DATA WRANGLING

Hubway dataset

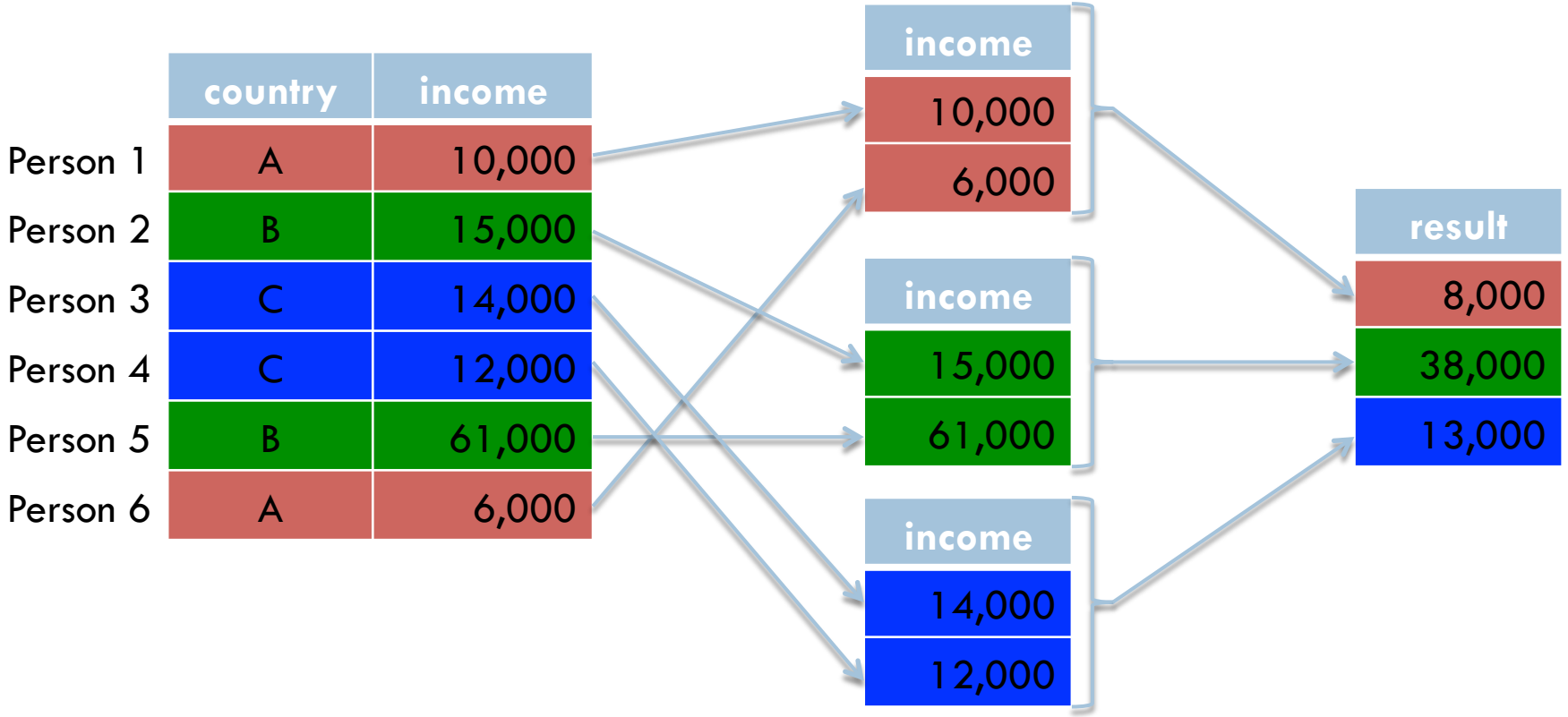
- Hubway is a Boston-area bike share company
- Released dataset as part of visualization challenge
- trips.csv: more than 550,000 bike trips (2011-2012)
  - time/date of start and end, duration
  - start/end station id
  - bike ID
  - zipcode, age, gender for registered users
- stations.csv: id/name/location of 95 stations

R data cleaning functions

- Read in time/date information
  - *strptime*
- Find/replace/manipulate strings
  - *grepl, gsub, strsplit*
- Converting between string, numeric, factor
  - *as.character, as.numeric, as.factor*
  - *factor -> numeric: as.numeric(as.character(fac))*
- Finding missing data
  - *is.na*

tapply() – summarizing by group

- "What is the mean income of each country?"
- tapply(income, country, mean)
- "Group Argument 1 by Argument 2 and apply Argument 3"

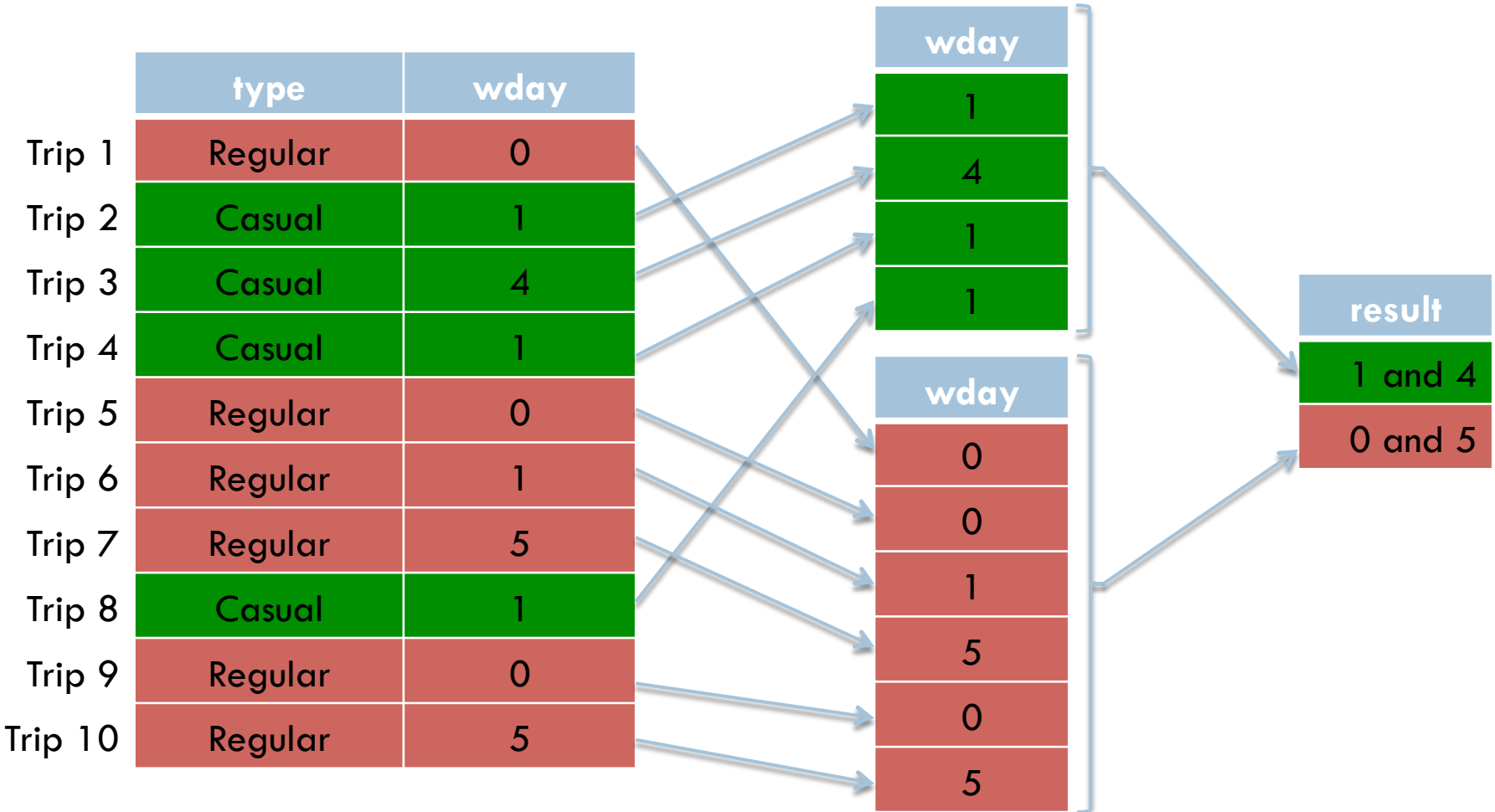


In-Class Exercise 1

- What is the average trip duration by gender?
- What is the sum of trip durations by gender?
- What is the average trip duration by day of the week at the start of the trip?
- What is the average trip duration by the month of the year at the end of the trip?
- Bonus: What is the proportion of users who are casual users by start location? Which start locations have the highest and lowest proportion?
  - Hint 1: Use == instead of = to check equality
  - Hint 2: The mean of TRUE/FALSE values is the proportion that are TRUE.

tapply() for top two days

- tapply(wday, type, get.top.2)



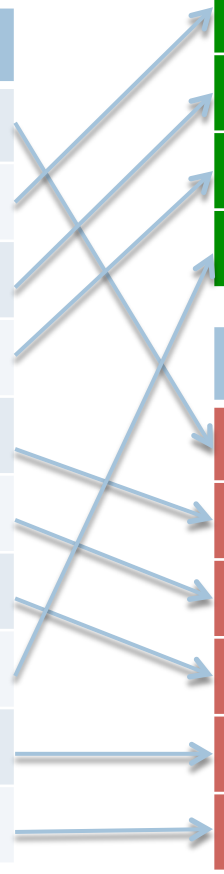
In-Class Exercise 2

- Hubway charges a fee for any trip at least 30 minutes long. Use tapply() to compute the proportion of trips from each start location at least 30 minutes long. (exercise2.R)
- Bonus: compute the most common subscription type (Registered/Casual/Tie) between each start/end location pair (treat A to B as different from B to A). How many of each type of pair are there?
  - Hint 1: check out ?paste for building the vector of start/end pair groupings
  - Hint 2: You can use a structure like if (abc) { return(def) } else if (hij) { return(lmn) } else { return(opq) }

Split-Apply-Combine

- spl = split(trips, station)

station	wday	duration	…
2	0	2500	…
1	1	1200	…
1	4	17	…
1	1	3601	…
2	0	1432	…
2	1	197	…
2	5	26	…
1	1	1494	…
2	0	1201	…
2	5	46	…



station	wday	duration	…
1	1	1200	…
1	4	17	…
1	1	3601	…
1	1	1494	…

station	wday	duration	…
2	0	2500	…
2	0	1432	…
2	1	197	…
2	5	26	…
2	0	1201	…
2	5	46	…

Split-**Apply**-Combine

- spl2 = lapply(spl, get.top.2.df)

station	wday	duration	…
1	1	1200	…
1	4	17	…
1	1	3601	…
1	1	1494	…

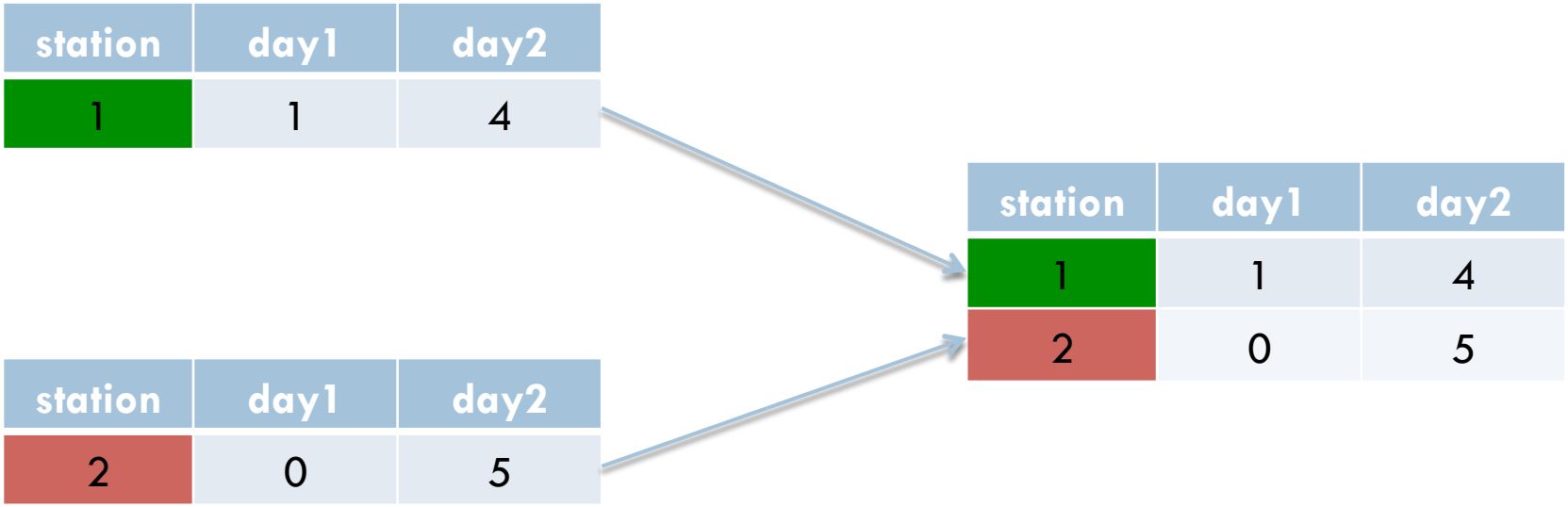
station	wday	duration	…
2	0	2500	…
2	0	1432	…
2	1	197	…
2	5	26	…
2	0	1201	…
2	5	46	…

station	day1	day2
1	1	4

station	day1	day2
2	0	5

Split-Apply-**Combine**

- station.info = rbind(spl2[[1]], spl2[[2]], …)
- station.info = do.call(rbind, spl2)



In-Class Exercise 3

- From trips, create data frame bicycle.info, where each row corresponds to a bicycle. Include the following variables:
  - *bike.nr*: This bike's bike number
  - *mean.duration*: This bike's avg. duration (min.)
  - *sd.duration:* This bike's std. deviation duration (min.)
  - *num.trips:* This number of trips on this bike
- Bonus: Add the following variables:
  - *multi.day:* Number of trips starting on one day and ending on another
  - *common.start*: Most common start location
  - *common.end:* Most common end location

merge() – inner join

- merge(trips, stations, by.x="station", by.y="id")

station	wday	…
2	0	…
1	1	…
1	4	…
1	1	…
2	0	…
2	1	…
2	5	…
1	1	…
2	0	…
2	5	…
4	6	…

id	lat	…
1	42.31	…
2	42.36	…
3	42.35	…

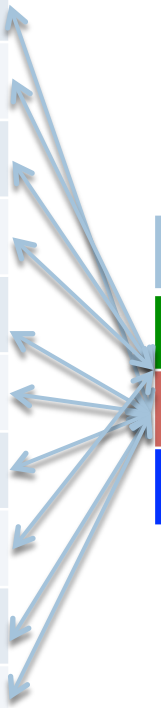
station	wday	lat	…
2	0	42.36	…
1	1	42.31	…
1	4	42.31	…
1	1	42.31	…
2	0	42.36	…
2	1	42.36	…
2	5	42.36	…
1	1	42.31	…
2	0	42.36	…
2	5	42.36	…

merge() – left outer join

- merge(trips, stations, by.x="station", by.y="id", **all.x=TRUE**)

station	wday	…
2	0	…
1	1	…
1	4	…
1	1	…
2	0	…
2	1	…
2	5	…
1	1	…
2	0	…
2	5	…
4	6	…

id	lat	…
1	42.31	…
2	42.36	…
3	42.35	…



station	wday	lat	…
2	0	42.36	…
1	1	42.31	…
1	4	42.31	…
1	1	42.31	…
2	0	42.36	…
2	1	42.36	…
2	5	42.36	…
1	1	42.31	…
2	0	42.36	…
2	5	42.36	…
4	6	NA	…

apply() – operating by row/column

- apply(lat.long, 1, lat.long.dist)
- "Call lat.long.dist() on every row (1) of my matrix lat.long"

lat.x	lat.y	lng.x	lng.y		distance
42.34967	42.34002	-71.07730	-71.10081	→	2.2141
42.34596	42.34002	-71.08258	-71.10081	→	1.6410
42.34391	42.34002	-71.10222	-71.10081	→	0.4475
42.35226	42.34002	-71.12383	-71.10081	→	2.3337
42.33717	42.34002	-71.10280	-71.10081	→	0.3563
42.35099	42.34002	-71.07364	-71.10081	→	2.5487

In-Class Exercise 4

- Hubway charges a variable amount per trip based on duration. Casual users pay the fee from the table, and registered users pay 75% that fee. Add a variable *fee* to trips with each trip's fee

Duration (min)	Fee ($)
[0, 30)	0
[30, 60)	2
[60, 420)	8*floor(min/30)-10
420+	100