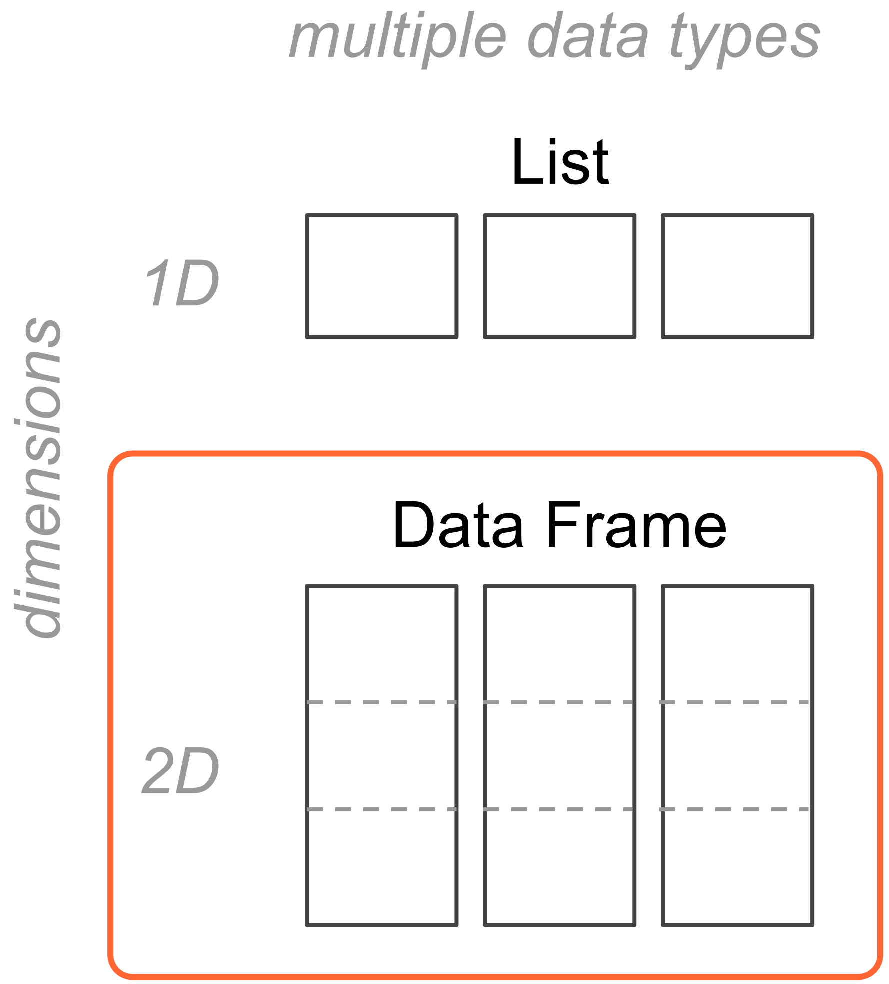
Importing Tables in R

Stat 133 by Gaston Sanchez

Importing tables in R

When reading a data table in R, it gets imported as a data frame


multiple data types
List
1D
dimensions
Data Frame
2D

R data frames

R data frames are special kinds of lists

Stored in R as a list of vectors (or factors)

Columns are typically atomic structures

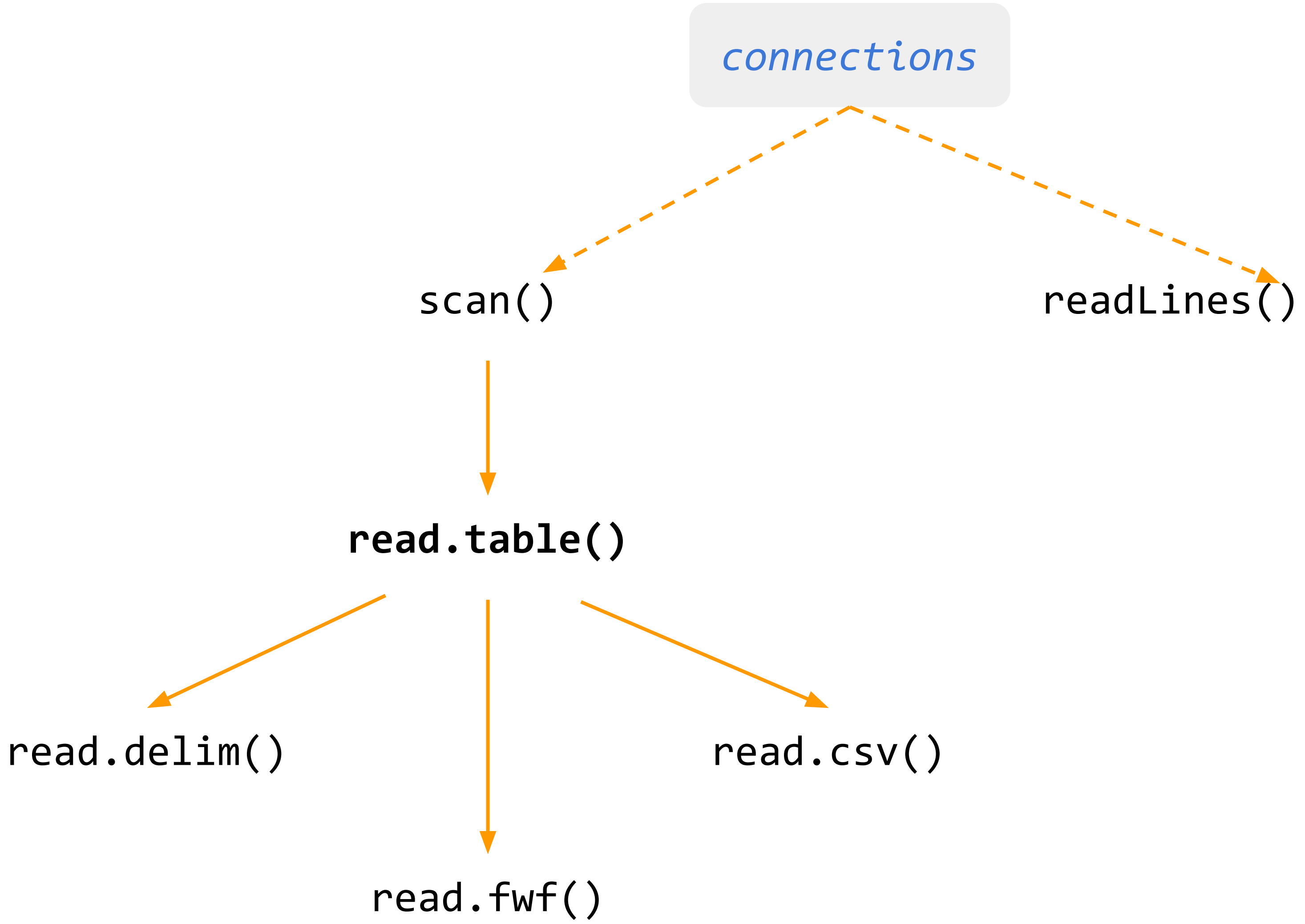
But since a data frame is a list, you can mix different types of columns

Functions to import tables

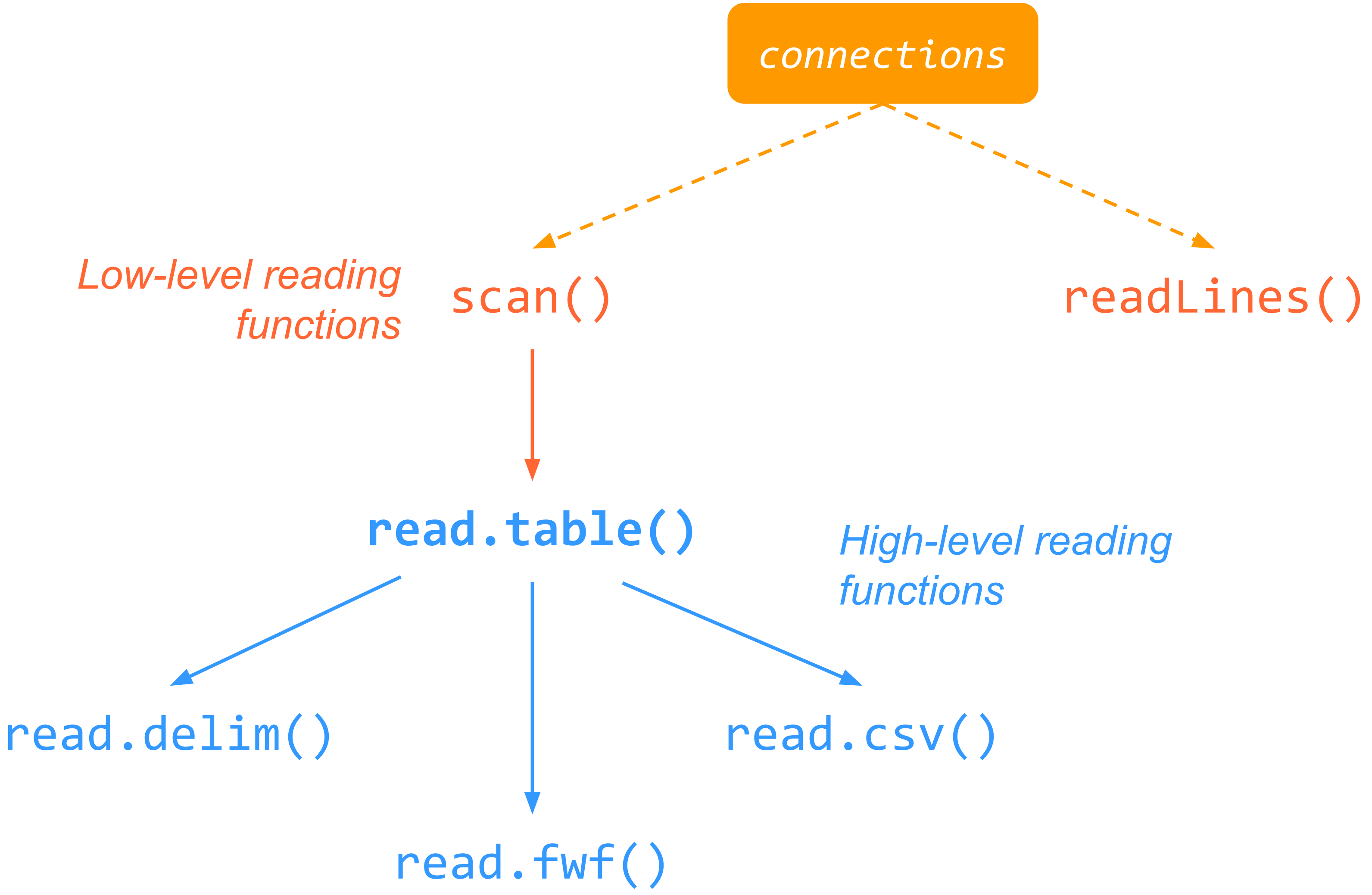
R comes with a family of functions that allows you to import most common data table formats

- `read.table()`
- `read.delim()`
- `read.csv()`, `read.csv2()`
- `read.fwf()`

Base R functions to read data



Base R functions to read data



Before importing a data table in R

What is the character(s) used as field delimiter?

Does the file contain names of columns?

Does the file contain a column for row names?

Are there any missing values?

How are missing values codified?

Before importing a data table in R

Do you need to convert delimiter characters? (e.g. from space to comma)

Can you determine the data-type of each column?

Are there any uninformative numbers?

Can you convert those uninformative numbers to informative labels?

Num	Name	Full	Gender	Height	Weight
1	Anakin	"Anakin Skywalker"	male	1.88	84
2	Padme	"Padme Amidala"	female	1.65	45
3	Luke	"Luke Skywalker"	male	1.72	77
4	Leia	"Leia Skywalker"	female	1.50	NA

row.names = 1

header = TRUE

Num	Name	Full	Gender	Height	Weight
1	Anakin	"Anakin Skywalker"	male	1.88	84
2	Padme	"Padme Amidala"	female	1.65	45
3	Luke	"Luke Skywalker"	male	1.72	77
4	Leia	"Leia Skywalker"	female	1.50	NA

dec = "."

quote = "\"'"

na.strings = "NA"

Strings and Factors

By default, strings are converted to factors when loading data frames.

This is the wrong default

Use **`stringsAsFactors = FALSE`**

You should always explicitly convert strings into factors later

R package “`readr`”

Data from google sheets

The package “readr” (by Wickham et al) is a relatively new package that makes it easy to read many types of tabular data

http://blog.rstudio.org/2015/04/09/readr-0-1-0

http://cran.r-project.org/web/packages/readr/vignettes/readr.html

Package “readr”

```
# remember to install it
install.packages("readr")

# load it
library(readr)
```

“readr” functions

Are around 10x faster than base functions

Are more consistent (better desgined)

Produce data frame that are easier to use

They have more flexible column specification

“readr” functions

Read delimited files with:

- `read_csv()`
- `read_csv2()`
- `read_delim()`
- `read_tsv()`

Read fixed width files with

- `read_table()`
- `read_fwf()`

Input arguments

`file_name`: (path) name of file

`col_names`: column names

`col_types`: data types of columns

`progress`: progress bar

Input arguments

`file` gives the file to read; a url or local path.

A local path can point to a a zipped, bzipped, xzipped, or gzipped file it'll be automatically uncompressed in memory before reading.

Input arguments

`col_names`: describes the column names (equivalent to header in base R). It has three possible values:

- □ `TRUE` will use the the first row of data as column names.
- □ `FALSE` will number the columns sequentially.
- □ A character vector to use as column names.

Input arguments

`col_types` (equivalent to `colClasses` automatically)

- □ `col_logical()`: contains only logical values
- □ `col_integer()`: integers
- □ `col_double()`: doubles (reals)
- □ `col_euro_double()` “Euro” doubles that use commas "," as decimal separator
- □ `col_date()`: Y-m-d dates
- □ `col_datetime()`: ISO8601 date times
- □ `col_character()`: everything else

Column types correspondence

Type	Abbreviation
`col_logical()`	`l`
`col_integer()`	`i`
`col_numeric()`	`n`
`col_double()`	`d`
`col_euro_double()`	`e`
`col_date()`	`D`
`col_datetime()`	`T`
`col_character()`	`c`
`col_skip()`	`_`

Column Types

Using a compact string: `"dc__d"`

Each letter corresponds to a column so this specification means: read first column as *double*, second as *character*, *skip* the next two, and read the last column as *double*.

Column Types

Another way to override the default choices of column types is by passing a list of col_… types

```
read_csv("iris.csv", col_types = list(
  Sepal.Length = col_double(),
  Sepal.Width = col_double(),
  Petal.Length = col_double(),
  Petal.Width = col_double(),
  Species = col_factor(c("setosa", "versicolor",
"virginica"))
))
```

Output

- Characters are never automatically converted to factors
- □Column names are left as is (i.e. there is no `check.names = TRUE`)
- □ Use backticks to refer to variables with unusual names: ``df$`Income ($000)` ``
- □Row names are never set
- □The output has class `c("tbl_df", "tbl", "data.frame")`

File: `starwarstoy.csv`

```
Name,Gender,Homeworld,Born,Jedi
Anakin,male,Tatooine,41.9BBY,yes
Amidala,female,Naboo,46BBY,no
Luke,male,Tatooine,19BBY,yes
Leia,female,Alderaan,19BBY,no
Obi-Wan,male,Stewjon,57BBY,yes
Han,male,Corellia,29BBY,no
Palpatine,male,Naboo,82BBY,no
R2-D2,unknown,Naboo,33BBY,no
```

String Columns as Factors

By default, functions in "readr" do not convert character strings into factors. But you can specify what columns to be imported as factors (you must specify the factor levels)

```r
sw1 <- read_csv(
  file = "starwarstoy.csv",
  col_types = list(
    gender = col_factor(c("male", "female"))
))
```

String Columns as Factors

"readr" allows you to import specific columns of a data set in an easier way than R base functions:

```
# import the first four columns
sw2 <- read_csv(
  file = "starwarstoy.csv",
  col_types = "ccnn___"
)
```

Other packages

Files from other programs

Type	Package	Function
Excel	gdata	`read.xls()`
Excel	xlsx	`read.xlsx()`
Excel	readxl	`read_excel()`
Excel	XLConnect	`readWorksheet()`
SPSS	foreign	`read.spss()`
SAS	foreign	`read.ssd()`
SAS	foreign	`read.xport()`
Matlab	R.matlab	`readMat()`
Stata	foreign	`read.dta()`
Octave	foreign	`read.octave()`
Minitab	foreign	`read.mtp()`
Systat	foreign	`read.systat()`

Data from google sheets?

Data from google sheets

R package “googlesheets” by Jennifer Bryan and Joanna Zhao

https://github.com/jennybc/googlesheets

After importing tables in R ....

There's a bunch of functions to inspect a data.frame object

Function	Description
`str()`	structure
`head()`	First rows
`tail()`	Last rows
`summary()`	Descriptive statistics
`dim()`	Dimensions (# rows, # columns)
`nrow()`	Number of rows
`ncol()`	Number of columns
`names()`	Column names
`colnames()`	Column names
`rownames()`	Row names
`dimnames()`	List with row and column names

Functions to inspect a data frame

```r
# display structure
str(airquality)

# display structure but showing
# few elements
str(airquality, vec.len = 1)
```

Functions to inspect a data frame

```
# first n rows
head(airquality, n = 5)

# last n rows
tail(airquality, n = 5)
```

Functions to inspect a data frame

```
# column summaries
summary(airquality)

# memory size
object.size(airquality)

# attributes
attributes(airquality)
```

Functions to inspect a data frame

```
# data frame dimensions
dim(airquality)

# number of rows
nrow(airquality)

# number of columns
ncol(airquality)
```

Functions to inspect a data frame

```
# row names
rownames(airquality)

# column names
colnames(airquality)

# column names
names(airquality)
```

Functions to inspect a data frame

```
# object class ('data.frame')
class(airquality)

# check if object is data.frame
is.data.frame(airquality)

# data.frame is also a list
is.list(airquality)
```