Learning *R*

Carl James Schwarz

StatMathComp Consulting by Schwarz
cschwarz.stat.sfu.ca @ gmail.com

Dealing with Dates and Times

Table of Contents I

Dealing with Dates and Times

Dates and Times are more complicated than you think!

- Does every year have 365 days?
- Does every day have 24 hours?
- Does every minute have 60 seconds?
- When it is 12:00 noon in Vancouver, what time is it in Toronto? In Regina?

Dates and Times are more complicated than you think!

- Does every year have 365 days?
  - Leap vs. non-leap years, but is 1900 a leap year?
- Does every day have 24 hours?
  - During changes to/from daylight savings time, days can have 23/25 hours.
- Does every minute have 60 seconds?
  - Leap seconds are occasionally added
- When it is 12:00 noon in Vancouver, what time is it in Toronto? In Regina?
  - Both Ontario and much of BC co-ordinate daylight savings time. Saskatchewan does not observe daylight savings time.

Date and Times in *R*

Types of Dates and Times

- Date values, e.g. 2013-01-12
- DateTime (time stamp) values, e.g. 2013-01-12 23:14
- Time values within a day, e.g. 23:14
- Duration values, e.g. 65:14 is 65 hours and 14 minutes and not a clock time; 9 months and 3 days for a pregnancy is a duration.
- Period values, e.g. 1 month can be 28, 29, 30 or 31 days long.

Some summary articles:

- `http://www.noamross.net/blog/2014/2/10/using-times-and-dates-in-r---presentation-code.html`
- `http://www.statmethods.net/input/dates.html`
- `http://www.jstatsoft.org/v40/i03/paper` *lubridate* package

Dates in *R*

Dealing with Dates only is 3 step process but is straightforward.

- Input Date values as a character string (e.g. "23/01/2013") (use the as.is=TRUE option on *read.table()* or *read.csv()*)
- Convert to internal form (number of days since 1970-01-01)
  - Use the *as.Date(string, format="format codes")* function with the format codes described in *help(strptime)*.
  - (Internal) negative values indicate days before the origin.
  - Allows arithmetic in the usual fashion.
- Convert to display format (default is yyyy-mm-dd) and/or extract parts of the date using *format(datevar, "format codes")*.

Leap years are properly handled (unlike Excel where 1900 is treated incorrectly).

Create the textConnection and then read in the following data.

```
testDates <- textConnection("
d1c,    d2c,     d3c
23aug01, 1970-07-10, 13/07/1956
14sep01, 1972-07-11, 14/07/1956
30feb01, 1972-13-12, 15/07/1956
1mar2001,72-08-14,   16/07/1956") # example of inputing dat

my.dates <- read.csv(testDates, header=TRUE,
                     as.is=TRUE, strip.white=TRUE)
my.dates
str(my.dates)
```

At this point, all of the variables are CHARACTER data type.

Dates in *R* - Converting to internal format

Convert to internal format using codes from *help(strptime)*

```
1  my.dates$d1 <- as.Date(my.dates$d1c,
2                         format="%d%b%y")
3  my.dates$d1
4  as.numeric(my.dates$d1)
5
6  my.dates$d2 <- as.Date(my.dates$d2c,
7                         format="%Y-%m-%d")
8  my.dates$d2
9  as.numeric(my.dates$d2)
10
11 my.dates$d3 <- as.Date(my.dates$d3c,
12                         format="%d/%m/%y")
13 my.dates$d3
14 as.numeric(my.dates$d3)
```

- CAUTION - century implied for %y format character. Always use 4-digit yyyy in input data.
- CAUTION - cannot mix yy and yyyy when using %y or %Y. Always use 4-digit yyyy in input data

Dates in *R* - Arithmetic on Dates

The usual arithmetic operations deal with the internal Dates in sensible ways

```
1  my.dates$d3 + 20   # adds 20 days
2
3  mean(my.dates$d3)
4  as.numeric(mean(my.dates$d3)) # correct average stored here
5
6  seq(from=my.dates$d3[1], by="3 weeks", length.out=3) #seque
7  seq(from=my.dates$d3[1], by="2 months", length.out=3)
8  seq(from=my.dates$d3[1], by="1 year",   length.out=3)
```

Dates in *R* - Arithmetic on Dates

Extracting parts of a Date. Use the format codes in *help(strptime)* CAUTION: results of *format()* are always CHARACTER and may need conversion to numeric values.

```
1  # Extract the day of the month
2  format(my.dates$d3, "%d")
3  format(my.dates$d3, "%d") + 1  # oops
4  as.numeric(format(my.dates$d3, "%d"))
5
6  # Extract day of the week (0=Sunday)
7  as.numeric(format(my.dates$d3, "%w"))
8
9  # Julian day (number of days since 1 Jan of that year) 001-
10 as.numeric(format(my.dates$d3, "%j"))
```

Other useful functions are available in the *lubridate* package.

- *ymd('2017-01-31')*, *dmy("31/01/17")*, etc
- *make_date(year=, month=, day=)*
- *year('2017-01-31')*, *month('2017-01-31')*, etc.

Dates in *R* - Exercise I

Refer to *road-accidents-summary-2010.csv* file in SampleData.

- Read data into *R*.
- Convert input date to internal *R* dates.
- Plot # accidents/day by day of year.
- Fit a *lowess()* smoother to data using *geom_smooth()*
- Find proportion of fatalities by day of year, plot, and fit lowess curve.
- Can create both plots in the same window (Hint: melt the two variables and use facetting).

Look at number of accident by day of the week

- Extract day of the week using *format()* or *weekdays()* functions.
- Use *geom_boxplot()* as seen earlier

Dates in *R* - Exercise I

```
1  # The accident data
2  naccidents <- read.csv(file.path(...,'road-accidents-2010-su
3                header=TRUE,
4                as.is=TRUE, strip.white=TRUE)
5  naccidents[1:5,]
6  str(naccidents)
```

```
> naccidents[1:5,]
        Date naccidents nfatal
1 01/01/2010        282      4
2 01/02/2010        657      5
3 01/03/2010        608      2
...
```

```
> str(naccidents)
'data.frame': 365 obs. of  3 variables:
 $ Date       : chr  "01/01/2010" "01/02/2010" "01/03/2010" "
...
```

Dates in *R* - Exercise I

```
1  # Convert date to internal date format
2  naccidents$mydate <- as.Date(naccidents$Date,
3            format="%d/%m/%Y")
4  sum(is.na(naccidents$mydate))
5  naccidents[1:5,]
6  str(naccidents)
```

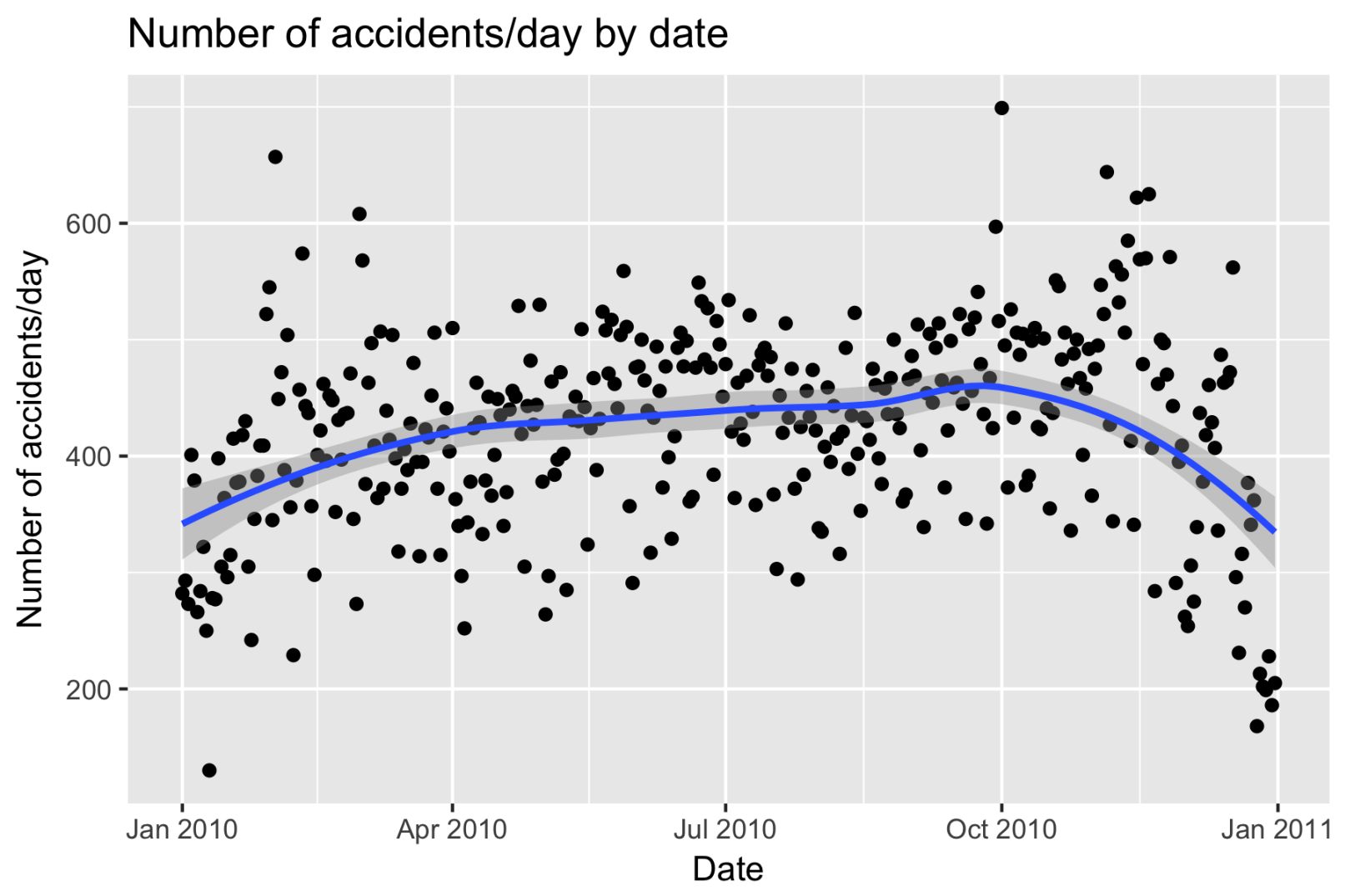
```
> naccidents[1:5,]
        Date naccidents nfatal     mydate
1 01/01/2010        282      4 2010-01-01
2 01/02/2010        657      5 2010-02-01

> str(naccidents)
'data.frame': 365 obs. of  4 variables:
 $ Date      : chr  "01/01/2010" "01/02/2010" "01/03/2010" "
 $ mydate    : Date, format: "2010-01-01" "2010-02-01" "201
>
```

Dates in *R* - Exercise I

```
1  plotnacc <- ggplot(data=naccidents, aes(x=mydate, y=naccider
2        ggtitle("Number of accidents/day by date")+
3        xlab("Date")+ylab("Number of accidents/day")+
4        geom_point()+
5        geom_smooth()
6  plotnacc
```

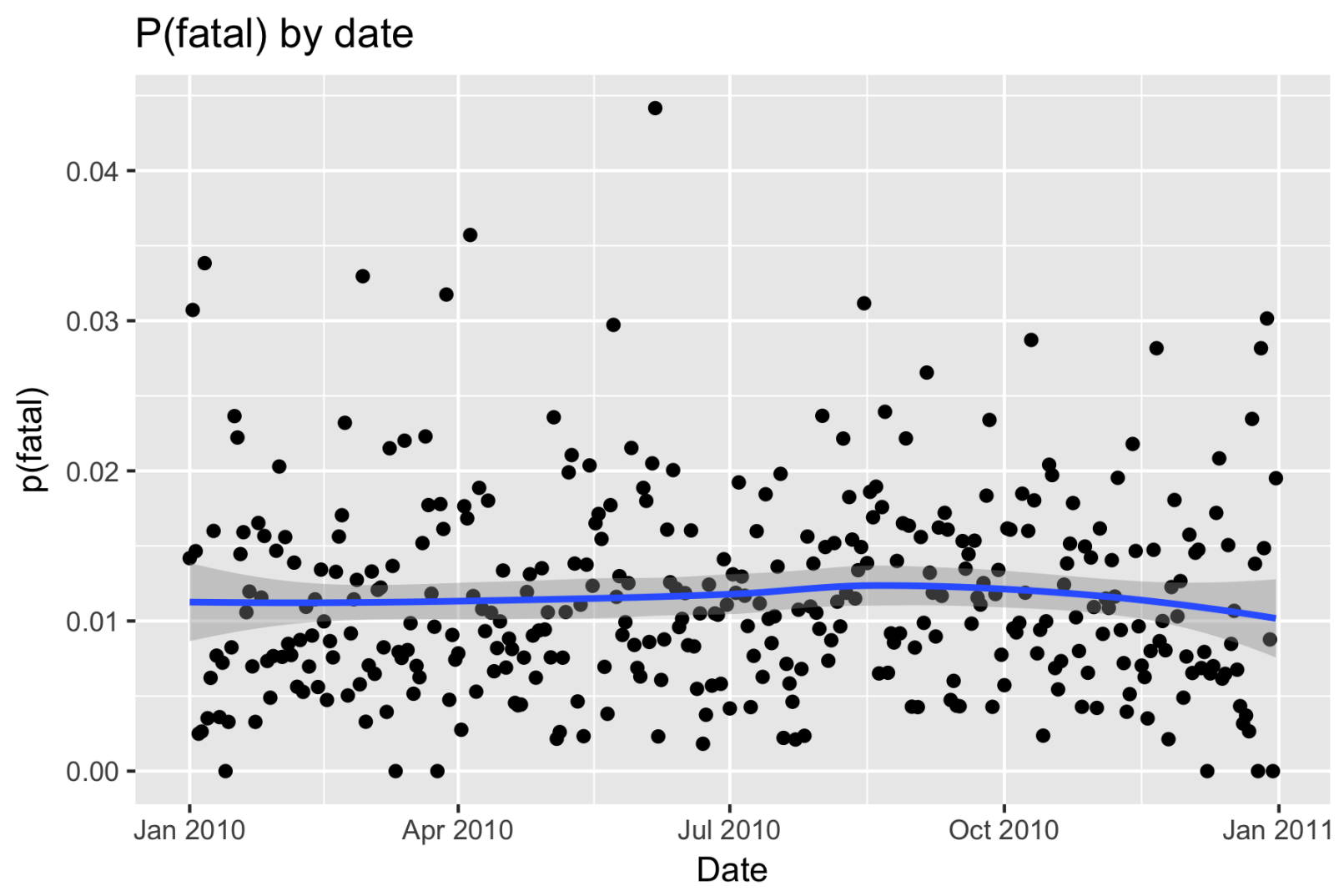
Dates in *R* - Exercise I



Dates in *R* - Exercise I

```
1  # look at proportion of fatalities
2  naccidents$pfatal <- naccidents$nfatal /
3                       naccidents$naccidents
4  plotpfatal <- ggplot(data=naccidents, aes(x=mydate, y=pfatal
5       ggtitle("P(fatal) by date")+
6       xlab("Date")+ylab("p(fatal)")+
7       geom_point()+
8       geom_smooth()
9  plotpfatal
```

Dates in *R* - Exercise I



Dates in *R* - Exercise I

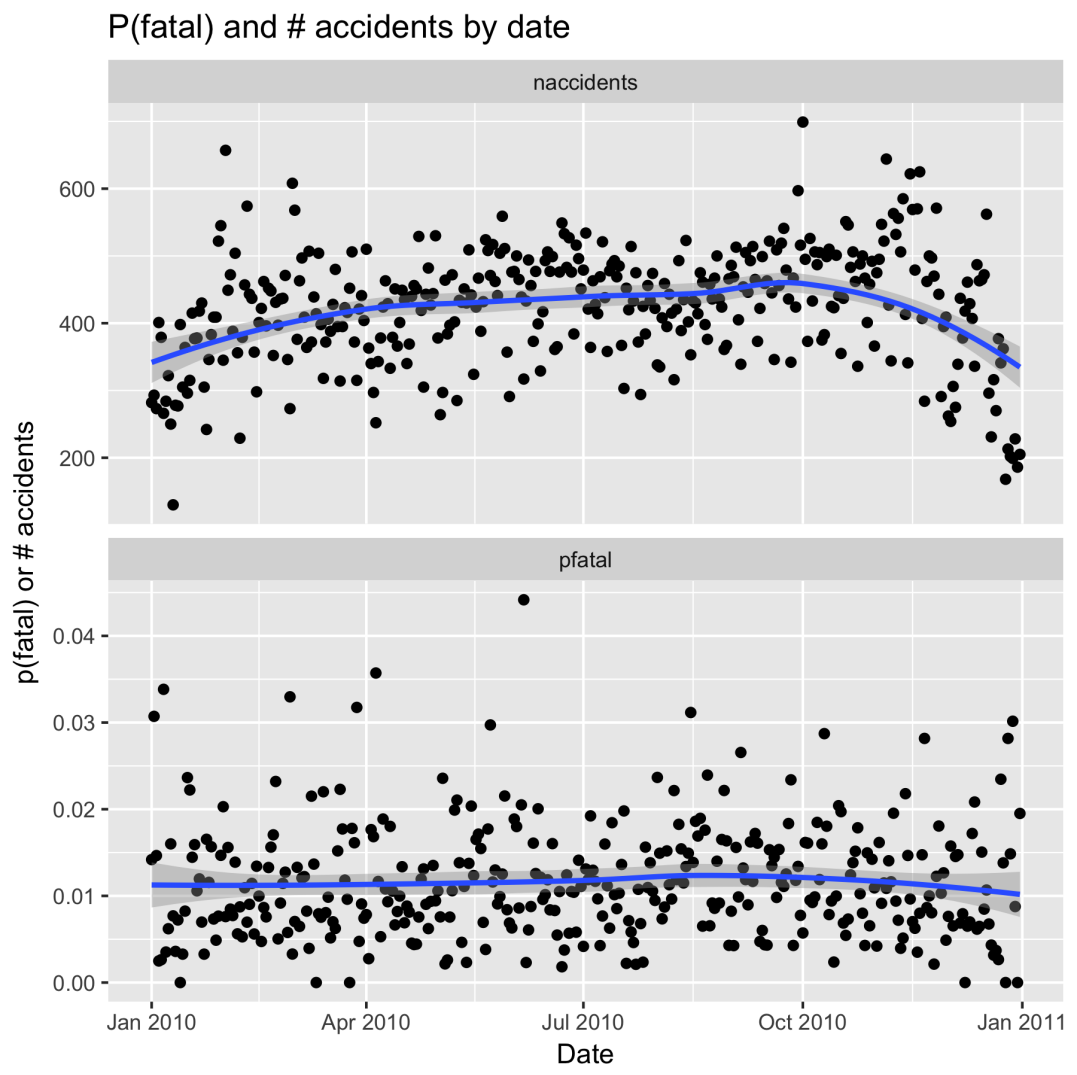
```
1  # melt the data set and plot
2  plotdata <- reshape2::melt(naccidents,
3                   id.var="mydate",
4                   measure.var=c("naccidents","pfatal"),
5                   variable.name="Measure",
6                   value.name="value")
7  head(plotdata,n=2)
8  tail(plotdata,n=2)
```

```
> head(plotdata,n=2)
      mydate    Measure value
1 2010-01-01 naccidents   282
2 2010-02-01 naccidents   657
> tail(plotdata,n=2)
        mydate Measure      value
729 2010-10-31  pfatal 0.01092896
730 2010-12-31  pfatal 0.01951220
```

Dates in *R* - Exercise I

```
plotboth <- ggplot(data=plotdata, aes(x=mydate, y=value))+
      ggtitle("P(fatal) and # accidents by date")+
      xlab("Date")+ylab("p(fatal) or # accidents")+
      geom_point()+
      geom_smooth()+
      facet_wrap(~Measure, ncol=1, scales="free_y")
plotboth
```

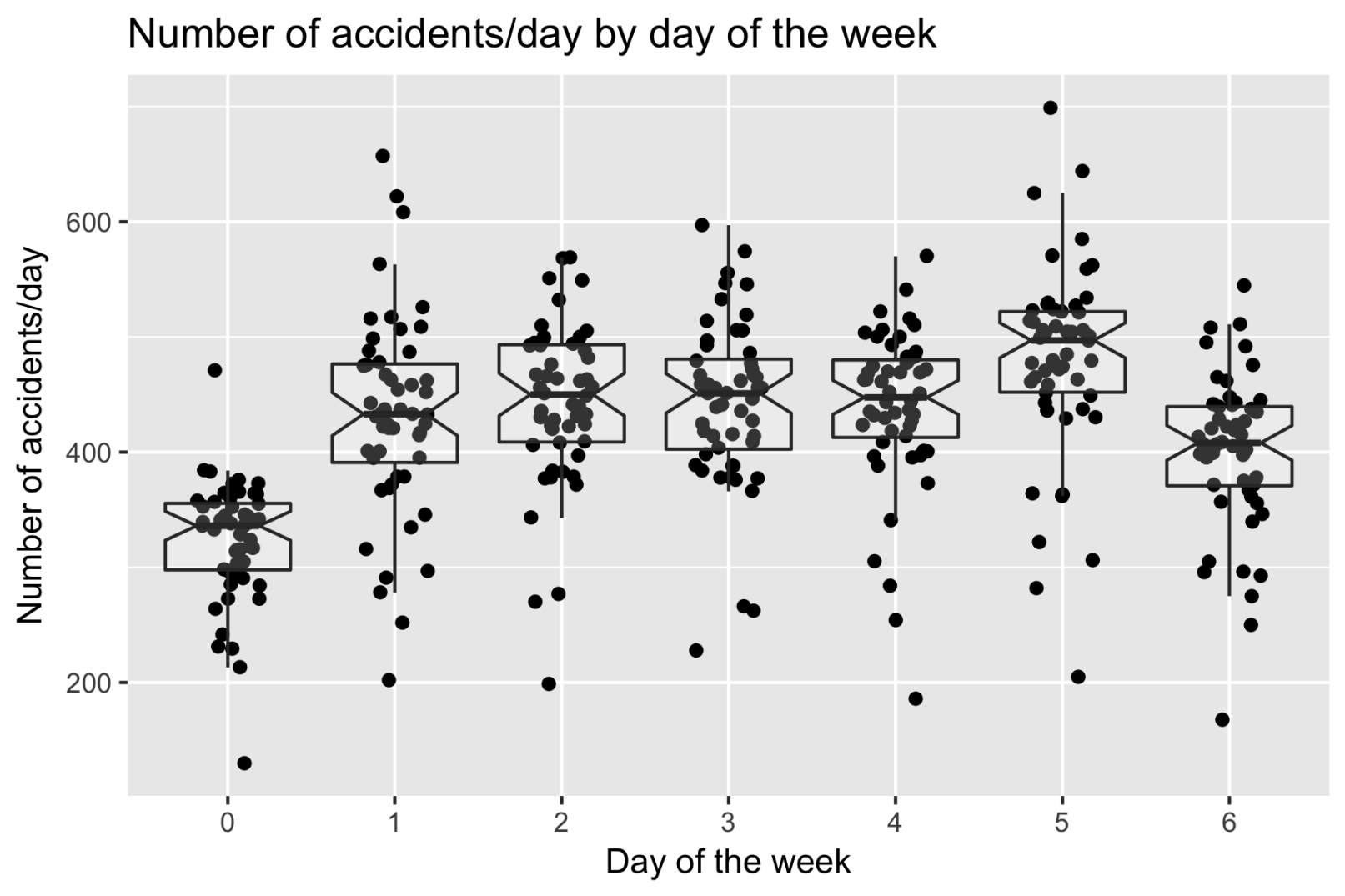
Dates in *R* - Exercise I



Dates in *R* - Exercise I

```
1  naccidents$weekday <- format(naccidents$mydate, format="%w")
2  naccidents[1:10,]
3
4  plotnacc2 <- ggplot(data=naccidents, aes(x=weekday, y=nacci
5    ggtitle("Number of accidents/day by day of the week")+
6    xlab("Day of the week")+ylab("Number of accidents/day")+
7    geom_point(position=position_jitter(w=0.2))+
8    geom_boxplot(notch=TRUE, alpha=0.2, outlier.size=-1, outli
9  plotnacc2
```

Dates in *R* - Exercise I



Dates in *R* - Exercise II

Refer to *road-accidents-2010.csv* file in SampleData.

- Read data into *R*.
- Convert input date to internal *R* dates.
- Find number of accidents by day of year (use *ddply()* and *summarize()* in *plyr* package)
- Plot # accidents/day by day of year.
- Fit a *lowess()* smoother to data using *geom_smooth()*

Look at number of accident by day of the week

- Extract day of the week using *format()* or *weekdays()* functions.
- Use *geom_boxplot()* as seen earlier

Dates in *R* - Exercise II

```
1  # The accident data
2  accidents <- read.csv(file.path(...,'road-accidents-2010.csv
3                header=TRUE,
4                as.is=TRUE, strip.white=TRUE)
5  accidents[1:5,]
6  str(accidents)
```

```
> accidents[1:5,]
.....
  Accident_Severity Number_of_Vehicles Number_of_Casualties
1                 3                  2                    1
2                 3                  1                    1

> str(accidents)
'data.frame': 154414 obs. of  33 variables:
...
 $ Date    : chr  "11/01/2010" "11/01/2010" "12/01/2010" "02/
...
```

Dates in *R* - Exercise II

```
1
2  # Convert date to internal date format
3  accidents$mydate <- as.Date(accidents$Date,
4                format="%d/%m/%Y")
5  sum(is.na(accidents$mydate))
6  accidents[1:5,]
7  str(accidents)
```

```
> accidents[1:5,]
...
  Urban_or_Rural_Area Did_Police_Officer_Attend_Scene_of_Acc
1                   1
2                   1
> str(accidents)
'data.frame': 154414 obs. of  33 variables:
 $ Date         : chr  "11/01/2010" "11/01/2010" "12/01/2010"
 $ mydate     : Date, format: "2010-01-11" "2010-01-11" "2010
>
```

Dates in *R* - Exercise II

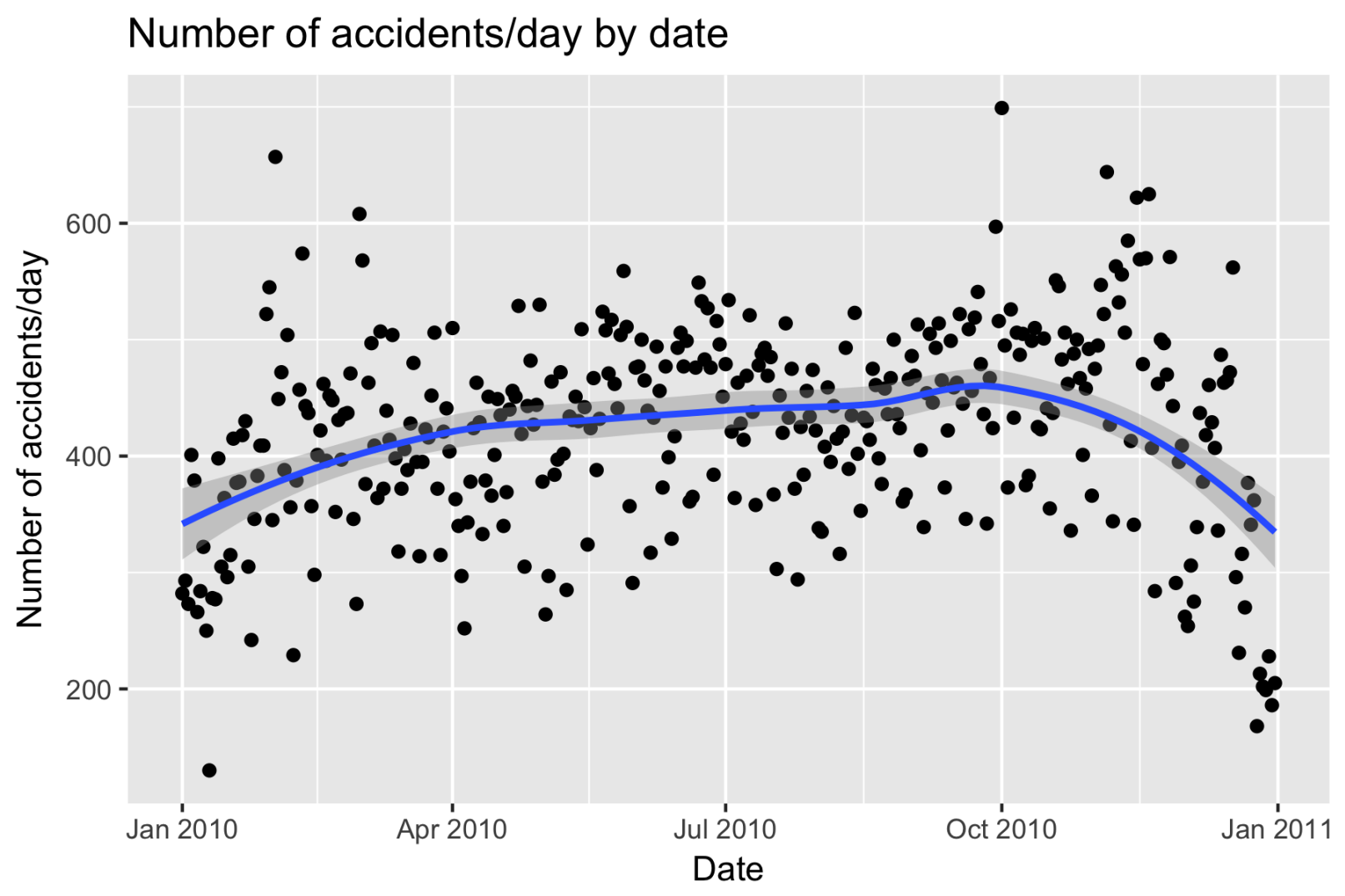
```
1  # Summarize number of accidents by date
2  library(plyr)
3  naccidents <- ddply(accidents, "mydate", summarize,
4                      freq=length(Accident_Index))
5  naccidents[1:5,]
6  str(naccidents)
```

```
> naccidents[1:5,]
      mydate freq
1 2010-01-01  282
2 2010-01-02  293
...
> str(naccidents)
'data.frame': 365 obs. of  2 variables:
 $ mydate: Date, format: "2010-01-01" "2010-01-02" "2010-01-
 $ freq  : int  282 293 273 401 379 266 284 322 250 130 ...
```

Dates in *R* - Exercise II

```
1  plotnacc <- ggplot(data=naccidents, aes(x=mydate, y=freq))+
2        ggtitle("Number of accidents/day by date")+
3        xlab("Date")+ylab("Number of accidents/day")+
4        geom_point()+
5        geom_smooth()
6  plotnacc
```

Dates in *R* - Exercise II



Dates in *R* - Exercise II

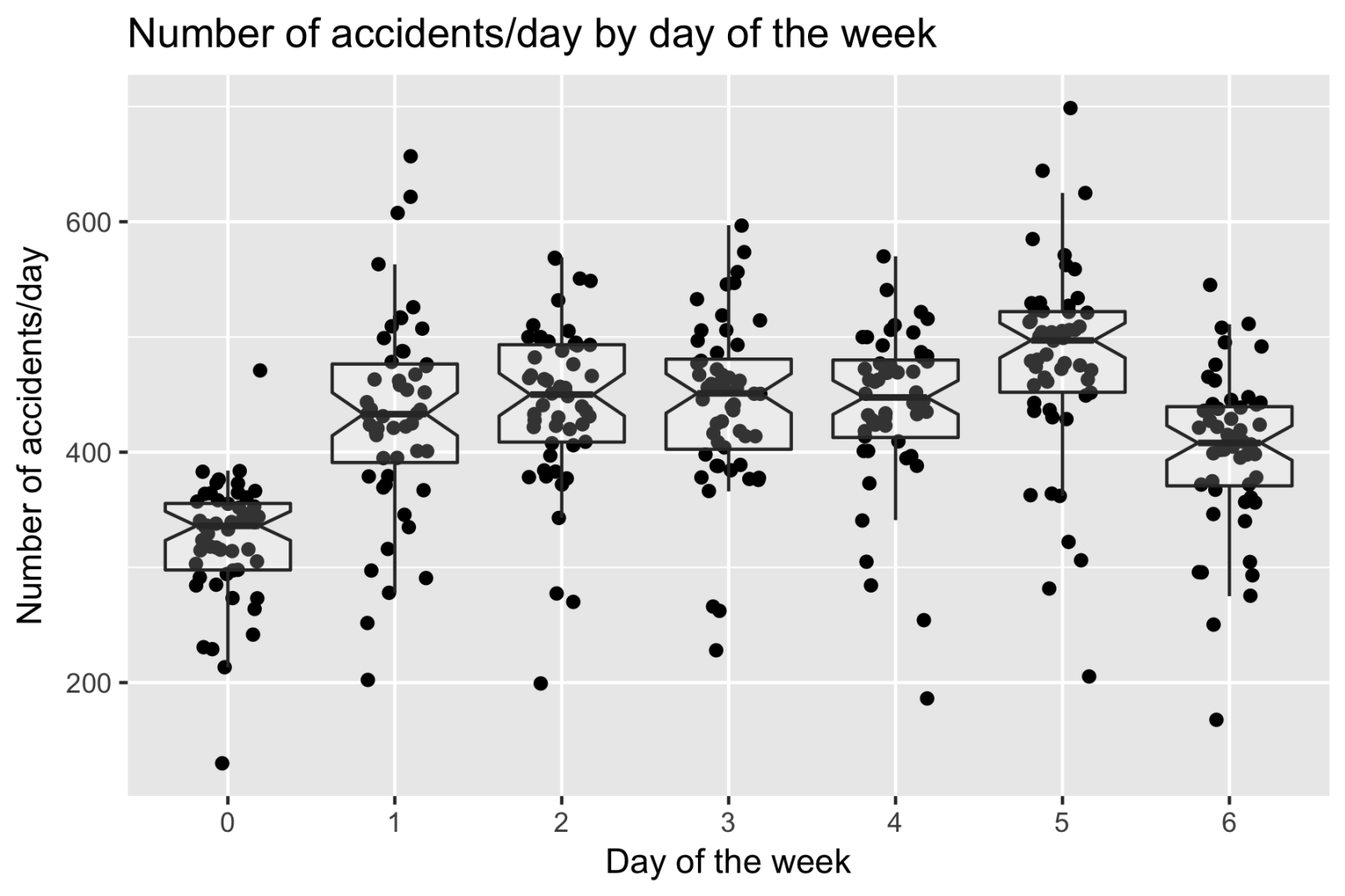
```
1  # Extract day of the week - leave as character values
2  naccidents$weekday <- format(naccidents$mydate, format="%w")
3  naccidents[1:10,]
```

```
> naccidents[1:10,]
        mydate freq weekday
1   2010-01-01  282       5
2   2010-01-02  293       6
3   2010-01-03  273       0
4   2010-01-04  401       1
5   2010-01-05  379       2
...
```

Dates in *R* - Exercise II

```
1  plotnacc2 <- ggplot(data=naccidents, aes(x=dow, y=freq))+
2    ggtitle("Number of accidents/day by day of the week")+
3    xlab("Day of the week")+ylab("Number of accidents/day")+
4    geom_point(position=position_jitter(w=0.2))+
5    geom_boxplot(notch=TRUE, alpha=0.2)
6  plotnacc2
```

Dates in *R* - Exercise II



Dates in *R* - Exercise III

Refer to *road-accidents-2010.csv* file in SampleData.

- Create 0/1 variable if fatality occurs (no or yes; check codebook for *Accident_Severity*).
  Use the magic incantation of *recode()* function in *car* package.
- Find proportion of accidents with fatality by day of year
  - The mean of a 0/1 variable is the proportion.
    Use the magic incantation of *ddply()* and *summarize()* in the *plyr* package.
- Plot proportion of fatalities by day of year.
- Fit a *lowess()* smoother to data from *geom_smooth()*
- Plot proportion of fatalities by day of the week
  - Hint: Extract weekday using *format()*.
  - Hint: Use *geom_boxplot()* as seen earlier with some jittering and notches.

Dates in *R* - Exercise III

```
1  names(accidents)
2  unique(accidents$Accident_Severity)
3  library(car)
4  accidents$Fatality <- recode(accidents$Accident_Severity,
5                ' 1=1; 2:hi=0')
6  accidents[1:5, c("Accident_Severity", "Fatality")]
7  xtabs(~Fatality + Accident_Severity, data=accidents)
```

```
> accidents[1:5, c("Accident_Severity", "Fatality")]
  Accident_Severity Fatality
1                 3        0
2                 3        0
```

```
> xtabs(~Fatality + Accident_Severity, data=accidents)
          Accident_Severity
Fatality      1      2      3
       0      0  20440 132243
       1   1731      0      0
```

Dates in *R* - Exercise III

The *summarize()* and *ddply()* functions in *plyr* package is quite useful for simple summaries by groups.
Example of the Split-Apply-Combine paradigm to be explained later.

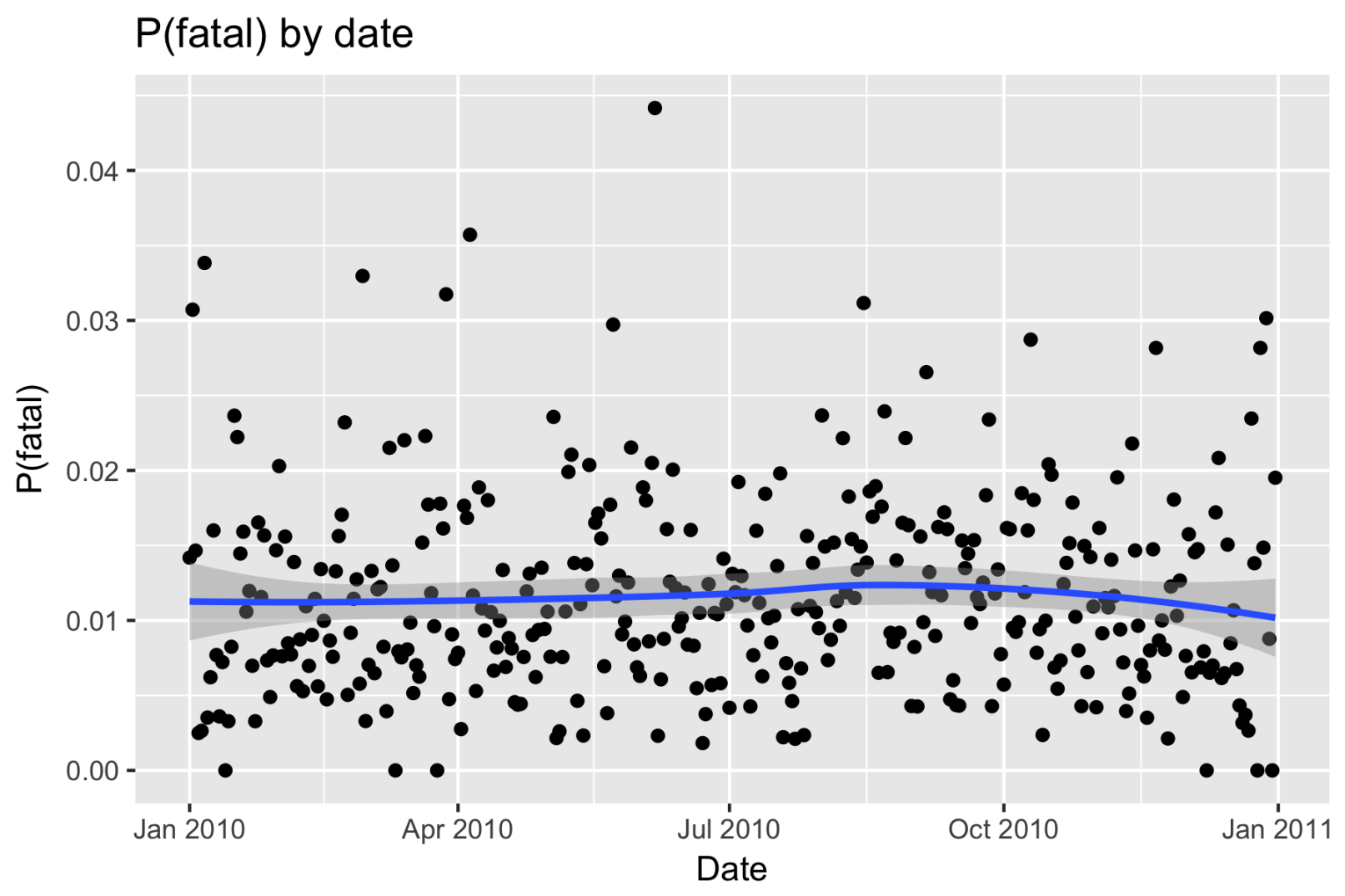
```
1 library(plyr)
2 pfatal.df <- ddply(accidents, "mydate", summarize,
3               freq=length(mydate),
4               pfatal=mean(Fatality))
5 pfatal.df[1:5,]
```

```
> pfatal.df[1:5,]
      mydate freq      pfatal
1 2010-01-01  282 0.014184397
2 2010-01-02  293 0.030716724
3 2010-01-03  273 0.014652015
4 2010-01-04  401 0.002493766
5 2010-01-05  379 0.002638522
```

Dates in *R* - Exercise II

```
plotpfatal <- ggplot(data=pfatal.df,
                     aes(x=mydate, y=pfatal))+
      ggtitle("P(fatal) by date")+
      xlab("Date")+ylab("P(fatal)")+
      geom_point()+
      geom_smooth()
plotpfatal
```

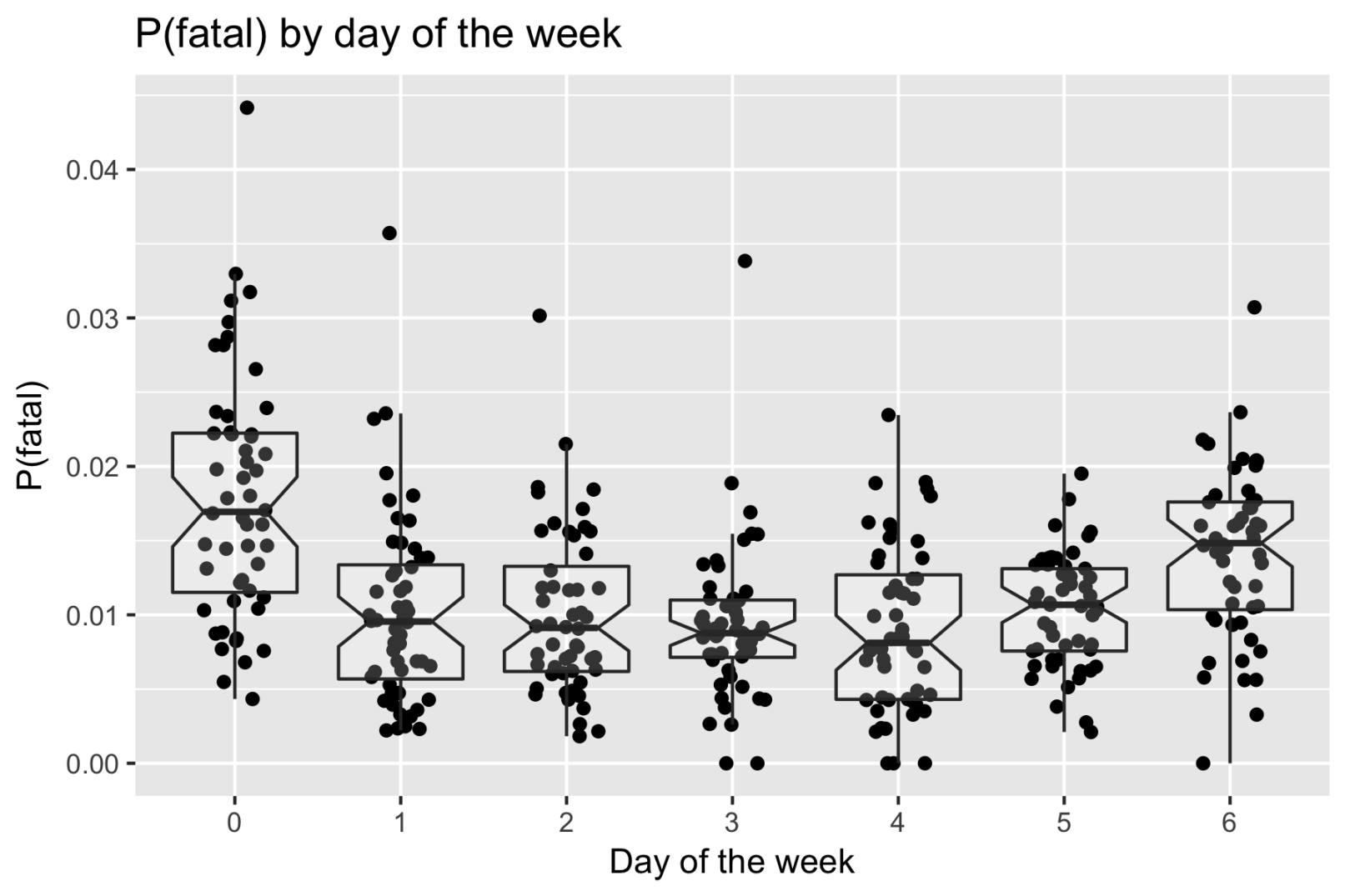
Dates in *R* - Exercise III



Dates in *R* - Exercise II

```
1  # Extract day of the week - leave as character
2  pfatal.df$weekday <- format(pfatal$mydate, format="%w") # l
3  pfatal.df[1:10,]
4
5  plotpfatal2 <- ggplot(data=pfatal.df, aes(x=weekday, y=pfata
6    ggtitle("P(fatal) by day of the week")+
7    xlab("Day of the week")+ylab("P(fatal)")+
8    geom_point(position=position_jitter(w=0.2))+
9    geom_boxplot(notch=TRUE, alpha=0.2)
10 plotpfatal2
```

Dates in *R* - Exercise III



Dates in *R* - Exercise IV

Refer to *flights* data frame in *nycflights13* package.

- Create 0/1 variable if fatality occurs (no or yes; check codebook for *Accident_Severity*).
  Use the magic incantation of *recode()* function in *car* package.
- Find proportion of accidents with fatality by day of year
  - The mean of a 0/1 variable is the proportion.
    Use the magic incantation of *ddply()* and *summarize()* in the *plyr* package.
- Plot proportion of fatalities by day of year.
- Fit a *lowess()* smoother to data from *geom_smooth()*
- Plot proportion of fatalities by day of the week
  - Hint: Extract weekday using *format()*.
  - Hint: Use *geom_boxplot()* as seen earlier with some jittering and notches.

Date+Times (Time Stamps) in *R*

NOT A SIMPLE TASK (not a fault of *R*)

- Local time vs Constant Time? Suggest you work in Constant Time to avoid problems when you change locations of computer.
- Time zone - your machine vs. where collected? Use UTC to avoid these problems.
- Daylight savings time? Always measure in Standard Time if possible.
- Leap seconds?

Dealing with Dates+Times is 3 step process:

- Input DateTime values as a character string (e.g. "2013-10-01 10:23") (*as.is=TRUE* on *read.table()* or *read.csv()*). You may need to *paste()* separate date and time.
- Convert to internal form (number of seconds since origin)
  - Use the *as.POSIXct(string, format=, tz="UTC")* function where the codes are found in *help(strptime)*
  - Allows arithmetic in the usual fashion
- Convert to display format (default is yyyy-mm-dd hh:mm:ss)

Create the textConnection and then read in the following data.

```
testDateTimes <- textConnection("
dt1c,   dt2c
2013-10-23 10:23, 2012-07-13 1:23:00
2013-11-23 25:23, 2012-07-14 3:23:00
2013-12-23 13:23, 2012-07-15 5:23:10
2013-12-24 10:62, 2013-07-15 7:23:23 ") # example of inputi

my.dt <- read.csv(testDateTimes, header=TRUE,
                  as.is=TRUE, strip.white=TRUE)
my.dt
str(my.dt)
```

At this point, all of the variables are CHARACTER data type.

Date+Times in *R*

Convert to internal format using codes from *help(strptime)*

```
1  # Convert from character to internal DateTime representatio
2  my.dt$dt1 <- as.POSIXct(my.dt$dt1c,
3                  format="%Y-%m-%d %H:%M", tz="UTC")
4  my.dt$dt1
5  as.numeric(my.dt$dt1)
6  str(my.dt1)
```

```
> my.dt$dt1
[1] "2013-10-23 10:23:00 UTC" NA    "2013-12-23 13:23:00 UTC
[4] NA
> as.numeric(my.dt$dt1)
[1] 1382523780         NA 1387804980         NA
> str(my.dt)
'data.frame': 4 obs. of  5 variables:
 $ dt1c: chr  "2013-10-23 10:23" "2013-11-23 25:23" "2013-12
 $ dt2c: chr  "2012-07-13 1:23:00" "2012-07-14 3:23:00" "201
 $ dt1 : POSIXct, format: "2013-10-23 10:23:00" NA "2013-12-
 $ dt2 : POSIXct, format: "2012-07-13 01:23:00" "2012-07-14
```

Convert to internal format using codes from *help(strptime)*

```
1  # Look what happens if you don't specify a time zone
2  my.dt$dt1b <- as.POSIXct(my.dt$dt1c,
3                           format="%Y-%m-%d %H:%M")
4  my.dt$dt1b
5  as.numeric(my.dt$dt1b)
6  str(my.dt)
```

```
> my.dt$dt1b
[1] "2013-10-23 10:23:00 PDT" NA   "2013-12-23 13:23:00 PST"
[4] NA
```

It only know the time zone from where your machine is currently located.

Arithmetic and sequence operations are allowed

```
# read in the the other data values
my.dt$dt2 <- as.POSIXct(my.dt$dt2c,
                format="%Y-%m-%d %H:%M", tz="UTC")
my.dt$dt2
as.numeric(my.dt$dt2)

# Arithmetic aoperations allowed
mean(my.dt$dt2)
as.numeric(mean(my.dt$dt2))

# sequences of dates etc
seq(from=my.dt$dt2[1], by="3 weeks", length.out=3)
```

Date+Times in *R*

Arithmetic and sequence operations are allowed

```
> my.dt$dt2
[1] "2012-07-13 01:23:00 UTC" "2012-07-14 03:23:00 UTC" "201
[4] "2013-07-15 07:23:00 UTC"
> as.numeric(my.dt$dt2)
[1] 1342142580 1342236180 1342329780 1373872980
>
> # Arithmetic operations allowed
> mean(my.dt$dt2)
[1] "2012-10-13 16:23:00 UTC"
> as.numeric(mean(my.dt$dt2))
[1] 1350145380
>
> # sequences of dates etc
> seq(from=my.dt$dt2[1], by="3 weeks", length.out=3)
[1] "2012-07-13 01:23:00 UTC" "2012-08-03 01:23:00 UTC" "201
>
```

Extracting parts of DateTime using codes from *help(strptime)* or *lubridate* package functions.

```
1  # extract the various bits from the date-time format
2  as.numeric(format(my.dt$dt2, "%d"))
3  as.numeric(format(my.dt$dt2, "%H"))
```

```
> as.numeric(format(my.dt$dt2, "%d"))
[1] 13 14 15 15
> as.numeric(format(my.dt$dt2, "%H"))
[1] 1 3 5 7
```

Convert to internal format using codes from *help(strptime)*
CAUTION - beware of changes due to Daylight savings times.

```
# Beware of daylight saving changes
# In BC, this occurred at 2013-11-03 at 2:00 am
# Compare the behaviour of
mytime <- as.POSIXct("2013-11-03 01:57:00", tz="UTC")
mytime
seq(mytime, by='1 min', length.out=10)
as.numeric(seq(mytime, by='1 min', length.out=10))

mytime <- as.POSIXct("2013-11-03 01:57:00")
mytime
seq(mytime, by='1 min', length.out=10)
as.numeric(seq(mytime, by='1 min', length.out=10))
```

Date+Times in *R*

Daylight savings time problems.

```
> mytime
[1] "2013-11-03 01:57:00 UTC"
> seq(mytime, by='1 min', length.out=10)
 [1] "2013-11-03 01:57:00 UTC" "2013-11-03 01:58:00 UTC" "20
 [4] "2013-11-03 02:00:00 UTC" "2013-11-03 02:01:00 UTC" "20
 [7] "2013-11-03 02:03:00 UTC" "2013-11-03 02:04:00 UTC" "20
[10] "2013-11-03 02:06:00 UTC"

> mytime
[1] "2013-11-03 01:57:00 PDT"
> seq(mytime, by='1 min', length.out=10)
 [1] "2013-11-03 01:57:00 PDT" "2013-11-03 01:58:00 PDT" "20
 [4] "2013-11-03 01:00:00 PST" "2013-11-03 01:01:00 PST" "20
 [7] "2013-11-03 01:03:00 PST" "2013-11-03 01:04:00 PST" "20
[10] "2013-11-03 01:06:00 PST"
```

CAUTION - beware of changes due to Daylight savings times.
In BC, this occurred at 2013-11-03 at 2:00 am

```
1  mytime <- as.POSIXct("2013-11-03 01:57:00", tz="UTC")
2  as.numeric(format(seq(mytime, by='1 min', length.out=10),
3                    "%H"))
4
5  mytime <- as.POSIXct("2013-11-03 01:57:00")
6  as.numeric(format(seq(mytime, by='1 min', length.out=10),
7                     "%H"))
```

Daylight savings time problems

```
> as.numeric(format(seq(mytime, by='1 min', length.out=10),
 [1] 1 1 1 2 2 2 2 2 2 2
>
> mytime <- as.POSIXct("2013-11-03 01:57:00")
> as.numeric(format(seq(mytime, by='1 min', length.out=10),
 [1] 1 1 1 1 1 1 1 1 1 1
```

Useful functions from lubridate

Useful functions from *lubridate* package.

- *arrive <- ymd_hms("2011-06-04 12:00:00", tz = "Pacific/Auckland")*
- *second('"2011-06-04 12:01:02")*
- *interval(dt1, dt2)* - time intervals

If you must deal with time zone

CAUTION: Heavy Lifting Required!

- Use IANA time zone codes, e.g. Australia, Canada, US all have EST, so need to use "America/New_York", "Pacific/Auckland" etc.
- Not all cities follow a consistent time zone pattern e.g. "America/New_York" had a different pattern than "America/Detroit"
- Refer to `http://www.iana.org/time-zones` for complete data base.
- *OlsonNames()* returns current time zone recognized by *R*.
- *lubridate* always uses "UTC"

Dates+Times in *R* - Exercise I

Refer to *road-accidents-2010.csv* file in SampleData.

- Read data into *R*.
- Combine Date and Time values into a DateTime value
- Convert the DateTime character string into internal format
  - Hint: use *paste()* to put Date and Time together
  - Hint: check for missing values of the converted DateTime values. Why did these occur?
- Find the minute of the accident and draw a histogram - explanation?
  - Hint: use the *binwidth=* option of *geom_histogram()* to get individual bar by minute
- Find proportion of fatalities by hour of the day and plot - explanation?

Dates+Times in *R* - Exercise I

```
1  accidents <- read.csv(file.path(...,'road-accidents-2010.csv
2                 as.is=TRUE, strip.white=TRUE)
3  accidents$DateTime <- paste(accidents$Date, accidents$Time)
4  accidents[1:5,c("Date","Time","DateTime")]
```

Dates+Times in *R* - Exercise I

```
> accidents[1:5,c("Date","Time","DateTime")]
       Date  Time          DateTime
1 11/01/2010 07:30 11/01/2010 07:30
2 11/01/2010 18:35 11/01/2010 18:35
3 12/01/2010 10:22 12/01/2010 10:22
4 02/01/2010 21:21 02/01/2010 21:21
5 04/01/2010 20:35 04/01/2010 20:35
```

Dates+Times in *R* - Exercise I

```
accidents$mydt <- as.POSIXct(accidents$DateTime,
                  format="%d/%m/%Y %H:%M", tz="UTC")
accidents[1:5,c("Date","Time","DateTime","mydt")]
str(accidents)
```

Dates+Times in *R* - Exercise I

```
> accidents[1:5,c("Date","Time","DateTime","mydt")]
        Date  Time          DateTime                 mydt
1 11/01/2010 07:30 11/01/2010 07:30 2010-01-11 07:30:00
2 11/01/2010 18:35 11/01/2010 18:35 2010-01-11 18:35:00
3 12/01/2010 10:22 12/01/2010 10:22 2010-01-12 10:22:00
4 02/01/2010 21:21 02/01/2010 21:21 2010-01-02 21:21:00
5 04/01/2010 20:35 04/01/2010 20:35 2010-01-04 20:35:00
> str(accidents)
'data.frame': 154414 obs. of  34 variables:
 $ Date     : chr  "11/01/2010" "11/01/2010" "12/01/2010" "0
 $ Time     : chr  "07:30" "18:35" "10:22" "21:21" ...
 $ DateTime   : chr  "11/01/2010 07:30" "11/01/2010 18:35" "
 $ mydt   : POSIXct, format: "2010-01-11 07:30:00" "2010-01-1
```

Dates+Times in *R* - Exercise I

```
1  sum(is.na(accidents$mydt))
2  accidents[is.na(accidents$mydt),c("Date","Time","mydt")]
```

```
> sum(is.na(accidents$mydt))
[1] 8
> accidents[is.na(accidents$mydt),c("Accident_Index","Date"
       Accident_Index       Date Time mydt
144656  201091NP08355 24/07/2010      <NA>
144823  2010921000796 17/02/2010      <NA>
145079  2010921002018 20/05/2010      <NA>
145082  2010921002035 20/05/2010      <NA>
145561  2010921004092 20/10/2010      <NA>
149625  2010961001411 26/07/2010      <NA>
149755  2010961001975 14/10/2010      <NA>
149780  2010961002066 25/10/2010      <NA>
```

Dates+Times in *R* - Exercise I

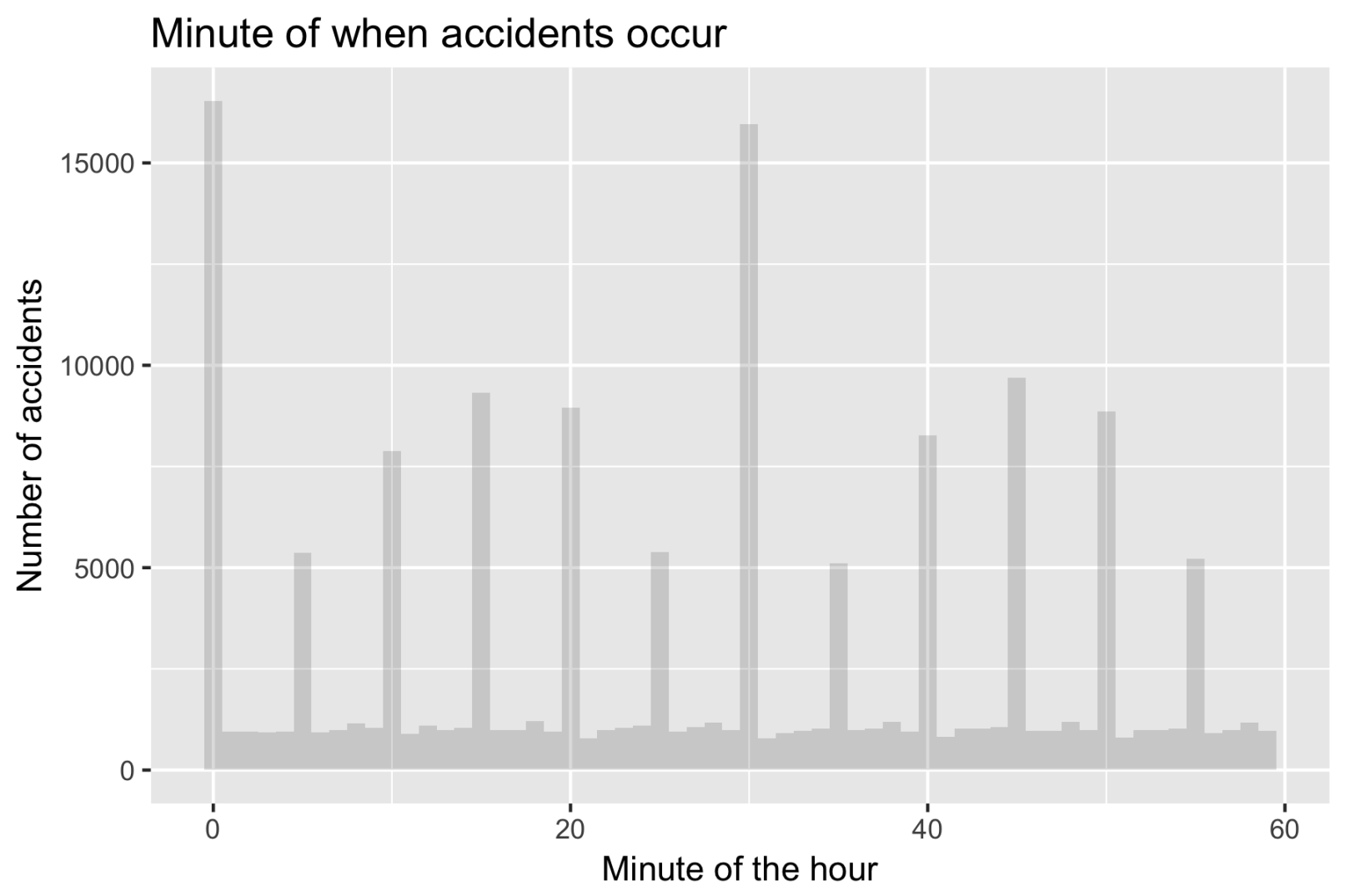
```
1  # Extract the minute of the accident
2  accidents$min <- as.numeric(format(accidents$mydt, "%M"))
3  accidents[1:5, c("Date","Time","mydt","min")]
```

Dates+Times in *R* - Exercise I

```
> accidents[1:5, c("Date","Time","mydt","min")]
       Date  Time                mydt min
1 11/01/2010 07:30 2010-01-11 07:30:00  30
2 11/01/2010 18:35 2010-01-11 18:35:00  35
3 12/01/2010 10:22 2010-01-12 10:22:00  22
4 02/01/2010 21:21 2010-01-02 21:21:00  21
5 04/01/2010 20:35 2010-01-04 20:35:00  35
```

Dates+Times in *R* - Exercise I

```
1  plotaccmin <- ggplot(data=accidents, aes(x=min))+
2    ggtitle("Minute of when accidents occur")+
3    xlab("Minute of the hour")+ylab("Number of accidents")+
4    geom_histogram(binwidth=1, alpha=0.2)
5  plotaccmin
```


Minute of when accidents occur
Number of accidents
15000
10000
5000
0
0
20
40
60
Minute of the hour

Find the proportion of fatalities by hour of the day.

```
accidents$Fatality <- accidents$Accident_Severity==1
xtabs(~Fatality + Accident_Severity, data=accidents)

accidents$hour <- as.numeric(format(
          accidents$mydt, "%H"))
accidents[1:5, c("Date","Time","mydt","Fatality","hour")]
```

Dates+Times in *R* - Exercise I

```
> accidents[1:5, c("Date","Time","mydt","Fatality","hour")]
       Date  Time                mydt Fatality hour
1 11/01/2010 07:30 2010-01-11 07:30:00    FALSE    7
2 11/01/2010 18:35 2010-01-11 18:35:00    FALSE   18
3 12/01/2010 10:22 2010-01-12 10:22:00    FALSE   10
4 02/01/2010 21:21 2010-01-02 21:21:00    FALSE   21
5 04/01/2010 20:35 2010-01-04 20:35:00    FALSE   20
```

Dates+Times in *R* - Exercise I

Find the proportion of fatalities by hour of the day.

```
1 library(plyr)
2 pfatal.df <- ddply(accidents, "Hour", summarize,
3                   pfatal=mean(Fatality))(
4 pfatal.df[1:5,]
```

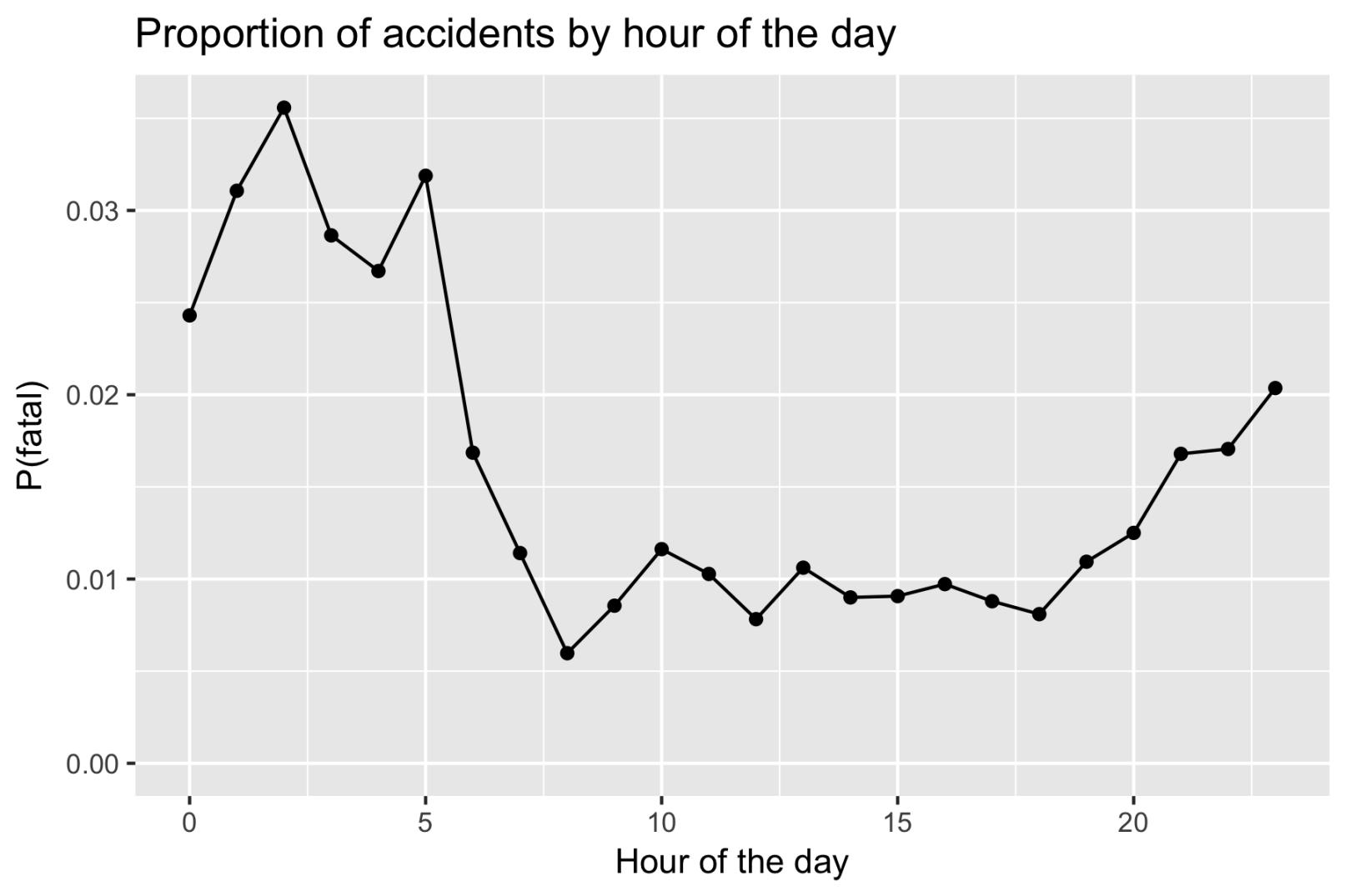
Dates+Times in *R* - Exercise I

```
> pfatal.df[1:5,]
  hour      pfatal
1    0 0.02430402
2    1 0.03106682
3    2 0.03557618
4    3 0.02864583
5    4 0.02671312
```

Dates+Times in *R* - Exercise I

```
1  plotpfatalhour <- ggplot(data=pfatal.df, aes(x=hour, y=pfata
2    ggtitle("Proportion of accidents by hour of the day")+
3    xlab("Hour of the day")+ylab("P(fatal)")+
4    geom_point()+
5    geom_line()
6  plotpfatalhour
```

Dates+Times in *R* - Exercise I



Dates+Times in *R* - Exercise II

Refer to flight data.

- Read data into *R*.
- Combine Date and Time values into a DateTime value
- Convert the DateTime character string into internal format
- Histogram of number of departures by hour of the day
- Histogram number of departures by minutes of the hour?
- Distribution of actual

Durations in *R*

- Lengths of time not connected to a particular date.
- Key issues are reading in duration data (65:13) and converting to seconds.
- *lubridate* package with duration data
  - *hm()*, *ms()*, *hms()* functions available.
    - CAUTION: Does 10:30 mean 10 minutes and 30 second, or 10 hours and 30 minutues?
  - Longer durations involving days, hours, minutes, seconds more difficult as no standard representation, i.e. what does 01:12:13:23 represent?
- Many helper functions in *lubridate* package to create durations, e.g. *ddays(2)* creates a duration object of 2 days (or 2*24*3600 seconds).

Durations in *R*

Dealing with Durations only is 2 step process but is straightforward.

- Input Duration values as a character string (e.g. “25:13”) (use the as.is=TRUE option on *read.table()* or *read.csv()*)
- Convert to internal form (number of seconds) using functions in *lubridate*
  - (Internal) negative values indicate "negative" durations (?)
  - Allows arithmetic but some care is needed to keep as duration values.

Durations in *R*

Create the textConnection and then read in the following data.

```
1  library(lubridate)
2  testDuration.csv <- textConnection("
3  du1c,   du2c
4  10:23, 1:23:00
5  25:23, 3:23:00
6  13:23, 5:23:10
7  10:62, 7:23:23 ") # example of inputing data inline
8
9  my.du <- read.csv(testDuration.csv, header=TRUE,  # notice
10              as.is=TRUE, strip.white=TRUE)
11 my.du
12 str(my.du)
```

At this point, all of the variables are CHARACTER data type.

Durations in *R* - Converting to internal format

Convert to internal format using functions from *lubridate*

```
1  # First duration is ambiguous
2  my.du$du1a <- ms(my.du$du1c)
3  my.du$du1a
4  as.numeric(my.du$du1a)
5
6  my.du$du1b <- hm(my.du$du1c)
7  my.du$du1b
8  as.numeric(my.du$du1b)
9
10 my.du$du2 <- hms(my.du$du2c)
11 my.du$du2
12 as.numeric(my.du$du2)
13
14 mean(my.du$du1a)   #???????
15 mean(as.duration(my.du$du1a))
```

Once converted to seconds, standard arithmetic (e.g. means can be computed) but need to specify that as.duration()

Durations in *R* - Converting to internal format

Convert to internal format using functions from *lubridate*

```
> my.du$du1a <- ms(my.du$du1c)
> my.du$du1a
[1] "10M 23S" "25M 23S" "13M 23S" "10M 62S"
> as.numeric(my.du$du1a)
[1]  623 1523  803  662

> my.du$du1b <- hm(my.du$du1c)
> my.du$du1b
[1] "10H 23M 0S" "25H 23M 0S" "13H 23M 0S" "10H 62M 0S"
> as.numeric(my.du$du1b)
[1] 37380 91380 48180 39720
```

Durations in *R* - Converting to internal format

Convert to internal format using functions from *lubridate*

```
> my.du$du2 <- hms(my.du$du2c)
> my.du$du2
[1] "1H 23M 0S"  "3H 23M 0S"  "5H 23M 10S" "7H 23M 23S"
> as.numeric(my.du$du2)
[1]  4980 12180 19390 26603

> mean(my.du$du1a)  #???????
[1] 32.75
> mean(as.duration(my.du$du1a))
[1] 902.75
```

Once converted to seconds, standard arithmetic (e.g. means can be computed) but need to specify that as.duration()

Durations in *R* - Exercise 1

Refer to flight data.

- Read data into *R*.
- Convert departure and arrive delay to duration data
- Histogram of departure and arrival delays.
- Departure delays related to hour of arrival or other variables?

Time of Day in *R*

- Time of day NOT connected to a particular date.
- Key issues are reading in tod data (hh:mm:ss) and converting to seconds.
- *hms* package with tod data
  - *parse_hms()* function available to convert from character form.
    - CAUTION: Does 10:30 mean 0:10:30, or 10:30:00?
  - Differences are odd (in which direction do you measure)?
  - Averages are odd (in which direction do you measure)?
- Refer to *hms* package for more details.

Time of Day in *R*

Dealing with TOD only is 2 step process but is straightforward.

- Input TOD values as a character string (e.g. "14:13") (use the as.is=TRUE option on *read.table()* or *read.csv()*)
- Convert to internal form (number of seconds) using functions in *hms*

Create the textConnection and then read in the following data.

```
1  library(hms)
2  testTod.csv <- textConnection("
3  tod1c,   tod2c
4  10:23, 1:23:00
5  25:23, 3:23:00
6  13:23, 5:23:10
7  10:62, 7:23:23 ") # example of inputing data inline
8
9  my.tod <- read.csv(testTod.csv, header=TRUE,  # notice no q
10                    as.is=TRUE, strip.white=TRUE)
11 my.tod
12 str(my.tod)
```

At this point, all of the variables are CHARACTER data type.

Time of Day in *R* - Converting to internal format

Convert to internal format using functions from *lubridate*

```
1  # convert to time of day
2  my.tod$tod1 <- hms::parse_hm(my.tod$tod1c)
3  my.tod$tod2 <- hms::parse_hm(my.tod$tod2c)
4  my.tod
```

```
> my.tod
  tod1c   tod2c     tod1     tod2
1 10:23 1:23:00 10:23:00 01:23:00
2 25:23 3:23:00       NA 03:23:00
3 13:23 5:23:10 13:23:00 05:23:00
4 10:62 7:23:23       NA 07:23:00
```

Durations in *R* - Converting to internal format

Convert to internal format using functions from *lubridate*

```
> # does averaging work? properly
> as.hms(mean(my.tod$tod2))
04:23:00
> as.hms(mean( c(hms::parse_hm("23:50"), hms::parse_hm("00:2
12:05:00
>
> my.tod$tod2[1]-my.tod$tod2[4]
Time difference of -21600 secs
> my.tod$tod2[4]-my.tod$tod2[1]
Time difference of 21600 secs
> hms::parse_hm("23:50") - hms::parse_hm("00:20")
Time difference of 84600 secs
> hms::parse_hm("00:20") - hms::parse_hm("23:50")
Time difference of -84600 secs
```

Often not clear how to do arithmetic on these values, but refer to the *circadian.mean()* in the *psych* package.

Dates+Times in *R* - Summary

- Date vs. DateTime vs. Duration vs. TOD vs. Periods
- *as.Date()* vs. *as.POSIXct()*
- Days since origin vs. Seconds since origin
- Careful of time zones/ daylight savings times/ etc
- Use *lubridate* package with pure time data or with duration data (e.g. 65:30 as 65 hours and 30 minutes)
- Use *lubridate* package with period data, e.g. 1 month doesn't have a standard length; what is 31 January + 1 month; does 31 January + 1 month = 28 February - 1 month?