Motivating example

- Import libraries and define options
- Start working with data
- First plot of data
- Aggregation by conventional looping: ozone
    - Monthly mean
    - Daily maximum
- Declarative approach
    - Few variables
    - Arbitrary number of variables
        - `group_by` operation
        - pivoting
        - `stat_summary`

In this lesson, we will compare procedural and declarative approaches to computing aggregate values (mean, maximum value) from time series of concentrations at a single site.

In general, you will find that an R script often follows a set of common operations:

1. import libraries
2. define additional functions
3. import data
4. apply manipulations
5. export figures, text files

Import libraries and define options

Load libraries

```
library(dplyr)
library(reshape2)
library(chron)
library(ggplot2)
```

```
source("GRB001.R")
```

Define options

```
Sys.setlocale("LC_TIME","C")
```

```
## [1] "C"
```

```
options(stringsAsFactors=FALSE)
options(chron.year.abb=FALSE)
theme_set(theme_bw()) # just my preference for plots
```

Start working with data

The data is available from the National Air Pollution Monitoring Network (NABEL) (http://www.bafu.admin.ch/luft/00612/00625/index.html?lang=en) of Switzerland.

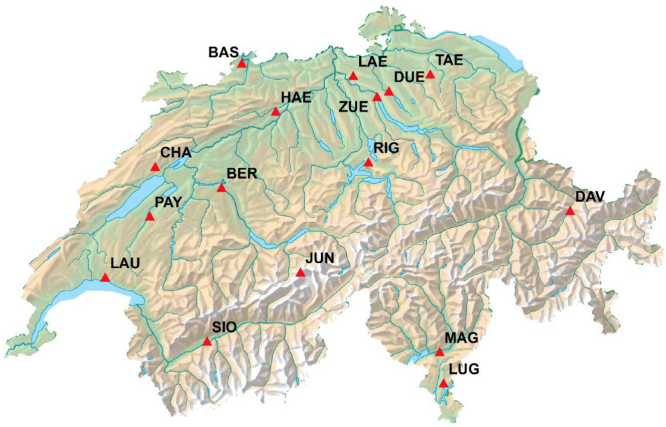


Image source: *Stations de mesure NABEL* report

We have downloaded hourly time series from 2013 for Lausanne from the NABEL data query (http://www.bafu.admin.ch/luft/luftbelastung/blick_zurueck/datenabfrage/index.html?lang=en), and placed this file in a folder called "data/2013/" located in the subdirectory of the *working directory*.

First, check your working directory:

```
getwd()
```

Define your input file relative to this path:

```
filename <- file.path("data", "2013", "LAU.csv")
file.exists(filename)
```

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

Read data table:

```
data <- read.table(filename,sep=";",skip=6,
  col.names=c("datetime","O3","NO2","CO","PM10","TEMP","PREC","RAD"))
```

Check a sample of your data:

```
head(data)
```

```
##           datetime   O3  NO2  CO PM10 TEMP PREC  RAD
## 1 31.12.2012 01:00  7.8 56.3 0.5 16.1  3.8    0 -2.4
## 2 31.12.2012 02:00 22.4 38.0 0.4 11.6  4.1    0 -2.3
## 3 31.12.2012 03:00 14.5 37.2 0.3 10.3  3.1    0 -2.1
## 4 31.12.2012 04:00 28.7 25.4 0.3 10.5  3.5    0 -2.2
## 5 31.12.2012 05:00 19.6 33.7 0.3  9.0  2.9    0 -2.2
## 6 31.12.2012 06:00 30.8 51.2 0.3  8.7  3.2    0 -2.3
```

Check the structure of your object:

```
str(data)
```

```
## 'data.frame':    8784 obs. of  8 variables:
##  $ datetime: chr  "31.12.2012 01:00" "31.12.2012 02:00" "31.12.2012 03:00" "31.12.2012 04:00"
...
##  $ O3      : num  7.8 22.4 14.5 28.7 19.6 30.8 25.1 28.3 19.4 23.7 ...
##  $ NO2     : num  56.3 38 37.2 25.4 33.7 51.2 51.7 44.2 60.6 53.8 ...
##  $ CO      : num  0.5 0.4 0.3 0.3 0.3 0.3 0.4 0.3 0.4 0.4 ...
##  $ PM10    : num  16.1 11.6 10.3 10.5 9 8.7 8.6 9.7 10.2 11.2 ...
##  $ TEMP    : num  3.8 4.1 3.1 3.5 2.9 3.2 3.3 2.9 2.8 3.5 ...
##  $ PREC    : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ RAD     : num  -2.4 -2.3 -2.1 -2.2 -2.2 ...
```

Check column classes:

```
ColClasses(data)
```

datetime O3 NO2 CO PM10 TEMP PREC RAD 1 character numeric numeric numeric numeric numeric numeric numeric

Convert date/time to useful data types - the hourly timestamps in the file denote the end of the measurement periods so we will convert them to start times by subtracting one hour (out of 24):

```
data[,"datetime"] <- as.chron(data[,"datetime"], "%d.%m.%Y %H:%M") - 1/24
data[,"month"] <- months(data[,"datetime"])
data[,"date"] <- dates(data[,"datetime"])
```

Check data sample, structure, and column classes:

```
head(data)
```

```
##                datetime   O3  NO2  CO PM10 TEMP PREC  RAD month       date
## 1 (12/31/2012 00:00:00)  7.8 56.3 0.5 16.1  3.8    0 -2.4   Dec 12/31/2012
## 2 (12/31/2012 01:00:00) 22.4 38.0 0.4 11.6  4.1    0 -2.3   Dec 12/31/2012
## 3 (12/31/2012 02:00:00) 14.5 37.2 0.3 10.3  3.1    0 -2.1   Dec 12/31/2012
## 4 (12/31/2012 03:00:00) 28.7 25.4 0.3 10.5  3.5    0 -2.2   Dec 12/31/2012
## 5 (12/31/2012 04:00:00) 19.6 33.7 0.3  9.0  2.9    0 -2.2   Dec 12/31/2012
## 6 (12/31/2012 05:00:00) 30.8 51.2 0.3  8.7  3.2    0 -2.3   Dec 12/31/2012
```

```
str(data)
```

```
## 'data.frame':    8784 obs. of  10 variables:
##  $ datetime: 'chron' num  (12/31/2012 00:00:00) (12/31/2012 01:00:00) (12/31/2012 02:00:00)
(12/31/2012 03:00:00) (12/31/2012 04:00:00) ...
##   ..- attr(*, "format")= Named chr  "m/d/y" "h:m:s"
##   .. ..- attr(*, "names")= chr  "dates" "times"
##   ..- attr(*, "origin")= Named num  1 1 1970
##   .. ..- attr(*, "names")= chr  "month" "day" "year"
##  $ O3      : num  7.8 22.4 14.5 28.7 19.6 30.8 25.1 28.3 19.4 23.7 ...
##  $ NO2     : num  56.3 38 37.2 25.4 33.7 51.2 51.7 44.2 60.6 53.8 ...
##  $ CO      : num  0.5 0.4 0.3 0.3 0.3 0.3 0.4 0.3 0.4 0.4 ...
##  $ PM10    : num  16.1 11.6 10.3 10.5 9 8.7 8.6 9.7 10.2 11.2 ...
##  $ TEMP    : num  3.8 4.1 3.1 3.5 2.9 3.2 3.3 2.9 2.8 3.5 ...
##  $ PREC    : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ RAD     : num  -2.4 -2.3 -2.1 -2.2 -2.2 ...
##  $ month   : Ord.factor w/ 12 levels "Jan"<"Feb"<"Mar"<..: 12 12 12 12 12 12 12 12 12 12 ...
##  $ date    : 'dates' num  12/31/2012 12/31/2012 12/31/2012 12/31/2012 12/31/2012 ...
##   ..- attr(*, "format")= Named chr  "m/d/y" "h:m:s"
##   .. ..- attr(*, "names")= chr  "dates" "times"
##   ..- attr(*, "origin")= Named num  1 1 1970
##   .. ..- attr(*, "names")= chr  "month" "day" "year"
```

```
ColClasses(data)
```

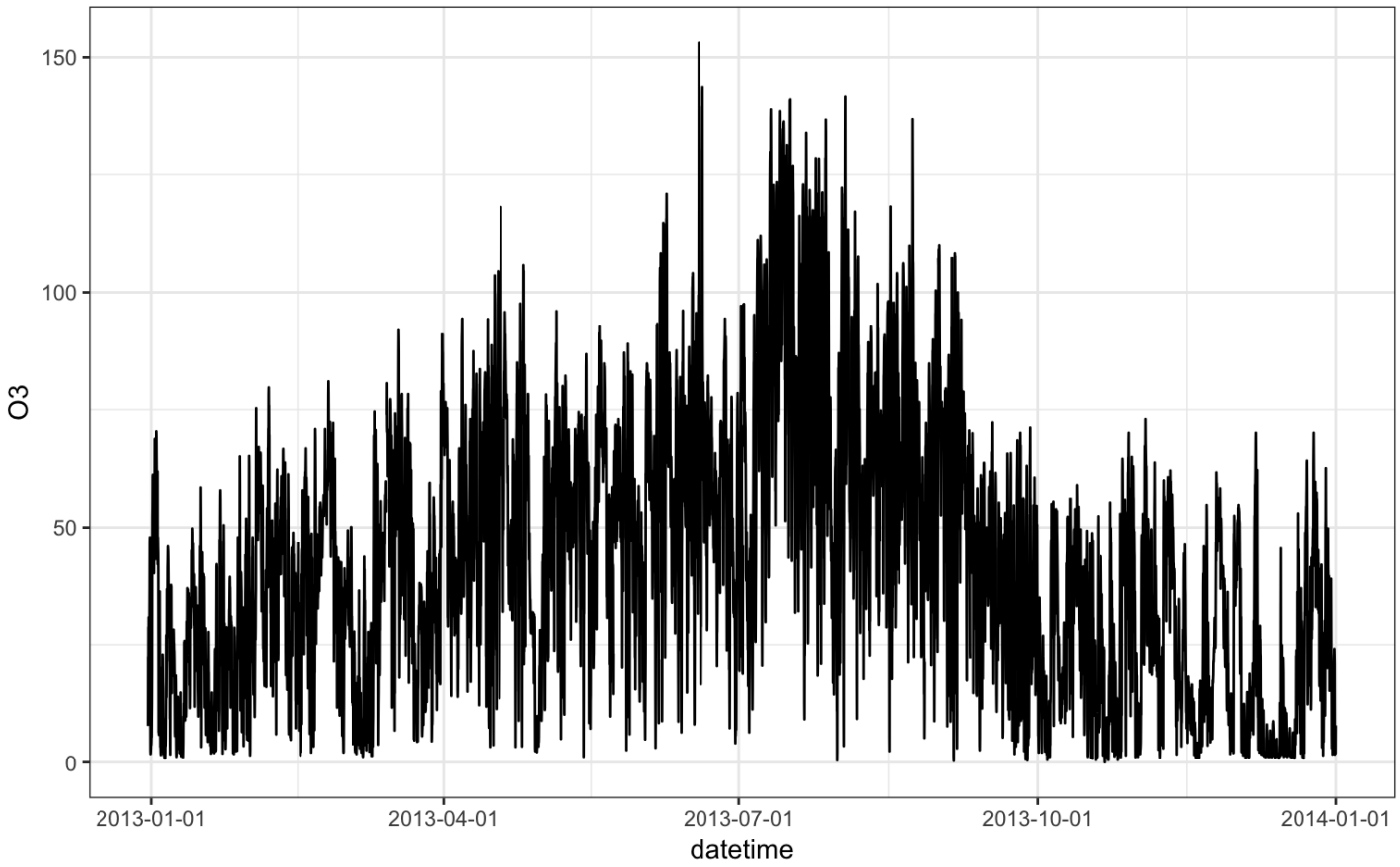
```
       datetime      O3     NO2      CO    PM10    TEMP    PREC
```

1 chron,dates,times numeric numeric numeric numeric numeric numeric RAD month date 1 numeric ordered,factor dates,times

First plot of data

View raw ozone concentrations:

```
ggp <- ggplot(data)+
  geom_line(aes(datetime, O3))+
    scale_x_chron()
print(ggp)
```



Aggregation by conventional looping: ozone

Monthly mean

Solve by looping. Note that we "grow" a data frame by the function `rbind` .

```
unique.months <- levels(data[,"month"])

O3.monthly <- NULL
for(.month in unique.months) {
  table <- filter(data, month == .month)
  tmp <- data.frame(month=.month, O3=mean(table[,"O3"], na.rm=TRUE))
  O3.monthly <- rbind(O3.monthly, tmp)
}

print(O3.monthly)
```

```
##    month       O3
## 1    Jan 22.68531
## 2    Feb 40.58036
## 3    Mar 35.06937
## 4    Apr 50.70181
## 5    May 51.88733
## 6    Jun 59.64025
## 7    Jul 76.14097
## 8    Aug 66.58543
## 9    Sep 43.10642
## 10   Oct 24.73957
## 11   Nov 25.89110
## 12   Dec 18.61682
```

Convert month column from character string to “factor”:

```
class(O3.monthly[,"month"])
```

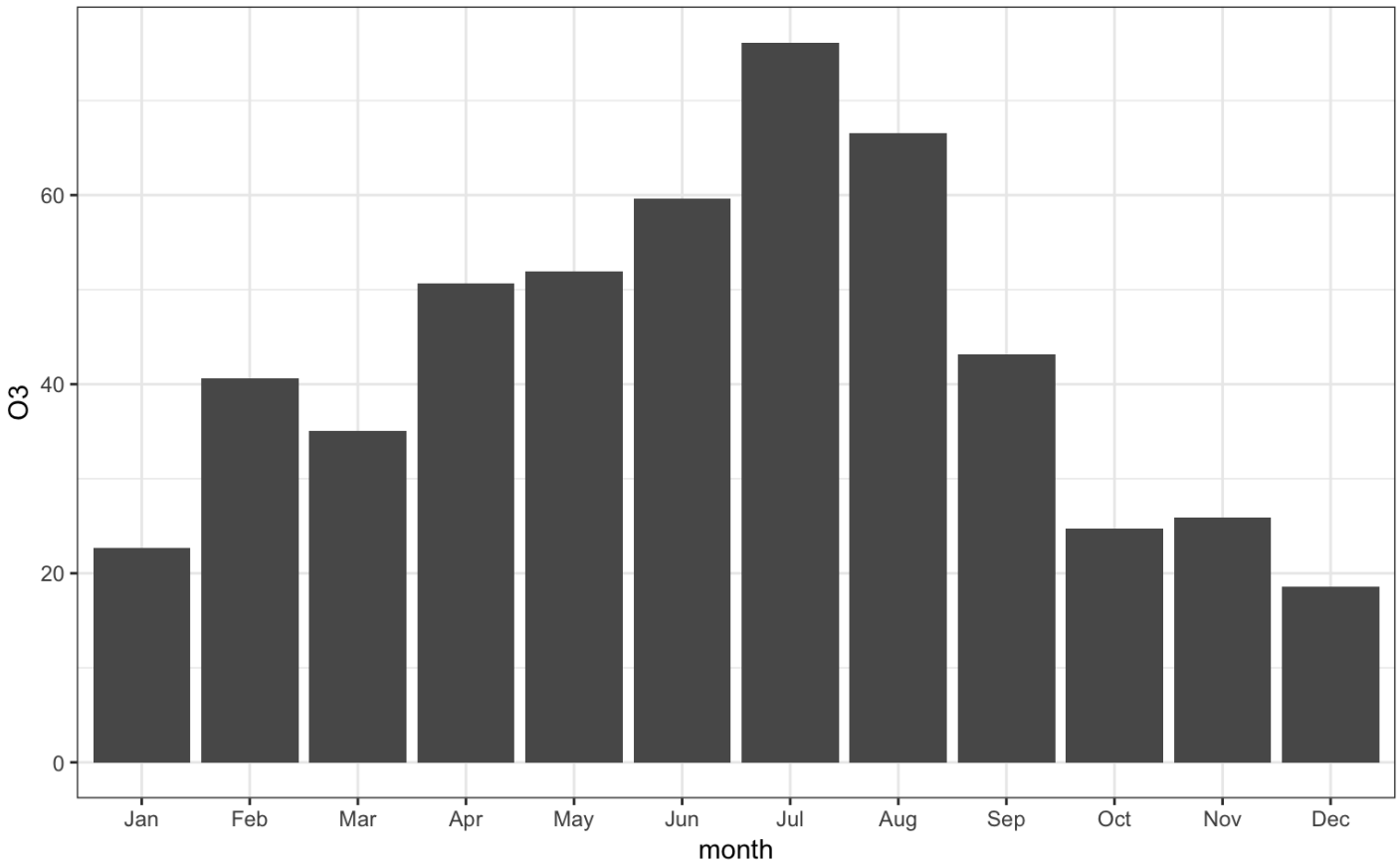
```
## [1] "character"
```

```
O3.monthly[,"month"] <- factor(O3.monthly[,"month"], unique.months)
class(O3.monthly[,"month"])
```

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

Another visual representation:

```
ggp <- ggplot(O3.monthly) +
  geom_bar(aes(month, O3), stat="identity")
print(ggp)
```



Daily maximum

Calculate:

```
unique.dates <- unique(data[,"date"])
O3.dailymax <- NULL
for(.date in unique.dates) {
  table <- data %>% filter(date == .date)
  tmp <- data.frame(date=.date, O3=max(table[,"O3"], na.rm=TRUE))
  O3.dailymax <- rbind(O3.dailymax, tmp)
}

head(O3.dailymax)
```

```
##    date   O3
## 1 15705 47.9
## 2 15706 61.2
## 3 15707 70.4
## 4 15708 47.9
## 5 15709 22.9
## 6 15710 36.8
```

Convert date column to chron object:

```
class(O3.dailymax[,"date"])
```

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

```
O3.dailymax[,"date"] <- as.chron(O3.dailymax[,"date"])
class(O3.dailymax[,"date"])
```

```
## [1] "dates" "times"
```

Inspect:

```
head(O3.dailymax)
```

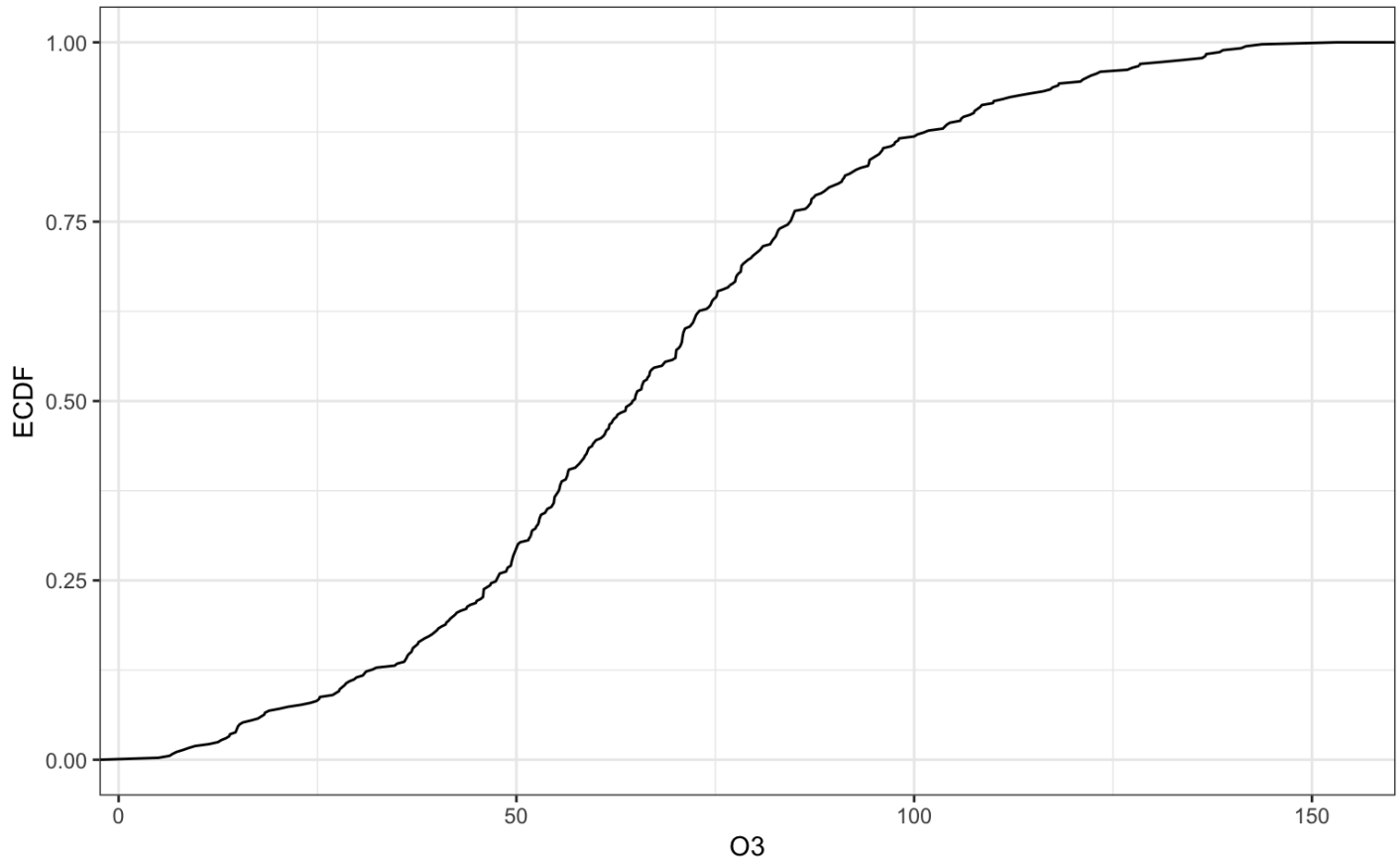
```
##         date   O3
## 1 12/31/2012 47.9
## 2 01/01/2013 61.2
## 3 01/02/2013 70.4
## 4 01/03/2013 47.9
## 5 01/04/2013 22.9
## 6 01/05/2013 36.8
```

```
tail(O3.dailymax)
```

```
##           date   O3
## 361 12/26/2013 57.4
## 362 12/27/2013 42.0
## 363 12/28/2013 62.6
## 364 12/29/2013 49.9
## 365 12/30/2013 39.0
## 366 12/31/2013 24.1
```

Plot ECDF (empirical cumulative distribution function):

```
ggp <- ggplot(O3.dailymax) +
  geom_line(aes(O3),stat="ecdf") +
  labs(y = "ECDF")
print(ggp)
```



Declarative approach

Few variables

With a single expression, we reproduced the loop used to create `O3.dailymax` :

```
data %>% group_by(month) %>%
  summarize(O3 = mean(O3, na.rm=TRUE))
```

```
## # A tibble: 12 x 2
##    month    O3
##    <ord> <dbl>
##  1 Jan    22.7
##  2 Feb    40.6
##  3 Mar    35.1
##  4 Apr    50.7
##  5 May    51.9
##  6 Jun    59.6
##  7 Jul    76.1
##  8 Aug    66.6
##  9 Sep    43.1
## 10 Oct    24.7
## 11 Nov    25.9
## 12 Dec    18.6
```

We can easily extend to two variables:

```
data %>% group_by(month) %>%
  summarize(O3 = mean(O3, na.rm=TRUE),
            NO2 = mean(NO2, na.rm=TRUE))
```

```
## # A tibble: 12 x 3
##    month    O3   NO2
##    <ord> <dbl> <dbl>
##  1 Jan    22.7  47.5
##  2 Feb    40.6  45.1
##  3 Mar    35.1  50.3
##  4 Apr    50.7  40.5
##  5 May    51.9  38.7
##  6 Jun    59.6  38.9
##  7 Jul    76.1  38.5
##  8 Aug    66.6  34.1
##  9 Sep    43.1  42.0
## 10 Oct    24.7  40.6
## 11 Nov    25.9  36.3
## 12 Dec    18.6  57.0
```

Arbitrary number of variables

We first transform the data frame:

```
lf <- melt(data, id.vars=c("datetime", "month", "date"))
```

Let us inspect this transformation:

```
head(lf)
```

```
##                datetime month       date variable value
## 1 (12/31/2012 00:00:00)   Dec 12/31/2012       O3   7.8
## 2 (12/31/2012 01:00:00)   Dec 12/31/2012       O3  22.4
## 3 (12/31/2012 02:00:00)   Dec 12/31/2012       O3  14.5
## 4 (12/31/2012 03:00:00)   Dec 12/31/2012       O3  28.7
## 5 (12/31/2012 04:00:00)   Dec 12/31/2012       O3  19.6
## 6 (12/31/2012 05:00:00)   Dec 12/31/2012       O3  30.8
```

```
tail(lf)
```

```
##                    datetime month       date variable value
## 61483 (12/31/2013 18:00:00)   Dec 12/31/2013      RAD  -2.4
## 61484 (12/31/2013 19:00:00)   Dec 12/31/2013      RAD  -2.4
## 61485 (12/31/2013 20:00:00)   Dec 12/31/2013      RAD  -2.3
## 61486 (12/31/2013 21:00:00)   Dec 12/31/2013      RAD  -2.2
## 61487 (12/31/2013 22:00:00)   Dec 12/31/2013      RAD  -1.6
## 61488 (12/31/2013 23:00:00)   Dec 12/31/2013      RAD  -1.3
```

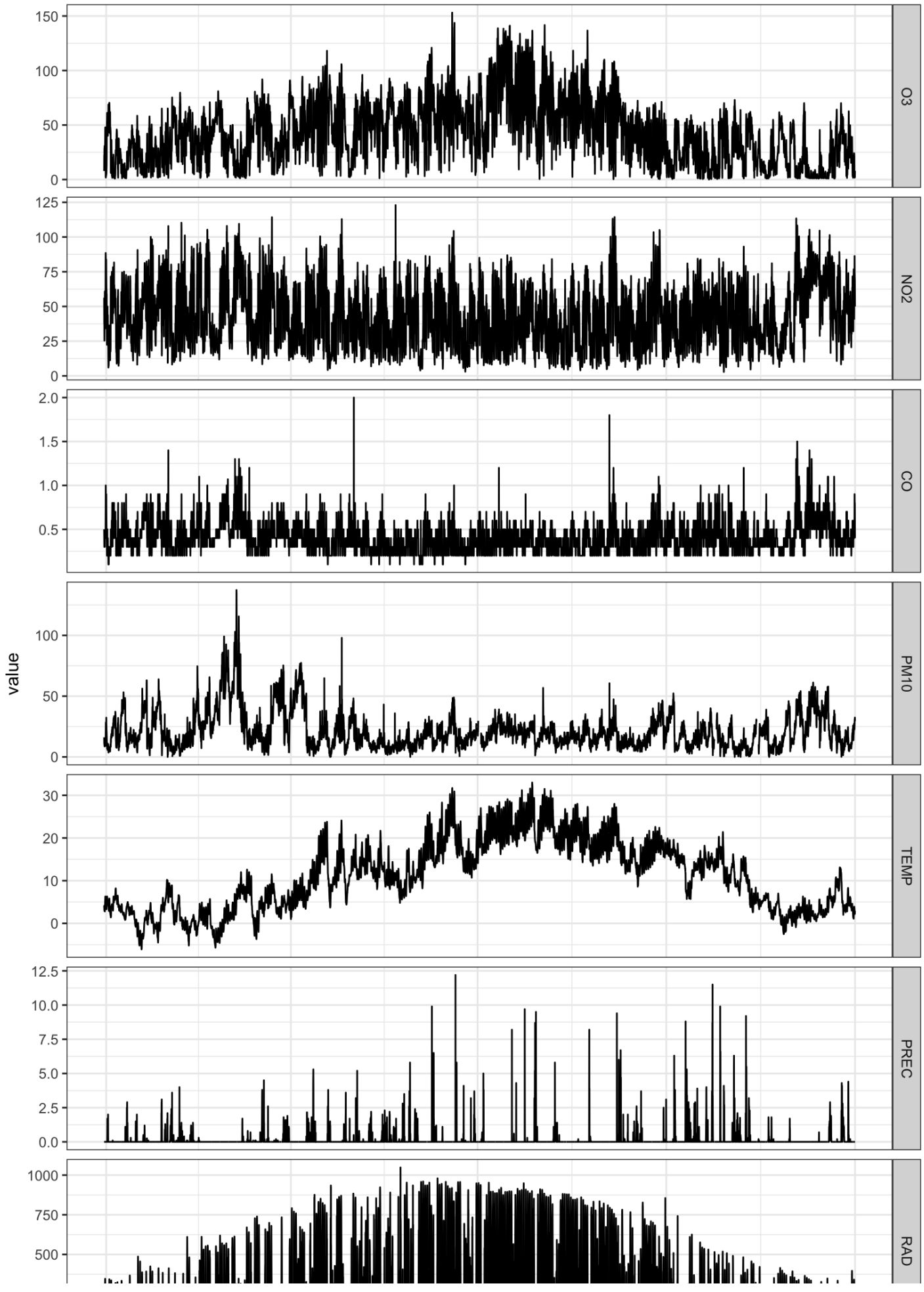
```
ColClasses(lf)
```

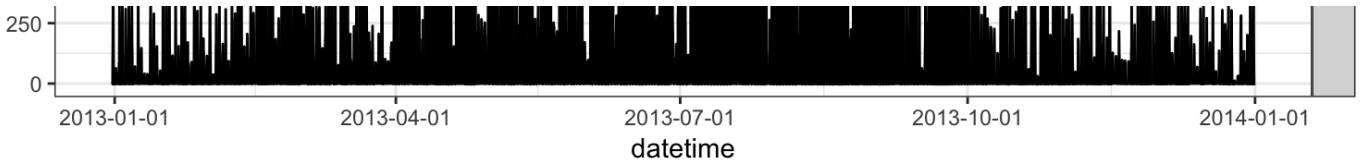
```
      datetime          month        date variable   value
```

1 chron,dates,times ordered,factor dates,times factor numeric

This is amenable for plotting:

```
ggp <- ggplot(lf) +
  geom_line(aes(datetime, value))+
  facet_grid(variable~., scale="free_y") +
  scale_x_chron()
print(ggp)
```


value
O3
NO2
CO
PM10
TEMP
PREC
RAD



Using this, we can aggregate using three approaches:

- `group_by` : as illustrated above
- `dcast` : aggregate statistics through pivoting
- `stat_summary` : called through geom operation

group_by operation

```
result <- lf %>% group_by(month, variable) %>%
  summarize(value = mean(value, na.rm=TRUE))
```

The inverse opration of melt:

```
dcast(result, month~variable)
```

```
##    month       O3      NO2        CO     PM10       TEMP       PREC
## 1    Jan 22.68531 47.48880 0.4436070 22.53567  2.2334677 0.08721400
## 2    Feb 40.58036 45.07351 0.4385417 28.83949  0.5916667 0.07113095
## 3    Mar 35.06937 50.29205 0.4845296 35.92396  4.7494624 0.10713324
## 4    Apr 50.70181 40.49846 0.3930556 25.51897 10.6244444 0.11930556
## 5    May 51.88733 38.65243 0.3339166 12.26275 11.9717742 0.18655914
## 6    Jun 59.64025 38.89764 0.3384722 17.57620 17.6844444 0.12420028
## 7    Jul 76.14097 38.49704 0.3684140 19.92669 22.3954301 0.16424731
## 8    Aug 66.58543 34.09595 0.3333333 17.26788 20.9392473 0.05397039
## 9    Sep 43.10642 42.02409 0.3945833 17.51944 17.0012500 0.12472222
## 10   Oct 24.73957 40.55760 0.4133065 17.47137 13.7209677 0.28721400
## 11   Nov 25.89110 36.34673 0.3694444 13.64259  5.9900000 0.15724234
## 12   Dec 18.61682 57.03859 0.5252604 25.66237  3.8727865 0.13151042
##          RAD
## 1   46.31935
## 2   84.99851
## 3   98.91478
## 4  170.55069
## 5  174.21237
## 6  248.54028
## 7  272.65780
## 8  241.67285
## 9  157.18528
## 10  82.58642
## 11  53.88472
## 12  46.68398
```

pivoting

The mean can also be calculated directly:

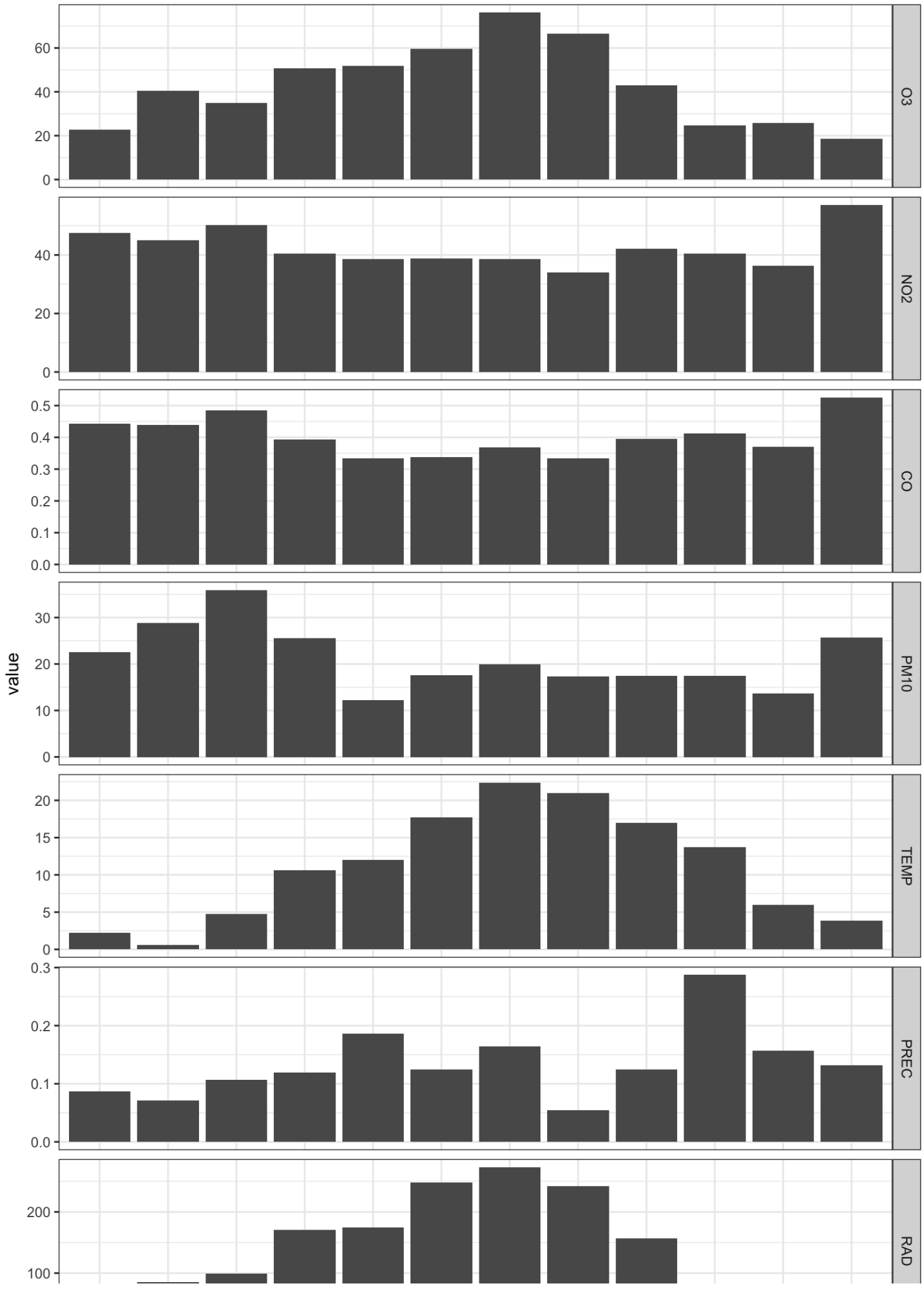
```
dcast(lf, month~variable, fun.aggregate=mean, na.rm=TRUE)
```

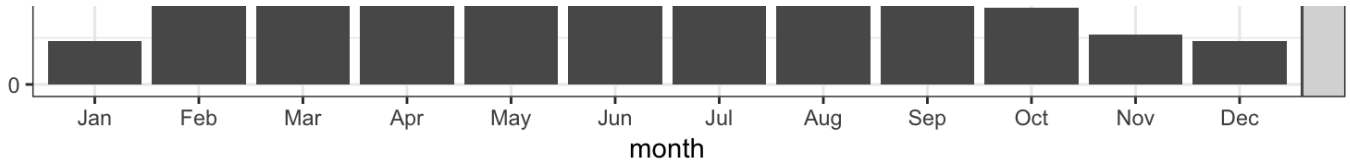
```
##    month       O3      NO2        CO     PM10       TEMP       PREC
## 1    Jan 22.68531 47.48880 0.4436070 22.53567  2.2334677 0.08721400
## 2    Feb 40.58036 45.07351 0.4385417 28.83949  0.5916667 0.07113095
## 3    Mar 35.06937 50.29205 0.4845296 35.92396  4.7494624 0.10713324
## 4    Apr 50.70181 40.49846 0.3930556 25.51897 10.6244444 0.11930556
## 5    May 51.88733 38.65243 0.3339166 12.26275 11.9717742 0.18655914
## 6    Jun 59.64025 38.89764 0.3384722 17.57620 17.6844444 0.12420028
## 7    Jul 76.14097 38.49704 0.3684140 19.92669 22.3954301 0.16424731
## 8    Aug 66.58543 34.09595 0.3333333 17.26788 20.9392473 0.05397039
## 9    Sep 43.10642 42.02409 0.3945833 17.51944 17.0012500 0.12472222
## 10   Oct 24.73957 40.55760 0.4133065 17.47137 13.7209677 0.28721400
## 11   Nov 25.89110 36.34673 0.3694444 13.64259  5.9900000 0.15724234
## 12   Dec 18.61682 57.03859 0.5252604 25.66237  3.8727865 0.13151042
##          RAD
## 1   46.31935
## 2   84.99851
## 3   98.91478
## 4  170.55069
## 5  174.21237
## 6  248.54028
## 7  272.65780
## 8  241.67285
## 9  157.18528
## 10  82.58642
## 11  53.88472
## 12  46.68398
```

stat_summary

The mean cal also be calculated in the process of plotting:

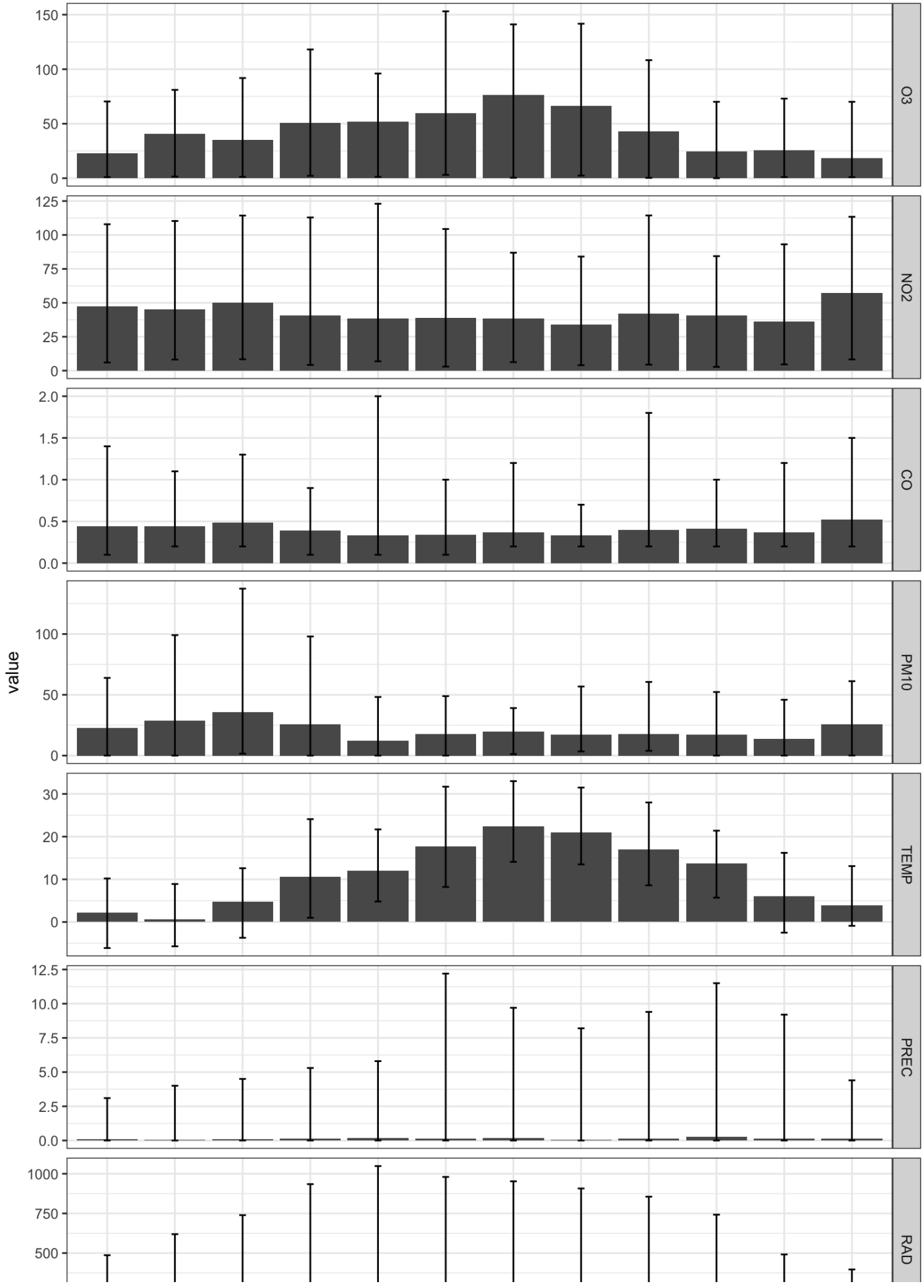
```
ggp <- ggplot(lf) +
  geom_bar(aes(month, value), stat="summary", fun.y="mean")+
  facet_grid(variable~., scale="free_y")
print(ggp)
```

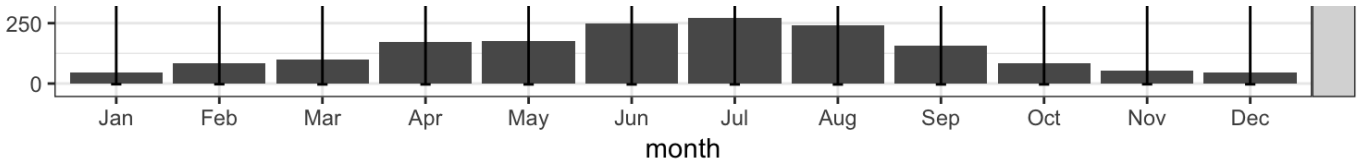

value
O3
NO2
CO
PM10
TEMP
PREC
RAD



Add errorbars to denote the full range in values:

```
ggp <- ggplot(lf, aes(month, value)) +
  geom_bar(stat="summary", fun.y="mean")+
  geom_errorbar(stat="summary",
                fun.ymin=min, #function(x) quantile(x, .25),
                fun.ymax=max, #function(x) quantile(x, .75))+
                width=0.1) +
  facet_grid(variable~., scale="free_y")
print(ggp)
```


value
O3
NO2
CO
PM10
TEMP
PREC
RAD


250
0
Jan
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec
month