Uncertainty - Tidyverse 1

R data types and code style

Introduction to Quantitative Social Science

Xiaolong Yang

University of Tokyo

June 30, 2022

Today's Game Plan

1. data types: **vector**
2. code style
3. new functions in **Chapter 7: Uncertainty**

- `geom_pointrange()`
- `facet_grid()`

Today's in-class assignment: `china-women`

Section 1

Data types

Visualizing Vectors: *2 types of vector in R*

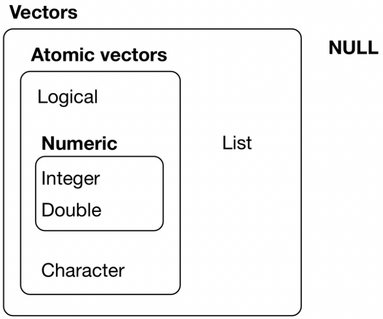


Figure 1: The hierarchy of R's vector types; source: R4DS

Vectors: *2 types of vector in R*

Atomic Vector

- Types
  - logical (TRUE/FALSE)
  - numeric (integer, double)
  - character
- **Homogeneous**: stores only one type of data
- `typeof()` and `length()`

```
x <- c(TRUE, TRUE, FALSE)

typeof(x)
```

```
[1] "logical"
```

```
length(x)
```

```
[1] 3
```

List

- **Heterogeneous**: stores different types of data

```
x <- list(1,
          c(2, 3),
          "QSS",
          list(4, 5))

str(x)
```

```
List of 4
 $ : num 1
 $ : num [1:2] 2 3
 $ : chr "QSS"
 $ :List of 2
  ..$ : num 4
  ..$ : num 5
```

Visualizing Vectors: *2 types of vector in R*

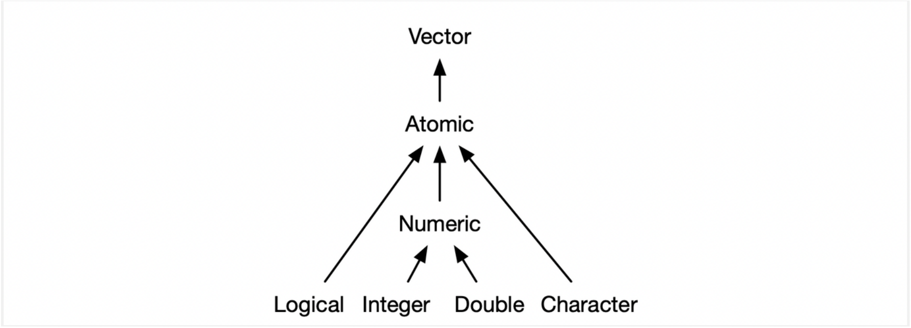


Figure 2: The hierarchy of Atomic vector; source: Advanced R

Visualizing lists

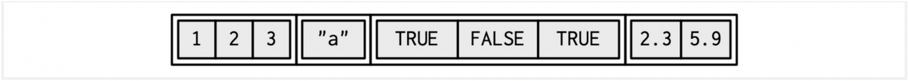


Figure 3: Visualization of a list; source: Advanced R

Test functions

- `in_logical()`
- `is_integer()`
- `is_double()`
- `is_numeric()`
- `is_character()`
- `is_atomic()`
- `is_list()`
- `is_list()`

Good additional resources on R data types by Jenny Bryan **Vectors and lists** and **R objects and indexing**

Data frames/tibbles

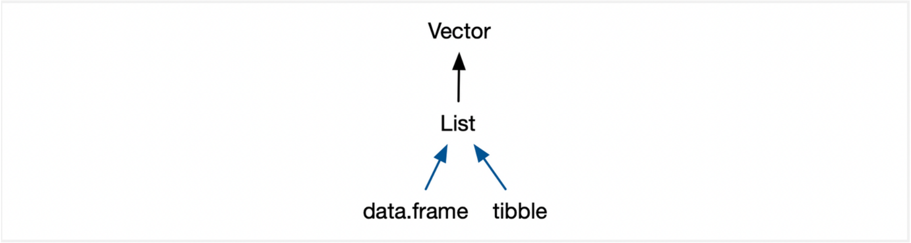


Figure 4: Visualization of data.frame and tibble as lists; source: Advanced R

Data frames/tibbles

A data frame is a **named list**, but all the elements have the same length

- elements in the list = columns
- every column has the same length (number of observations)
- `nth` row = `nth` items from each vector (nth observations)

```
class(FLVoters)
```

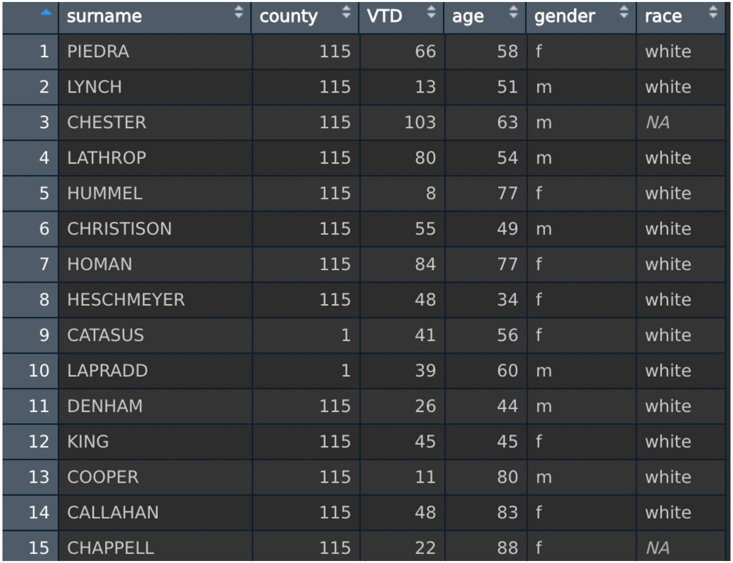
```
[1] "data.frame"
```

```
typeof(FLVoters)
```

```
[1] "list"
```

```
length(FLVoters)
```

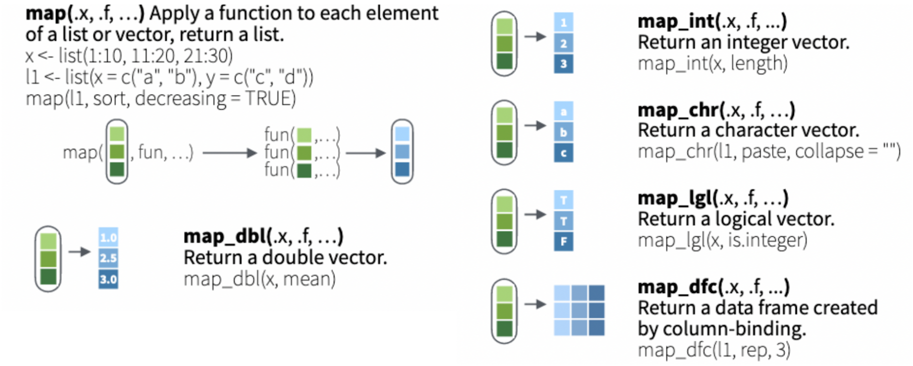
```
[1] 6
```



	surname	county	VTD	age	gender	race
1	PIEDRA	115	66	58	f	white
2	LYNCH	115	13	51	m	white
3	CHESTER	115	103	63	m	*NA*
4	LATHROP	115	80	54	m	white
5	HUMMEL	115	8	77	f	white
6	CHRISTISON	115	55	49	m	white
7	HOMAN	115	84	77	f	white
8	HESCHMEYER	115	48	34	f	white
9	CATASUS	1	41	56	f	white
10	LAPRADD	1	39	60	m	white
11	DENHAM	115	26	44	m	white
12	KING	115	45	45	f	white
13	COOPER	115	11	80	m	white
14	CALLAHAN	115	48	83	f	white
15	CHAPPELL	115	22	88	f	*NA*

Figure 5: View FLVoters in RStudio

purrr map() function revisited



map(.x, .f, …**)** Apply a function to each element of a list or vector, return a list.
x <- list(1:10, 11:20, 21:30)
l1 <- list(x = c("a", "b"), y = c("c", "d"))
map(l1, sort, decreasing = TRUE)

map_dbl(.x, .f, …**)**
Return a double vector.
map_dbl(x, mean)

map_int(.x, .f, ...**)**
Return an integer vector.
map_int(x, length)

map_chr(.x, .f, …**)**
Return a character vector.
map_chr(l1, paste, collapse = "")

map_lgl(.x, .f, …**)**
Return a logical vector.
map_lgl(x, is.integer)

map_dfc(.x, .f, ...**)**
Return a data frame created by column-binding.
map_dfc(l1, rep, 3)

Figure 6: Source: `purrr` cheatsheet

Short summary: data structure in R

Vector as the most basic data type: workhorse of R

- atomic vector
  - **integer**, **double**, **character**, **logical** (raw, complex)
  - **homogeneous**
- list
  - a data frame/tibble as a list of equal-length elements
  - **heterogeneous**

Section 2

Code style

Code style: syntax

- object names: snake_case

```
# Good
day_one
day_1

# Bad
DayOne
dayone
```

Code style: syntax

- spacing: commas, parentheses

```
# Good
x[, 1]
mean(x, na.rm = TRUE)

# Bad
x[,1]
x[ ,1]
x[ , 1]
mean (x, na.rm = TRUE)
mean( x, na.rm = TRUE )
```

Code style: syntax

- infix operators (==, +, −, <−, etc.)

```
# Good
height <- (feet * 12) + inches
mean(x, na.rm = TRUE)

# Bad
height<-feet*12+inches
mean(x, na.rm=TRUE)
```

Code style: syntax

- long lines: 80 characters per line
  - use one line each for the function name, each argument, and the closing

```
# Good
do_something_very_complicated(
  something = "that",
  requires = many,
  arguments = "some of which may be long"
)

# Bad
do_something_very_complicated("that", requires, many, argument
                              "some of which may be long"
                              )
```

Code style: syntax

- assignment

```
# Good
x <- 5

# Bad
x = 5
```

Code style: syntax

- logical vectors

```
# Good
na.rm = TRUE
na.rm = FALSE

# Bad
na.rm = T
na.rm = F
```

Code style: syntax

- quotation marks

```
# Good
"Text"
'Text with "quotes"'

# Bad
'Text'
'Text with "double" and \'single\' quotes'
```

Code style: syntax

- Comments: Each line of a comment begins with # and a single space

```
# regress y on x
fit <- lm(y ~ x, data = df) # why lm()
```

Code style guides

Note

- **Google's R Style Guide**
- **The Tidyverse Style Guide**
- **Computer Programming: Pseudocode by Harvard CS50**

Section 3

New functions in **Chapter 7: Uncertainty**

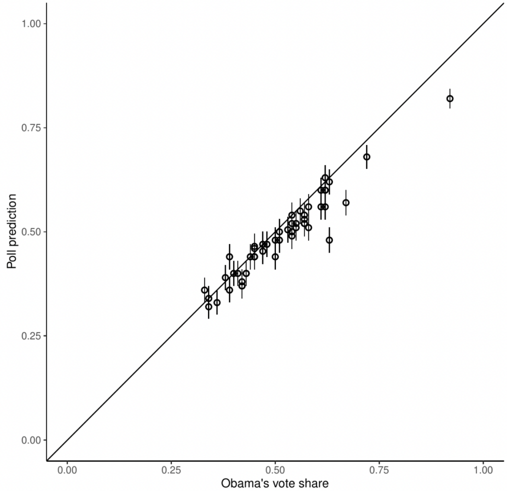
ggplot: geom_pointrange()

- draws points that shows a vertical interval defined by x, ymin and ymax
  - the 95% confidence intervals

```
ggplot(poll_pred, aes(actual, Obama)) +
  geom_abline(intercept = 0,
              slope = 1) +
  geom_pointrange(aes(ymin = ci_lower,
                      ymax = ci_upr)) +
  ...
```

ggplot: geom_pointrange()

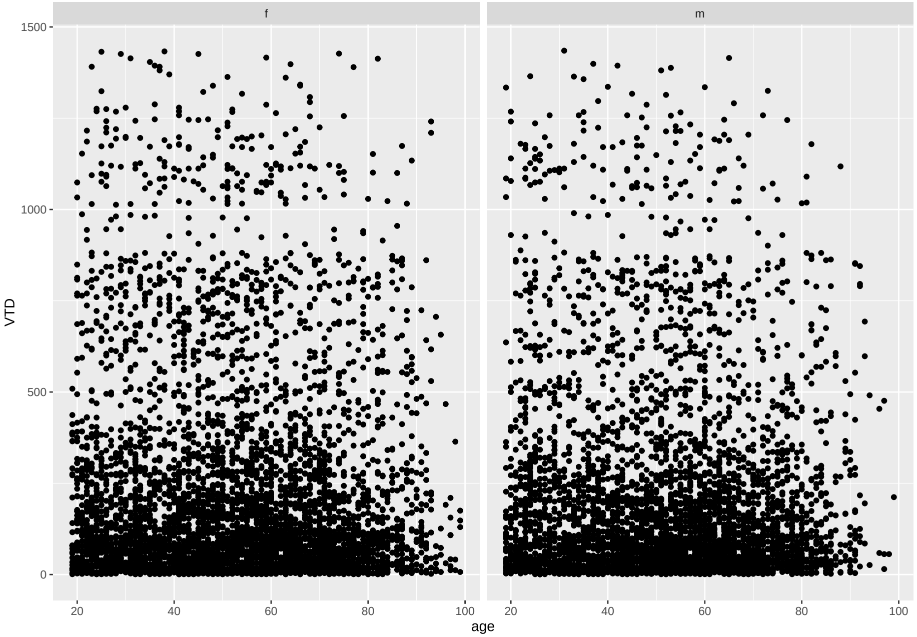
- draws points that show a vertical interval defined by x, ymin and ymax
  - the 95% confidence intervals



ggplot: facet_grid()

- Create separate panels for different class types defined by row and column faceting variables
- facet_grid(. ∼ y)
  - spreads y across columns → comparison of y positions

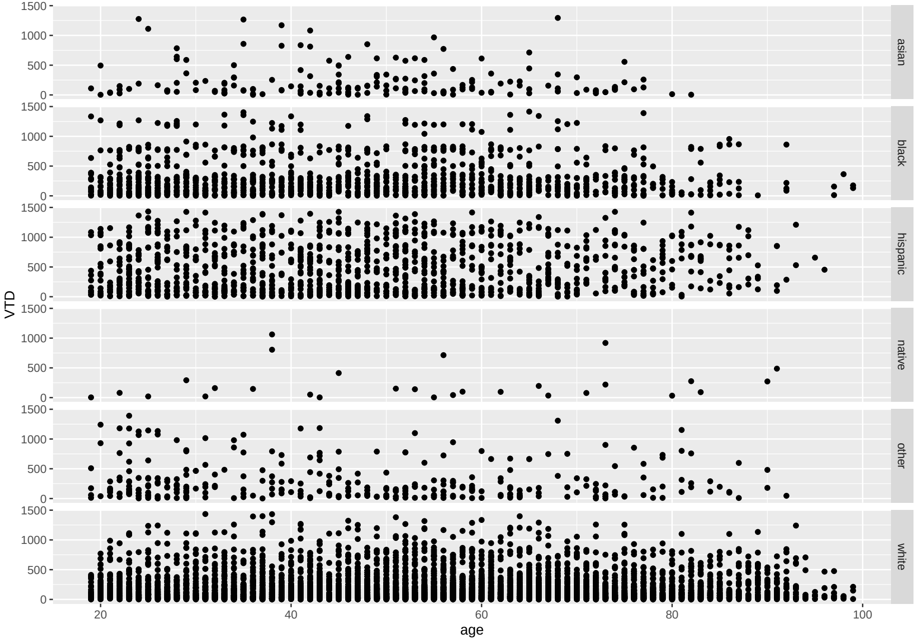
```r
base <- FLVoters %>%
  na.omit() %>%
  ggplot(aes(age, VTD)) +
  geom_point()

base +
  facet_grid(. ~ gender)
```


f
m
VTD
age
1500
1000
500
0
20
40
60
80
100

ggplot: facet_grid()

- Create separate panels for different class types defined by row and column faceting variables
- facet_grid(x ~ .)
    - spreads x across rows → comparison of x positions

```
base +
  facet_grid(race ~ .)
```

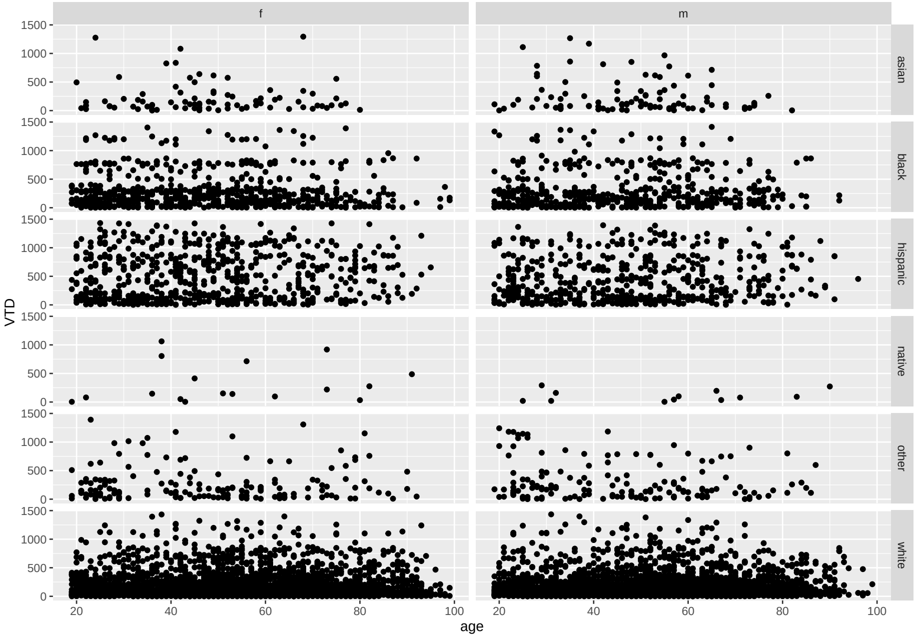


1500
1000
500
0
asian
black
hispanic
native
other
white
VTD
20
40
60
80
100
age

ggplot: facet_grid()

- Create separate panels for different class types defined by row and column faceting variables
- `facet_grid(x ~ y)`

```
base +
  facet_grid(race ~ gender)
```


f
m
asian
black
hispanic
native
other
white
VTD
age
0
500
1000
1500
20
40
60
80
100

What we learnt

- data types: **vector**
- code style
- new `ggplot` functions

Future Game Plan

- reducing duplication: **iteration**
- new functions in **chapter 7: Uncertainty (7.2, 7.3)**