Reading and Writing Data

readr and haven

2024-07-18

readr

Function	Reads
`read_csv()`	Comma separated values
`read_csv2()`	Semi-colon separate values
`read_delim()`	General delimited files
`read_fwf()`	Fixed width files
`read_log()`	Apache log files
`read_table()`	Space separated files
`read_tsv()`	Tab delimited values

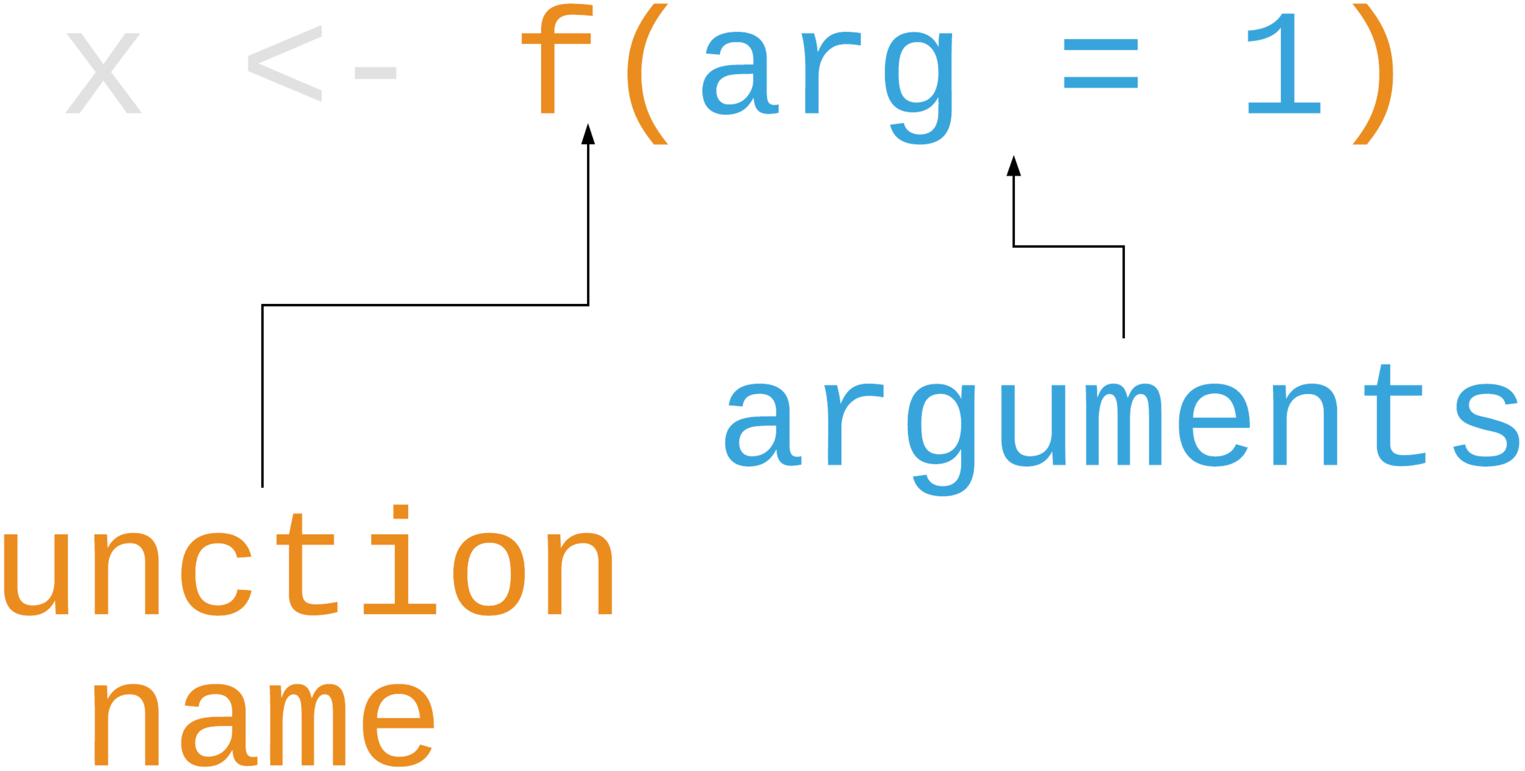
Importing Data

```r
dataset <- read_csv("file_name.csv")
dataset
```

R functions

```r
x <- f(arg = 1)
```

R functions

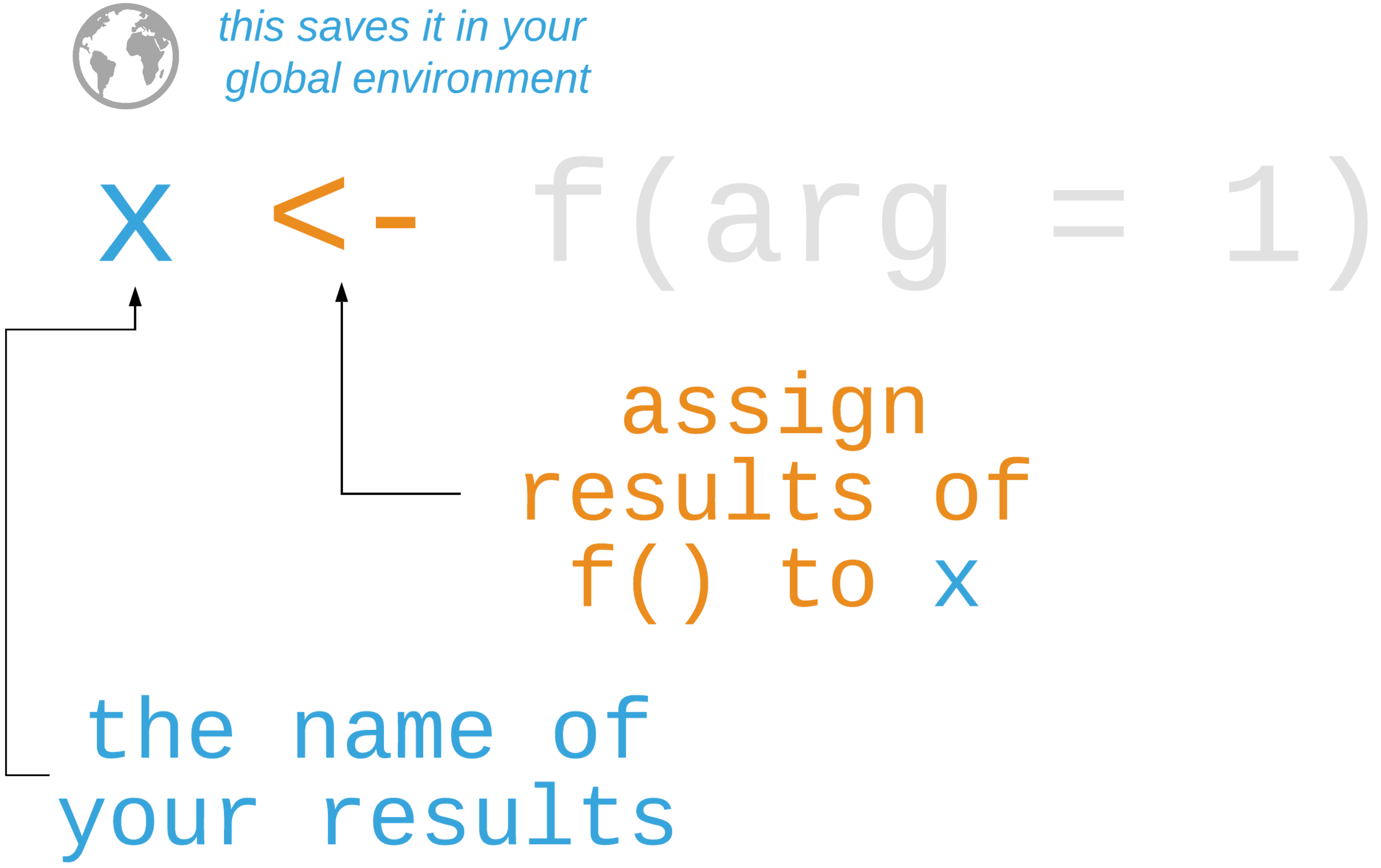


```
x <- f(arg = 1)
```

arguments

function name

R functions



this saves it in your global environment

```
x <- f(arg = 1)
```

assign results of f() to x

the name of your results

Your Turn 1

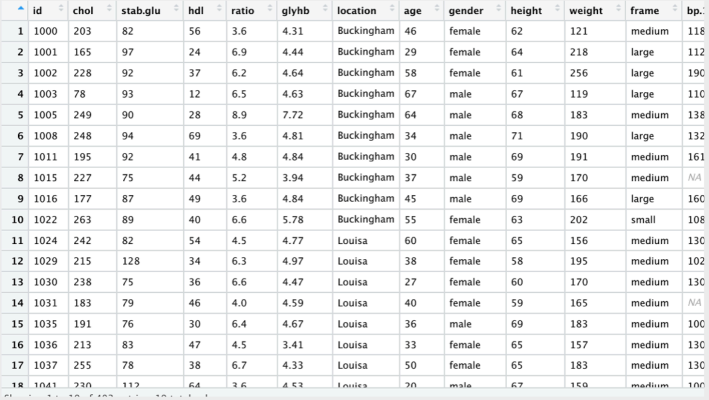
Find `diabetes.csv` on your computer. Then read it into an object. Then view the results.

. . .

```r
diabetes <- read_csv("diabetes.csv")
```

new data alert!

diabetes



	id	chol	stab.glu	hdl	ratio	glyhb	location	age	gender	height	weight	frame	bp.1
1	1000	203	82	56	3.6	4.31	Buckingham	46	female	62	121	medium	118
2	1001	165	97	24	6.9	4.44	Buckingham	29	female	64	218	large	112
3	1002	228	92	37	6.2	4.64	Buckingham	58	female	61	256	large	190
4	1003	78	93	12	6.5	4.63	Buckingham	67	male	67	119	large	110
5	1005	249	90	28	8.9	7.72	Buckingham	64	male	68	183	medium	138
6	1008	248	94	69	3.6	4.81	Buckingham	34	male	71	190	large	132
7	1011	195	92	41	4.8	4.84	Buckingham	30	male	69	191	medium	161
8	1015	227	75	44	5.2	3.94	Buckingham	37	male	59	170	medium	NA
9	1016	177	87	49	3.6	4.84	Buckingham	45	male	69	166	large	160
10	1022	263	89	40	6.6	5.78	Buckingham	55	female	63	202	small	108
11	1024	242	82	54	4.5	4.77	Louisa	60	female	65	156	medium	130
12	1029	215	128	34	6.3	4.97	Louisa	38	female	58	195	medium	102
13	1030	238	75	36	6.6	4.47	Louisa	27	female	60	170	medium	130
14	1031	183	79	46	4.0	4.59	Louisa	40	female	59	165	medium	NA
15	1035	191	76	30	6.4	4.67	Louisa	36	male	69	183	medium	100
16	1036	213	83	47	4.5	3.41	Louisa	33	female	65	157	medium	130
17	1037	255	78	38	6.7	4.33	Louisa	50	female	65	183	medium	130
18	1041	230	112	64	3.6	4.53	Louisa	20	male	67	159	medium	100

Where does it come from?

diabetes.csv (etc)
study: diabetes in African Americans

How can I use it?

```
diabetes <-
 readr::read_csv("diabetes.csv")
View(diabetes)
```

this saves it in your global environment

```r
diabetes
```

```
# A tibble: 403 × 19
      id  chol stab.glu   hdl ratio glyhb location     age
   <dbl> <dbl>    <dbl> <dbl> <dbl> <dbl> <chr>      <dbl>
 1  1000   203       82    56  3.60  4.31 Buckingham    46
 2  1001   165       97    24  6.90  4.44 Buckingham    29
 3  1002   228       92    37  6.20  4.64 Buckingham    58
 4  1003    78       93    12  6.5   4.63 Buckingham    67
 5  1005   249       90    28  8.90  7.72 Buckingham    64
 6  1008   248       94    69  3.60  4.81 Buckingham    34
 7  1011   195       92    41  4.80  4.84 Buckingham    30
 8  1015   227       75    44  5.20  3.94 Buckingham    37
 9  1016   177       87    49  3.60  4.84 Buckingham    45
10  1022   263       89    40  6.60  5.78 Buckingham    55
# ℹ 393 more rows
```

Tibbles

`data.frames` are the basic form of rectangular data in R (columns of variables, rows of observations)

`read_csv()` reads the data into a `tibble`, a modern version of the data frame.

a tibble **is** a data frame

Missing values

It’s common to use codes for **missing values** (-99, 8888)

The na option can change these values to NA

```r
read_csv(
  "a,b,c,d
  1,-99,3,4
  5,6,-99,8",
  na = "-99"
)
```

```
# A tibble: 2 × 4
      a     b     c     d
  <dbl> <dbl> <dbl> <dbl>
1     1    NA     3     4
2     5     6    NA     8
```

Parsing data types

The read functions in readr try to *guess* each data type, but sometimes it’s *wrong*

To tell readr how to parse the columns, add the argument **`col_types`** to `read_csv()`

```r
diabetes <- read_csv(
  "diabetes.csv",
  col_types = list(id = col_character())
)
```

Parsing data types

Or use a string for each variable type: `col_type = "cci"`

Parsing data types

Or use a string for each variable type: col_type = “cci”

letter	type
`c`	character
`i`	integer
`n`	number
`d`	double
`l`	logical
`D`	date
`T`	date time
`t`	time
`?`	guess the type
`_` or `-`	skip the column

Your Turn 2

Set the 4 column types to be: integer, double, character, and unknown (guess)

```r
read_csv(
  "a,b,c,d
  1,2,3,4
  5,6,7,8",
  col_types = ""
)
```

Your Turn 2

Set the 4 column types to be: integer, double, character, and unknown (guess)

```
read_csv(
  "a,b,c,d
  1,2,3,4
  5,6,7,8",
  col_types = "idc?"
)
```

```
# A tibble: 2 × 4
      a     b c         d
  <int> <dbl> <chr> <dbl>
1     1     2 3         4
2     5     6 7         8
```

haven

Function	Software
`read_sas()`	SAS
`read_xpt()`	SAS
`read_spss()`	SPSS
`read_sav()`	SPSS
`read_por()`	SPSS
`read_stata()`	Stata
`read_dta()`	Stata

Heads up!

haven is *not* a core member of the tidyverse. That means you need to load it with `library(haven)`.

Your Turn 3

There are several versions of the diabetes file besides CSV. Pick a file format you or your colleagues use and import them using the corresponding function from haven.

Your Turn 3

```r
library(haven)
diabetes <- read_sas("diabetes.sas7bdat")
```

Your Turn 3

```r
diabetes
```

```
# A tibble: 403 × 19
      id  chol stab_glu   hdl ratio glyhb location     age
   <dbl> <dbl>    <dbl> <dbl> <dbl> <dbl> <chr>      <dbl>
 1  1000   203       82    56  3.60  4.31 Buckingham    46
 2  1001   165       97    24  6.90  4.44 Buckingham    29
 3  1002   228       92    37  6.20  4.64 Buckingham    58
 4  1003    78       93    12  6.5   4.63 Buckingham    67
 5  1005   249       90    28  8.90  7.72 Buckingham    64
 6  1008   248       94    69  3.60  4.81 Buckingham    34
 7  1011   195       92    41  4.80  4.84 Buckingham    30
 8  1015   227       75    44  5.20  3.94 Buckingham    37
 9  1016   177       87    49  3.60  4.84 Buckingham    45
10  1022   263       89    40  6.60  5.78 Buckingham    55
# ℹ 393
```

Writing data

Function	Writes
`write_csv()`	Comma separated values
`write_excel_csv()`	CSV that you plan to open in Excel
`write_delim()`	General delimited files
`write_file()`	A single string, written as is
`write_lines()`	A vector of strings, one string per line
`write_tsv()`	Tab delimited values
`write_rds()`	A data type used by R to save objects
`write_xpt()`	SAS transport format, `.xpt`
`write_sas()`	SAS `.sas7bdat` files (experimental)

Function	Writes
`write_sav()`	SPSS `.sav` files
`write_stata()`	Stata `.dta` files

Writing data

```
write_csv(diabetes, file = "diabetes-clean.csv")
```

Your Turn 4

R has a few data file types, such as RDS and .Rdata. Save `diabetes` as `"diabetes.Rds"`.

. . .

```r
write_rds(diabetes, "diabetes.Rds")
```