Here's what messy data look like

```
# View the first 6 rows of data
head(weather, n=6)


# View the last 6 rows of data
tail(weather , n=6)


# View a condensed summary of the data
str(weather)
```

Here's what clean data look like

```
# View the first 6 rows of data
head(weather_clean ,n=6)


# View the last 6 rows of data
tail(weather_clean , n=6)


# View a condensed summary of the data
str(weather_clean)
```

Getting a feel for your data

```
# Check the class of bmi
class(bmi)


# Check the dimensions of bmi
```

```
dim(bmi)

# View the column names of bmi
names(bmi)
```

Viewing the structure of your data

```
# Check the structure of bmi
str(bmi)

# Load dplyr
library(dplyr)

# Check the structure of bmi, the dplyr way
glimpse(bmi)

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

Looking at your data

```
# Print bmi to the console
print(bmi)

# View the first 6 rows
head(bmi)

# View the first 15 rows
head(bmi,n=15)
```

```
# View the last 6 rows
tail(bmi)

# View the last 10 rows
tail(bmi,n=10)
```

Visualizing your data

```
# Histogram of BMIs from 2008
hist(bmi$Y2008)

# Scatter plot comparing BMIs from 1980 to those from 2008
plot(bmi$Y1980,bmi$Y2008)
```

Gathering columns into key-value pairs

```
# Apply gather() to bmi and save the result as bmi_long
bmi_long <- gather(bmi, year, value, -Country)

# View the first 20 rows of the result
head(bmi_long,n=20)
```

Spreading key-value pairs into columns

```
# Apply spread() to bmi_long
bmi_wide <- spread(bmi_long, year, bmi_val)

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

Separating columns

```
# Apply separate() to bmi_cc
bmi_cc_clean <- separate(bmi_cc, col = Country_ISO, into = c("Country", "ISO"), sep = "/")

# Print the head of the result
head(bmi_cc_clean)
```

Uniting columns

```
# Apply unite() to bmi_cc_clean
bmi_cc <- unite(bmi_cc_clean, Country_ISO, Country, ISO, sep = "-")

# View the head of the result
head(bmi_cc)
```

Column headers are values, not variable names

```
## tidyr and dplyr are already loaded for you

# View the head of census
head(census)

# Gather the month columns
census2 <- gather(census,month,amount,-YEAR)

# Arrange rows by YEAR using dplyr's arrange
census2 <- arrange(census2, YEAR)
```

```
# View first 20 rows of census2
head(census2,n=20)
```

Variables are stored in both rows and columns

```
## tidyr is already loaded for you

# View first 50 rows of census_long
head(census_long,n=50)

# Spread the type column
census_long2 <- spread(census_long, type, amount)

# View first 20 rows of census_long2
head(census_long2,n=20)
```

Multiple values are stored in one column

```
## tidyr is already loaded for you

# View the head of census_long3
head(census_long3)

# Separate the yr_month column into two
census_long4 <- separate(census_long3,yr_month,c("year","month"))

# View the first 6 rows of the result
head(census_long4)
```

Types of variables in R

```
# Make this evaluate to character

class("true")

a <- as.character("true")
```

```
# Make this evaluate to numeric

class(8484.00)

b <-as.numeric(8484.00)
```

```
# Make this evaluate to integer

class(99L)

c <- as.integer(99)
```

```
# Make this evaluate to factor

class(factor("factor"))

d <- as.factor("factor")
```

```
# Make this evaluate to logical

class(FALSE)

e <- as.logical(FALSE)
```

Working with dates

```
# Preview students2 with str()

str(students2)
```

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

# Parse as date
dmy("17 Sep 2015")

# Parse as date and time (with no seconds!)
mdy_hm("July 15, 2012 12:56")

# Coerce dob to a date (with no time)
students2$dob <- ymd(students2$dob)

# Coerce nurse_visit to a date and time
students2$nurse_visit <- ymd_hms(students2$nurse_visit)

# Look at students2 once more with str()
str(students2)
```

Trimming and padding strings

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

# Trim all leading and trailing whitespace
str_trim(c("   Filip ", "Nick  ", " Jonathan"))

# Pad these strings with leading zeros
str_pad(c("23485W", "8823453Q", "994Z"),width = 9, side = "left", pad = "0")
```

Upper and lower case

```
# Print state abbreviations

print(states)


# Make states all uppercase and save result to states_upper

states_upper <- toupper(states)


# Make states_upper all lowercase again

tolower(states)
```

Finding and replacing strings

```
## stringr has been loaded for you


# Look at the head of students2

head(students2)


# Detect all dates of birth (dob) in 1997

str_detect(students2$dob, "1997")


# In the sex column, replace "F" with "Female"...

students2$sex <- str_replace(students2$sex,"F","Female")


# ...And "M" with "Male"

students2$sex <- str_replace(students2$sex,"M","Male")


# View the head of students2

head(students2)
```

Finding missing values

```
# Call is.na() on the full social_df to spot all NAs

is.na(social_df)


# Use the any() function to ask whether there are any NAs in the data

any(is.na(social_df))


# View a summary() of the dataset

summary(social_df)


# Call table() on the status column

table(social_df$status)
```

Dealing with missing values

```
## The stringr package is preloaded


# Replace all empty strings in status with NA

social_df$status[social_df$status == ""] <- NA


# Print social_df to the console

social_df


# Use complete.cases() to see which rows have no missing values

complete.cases(social_df)
```

```
# Use na.omit() to remove all rows with any missing values

na.omit(social_df)
```

Outliers and obvious errors
Dealing with outliers and obvious errors

```
# Look at a summary() of students3

summary(students3)


# View a histogram of the age variable

hist(students3$age)


# View a histogram of the absences variable

hist(students3$absences)


# View a histogram of absences, but force zeros to be bucketed to the right of zero

hist(students3$absences,right = FALSE)


# View a boxplot of age

boxplot(students3$age)


# View a boxplot of absences

boxplot(students3$absences)
```

Case Study: Get a feel for the data

```
# Verify that weather is a data.frame

class(weather)
```

```
# Check the dimensions
dim(weather)

# View the column names
names(weather)
```

Summarize the data

```
# View the structure of the data
str(weather)

# Load dplyr package
library(dplyr)

# Look at the structure using dplyr's glimpse()
glimpse(weather)

# View a summary of the data
summary(weather)
```

Take a closer look

Column names are values

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

# Gather the columns
weather2 <- gather(weather, day, value,X1:X31, -year, na.rm = TRUE)
```

```
# View the head

head(weather2)
```

Values are variable names

```
## The tidyr package is already loaded


# First remove column of row names

weather2 <- weather2[, -1]


# Spread the data

weather3 <- spread(weather2, measure, value)


# View the head

head(weather3)
```

Clean up dates

```
## tidyr and dplyr are already loaded


# Load the stringr and lubridate packages

library(stringr)

library(lubridate)



# Remove X's from day column

weather3$day <- str_replace(weather3$day,"X","")
```

```
# Unite the year, month, and day columns
weather4 <- unite(weather3, date, year, month, day, sep = "-")

# Convert date column to proper date format using lubridates's ymd()
weather4$date <- ymd(weather4$date)

# Rearrange columns using dplyr's select()
weather5 <- select(weather4, date, Events, CloudCover:WindDirDegrees)

# View the head of weather5
head(weather5)
```

A closer look at column types

```
# View the structure of weather5
str(weather5)

# Examine the first 20 rows of weather5. Are most of the characters numeric?
head(weather5,n=20)

# See what happens if we try to convert PrecipitationIn to numeric
as.numeric(weather5$PrecipitationIn)
```

Column type conversions

```
## The dplyr and stringr packages are already loaded

# Replace T with 0 (T = trace)
weather5$PrecipitationIn <- str_replace(weather5$PrecipitationIn,"T",0)
```

```
# Convert characters to numerics

weather6 <- mutate_each(weather5, funs(as.numeric), CloudCover:WindDirDegrees)


# Look at result

str(weather6)
```

Find missing values

```
# Count missing values

sum(is.na(weather6))


# Find missing values

summary(weather6)


# Find indices of NAs in Max.Gust.SpeedMPH

ind <- which(is.na(weather6$Max.Gust.SpeedMPH))


# Look at the full rows for records missing Max.Gust.SpeedMPH

weather6[ind, ]
```

An obvious error

```
# Review distributions for all variables

summary(weather6)


# Find row with Max.Humidity of 1000

ind <- which(weather6$Max.Humidity == 1000)
```

```
# Look at the data for that day
weather6[ind, ]

# Change 1000 to 100
weather6$Max.Humidity[ind] <- 100
```

Another obvious error

```
# Look at summary of Mean.VisibilityMiles
summary(weather6$Mean.VisibilityMiles)

# Get index of row with -1 value
ind <- which(weather6$Mean.VisibilityMiles == -1)

# Look at full row
weather6[ind, ]

# Set Mean.VisibilityMiles to the appropriate value
weather6$Mean.VisibilityMiles[ind] <- 10
```

Check other extreme values

```
# Review summary of full data once more
summary(weather6 )

# Look at histogram for MeanDew.PointF
hist(weather6$MeanDew.PointF)

# Look at histogram for Min.TemperatureF
```

```
hist(weather6$Min.TemperatureF)

# Compare to histogram for Mean.TemperatureF
hist(weather6$Mean.TemperatureF)
```

Finishing touches

```
# Clean up column names
names(weather6) <- new_colnames

# Replace empty cells in events column
weather6$events[weather6$events == ""] <- "None"

# Print the first 6 rows of weather6
head(weather6)
```