Understanding Tidyverse - Part2

Dr Ebin Deni Raj

05/02/2022

Contents

Note: the document is prepared for the first semester Mtech for Data Science.

Cleaning Data

We have seen the gapminder dataset extensively for the past few sessions. Fortunately the gapminder dataset did not have any issues such as missing values,data type issues etc.. The package dplyr provides easy tools for the most common data manipulation tasks. It is built to work directly with data frames, with many common tasks optimized by being written in a compiled language (C++). An additional feature is the ability to work directly with data stored in an external database. The benefits of doing this are that the data can be

managed natively in a relational database, queries can be conducted on that database, and only the results of the query are returned.

This addresses a common problem with R in that all operations are conducted in-memory and thus the amount of data you can work with is limited by available memory. The database connections essentially remove that limitation in that you can connect to a database of many hundreds of GB, conduct queries on it directly, and pull back into R only what you need for analysis.

But in real life most of the datasets need to be checked thoroughly for issues such as messy data.

Messy Data

As discussed in the last session, the real ground data, most of the time will be really messy. Analysis may not be easy in such a scenario.It's commonly said that data scientists spend 80% of their time cleaning and manipulating data and only 20% of their time actually analyzing it. For this reason, it is critical to become familiar with the data cleaning process and all of the tools available to you along the way.This document provides a very basic introduction to cleaning data in R using the tidyr, dplyr, and stringr packages. Load the weatherdata object(this was shared via email earlier)

When you start to do data analysis or modeling, the availability of clean data is of utmost importance. Hence you need to learn the different techniques to clean messy data.

With the advent of big data, it is critical to understand that data cleaning is an important part of any data science project. You can categorize any data science project into 4 simple steps.

- Acquiring or collecting the data.
- Data Cleaning.
- Modeling or Analyzing data.
- Reporting insights to the relevant audience. If we attempt to omit the data cleaning step, we will run into issues in any data science project, as raw data is tough to deal with, using traditional tools like Python or R.

Data cleaning is not only an essential component but also it is the one which takes most of the time in any data science project

You are given a messy, real-world dataset containing an entire year's worth of weather data from Boston, USA. Among other things, you'll be presented with variables that contain column names, column names that should be values, numbers coded as character strings, and values that are missing, extreme, and downright erroneous!

```
weather<-readRDS("datasets/weather.rds")
bmi<-read.csv("datasets/bmi.csv")
sales<- read.csv("datasets/sales.csv",stringsAsFactors=FALSE)
```

Take a close look at the weather dataset

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

```
##   X year month           measure X1 X2 X3 X4 X5 X6 X7 X8 X9 X10 X11 X12 X13 X14
## 1 1 2014    12  Max.TemperatureF 64 42 51 43 42 45 38 29 49  48  39  39  42  45
## 2 2 2014    12 Mean.TemperatureF 52 38 44 37 34 42 30 24 39  43  36  35  37  39
## 3 3 2014    12  Min.TemperatureF 39 33 37 30 26 38 21 18 29  38  32  31  32  33
## 4 4 2014    12    Max.Dew.PointF 46 40 49 24 37 45 36 28 49  45  37  28  28  29
## 5 5 2014    12    MeanDew.PointF 40 27 42 21 25 40 20 16 41  39  31  27  26  27
## 6 6 2014    12     Min.DewpointF 26 17 24 13 12 36 -3  3 28  37  27  25  24  25
##   X15 X16 X17 X18 X19 X20 X21 X22 X23 X24 X25 X26 X27 X28 X29 X30 X31
## 1  42  44  49  44  37  36  36  44  47  46  59  50  52  52  41  30  30
```

```
## 2  37  40  45  40  33  32  33  39  45  44  52  44  45  46  36  26  25
## 3  32  35  41  36  29  27  30  33  42  41  44  37  38  40  30  22  20
## 4  33  42  46  34  25  30  30  39  45  46  58  31  34  42  26  10   8
## 5  29  36  41  30  22  24  27  34  42  44  43  29  31  35  20   4   5
## 6  27  30  32  26  20  20  25  25  37  41  29  28  29  27  10  -6   1
```

```
# View the last 6 rows of data
tail(weather)
```

```
##       X year month            measure   X1   X2   X3   X4   X5   X6   X7   X8
## 281 281 2015    12 Mean.Wind.SpeedMPH    6 <NA> <NA> <NA> <NA> <NA> <NA> <NA>
## 282 282 2015    12  Max.Gust.SpeedMPH   17 <NA> <NA> <NA> <NA> <NA> <NA> <NA>
## 283 283 2015    12    PrecipitationIn 0.14 <NA> <NA> <NA> <NA> <NA> <NA> <NA>
## 284 284 2015    12         CloudCover    7 <NA> <NA> <NA> <NA> <NA> <NA> <NA>
## 285 285 2015    12             Events Rain <NA> <NA> <NA> <NA> <NA> <NA> <NA>
## 286 286 2015    12     WindDirDegrees  109 <NA> <NA> <NA> <NA> <NA> <NA> <NA>
##       X9  X10  X11  X12  X13  X14  X15  X16  X17  X18  X19  X20  X21  X22  X23
## 281 <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
## 282 <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
## 283 <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
## 284 <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
## 285 <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
## 286 <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
##      X24  X25  X26  X27  X28  X29  X30  X31
## 281 <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
## 282 <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
## 283 <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
## 284 <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
## 285 <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
## 286 <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
```

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

```
## 'data.frame':    286 obs. of  35 variables:
##  $ X      : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ year   : int  2014 2014 2014 2014 2014 2014 2014 2014 2014 2014 ...
##  $ month  : int  12 12 12 12 12 12 12 12 12 12 ...
##  $ measure: chr  "Max.TemperatureF" "Mean.TemperatureF" "Min.TemperatureF" "Max.Dew.PointF" ...
##  $ X1     : chr  "64" "52" "39" "46" ...
##  $ X2     : chr  "42" "38" "33" "40" ...
##  $ X3     : chr  "51" "44" "37" "49" ...
##  $ X4     : chr  "43" "37" "30" "24" ...
##  $ X5     : chr  "42" "34" "26" "37" ...
##  $ X6     : chr  "45" "42" "38" "45" ...
##  $ X7     : chr  "38" "30" "21" "36" ...
##  $ X8     : chr  "29" "24" "18" "28" ...
##  $ X9     : chr  "49" "39" "29" "49" ...
##  $ X10    : chr  "48" "43" "38" "45" ...
##  $ X11    : chr  "39" "36" "32" "37" ...
##  $ X12    : chr  "39" "35" "31" "28" ...
##  $ X13    : chr  "42" "37" "32" "28" ...
##  $ X14    : chr  "45" "39" "33" "29" ...
##  $ X15    : chr  "42" "37" "32" "33" ...
##  $ X16    : chr  "44" "40" "35" "42" ...
##  $ X17    : chr  "49" "45" "41" "46" ...
```

```
##  $ X18    : chr  "44" "40" "36" "34" ...
##  $ X19    : chr  "37" "33" "29" "25" ...
##  $ X20    : chr  "36" "32" "27" "30" ...
##  $ X21    : chr  "36" "33" "30" "30" ...
##  $ X22    : chr  "44" "39" "33" "39" ...
##  $ X23    : chr  "47" "45" "42" "45" ...
##  $ X24    : chr  "46" "44" "41" "46" ...
##  $ X25    : chr  "59" "52" "44" "58" ...
##  $ X26    : chr  "50" "44" "37" "31" ...
##  $ X27    : chr  "52" "45" "38" "34" ...
##  $ X28    : chr  "52" "46" "40" "42" ...
##  $ X29    : chr  "41" "36" "30" "26" ...
##  $ X30    : chr  "30" "26" "22" "10" ...
##  $ X31    : chr  "30" "25" "20" "8" ...
```

Even if you dont feel that the data is messy, its okay. By the end of this document you will be able to clearly understand what is meant by messy data and how to clean the messiness.We will come back to this dataset at the end.

Prerequisite for cleaning data

This section will introduce you to R packages such as stringr, lubridate etc.. *Basic Type conversions in R* Conversion from one type of data to another one will be frequently required when dealig with daatsets involving multiple data types. *Types of variables in R* As in other programming languages, R is capable of storing data in many different formats, most of which you've probably seen by now.

Loosely speaking, the class() function tells you what type of object you're working with. (There are subtle differences between the class, type, and mode of an object, but these distinctions are beyond the scope of this course.) Change the argument of each call to the class() function so it evaluates to the following (in order):

"character" "numeric" "integer" "factor" "logical"

```
# Make this evaluate to "character"
class("TRUE")
```

```
## [1] "character"
```

```
# Make this evaluate to "numeric"
class(8484.00)
```

```
## [1] "numeric"
```

```
# Make this evaluate to "integer"
class(99L)
```

```
## [1] "integer"
```

```
# Make this evaluate to "factor"
class(factor("factor"))
```

```
## [1] "factor"
```

```
# Make this evaluate to "logical"
class(FALSE)
```

```
## [1] "logical"
```

Type conversions in R

```
 as.character(2016)
```

```
## [1] "2016"
```

```
as.numeric(TRUE)
```

```
## [1] 1
```

```
 as.integer(99)
```

```
## [1] 99
```

```
 as.factor("something")
```

```
## [1] something
## Levels: something
```

```
  as.logical(0)
```

```
## [1] FALSE
```

Date conversions in R Date conversions is can be done using lubridate package Install: lubridate package in your R shell

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

```
## Warning in system("timedatectl", intern = TRUE): running command 'timedatectl'
## had status 1
```

```
##
## Attaching package: 'lubridate'
```

```
## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```
# Experiment with basic lubridate functions
ymd("2015-08-25")
```

```
## [1] "2015-08-25"
```

```
 ymd("2015 August 25")
```

```
## [1] "2015-08-25"
```

```
  mdy("August 25, 2015")
```

```
## [1] "2015-08-25"
```

```
  hms("13:33:09")
```

```
## [1] "13H 33M 9S"
```

```
   ymd_hms("2015/08/25 13.33.09")
```

```
## [1] "2015-08-25 13:33:09 UTC"
```

Dealing with string functions in R For this install stringr package.

```
install.packages("stringr")
```

This package is a set of helpful functions for working with strings.

```
library(stringr)
# Trim leading and trailing white space
str_trim(" this is a test ")
```

```
## [1] "this is a test"
```

```
# Pad string with zeros
 str_pad("24493", width = 7, side = "left", pad = "0")
```

```
## [1] "0024493"
```

```
# Create character vector of names
friends <- c("Ann", "Vinod", "Ashok")
# Search for string in vector
str_detect(friends, "Vinod")
```

```
## [1] FALSE  TRUE FALSE
```

```
# Replace string in vector
 str_replace(friends, "Ashok", "Thomas")
```

```
## [1] "Ann"    "Vinod"  "Thomas"
```

```
# Make all lowercase
 tolower("I AM TALKING LOUDLY!!")
```

```
## [1] "i am talking loudly!!"
```

```
 # Make all uppercase
toupper("I am whispering...")
```

```
## [1] "I AM WHISPERING..."
```

Missing values in a dataset

When you are dealing with real datasets, the probability of encountering missing values will be very high. This can happen at random but is a very dangerours threat in the analysis of data. These values can have great impact on the analysis/outcome of interest, if left unattended. In R missing values are usually denoted as NA, which means Not Available. There are other special values such as * Inf - “Infinite value” (indicative of outliers?) * NaN - “Not a number” (rethink a variable?)

How to find missing values?

```
# Create small dataset
df <- data.frame(A = c(1, NA, 8, NA),
 B = c(3, NA, 88, 23),
 C = c(2, 45, 3, 1))
# Check for NAs
 is.na(df)
```

```
##          A     B     C
## [1,] FALSE FALSE FALSE
## [2,]  TRUE  TRUE FALSE
## [3,] FALSE FALSE FALSE
## [4,]  TRUE FALSE FALSE
```

```
# Are there any NAs?
any(is.na(df))
```

```
## [1] TRUE
```

```
# Count number of NAs
 sum(is.na(df))
```

```
## [1] 3
```

```
 # Use summary() to find NAs
 summary(df)
```

```
##        A               B               C
##  Min.   :1.00   Min.   : 3.0   Min.   : 1.00
##  1st Qu.:2.75   1st Qu.:13.0   1st Qu.: 1.75
##  Median :4.50   Median :23.0   Median : 2.50
##  Mean   :4.50   Mean   :38.0   Mean   :12.75
##  3rd Qu.:6.25   3rd Qu.:55.5   3rd Qu.:13.50
##  Max.   :8.00   Max.   :88.0   Max.   :45.00
##  NA's   :2      NA's   :1
```

Dealing with missing values

```
# Find rows with no missing values
complete.cases(df)
```

```
## [1]  TRUE FALSE  TRUE FALSE
```

```
# Subset data, keeping only complete cases
 df[complete.cases(df), ]
```

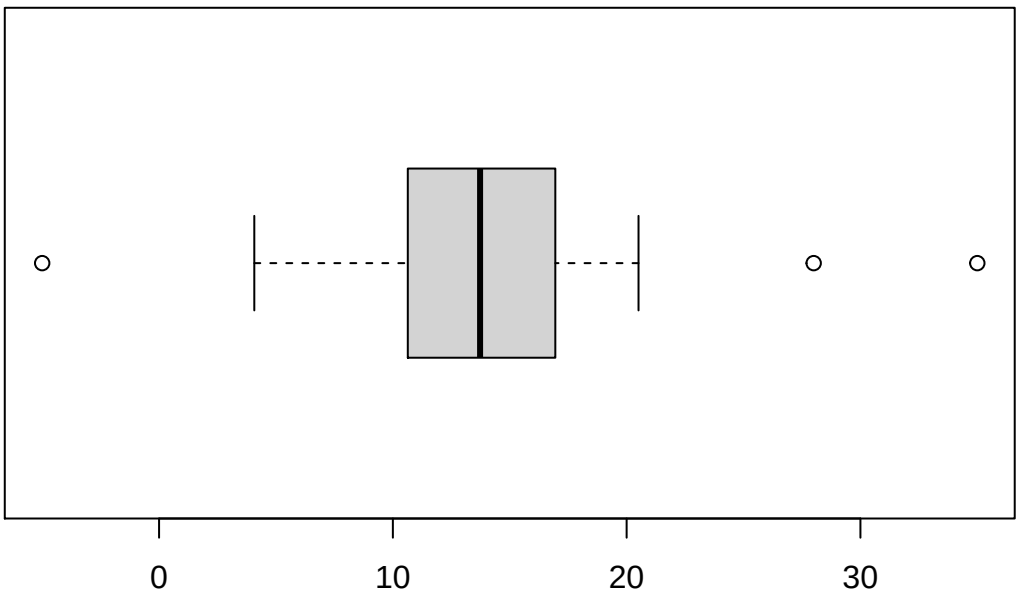
```
##   A  B C
## 1 1  3 2
## 3 8 88 3
```

```
# Another way to remove rows with NAs
 na.omit(df)
```

```
##   A  B C
## 1 1  3 2
## 3 8 88 3
```

Outliers vs Errors Outliers are those extreme values distant from other values. There are several causes for outliers in a datset. It may be discared or retained depending on cause * Obvious errors Values so extreme they can't be plausible (e.g. person aged 243).
Values that don't make sense (e.g. negative age) * Several causes Measurement error
Data entry error
Special code for missing data (e.g. -1 means missing) Should generally be removed or replaced

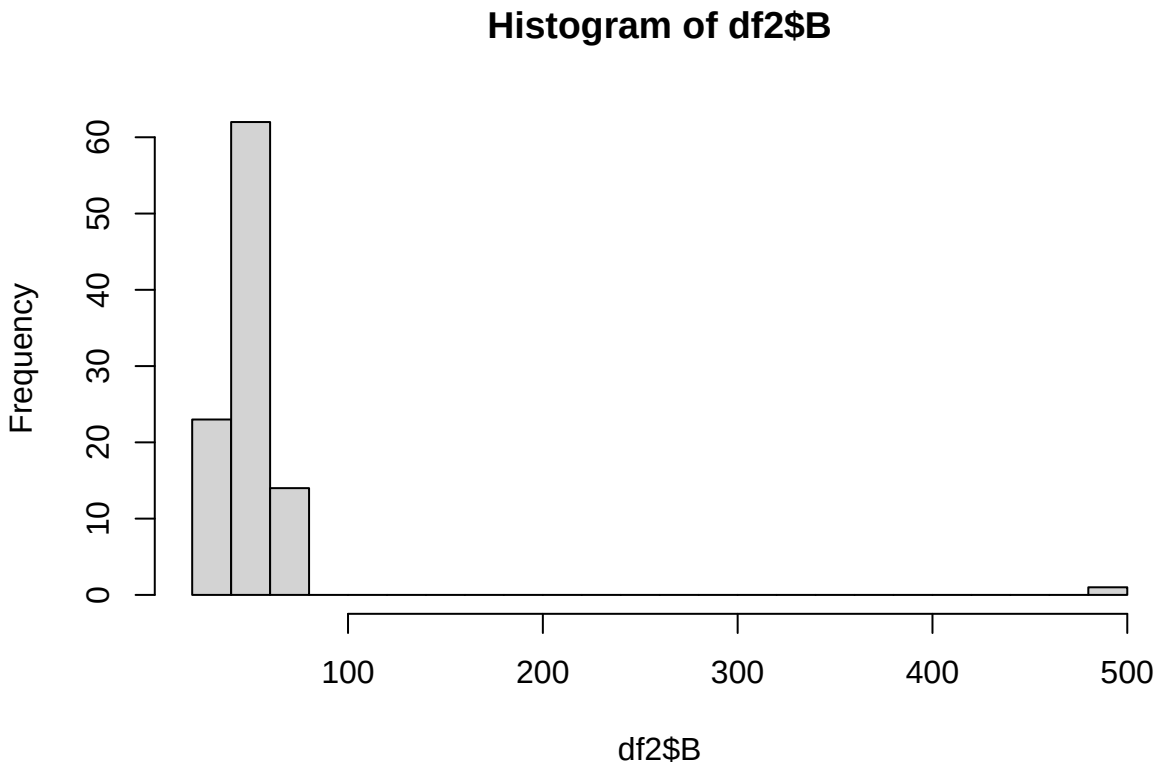
```
# Simulate some data
set.seed(10)
 x <- c(rnorm(30, mean = 15, sd = 5), -5, 28, 35)
# View a boxplot
boxplot(x, horizontal = TRUE)
```



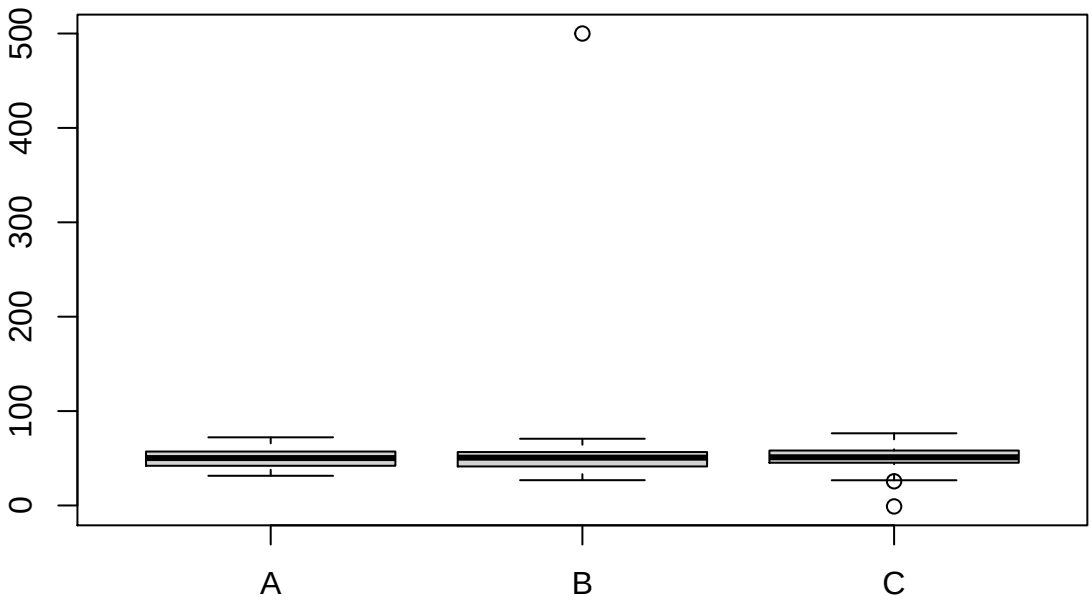

```
# Create another small dataset
df2 <- data.frame(A = rnorm(100, 50, 10),
 B = c(rnorm(99, 50, 10), 500),
 C = c(rnorm(99, 50, 10), -1))
# View a summary
 summary(df2)
```

```
##        A                B                C
##  Min.   :31.46   Min.   : 26.79   Min.   :-1.00
##  1st Qu.:42.21   1st Qu.: 41.35   1st Qu.:45.29
##  Median :50.20   Median : 50.67   Median :51.06
##  Mean   :49.70   Mean   : 53.62   Mean   :50.88
##  3rd Qu.:57.12   3rd Qu.: 56.57   3rd Qu.:58.13
##  Max.   :72.21   Max.   :500.00   Max.   :76.44
```

```
 # View a histogram
 hist(df2$B, breaks = 20)
```




```
# view a box plot
  boxplot(df2)
```



We will be larning more about dealing and managing Missing values in dataset at a later point in this course.

Principles of Tidy Data

- Observation as rows- Each observation has its own row
- Variables as columns- Each variable has its own column

- One type of observational unit per table, i.e Each type of observational unit forms a table
- Column headers are values, not variable names

Symptoms of Messy Data

There are two kinds of datasets 1) wide 2) Long The most important function in tidyr is *gather()*. It should be used when you have columns that are not variables and you want to collapse them into *key-value* pairs.

The easiest way to visualize the effect of *gather()* is that it makes wide datasets long.

gather() takes four principal arguments:

1. the data
2. the key column variable we wish to create from column names.
3. the values column variable we wish to create and fill with values associated with the key.
4. the names of the columns we use to fill the key variable (or to drop).

The opposite of *gather()* is *spread()*, which takes key-values pairs and spreads them across multiple columns. This is useful when values in a column should actually be column names (i.e. variables). It can also make data more compact and easier to read.

The easiest way to visualize the effect of *spread()* is that it makes long datasets wide. spread() takes three principal arguments:

1. the data
2. the key column variable whose values will become new column names.
3. the value column variable whose values will fill the new column variables.

running the following command will make long_df wide: You need to install tidyr package for this This can be demonstarted with a small example as shown below

```
library(tidyr)
col<-c("X","Y")
A<-c(1,4)
B<-c(2,5)
C<-c(3,6)
wide_df<-data.frame(col,A,B,C)
wide_df
```

```
##   col A B C
## 1   X 1 2 3
## 2   Y 4 5 6
```

```
#Gather columns into key-value pairs
gather(wide_df, my_key, my_val, -col)
```

```
##   col my_key my_val
## 1   X      A      1
## 2   Y      A      4
## 3   X      B      2
## 4   Y      B      5
## 5   X      C      3
## 6   Y      C      6
```

```
long_df<-gather(wide_df, my_key, my_val, -col)
long_df
```

```
##   col my_key my_val
## 1   X      A      1
## 2   Y      A      4
## 3   X      B      2
## 4   Y      B      5
## 5   X      C      3
## 6   Y      C      6
```

```
spread(long_df, my_key, my_val)
```

```
##   col A B C
## 1   X 1 2 3
## 2   Y 4 5 6
```

Applying gather and spread concepts to the BMI dataset 1. Apply the gather() function to bmi, saving the result to bmi_long. This will create two new columns:year, containing as values what are currently column headers bmi_val, the actual BMI values View the first 20 rows of bmi_long.

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

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

```
##                Country  year  bmi_val
## 1          Afghanistan Y1980 21.48678
## 2              Albania Y1980 25.22533
## 3              Algeria Y1980 22.25703
## 4              Andorra Y1980 25.66652
## 5               Angola Y1980 20.94876
## 6  Antigua and Barbuda Y1980 23.31424
## 7            Argentina Y1980 25.37913
## 8              Armenia Y1980 23.82469
## 9            Australia Y1980 24.92729
## 10             Austria Y1980 24.84097
## 11          Azerbaijan Y1980 24.49375
## 12             Bahamas Y1980 24.21064
## 13             Bahrain Y1980 23.97588
## 14          Bangladesh Y1980 20.51918
## 15            Barbados Y1980 24.36372
## 16             Belarus Y1980 24.90898
## 17             Belgium Y1980 25.09879
## 18              Belize Y1980 24.54345
## 19               Benin Y1980 20.80754
## 20             Bermuda Y1980 25.07881
```

2. Use *spread()* to reverse the operation that you performed in the last exercise with gather(). In other words, make bmi_long wide again, saving the result to bmi_wide.View the head of bmi_wide.

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

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

```
##               Country    Y1980    Y1981    Y1982    Y1983    Y1984    Y1985
## 1         Afghanistan 21.48678 21.46552 21.45145 21.43822 21.42734 21.41222
```

```
## 2             Albania 25.22533 25.23981 25.25636 25.27176 25.27901 25.28669
## 3             Algeria 22.25703 22.34745 22.43647 22.52105 22.60633 22.69501
## 4             Andorra 25.66652 25.70868 25.74681 25.78250 25.81874 25.85236
## 5              Angola 20.94876 20.94371 20.93754 20.93187 20.93569 20.94857
## 6 Antigua and Barbuda 23.31424 23.39054 23.45883 23.53735 23.63584 23.73109
##      Y1986    Y1987    Y1988    Y1989    Y1990    Y1991    Y1992    Y1993
## 1 21.40132 21.37679 21.34018 21.29845 21.24818 21.20269 21.14238 21.06376
## 2 25.29451 25.30217 25.30450 25.31944 25.32357 25.28452 25.23077 25.21192
## 3 22.76979 22.84096 22.90644 22.97931 23.04600 23.11333 23.18776 23.25764
## 4 25.89089 25.93414 25.98477 26.04450 26.10936 26.17912 26.24017 26.30356
## 5 20.96030 20.98025 21.01375 21.05269 21.09007 21.12136 21.14987 21.13938
## 6 23.83449 23.93649 24.05364 24.16347 24.26782 24.36568 24.45644 24.54096
##      Y1994    Y1995    Y1996    Y1997    Y1998    Y1999    Y2000    Y2001
## 1 20.97987 20.91132 20.85155 20.81307 20.78591 20.75469 20.69521 20.62643
## 2 25.22115 25.25874 25.31097 25.33988 25.39116 25.46555 25.55835 25.66701
## 3 23.32273 23.39526 23.46811 23.54160 23.61592 23.69486 23.77659 23.86256
## 4 26.36793 26.43569 26.50769 26.58255 26.66337 26.75078 26.83179 26.92373
## 5 21.14186 21.16022 21.19076 21.22621 21.27082 21.31954 21.37480 21.43664
## 6 24.60945 24.66461 24.72544 24.78714 24.84936 24.91721 24.99158 25.05857
##      Y2002    Y2003    Y2004    Y2005    Y2006    Y2007    Y2008
## 1 20.59848 20.58706 20.57759 20.58084 20.58749 20.60246 20.62058
## 2 25.77167 25.87274 25.98136 26.08939 26.20867 26.32753 26.44657
## 3 23.95294 24.05243 24.15957 24.27001 24.38270 24.48846 24.59620
## 4 27.02525 27.12481 27.23107 27.32827 27.43588 27.53363 27.63048
## 5 21.51765 21.59924 21.69218 21.80564 21.93881 22.08962 22.25083
## 6 25.13039 25.20713 25.29898 25.39965 25.51382 25.64247 25.76602
```

Working with ticket sales data

```
# View dimensions of sales
dim(sales)
```

```
## [1] 5000   46
```

```
# Inspect first 6 rows of sales
head(sales)
```

```
##   X             event_id       primary_act_id     secondary_act_id
## 1 1 abcaf1adb99a935fc661 43f0436b905bfa7c2eec b85143bf51323b72e53c
## 2 2 6c56d7f08c95f2aa453c 1a3e9aecd0617706a794 f53529c5679ea6ca5a48
## 3 3 c7ab4524a121f9d687d2 4b677c3f5bec71eec8d1 b85143bf51323b72e53c
## 4 4 394cb493f893be9b9ed1 b1ccea01ad6ef8522796 b85143bf51323b72e53c
## 5 5 55b5f67e618557929f48 91c03a34b562436efa3c b85143bf51323b72e53c
## 6 6 4f10fd8b9f550352bd56 ac4b847b3fde66f2117e 63814f3d63317f1b56c4
##    purch_party_lkup_id
## 1 7dfa56dd7d5956b17587
## 2 4f9e6fc637eaf7b736c2
## 3 6c2545703bd527a7144d
## 4 527d6b1eaffc69ddd882
## 5 8bd62c394a35213bdf52
## 6 3b3a628f83135acd0676
##                                                  event_name
## 1 Xfinity Center Mansfield Premier Parking: Florida Georgia Line
## 2                  Gorge Camping - dave matthews band - sept 3-7
```

```
## 3                    Dodge Theatre Adams Street Parking - benise
## 4   Gexa Energy Pavilion Vip Parking : kid rock with sheryl crow
## 5                                 Premier Parking - motley crue
## 6                                     Fast Lane Access: Journey
##                            primary_act_name secondary_act_name major_cat_name
## 1 XFINITY Center Mansfield Premier Parking               NULL           MISC
## 2                            Gorge Camping Dave Matthews Band           MISC
## 3                            Parking Event               NULL           MISC
## 4         Gexa Energy Pavilion VIP Parking               NULL           MISC
## 5 White River Amphitheatre Premier Parking               NULL           MISC
## 6                          Fast Lane Access            Journey           MISC
##           minor_cat_name la_event_type_cat
## 1                PARKING           PARKING
## 2                CAMPING           INVALID
## 3                PARKING           PARKING
## 4                PARKING           PARKING
## 5                PARKING           PARKING
## 6 SPECIAL ENTRY (UPSELL)            UPSELL
##                                                event_disp_name
## 1 Xfinity Center Mansfield Premier Parking: Florida Georgia Line
## 2                  Gorge Camping - dave matthews band - sept 3-7
## 3                    Dodge Theatre Adams Street Parking - benise
## 4   Gexa Energy Pavilion Vip Parking : kid rock with sheryl crow
## 5                                 Premier Parking - motley crue
## 6                                     Fast Lane Access: Journey
##
## 1    THIS TICKET IS VALID        FOR PARKING ONLY         GOOD THIS DAY ONLY       PREMIER PARKING PA
## 2                                                                %OVERNIGHT C A M P I N G%* * * * * *
## 3                              ADAMS STREET GARAGE%PARKING FOR 4/21/06 ONLY%DODGE THEATRE PARKING PA
## 4    THIS TICKET IS VALID        FOR PARKING ONLY      GOOD FOR THIS DATE ONLY       VIP PARKING PASS
## 5                             THIS TICKET IS VALID%FOR PARKING ONLY%GOOD THIS DATE ONLY%PREMIER PARK
## 6         FAST LANE                   JOURNEY               FAST LANE EVENT         THIS IS NOT A TIC
##   tickets_purchased_qty trans_face_val_amt delivery_type_cd  event_date_time
## 1                     1                 45          eTicket 12-09-2015 23:30
## 2                     1                 75       TicketFast 05-09-2009 01:00
## 3                     1                  5       TicketFast 22-04-2006 01:30
## 4                     1                 20             Mail 03-09-2011 00:00
## 5                     1                 20             Mail 31-07-2005 01:00
## 6                     2                 10       TicketFast 22-07-2012 02:00
##     event_dt presale_dt  onsale_dt sales_ord_create_dttm sales_ord_tran_dt
## 1 12-09-2015       NULL 15-05-2015      11-09-2015 18:17        11-09-2015
## 2 04-09-2009       NULL 13-03-2009      06-07-2009 00:00        05-07-2009
## 3 21-04-2006       NULL 25-02-2006      05-04-2006 00:00        05-04-2006
## 4 02-09-2011       NULL 22-04-2011      01-07-2011 17:38        01-07-2011
## 5 30-07-2005 02-03-2005 04-03-2005      18-06-2005 00:00        18-06-2005
## 6 21-07-2012       NULL 11-04-2012      21-07-2012 17:20        21-07-2012
##     print_dt timezn_nm     venue_city   venue_state venue_postal_cd_sgmt_1
## 1 12-09-2015       EST      MANSFIELD MASSACHUSETTS                  2048
## 2 01-09-2009       PST         QUINCY    WASHINGTON                 98848
## 3 05-04-2006       MST        PHOENIX       ARIZONA                 85003
## 4 06-07-2011       CST         DALLAS         TEXAS                 75210
## 5 28-06-2005       PST         AUBURN    WASHINGTON                 98092
## 6 21-07-2012       PST SAN BERNARDINO    CALIFORNIA                 92407
##             sales_platform_cd print_flg la_valid_tkt_event_flg  fin_mkt_nm
```

```
## 1 www.concerts.livenation.com        T                    N       Boston
## 2                         NULL       T                    N      Seattle
## 3                         NULL       T                    N      Arizona
## 4                         NULL       T                    N       Dallas
## 5                         NULL       T                    N      Seattle
## 6          www.livenation.com        T                    N  Los Angeles
##   web_session_cookie_val gndr_cd age_yr income_amt edu_val edu_1st_indv_val
## 1   7dfa56dd7d5956b17587    <NA>   <NA>       <NA>    <NA>             <NA>
## 2   4f9e6fc637eaf7b736c2    <NA>   <NA>       <NA>    <NA>             <NA>
## 3   6c2545703bd527a7144d    <NA>   <NA>       <NA>    <NA>             <NA>
## 4   527d6b1eaffc69ddd882    <NA>   <NA>       <NA>    <NA>             <NA>
## 5   8bd62c394a35213bdf52    <NA>   <NA>       <NA>    <NA>             <NA>
## 6   3b3a628f83135acd0676    <NA>   <NA>       <NA>    <NA>             <NA>
##   edu_2nd_indv_val adults_in_hh_num married_ind child_present_ind
## 1             <NA>             <NA>        <NA>              <NA>
## 2             <NA>             <NA>        <NA>              <NA>
## 3             <NA>             <NA>        <NA>              <NA>
## 4             <NA>             <NA>        <NA>              <NA>
## 5             <NA>             <NA>        <NA>              <NA>
## 6             <NA>             <NA>        <NA>              <NA>
##   home_owner_ind occpn_val occpn_1st_val occpn_2nd_val dist_to_ven
## 1           <NA>      <NA>          <NA>          <NA>          NA
## 2           <NA>      <NA>          <NA>          <NA>          59
## 3           <NA>      <NA>          <NA>          <NA>          NA
## 4           <NA>      <NA>          <NA>          <NA>          NA
## 5           <NA>      <NA>          <NA>          <NA>          NA
## 6           <NA>      <NA>          <NA>          <NA>          NA
```

```
# View column names of sales
names(sales)
```

```
##  [1] "X"                     "event_id"              "primary_act_id"
##  [4] "secondary_act_id"      "purch_party_lkup_id"   "event_name"
##  [7] "primary_act_name"      "secondary_act_name"    "major_cat_name"
## [10] "minor_cat_name"        "la_event_type_cat"     "event_disp_name"
## [13] "ticket_text"           "tickets_purchased_qty" "trans_face_val_amt"
## [16] "delivery_type_cd"      "event_date_time"       "event_dt"
## [19] "presale_dt"            "onsale_dt"             "sales_ord_create_dttm"
## [22] "sales_ord_tran_dt"     "print_dt"              "timezn_nm"
## [25] "venue_city"            "venue_state"           "venue_postal_cd_sgmt_1"
## [28] "sales_platform_cd"     "print_flg"             "la_valid_tkt_event_flg"
## [31] "fin_mkt_nm"            "web_session_cookie_val" "gndr_cd"
## [34] "age_yr"                "income_amt"            "edu_val"
## [37] "edu_1st_indv_val"      "edu_2nd_indv_val"      "adults_in_hh_num"
## [40] "married_ind"           "child_present_ind"     "home_owner_ind"
## [43] "occpn_val"             "occpn_1st_val"         "occpn_2nd_val"
## [46] "dist_to_ven"
```

Summarizing the data

Luckily, the rows and columns appear to be arranged in a meaningful way: each row represents an observation and each column a variable, or piece of information about that observation.

In R, there are a great many tools at your disposal to help get a feel for your data. Besides the three you used in the previous exercise, the functions str() and summary() can be very helpful.

The dplyr package, introduced in the last session, offers the glimpse() function, which can also be used for this purpose 3. Look at the structure of sales. View a summary of your data. Load the dplyr package using library(). Use glimpse() to look at your data.

```
# Look at structure of sales
str(sales)
```

```
## 'data.frame':    5000 obs. of  46 variables:
##  $ X                     : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ event_id              : chr  "abcaf1adb99a935fc661" "6c56d7f08c95f2aa453c" "c7ab4524a121f9d687d2"
##  $ primary_act_id        : chr  "43f0436b905bfa7c2eec" "1a3e9aecd0617706a794" "4b677c3f5bec71eec8d1"
##  $ secondary_act_id      : chr  "b85143bf51323b72e53c" "f53529c5679ea6ca5a48" "b85143bf51323b72e53c"
##  $ purch_party_lkup_id   : chr  "7dfa56dd7d5956b17587" "4f9e6fc637eaf7b736c2" "6c2545703bd527a7144d"
##  $ event_name            : chr  "Xfinity Center Mansfield Premier Parking: Florida Georgia Line" "Go
##  $ primary_act_name      : chr  "XFINITY Center Mansfield Premier Parking" "Gorge Camping" "Parking
##  $ secondary_act_name    : chr  "NULL" "Dave Matthews Band" "NULL" "NULL" ...
##  $ major_cat_name        : chr  "MISC" "MISC" "MISC" "MISC" ...
##  $ minor_cat_name        : chr  "PARKING" "CAMPING" "PARKING" "PARKING" ...
##  $ la_event_type_cat     : chr  "PARKING" "INVALID" "PARKING" "PARKING" ...
##  $ event_disp_name       : chr  "Xfinity Center Mansfield Premier Parking: Florida Georgia Line" "Go
##  $ ticket_text           : chr  "   THIS TICKET IS VALID        FOR PARKING ONLY         GOOD THIS DA
##  $ tickets_purchased_qty : int  1 1 1 1 1 2 1 1 1 1 ...
##  $ trans_face_val_amt    : num  45 75 5 20 20 10 30 28 20 25 ...
##  $ delivery_type_cd      : chr  "eTicket" "TicketFast" "TicketFast" "Mail" ...
##  $ event_date_time       : chr  "12-09-2015 23:30" "05-09-2009 01:00" "22-04-2006 01:30" "03-09-2011
##  $ event_dt              : chr  "12-09-2015" "04-09-2009" "21-04-2006" "02-09-2011" ...
##  $ presale_dt            : chr  "NULL" "NULL" "NULL" "NULL" ...
##  $ onsale_dt             : chr  "15-05-2015" "13-03-2009" "25-02-2006" "22-04-2011" ...
##  $ sales_ord_create_dttm : chr  "11-09-2015 18:17" "06-07-2009 00:00" "05-04-2006 00:00" "01-07-2011
##  $ sales_ord_tran_dt     : chr  "11-09-2015" "05-07-2009" "05-04-2006" "01-07-2011" ...
##  $ print_dt              : chr  "12-09-2015" "01-09-2009" "05-04-2006" "06-07-2011" ...
##  $ timezn_nm             : chr  "EST" "PST" "MST" "CST" ...
##  $ venue_city            : chr  "MANSFIELD" "QUINCY" "PHOENIX" "DALLAS" ...
##  $ venue_state           : chr  "MASSACHUSETTS" "WASHINGTON" "ARIZONA" "TEXAS" ...
##  $ venue_postal_cd_sgmt_1: chr  "2048" "98848" "85003" "75210" ...
##  $ sales_platform_cd     : chr  "www.concerts.livenation.com" "NULL" "NULL" "NULL" ...
##  $ print_flg             : chr  "T " "T " "T " "T " ...
##  $ la_valid_tkt_event_flg: chr  "N " "N " "N " "N " ...
##  $ fin_mkt_nm            : chr  "Boston" "Seattle" "Arizona" "Dallas" ...
##  $ web_session_cookie_val: chr  "7dfa56dd7d5956b17587" "4f9e6fc637eaf7b736c2" "6c2545703bd527a7144d"
##  $ gndr_cd               : chr  NA NA NA NA ...
##  $ age_yr                : chr  NA NA NA NA ...
##  $ income_amt            : chr  NA NA NA NA ...
##  $ edu_val               : chr  NA NA NA NA ...
##  $ edu_1st_indv_val      : chr  NA NA NA NA ...
##  $ edu_2nd_indv_val      : chr  NA NA NA NA ...
##  $ adults_in_hh_num      : chr  NA NA NA NA ...
##  $ married_ind           : chr  NA NA NA NA ...
##  $ child_present_ind     : chr  NA NA NA NA ...
##  $ home_owner_ind        : chr  NA NA NA NA ...
##  $ occpn_val             : chr  NA NA NA NA ...
##  $ occpn_1st_val         : chr  NA NA NA NA ...
##  $ occpn_2nd_val         : chr  NA NA NA NA ...
##  $ dist_to_ven           : int  NA 59 NA NA NA NA NA NA NA NA ...
```

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

```
##        X          event_id         primary_act_id     secondary_act_id
##  Min.   :   1   Length:5000        Length:5000        Length:5000
##  1st Qu.:1251   Class :character   Class :character   Class :character
##  Median :2500   Mode  :character   Mode  :character   Mode  :character
##  Mean   :2500
##  3rd Qu.:3750
##  Max.   :5000
##
##  purch_party_lkup_id  event_name        primary_act_name   secondary_act_name
##  Length:5000         Length:5000        Length:5000        Length:5000
##  Class :character    Class :character   Class :character   Class :character
##  Mode  :character    Mode  :character   Mode  :character   Mode  :character
##
##
##
##
##  major_cat_name     minor_cat_name     la_event_type_cat  event_disp_name
##  Length:5000        Length:5000        Length:5000        Length:5000
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
##
##
##
##  ticket_text        tickets_purchased_qty trans_face_val_amt delivery_type_cd
##  Length:5000        Min.   :1.000         Min.   :   1.00    Length:5000
##  Class :character   1st Qu.:1.000         1st Qu.:  20.00    Class :character
##  Mode  :character   Median :1.000         Median :  30.00    Mode  :character
##                     Mean   :1.639         Mean   :  77.08
##                     3rd Qu.:2.000         3rd Qu.:  85.00
##                     Max.   :8.000         Max.   :1520.88
##
##  event_date_time      event_dt          presale_dt         onsale_dt
##  Length:5000        Length:5000        Length:5000        Length:5000
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
##
##
##
##  sales_ord_create_dttm sales_ord_tran_dt    print_dt          timezn_nm
##  Length:5000           Length:5000        Length:5000        Length:5000
##  Class :character      Class :character   Class :character   Class :character
##  Mode  :character      Mode  :character   Mode  :character   Mode  :character
##
##
##
##
##   venue_city        venue_state        venue_postal_cd_sgmt_1
##  Length:5000        Length:5000        Length:5000
##  Class :character   Class :character   Class :character
```

```
##  Mode  :character   Mode  :character   Mode  :character
##
##
##
##
##  sales_platform_cd   print_flg         la_valid_tkt_event_flg
##  Length:5000        Length:5000        Length:5000
##  Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character
##
##
##
##
##   fin_mkt_nm        web_session_cookie_val   gndr_cd
##  Length:5000        Length:5000            Length:5000
##  Class :character   Class :character       Class :character
##  Mode  :character   Mode  :character       Mode  :character
##
##
##
##
##     age_yr           income_amt          edu_val          edu_1st_indv_val
##  Length:5000        Length:5000        Length:5000        Length:5000
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
##
##
##
##  edu_2nd_indv_val   adults_in_hh_num   married_ind        child_present_ind
##  Length:5000        Length:5000        Length:5000        Length:5000
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
##
##
##
##  home_owner_ind      occpn_val         occpn_1st_val      occpn_2nd_val
##  Length:5000        Length:5000        Length:5000        Length:5000
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
##
##
##
##   dist_to_ven
##  Min.   :   0.0
##  1st Qu.:  12.0
##  Median :  26.0
##  Mean   : 158.2
##  3rd Qu.:  77.5
##  Max.   :2548.0
##  NA's   :4677
```

```r
# Load dplyr
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

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

```
## Rows: 5,000
## Columns: 46
## $ X                       <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, ~
## $ event_id                <chr> "abcaf1adb99a935fc661", "6c56d7f08c95f2aa453c",~
## $ primary_act_id          <chr> "43f0436b905bfa7c2eec", "1a3e9aecd0617706a794",~
## $ secondary_act_id        <chr> "b85143bf51323b72e53c", "f53529c5679ea6ca5a48",~
## $ purch_party_lkup_id     <chr> "7dfa56dd7d5956b17587", "4f9e6fc637eaf7b736c2",~
## $ event_name              <chr> "Xfinity Center Mansfield Premier Parking: Flor~
## $ primary_act_name        <chr> "XFINITY Center Mansfield Premier Parking", "Go~
## $ secondary_act_name      <chr> "NULL", "Dave Matthews Band", "NULL", "NULL", "~
## $ major_cat_name          <chr> "MISC", "MISC", "MISC", "MISC", "MISC", "MISC",~
## $ minor_cat_name          <chr> "PARKING", "CAMPING", "PARKING", "PARKING", "PA~
## $ la_event_type_cat       <chr> "PARKING", "INVALID", "PARKING", "PARKING", "PA~
## $ event_disp_name         <chr> "Xfinity Center Mansfield Premier Parking: Flor~
## $ ticket_text             <chr> "   THIS TICKET IS VALID        FOR PARKING ONL~
## $ tickets_purchased_qty   <int> 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 2, 4, 1, 1, 1,~
## $ trans_face_val_amt      <dbl> 45, 75, 5, 20, 20, 10, 30, 28, 20, 25, 20, 90, ~
## $ delivery_type_cd        <chr> "eTicket", "TicketFast", "TicketFast", "Mail", ~
## $ event_date_time         <chr> "12-09-2015 23:30", "05-09-2009 01:00", "22-04-~
## $ event_dt                <chr> "12-09-2015", "04-09-2009", "21-04-2006", "02-0~
## $ presale_dt              <chr> "NULL", "NULL", "NULL", "NULL", "02-03-2005", "~
## $ onsale_dt               <chr> "15-05-2015", "13-03-2009", "25-02-2006", "22-0~
## $ sales_ord_create_dttm   <chr> "11-09-2015 18:17", "06-07-2009 00:00", "05-04-~
## $ sales_ord_tran_dt       <chr> "11-09-2015", "05-07-2009", "05-04-2006", "01-0~
## $ print_dt                <chr> "12-09-2015", "01-09-2009", "05-04-2006", "06-0~
## $ timezn_nm               <chr> "EST", "PST", "MST", "CST", "PST", "PST", "EST"~
## $ venue_city              <chr> "MANSFIELD", "QUINCY", "PHOENIX", "DALLAS", "AU~
## $ venue_state             <chr> "MASSACHUSETTS", "WASHINGTON", "ARIZONA", "TEXA~
## $ venue_postal_cd_sgmt_1  <chr> "2048", "98848", "85003", "75210", "98092", "92~
## $ sales_platform_cd       <chr> "www.concerts.livenation.com", "NULL", "NULL", ~
## $ print_flg               <chr> "T ", "T ", "T ", "T ", "T ", "T ", "T ", "T ",~
## $ la_valid_tkt_event_flg  <chr> "N ", "N ", "N ", "N ", "N ", "N ", "N ", "N ",~
## $ fin_mkt_nm              <chr> "Boston", "Seattle", "Arizona", "Dallas", "Seat~
## $ web_session_cookie_val  <chr> "7dfa56dd7d5956b17587", "4f9e6fc637eaf7b736c2",~
## $ gndr_cd                 <chr> NA, NA, NA, NA, NA, NA, "M", NA, NA, NA, "M", N~
## $ age_yr                  <chr> NA, NA, NA, NA, NA, NA, "28", NA, NA, NA, "80",~
## $ income_amt              <chr> NA, NA, NA, NA, NA, NA, "112500", NA, NA, NA, "~
## $ edu_val                 <chr> NA, NA, NA, NA, NA, NA, "High School", NA, NA, ~
## $ edu_1st_indv_val        <chr> NA, NA, NA, NA, NA, NA, "High School", NA, NA, ~
```

```
## $ edu_2nd_indv_val       <chr> NA, NA, NA, NA, NA, NA, "NULL", NA, NA, NA, "Hi~
## $ adults_in_hh_num       <chr> NA, NA, NA, NA, NA, NA, "4", NA, NA, NA, "2", N~
## $ married_ind            <chr> NA, NA, NA, NA, NA, NA, "0", NA, NA, NA, "1", N~
## $ child_present_ind      <chr> NA, NA, NA, NA, NA, NA, "1", NA, NA, NA, "NULL"~
## $ home_owner_ind         <chr> NA, NA, NA, NA, NA, NA, "0", NA, NA, NA, "1", N~
## $ occpn_val              <chr> NA, NA, NA, NA, NA, NA, "NULL", NA, NA, NA, "Re~
## $ occpn_1st_val          <chr> NA, NA, NA, NA, NA, NA, "Craftsman Blue Collar"~
## $ occpn_2nd_val          <chr> NA, NA, NA, NA, NA, NA, "NULL", NA, NA, NA, "Re~
## $ dist_to_ven            <int> NA, 59, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
```

Removing redundant info

You may have noticed that the first column of data is just a duplication of the row numbers. Not very useful. Go ahead and delete that column. Remember that nrow() and ncol() return the number of rows and columns in a data frame, respectively.

4. Take a subset of sales to omit the first column. Assign the result to sales2.

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

Information not worth keeping

Many of the columns have information that's of no use to us. For example, the first four columns contain internal codes representing particular events. The last fifteen columns also aren't worth keeping; there are too many missing values to make them worthwhile.

An easy way to get rid of unnecessary columns is to create a vector containing the column indices you want to keep, then subset the data based on that vector using single bracket subsetting.

5. Create a vector called keep that contains the indices of the columns you want to save.Remember:you want to keep everything besides the first 4 and last 15 columns of sales2. Subset the columns of sales2 using your vector and assign the result to sales3.

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

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

Separating columns

Some of the columns in your data frame include multiple pieces of information that should be in separate columns. In this exercise, you will separate such a column into two: one for date and one for time. You will use the separate() function from the tidyr package (already installed for you).

Take a look at the event_date_time column by typing head(sales3$event_date_time) in the console. You'll notice that the date and time are separated by a space. Therefore, you'll use sep = " " as an argument to separate().

6. Load the tidyr package. Split the event_date_time column of sales3 into "event_dt" and "event_time". Assign the result to sales4. Split the sales_ord_create_dttm column of sales4 into "ord_create_dt" and "ord_create_time". Assign the result to sales5.

```
# 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=" ")
```

```
## Warning: Expected 2 pieces. Missing pieces filled with `NA` in 4 rows [2516,
## 3863, 4082, 4183].
```

Dealing with warnings

Looks like that second call to separate() threw a warning. Not to worry; warnings aren't as bad as error messages. It's not saying that the command didn't execute; it's just a heads-up that something unusual happened.

The warning says Too few values at 4 locations. You may be able to guess already what the issue is, but it's still good to take a look.

The locations (i.e. rows) given in the warning are 2516, 3863, 4082, and 4183. Have a look at the contents of the sales_ord_create_dttm column in those rows. 7. Assign a vector issues that contains the indices of the four troublesome rows: 2516, 3863, 4082, and 4183. Subset $sales3sales_ord_create_dttm$ to look at these observations. Remember to use sales3 (not sales4), since you want the data frame from before you separated columns! For comparison, print element 2517 of sales3$sales_ord_create_dttm, which did not cause a warning.

```
# 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]
```

```
## [1] "NULL" "NULL" "NULL" "NULL"
```

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

```
## [1] "04-08-2013 23:07"
```

Identifying dates

Some of the columns in your dataset contain dates of different events. Right now, they are stored as character strings. That's fine if all you want to do is look up the date associated with an event, but if you want to do any comparisons or math with the dates, it's MUCH easier to store them as Date objects.

Luckily, all of the date columns in this dataset have the substring "dt" in their name, so you can use the str_detect() function of the stringr package to find the date columns. Then you can coerce them to Date objects using a function from the lubridate package.

You'll use lapply() to apply the appropriate lubridate function to all of the columns that contain dates. Recall the following syntax for lapply() applied to some data frame columns of interest:

lapply(my_data_frame[, cols], function_name) Also recall that function names in lubridate combine the letters y, m, d, h, m, and s depending on the format of the date/time string being read in. 8. Load the stringr package. Use str_detect() to find values in the names() of sales5 containing the string "dt". Assign the resulting logical vector to the variable date_cols. Load the lubridate package. Coerce the date_cols into Date objects using lapply() and the appropriate function from lubridate. Conveniently, all date columns in sales5 are in year-month-day format, so you can use the ymd() function from lubridate.

```
# 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)
```

```
## Warning: All formats failed to parse. No formats found.

## Warning: All formats failed to parse. No formats found.

## Warning: All formats failed to parse. No formats found.

## Warning: All formats failed to parse. No formats found.

## Warning: All formats failed to parse. No formats found.

## Warning: All formats failed to parse. No formats found.
```

More warnings!

As you saw, some of the calls to ymd() caused a failure to parse warning. That's probably because of more missing data, but again, it's good to check to be sure.

The first two lines of code (provided for you here) create a list of logical vectors called missing. Each vector in the list indicates the presence (or absence) of missing values in the corresponding column of sales5. See if the number of missing values in each column is the same as the number of rows that failed to parse in the previous exercise. 9. Run the first line as-is to regenerate the date_cols vector. Run the second line as-is to generate a list of logical vectors representing missing values in the date columns of sales5. Use sapply() to create a numerical vector containing the number of NA values in each date column. Call this vector num_missing. Print out the num_missing vector.

```
# 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
print(num_missing)
```

```
##          event_dt        presale_dt         onsale_dt     ord_create_dt
##              5000              5000              5000              5000
## sales_ord_tran_dt          print_dt
##              5000              5000
```

Combining columns

Sure enough, the number of NAs in each column match the numbers from the warning messages, so missing data is the culprit. How to proceed depends on your desired analysis. If you really need complete sets of date/time information, you might delete the rows or columns containing NAs.

As your last step, you'll use the tidyr function unite() to combine the venue_city and venue_state columns into one column with the two values separated by a comma and a space. For example, "PORTLAND" "MAINE" should become "PORTLAND, MAINE".

10. Combine the venue_city and venue_state columns of sales5 into a new column called venue_city_state, containing the city and state names separated by a comma and a space. Call the resulting data frame sales6. View the first 6 rows of sales6.

```
# 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)
```

```
##                                                    event_name
## 1 Xfinity Center Mansfield Premier Parking: Florida Georgia Line
## 2                  Gorge Camping - dave matthews band - sept 3-7
## 3                    Dodge Theatre Adams Street Parking - benise
## 4   Gexa Energy Pavilion Vip Parking : kid rock with sheryl crow
## 5                                 Premier Parking - motley crue
## 6                                     Fast Lane Access: Journey
##                          primary_act_name secondary_act_name major_cat_name
## 1 XFINITY Center Mansfield Premier Parking               NULL           MISC
## 2                           Gorge Camping Dave Matthews Band           MISC
## 3                           Parking Event               NULL           MISC
## 4         Gexa Energy Pavilion VIP Parking               NULL           MISC
## 5 White River Amphitheatre Premier Parking               NULL           MISC
## 6                         Fast Lane Access            Journey           MISC
##           minor_cat_name la_event_type_cat
## 1                PARKING           PARKING
## 2                CAMPING           INVALID
## 3                PARKING           PARKING
## 4                PARKING           PARKING
## 5                PARKING           PARKING
## 6 SPECIAL ENTRY (UPSELL)            UPSELL
##                                            event_disp_name
## 1 Xfinity Center Mansfield Premier Parking: Florida Georgia Line
## 2                  Gorge Camping - dave matthews band - sept 3-7
## 3                    Dodge Theatre Adams Street Parking - benise
## 4   Gexa Energy Pavilion Vip Parking : kid rock with sheryl crow
## 5                                 Premier Parking - motley crue
## 6                                     Fast Lane Access: Journey
##
```

```
## 1    THIS TICKET IS VALID        FOR PARKING ONLY         GOOD THIS DAY ONLY       PREMIER PARKING PA
## 2                                                                %OVERNIGHT C A M P I N G%* * * * * *
## 3                             ADAMS STREET GARAGE%PARKING FOR 4/21/06 ONLY%DODGE THEATRE PARKING PA
## 4    THIS TICKET IS VALID        FOR PARKING ONLY      GOOD FOR THIS DATE ONLY       VIP PARKING PASS
## 5                            THIS TICKET IS VALID%FOR PARKING ONLY%GOOD THIS DATE ONLY%PREMIER PARK
## 6         FAST LANE                  JOURNEY              FAST LANE EVENT         THIS IS NOT A TICK
##   tickets_purchased_qty trans_face_val_amt delivery_type_cd event_dt event_time
## 1                     1                45          eTicket     <NA>      23:30
## 2                     1                75       TicketFast     <NA>      01:00
## 3                     1                 5       TicketFast     <NA>      01:30
## 4                     1                20             Mail     <NA>      00:00
## 5                     1                20             Mail     <NA>      01:00
## 6                     2                10       TicketFast     <NA>      02:00
##   presale_dt onsale_dt ord_create_dt ord_create_time sales_ord_tran_dt print_dt
## 1       <NA>      <NA>          <NA>           18:17              <NA>     <NA>
## 2       <NA>      <NA>          <NA>           00:00              <NA>     <NA>
## 3       <NA>      <NA>          <NA>           00:00              <NA>     <NA>
## 4       <NA>      <NA>          <NA>           17:38              <NA>     <NA>
## 5       <NA>      <NA>          <NA>           00:00              <NA>     <NA>
## 6       <NA>      <NA>          <NA>           17:20              <NA>     <NA>
##   timezn_nm           venue_city_state venue_postal_cd_sgmt_1
## 1       EST   MANSFIELD, MASSACHUSETTS                  2048
## 2       PST         QUINCY, WASHINGTON                 98848
## 3       MST           PHOENIX, ARIZONA                 85003
## 4       CST              DALLAS, TEXAS                 75210
## 5       PST         AUBURN, WASHINGTON                 98092
## 6       PST SAN BERNARDINO, CALIFORNIA                 92407
##             sales_platform_cd print_flg la_valid_tkt_event_flg  fin_mkt_nm
## 1 www.concerts.livenation.com         T                      N       Boston
## 2                        NULL         T                      N      Seattle
## 3                        NULL         T                      N      Arizona
## 4                        NULL         T                      N       Dallas
## 5                        NULL         T                      N      Seattle
## 6          www.livenation.com         T                      N  Los Angeles
```

Gapminder dataset

Until now, we've been using the nicely formatted original gapminder dataset, but 'real' data (i.e. our own research data) will never be so well organized. Here let's start with the wide formatted version of the gapminder dataset.

```
gap_wide <- read.csv("datasets/gapminder_wide.csv", stringsAsFactors = FALSE)
str(gap_wide)
```

```
## 'data.frame':    142 obs. of  38 variables:
##  $ continent     : chr  "Africa" "Africa" "Africa" "Africa" ...
##  $ country       : chr  "Algeria" "Angola" "Benin" "Botswana" ...
##  $ gdpPercap_1952: num  2449 3521 1063 851 543 ...
##  $ gdpPercap_1957: num  3014 3828 960 918 617 ...
##  $ gdpPercap_1962: num  2551 4269 949 984 723 ...
##  $ gdpPercap_1967: num  3247 5523 1036 1215 795 ...
##  $ gdpPercap_1972: num  4183 5473 1086 2264 855 ...
##  $ gdpPercap_1977: num  4910 3009 1029 3215 743 ...
```

```
##  $ gdpPercap_1982: num  5745 2757 1278 4551 807 ...
##  $ gdpPercap_1987: num  5681 2430 1226 6206 912 ...
##  $ gdpPercap_1992: num  5023 2628 1191 7954 932 ...
##  $ gdpPercap_1997: num  4797 2277 1233 8647 946 ...
##  $ gdpPercap_2002: num  5288 2773 1373 11004 1038 ...
##  $ gdpPercap_2007: num  6223 4797 1441 12570 1217 ...
##  $ lifeExp_1952  : num  43.1 30 38.2 47.6 32 ...
##  $ lifeExp_1957  : num  45.7 32 40.4 49.6 34.9 ...
##  $ lifeExp_1962  : num  48.3 34 42.6 51.5 37.8 ...
##  $ lifeExp_1967  : num  51.4 36 44.9 53.3 40.7 ...
##  $ lifeExp_1972  : num  54.5 37.9 47 56 43.6 ...
##  $ lifeExp_1977  : num  58 39.5 49.2 59.3 46.1 ...
##  $ lifeExp_1982  : num  61.4 39.9 50.9 61.5 48.1 ...
##  $ lifeExp_1987  : num  65.8 39.9 52.3 63.6 49.6 ...
##  $ lifeExp_1992  : num  67.7 40.6 53.9 62.7 50.3 ...
##  $ lifeExp_1997  : num  69.2 41 54.8 52.6 50.3 ...
##  $ lifeExp_2002  : num  71 41 54.4 46.6 50.6 ...
##  $ lifeExp_2007  : num  72.3 42.7 56.7 50.7 52.3 ...
##  $ pop_1952      : num  9279525 4232095 1738315 442308 4469979 ...
##  $ pop_1957      : num  10270856 4561361 1925173 474639 4713416 ...
##  $ pop_1962      : num  11000948 4826015 2151895 512764 4919632 ...
##  $ pop_1967      : num  12760499 5247469 2427334 553541 5127935 ...
##  $ pop_1972      : num  14760787 5894858 2761407 619351 5433886 ...
##  $ pop_1977      : num  17152804 6162675 3168267 781472 5889574 ...
##  $ pop_1982      : num  20033753 7016384 3641603 970347 6634596 ...
##  $ pop_1987      : num  23254956 7874230 4243788 1151184 7586551 ...
##  $ pop_1992      : num  26298373 8735988 4981671 1342614 8878303 ...
##  $ pop_1997      : num  29072015 9875024 6066080 1536536 10352843 ...
##  $ pop_2002      : int  31287142 10866106 7026113 1630347 12251209 7021078 15929988 4048013 8835739 (
##  $ pop_2007      : int  33333216 12420476 8078314 1639131 14326203 8390505 17696293 4369038 10238807
```

To change this very wide dataframe layout back to our nice, intermediate (or longer) layout, we will use one of the two available pivot functions from the tidyr package. To convert from wide to a longer format, we will use the pivot_longer() function. pivot_longer() makes datasets longer by increasing the number of rows and decreasing the number of columns, or 'lengthening' your observation variables into a single variable.

Here we we can use piping syntax which is similar to what we were doing in the previous lesson with dplyr. In fact, these are compatible and you can use a mix of tidyr and dplyr functions by piping them together.

We first provide to pivot_longer() a vector of column names that will be pivoted into longer format. We could type out all the observation variables, but as in the select() function (see dplyr lesson), we can use the starts_with() argument to select all variables that start with the desired character string. pivot_longer() also allows the alternative syntax of using the - symbol to identify which variables are not to be pivoted (i.e. ID variables).

The next arguments to pivot_longer() are names_to for naming the column that will contain the new ID variable (obstype_year) and values_to for naming the new amalgamated observation variable (obs_value). We supply these new column names as strings.

```
gap_long <- gap_wide %>%
  pivot_longer(
    cols = c(starts_with('pop'), starts_with('lifeExp'), starts_with('gdpPercap')),
    names_to = "obstype_year", values_to = "obs_values"
  )
str(gap_long)
```

```
## tibble [5,112 x 4] (S3: tbl_df/tbl/data.frame)
```

```
##  $ continent   : chr [1:5112] "Africa" "Africa" "Africa" "Africa" ...
##  $ country     : chr [1:5112] "Algeria" "Algeria" "Algeria" "Algeria" ...
##  $ obstype_year: chr [1:5112] "pop_1952" "pop_1957" "pop_1962" "pop_1967" ...
##  $ obs_values  : num [1:5112] 9279525 10270856 11000948 12760499 14760787 ...
```

Now obstype_year actually contains 2 pieces of information, the observation type (pop,lifeExp, or gdpPercap) and the year. We can use the separate() function to split the character strings into multiple variables

```
gap_long <- gap_long %>% separate(obstype_year, into = c('obs_type', 'year'), sep = "_")
gap_long$year <- as.integer(gap_long$year)
```

11. Using gap_long, calculate the mean life expectancy, population, and gdpPercap for each continent. Hint: use the group_by() and summarize() functions we learned in the dplyr lesson

```
gap_long %>% group_by(continent, obs_type) %>%
   summarize(means=mean(obs_values))
```

```
## `summarise()` has grouped output by 'continent'. You can override using the `.groups` argument.
```

```
## # A tibble: 15 x 3
## # Groups:   continent [5]
##    continent obs_type       means
##    <chr>     <chr>          <dbl>
##  1 Africa    gdpPercap     2194.
##  2 Africa    lifeExp         48.9
##  3 Africa    pop        9916003.
##  4 Americas  gdpPercap     7136.
##  5 Americas  lifeExp         64.7
##  6 Americas  pop       24504795.
##  7 Asia      gdpPercap     7902.
##  8 Asia      lifeExp         60.1
##  9 Asia      pop       77038722.
## 10 Europe    gdpPercap    14469.
## 11 Europe    lifeExp         71.9
## 12 Europe    pop       17169765.
## 13 Oceania   gdpPercap    18622.
## 14 Oceania   lifeExp         74.3
## 15 Oceania   pop        8874672.
```

It is always good to check work. So, let's use the second pivot function, pivot_wider(), to 'widen' our observation variables back out. pivot_wider() is the opposite of pivot_longer(), making a dataset wider by increasing the number of columns and decreasing the number of rows. We can use pivot_wider() to pivot or reshape our gap_long to the original intermediate format or the widest format. Let's start with the intermediate format.

The pivot_wider() function takes names_from and values_from arguments.

To names_from we supply the column name whose contents will be pivoted into new output columns in the widened dataframe. The corresponding values will be added from the column named in the values_from argument.

```
gap_normal <- gap_long %>%
  pivot_wider(names_from = obs_type, values_from = obs_values)
dim(gap_normal)
```

```
## [1] 1704    6
```

Cleaning the weather dataset

As mentioned in the beginning of this document, we started off with a messy dataset. Now lets try to clean it with whatever we ahve learned. After understanding the structure of the data and looking at some brief summaries, it often helps to preview the actual data. The functions head() and tail() allow you to view the top and bottom rows of the data, respectively. Recall you'll be shown 6 rows by default, but you can alter this behavior with a second argument to the function.

```
# View first 6 rows
head(weather)
```

```
##   X year month           measure X1 X2 X3 X4 X5 X6 X7 X8 X9 X10 X11 X12 X13 X14
## 1 1 2014    12  Max.TemperatureF 64 42 51 43 42 45 38 29 49  48  39  39  42  45
## 2 2 2014    12 Mean.TemperatureF 52 38 44 37 34 42 30 24 39  43  36  35  37  39
## 3 3 2014    12  Min.TemperatureF 39 33 37 30 26 38 21 18 29  38  32  31  32  33
## 4 4 2014    12    Max.Dew.PointF 46 40 49 24 37 45 36 28 49  45  37  28  28  29
## 5 5 2014    12    MeanDew.PointF 40 27 42 21 25 40 20 16 41  39  31  27  26  27
## 6 6 2014    12     Min.DewpointF 26 17 24 13 12 36 -3  3 28  37  27  25  24  25
##   X15 X16 X17 X18 X19 X20 X21 X22 X23 X24 X25 X26 X27 X28 X29 X30 X31
## 1  42  44  49  44  37  36  36  44  47  46  59  50  52  52  41  30  30
## 2  37  40  45  40  33  32  33  39  45  44  52  44  45  46  36  26  25
## 3  32  35  41  36  29  27  30  33  42  41  44  37  38  40  30  22  20
## 4  33  42  46  34  25  30  30  39  45  46  58  31  34  42  26  10   8
## 5  29  36  41  30  22  24  27  34  42  44  43  29  31  35  20   4   5
## 6  27  30  32  26  20  20  25  25  37  41  29  28  29  27  10  -6   1
```

```
# View first 15 rows
head(weather,15)
```

```
##     X year month                    measure    X1    X2    X3    X4    X5    X6
## 1   1 2014    12           Max.TemperatureF    64    42    51    43    42    45
## 2   2 2014    12          Mean.TemperatureF    52    38    44    37    34    42
## 3   3 2014    12           Min.TemperatureF    39    33    37    30    26    38
## 4   4 2014    12             Max.Dew.PointF    46    40    49    24    37    45
## 5   5 2014    12             MeanDew.PointF    40    27    42    21    25    40
## 6   6 2014    12              Min.DewpointF    26    17    24    13    12    36
## 7   7 2014    12                Max.Humidity    74    92   100    69    85   100
## 8   8 2014    12               Mean.Humidity    63    72    79    54    66    93
## 9   9 2014    12                Min.Humidity    52    51    57    39    47    85
## 10 10 2014    12  Max.Sea.Level.PressureIn 30.45 30.71  30.4 30.56 30.68 30.42
## 11 11 2014    12 Mean.Sea.Level.PressureIn 30.13 30.59 30.07 30.33 30.59 30.24
## 12 12 2014    12  Min.Sea.Level.PressureIn 30.01  30.4 29.87 30.09 30.45 30.16
## 13 13 2014    12       Max.VisibilityMiles    10    10    10    10    10    10
## 14 14 2014    12      Mean.VisibilityMiles    10     8     5    10    10     4
## 15 15 2014    12       Min.VisibilityMiles    10     2     1    10     5     0
##       X7    X8    X9   X10   X11   X12   X13   X14   X15   X16   X17   X18
## 1     38    29    49    48    39    39    42    45    42    44    49    44
## 2     30    24    39    43    36    35    37    39    37    40    45    40
## 3     21    18    29    38    32    31    32    33    32    35    41    36
## 4     36    28    49    45    37    28    28    29    33    42    46    34
## 5     20    16    41    39    31    27    26    27    29    36    41    30
## 6     -3     3    28    37    27    25    24    25    27    30    32    26
## 7     92    92   100   100    92    85    75    82    89    96   100    89
## 8     61    70    93    95    87    75    65    68    75    85    85    73
## 9     29    47    86    89    82    64    55    53    60    73    70    57
```

```
## 10 30.69 30.77 30.51 29.58 29.81 29.88 29.86 29.91 30.15 30.17 29.91 29.87
## 11 30.46 30.67 30.04  29.5 29.61 29.85 29.82 29.83 30.05 30.09 29.75 29.78
## 12 30.24 30.51 29.49 29.43 29.44 29.81 29.78 29.78 29.91 29.92 29.69 29.71
## 13    10    10    10    10    10    10    10    10    10    10    10    10
## 14    10     8     2     3     7    10    10    10    10     9     6    10
## 15     5     2     1     1     1     7    10    10    10     5     1    10
##      X19   X20   X21   X22   X23   X24   X25   X26   X27   X28   X29   X30
## 1     37    36    36    44    47    46    59    50    52    52    41    30
## 2     33    32    33    39    45    44    52    44    45    46    36    26
## 3     29    27    30    33    42    41    44    37    38    40    30    22
## 4     25    30    30    39    45    46    58    31    34    42    26    10
## 5     22    24    27    34    42    44    43    29    31    35    20     4
## 6     20    20    25    25    37    41    29    28    29    27    10    -6
## 7     69    89    85    89   100   100   100    70    70    76    64    50
## 8     63    79    77    79    91    98    75    60    60    65    51    38
## 9     56    69    69    69    82    96    49    49    50    53    37    26
## 10 30.15 30.31 30.37  30.4 30.31 30.13 29.96 30.16 30.22 29.99 30.22 30.36
## 11 29.98 30.26 30.32 30.35 30.23  29.9 29.63 30.11 30.14 29.87 30.12 30.32
## 12 29.86 30.17 30.28  30.3 30.16 29.55 29.47 29.99 30.03 29.77    30 30.23
## 13    10    10    10    10    10     2    10    10    10    10    10    10
## 14    10    10     9    10     5     1     8    10    10    10    10    10
## 15    10     7     6     4     1     0     1    10    10    10    10    10
##      X31
## 1     30
## 2     25
## 3     20
## 4      8
## 5      5
## 6      1
## 7     57
## 8     44
## 9     31
## 10 30.32
## 11 30.25
## 12 30.13
## 13    10
## 14    10
## 15    10
```

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

```
##       X year month            measure   X1   X2   X3   X4   X5   X6   X7   X8
## 281 281 2015    12 Mean.Wind.SpeedMPH    6 <NA> <NA> <NA> <NA> <NA> <NA> <NA>
## 282 282 2015    12  Max.Gust.SpeedMPH   17 <NA> <NA> <NA> <NA> <NA> <NA> <NA>
## 283 283 2015    12    PrecipitationIn 0.14 <NA> <NA> <NA> <NA> <NA> <NA> <NA>
## 284 284 2015    12         CloudCover    7 <NA> <NA> <NA> <NA> <NA> <NA> <NA>
## 285 285 2015    12             Events Rain <NA> <NA> <NA> <NA> <NA> <NA> <NA>
## 286 286 2015    12     WindDirDegrees  109 <NA> <NA> <NA> <NA> <NA> <NA> <NA>
##       X9  X10  X11  X12  X13  X14  X15  X16  X17  X18  X19  X20  X21  X22  X23
## 281 <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
## 282 <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
## 283 <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
## 284 <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
## 285 <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
```

```
## 286 <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
##      X24  X25  X26  X27  X28  X29  X30  X31
## 281 <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
## 282 <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
## 283 <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
## 284 <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
## 285 <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
## 286 <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
```

```
# View the last 10 rows
tail(weather,10)
```

```
##       X year month              measure   X1   X2   X3   X4   X5   X6   X7   X8
## 277 277 2015    12  Max.VisibilityMiles   10 <NA> <NA> <NA> <NA> <NA> <NA> <NA>
## 278 278 2015    12 Mean.VisibilityMiles    8 <NA> <NA> <NA> <NA> <NA> <NA> <NA>
## 279 279 2015    12  Min.VisibilityMiles    1 <NA> <NA> <NA> <NA> <NA> <NA> <NA>
## 280 280 2015    12    Max.Wind.SpeedMPH   15 <NA> <NA> <NA> <NA> <NA> <NA> <NA>
## 281 281 2015    12   Mean.Wind.SpeedMPH    6 <NA> <NA> <NA> <NA> <NA> <NA> <NA>
## 282 282 2015    12    Max.Gust.SpeedMPH   17 <NA> <NA> <NA> <NA> <NA> <NA> <NA>
## 283 283 2015    12      PrecipitationIn 0.14 <NA> <NA> <NA> <NA> <NA> <NA> <NA>
## 284 284 2015    12           CloudCover    7 <NA> <NA> <NA> <NA> <NA> <NA> <NA>
## 285 285 2015    12              Events Rain <NA> <NA> <NA> <NA> <NA> <NA> <NA>
## 286 286 2015    12       WindDirDegrees  109 <NA> <NA> <NA> <NA> <NA> <NA> <NA>
##       X9  X10  X11  X12  X13  X14  X15  X16  X17  X18  X19  X20  X21  X22  X23
## 277 <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
## 278 <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
## 279 <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
## 280 <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
## 281 <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
## 282 <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
## 283 <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
## 284 <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
## 285 <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
## 286 <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
##      X24  X25  X26  X27  X28  X29  X30  X31
## 277 <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
## 278 <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
## 279 <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
## 280 <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
## 281 <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
## 282 <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
## 283 <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
## 284 <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
## 285 <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
## 286 <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
```

Here too Column names are values

The weather dataset suffers from one of the five most common symptoms of messy data: column names are values. In particular, the column names X1-X31 represent days of the month, which should really be values of a new variable called day.

The tidyr package provides the gather() function for exactly this scenario. To remind you of how it works, we've loaded a small dataset called df in your workspace. Give the following a try in the console before

attempting the instructions below.

gather(df, time, val, t1:t3) Notice that gather() allows you to select multiple columns to be gathered by using the : operator.

11. Load the tidyr package.Call gather() on the weather data to gather columns X1-X31. The two columns created as a result should be called day and value. Save the result as weather2. View the result with head().

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

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

# View the head
head(weather2)
```

```
##   X year month           measure day value
## 1 1 2014    12  Max.TemperatureF  X1    64
## 2 2 2014    12 Mean.TemperatureF  X1    52
## 3 3 2014    12  Min.TemperatureF  X1    39
## 4 4 2014    12    Max.Dew.PointF  X1    46
## 5 5 2014    12    MeanDew.PointF  X1    40
## 6 6 2014    12     Min.DewpointF  X1    26
```

Values are variable names

Our data suffer from a second common symptom of messy data: values are variable names. Specifically, values in the measure column should be variables (i.e. column names) in our dataset.

The spread() function from tidyr is designed to help with this. To remind you of how this function works, we've loaded another small dataset called df2 (which is the result of applying gather() to the original df from last exercise). Give the following a try before attempting the instructions below.

spread(df2, time, val) Note how the values of the time column now become column names

12. Spread the measure column of without_x and save the result to weather3. View the result with head()

```
# First remove column of row names
without_x <- weather2[, -1]

# Spread the data
weather3 <- spread(without_x, measure,value)

# View the head
head(weather3)
```

```
##   year month day CloudCover    Events Max.Dew.PointF Max.Gust.SpeedMPH
## 1 2014    12  X1          6      Rain             46                29
## 2 2014    12 X10          8      Rain             45                29
## 3 2014    12 X11          8 Rain-Snow             37                28
## 4 2014    12 X12          7      Snow             28                21
## 5 2014    12 X13          5                       28                23
## 6 2014    12 X14          4                       29                20
##   Max.Humidity Max.Sea.Level.PressureIn Max.TemperatureF Max.VisibilityMiles
## 1           74                    30.45               64                  10
```

```
## 2          100                   29.58               48                   10
## 3           92                   29.81               39                   10
## 4           85                   29.88               39                   10
## 5           75                   29.86               42                   10
## 6           82                   29.91               45                   10
##   Max.Wind.SpeedMPH Mean.Humidity Mean.Sea.Level.PressureIn Mean.TemperatureF
## 1                22            63                     30.13                52
## 2                23            95                      29.5                43
## 3                21            87                     29.61                36
## 4                16            75                     29.85                35
## 5                17            65                     29.82                37
## 6                15            68                     29.83                39
##   Mean.VisibilityMiles Mean.Wind.SpeedMPH MeanDew.PointF Min.DewpointF
## 1                   10                 13             40            26
## 2                    3                 13             39            37
## 3                    7                 13             31            27
## 4                   10                 11             27            25
## 5                   10                 12             26            24
## 6                   10                 10             27            25
##   Min.Humidity Min.Sea.Level.PressureIn Min.TemperatureF Min.VisibilityMiles
## 1           52                   30.01               39                   10
## 2           89                   29.43               38                    1
## 3           82                   29.44               32                    1
## 4           64                   29.81               31                    7
## 5           55                   29.78               32                   10
## 6           53                   29.78               33                   10
##   PrecipitationIn WindDirDegrees
## 1            0.01            268
## 2            0.28            357
## 3            0.02            230
## 4               T            286
## 5               T            298
## 6            0.00            306
```

Clean up dates

Now that the weather dataset adheres to tidy data principles, the next step is to prepare it for analysis. We'll start by combining the year, month, and day columns and recoding the resulting character column as a date. We can use a combination of base R, stringr, and lubridate to accomplish this task. tidyr and dplyr are already loaded.

13. Load the stringr and lubridate packages. Use stringr's str_replace() to remove the Xs from the day column of weather3. Create a new column called date. Use the unite() function from tidyr to paste together the year, month, and day columns in order, using - as a separator (see ?unite if you need help). Coerce the date column using the appropriate function from lubridate. Use the code provided (select()) to reorder columns, saving the result to weather5. View the head of weather5

```
# Load the stringr and lubridate packages
library(stringr)
library(lubridate)


# Remove X's from day column
weather3$day <- str_replace(weather3$day,"X","")
```

```r
# 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)
```

```
##         date    Events CloudCover Max.Dew.PointF Max.Gust.SpeedMPH Max.Humidity
## 1 2014-12-01      Rain          6             46                29           74
## 2 2014-12-10      Rain          8             45                29          100
## 3 2014-12-11 Rain-Snow          8             37                28           92
## 4 2014-12-12      Snow          7             28                21           85
## 5 2014-12-13                    5             28                23           75
## 6 2014-12-14                    4             29                20           82
##   Max.Sea.Level.PressureIn Max.TemperatureF Max.VisibilityMiles
## 1                    30.45              64                  10
## 2                    29.58              48                  10
## 3                    29.81              39                  10
## 4                    29.88              39                  10
## 5                    29.86              42                  10
## 6                    29.91              45                  10
##   Max.Wind.SpeedMPH Mean.Humidity Mean.Sea.Level.PressureIn Mean.TemperatureF
## 1                22            63                     30.13                52
## 2                23            95                      29.5                43
## 3                21            87                     29.61                36
## 4                16            75                     29.85                35
## 5                17            65                     29.82                37
## 6                15            68                     29.83                39
##   Mean.VisibilityMiles Mean.Wind.SpeedMPH MeanDew.PointF Min.DewpointF
## 1                   10                 13             40            26
## 2                    3                 13             39            37
## 3                    7                 13             31            27
## 4                   10                 11             27            25
## 5                   10                 12             26            24
## 6                   10                 10             27            25
##   Min.Humidity Min.Sea.Level.PressureIn Min.TemperatureF Min.VisibilityMiles
## 1           52                    30.01              39                  10
## 2           89                    29.43              38                   1
## 3           82                    29.44              32                   1
## 4           64                    29.81              31                   7
## 5           55                    29.78              32                  10
## 6           53                    29.78              33                  10
##   PrecipitationIn WindDirDegrees
## 1            0.01            268
## 2            0.28            357
## 3            0.02            230
## 4               T            286
## 5               T            298
```

```
## 6            0.00            306
```

A closer look at column types

It's important for analysis that variables are coded appropriately. This is not yet the case with our weather data. Recall that functions such as as.numeric() and as.character() can be used to coerce variables into different types.

It's important to keep in mind that coercions are not always successful, particularly if there's some data in a column that you don't expect. For example, the following will cause problems:

as.numeric(c(4, 6.44, "some string", 222)) If you run the code above in the console, you'll get a warning message saying that R introduced an NA in the process of coercing to numeric. This is because it doesn't know how to make a number out of a string ("some string"). Watch out for this in our weather data!

14. Use str() to see how variables are stored in weather5. View the first 20 rows of weather5. Keep an eye out for strange values! Try coercing the PrecipitationIn column of weather5 to numeric without saving the result.

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

```
## 'data.frame':    366 obs. of  23 variables:
##  $ date                     : Date, format: "2014-12-01" "2014-12-10" ...
##  $ Events                   : chr  "Rain" "Rain" "Rain-Snow" "Snow" ...
##  $ CloudCover               : chr  "6" "8" "8" "7" ...
##  $ Max.Dew.PointF           : chr  "46" "45" "37" "28" ...
##  $ Max.Gust.SpeedMPH        : chr  "29" "29" "28" "21" ...
##  $ Max.Humidity             : chr  "74" "100" "92" "85" ...
##  $ Max.Sea.Level.PressureIn : chr  "30.45" "29.58" "29.81" "29.88" ...
##  $ Max.TemperatureF         : chr  "64" "48" "39" "39" ...
##  $ Max.VisibilityMiles      : chr  "10" "10" "10" "10" ...
##  $ Max.Wind.SpeedMPH        : chr  "22" "23" "21" "16" ...
##  $ Mean.Humidity            : chr  "63" "95" "87" "75" ...
##  $ Mean.Sea.Level.PressureIn: chr  "30.13" "29.5" "29.61" "29.85" ...
##  $ Mean.TemperatureF        : chr  "52" "43" "36" "35" ...
##  $ Mean.VisibilityMiles     : chr  "10" "3" "7" "10" ...
##  $ Mean.Wind.SpeedMPH       : chr  "13" "13" "13" "11" ...
##  $ MeanDew.PointF           : chr  "40" "39" "31" "27" ...
##  $ Min.DewpointF            : chr  "26" "37" "27" "25" ...
##  $ Min.Humidity             : chr  "52" "89" "82" "64" ...
##  $ Min.Sea.Level.PressureIn : chr  "30.01" "29.43" "29.44" "29.81" ...
##  $ Min.TemperatureF         : chr  "39" "38" "32" "31" ...
##  $ Min.VisibilityMiles      : chr  "10" "1" "1" "7" ...
##  $ PrecipitationIn          : chr  "0.01" "0.28" "0.02" "T" ...
##  $ WindDirDegrees           : chr  "268" "357" "230" "286" ...
```

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

```
##          date    Events CloudCover Max.Dew.PointF Max.Gust.SpeedMPH
## 1  2014-12-01      Rain          6             46                29
## 2  2014-12-10      Rain          8             45                29
## 3  2014-12-11 Rain-Snow          8             37                28
## 4  2014-12-12      Snow          7             28                21
## 5  2014-12-13                    5             28                23
```

```
## 6  2014-12-14                   4             29                20
## 7  2014-12-15                   2             33                21
## 8  2014-12-16      Rain         8             42                10
## 9  2014-12-17      Rain         8             46                26
## 10 2014-12-18      Rain         7             34                30
## 11 2014-12-19                   4             25                23
## 12 2014-12-02 Rain-Snow         7             40                29
## 13 2014-12-20      Snow         6             30                26
## 14 2014-12-21      Snow         8             30                20
## 15 2014-12-22      Rain         7             39                22
## 16 2014-12-23      Rain         8             45                25
## 17 2014-12-24  Fog-Rain         8             46                15
## 18 2014-12-25      Rain         6             58                40
## 19 2014-12-26                   1             31                25
## 20 2014-12-27                   3             34                21
##    Max.Humidity Max.Sea.Level.PressureIn Max.TemperatureF Max.VisibilityMiles
## 1            74                    30.45               64                  10
## 2           100                    29.58               48                  10
## 3            92                    29.81               39                  10
## 4            85                    29.88               39                  10
## 5            75                    29.86               42                  10
## 6            82                    29.91               45                  10
## 7            89                    30.15               42                  10
## 8            96                    30.17               44                  10
## 9           100                    29.91               49                  10
## 10           89                    29.87               44                  10
## 11           69                    30.15               37                  10
## 12           92                    30.71               42                  10
## 13           89                    30.31               36                  10
## 14           85                    30.37               36                  10
## 15           89                     30.4               44                  10
## 16          100                    30.31               47                  10
## 17          100                    30.13               46                   2
## 18          100                    29.96               59                  10
## 19           70                    30.16               50                  10
## 20           70                    30.22               52                  10
##    Max.Wind.SpeedMPH Mean.Humidity Mean.Sea.Level.PressureIn Mean.TemperatureF
## 1                 22            63                    30.13                52
## 2                 23            95                     29.5                43
## 3                 21            87                    29.61                36
## 4                 16            75                    29.85                35
## 5                 17            65                    29.82                37
## 6                 15            68                    29.83                39
## 7                 15            75                    30.05                37
## 8                  8            85                    30.09                40
## 9                 20            85                    29.75                45
## 10                23            73                    29.78                40
## 11                17            63                    29.98                33
## 12                24            72                    30.59                38
## 13                21            79                    30.26                32
## 14                16            77                    30.32                33
## 15                18            79                    30.35                39
## 16                20            91                    30.23                45
## 17                13            98                     29.9                44
```

```
## 18               28           75                   29.63               52
## 19               18           60                   30.11               44
## 20               17           60                   30.14               45
##    Mean.VisibilityMiles Mean.Wind.SpeedMPH MeanDew.PointF Min.DewpointF
## 1                    10                 13             40            26
## 2                     3                 13             39            37
## 3                     7                 13             31            27
## 4                    10                 11             27            25
## 5                    10                 12             26            24
## 6                    10                 10             27            25
## 7                    10                  6             29            27
## 8                     9                  4             36            30
## 9                     6                 11             41            32
## 10                   10                 14             30            26
## 11                   10                 11             22            20
## 12                    8                 15             27            17
## 13                   10                 10             24            20
## 14                    9                  9             27            25
## 15                   10                  8             34            25
## 16                    5                 13             42            37
## 17                    1                  6             44            41
## 18                    8                 14             43            29
## 19                   10                 11             29            28
## 20                   10                  9             31            29
##    Min.Humidity Min.Sea.Level.PressureIn Min.TemperatureF Min.VisibilityMiles
## 1            52                    30.01              39                  10
## 2            89                    29.43              38                   1
## 3            82                    29.44              32                   1
## 4            64                    29.81              31                   7
## 5            55                    29.78              32                  10
## 6            53                    29.78              33                  10
## 7            60                    29.91              32                  10
## 8            73                    29.92              35                   5
## 9            70                    29.69              41                   1
## 10           57                    29.71              36                  10
## 11           56                    29.86              29                  10
## 12           51                     30.4              33                   2
## 13           69                    30.17              27                   7
## 14           69                    30.28              30                   6
## 15           69                     30.3              33                   4
## 16           82                    30.16              42                   1
## 17           96                    29.55              41                   0
## 18           49                    29.47              44                   1
## 19           49                    29.99              37                  10
## 20           50                    30.03              38                  10
##    PrecipitationIn WindDirDegrees
## 1             0.01            268
## 2             0.28            357
## 3             0.02            230
## 4                T            286
## 5                T            298
## 6             0.00            306
## 7             0.00            324
## 8                T             79
```

```
## 9             0.43           311
## 10            0.01           281
## 11            0.00           305
## 12            0.10            62
## 13               T           350
## 14               T             2
## 15            0.05            24
## 16            0.25            63
## 17            0.56            12
## 18            0.14           250
## 19            0.00           255
## 20            0.00           251
```

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

```
## Warning: NAs introduced by coercion
```

```
##   [1] 0.01 0.28 0.02   NA   NA 0.00 0.00   NA 0.43 0.01 0.00 0.10   NA   NA 0.05
##  [16] 0.25 0.56 0.14 0.00 0.00 0.01 0.00 0.44 0.00 0.00 0.00 0.11 1.09 0.13 0.03
##  [31] 2.90 0.00 0.00 0.00 0.20 0.00   NA 0.12 0.00 0.00 0.15 0.00 0.00 0.00 0.00
##  [46]   NA 0.00 0.71 0.00 0.10 0.95 0.01   NA 0.62 0.06 0.05 0.57 0.00 0.02   NA
##  [61] 0.00 0.01 0.00 0.05 0.01 0.03 0.00 0.23 0.39 0.00 0.02 0.01 0.06 0.78 0.00
##  [76] 0.17 0.11 0.00   NA 0.07 0.02 0.00 0.00 0.00 0.00 0.09   NA 0.07 0.37 0.88
##  [91] 0.17 0.06 0.01 0.00 0.00 0.80 0.27 0.00 0.14 0.00 0.00 0.01 0.05 0.09 0.00
## [106] 0.00 0.00 0.04 0.80 0.21 0.12 0.00 0.26   NA 0.00 0.02   NA 0.00 0.00   NA
## [121] 0.00 0.00 0.09 0.00 0.00 0.00 0.01 0.00 0.00 0.06 0.00 0.00 0.00 0.61 0.54
## [136]   NA 0.00   NA 0.00 0.00 0.10 0.07 0.00 0.03 0.00 0.39 0.00 0.00 0.03 0.26
## [151] 0.09 0.00 0.00 0.00 0.02 0.00 0.00 0.00   NA 0.00 0.00 0.27 0.00 0.00 0.00
## [166]   NA 0.00 0.00   NA 0.00 0.00   NA 0.00 0.00 0.00 0.91 0.00 0.02 0.00 0.00
## [181] 0.00 0.00 0.38 0.00 0.00 0.00   NA 0.00 0.40   NA 0.00 0.00 0.00 0.74 0.04
## [196] 1.72 0.00 0.01 0.00 0.00   NA 0.20 1.43   NA 0.00 0.00 0.00   NA 0.09 0.00
## [211]   NA   NA 0.50 1.12 0.00 0.00 0.00 0.03   NA 0.00   NA 0.14   NA 0.00   NA
## [226]   NA 0.00 0.00 0.01 0.00   NA 0.06 0.00 0.00 0.00 0.02 0.00   NA 0.00 0.00
## [241] 0.02   NA 0.15   NA 0.00 0.83 0.00 0.00 0.00 0.08 0.00 0.00 0.14 0.00 0.00
## [256] 0.00 0.63   NA 0.02   NA 0.00   NA 0.00 0.00 0.00 0.00 0.00 0.00 0.49 0.00
## [271] 0.00 0.00 0.00 0.00 0.00 0.17 0.66 0.01 0.38 0.00 0.00 0.00 0.00 0.00 0.00
## [286] 0.00   NA 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.04 0.01 2.46   NA 0.00
## [301] 0.00 0.00 0.20 0.00   NA 0.00 0.00 0.00 0.12 0.00 0.00   NA   NA   NA 0.00
## [316] 0.08   NA 0.07   NA 0.00 0.00 0.03 0.00 0.00 0.36 0.73 0.01 0.00 0.00 0.00
## [331] 0.00 0.00 0.00 0.00 0.34   NA 0.07 0.54 0.04 0.01 0.00 0.00 0.00 0.00 0.00
## [346]   NA 0.00 0.86 0.00 0.30 0.04 0.00 0.00 0.00 0.00 0.21 0.00 0.00 0.00 0.00
## [361] 0.00 0.00 0.00 0.00 0.00 0.14
```

Column type conversions

As you saw in the last exercise, "T" was used to denote a trace amount (i.e. too small to be accurately measured) of precipitation in the PrecipitationIn column. In order to coerce this column to numeric, you'll need to deal with this somehow. To keep things simple, we will just replace "T" with zero, as a string ("0").

15. Use str_replace() from stringr to make the proper replacements in the PrecipitationIn column of weather5. Run the call to *mutate_at* as-is to conveniently apply as.numeric() to all columns from CloudCover through WindDirDegrees (reading left to right in the data), saving the result to weather6. View the structure of *weather6* to confirm the coercions were successful.

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

# Convert characters to numerics
weather6 <- mutate_at(weather5, vars(CloudCover:WindDirDegrees), funs(as.numeric))
```

```
## Warning: `funs()` was deprecated in dplyr 0.8.0.
## Please use a list of either functions or lambdas:
##
##   # Simple named list:
##   list(mean = mean, median = median)
##
##   # Auto named with `tibble::lst()`:
##   tibble::lst(mean, median)
##
##   # Using lambdas
##   list(~ mean(., trim = .2), ~ median(., na.rm = TRUE))
```

```
# Look at result
str(weather6)
```

```
## 'data.frame':    366 obs. of  23 variables:
##  $ date                     : Date, format: "2014-12-01" "2014-12-10" ...
##  $ Events                   : chr  "Rain" "Rain" "Rain-Snow" "Snow" ...
##  $ CloudCover               : num  6 8 8 7 5 4 2 8 8 7 ...
##  $ Max.Dew.PointF           : num  46 45 37 28 28 29 33 42 46 34 ...
##  $ Max.Gust.SpeedMPH        : num  29 29 28 21 23 20 21 10 26 30 ...
##  $ Max.Humidity             : num  74 100 92 85 75 82 89 96 100 89 ...
##  $ Max.Sea.Level.PressureIn : num  30.4 29.6 29.8 29.9 29.9 ...
##  $ Max.TemperatureF         : num  64 48 39 39 42 45 42 44 49 44 ...
##  $ Max.VisibilityMiles      : num  10 10 10 10 10 10 10 10 10 10 ...
##  $ Max.Wind.SpeedMPH        : num  22 23 21 16 17 15 15 8 20 23 ...
##  $ Mean.Humidity            : num  63 95 87 75 65 68 75 85 85 73 ...
##  $ Mean.Sea.Level.PressureIn: num  30.1 29.5 29.6 29.9 29.8 ...
##  $ Mean.TemperatureF        : num  52 43 36 35 37 39 37 40 45 40 ...
##  $ Mean.VisibilityMiles     : num  10 3 7 10 10 10 10 9 6 10 ...
##  $ Mean.Wind.SpeedMPH       : num  13 13 13 11 12 10 6 4 11 14 ...
##  $ MeanDew.PointF           : num  40 39 31 27 26 27 29 36 41 30 ...
##  $ Min.DewpointF            : num  26 37 27 25 24 25 27 30 32 26 ...
##  $ Min.Humidity             : num  52 89 82 64 55 53 60 73 70 57 ...
##  $ Min.Sea.Level.PressureIn : num  30 29.4 29.4 29.8 29.8 ...
##  $ Min.TemperatureF         : num  39 38 32 31 32 33 32 35 41 36 ...
##  $ Min.VisibilityMiles      : num  10 1 1 7 10 10 10 5 1 10 ...
##  $ PrecipitationIn          : num  0.01 0.28 0.02 0 0 0 0 0 0.43 0.01 ...
##  $ WindDirDegrees           : num  268 357 230 286 298 306 324 79 311 281 ...
```

Find missing values

Before dealing with missing values in the data, it's important to find them and figure out why they exist in the first place. If your dataset is too big to look at all at once, like it is here, remember you can use sum() and is.na() to quickly size up the situation by counting the number of NA values.

The summary() function may also come in handy for identifying which variables contain the missing values. Finally, the which() function is useful for locating the missing values within a particular column.

16. Use sum() and is.na() to count the number of NA values in weather6. Look at a summary() of weather6 to figure out how the missings are distributed among the different variables. Use which() to identify the indices (i.e. row numbers) where Max.Gust.SpeedMPH is NA and save the result to ind (for indices). Use ind to look at the full rows of weather6 for which Max.Gust.SpeedMPH is missing.

```
# Count missing values
sum(is.na(weather6))
```

```
## [1] 6
```

```
# Find missing values
summary(weather6)
```

```
##       date                Events             CloudCover    Max.Dew.PointF
##  Min.   :2014-12-01   Length:366         Min.   :0.000   Min.   :-6.00
##  1st Qu.:2015-03-02   Class :character   1st Qu.:3.000   1st Qu.:32.00
##  Median :2015-06-01   Mode  :character   Median :5.000   Median :47.50
##  Mean   :2015-06-01                      Mean   :4.708   Mean   :45.48
##  3rd Qu.:2015-08-31                      3rd Qu.:7.000   3rd Qu.:61.00
##  Max.   :2015-12-01                      Max.   :8.000   Max.   :75.00
##
##  Max.Gust.SpeedMPH  Max.Humidity     Max.Sea.Level.PressureIn Max.TemperatureF
##  Min.   : 0.00     Min.   :  39.00   Min.   :29.58            Min.   :18.00
##  1st Qu.:21.00     1st Qu.:  73.25   1st Qu.:30.00            1st Qu.:42.00
##  Median :25.50     Median :  86.00   Median :30.14            Median :60.00
##  Mean   :26.99     Mean   :  85.69   Mean   :30.16            Mean   :58.93
##  3rd Qu.:31.25     3rd Qu.:  93.00   3rd Qu.:30.31            3rd Qu.:76.00
##  Max.   :94.00     Max.   :1000.00   Max.   :30.88            Max.   :96.00
##  NA's   :6
##  Max.VisibilityMiles Max.Wind.SpeedMPH Mean.Humidity
##  Min.   : 2.000      Min.   : 8.00     Min.   :28.00
##  1st Qu.:10.000      1st Qu.:16.00     1st Qu.:56.00
##  Median :10.000      Median :20.00     Median :66.00
##  Mean   : 9.907      Mean   :20.62     Mean   :66.02
##  3rd Qu.:10.000      3rd Qu.:24.00     3rd Qu.:76.75
##  Max.   :10.000      Max.   :38.00     Max.   :98.00
##
##  Mean.Sea.Level.PressureIn Mean.TemperatureF Mean.VisibilityMiles
##  Min.   :29.49             Min.   : 8.00     Min.   :-1.000
##  1st Qu.:29.87             1st Qu.:36.25     1st Qu.: 8.000
##  Median :30.03             Median :53.50     Median :10.000
##  Mean   :30.04             Mean   :51.40     Mean   : 8.861
##  3rd Qu.:30.19             3rd Qu.:68.00     3rd Qu.:10.000
##  Max.   :30.77             Max.   :84.00     Max.   :10.000
##
##  Mean.Wind.SpeedMPH MeanDew.PointF   Min.DewpointF     Min.Humidity
##  Min.   : 4.00      Min.   :-11.00   Min.   :-18.00   Min.   :16.00
##  1st Qu.: 8.00      1st Qu.: 24.00   1st Qu.: 16.25   1st Qu.:35.00
##  Median :10.00      Median : 41.00   Median : 35.00   Median :46.00
##  Mean   :10.68      Mean   : 38.96   Mean   : 32.25   Mean   :48.31
##  3rd Qu.:13.00      3rd Qu.: 56.00   3rd Qu.: 51.00   3rd Qu.:60.00
##  Max.   :22.00      Max.   : 71.00   Max.   : 68.00   Max.   :96.00
##
##  Min.Sea.Level.PressureIn Min.TemperatureF Min.VisibilityMiles PrecipitationIn
##  Min.   :29.16            Min.   :-3.00    Min.   : 0.000      Min.   :0.0000
##  1st Qu.:29.76            1st Qu.:30.00    1st Qu.: 2.000      1st Qu.:0.0000
```

```
##  Median :29.94           Median :46.00    Median :10.000      Median :0.0000
##  Mean   :29.93           Mean   :43.33    Mean   : 6.716      Mean   :0.1016
##  3rd Qu.:30.09           3rd Qu.:60.00    3rd Qu.:10.000      3rd Qu.:0.0400
##  Max.   :30.64           Max.   :74.00    Max.   :10.000      Max.   :2.9000
##
##  WindDirDegrees
##  Min.   :  1.0
##  1st Qu.:113.0
##  Median :222.0
##  Mean   :200.1
##  3rd Qu.:275.0
##  Max.   :360.0
##
```

```
# 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, ]
```

```
##           date Events CloudCover Max.Dew.PointF Max.Gust.SpeedMPH Max.Humidity
## 161 2015-05-18    Fog          6             52                NA          100
## 205 2015-06-03                 7             48                NA           93
## 273 2015-08-08                 4             61                NA           87
## 275 2015-09-01                 1             63                NA           78
## 308 2015-10-12                 0             56                NA           89
## 358 2015-11-03                 1             44                NA           82
##     Max.Sea.Level.PressureIn Max.TemperatureF Max.VisibilityMiles
## 161                    30.30              58                  10
## 205                    30.31              56                  10
## 273                    30.02              76                  10
## 275                    30.06              79                  10
## 308                    29.86              76                  10
## 358                    30.25              73                  10
##     Max.Wind.SpeedMPH Mean.Humidity Mean.Sea.Level.PressureIn Mean.TemperatureF
## 161                16            79                     30.23                54
## 205                14            82                     30.24                52
## 273                14            68                     29.99                69
## 275                15            65                     30.02                74
## 308                15            65                     29.80                64
## 358                16            57                     30.13                60
##     Mean.VisibilityMiles Mean.Wind.SpeedMPH MeanDew.PointF Min.DewpointF
## 161                    8                10             48            43
## 205                   10                 7             45            43
## 273                   10                 6             57            54
## 275                   10                 9             62            59
## 308                   10                 8             51            48
## 358                   10                 8             42            40
##     Min.Humidity Min.Sea.Level.PressureIn Min.TemperatureF Min.VisibilityMiles
## 161           57                    30.12              49                   0
## 205           71                    30.19              47                  10
## 273           49                    29.95              61                  10
## 275           52                    29.96              69                  10
## 308           41                    29.74              51                  10
## 358           31                    30.06              47                  10
```

```
##     PrecipitationIn WindDirDegrees
## 161               0             72
## 205               0             90
## 273               0             45
## 275               0             54
## 308               0            199
## 358               0            281
```

Finding an obvious error

Besides missing values, we want to know if there are values in the data that are too extreme or bizarre to be plausible. A great way to start the search for these values is with summary().

Once implausible values are identified, they must be dealt with in an intelligent and informed way. Sometimes the best way forward is obvious and other times it may require some research and/or discussions with the original collectors of the data.

17. View a summary() of weather6. Use which() to find the index of the erroneous element of weather6$Max.Humidity, saving the result to ind. Use ind to look at the full row of weather6 for that day. You discover an extra zero was accidentally added to this value. Correct it in the data.

```
# Review distributions for all variables
summary(weather6)
```

```
##       date                Events            CloudCover    Max.Dew.PointF
##  Min.   :2014-12-01   Length:366         Min.   :0.000   Min.   :-6.00
##  1st Qu.:2015-03-02   Class :character   1st Qu.:3.000   1st Qu.:32.00
##  Median :2015-06-01   Mode  :character   Median :5.000   Median :47.50
##  Mean   :2015-06-01                      Mean   :4.708   Mean   :45.48
##  3rd Qu.:2015-08-31                      3rd Qu.:7.000   3rd Qu.:61.00
##  Max.   :2015-12-01                      Max.   :8.000   Max.   :75.00
##
##  Max.Gust.SpeedMPH  Max.Humidity     Max.Sea.Level.PressureIn Max.TemperatureF
##  Min.   : 0.00     Min.   :  39.00   Min.   :29.58            Min.   :18.00
##  1st Qu.:21.00     1st Qu.:  73.25   1st Qu.:30.00            1st Qu.:42.00
##  Median :25.50     Median :  86.00   Median :30.14            Median :60.00
##  Mean   :26.99     Mean   :  85.69   Mean   :30.16            Mean   :58.93
##  3rd Qu.:31.25     3rd Qu.:  93.00   3rd Qu.:30.31            3rd Qu.:76.00
##  Max.   :94.00     Max.   :1000.00   Max.   :30.88            Max.   :96.00
##  NA's   :6
##  Max.VisibilityMiles Max.Wind.SpeedMPH Mean.Humidity
##  Min.   : 2.000      Min.   : 8.00     Min.   :28.00
##  1st Qu.:10.000      1st Qu.:16.00     1st Qu.:56.00
##  Median :10.000      Median :20.00     Median :66.00
##  Mean   : 9.907      Mean   :20.62     Mean   :66.02
##  3rd Qu.:10.000      3rd Qu.:24.00     3rd Qu.:76.75
##  Max.   :10.000      Max.   :38.00     Max.   :98.00
##
##  Mean.Sea.Level.PressureIn Mean.TemperatureF Mean.VisibilityMiles
##  Min.   :29.49             Min.   : 8.00     Min.   :-1.000
##  1st Qu.:29.87             1st Qu.:36.25     1st Qu.: 8.000
##  Median :30.03             Median :53.50     Median :10.000
##  Mean   :30.04             Mean   :51.40     Mean   : 8.861
##  3rd Qu.:30.19             3rd Qu.:68.00     3rd Qu.:10.000
##  Max.   :30.77             Max.   :84.00     Max.   :10.000
```

```
##
##  Mean.Wind.SpeedMPH MeanDew.PointF   Min.DewpointF     Min.Humidity
##  Min.   : 4.00      Min.   :-11.00   Min.   :-18.00   Min.   :16.00
##  1st Qu.: 8.00      1st Qu.: 24.00   1st Qu.: 16.25   1st Qu.:35.00
##  Median :10.00      Median : 41.00   Median : 35.00   Median :46.00
##  Mean   :10.68      Mean   : 38.96   Mean   : 32.25   Mean   :48.31
##  3rd Qu.:13.00      3rd Qu.: 56.00   3rd Qu.: 51.00   3rd Qu.:60.00
##  Max.   :22.00      Max.   : 71.00   Max.   : 68.00   Max.   :96.00
##
##  Min.Sea.Level.PressureIn Min.TemperatureF Min.VisibilityMiles PrecipitationIn
##  Min.   :29.16            Min.   :-3.00    Min.   : 0.000      Min.   :0.0000
##  1st Qu.:29.76            1st Qu.:30.00    1st Qu.: 2.000      1st Qu.:0.0000
##  Median :29.94            Median :46.00    Median :10.000      Median :0.0000
##  Mean   :29.93            Mean   :43.33    Mean   : 6.716      Mean   :0.1016
##  3rd Qu.:30.09            3rd Qu.:60.00    3rd Qu.:10.000      3rd Qu.:0.0400
##  Max.   :30.64            Max.   :74.00    Max.   :10.000      Max.   :2.9000
##
##  WindDirDegrees
##  Min.   :  1.0
##  1st Qu.:113.0
##  Median :222.0
##  Mean   :200.1
##  3rd Qu.:275.0
##  Max.   :360.0
##
```

```
# Find row with Max.Humidity of 1000
ind <- which(weather6$Max.Humidity == 1000)

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

```
##           date                Events CloudCover Max.Dew.PointF
## 135 2015-04-21 Fog-Rain-Thunderstorm          6             57
##     Max.Gust.SpeedMPH Max.Humidity Max.Sea.Level.PressureIn Max.TemperatureF
## 135                94         1000                    29.75               65
##     Max.VisibilityMiles Max.Wind.SpeedMPH Mean.Humidity
## 135                  10                20            71
##     Mean.Sea.Level.PressureIn Mean.TemperatureF Mean.VisibilityMiles
## 135                      29.6               56                    5
##     Mean.Wind.SpeedMPH MeanDew.PointF Min.DewpointF Min.Humidity
## 135                 10             49            36           42
##     Min.Sea.Level.PressureIn Min.TemperatureF Min.VisibilityMiles
## 135                    29.53              46                   0
##     PrecipitationIn WindDirDegrees
## 135            0.54            184
```

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

Another obvious error

You've discovered and repaired one obvious error in the data, but it appears that there's another. Sometimes you get lucky and can infer the correct or intended value from the other data. For example, if you know the

minimum and maximum values of a particular metric on a given day...

18. Use summary() to look at the value of only the Mean.VisibilityMiles variable of weather6. Determine the element of the value that is clearly erroneous in this column, saving the result to ind. Use ind to look at the full row of weather6 for this day. Inspect the values of other variables for this day to determine the correct value of Mean.VisibilityMiles, then make the appropriate fix.

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

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  -1.000   8.000  10.000   8.861  10.000  10.000
```

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

# Look at full row
weather6[ind,]
```

```
##           date Events CloudCover Max.Dew.PointF Max.Gust.SpeedMPH Max.Humidity
## 192 2015-06-18                 5             54                23           72
##     Max.Sea.Level.PressureIn Max.TemperatureF Max.VisibilityMiles
## 192                    30.14               76                  10
##     Max.Wind.SpeedMPH Mean.Humidity Mean.Sea.Level.PressureIn Mean.TemperatureF
## 192                17            59                     30.04                67
##     Mean.VisibilityMiles Mean.Wind.SpeedMPH MeanDew.PointF Min.DewpointF
## 192                   -1                 10             49            45
##     Min.Humidity Min.Sea.Level.PressureIn Min.TemperatureF Min.VisibilityMiles
## 192           46                    29.93               57                  10
##     PrecipitationIn WindDirDegrees
## 192               0            189
```

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

Check other extreme values

In addition to dealing with obvious errors in the data, we want to see if there are other extreme values. In addition to the trusty summary() function, hist() is useful for quickly getting a feel for how different variables are distributed.

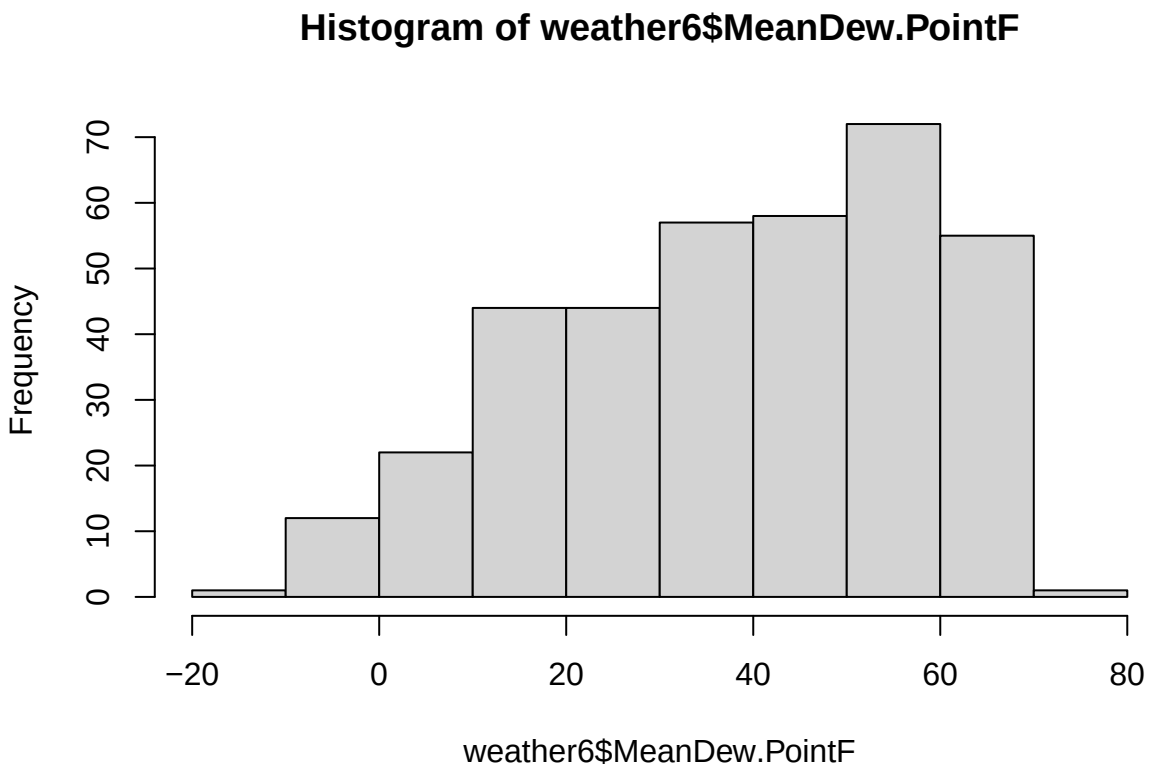
19. Check a summary() of weather6 one more time for extreme or unexpected values. View a histogram for MeanDew.PointF. Do the same for Min.TemperatureF. And once more for Mean.TemperatureF to compare distributions.

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

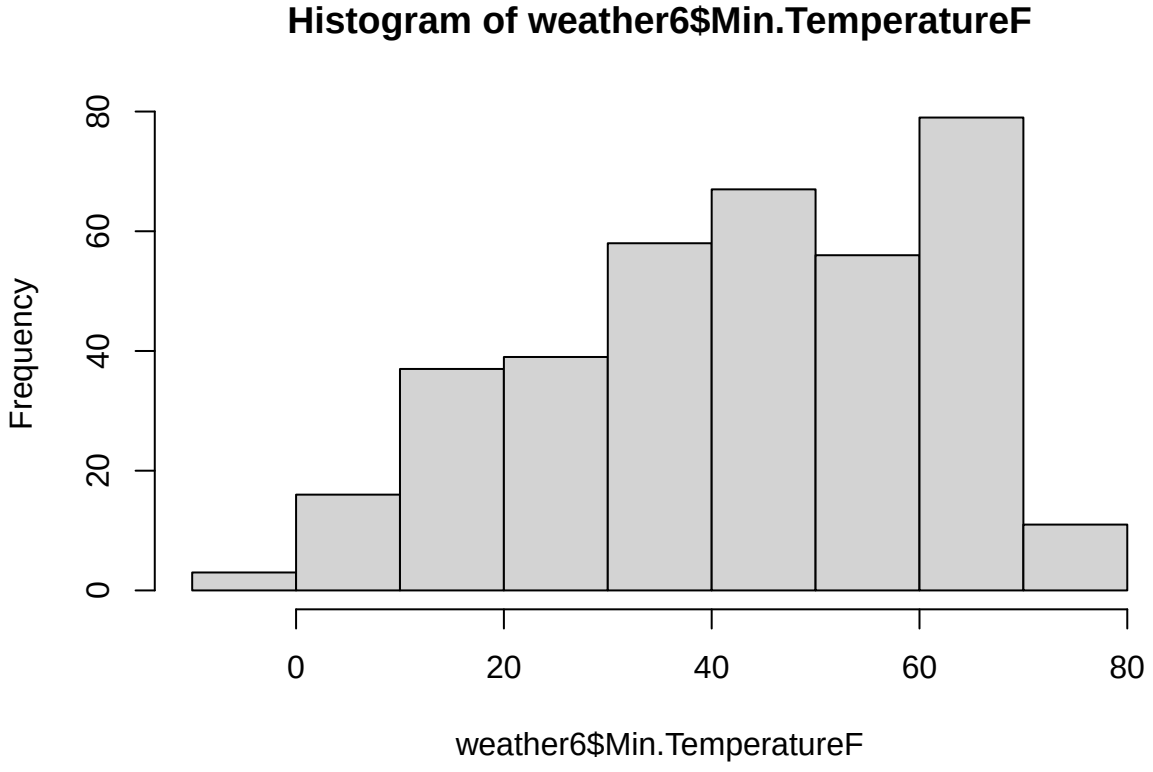
```
##       date                Events            CloudCover    Max.Dew.PointF
##  Min.   :2014-12-01   Length:366         Min.   :0.000   Min.   :-6.00
##  1st Qu.:2015-03-02   Class :character   1st Qu.:3.000   1st Qu.:32.00
##  Median :2015-06-01   Mode  :character   Median :5.000   Median :47.50
##  Mean   :2015-06-01                      Mean   :4.708   Mean   :45.48
##  3rd Qu.:2015-08-31                      3rd Qu.:7.000   3rd Qu.:61.00
##  Max.   :2015-12-01                      Max.   :8.000   Max.   :75.00
##
##  Max.Gust.SpeedMPH  Max.Humidity    Max.Sea.Level.PressureIn Max.TemperatureF
```

```
##  Min.   : 0.00     Min.   : 39.00   Min.   :29.58           Min.   :18.00
##  1st Qu.:21.00     1st Qu.: 73.25   1st Qu.:30.00           1st Qu.:42.00
##  Median :25.50     Median : 86.00   Median :30.14           Median :60.00
##  Mean   :26.99     Mean   : 83.23   Mean   :30.16           Mean   :58.93
##  3rd Qu.:31.25     3rd Qu.: 93.00   3rd Qu.:30.31           3rd Qu.:76.00
##  Max.   :94.00     Max.   :100.00   Max.   :30.88           Max.   :96.00
##  NA's   :6
##  Max.VisibilityMiles Max.Wind.SpeedMPH Mean.Humidity
##  Min.   : 2.000      Min.   : 8.00     Min.   :28.00
##  1st Qu.:10.000      1st Qu.:16.00     1st Qu.:56.00
##  Median :10.000      Median :20.00     Median :66.00
##  Mean   : 9.907      Mean   :20.62     Mean   :66.02
##  3rd Qu.:10.000      3rd Qu.:24.00     3rd Qu.:76.75
##  Max.   :10.000      Max.   :38.00     Max.   :98.00
##
##  Mean.Sea.Level.PressureIn Mean.TemperatureF Mean.VisibilityMiles
##  Min.   :29.49             Min.   : 8.00     Min.   : 1.000
##  1st Qu.:29.87             1st Qu.:36.25     1st Qu.: 8.000
##  Median :30.03             Median :53.50     Median :10.000
##  Mean   :30.04             Mean   :51.40     Mean   : 8.891
##  3rd Qu.:30.19             3rd Qu.:68.00     3rd Qu.:10.000
##  Max.   :30.77             Max.   :84.00     Max.   :10.000
##
##  Mean.Wind.SpeedMPH MeanDew.PointF   Min.DewpointF     Min.Humidity
##  Min.   : 4.00      Min.   :-11.00   Min.   :-18.00   Min.   :16.00
##  1st Qu.: 8.00      1st Qu.: 24.00   1st Qu.: 16.25   1st Qu.:35.00
##  Median :10.00      Median : 41.00   Median : 35.00   Median :46.00
##  Mean   :10.68      Mean   : 38.96   Mean   : 32.25   Mean   :48.31
##  3rd Qu.:13.00      3rd Qu.: 56.00   3rd Qu.: 51.00   3rd Qu.:60.00
##  Max.   :22.00      Max.   : 71.00   Max.   : 68.00   Max.   :96.00
##
##  Min.Sea.Level.PressureIn Min.TemperatureF Min.VisibilityMiles PrecipitationIn
##  Min.   :29.16            Min.   :-3.00    Min.   : 0.000      Min.   :0.0000
##  1st Qu.:29.76            1st Qu.:30.00    1st Qu.: 2.000      1st Qu.:0.0000
##  Median :29.94            Median :46.00    Median :10.000      Median :0.0000
##  Mean   :29.93            Mean   :43.33    Mean   : 6.716      Mean   :0.1016
##  3rd Qu.:30.09            3rd Qu.:60.00    3rd Qu.:10.000      3rd Qu.:0.0400
##  Max.   :30.64            Max.   :74.00    Max.   :10.000      Max.   :2.9000
##
##  WindDirDegrees
##  Min.   :  1.0
##  1st Qu.:113.0
##  Median :222.0
##  Mean   :200.1
##  3rd Qu.:275.0
##  Max.   :360.0
##
```

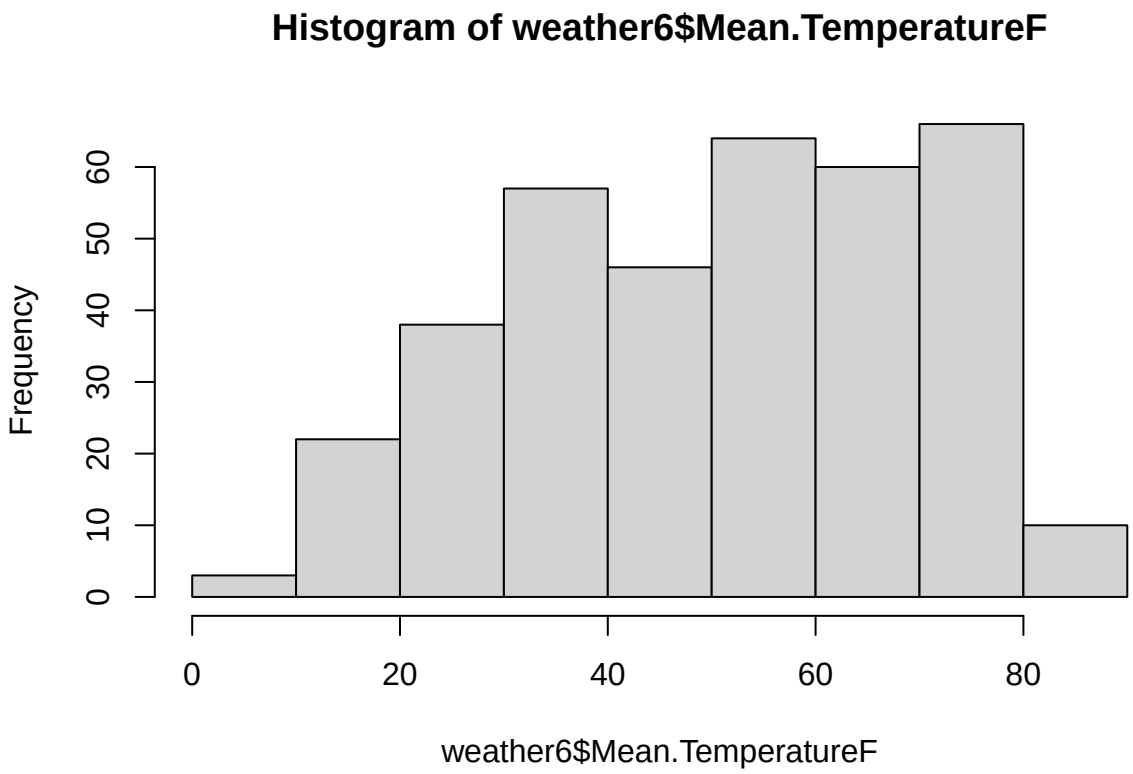
```
# Look at histogram for MeanDew.PointF
hist(weather6$MeanDew.PointF)
```




```
# Look at histogram for Min.TemperatureF
hist(weather6$Min.TemperatureF)
```




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



Finishing touches

Before officially calling our weather data clean, we want to put a couple of finishing touches on the data. These are a bit more subjective and may not be necessary for analysis, but they will make the data easier for others to interpret, which is generally a good thing.

There are a number of stylistic conventions in the R language. Depending on who you ask, these conventions may vary. Because the period (.) has special meaning in certain situations, we generally recommend using underscores (_) to separate words in variable names. We also prefer all lowercase letters so that no one has to remember which letters are uppercase or lowercase.

Finally, the events column (renamed to be all lowercase in the first instruction) contains an empty string ("") for any day on which there was no significant weather event such as rain, fog, a thunderstorm, etc. However, if it's the first time you're seeing these data, it may not be obvious that this is the case, so it's best for us to be explicit and replace the empty strings with something more meaningful.

20. Replace all empty strings in the events column of weather6 with "None". One last time, print out the first 6 rows of the weather6 data frame to see the changes.

```
# Clean up column names
new_colnames<-read.csv("new_colnames.csv")
new_colnames<-as.vector(new_colnames)
names(weather6) <- new_colnames
# Replace empty cells in events column
weather6$events[weather6$events == " "] <- "None"

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