Import Data with
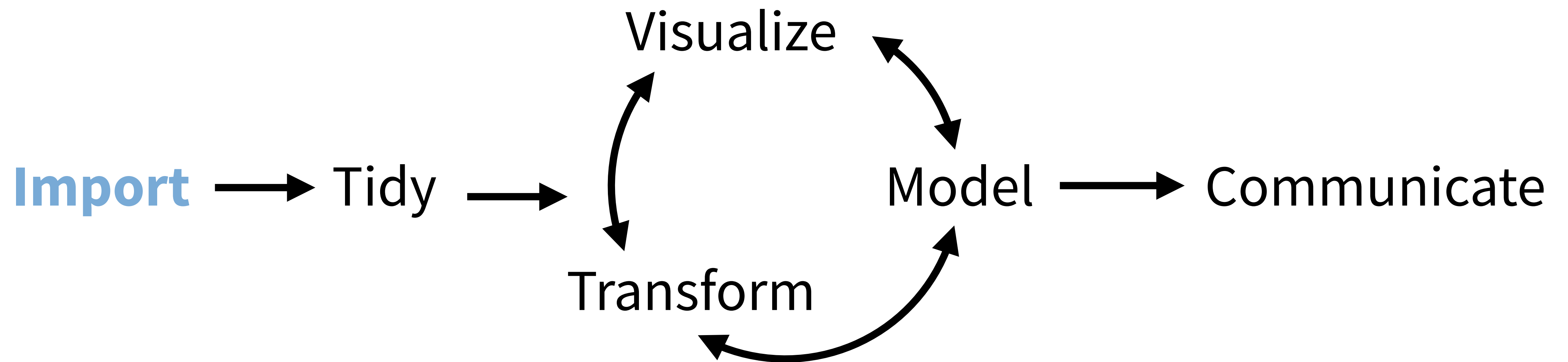
Open **04-Import-Data.Rmd**

(Applied) Data Science

Importing Data

readr

Simple, consistent functions for working with strings.

```
# install.packages("tidyverse")
library(tidyverse)
```

Compared to read.table and its derivatives, readr functions are:

1. ~ 10 times faster

2. Return tibbles

3. Have more intuitive defaults. No row names, no strings as factors.

readr functions

function	reads
read_csv()	Comma separated values
read_csv2()	Semi-comma separated values
read_delim()	General delimited files
read_fwf()	Fixed width files
read_log()	Apache log files
read_table()	Space separated
read_tsv()	Tab delimited values

readr functions

function	reads
read_csv()	**Comma separated values**
read_csv2()	Semi-comma separated values
read_delim()	General delimited files
read_fwf()	Fixed width files
read_log()	Apache log files
read_table()	Space separated
read_tsv()	Tab delimited values

nimbus.csv

```
date,longitude,latitude,ozone
1985-10-01T00:00:00Z,-179.375,-87.5,.
1985-10-01T00:00:00Z,-178.125,-87.5,.
1985-10-01T00:00:00Z,-176.875,-87.5,.
1985-10-01T00:00:00Z,-175.625,-87.5,.
1985-10-01T00:00:00Z,-174.375,-87.5,.
1985-10-01T00:00:00Z,-173.125,-87.5,.
1985-10-01T00:00:00Z,-171.875,-87.5,.
1985-10-01T00:00:00Z,-170.625,-87.5,.
1985-10-01T00:00:00Z,-169.375,-87.5,.
```

nimbus.csv

```
date,longitude,latitude,ozone
1985-10-01T00:00:00Z,-179.375,-87.5,.
1985-10-01T00:00:00Z,-178.125,-87.5,.
1985-10-01T00:00:00Z,-176.875,-87.5,.
1985-10-01T00:00:00Z,-175.625,-87.5,.
1985-10-01T00:00:00Z,-174.375,-87.5,.
1985-10-01T00:00:00Z,-173.125,-87.5,.
1985-10-01T00:00:00Z,-171.875,-87.5,.
1985-10-01T00:00:00Z,-170.625,-87.5,.
1985-10-01T00:00:00Z,-169.375,-87.5,.
```

read_csv()

readr functions share a common syntax

```
df <- read_csv("path/to/file.csv", …)
```

object to save output into

path from working directory to file

Your Turn 1

Find **nimbus.csv** on your server or computer. Then read it into an object. Then view the results.

`02:00`

Your Turn 1

Find **nimbus.csv** on your server or computer. Then read it into an object. Then view the results.
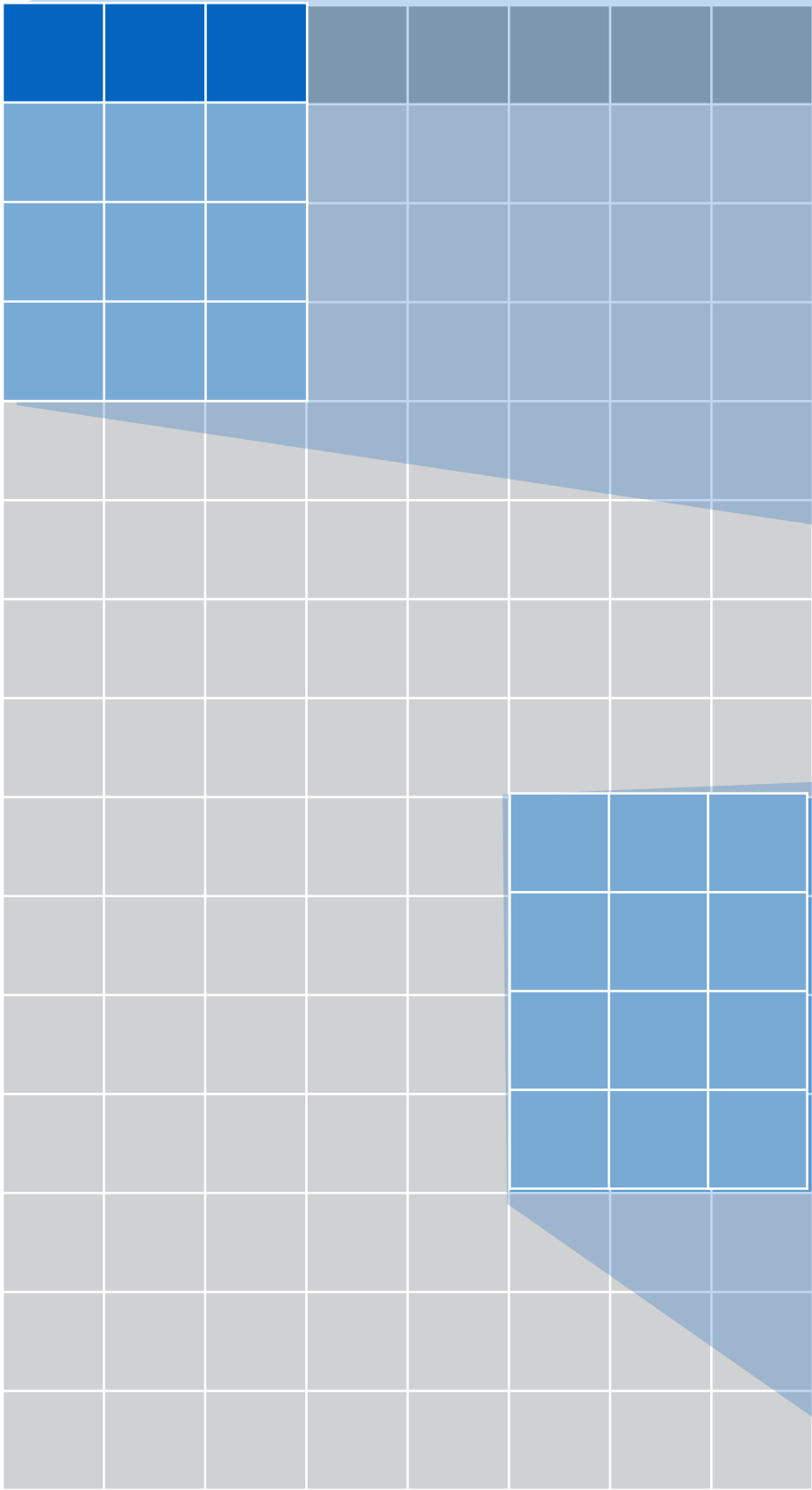
nimbus <- read_csv("nimbus.csv")

nimbus

tibbles

read.csv() vs. read_csv()

```
# A tibble: 234 × 6
    manufacturer       model displ
           <chr>       <chr> <dbl>
1           audi          a4   1.8
2           audi          a4   1.8
3           audi          a4   2.0
4           audi          a4   2.0
5           audi          a4   2.8
6           audi          a4   2.8
7           audi          a4   3.1
8           audi a4 quattro   1.8
9           audi a4 quattro   1.8
10          audi a4 quattro   2.0
# ... with 224 more rows, and 3
#   more variables: year <int>,
#   cyl <int>, trans <chr>
```

tibble display

```
156 1999   6    auto(l4)
157 1999   6    auto(l4)
158 2008   6    auto(l4)
159 2008   8    auto(s4)
160 1999   4 manual(m5)
161 1999   4    auto(l4)
162 2008   4 manual(m5)
163 2008   4 manual(m5)
164 2008   4    auto(l4)
165 2008   4    auto(l4)
166 1999   4    auto(l4)
 [ reached getOption("max.print") --
omitted 68 rows ]
```

data frame display

A large table to display

tibbles

A type of data frame common throughout tidyverse packages. Tibbles enhance data frames in three ways:

1. **Subsetting** - [always returns a new tibble, [[and $ always return a new vector
2. **No partial matching** - You must use full column names when subsetting
3. **Display** - When you print a tibble, R provides a concise view of the data that fits on one screen

tibble

A package with several helper functions for tibbles:

- **as_tibble()** - convert a data frame to a tibble

- **as.data.frame()** - convert a tibble to a data frame

- **tribble()** - make a tibble (transversed)

```
tribble(
   ~x,  ~y,
   1, "a",
   2, "b",
   3, "c")
```

x	y
1	a
2	b
3	c

Tibbles - an enhanced data frame

The **tibble** package provides a new S3 class for storing tabular data, the tibble. Tibbles inherit the data frame class, but improve two behaviors:

- **Display** - When you print a tibble, R provides a concise view of the data that fits on one screen.

- **Subsetting** - [always returns a new tibble, [[and $ always return a vector.

- **No partial matching** - You must use full column names when subsetting

```
# A tibble: 234 × 6
   manufacturer        model displ
          <chr>        <chr> <dbl>
1          audi           a4   1.8
2          audi           a4   1.8
3          audi           a4   2.0
4          audi           a4   2.0
5          audi           a4   2.8
6          audi           a4   2.8
7          audi           a4   3.1
8          audi   a4 quattro   1.8
9          audi   a4 quattro   1.8
10         audi   a4 quattro   2.0
# ... with 224 more rows, and 3
#   more variables: year <int>,
#   cyl <int>, trans <chr>
```

tibble display

A large table to display

```
156 1999      6     auto(l4)
157 1999      6     auto(l4)
158 2008      6     auto(s4)
159 2008      8     auto(s4)
160 1999      4   manual(m5)
161 1999      4     auto(l4)
162 2008      4   manual(m5)
163 2008      4   manual(m5)
164 2008      4     auto(l4)
165 2008      4     auto(l4)
166 1999      4     auto(l4)
 [ reached getOption("max.print")
 -- omitted 68 rows ]
```

data frame display

- Control the default appearance with options:

  options(tibble.print_max = n, tibble.print_min = m, tibble.width = Inf)

- View entire data set with **View**(x, title) or **glimpse**(x, width = NULL, …)

- Revert to data frame with **as.data.frame()** (required for some older packages)

Construct a tibble in two ways

tibble(…)
Construct by columns.
tibble(x = 1:3,
*   y = c("a", "b", "c"))*

Both make this tibble

tribble(…)
Construct by rows.
tribble(
* ~x, ~y,*
*   1, "a",*
*   2, "b",*
*   3, "c")*

```
A tibble: 3 × 2
      x     y
  <int> <dbl>
1     1     1
2     2     b
3     3     c
```

as_tibble(x, …) Convert data frame to tibble.

enframe(x, name = "name", value = "value")
Converts named vector to a tibble with a names column and a values column.

is_tibble(x) Test whether x is a tibble.

Tidy Data with tidyr

Tidy data is a way to organize tabular data. It provides a consistent data structure across packages.
A table is tidy if:

Each variable is in its own column | Each observation, or case, is in its own row | Tidy data | Makes variables easy to access as vectors | Preserves cases during vectorized operations

Reshape Data - change the layout of values in a table

Use **gather()** and **spread()** to reorganize the values of a table into a new layout. Each uses the idea of a key column : value column pair.

gather(data, key, value, …, na.rm = FALSE, convert = FALSE, factor_key = FALSE)

Gather moves column names into a key column, gathering the column values into a single value column.

gather(table4a, '1999', '2000', key = "year", value = "cases")

spread(data, key, value, fill = NA, convert = FALSE, drop = TRUE, sep = NULL)

Spread moves the unique values of a key column into the column names, spreading the values of a value column across the new columns that result.

spread(table2, type, count)

Handle Missing Values

drop_na(data, …)
Drop rows containing NA's in … columns.

drop_na(x, x2)

fill(data, …, direction = c("down", "up"))
Fill in NA's in … columns with most recent non-NA values.

fill(x, x2)

replace_na(data, replace = list(), …)
Replace NA's by column.

replace_na(x, list(x2 = 2), x2)

Expand Tables - quickly create tables with combinations of values

complete(data, …, fill = list())
Adds to the data missing combinations of the values of the variables listed in …

complete(mtcars, cyl, gear, carb)

expand(data, …)
Create new tibble with all possible combinations of the values of the variables listed in …

expand(mtcars, cyl, gear, carb)

Split and Combine Cells

Use these functions to split or combine cells into individual, isolated values.

separate(data, col, into, sep = "[^[:alnum:]]+", remove = TRUE, convert = FALSE, extra = "warn", fill = "warn", …)

Separate each cell in a column to make several columns.

separate(table3, rate, into = c("cases", "pop"))

separate_rows(…, sep = "[^[:alnum:]]+", convert = FALSE)

Separate each cell in a column to make several rows. Also **separate_rows**_().

unite(data, col, …, sep = "_", remove = TRUE)

Collapse cells across several columns to make a single column.

unite(table5, century, year, col = "year", sep = "")

---

country	year	rate
A	1999	0.7K/19M
A	2000	2K/20M
B	1999	37K/172M
B	2000	80K/174M

table3

country	year	cases	pop
A	1999	0.7K	19M
A	2000	2K	20M
B	1999	37K	172
B	2000	80K	174

readr

Parsing

. = NA

nimbus

date <S3: POSIXct>	longitude <dbl>	latitude <dbl>	ozone <chr>
1985-10-01	-179.375	-87.5	.
1985-10-01	-178.125	-87.5	.
1985-10-01	-176.875	-87.5	.
1985-10-01	-175.625	-87.5	.
1985-10-01	-174.375	-87.5	.
1985-10-01	-173.125	-87.5	.
1985-10-01	-171.875	-87.5	.
1985-10-01	-170.625	-87.5	.
1985-10-01	169.375	87.5	.

readr

read_csv()

readr functions share a common syntax

```
nimbus <- read_csv("nimbus.csv", na = ".")
```

object to save output into

path from working directory to file

Value(s) to convert to NA

readr

```
nimbus <- read_csv("nimbus.csv", na = ".")
```

date <S3: POSIXct>	longitude <dbl>	latitude <dbl>	ozone <chr>
1985-10-01	-179.375	-87.5	NA
1985-10-01	-178.125	-87.5	NA
1985-10-01	-176.875	-87.5	NA
1985-10-01	-175.625	-87.5	NA
1985-10-01	-174.375	-87.5	NA
1985-10-01	-173.125	-87.5	NA
1985-10-01	-171.875	-87.5	NA
1985-10-01	-170.625	-87.5	NA
1985-10-01	-169.375	-87.5	NA
1985-10-01	-168.125	-87.5	NA

<chr> stands for character string (not a number)

```

readr
```

read_csv()

readr functions share a common syntax

```
nimbus <- read_csv("nimbus.csv", na = "."),
   col_types = list(ozone = col_double()))
```

Manually specify column types.

list

column name

Column type function

readr

type function	data type
col_character()	character
col_date()	Date
col_datetime()	POSIXct (date-time)
col_double()	double (numeric)
col_factor()	factor
col_guess()	let readr guess (default)
col_integer()	integer
col_logical()	logical
col_number()	numbers mixed with non-number characters
col_numeric()	double or integer
col_skip()	do not read
col_time()	time

type function	data type
col_character()	character
col_date()	Date
col_datetime()	POSIXct (date-time)
col_double()	**double (numeric)**
col_factor()	factor
col_guess()	let readr guess (default)
col_integer()	integer
col_logical()	logical
col_number()	numbers mixed with non-number characters
col_numeric()	double or integer
col_skip()	do not read
col_time()	time

```
nimbus <- read_csv("nimbus.csv", na = ".",
  col_types = list(ozone = col_double()))

library(viridis)
world <- map_data(map = "world")
nimbus %>%
  ggplot() +
    geom_point(aes(longitude, latitude, color = ozone)) +
    geom_path(aes(long, lat, group = group), data = world) +
    coord_map("ortho", orientation=c(-90, 0, 0)) +
    scale_color_viridis(option = "A")
```
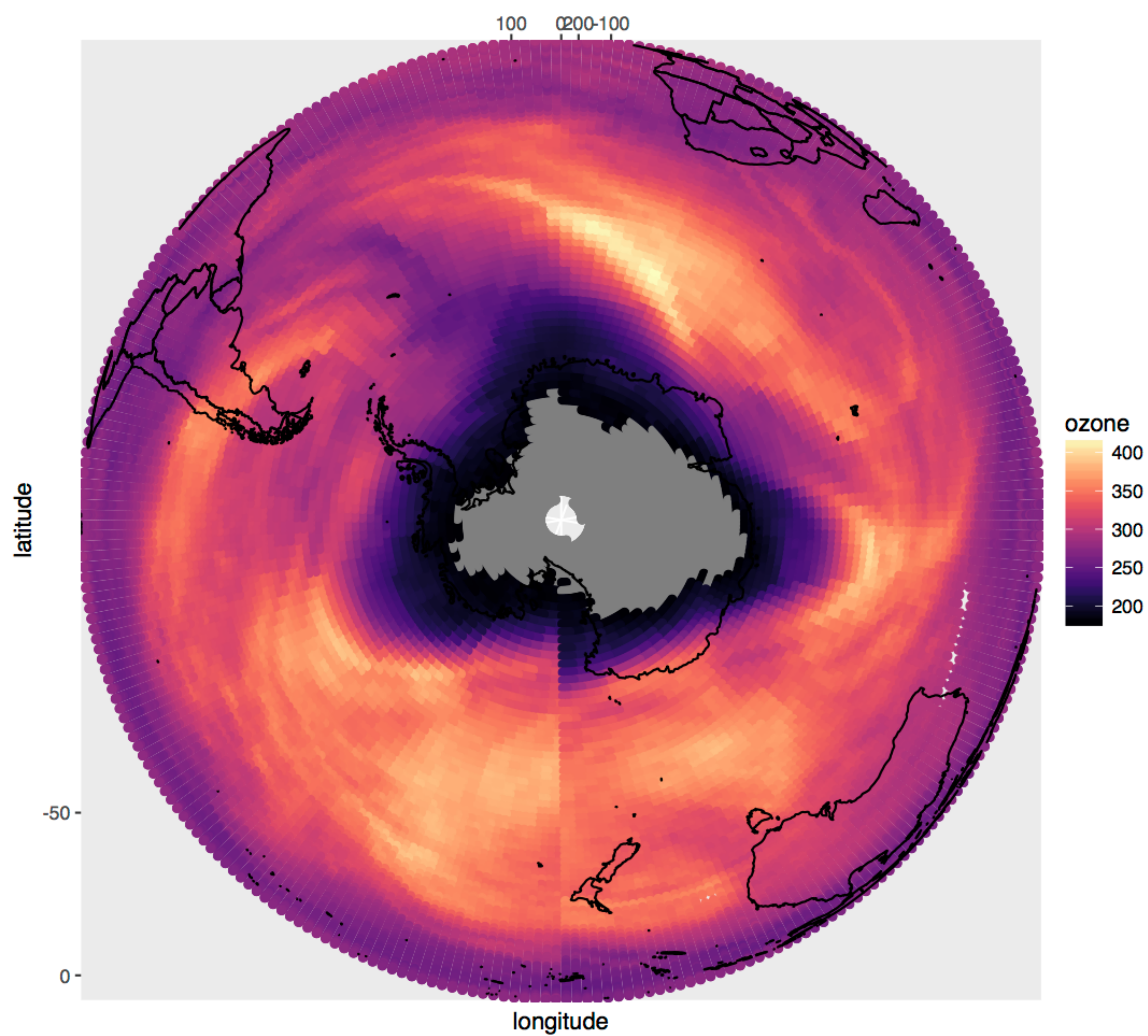
Writing

readr functions

function	reads
write_csv()	Comma separated values
write_excel_csv()	CSV intended for opening in Excel
write_delim()	General delimited files
write_file()	Single string, written as is
write_lines()	Vector of strings, one element per line
write_tsv()	Tab delimited values

write_csv()

Saves data set as a csv on your computer.

```
write_csv(nimbus, file = "nimbus2.csv")
```

Table to save

**file
path to save at**

Other types of data

package	accesses
haven	SPSS, Stata, and SAS files
readxl	excel files (.xls, .xlsx)
jsonlite	json
xml2	xml
httr	web API's
rvest	web pages (web scraping)
DBI	databases
sparklyr	data loaded into spark

readr

Import Data with