You'll be importing and cleaning **four** real datasets. Your **first** dataset describes online ticket sales for various events across the country.

1 *Ticket Sales Data*

Importing the data

```
# Import sales.csv: sales

sales <-read.csv("sales.csv",stringsAsFactors=FALSE)
```

Examining the data

```
# View dimensions of sales

dim(sales)


# Inspect first 6 rows of sales

head(sales)


# View column names of sales

names(sales)
```

Summarizing the data

```
# Look at structure of sales

str(sales)


# View a summary of sales

summary(sales)


# Load dplyr
```

```
library(dplyr)

# Get a glimpse of sales
glimpse(sales)
```

Removing redundant info

```
## sales is available in your workspace

# Remove the first column of sales: sales2
sales2 <- sales[,-1]
```

Information not worth keeping

```
## sales2 is available in your workspace

# Define a vector of column indices: keep
keep <- 5:(ncol(sales2) - 15)

# Subset sales2 using keep: sales3
sales3 <- sales2[,keep]
```

Separating columns

```
# Load tidyr
library(tidyr)

# Split event_date_time: sales4
sales4 <- separate(sales3, event_date_time,
```

```
        c("event_dt", "event_time"), sep = " ")

# Split sales_ord_create_dttm: sales5
sales5 <- separate(sales4, sales_ord_create_dttm,
        c("ord_create_dt", "ord_create_time"), sep = " ")
```

Dealing with warnings

```
# Define an issues vector
issues <- c(2516, 3863, 4082, 4183)

# Print values of sales_ord_create_dttm at these indices
sales3$sales_ord_create_dttm[issues]

# Print a well-behaved value of sales_ord_create_dttm
sales3$sales_ord_create_dttm[2517]
```

Identifying dates

```
## sales5 is pre-loaded

# Load stringr
library(stringr)

# Find columns of sales5 containing "dt": date_cols
date_cols <- str_detect(names(sales5), "dt")

# Load lubridate
library(lubridate)
```

```
# Coerce date columns into Date objects
sales5[, date_cols] <- lapply(sales5[, date_cols], ymd)
```

More warnings!

```
## stringr is loaded

# Find date columns (don't change)
date_cols <- str_detect(names(sales5), "dt")

# Create logical vectors indicating missing values (don't change)
missing <- lapply(sales5[, date_cols], is.na)

# Create a numerical vector that counts missing values: num_missing
num_missing <- sapply(missing, sum)

# Print num_missing
num_missing
```

Combining columns

```
## tidyr is loaded

# Combine the venue_city and venue_state columns
sales6 <- unite(sales5, venue_city_state,
                venue_city, venue_state, sep = ", ")

# View the head of sales6
```

```
head(sales6)
```

2 *MBTA Ridership Data*

Using readxl

```
# Load readxl
library(readxl)

# Import mbta.xlsx and skip first row: mbta
mbta <- read_excel("mbta.xlsx",skip=1)
```

Examining the data

```
## mbta is pre-loaded

# View the structure of mbta
str(mbta)

# View the first 6 rows of mbta
head(mbta)

# View a summary of mbta
summary(mbta)
```

Removing unnecessary rows and columns

```
# Remove rows 1, 7, and 11 of mbta: mbta2
rm <- c(1,7,11)
```

```
mbta2 <- mbta[-(rm),]

# Remove the first column of mbta2: mbta3
mbta3 <- mbta2[,-1]
```

Observations are stored in columns

```
## mbta3 is pre-loaded

# Load tidyr
library(tidyr)

# Gather columns of mbta3: mbta4
mbta4 <- gather(mbta3,month,thou_riders,-mode)

# View the head of mbta4
head(mbta4)
```

Type conversions

```
## mbta4 is pre-loaded

# Coerce thou_riders to numeric
mbta4$thou_riders <- as.numeric(mbta4$thou_riders)
class(mbta4$thou_riders)
```

Variables are stored in both rows and columns

```
## tidyr is pre-loaded
```

```
# Spread the contents of mbta4: mbta5
mbta5 <- spread(mbta4,mode,thou_riders)

# View the head of mbta5
head(mbta5)
```

Separating columns

```
## tidyr and mbta5 are pre-loaded

# View the head of mbta5
head(mbta5)

# Split month column into month and year: mbta6
mbta6 <- separate(mbta5,month,c("year","month"),sep="-")

# View the head of mbta6
head(mbta6)
```

Do your values seem reasonable?

```
## mbta6 is pre-loaded

# View a summary of mbta6
summary(mbta6)

# Generate a histogram of Boat ridership
hist(mbta6$Boat)
```

Dealing with entry error

```
# Find the row number of the incorrect value: i
i <- which(mbta6$Boat==40)

# Replace the incorrect value with 4
mbta6$Boat[i] <- 4

# Generate a histogram of Boat column
hist(mbta6$Boat)

# Look at Boat and Trackless Trolley ridership over time (don't change)
ggplot(mbta_boat, aes(x = month, y = thou_riders, col = mode)) +  geom_point() +
  scale_x_discrete(name = "Month", breaks = c(200701, 200801, 200901, 201001, 201101)) +
  scale_y_continuous(name = "Avg Weekday Ridership (thousands)")

# Look at all T ridership over time (don't change)
ggplot(mbta_all, aes(x = month, y = thou_riders, col = mode)) + geom_point() +
  scale_x_discrete(name = "Month", breaks = c(200701, 200801, 200901, 201001, 201101)) +
  scale_y_continuous(name = "Avg Weekday Ridership (thousands)")
```

3 *World Food Facts*

Importing the data

```
# Load data.table
library(data.table)
```

```
# Import food.csv: food
food <- fread("food.csv")

# Convert food to a data frame
data.frame("food")
```

Examining the data

```
## food is pre-loaded

# View summary of food
summary(food)

# View head of food
head(food)

# View structure of food
str(food)
```

Inspecting variables

```
# Load dplyr
library(dplyr)

# View a glimpse of food
glimpse(food)
```

```
# View column names of food

names(food)
```

Removing duplicate info

```
# Define vector of duplicate cols (don't change)

duplicates <- c(4, 6, 11, 13, 15, 17, 18, 20, 22,

                24, 25, 28, 32, 34, 36, 38, 40,

                44, 46, 48, 51, 54, 65, 158)


# Remove duplicates from food: food2

food2 <- food[,-(duplicates)]
```

Removing useless info

```
## food2 is pre-loaded


# Define useless vector (don't change)

useless <- c(1, 2, 3, 32:41)


# Remove useless columns from food2: food3

food3 <- food2[,-useless]
```

Finding columns

```
## stringr and food3 are pre-loaded


# Create vector of column indices: nutrition
```

```
nutrition <- str_detect(names(food3), "100g")

# View a summary of nutrition columns
summary(food3[, nutrition])
```

Replacing missing values

```
# Find indices of sugar NA values: missing
missing <- is.na(food3$sugars_100g)

# Replace NA values with 0
food3$sugars_100g[missing] <- 0

# Create first histogram
hist(food3$sugars_100g, breaks = 100)

# Create food4
food4 <- food3[food3$sugars_100g > 0, ]

# Create second histogram
hist(food4$sugars_100g, breaks = 100)
```

Dealing with messy data

```
## stringr is loaded

# Find entries containing "plasti": plastic
plastic <- str_detect(food3$packaging_tags,"plasti")
```

4 School Attendance Data

Importing the data

```
# Load the gdata package
library(gdata)

# Import the spreadsheet: att
att <- read.xls("attendance.xls")

# Print the sum of plastic
sum(plastic)
```

Examining the data

```
# Print the column names
names(att)

# Print the first 6 rows
head(att)

# Print the last 6 rows
tail(att)

# Print the structure
str(att)
```

Removing unnecessary rows

```
# Create remove
remove <- c(3, 56, 57, 58,59)

# Create att2
att2 <- att[-remove,]
```

Removing useless columns

```
# Create remove
remove <- c(3, 5, 7, 9, 11, 13, 15, 17)

# Create att3
att3 <- att2[,-remove]
```

Splitting the data

```
## att3 is pre-loaded

# Subset just elementary schools: att_elem
att_elem <- att3[,c(1,6,7)]

# Subset just secondary schools: att_sec
att_sec <- att3[,c(1,8,9)]

# Subset all schools: att4
att4 <- att3[,1:5]
```

Replacing the names

```
## att4 is pre-loaded
```

```
# Define cnames vector (don't change)
cnames <- c("state", "avg_attend_pct", "avg_hr_per_day",
        "avg_day_per_yr", "avg_hr_per_yr")

# Assign column names of att4
colnames(att4) <- cnames

# Remove first two rows of att4: att5
att5 <- att4[-c(1,2),]

# View the names of att5
names(att5)
```

Cleaning up extra characters

```
## stringr and att5 are pre-loaded

# Remove all periods in state column
att5$state <- str_replace_all(att5$state, "\\.", "")

# Remove white space around state names
att5$state <- str_trim(att5$state)

# View the head of att5
head(att5)
```

Some final type conversions

```
# Change columns to numeric using dplyr (don't change)
library(dplyr)
example <- mutate_each(att5, funs(as.numeric), -state)

# Define vector containing numerical columns: cols
cols <- c(2,3,4,5)

# Use sapply to coerce cols to numeric
att5[, cols] <- sapply(att5[,cols],as.numeric)
```