# How to Wrangle Data

## using R with `tidyr` and `dplyr`

Ken Butler

March 30, 2015

# It is said that . . .

- 80% of data analysis: getting the data into the right form
- maybe 20% is making graphs, fitting models etc.
- thus, think about the 80%, “data wrangling”.

# Hadley Wickham

Inventor of:

- `tidyr`
- `dplyr`
- `ggplot2`
- `stringr`

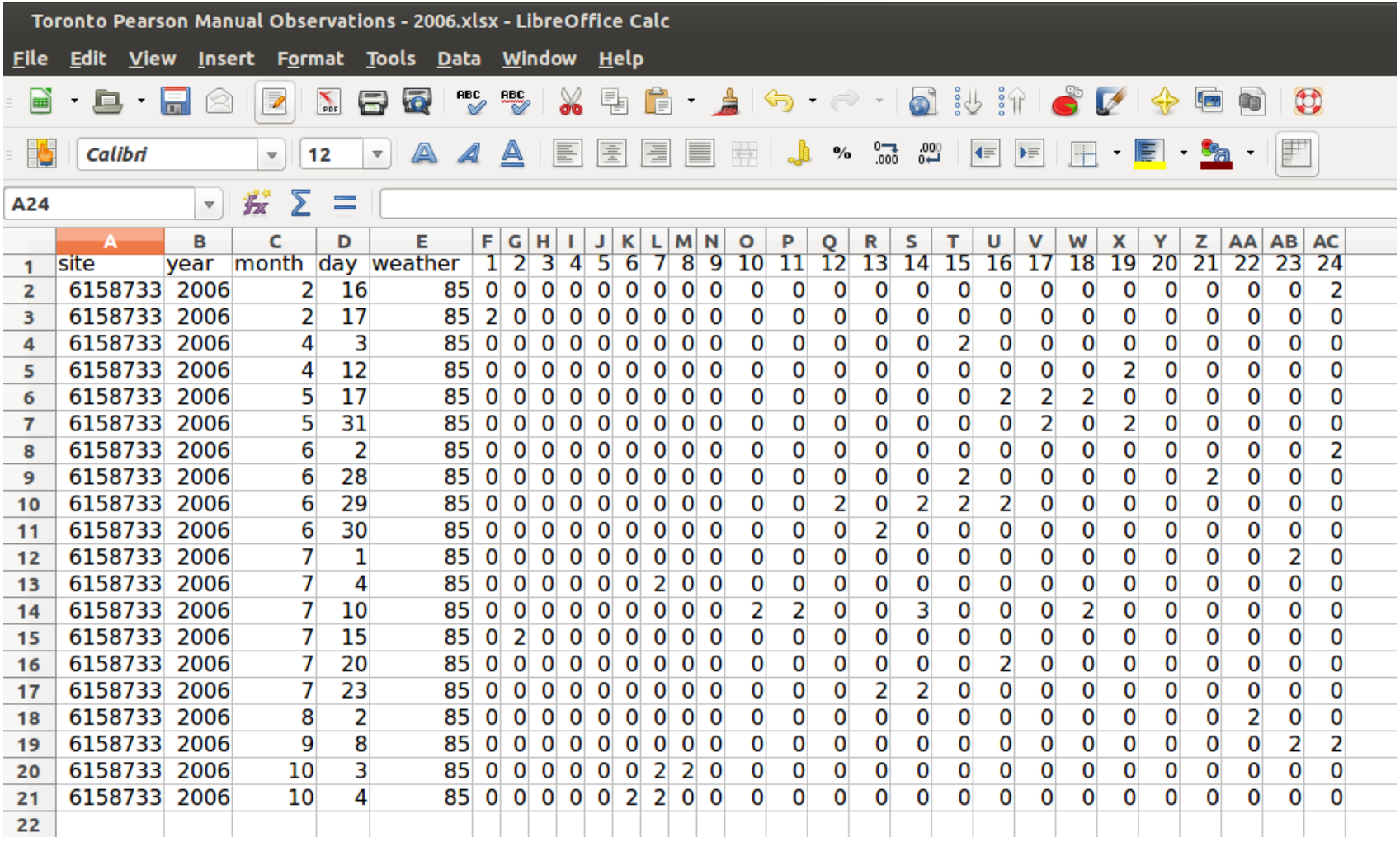
and other things besides.

# Tidy data (Wickham)

- Every value belongs to a *variable* and an *observation*.
- Variables in columns.
- Observations in rows.
- If this is done, data called “tidy”, ready for further analysis.
- If not, have “untidy” data, needs tidying.
- “Tidy” depends (somewhat) on kind of analysis you want to do.

# Gratuitous image

# Steven Huryn's thunderstorm data


Toronto Pearson Manual Observations - 2006.xlsx - LibreOffice Calc

File Edit View Insert Format Tools Data Window Help

Calibri 12

A24

| | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | AA | AB | AC |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | site | year | month | day | weather | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 |
| 2 | 6158733 | 2006 | 2 | 16 | 85 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 |
| 3 | 6158733 | 2006 | 2 | 17 | 85 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4 | 6158733 | 2006 | 4 | 3 | 85 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 5 | 6158733 | 2006 | 4 | 12 | 85 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 0 |
| 6 | 6158733 | 2006 | 5 | 17 | 85 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 2 | 2 | 0 | 0 | 0 | 0 | 0 | 0 |
| 7 | 6158733 | 2006 | 5 | 31 | 85 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 2 | 0 | 0 | 0 | 0 | 0 |
| 8 | 6158733 | 2006 | 6 | 2 | 85 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 |
| 9 | 6158733 | 2006 | 6 | 28 | 85 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 0 |
| 10 | 6158733 | 2006 | 6 | 29 | 85 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 2 | 2 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 11 | 6158733 | 2006 | 6 | 30 | 85 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 12 | 6158733 | 2006 | 7 | 1 | 85 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 0 |
| 13 | 6158733 | 2006 | 7 | 4 | 85 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 14 | 6158733 | 2006 | 7 | 10 | 85 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 2 | 0 | 0 | 3 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 0 |
| 15 | 6158733 | 2006 | 7 | 15 | 85 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 16 | 6158733 | 2006 | 7 | 20 | 85 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 17 | 6158733 | 2006 | 7 | 23 | 85 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 18 | 6158733 | 2006 | 8 | 2 | 85 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 0 |
| 19 | 6158733 | 2006 | 9 | 8 | 85 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 2 |
| 20 | 6158733 | 2006 | 10 | 3 | 85 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 21 | 6158733 | 2006 | 10 | 4 | 85 | 0 | 0 | 0 | 0 | 0 | 2 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 22 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

# The data

- Data from Pearson Airport, 2006
- Manual observer notes whether thunder or lightning observed within an hour. 2 is “thunderstorm”, 3 is “intense thunderstorm”.
- Spreadsheet contains only days where at least one thunderstorm hour was recorded.
- Columns labelled 1–24 indicate whether thunderstorm was observed in that hour (eg. “1” means midnight-1:00 am, “17” is 4:00–5:00 pm).
- Cols 1–24 all represent whether or not thunderstorm observed: not tidy!

# Reading in the data

- Save as `.csv`, then:

```
thunder=read.csv("thunder.csv",header=T)
thunder[1:6,1:9]
```

```
##       site year month day weather X1 X2 X3 X4
## 1 6158733 2006     2  16       85  0  0  0  0
## 2 6158733 2006     2  17       85  2  0  0  0
## 3 6158733 2006     4   3       85  0  0  0  0
## 4 6158733 2006     4  12       85  0  0  0  0
## 5 6158733 2006     5  17       85  0  0  0  0
## 6 6158733 2006     5  31       85  0  0  0  0
```

- Hour columns have gained an X.
- All those columns show whether/not thunderstorm recorded.
- Need to be combined together into one, with indication of which hour.

# gather

- Combine columns that all measure same thing.
- Input: data frame, what makes columns different, what makes them same, columns to combine:

```
library(tidyr)
thunder.2=gather(thunder,hour,is.thunder,
  X1:X24)
```

- `hour` and `is.thunder` are new variables; `X1` through `X24` disappear.

# Some of the result

```
thunder.2[c(1:6,33:39),c(2:4,6:7)]

##     year month day hour is.thunder
## 1   2006     2  16   X1           0
## 2   2006     2  17   X1           2
## 3   2006     4   3   X1           0
## 4   2006     4  12   X1           0
## 5   2006     5  17   X1           0
## 6   2006     5  31   X1           0
## 33 2006     7  10   X2           0
## 34 2006     7  15   X2           2
## 35 2006     7  20   X2           0
## 36 2006     7  23   X2           0
## 37 2006     8   2   X2           0
## 38 2006     9   8   X2           0
## 39 2006    10   3   X2           0
```

# Next

- Take out rows with no thunderstorm
- Remove columns `site` and `weather`
- Obtain actual hour from eg. `X6`
- Construct actual date and time of thunderstorm hour
- Make pretty picture of data.

Uses ideas from `tidyr` (extracting actual hour) and `dplyr` (the rest).

# Take out rows with no thunderstorm

- Use `filter` from `dplyr` with condition saying what rows you want to keep:

```
library(dplyr)
```

```
thunder.3=filter(thunder.2,is.thunder>0)
head(thunder.3)
```

```
##       site year month day weather hour is.thunder
## 1 6158733 2006     2  17      85   X1          2
## 2 6158733 2006     7  15      85   X2          2
## 3 6158733 2006    10   4      85   X6          2
## 4 6158733 2006     7   4      85   X7          2
## 5 6158733 2006    10   3      85   X7          2
## 6 6158733 2006    10   4      85   X7          2
```

# Doing things in sequence

- So far, we did

```
thunder.2=gather(thunder,hour,is.thunder,X1:X24)
thunder.3=filter(thunder.2,is.thunder>0)
```

This uses too many temporary variables. Compare with

```
thunder %>%
  gather(hour,is.thunder,X1:X24) %>%
  filter(is.thunder>0) %>%
  head()
```

```
##       site year month day weather hour is.thunder
## 1 6158733 2006     2  17       85   X1          2
## 2 6158733 2006     7  15       85   X2          2
## 3 6158733 2006    10   4       85   X6          2
## 4 6158733 2006     7   4       85   X7          2
## 5 6158733 2006    10   3       85   X7          2
## 6 6158733 2006    10   4       85   X7          2
```

# The chain operator %>%

- Read as “and then”.
- Allows layout of whole sequence of operations readably: “start with `thunder`, and then. . . ”.
- Output from one part of chain is input to next part, thus:
  - `thunder` is input to `gather`
  - output from `gather` (unnamed) input to `filter`
  - output from `filter` input to `head`
- Data frame input to all of these functions not specified: taken from output of previous part of chain.

# Getting rid of `site` and `weather`

- Uses `select` from `dplyr`. Add it to the chain:

```
thunder %>%
  gather(hour,is.thunder,X1:X24) %>%
  filter(is.thunder>0) %>%
  select(c(-site,-weather)) %>%
  head()
```

```
##   year month day hour is.thunder
## 1 2006     2  17   X1          2
## 2 2006     7  15   X2          2
## 3 2006    10   4   X6          2
## 4 2006     7   4   X7          2
## 5 2006    10   3   X7          2
## 6 2006    10   4   X7          2
```

Using column names in `select` selects just those. Putting minus sign before column *omits* column named.

# Splitting up `X` and hour number

- Uses `separate` from `tidyr`.
- Separate:
  - what to separate (`hour`),
  - what to call the separated bits (`junk` and `hour.num`)
  - and where to separate (after 1st character).
- Add to chain:

# The chain now

```
thunder %>%
  gather(hour,is.thunder,X1:X24) %>%
  filter(is.thunder>0) %>%
  select(c(-site,-weather)) %>%
  separate(hour,into=c("junk","hour.num"),
    sep=1) %>%
  head()

##   year month day junk hour.num is.thunder
## 1 2006     2  17    X        1          2
## 2 2006     7  15    X        2          2
## 3 2006    10   4    X        6          2
## 4 2006     7   4    X        7          2
## 5 2006    10   3    X        7          2
## 6 2006    10   4    X        7          2
```

# Saving results

- Can do the usual: `thunder.1=thunder %>% ...` to save in variable `thunder.1`.
- Or:

```
thunder %>%
  gather(hour,is.thunder,X1:X24) %>%
  filter(is.thunder>0) %>%
  select(c(-site,-weather)) %>%
  separate(hour,into=c("junk","hour.num"),
    sep=1) -> thunder.1
```

"...and then save in `thunder.1`".

# Debugging a chain

- *One line at a time*.
- Do each step of the chain, and convince yourself that you have the right thing.
- Can use `head()` as the last step to show the first few lines of the result, eg.

```
thunder %>%
  gather(hour,is.thunder,X1:X24) %>%
  filter(is.thunder>0) %>%
  head()

##      site year month day weather hour is.thunder
## 1 6158733 2006     2  17      85   X1          2
## 2 6158733 2006     7  15      85   X2          2
## 3 6158733 2006    10   4      85   X6          2
## 4 6158733 2006     7   4      85   X7          2
## 5 6158733 2006    10   3      85   X7          2
## 6 6158733 2006    10   4      85   X7          2
```

- Yes, we only have lines where there actually is thunder.

# Next steps

- Turn year, month, day, hour into a date and time
- throw away `junk`
- maybe throw away `is.thunder`

# Dates and times

- R has class `POSIXct`: represents date-times as seconds since Jan 1 1970:

```
d=as.POSIXct("2015-03-30 15:00")
d

## [1] "2015-03-30 15:00:00 EDT"

print.default(d)

## [1] 1.428e+09
## attr(,"class")
## [1] "POSIXct" "POSIXt"
## attr(,"tzone")
## [1] ""
```

# Dates and times (2)

- Also class `POSIXlt`: represents as list, easier for extracting month, day etc:

```
d=as.POSIXlt("2015-03-30 15:00")
d

## [1] "2015-03-30 15:00:00 EDT"

d$mday

## [1] 30

d$hour

## [1] 15
```

# Time zones

- These classes handle time zones: unless you specify one, uses local time zone for your computer.
- Also handles daylight savings:

```
v=c("2015-03-30 15:00", "2015-03-20 15:00",
    "2015-03-01 15:00")
as.POSIXct(v)

## [1] "2015-03-30 15:00:00 EDT"
## [2] "2015-03-20 15:00:00 EDT"
## [3] "2015-03-01 15:00:00 EST"

as.POSIXct(v,tz="CET")

## [1] "2015-03-30 15:00:00 CEST"
## [2] "2015-03-20 15:00:00 CET"
## [3] "2015-03-01 15:00:00 CET"

c(as.POSIXct(v,tz="CET"))

## [1] "2015-03-30 09:00:00 EDT"
## [2] "2015-03-20 10:00:00 EDT"
## [3] "2015-03-01 09:00:00 EST"
```

- Thunderstorm data: date-time-as-number (`POSIXct`) fine.

# Construct hours

- 1,2,. . . ,24 are 12:00-1:00am, . . . , 11:00pm-midnight.
- To align date and time, use *starting* hour (subtract 1 from hour).
- Paste on ":00:00":

```
tt=thunder.1$hour.num
tt=paste0(as.numeric(tt)-1,":00:00")
head(tt)

## [1] "0:00:00" "1:00:00" "5:00:00"
## [4] "6:00:00" "6:00:00" "6:00:00"
```

- Paste year-month-day together separated by `-`.
- Paste the two results together, and pass into `as.POSIXct`.
- In a chain: `mutate` creates new variables from old.

# Making date-times

```
thunder.1 %>%
  mutate(tt=paste0(as.numeric(hour.num)-1,":00:00")) %>%
  mutate(dd=paste(year,month,day,sep="-")) %>%
  select(c(tt,dd)) %>%
  head()

##         tt         dd
## 1 0:00:00 2006-2-17
## 2 1:00:00 2006-7-15
## 3 5:00:00 2006-10-4
## 4 6:00:00  2006-7-4
## 5 6:00:00 2006-10-3
## 6 6:00:00 2006-10-4
```

That works. (Compare with data source, reading down columns.)

# Paste together into one

```
thunder.1 %>%
  mutate(tt=paste0(as.numeric(hour.num)-1,":00:00")
  mutate(dd=paste(year,month,day,sep="-")) %>%
  mutate(dt=as.POSIXct(paste(dd,tt))) %>%
  select(dt,is.thunder) %>%
  head()

##                    dt is.thunder
## 1 2006-02-17 00:00:00          2
## 2 2006-07-15 01:00:00          2
## 3 2006-10-04 05:00:00          2
## 4 2006-07-04 06:00:00          2
## 5 2006-10-03 06:00:00          2
## 6 2006-10-04 06:00:00          2
```

Yep. Save it.

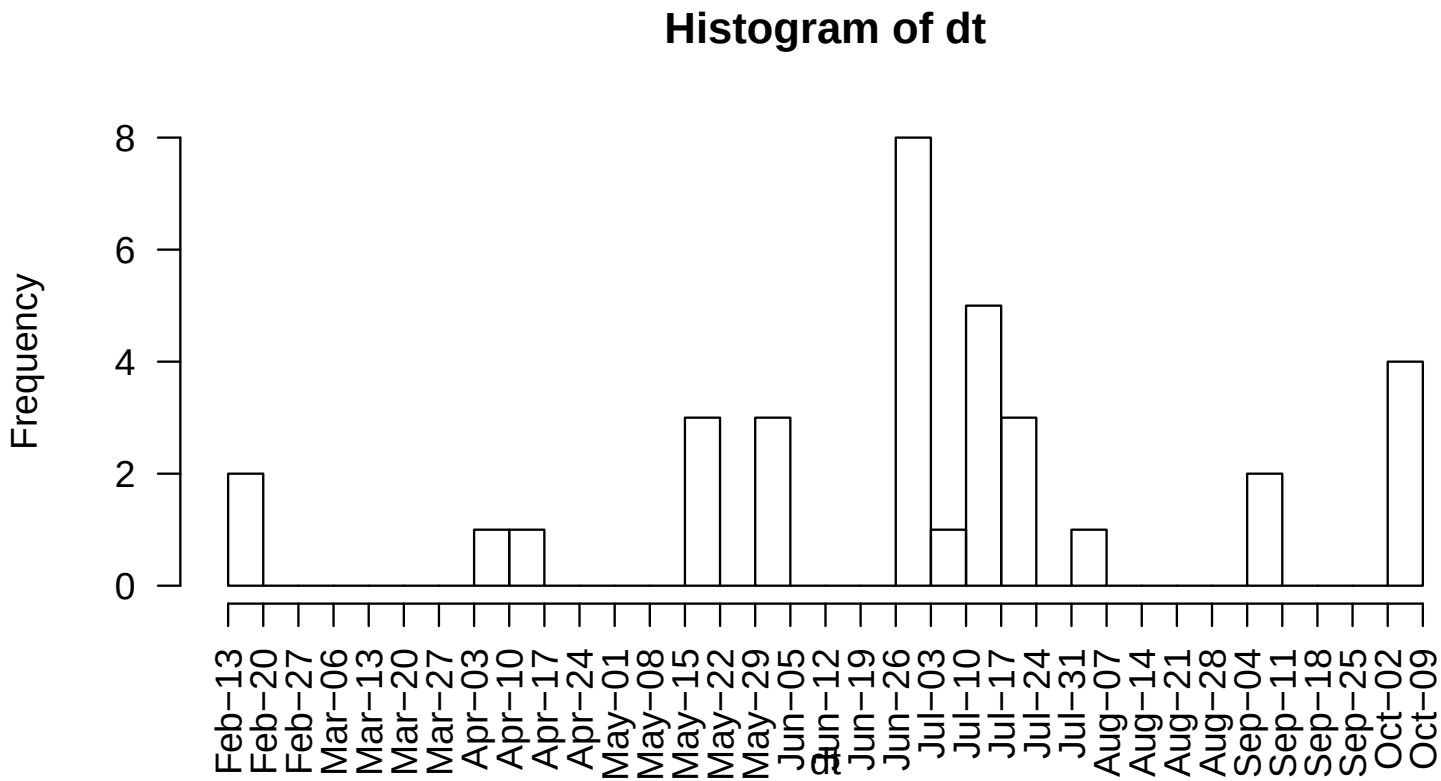
# Saving into `thunder.2`

```
thunder.1 %>%
  mutate(tt=paste0(as.numeric(hour.num)-1,":00:00")
  mutate(dd=paste(year,month,day,sep="-")) %>%
  mutate(dt=as.POSIXct(paste(dd,tt))) %>%
  select(dt,is.thunder) -> thunder.2
```

# What time of year do thunderstorms happen?

- Make a histogram of the dates
- They are actually numbers, so this works (with fiddling).

```
attach(thunder.2)
hist(dt,breaks="weeks",freq=T,
   format="%b-%d",las=2)
```



# Jun 26 – July 3: were there really 8 thunderstorm hours?

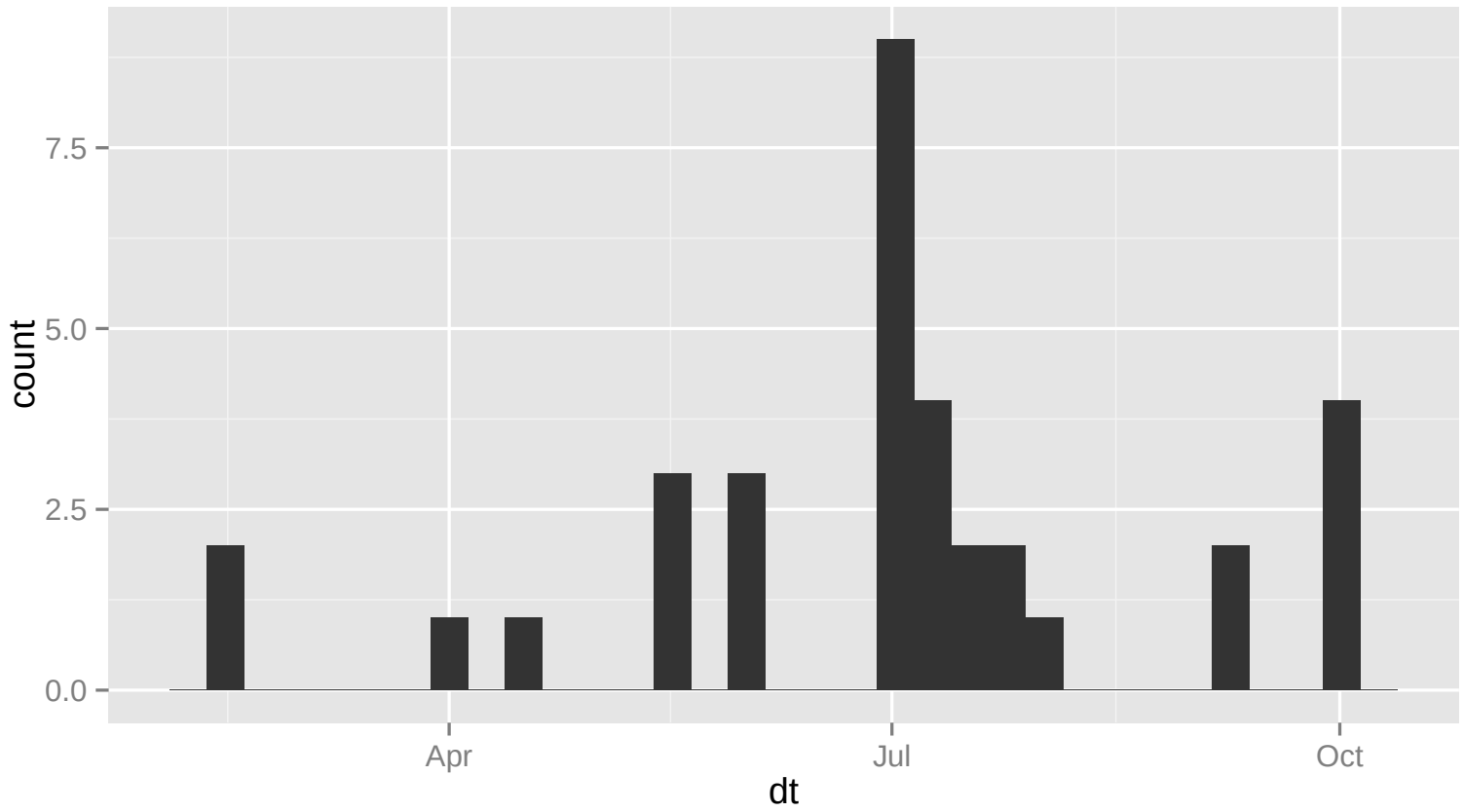
```
d=c("2006-06-25","2006-07-04")
dates=as.POSIXct(d)
thunder.2 %>% filter(dt>dates[1],dt<dates[2])

##                    dt is.thunder
## 1 2006-06-29 11:00:00           2
## 2 2006-06-30 12:00:00           2
## 3 2006-06-29 13:00:00           2
## 4 2006-06-28 14:00:00           2
## 5 2006-06-29 14:00:00           2
## 6 2006-06-29 15:00:00           2
## 7 2006-06-28 20:00:00           2
## 8 2006-07-01 22:00:00           2
```

# ggplot and chains

- The ggplot graphing mechanism fits nicely into a chain:

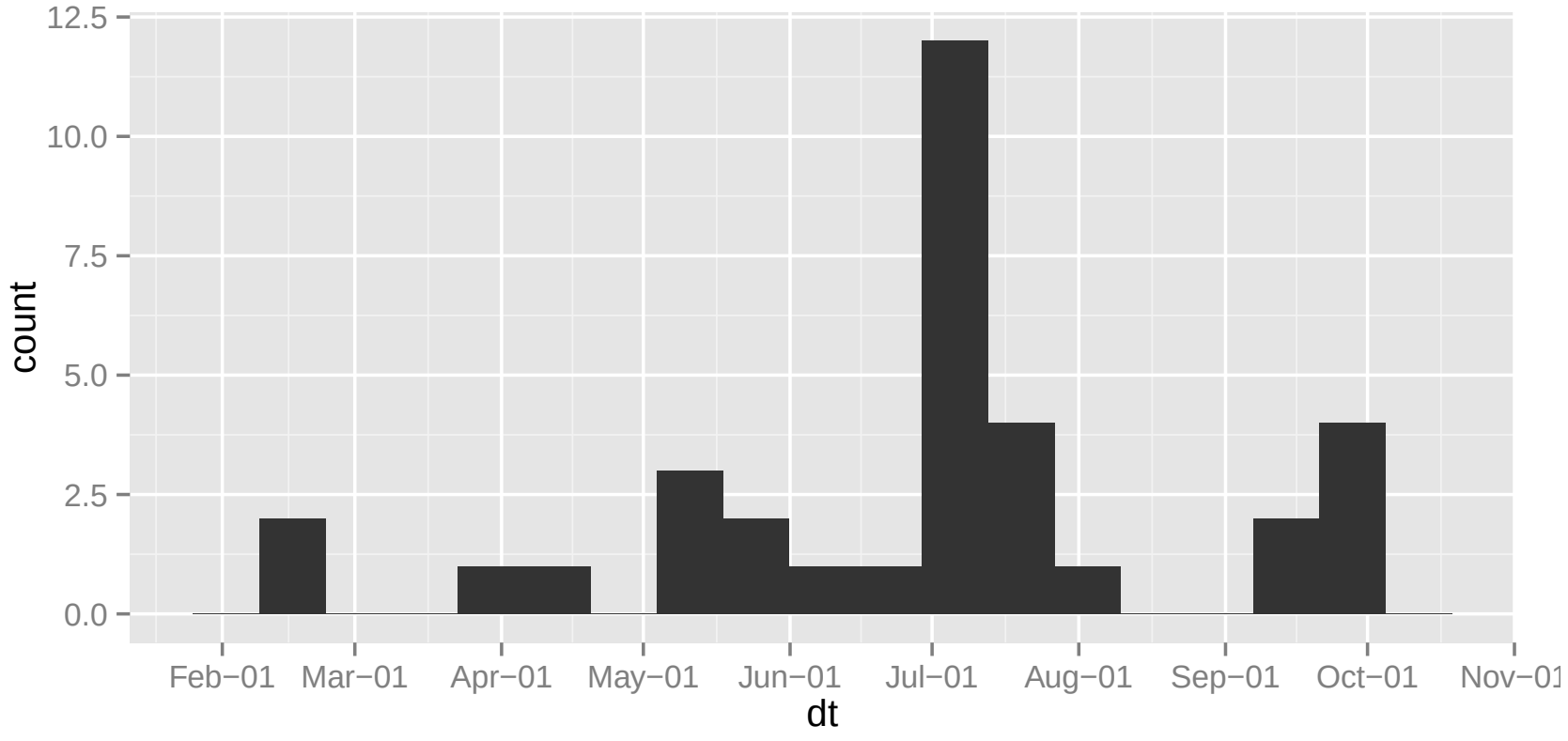
```
library(ggplot2)
```

```
thunder.2 %>% ggplot(aes(dt))+geom_histogram()
```



# Or, tarted up

```
library(scales)
thunder.2 %>% ggplot(aes(dt))+
  geom_histogram(binwidth=3600*24*7*2)+
  scale_x_datetime(breaks=date_breaks("1 month"),
                   labels=date_format("%b-%d"))
```



# Some different climate data

- Daily weather record for one year for a secret location:

```
weather=read.csv("weather_2014.csv",header=T)
names(weather)
```

```
##  [1] "day.count"       "day"
##  [3] "month"           "season"
##  [5] "l.temp"          "h.temp"
##  [7] "ave.temp"        "l.temp.time"
##  [9] "h.temp.time"     "rain"
## [11] "ave.wind"        "gust.wind"
## [13] "gust.wind.time"  "dir.wind"
```

# Plot daily high and low against date

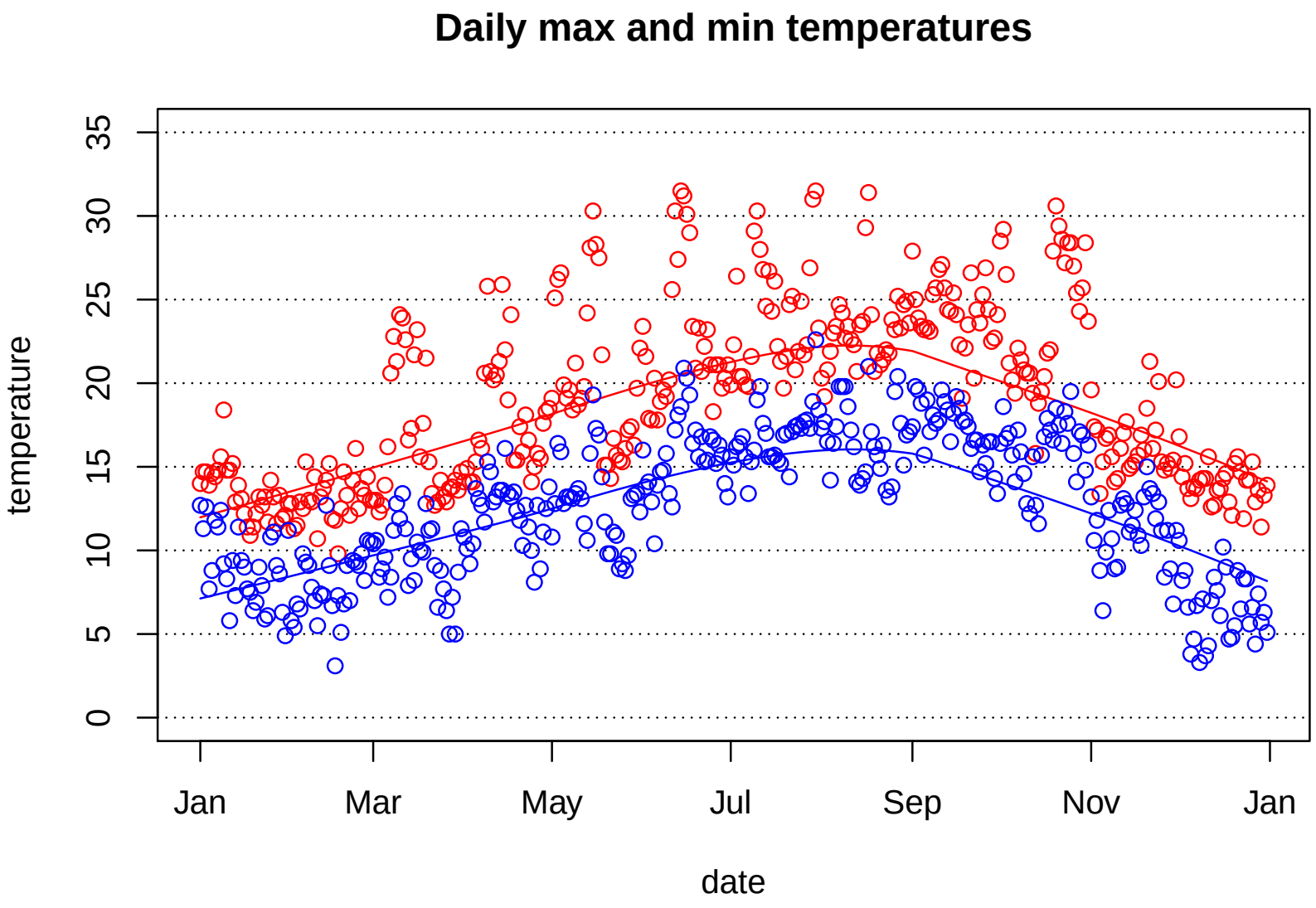
- First need date as date (easier than for time):

```
weather %>%
  mutate(date1=paste("2014",month,day,
    sep="-")) %>%
  mutate(date=as.Date(date1)) -> weather.2
```

- Make empty graph, add points and lowess curves in colour:

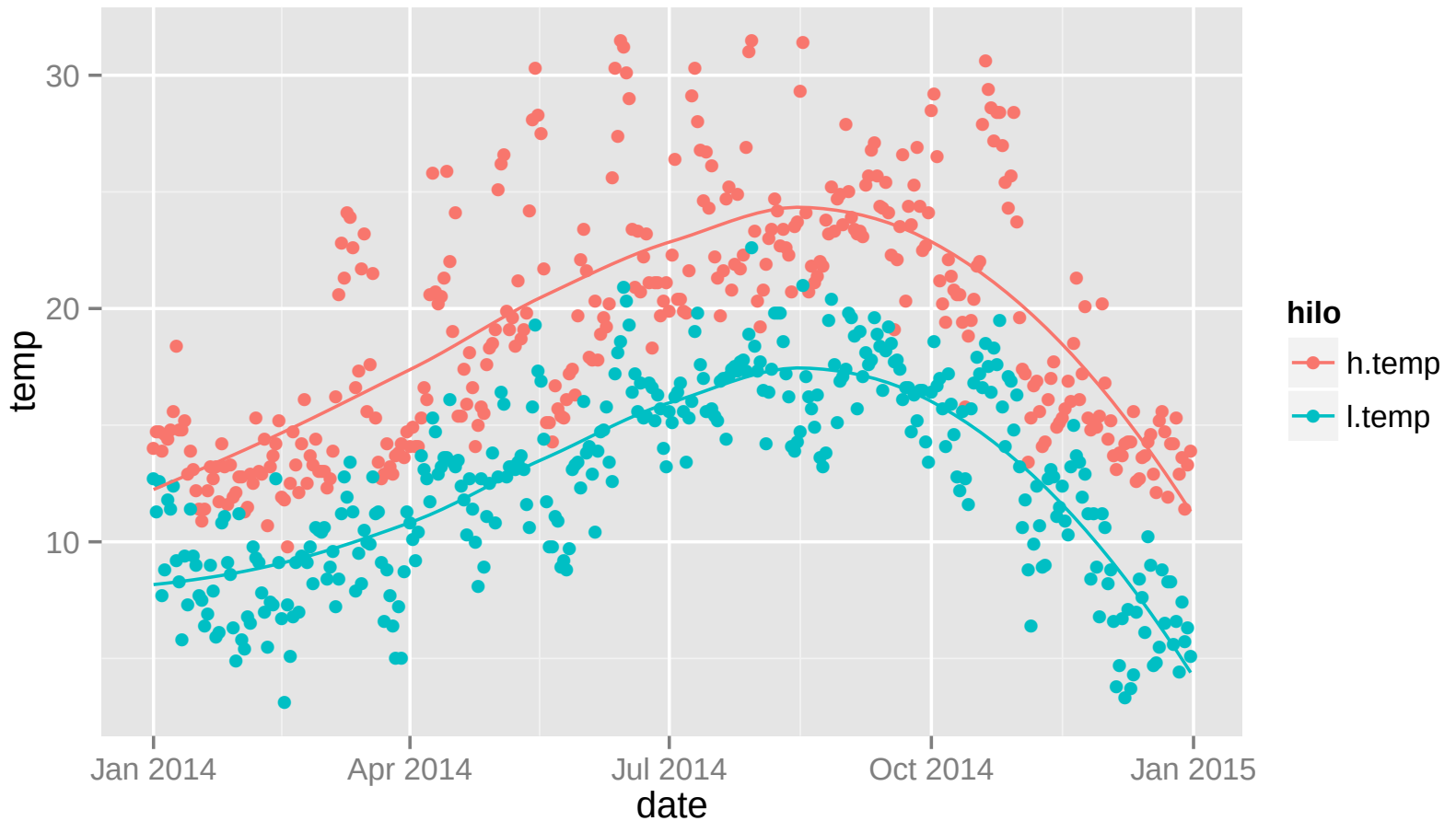
```
attach(weather.2)
plot(date,h.temp,type="n",ylim=c(0,35),
  ylab="temperature",
  main="Daily max and min temperatures")
grid(NA,NULL,col="black")
points(date,h.temp,col="red")
lines(lowess(date,h.temp),col="red")
points(date,l.temp,col="blue")
lines(lowess(date,l.temp),col="blue")
detach(weather.2)
```

# The plot


Daily max and min temperatures
temperature
0
5
10
15
20
25
30
35
Jan
Mar
May
Jul
Sep
Nov
Jan
date

# Same with ggplot

```
weather.2 %>%
  gather(hilo,temp,h.temp:l.temp) %>%
  ggplot(aes(date,temp,group=hilo,colour=hilo))+
    geom_point()+geom_smooth(se=F)
```



# Time of day of max and min temperature

- `gather` the max and min times, keeping track of whether max or min.
- Pull out the hour of the time of max or min, save:

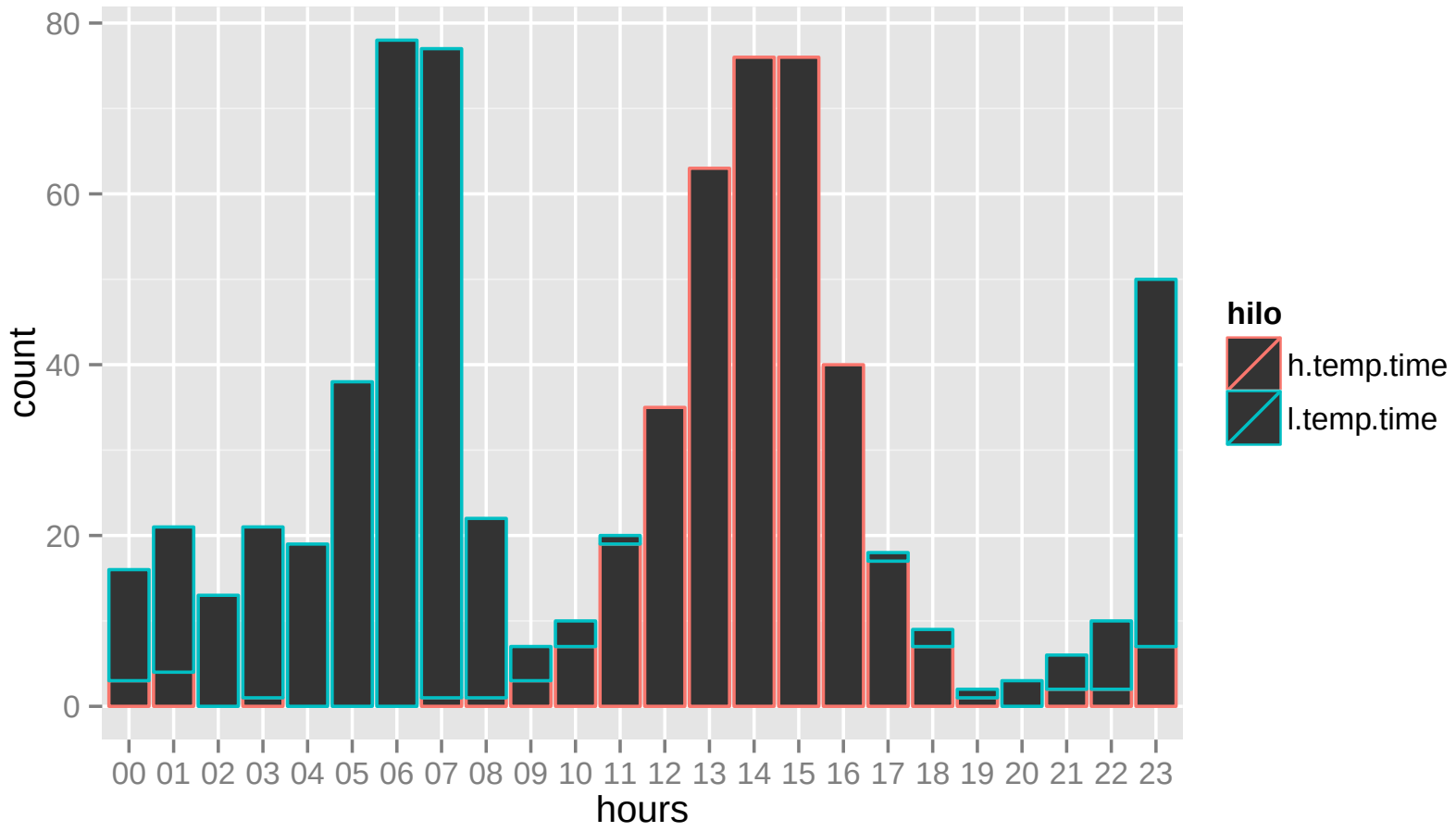
```
library(stringr)
weather %>%
gather(hilo,times,h.temp.time:l.temp.time) %>%
mutate(hours=str_extract(times,"[0-9]+")) %>%
select(hilo,hours) -> weather.3
```

# weather.3

```
weather.3 %>% head(10)

##              hilo hours
## 1  h.temp.time    23
## 2  h.temp.time    11
## 3  h.temp.time    14
## 4  h.temp.time    01
## 5  h.temp.time    12
## 6  h.temp.time    00
## 7  h.temp.time    14
## 8  h.temp.time    13
## 9  h.temp.time    14
## 10 h.temp.time    12
```

# Plot, grouping/colouring by hi/lo

```
weather.3 %>%
  ggplot(aes(hours,group=hilo,colour=hilo))+
  geom_histogram()
```



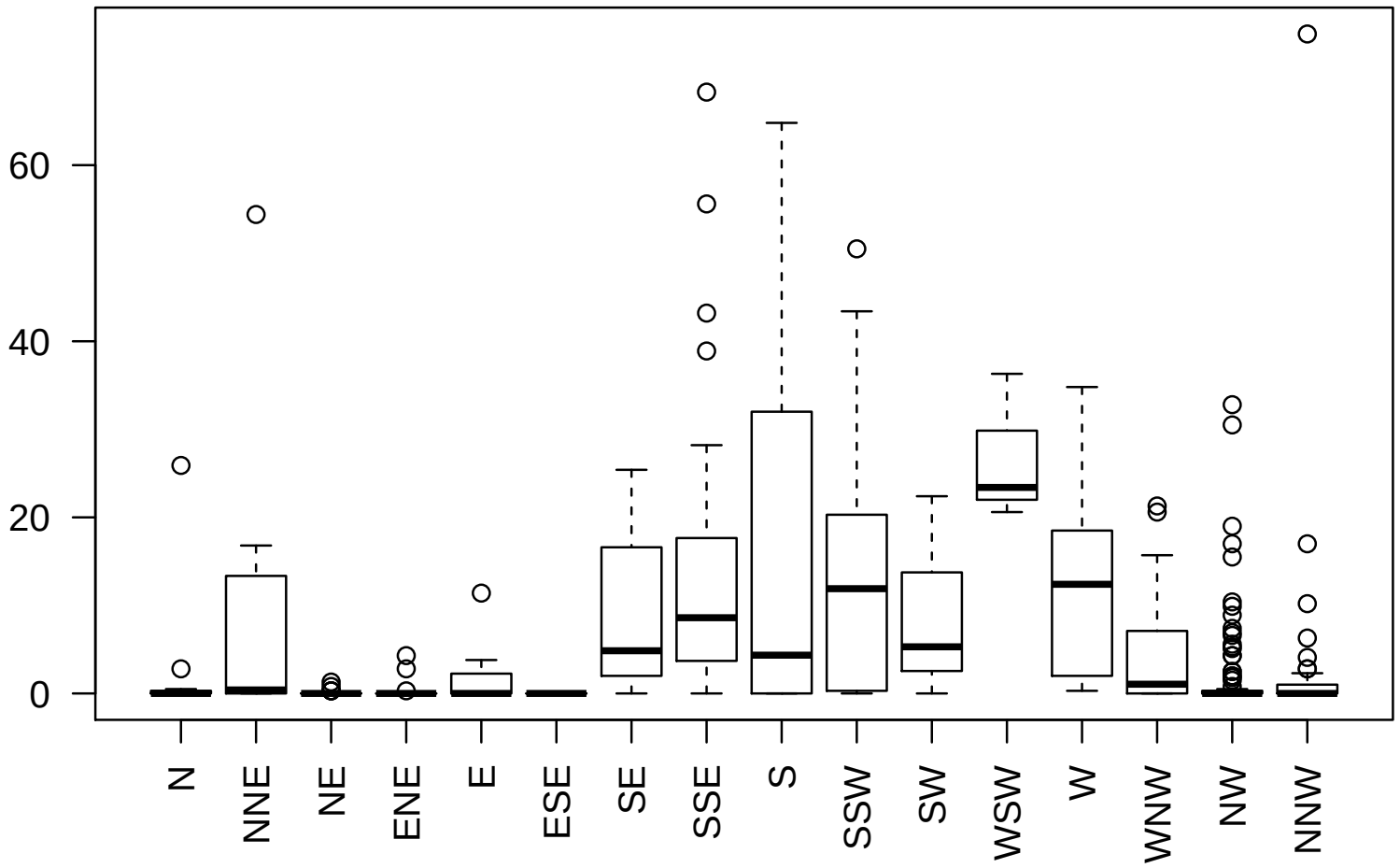
# Rainfall and wind direction

- Make boxplot of rainfall for each wind direction.
- Set wind directions to be in sensible order:

```
attach(weather.2)
dir2=ordered(dir.wind, levels=
  c("N","NNE","NE","ENE","E","ESE","SE","SSE",
    "S","SSW","SW","WSW","W","WNW","NW","NNW"))
```

# The boxplot

```
boxplot(rain~dir2,las=2)
```



# Median and count of rainfall by wind direction

```
my.sum=function(x) c(md=median(x), n=length(x))
v=aggregate(rain~dir2,FUN=my.sum)
```

```
v[1:8,]
```

```
##   dir2 rain.md rain.n
## 1    N    0.00  18.00
## 2  NNE    0.40   8.00
## 3   NE    0.00  25.00
## 4  ENE    0.00  15.00
## 5    E    0.00  11.00
## 6  ESE    0.00   2.00
## 7   SE    4.85  24.00
## 8  SSE    8.60  31.00
```

```
v[9:16,]
```

```
##    dir2 rain.md rain.n
## 9     S    4.35  26.00
## 10  SSW   11.90  17.00
## 11   SW    5.30  11.00
## 12  WSW   23.40   3.00
## 13    W   12.40   5.00
## 14  WNW    1.05  24.00
## 15   NW    0.00 108.00
## 16  NNW    0.00  37.00
```

```
detach(weather.2)
```

# Comments on weather at mystery location

- Rainfall typically highest when wind from South to West, and low when wind from North and East.
- Wind only sometimes out of S or W, but when it is, there can be a lot of rain.
- Can be up to 60 mm of rain in a day.
- Sometimes winter lows as low as 5 C.
- Sometimes summer highs over 30 C (but not often).

# So where is it?

```
library(ggmap)
map=get_map(c(-5,40),zoom=6)
```

```
# my.loc is the (hidden) name of the place
ll=geocode(my.loc)
ll

##       lon   lat
## 1 -8.629 41.16
```

# All is revealed!

```
ggmap(map)+geom_point(aes(x=lon,y=lat),
  data=ll,colour="red")
```