Exploring data #1

Data types and vector classes

Data types and vector classes

Here are a few common vector classes in R:

Class	Example
`character`	"Chemistry", "Physics", "Mathematics"
`numeric`	10, 20, 30, 40
`factor`	Male [underlying number: 1], Female [2]
`Date`	"2010-01-01" [underlying number: 14,610]
`logical`	TRUE, FALSE [underlying numbers: 1, 0]

Numeric vectors

To explore numeric vectors, there are a few base R functions that are very helpful. For example:

Function	Description
`min()`	Minimum of values in the vector
`max()`	Maximum of values in the vector
`mean()`	Mean of values in the vector
`median()`	Median of values in the vector

Simple statistic examples

All of these take, as the main argument, the vector(s) for which you want the statistic.

```
mean(x = beijing_pm$value)
```

```
## [1] 63.18646
```

```
min(x = beijing_pm$value)
```

```
## [1] -999
```

If there are missing values in the vector, you'll need to add an option to say what to do when them (e.g., `na.rm` or `use="complete.obs"`—see help files).

Simple statistic examples

These functions require a **numeric vector** as input.

Remember that you can pull a column from a dataframe as a vector using either `$` or the `pluck` function from `purrr`. Therefore, you can use either of these calls to get the mean weight of the children in the dataset:

```
mean(beijing_pm$value)
```

```
## [1] 63.18646
```

```
library("purrr")
```

```
## Warning: package 'purrr' was built under R version
## 3.5.2
```

```
beijing_pm %>%
  pluck("value") %>%
  mean()
```

```
## [1] 63.18646
```

The `summarize` function

Within a "tidy" workflow, you can use the `summarize` function from the `dplyr` package to create summary statistics for a dataframe. This function inputs a dataframe and outputs a dataframe with the specified summary measures.

The `summarize` function

The basic format for using `summarize` is:

```
## Generic code
summarize(dataframe,
          summary_column_1 = function(existing_columns),
          summary_column_2 = function(existing_columns))
```

The `summarize` function

As an example, to summarize the `beijing_pm` dataset to get the minimum, mean, and maximum $PM_{2.5}$ concentrations, you could run:

```
summarize(beijing_pm,
          min_pm = min(value),
          mean_pm = mean(value),
          max_pm = max(value))
```

```
## # A tibble: 1 x 3
##   min_pm mean_pm max_pm
##    <dbl>   <dbl>  <dbl>
## 1   -999    63.2    684
```

Notice that the output is one row (since the summary was on ungrouped data), with three columns (since we defined three summaries in the `summarize` function).

The `summarize` function

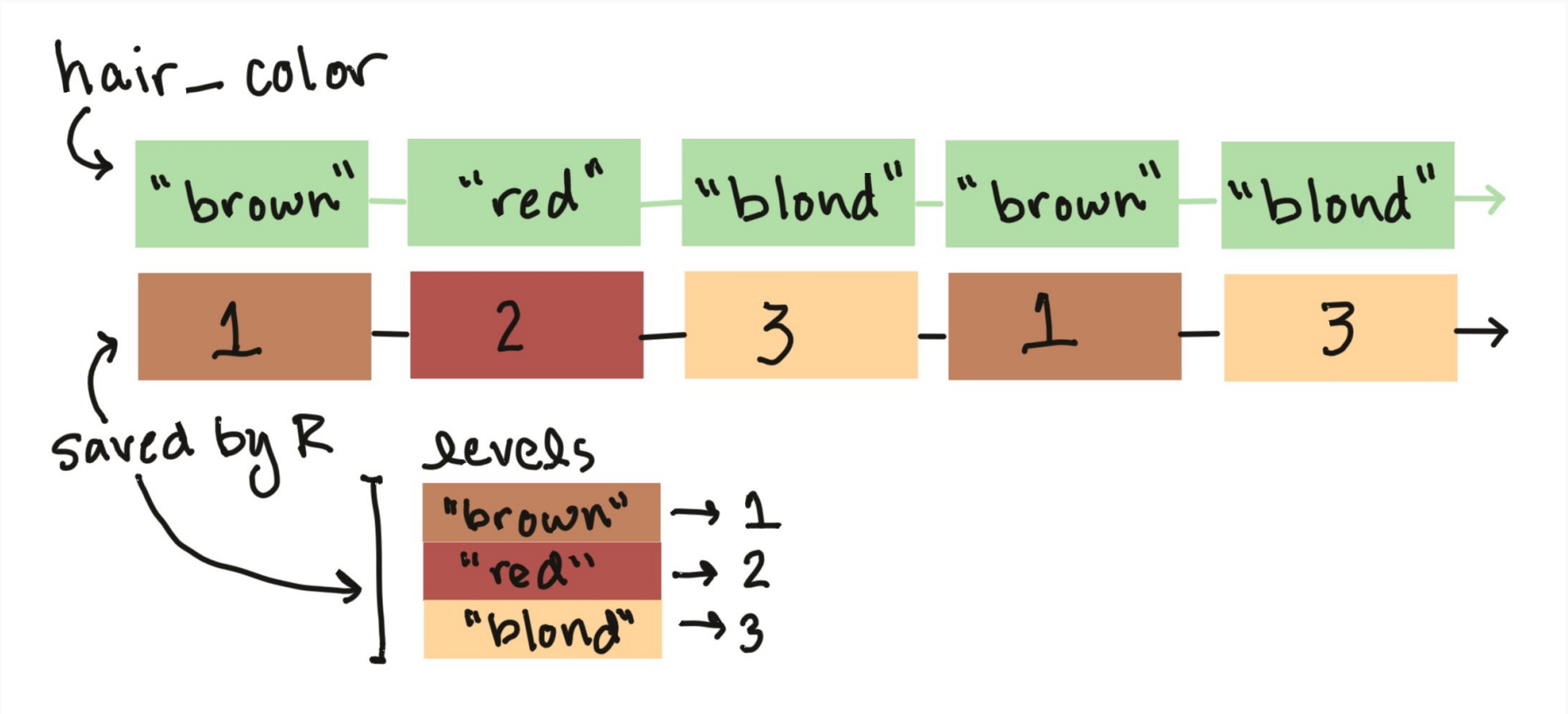
Because the first input to the `summarize` function is a dataframe, you can "pipe into" a `summarize` call. For example, we could have written the code on the previous slide as:

```
beijing_pm %>%
  summarize(min_pm = min(value),
            mean_pm = mean(value),
            max_pm = max(value))
```

As another note, because the output from `summarize` is also a dataframe, we could also "pipe into" another tidyverse function after running `summarize`.

Factors in R

Factor vectors are used in R for **categorical variables**, where more than one observation can have the same category.



Factor variables have one or more **levels**. While you will always see a factor printed with its factor level labels, R "remembers" the variable with each level assigned a number.

Factors in R

In tibbles, factors will be noted with "fctr" under the column name. For example, look at the `aqi` column in the `beijing_pm` data:

```
head(beijing_pm, n = 3)
```

```
## # A tibble: 3 x 4
##   sample_time   value qc    aqi
##   <chr>         <dbl> <chr> <fct>
## 1 1/1/2017 0:00   505 Valid Beyond Index
## 2 1/1/2017 1:00   485 Valid Hazardous
## 3 1/1/2017 2:00   466 Valid Hazardous
```

Factors in R

You can use the `levels` function to see the levels of a factor vector, as well as the order those levels are recorded in R.

```
levels(beijing_pm$aqi)
```

```
## [1] "Good"
## [2] "Moderate"
## [3] "Unhealthy for Sensitive Groups"
## [4] "Unhealthy"
## [5] "Very Unhealthy"
## [6] "Hazardous"
## [7] "Beyond Index"
```

Factors in R

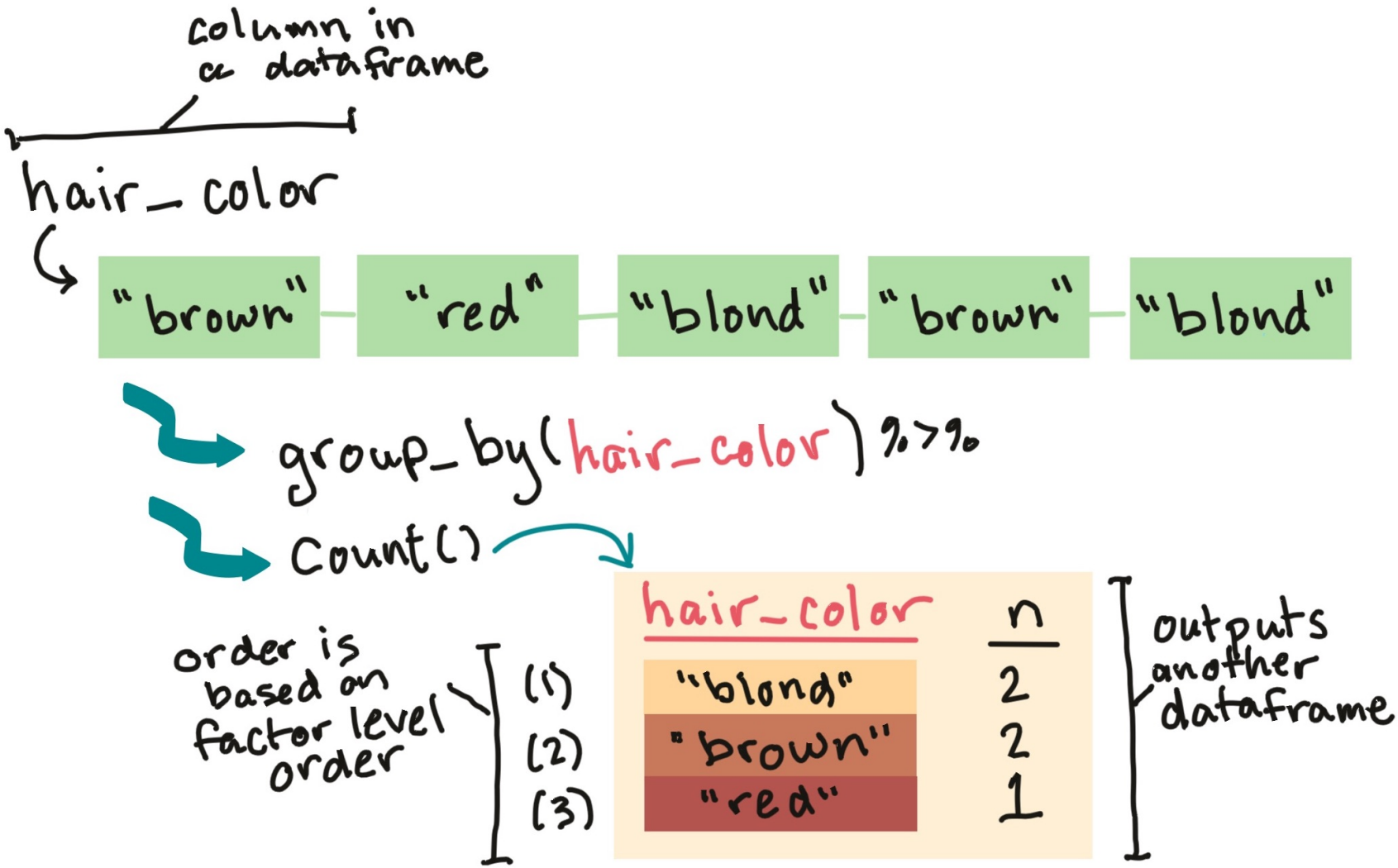
To explore a factor vector, you'll often want to **count** the number of observations in each category. You can do that with two functions in the `dplyr` package, `group_by` and `count`.

Start with a dataframe that includes the factor variable as a column. First `group_by` the factor, then pipe the output of that into the `count` function.

This will create a new summary dataframe, with a row for each level of the factor. A column called `n` will give the number of observations in the original data that had that level of the factor.

Factors in R

You can **count** how many observations have each level of a factor.



Factors in R

```
beijing_pm %>%
  group_by(aqi) %>%
  count()
```

```
## # A tibble: 8 x 2
## # Groups:   aqi [8]
##   aqi                                n
##   <fct>                          <int>
## 1 Good                            2438
## 2 Moderate                        1021
## 3 Unhealthy for Sensitive Groups   374
## 4 Unhealthy                        167
## 5 Very Unhealthy                   179
## 6 Hazardous                        107
## 7 Beyond Index                      27
## 8 <NA>                              31
```

Factors in R

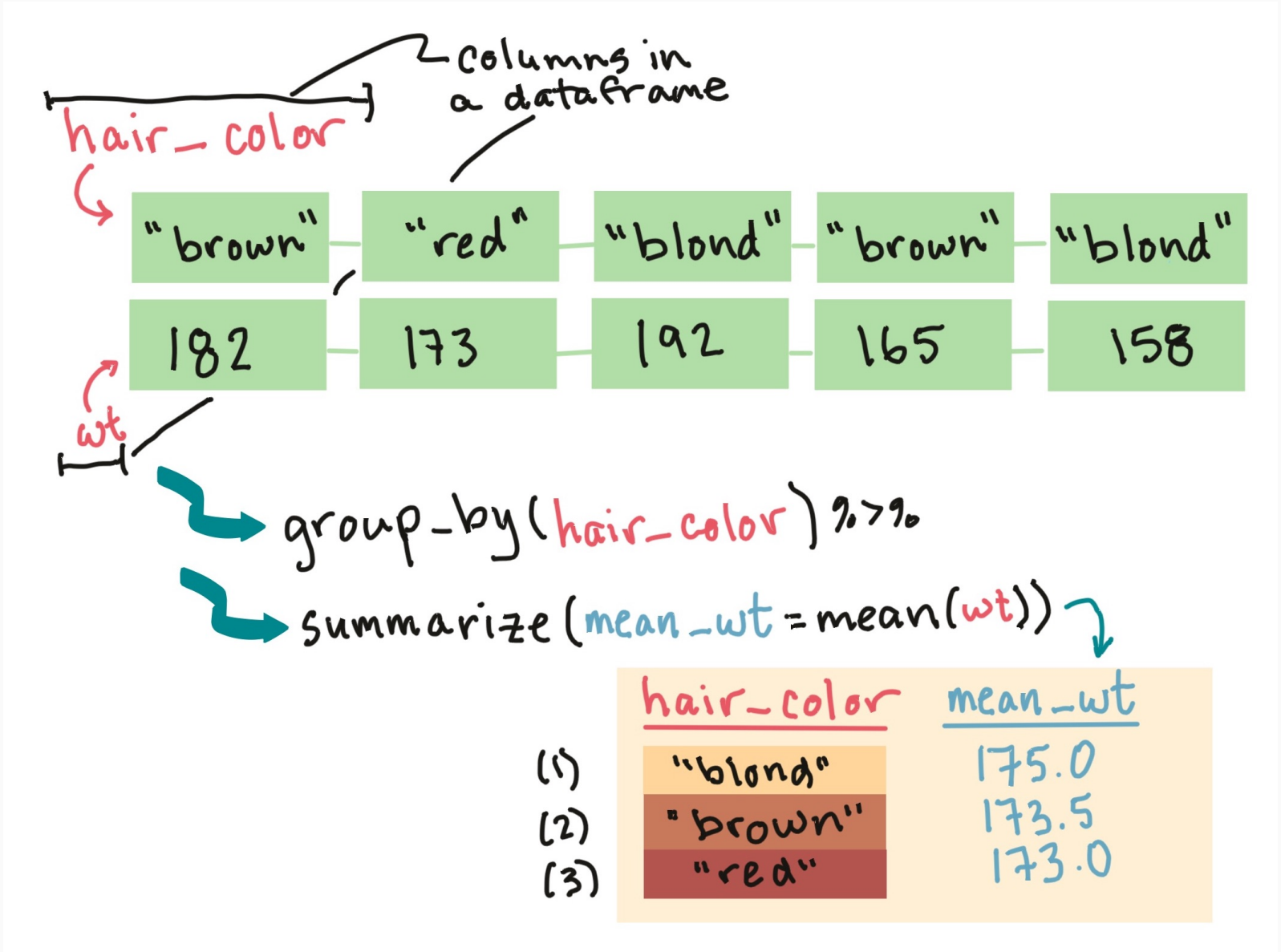
You can jointly explore multiple columns in a dataframe.

For example, if one column is a factor and one is numeric, it can be useful to explore values of the numeric column within each level of the factor column.

For the Beijing data, you may want to find out the mean comcentration of $PM_{2.5}$ within each AQI level.

Factors in R

You can **summarize** a numeric column within levels of a factor column:



Factors in R

To do this, pipe the dataframe into `group_by` (where you can group by the factor column) and then into `summarize`, where you can calculate summaries.

```
beijing_pm %>%
  group_by(aqi) %>%
  summarize(mean_pm = mean(value))
```

```
## # A tibble: 8 x 2
##   aqi                                mean_pm
##   <fct>                                <dbl>
## 1 Good                                  23.5
## 2 Moderate                              70.7
## 3 Unhealthy for Sensitive Groups       122.
## 4 Unhealthy                            172.
## 5 Very Unhealthy                       243.
## 6 Hazardous                            378.
## 7 Beyond Index                         554.
```

Factors in R

You can create several summaries at once:

```
beijing_pm %>%
  group_by(aqi) %>%
  summarize(min_pm = min(value),
            max_pm = max(value))
```

```
## # A tibble: 8 x 3
##   aqi                                min_pm max_pm
##   <fct>                               <dbl>  <dbl>
## 1 Good                                    1     50
## 2 Moderate                               51    100
## 3 Unhealthy for Sensitive Groups        101    150
## 4 Unhealthy                             151    200
## 5 Very Unhealthy                        202    300
## 6 Hazardous                             301    500
## 7 Beyond Index                          505    684
## 8 <NA>                                 -999     -2
```

Factors in R

As a note, there's a function called `n()` that you can use inside summarize to replace `count`. For example, these two expressions give the same output:

```
beijing_pm %>%
  group_by(aqi) %>%
  count()

beijing_pm %>%
  group_by(aqi) %>%
  summarize(n = n())
```

Factors in R

If a column is in a character class, but you'd like it to be a factor, you can use `as.factor`:

```
beijing_pm %>%
  mutate(qc = as.factor(qc))
```

```
## # A tibble: 4,344 x 4
##    sample_time    value qc    aqi
##    <chr>          <dbl> <fct> <fct>
##  1 1/1/2017 0:00    505 Valid Beyond Index
##  2 1/1/2017 1:00    485 Valid Hazardous
##  3 1/1/2017 2:00    466 Valid Hazardous
##  4 1/1/2017 3:00    435 Valid Hazardous
##  5 1/1/2017 4:00    405 Valid Hazardous
##  6 1/1/2017 5:00    402 Valid Hazardous
##  7 1/1/2017 6:00    407 Valid Hazardous
##  8 1/1/2017 7:00    435 Valid Hazardous
##  9 1/1/2017 8:00    472 Valid Hazardous
```