Results: 20, 30, or 60

datetime yyyy-mm-dd hr:hr:min:min:sec:sec -08

datetime 2-1 2008-06-20 15:00:00

2008-06-20 15:30:00

= time difference of 30 minutes

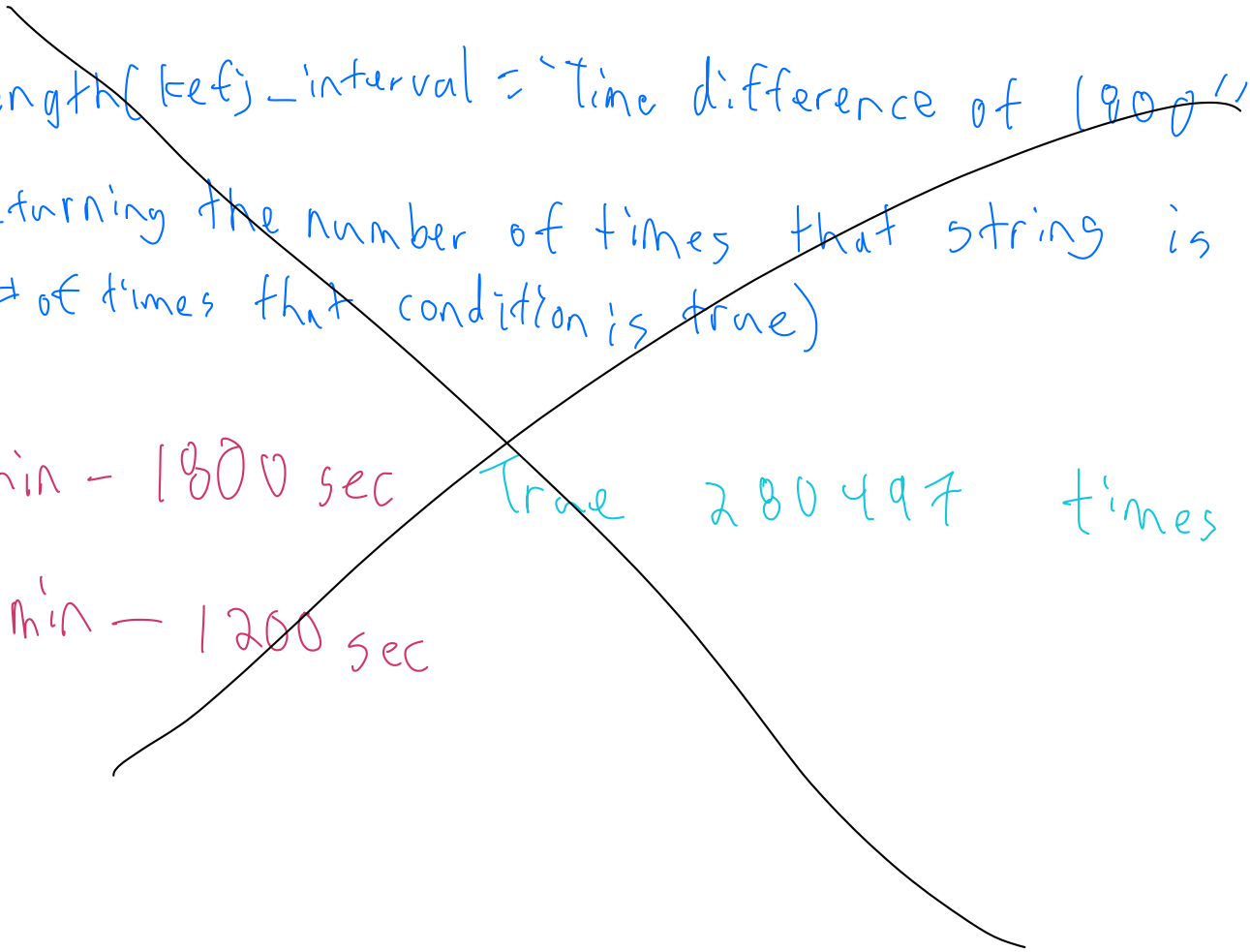
harris_interval: datetime 2-1, 3-2 ... 2187966-2187965

harris_interval ← datetime[-1] - datetime[-2]

= time difference

1) harris_datetime contains only sampling events from harris sites

Hottest



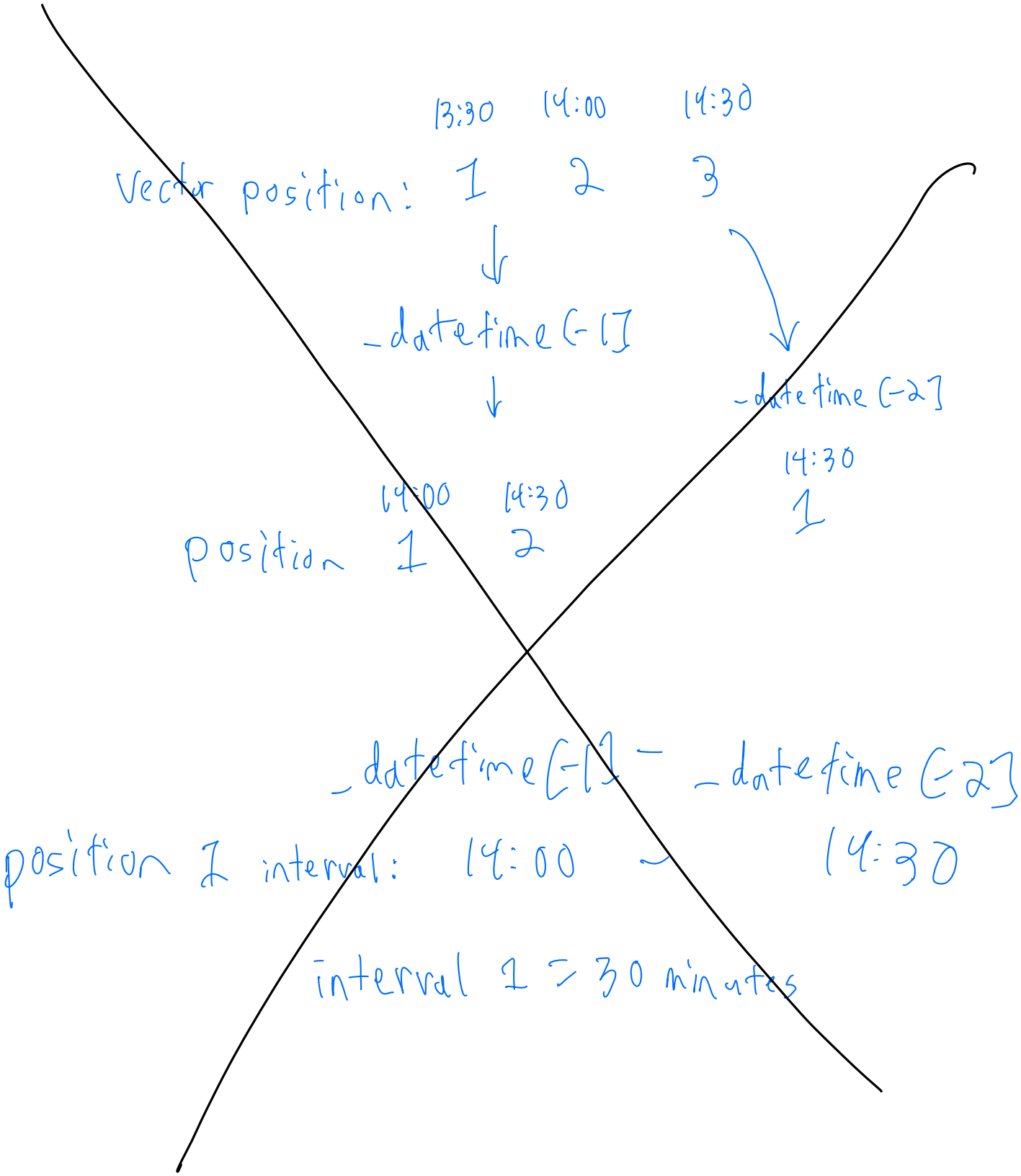
Length(ref)_interval = "time difference of 1800"

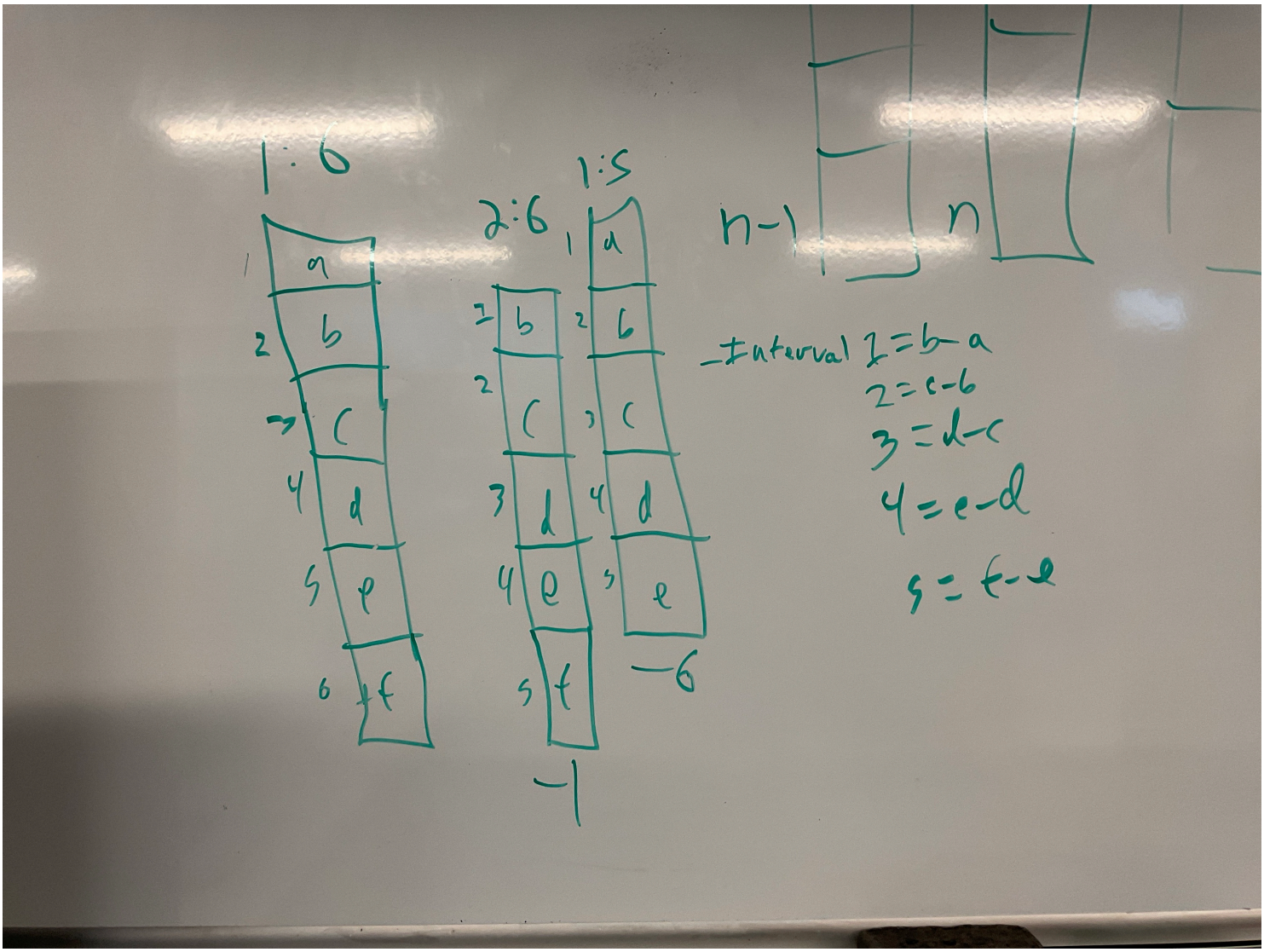
Returning the number of times that string is found
(# of times that condition is true)

30 min - 1800 sec

20 min - 1200 sec

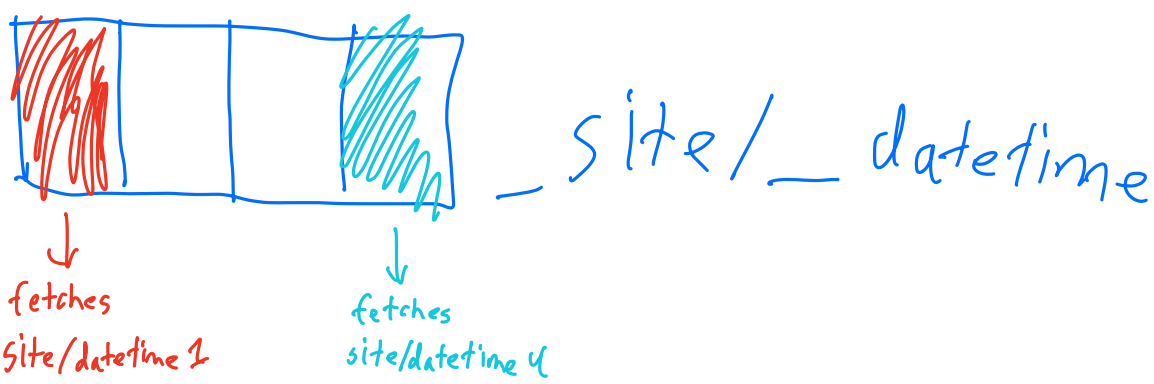
True 280497 times


13:30 14:00 14:30
Vector position: 1 2 3
_datetime[-1]
_datetime[-2]
14:30
1
14:00 14:30
position 1 2
_datetime[-1] - _datetime[-2]
position 1 interval: 14:00 ~ 14:30
interval 1 = 30 minutes


1:6
a
b
c
d
e
f
2:6
b
c
d
e
f
1:5
a
b
c
d
e
-6
-1
n-1
n
Interval 1 = b - a
2 = c - b
3 = d - c
4 = e - d
5 = f - e

① store the out put of which.max(_temp) as a variable. hottest_idx now directs to element 15862 when used to index.

② use that variable to store the hottest time and site



_site(which.min(_temperature)]=_site[4]

A familiar way of thinking about it: f(g(x))

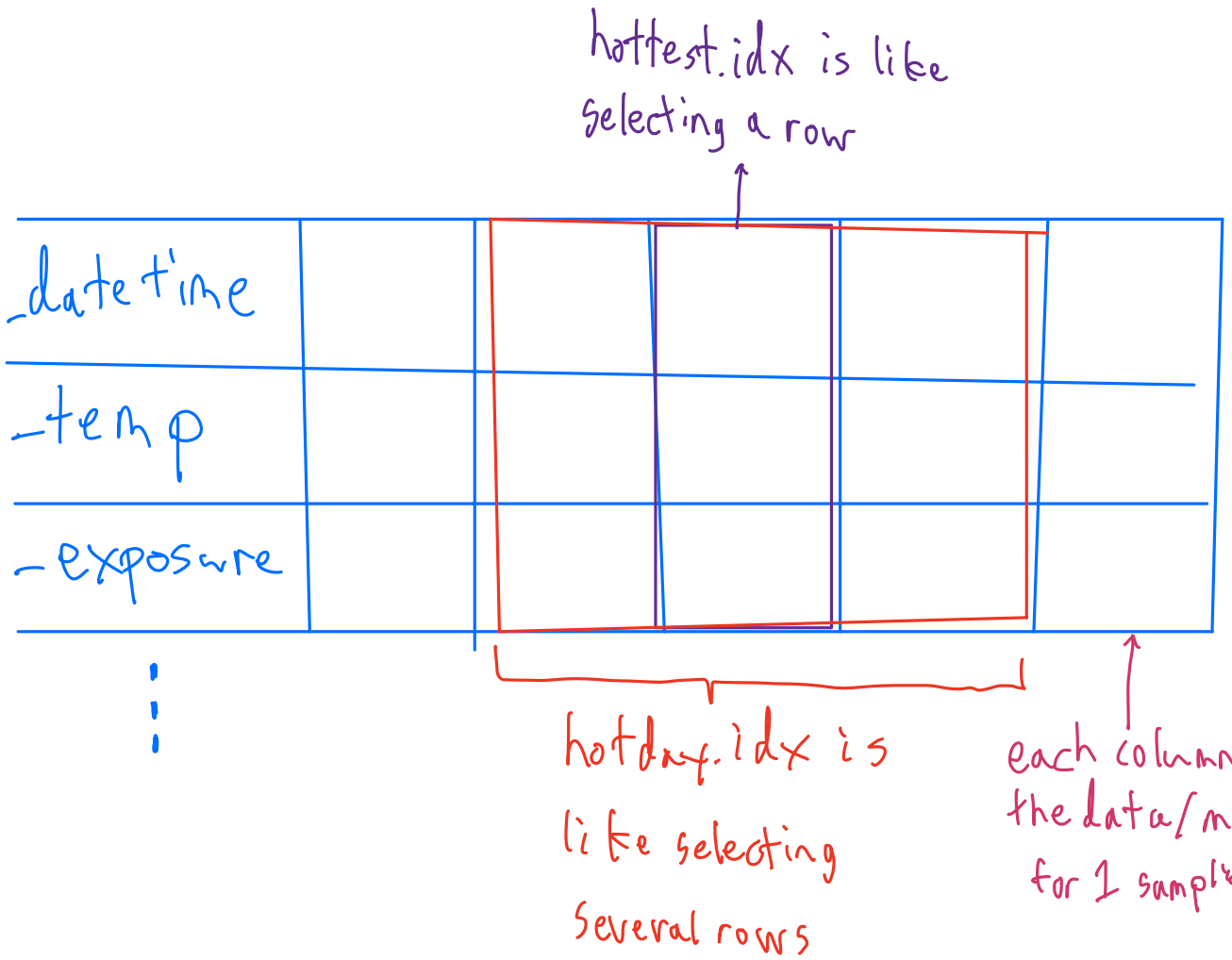
hottest = variables

- idx: element #
- time: time
- site: site name
- time_obj: time as an obj for start & end vars
- start: 12 am of day with hottest measurement
- end: 12 am of next day

Hot day = vectors

Hott day_idx: TRUE FALSE for all elements in our vectors

Vector visualization: Table



.idx → index var/vectors that are used in vector brackets

* V* means vector or variable

.idx

Variable ← function that selects 1 element of a vector → stores the element # of the element selected by the function
↳ variable containing an element of type integer

Vector$_2$ ← vector w/condition(s) → stores the element # where that condition(s) is met. → vector containing 'TRUE' or 'False' for every element in the ogvec

V* ← vector[#/.idx/condition] → stores element(s) of the original vector. → V_2^* contains element(s) the same type as the og vector → eg. decimal #, string, date time, etc.

final step: average # of hours extreme_summer/summer length

* # of days to be averaging over?

of extreme hours

extreme_summer <- extreme_summer/2 convert # of 30 mins into # of hours

extreme_summer <- length(summer_ex_times) Number of extremes at Nuka Bay in the summer

summer_ex_times: kelj_datetime[summer_ex_idx] pulls out every sampling event with extreme temps at Nuka Bay in the summer Try combining these steps by doing <- datetime[c(_idx, _idx]

summer_ex_idx: in summer & at Nuka bay & extreme temp Make an index for extreme sampling events only at Nuka_bay in the summer

summer_idx == 'TRUE' extreme_temp_idx == 'TRUE'

extreme_temp_idx: ≥ 25 or ≤ -4 Make an index for <u>everywhere</u> there's an extreme air temp → this can be reused for the next site/season

(kelj_temperature >= 25 | kelj_temperature <= -4)

& recording air temp

(_exposure == 'air' | _exposure == 'air/transition') → not technically necessary for this data set b/c only air temp are variable enough to meet the first condition anyways but it's good to think about

of days in the season at the site

summer_idx: in summer & at Nuka bay Index for every time both conditions are met

_season == summer _site == Nuka_bay

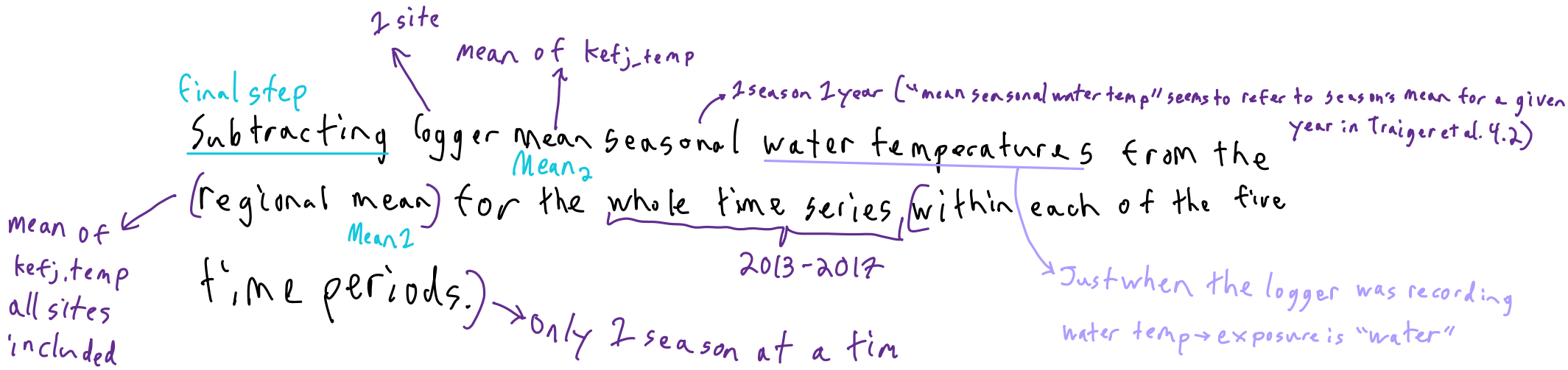
season	Summer	Fall	Summer	Spring
site	Harris	Aialik	Nuka_bay	Nuka bay

Select only row 3 b/c both conditions are met

summer_times: kelj_datetime[summer_idx] pulls out every sampling event at Nuka Bay in the summer These two steps could be replaced with: summer_length <- length(summer_idx == 'TRUE') but i like this more - it's clearer imo

summer_length: length(summer_times) Actual times don't matter, just the # of times there are

summer_length: summer_length/48 turns # of 30 min → # of days



	Mean 1	Mean 2
sites	All kelj	Just 1 site
years	All years w/data	Just 1 year → no idea how to just scoop up the data for 1 year... I've tried.
seasons	Just 1	Just 1
exposure	Just water	Just water

Get the temperature anomolies for one site in one season in one year

kefj_spring: kefj_temp[kefj_season == 'spring' & _exposure == 'water']

harris_spring: kefj_temp[kefj_season == 'spring' &
_exposure == 'water' &
_site == 'Harris' &

year is 2014 Not_datetime == 2018 or 2018* or '2018' or '2018*'

↳ 2. format _datetime so Rstudio recocnizes the elements as times.

2. Extract the years - make a vector that is just the year for every sampling event

3. Make an index vector that is true only when the year vector is 2014

Is there a simpler way to do this? Perhaps, but this is what google had for me.

last step: (mean(kefj_spring)) - (mean(Harris_spring))