Subset & aggregate time series precipitation data in R

Bonus / graduate activity. In this lesson, you will PLOT precipitation data in R. However, these data were collected over several decades and sometimes there are multiple data points per day. The data are also not cleaned. You will find heading names that may not be meaningful, and other issues with the data.

This lesson provides the basic skills that you need to create a plot of daily precipitation, for 30 years surrounding the 2013 flood. You will use the skills that you learned in the previous lessons, coupled with the skills in this lesson to process the data.

Learning Objectives

After completing this tutorial, you will be able to:

- Aggregate data by a day in R
- View names and rename columns in a dataframe.

Things You'll Need To Complete This Lesson

Please be sure you have the most current version of R and, preferably, `RStudio` to write your code.

R Skill Level: Intermediate - To succeed in this tutorial, you will need to have basic knowledge for use of the `R` software program.

R Libraries to Install:

- **ggplot2:** `install.packages("ggplot2")`
- **plotly:** `install.packages("dplyr")`

Data Download

If you haven't already downloaded this data (from the previous lesson), do so now.

Download Week 2 Data{:data-proofer-ignore=" .btn }

Work with Precipitation Data

R Libraries

To get started, load the `ggplot2` and `dplyr` libraries, setup your working directory and set `stringsAsFactors` to FALSE using `options()`.

Import Precipitation Data

We will use the `805333-precip-daily-1948-2013.csv` dataset for this assignment. in this analysis. This dataset contains the precipitation values collected daily from the COOP station 050843 in Boulder, CO for 1 January 2003 through 31 December 2013.

Import the data into R and then view the data structure.

```
##       STATION    STATION_NAME ELEVATION LATITUDE LONGITUDE           DATE
## 1 COOP:050843 BOULDER 2 CO US   unknown  unknown   unknown 19480801 01:00
## 2 COOP:050843 BOULDER 2 CO US   unknown  unknown   unknown 19480802 15:00
## 3 COOP:050843 BOULDER 2 CO US   unknown  unknown   unknown 19480803 09:00
## 4 COOP:050843 BOULDER 2 CO US   unknown  unknown   unknown 19480803 14:00
## 5 COOP:050843 BOULDER 2 CO US   unknown  unknown   unknown 19480803 15:00
## 6 COOP:050843 BOULDER 2 CO US   unknown  unknown   unknown 19480804 01:00
##   HPCP Measurement.Flag Quality.Flag
## 1 0.00                g
## 2 0.05
## 3 0.01
## 4 0.03
## 5 0.03
## 6 0.05
## 'data.frame':    14476 obs. of  9 variables:
##  $ STATION         : chr  "COOP:050843" "COOP:050843" "COOP:050843" "COOP:050843" ...
##  $ STATION_NAME    : chr  "BOULDER 2 CO US" "BOULDER 2 CO US" "BOULDER 2 CO US" "BOULDER 2 CO US" ..
##  $ ELEVATION       : chr  "unknown" "unknown" "unknown" "unknown" ...
##  $ LATITUDE        : chr  "unknown" "unknown" "unknown" "unknown" ...
##  $ LONGITUDE       : chr  "unknown" "unknown" "unknown" "unknown" ...
##  $ DATE            : chr  "19480801 01:00" "19480802 15:00" "19480803 09:00" "19480803 14:00" ...
##  $ HPCP            : num  0 0.05 0.01 0.03 0.03 0.05 0.02 0.01 0.01 0.01 ...
##  $ Measurement.Flag: chr  "g" " " " " " " " " ...
##  $ Quality.Flag    : chr  " " " " " " " " " " ...
```

About the Data

The structure of the data are similar to what you saw in previous lessons. HPCP is the total precipitation given in inches, recorded for the hour ending at the time specified by DATE. There is a designated missing data value of 999.99. Note that hours with no precipitation are not recorded.

The metadata for this file is located in your week2 directory: `PRECIP_HLY_documentation.pdf` file that can be downloaded along with the data. (Note, as of Sept. 2016, there is a mismatch in the data downloaded and the documentation. The differences are in the units and missing data value: inches/999.99 (standard) or millimeters/25399.75 (metric)).

NoData Values

Next, check out the data. Are there no data values? If so, make sure to adjust your data import code above to account for no data values. Then determine how many no data values you have in your dataset.

```
print("how many NA values are there?")
## [1] "how many NA values are there?"
sum(is.na(precip.boulder))
## [1] 401
```

Convert Date and Time

Compared to the previous lessons, notice that we now have date & time in our date field. To deal with both date and time, we use the `as.POSIXct()` method rather than as.date which we used previously. The syntax to convert to POSIXct is similar to what we used previously, but now, we will add the hour (H) and minute (M) to the format argument as follows:

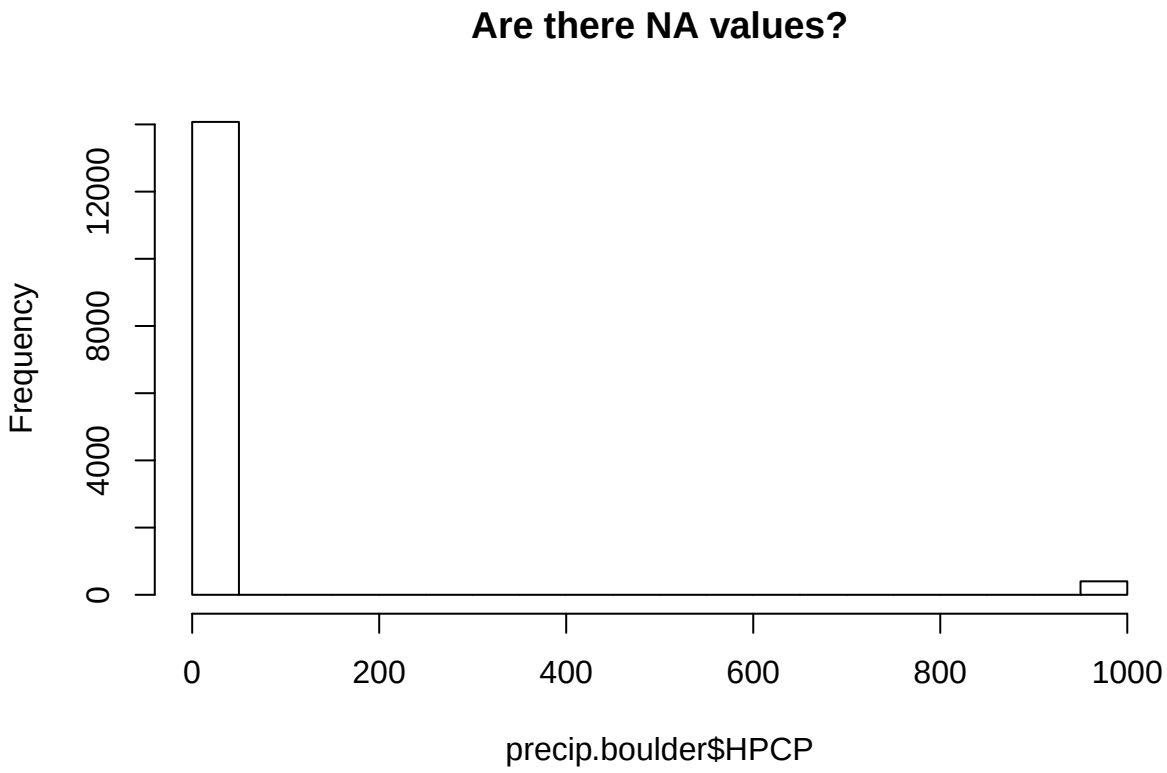

Figure 1: histogram of data

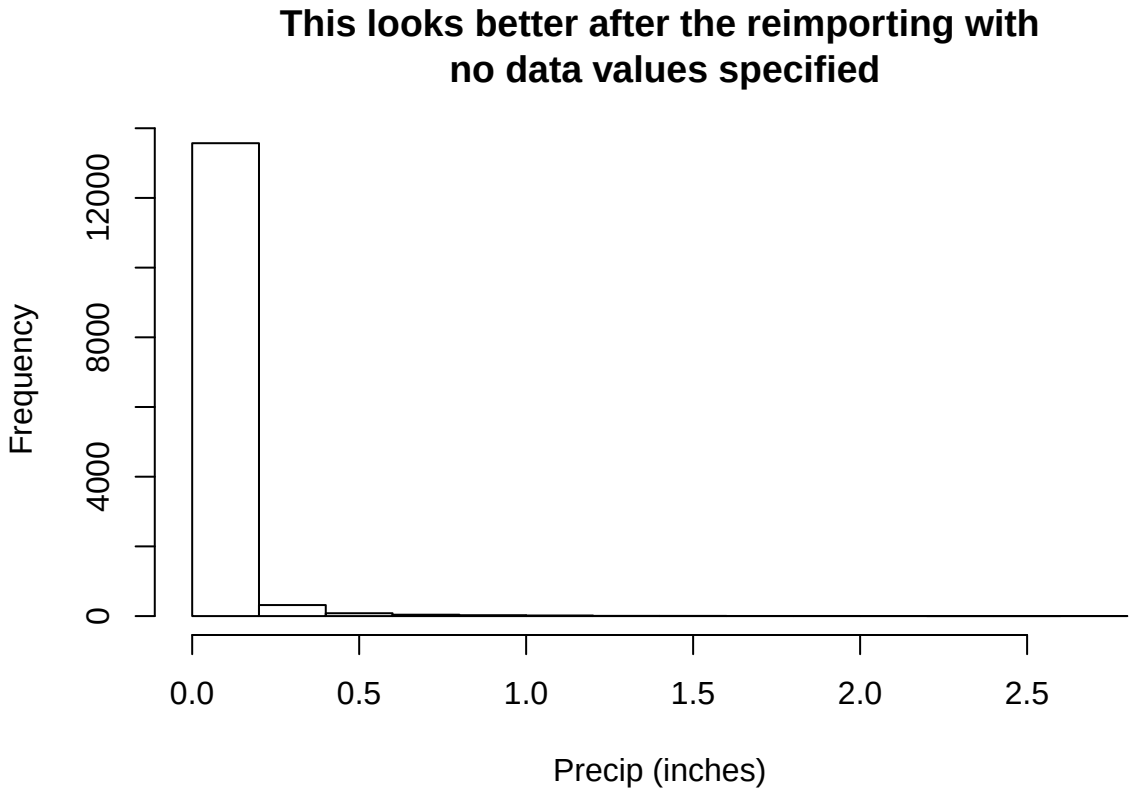

Figure 2: histogram of data

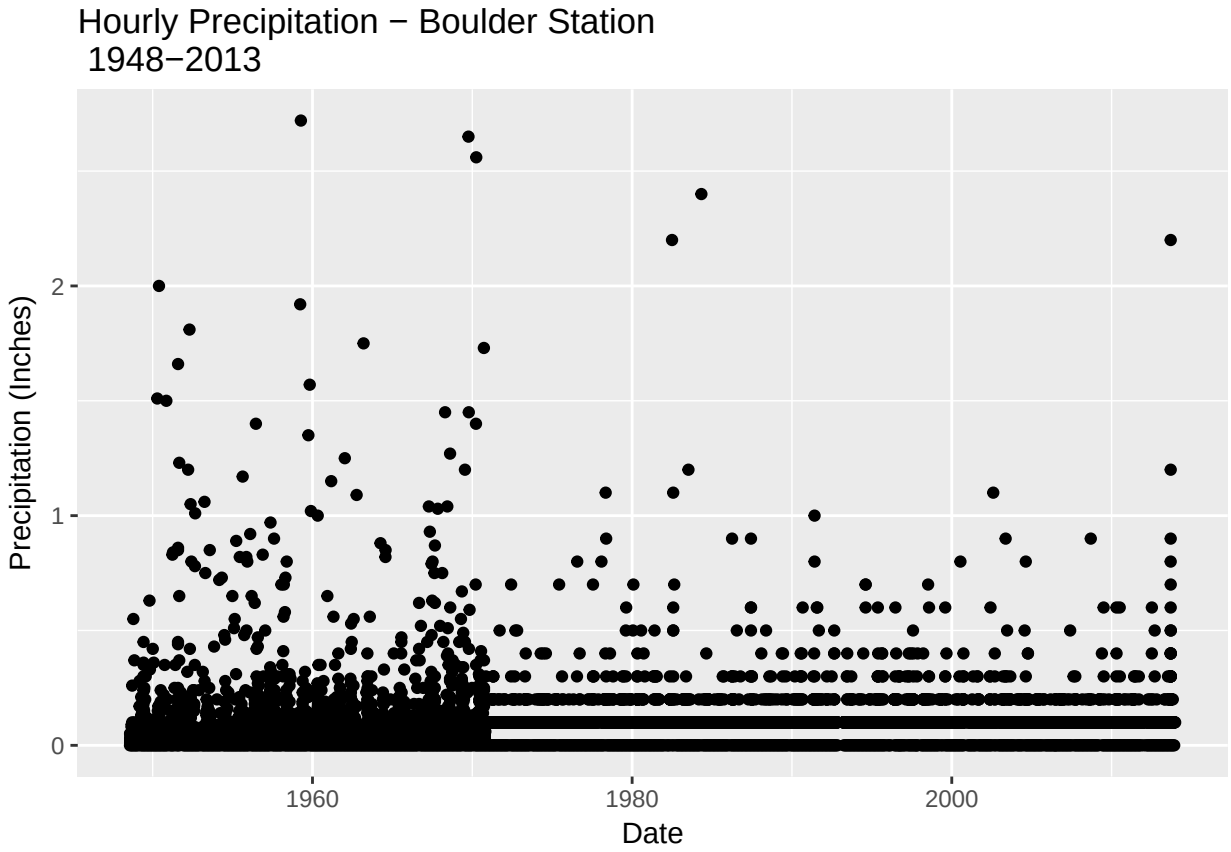

Figure 3: hourly precipitation

```
as.POSIXct(column-you-want-to-convert-here, format="%Y%m%d %H:%M")
```

- For more information on date/time classes, see the NEON tutorial *Dealing With Dates & Times in R - as.Date, POSIXct, POSIXlt.*

Plot Precipitation Data

Next, let's have a look at the data. Plot using `ggplot()`. Format the plot using the colors, labels, etc that are most clear and look the best. Your plot does not need to look like the one below!

```
## Warning: Removed 401 rows containing missing values (geom_point).
```

Differences in the data

Any ideas what might be causing the notable difference in the plotted data through time?

```
## Warning: Removed 401 rows containing missing values (geom_point).
```

It is difficult to interpret this plot which spans so many years at such a fine temporal scale. For our research project, we only need to explore 30 years of data. Let's do the following:

1. Aggregate the precipitation totals (sum) by day.
2. Subset the data for 30 years (we learned how to do this in a previous lesson).

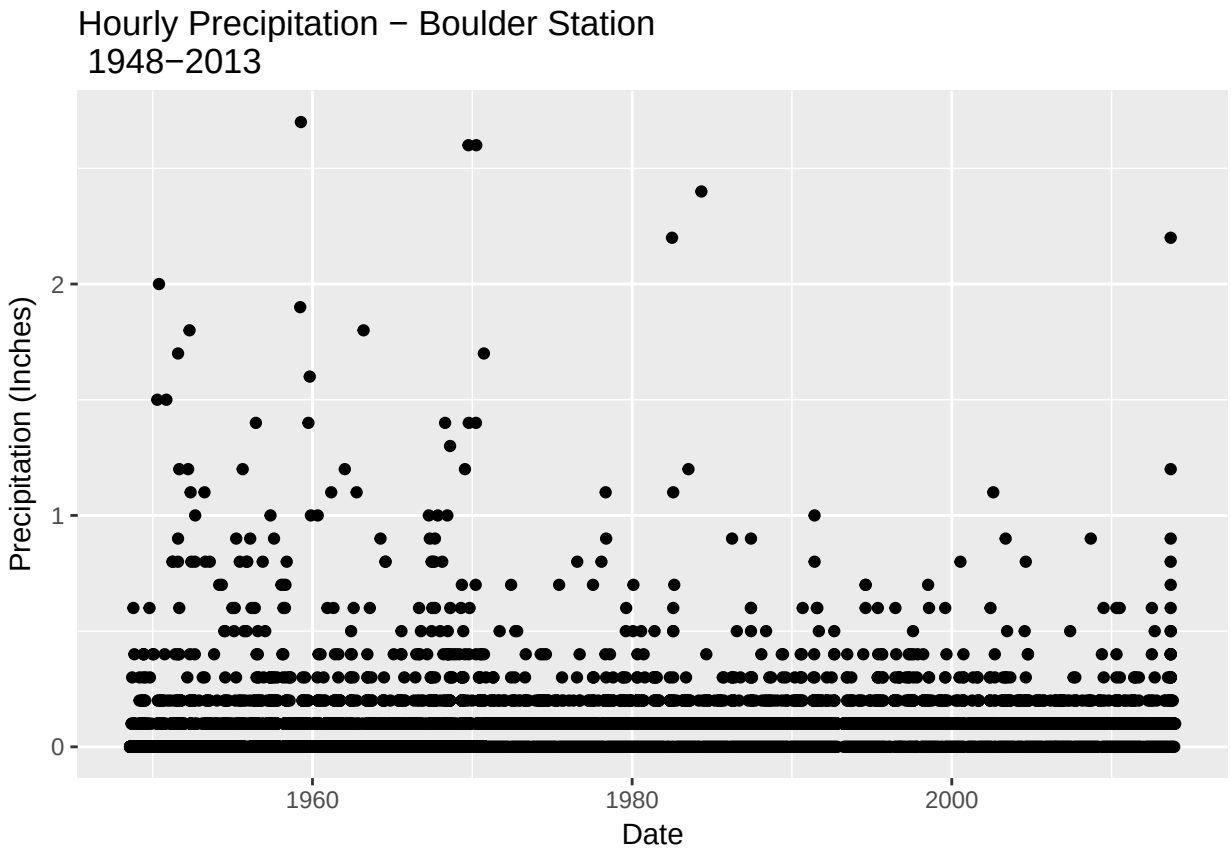

Figure 4: hourly precipitation

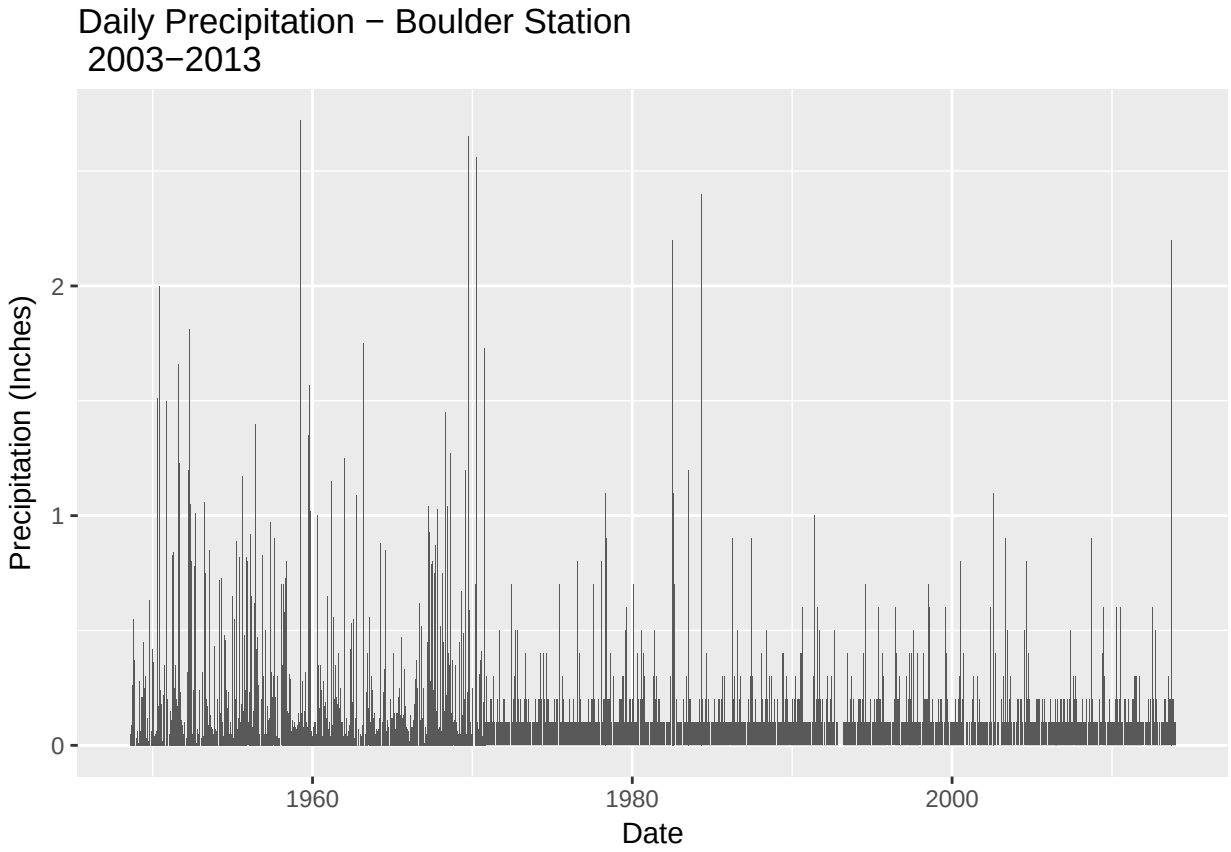

Figure 5: Daily precip plot

Aggregating and summarizing data

To aggregate data by a particular variable or time period, we can create a new column in our dataset called day. We will take all of the values for each day and add them using the `sum()` function. We can do all of this efficiently using dplyr mutate() function.

We use the `mutate()` function to add a new column called **day** to a new data.frame called **daily_sum_precip**. Note that we used `as.Date()` to just grab the dates rather than dates and times which are stored in the POSIX format.

```
# use dplyr
daily_sum_precip <- precip.boulder %>%
  mutate(day = as.Date(DATE, format="%Y-%m-%d"))    # create a new column called day w the date


# let's look at the new column
head(daily_sum_precip$day)
## [1] "1948-08-01" "1948-08-02" "1948-08-03" "1948-08-03" "1948-08-03"
## [6] "1948-08-04"
```

Next we `summarize()` the precipitation column (total_precip) - grouped by day. What this means is that we ADD UP all of the values for each day to get a grand total amount of precipitation each day.

```
# use dplyr
daily_sum_precip <- precip.boulder %>%
  mutate(day = as.Date(DATE, format="%Y-%m-%d")) %>%
```

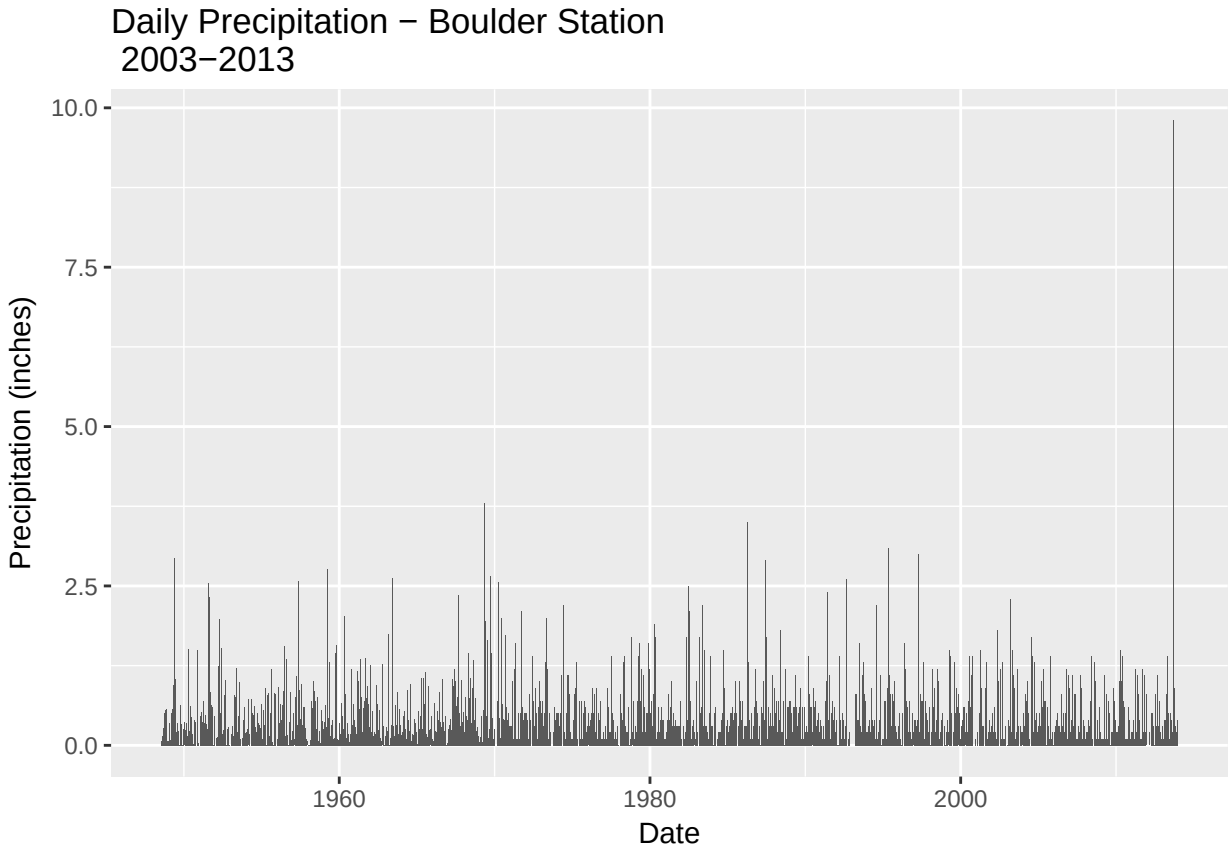

Figure 6: Daily precipitation for boulder

```
  group_by(day) %>% # group by the day column
  summarise(total_precip=sum(HPCP)) # calculate the SUM of all precipitation that occured on each day

# how large is the resulting data frame?
nrow(daily_sum_precip)
## [1] 4899

# view the results
head(daily_sum_precip)
## # A tibble: 6 × 2
##          day total_precip
##       <date>        <dbl>
## 1 1948-08-01         0.00
## 2 1948-08-02         0.05
## 3 1948-08-03         0.07
## 4 1948-08-04         0.14
## 5 1948-08-06         0.02
## 6 1948-08-14         0.03

# view column names
names(daily_sum_precip)
## [1] "day"          "total_precip"
```

Now plot the daily data.

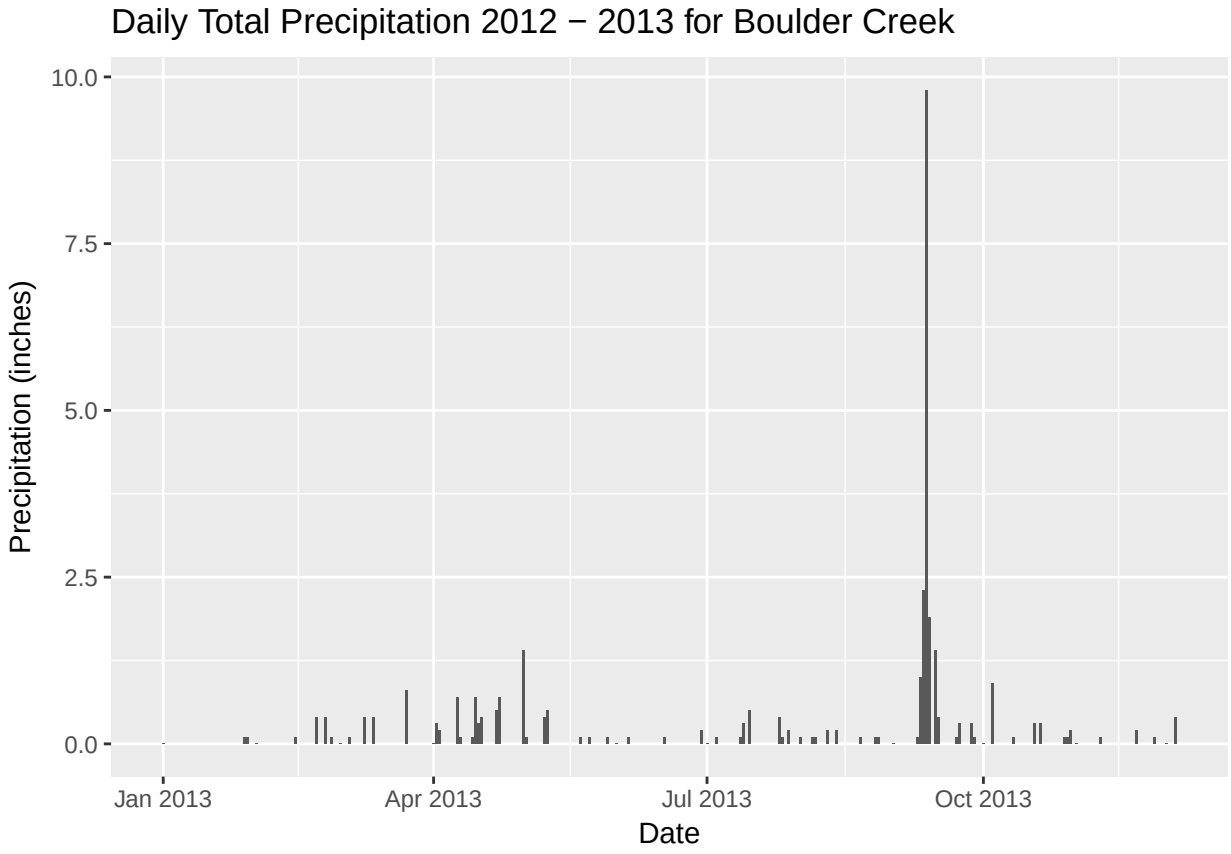

Figure 7: final precip plot daily sum

Finally, plot a temporal subsets of the data from 2000-2013. We learned how to do this in the previous lessons.

Now we can easily see the dramatic rainfall event in mid-September!

R Tip: If you are using a date-time class, instead of just a date class, you need to use `scale_x_datetime()`. {: .notice}

Subset The Data

If we wanted to, we could subset this data set using the same code that we used previously to subset! An example of the subsetted plot is below.

```
## Warning: Removed 4 rows containing missing values (position_stack).
```

Additional Resources

- How to subset data by weeks