Session 2 assmt 3

1. How to Import SAS XPORT Files into R With The foreign package

From foreign library to load in SAS data in R. In such cases, start from a SAS Permanent Dataset or a SAS XPORT Format Library with the read.ssd() and read.xport() functions, respectively.

Usage

```
lookup.xport(file)
```

Arguments

file	character variable with the name of the file to read. The file must be in SAS XPORT format.

Value

A list with one component for each dataset in the XPORT format library.

`read.xport`

Examples

script.R

```
1  ## Not run: ## no XPORT file is installed.
2  # lookup.xport("test.xpt")
3  # ## End(Not run)
```

2. How To Import SAS Files into R With The haven Package?

Haven enables R to read and write various data formats used by other statistical packages by wrapping the fantastic ReadStat C library written by Evan Miller. Haven is part of the tidyverse. Currently it supports:

- SAS: `read_sas()` reads `.sas7bdat` + `.sas7bcat` files and `read_xpt()` reads SAS transport files (version 5 and version 8).

Usage

```
library(haven)

# SAS
read_sas("mtcars.sas7bdat")
write_sas(mtcars, "mtcars.sas7bdat")
```

Installation

```
# The easiest way to get haven is to install the whole tidyverse:
install.packages("tidyverse")

# Alternatively, install just haven:
install.packages("haven")

# Or the the development version from GitHub:
# install.packages("devtools")
devtools::install_github("tidyverse/haven")
```

3. How to read Weka Attribute-Relation File Format (ARFF) files in R?

Usage

read.arff(file)

Arguments

file

a character string with the name of the ARFF file to read from, or a connection which will be opened if necessary, and if so closed at the end of the function call.

Value

A data frame containing the data from the ARFF file

4. How to read a heavy csv/tsv file using readr package?

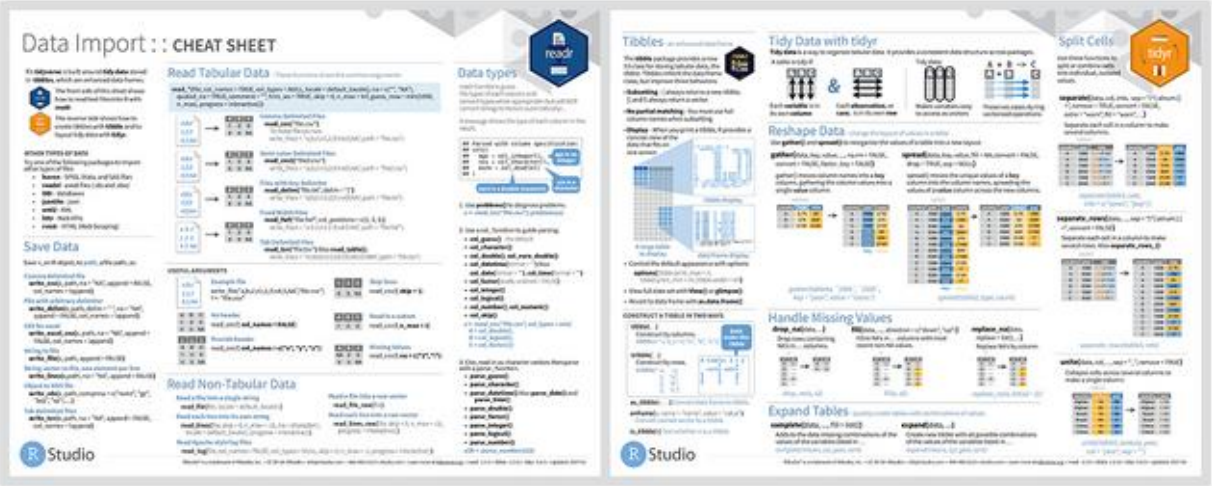
Installation

```
# The easiest way to get readr is to install the whole tidyverse:
install.packages("tidyverse")

# Alternatively, install just readr:
install.packages("readr")

# Or the the development version from GitHub:
# install.packages("devtools")
devtools::install_github("tidyverse/readr")
```

Cheatsheet



Usage

readr is part of the core tidyverse, so load it with:

```
library(tidyverse)
#> ── Attaching packages ─────────────────────────────────── tidyverse 1.2.1 ──
#> ✓ ggplot2 3.1.0     ✓ purrr   0.2.5
#> ✓ tibble  1.4.2     ✓ dplyr   0.7.8
#> ✓ tidyr   0.8.2     ✓ stringr 1.3.1
#> ✓ readr   1.3.1     ✓ forcats 0.3.0
#> ── Conflicts ────────────────────────────────────── tidyverse_conflicts() ──
#> ✖ dplyr::filter() masks stats::filter()
#> ✖ dplyr::lag()    masks stats::lag()
```

readr supports seven file formats with seven `read_` functions:

- `read_csv()` : comma separated (CSV) files
- `read_tsv()` : tab separated files
- `read_delim()` : general delimited files
- `read_fwf()` : fixed width files
- `read_table()` : tabular files where columns are separated by white-space.
- `read_log()` : web log files

In many cases, these functions will just work: you supply the path to a file and you get a tibble back. The following example loads a sample file bundled with readr:

```
mtcars <- read_csv(readr_example("mtcars.csv"))
#> Parsed with column specification:
#> cols(
#>   mpg = col_double(),
#>   cyl = col_double(),
#>   disp = col_double(),
#>   hp = col_double(),
#>   drat = col_double(),
#>   wt = col_double(),
#>   qsec = col_double(),
#>   vs = col_double(),
#>   am = col_double(),
#>   gear = col_double(),
#>   carb = col_double()
#> )
```