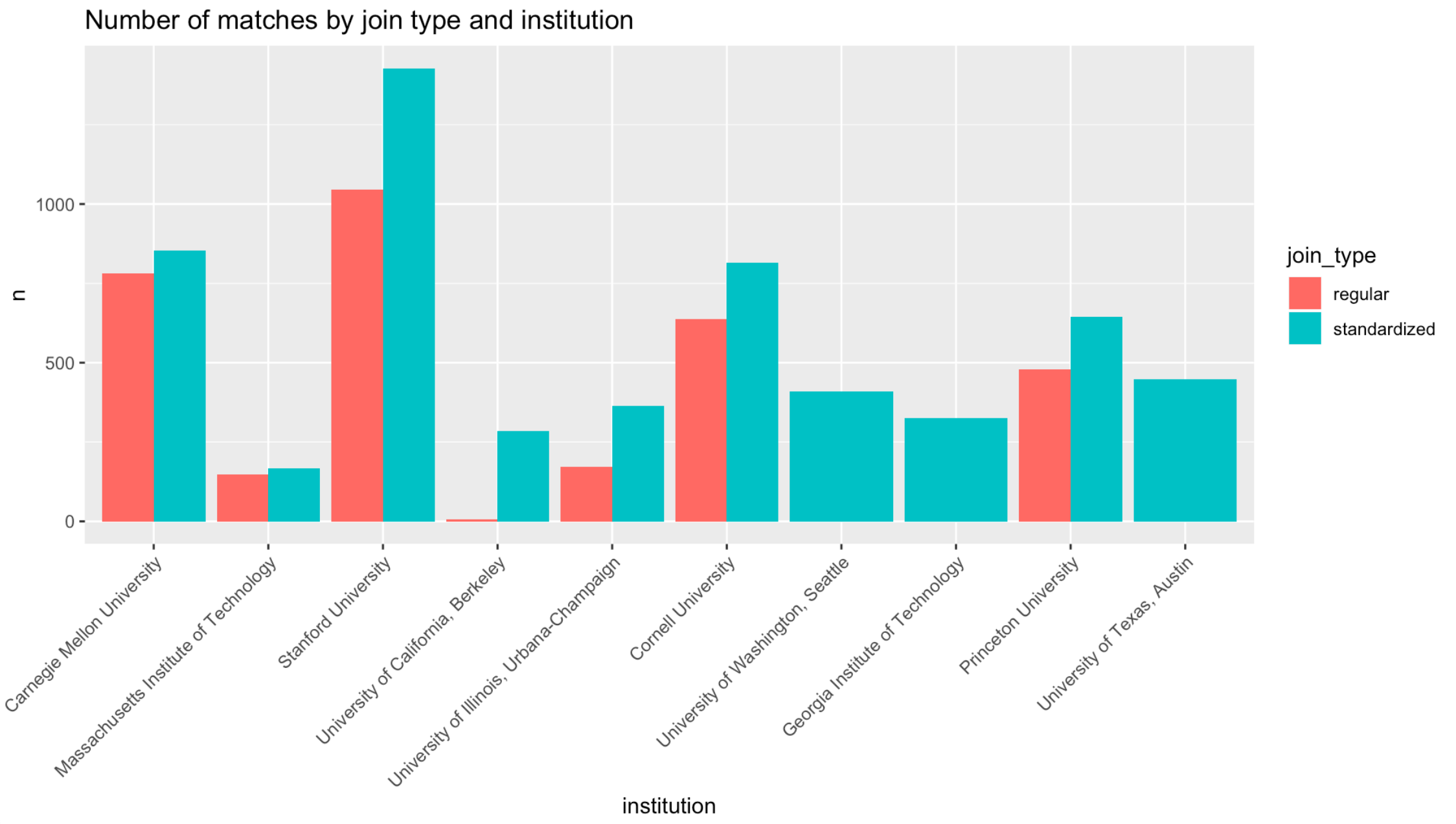
```
# ADD A STANDARDIZED KEY TO BOTH TABLES
rankings_adj <- rankings %>%
  mutate(institution_key = standardize_key(institution))

admissions_results_adj <- admissions_results %>%
  mutate(institution_key = standardize_key(institution))

# INNER JOIN ON STANDARDIZED KEY
join_results_adj <- inner_join(rankings_adj, admissions_results_adj,
                               by = "institution_key")
join_results_adj
```

How many more matches do we see?

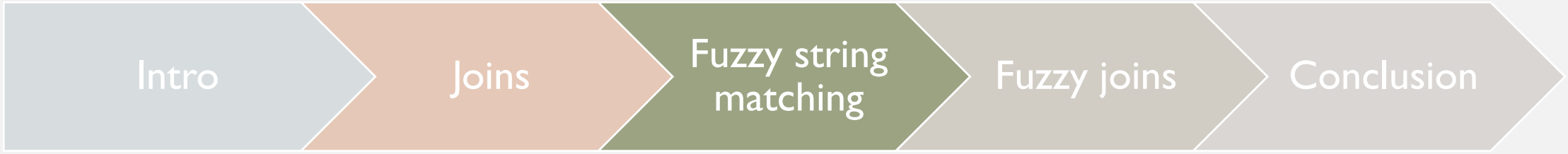
# This looks better – at least all 10 institutions match some results.



# Idiosyncratic number of matches suggests more room for improvement.



# MEASURING CLOSENESS OF STRINGS



The difference between two strings can be measured by a string distance function. These functions come in many different flavors and are useful in different contexts – we'll discuss four of them.

```
stringdist(a, b, method = c("osa", "lv", "dl", "hamming", "lcs", "qgram",
  "cosine", "jaccard", "jw", "soundex"), useBytes = FALSE, weight = c(d
  = 1, i = 1, s = 1, t = 1), q = 1, p = 0, bt = 0,
  nthread = getOption("sd_num_thread"))
```

`a` = first string

`b` = second string

`method` = distance function name

`weight, q, p, bt` = auxiliary arguments for methods

**1. Levenshtein distance:** given two strings, at least how many character insertions, deletions, and substitutions would it take to transform the first string to a second string?

Start with “Northwestern”.

“Northwestern”

distance = **0**

**1. Levenshtein distance:** given two strings, at least how many character insertions, deletions, and substitutions would it take to transform the first string to a second string?

Start with "Northwestern". Substitute **w** for **e**.

"North**w**estern" → "North**e**estern"

distance = **1**

**1. Levenshtein distance:** given two strings, at least how many character insertions, deletions, and substitutions would it take to transform the first string to a second string?

Start with "Northwestern". Substitute **w** for **e**. Substitute **e** for **a**.

"Northwestern" → "Northeestern" → "Northeastern"

distance = **2**

Check for yourself:

```
stringdist("Northwestern", "Northeastern", method = "lv")
```

**2. Levenshtein distance ratio:** given two strings, compute the Levenshtein distance. Then, divide by the length of the longer string.

"Northwestern" → "Northeestern" → "Northeastern"

**2. Levenshtein distance ratio:** given two strings, compute the Levenshtein distance. Then, divide by the length of the longer string.

"Northwestern" → "Northeestern" → "Northeastern" = 2

`str_length`("Northwestern") = `str_length`("Northeastern") = 12

**2. Levenshtein distance ratio:** given two strings, compute the Levenshtein distance. Then, divide by the length of the longer string.

"Northwestern" → "Northeestern" → "Northeastern" = 2

`str_length`("Northwestern") = `str_length`("Northeastern") = 12

so, the distance ratio is 2/12 = **.167**

Check for yourself:

```
lvr("Northwestern", "Northeastern")
```

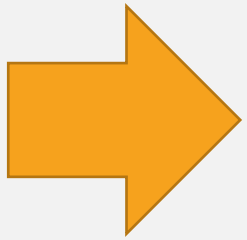
**3. q-gram distance:** given two strings, divide them into *grams* of size *q*, throwing out duplicates. How many grams are *not* shared by the strings?

($q$ = 2)

2-grams

Northwestern

*No*

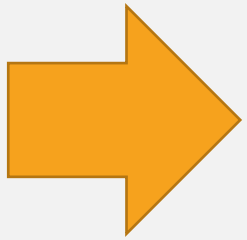


**3. q-gram distance:** given two strings, divide them into *grams* of size *q*, throwing out duplicates. How many grams are *not* shared by the strings?

($q$ = 2)

Northwestern

Northwestern



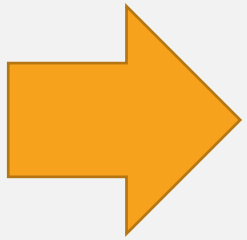
2-grams

*No* *or*

**3. q-gram distance:** given two strings, divide them into *grams* of size *q*, throwing out duplicates. How many grams are *not* shared by the strings?

($q$ = 2)

| | |
|---|---|
| Northwestern | Northwestern |
| Northwestern | Northwestern |
| Northwestern | Northwestern |
| Northwestern | Northwestern |
| Northwestern | Northwestern |
| Northwestern | |

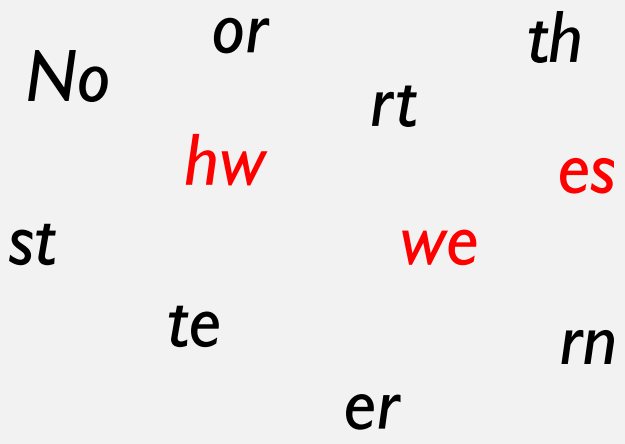


2-grams

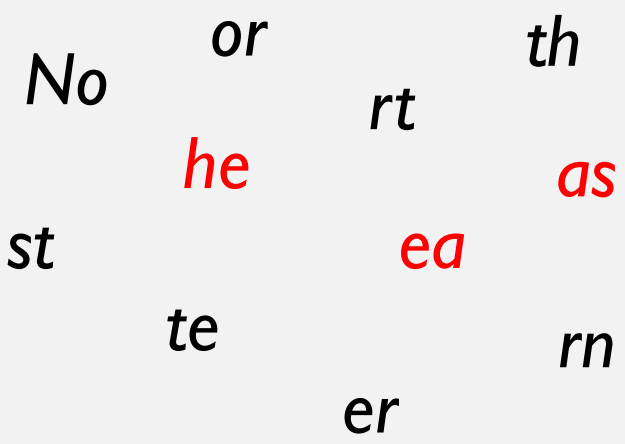
*No* *or* *th* *rt* *hw* *es* *st* *we* *te* *rn* *er*

**3. q-gram distance:** given two strings, divide them into *grams* of size *q*, throwing out duplicates. How many grams are *not* shared by the strings?

Northwestern




Northeastern




distance = **6**

Check for yourself:

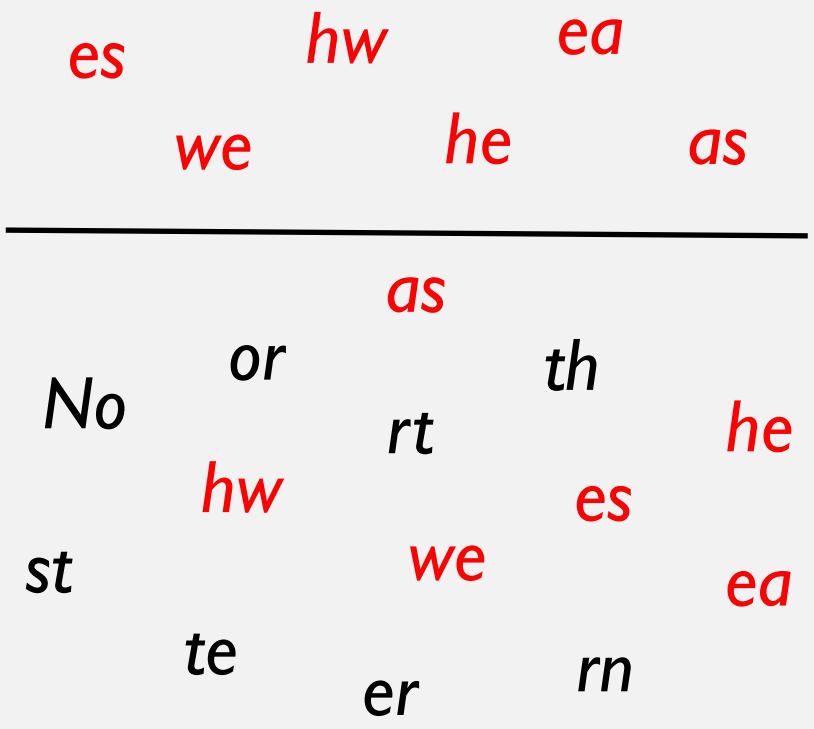
```
stringdist("Northwestern", "Northeastern", method = "qgram", q = 2)
```

**4. Jaccard distance:** given two strings, compute the *q*-gram distance, and divide by the number of unique grams between the two strings.

*es* *hw* *ea*

*we* *he* *as*

**4. Jaccard distance:** given two strings, compute the *q*-gram distance, and divide by the number of unique grams between the two strings.



**4. Jaccard distance:** given two strings, compute the *q*-gram distance, and divide by the number of unique grams between the two strings.

$$\frac{\begin{array}{llll} es & hw & ea & \\ we & he & as & \\ as & & & \end{array}}{\begin{array}{llll} No & or & th & \\ rt & he & & \\ hw & es & & \\ st & we & ea & \\ te & er & rn & \end{array}} = \frac{6}{14} = .429$$

Check for yourself:

```
stringdist("Northwestern", "Northeastern", method = "jaccard", q = 2)
```

**Notes:**

The value of *q* should generally get larger as the keys get longer; q = 2 or 3 suffices for most strings of sentence length or shorter.

Increasing *q* generally means

- distances overall will increase – bigger library of grams, more mismatches
- you care more about the *structure* of the word than letters

If one string pair has a lower distance at *q* = 2 than another, that does not guarantee that pair will have a lower distance at *q* = 3

**Exercise:** find the closest match (smallest distance) among the pairs of strings below by (a) Levenshtein distance ratio; (b) Jaccard distance, q = 2.

**Dartmouth College** and **Darmtouth College**

**University of California** and **The University of California**

**Stanford University** and **University of Stanford**

How do the distances vary between the two metrics?

**Given that these strings are all matches**, do you prefer one metric?

# COMBINING DATA SETS, FUZZILY

Intro | Joins | Fuzzy string matching | Fuzzy joins | Conclusion

```
stringdist_join(x, y, by = NULL, max_dist = 2, method = c("osa",
  "lv", "dl", "hamming", "lcs", "qgram", "cosine", "jaccard", "jw",
  "soundex"), mode = "inner", ignore_case = FALSE,
  distance_col = NULL, ...)
```

`x`: name of first table

`y`: name of first table

`by`: the name(s) of the **key(s)**, or column with values to be fuzzily matched between `x` and `y`

`max_dist`: returns only matches with distance smaller than this threshold

`method`: specifies the string distance function to use

`distance_col`: the name of a column to be added with the key distance for each row

# (Jump to R script)

```
##### 4: COMBINING DATA SETS FUZZILY
#####
#####

# run fuzzy join
fuzzy_join_results <- stringdist_inner_join(rankings_adj,
                                            admissions_results_adj,
                                            by = c("institution_key" = "institution_key"),
                                            method = "jaccard", q = 2, max_dist = .2,
                                            distance_col = "dist")
fuzzy_join_results

# see most different matches first
fuzzy_join_results %>%
  filter(dist > 0) %>%
  group_by(institution.x, institution.y) %>%
  summarize(num_results = n(),
            dist = first(dist)) %>%
  arrange(desc(dist)) %>%
  View()
```

## Selecting a `max_dist`: assessing your appetite for potential mismatches.

(`max_dist` = 0)

(`max_dist` = 1,∞)

# Selecting a `max_dist`: assessing your appetite for potential mismatches.


(`max_dist` = 0) (`max_dist` = 1,∞)

- Only perfect matches
- No mismatches (0 *false positives*, perfect *precision*)

## Selecting a `max_dist`: assessing your appetite for potential mismatches.



(`max_dist` = 0)

- Only perfect matches
- No mismatches (0 *false positives*, perfect *precision*)

(`max_dist` = 1,∞)

- EVERYTHING matches!
- Lots of *false positives*, *perfect recall*

# Selecting a `max_dist`: assessing your appetite for potential mismatches.



(`max_dist` = 0)

- Only perfect matches
- No mismatches (0 *false positives*, perfect *precision*)

- Happy medium!
- Lots of correct matches, few mismatches
- Varies by **user** and **use case**

(`max_dist` = 1,∞)

- EVERYTHING matches!
- Lots of *false positives*, *perfect recall*

A general strategy for finding your happy medium:

1. Start with near-tyranny (low `max_dist`, say .05)
2. Pick a *stopping rule*
3. Increase max_dist a bit, and look at the matches with the highest distance between them. Do they break your stopping rule?

Sample stopping rules:

- *I won't accept any mismatches at all (no risk, lowest reward)*
- *I'll accept at most X mismatches total (low risk, moderate reward)*
- *increasing* `max_dist` *must result in Y times as many true matches than mismatches (medium risk, highest reward)*

# Setting the threshold at 0.200:

| | institution.x | institution.y | num_results | dist |
|---|---|---|---|---|
| 1 | Carnegie Mellon University | Canegie Mellon | 1 | 0.20000000 |
| 2 | Carnegie Mellon University | Carnege Mellon | 1 | 0.20000000 |
| 3 | Carnegie Mellon University | Carnege Mellon University | 24 | 0.20000000 |
| 4 | Carnegie Mellon University | Carnegi Mellon University | 3 | 0.20000000 |
| 5 | University of Illinois, Urbana-Champaign | Illinois At Urbana-Champaign (UIUC)inois At Urbana-... | 5 | 0.20000000 |
| 6 | University of Illinois, Urbana-Champaign | Illinois At Urbana-Champaign (UIUC)s | 1 | 0.20000000 |
| 7 | Princeton University | Princeton U | 2 | 0.20000000 |
| 8 | Massachusetts Institute of Technology | Massachussets Institute Of Technology (MIT) | 6 | 0.19230769 |
| 9 | Georgia Institute of Technology | GaTech (Georgia Institute Of Technology) | 220 | 0.19047619 |
| 10 | Georgia Institute of Technology | Georgia Institute Of Technology - Georgia Tech - GT | 3 | 0.19047619 |
| 11 | Georgia Institute of Technology | Georgia Institute Of Technology ( GaTech ) | 15 | 0.19047619 |
| 12 | Georgia Institute of Technology | Georgia Institute Of Technology (GaTech | 7 | 0.19047619 |
| 13 | Georgia Institute of Technology | Georgia Institute Of Technology (gatech) | 1 | 0.19047619 |
| 14 | Georgia Institute of Technology | Georgia Institute Of Technology (Gatech) | 7 | 0.19047619 |
| 15 | Georgia Institute of Technology | Georgia Institute Of Technology (GaTech) | 163 | 0.19047619 |
| 16 | Georgia Institute of Technology | Georgia Institute Of Technology (GATech) | 4 | 0.19047619 |
| 17 | Georgia Institute of Technology | Georgia Institute Of Technology(GaTech) | 1 | 0.19047619 |
| 18 | Carnegie Mellon University | Carneigie Mellon University | 1 | 0.18750000 |

**Exercise:** find the threshold you are most comfortable with in our fuzzy join problem. You can choose to use one of the example stopping rules, or just come up with your own by feel.

1. Rerun the join with a higher `max_dist` parameter
2. Check the matches with the largest distance using the data viewer
3. If you're still comfortable, repeat steps above and reassess

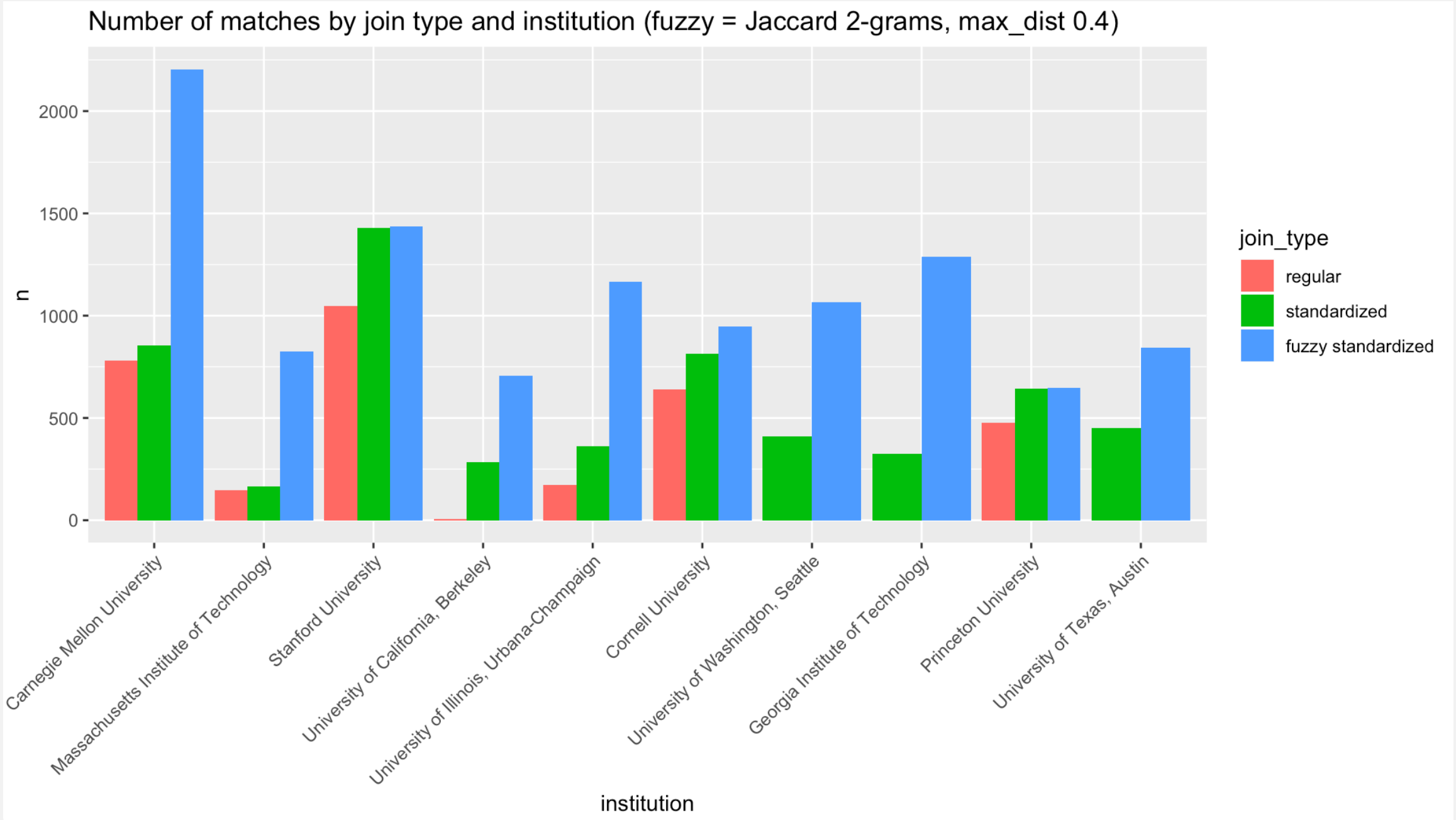
**If you like:** use whichever string distance you prefer. There's no built-in method for a join on Levenshtein ratio; I've written a wrapper that will do the join for you (`lvr_inner_join`), already loaded into your R session.

# Setting the threshold at 0.518:

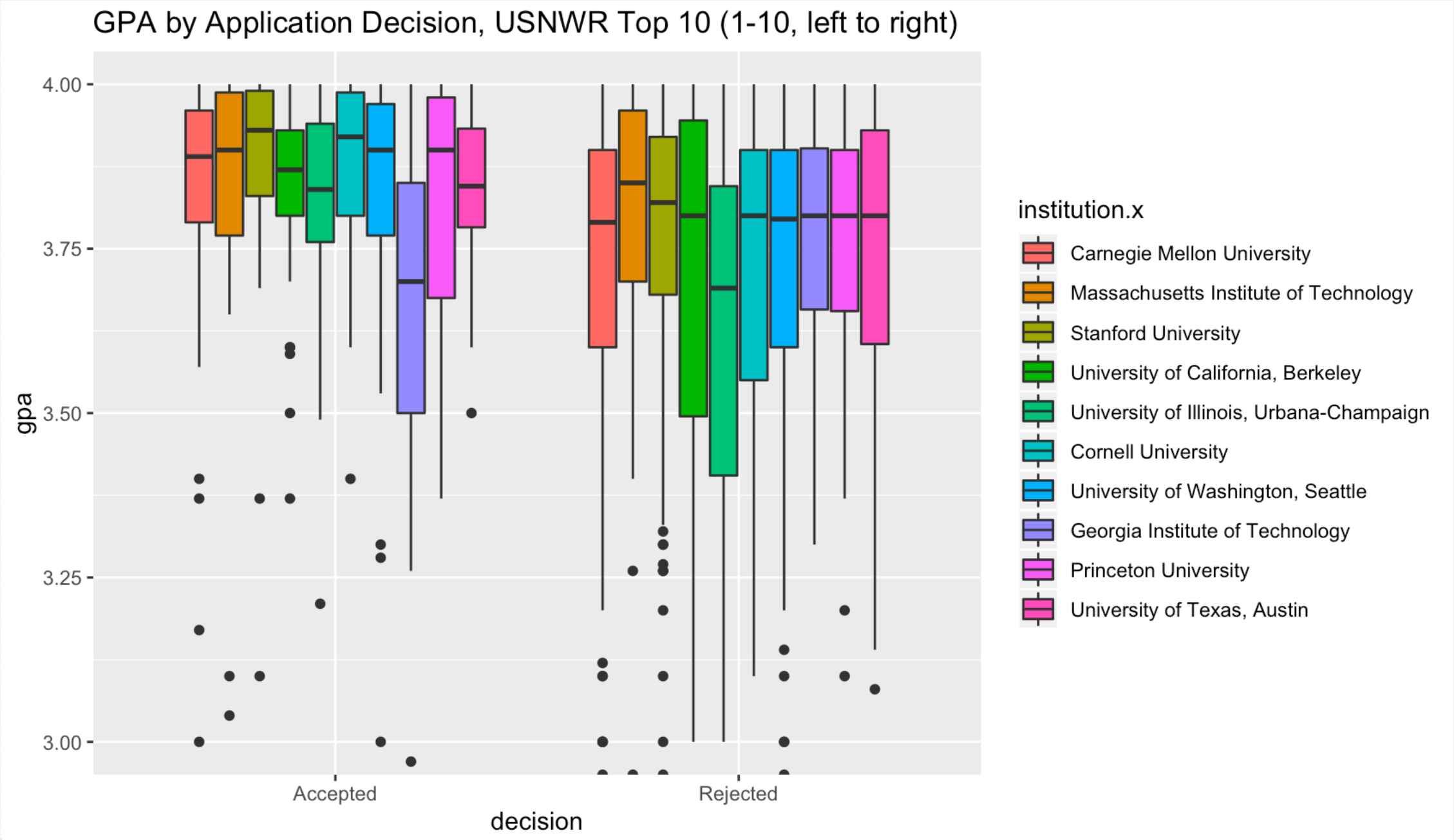
| | | | | |
|---|---|---|---|---|
| 179 | Georgia Institute of Technology | California Institute Of Technology (Caltech) | 68 | 0.5185185 |
| 180 | Carnegie Mellon University | Carnegie Mellon Univeristy (CMU) | 2 | 0.5172414 |
| 181 | Carnegie Mellon University | Carnegie Mellon University (CMU, CMU School Of Music) | 1 | 0.5172414 |
| 182 | Carnegie Mellon University | Carnegie Mellon University (CMU) – Robotics Instituteu | 1 | 0.5172414 |
| 183 | Carnegie Mellon University | Carnagie Mellon University (CMU) LTI | 15 | 0.5000000 |
| 184 | Massachusetts Institute of Technology | University Of Massachusett | 1 | 0.5000000 |
| 185 | University of California, Berkeley | University Of California | 2 | 0.5000000 |
| 186 | University of Washington | University Of Washington Milwaukee | 1 | 0.5000000 |
| 187 | University of Washington | University Of Washingtonhington (Seattle) | 1 | 0.5000000 |
| 188 | University of Washington | Uversity Of Washington | 1 | 0.5000000 |
| 189 | Georgia Institute of Technology | GeorgiaTech | 1 | 0.5000000 |
| 190 | Georgia Institute of Technology | Gorgia Tech | 2 | 0.5000000 |
| 191 | Georgia Institute of Technology | Indian Institute Of Technology | 1 | 0.5000000 |

| | | | | |
|---|---|---|---|---|
| 6 | University of California, Berkeley | UC Berkeley | 365 | 0.6000000 |

# Now we have much more data at minimal cost, and improved parity:



# The colorful fruits of our labor!



# CONCLUSION

# RECAP

Today, we…

- discussed techniques for increasing the number of matches in a join via string distances
- compared a few choices of string distance for this purpose
- tuned parameters to maximize the true match rate while minimizing the false match rate (maximize recall at minimal cost to precision)
- worked through a real example on scraped data

# CHALLENGE EXERCISE

In the `scripts` folder, `challenge_exercise.R` loads the CS rankings and admissions data we just worked with, as well as rankings and scraped admissions data for **statistics** programs.

Try the following:

- use `stringdist_join` to match the each of the top 10 statistics and CS programs with their admissions results
- using `summarize`, compute mean GPA for **accepted PhD applicants** in each discipline and store results in tables
- using `inner_join`, compare institutions with top 10 programs in **both** disciplines – among applicants reporting acceptance, which discipline has a higher average GPA?

# THANKS FOR YOUR TIME! QUESTIONS?