Lab 1 2020-01-24 V1.01 - Exercise answers

Biomedical Data Science

Question 1

Minimum river length and corresponding index:

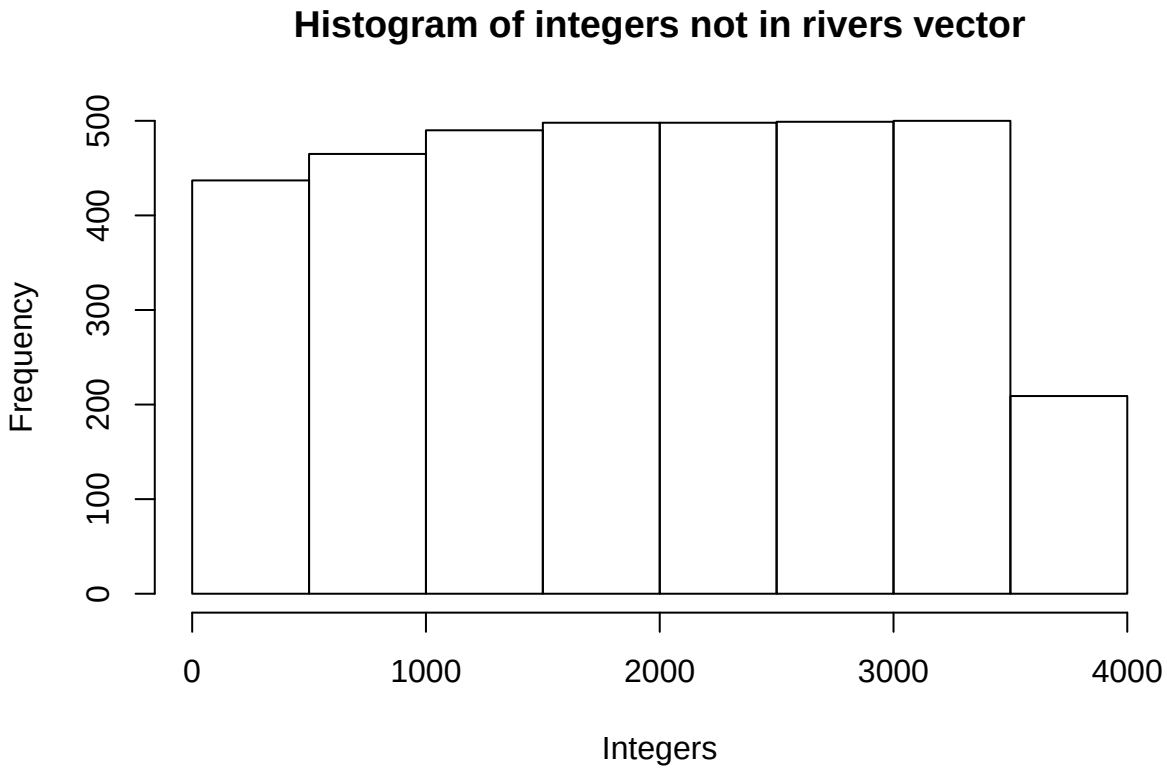
```
> min(rivers)
[1] 135
> which(rivers == min(rivers))     # same as which.min(rivers)
[1] 8
```

Maximum river length and corresponding index:

```
> max(rivers)
[1] 3710
> which(rivers == max(rivers))     # same as which.max(rivers)
[1] 68
```

Histogram of integers that do not appear in the `rivers` vector:

```
> ints.not.in.rivers <- setdiff(1:max(rivers), rivers)
> hist(ints.not.in.rivers, main="Histogram of integers not in rivers vector",
+      xlab="Integers")
```

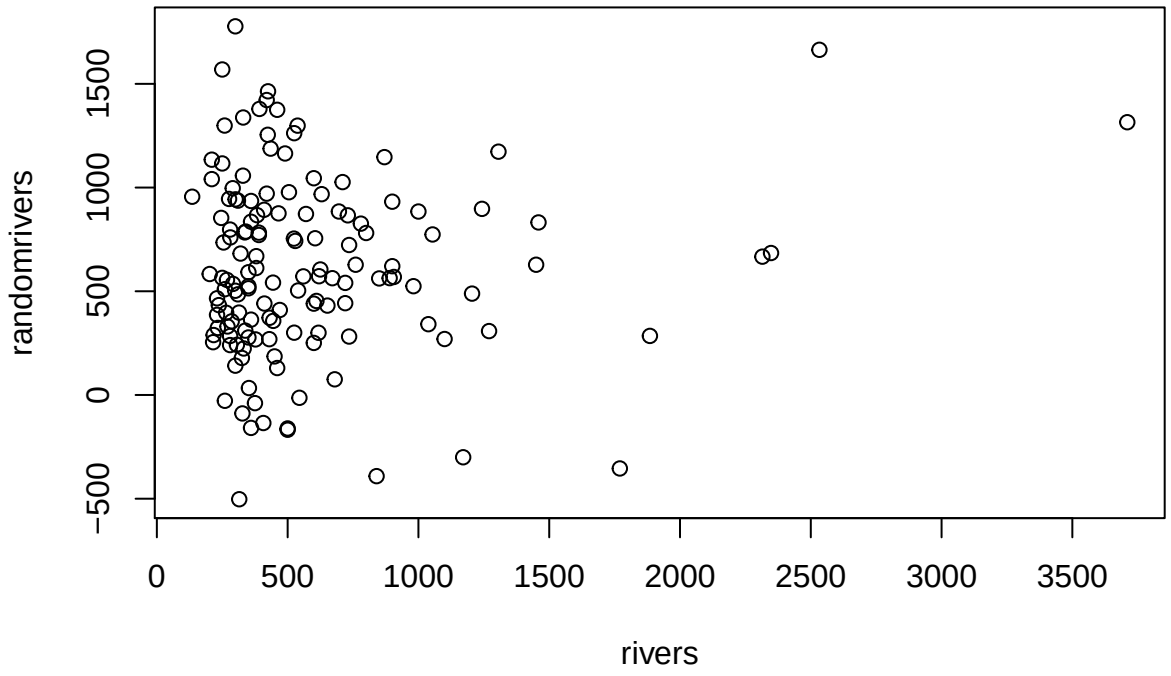


Creation of the `randomrivers` vector:

```
> n <- length(rivers)
> mu <- mean(rivers)
> ss <- sd(rivers)
> set.seed(1)       # set the random seed before generating the random numbers
> randomrivers <- rnorm(n, mu, ss)
> sum(randomrivers < 0)
[1] 12
> sum(randomrivers > 2 * rivers)
[1] 39
```

Scatter plot and correlation coefficient:

```
> plot(rivers, randomrivers)
```

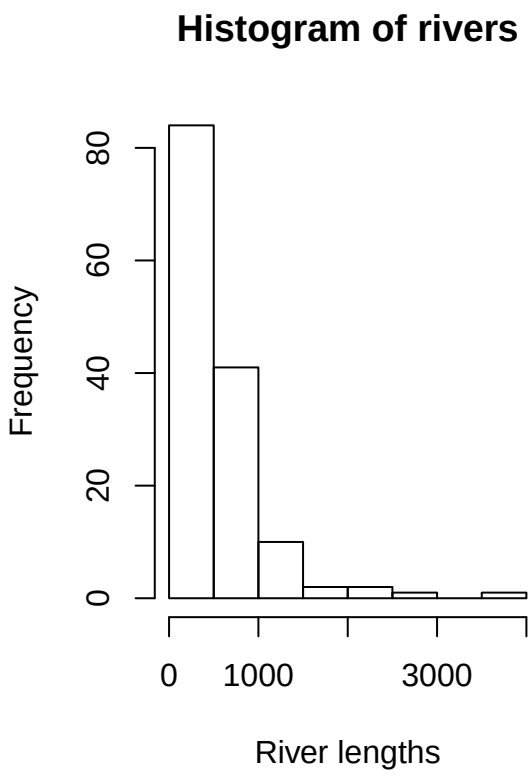


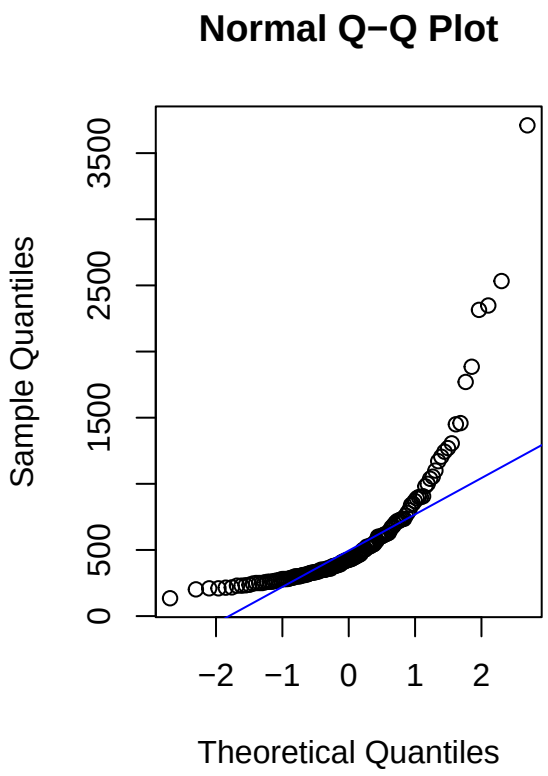

```
> signif(cor(rivers, randomrivers), 3)
[1] 0.0892
```

Given the way the points of `rivers` and `randomrivers` are scattered in the plot, and knowing that `randomrivers` is randomly distributed, we can say that `rivers` is not randomly distributed.

The simpler and more appropriate way of stating that would have been to produce either a histogram or a QQ plot of rivers. The histogram shows that the distribution of `rivers` is right skewed. The QQ plot shows that the quantiles of `rivers` (sample quantiles) do not match the quantiles of a random variable (theoretical quantiles): if they did, the points would lie closely to the blue line.

```
> par(mfrow=c(1, 2))    # two plots in the same row
> hist(rivers, xlab="River lengths")
> qqnorm(rivers)
> qqline(rivers, col="blue")
```





Question 2

Names and sizes of islands with area in the first quantile:

```
> first.quartile.idx <- which(islands < quantile(islands, 0.25))
> islands[first.quartile.idx]
    Axel Heiberg           Hainan           Kyushu         Melville
              16               13               14               16
     New Britain  Prince of Wales      Southampton      Spitsbergen
              15               13               16               15
          Taiwan Tierra del Fuego            Timor        Vancouver
              14               19               13               12
> median(islands[first.quartile.idx])
[1] 14.5
```

Size of the 10th largest and 10th smallest:

```
> sort(islands, decreasing=TRUE)[10]
Borneo
   280
> sort(islands)[10]
Melville
      16
```

Number of islands with an odd area:

```
> sum(islands %% 2 == 1)
[1] 21
```

Islands with area divisible by 3:

```
> islands3 <- islands[islands %% 3 == 0]
> median(islands3)
[1] 36
> quantile(islands3, c(0.25, 0.75))
  25%   75%
 25.5 244.5
```

Smallest area in `islands3` that is also in `rivers`:

```
> min(intersect(islands3, rivers))
[1] 306
```

Mean area of `island3` not in `rivers`:

```
> round(mean(islands3[!islands3 %in% rivers]), 1)
[1] 1283.5
```

Note that the result above is different to the one obtained by using `setdiff()`:

```
> round(mean(setdiff(islands3, rivers)), 1)
[1] 1512.8
```

This is due to the fact that `setdiff()` implicitly discards any duplicated values, as it treats the vectors as sets. Compare the following two vectors:

```
> islands3[!islands3 %in% rivers]  # two elements with value 30, two with value 15
      Britain         Devon    Hispaniola      Hokkaido       Ireland
           84            21            30            30            33
        Luzon      Mindanao   New Britain North America South America
           42            36            15          9390          6795
  Spitsbergen       Sumatra     Vancouver
           15           183            12
> setdiff(islands3, rivers)        # one element with value 30, one with value 15
 [1]   84   21   30   33   42   36   15 9390 6795  183   12
```

Question 3

Median, range and interquartile range of the "rating" variable

```
> mean(attitude$rating)
[1] 64.63333
> range(attitude$rating)
[1] 40 85
> quantile(attitude$rating, c(0.25, 0.75))
  25%   75%
58.75 71.75
```

Median rating for observations that have above median values for the "raises" variable:

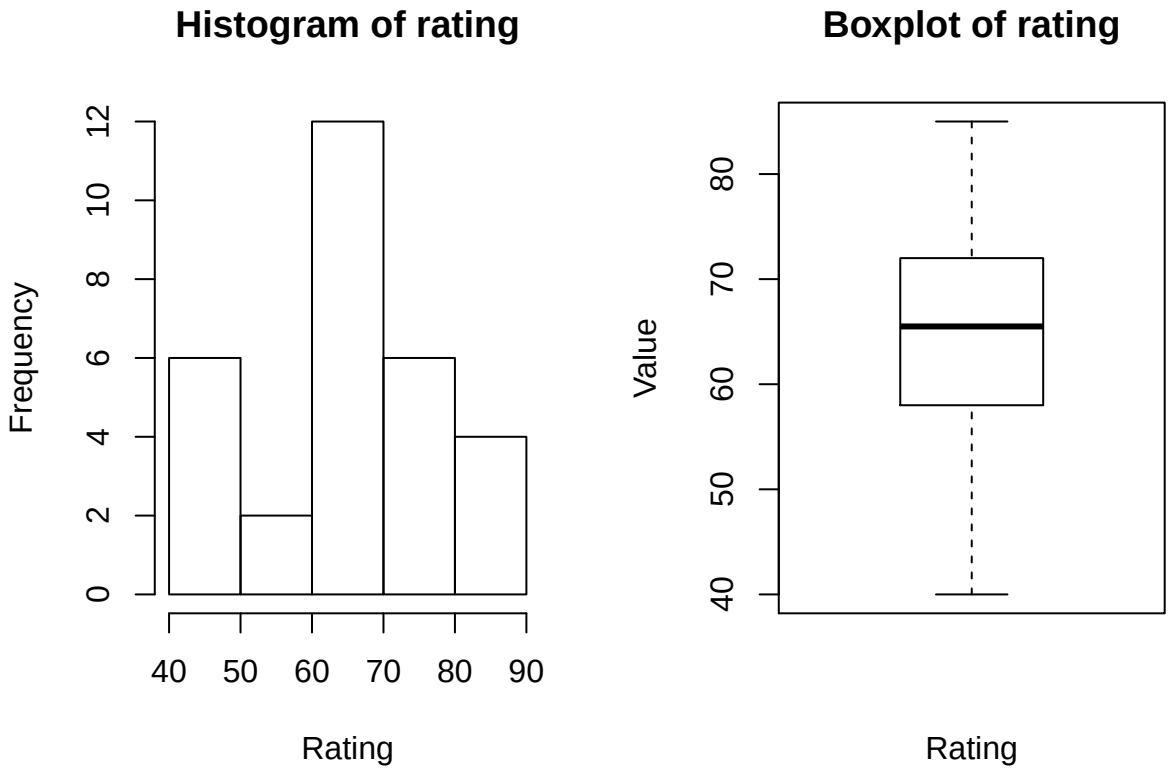
```
> ## option 1
> median(attitude$rating[attitude$raises > median(attitude$raises)])
[1] 71
> ## option 2
> with(attitude, median(rating[raises > median(raises)]))
[1] 71
```

Standard deviation of "advance" before and after removing extreme values:

```
> with(attitude, sd(advance))       # same as sd(attitude$advance)
[1] 10.28871
> with(attitude, sd(advance[advance > min(advance) &
+                           advance < max(advance)]))
[1] 8.386418
```

Histogram and boxplot:

```
> par(mfrow=c(1,2))
> hist(attitude$rating, main="Histogram of rating", xlab="Rating")
> boxplot(attitude$rating, main="Boxplot of rating", xlab="Rating", ylab="Value")
```

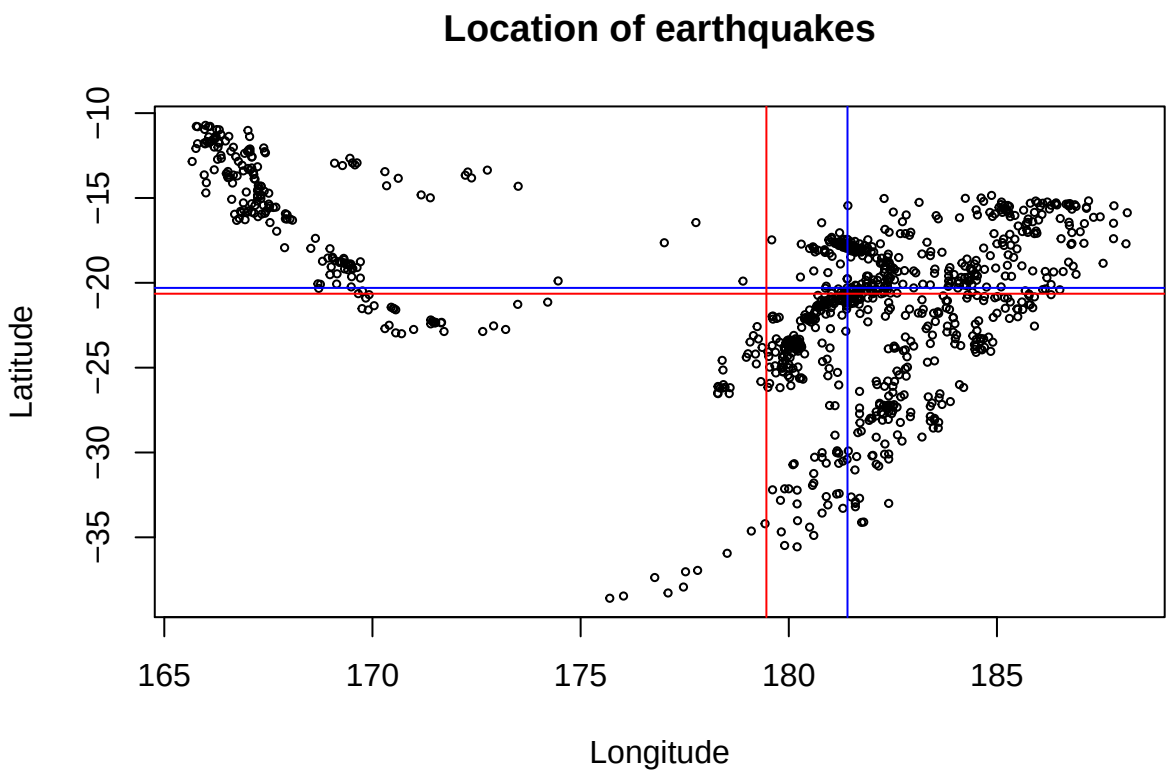



```
> hist(attitude$complaints, main="Histogram of complaints", xlab="Complaints")
> boxplot(attitude$complaints, main="Boxplot of complaints",
+         xlab="Complaints", ylab="Value")
> # and so on for all other variables
```

Question 4

Scatter plot of longitude and latitude:

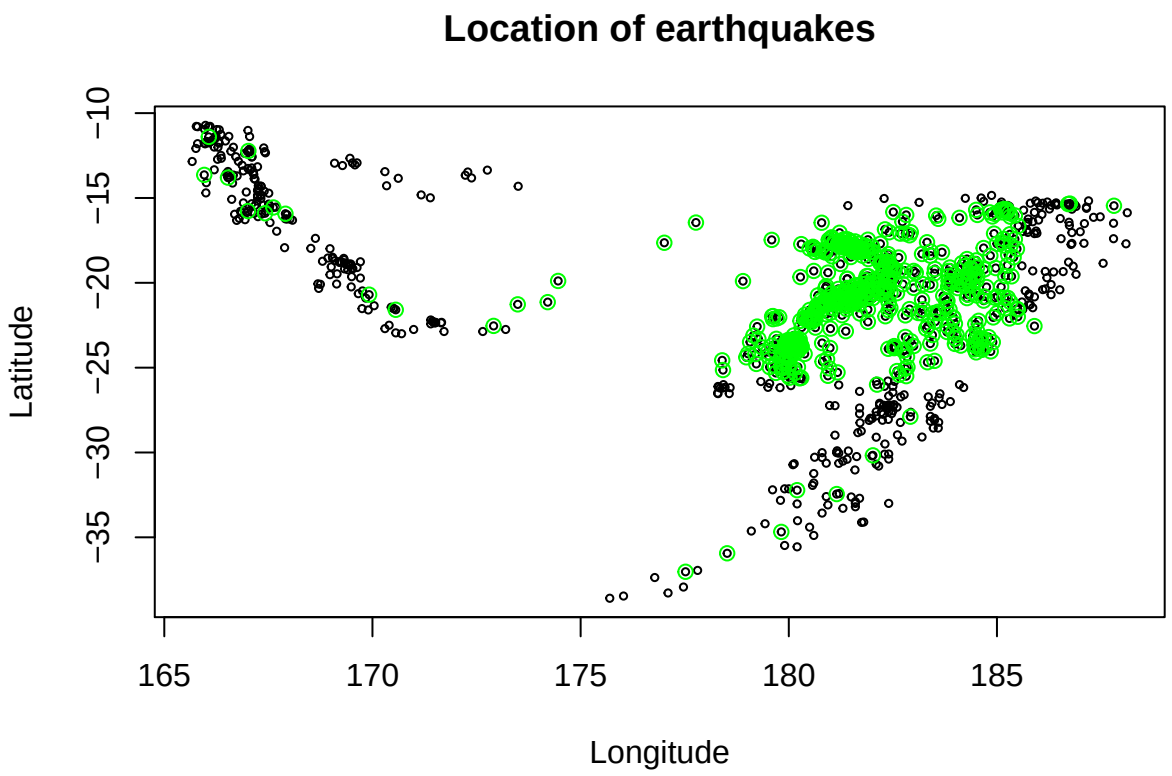
```
> with(quakes, plot(long, lat, main="Location of earthquakes",
+                   xlab="Longitude", ylab="Latitude", cex=0.5))
> abline(h=median(quakes$lat), col="blue")   # horizontal line
> abline(v=median(quakes$long), col="blue")  # vertical line
> abline(h=mean(quakes$lat), col="red")
> abline(v=mean(quakes$long), col="red")
```



Note how the coordinates of the means (in red) correspond to a location where no earthquakes were reported, while the coordinates of the medians (in blue) do.

Creation of `quakes.1sd`:

```
> long.mu <- mean(quakes$long)
> long.sd <- sd(quakes$long)
> lat.mu <- mean(quakes$lat)
> lat.sd <- sd(quakes$lat)
> quakes.1sd <- subset(quakes, (long < long.mu + 1 * long.sd &
+                               long > long.mu - 1 * long.sd &
+                               lat < lat.mu + 1 * lat.sd &
+                               lat > lat.mu - 1 * lat.sd) |
+                               mag >= 5.5)
> # cex=0.5 is specified to draw a smaller circle
> with(quakes, plot(long, lat, main="Location of earthquakes",
+                   xlab="Longitude", ylab="Latitude", cex=0.5))
> with(quakes.1sd, points(long, lat, col="green"))
```



Addition of "damage":

```
> quakes.1sd$damage <- with(quakes.1sd,
+                           sqrt(max(depth) - depth) + 5 * mag + stations^0.25)
> nrow(quakes.1sd)
[1] 585
> round(range(quakes.1sd$damage), 2)
[1] 23.17 58.84
> all.corrs <- cor(quakes.1sd)    # correlations between all variables
> all.corrs[, "damage"]
       lat       long      depth        mag   stations     damage
-0.1396115  0.1562822 -0.9327817  0.5631355  0.4802890  1.0000000
```

Creation of quakes.40s:

```
> quakes.40s <- subset(quakes, stations > 40)
> sum(nchar(rownames(quakes.40s)) == 3)      # points with row names of length 3
[1] 243
> length(grep("7", rownames(quakes.40s)))    # points containing character '7'
[1] 77
>
> # points containing 9 but not in the first position
> length(setdiff(grep("9", rownames(quakes.40s)),
+                grep("^9", rownames(quakes.40s))))
[1] 44
> # alternative way
> length(grep("[0-8].*9", rownames(quakes.40s)))
[1] 44
```

Question 5

Take the `nottem` time series data and convert to a data table. The first part of the code takes the nottem time series variables `temp` and `time` adding them to a data table. The result of that data table is then `chained` to another to split the time data into month and year columns. Note the use of the `format()` funtion to convert into other time formats. The temp is added back as-is. This results in the data in a long format.

```
> library(lubridate)
> library(data.table)
>
> nottem.dt <- data.table(temp = c(nottem), year = (c(time(nottem))))[,
+       .(month=format(date_decimal(year+.01), "%b"),
+       Year=format(date_decimal(year+.01), "%Y"), temp)]
```

The `reshape()` function from the lubridate package is then used to convert the data into a wide format. The `reshape()` command creates column names as a concatenated string of the data column and timevar. Eg. `temp.Jan` for the first temperature column. The gsub command is used to replace the first part of the string with an empty string, thus removing the prefix and returning a data table that looks exactly like the time series data.

```
> nottem.dt <- reshape(nottem.dt, idvar = "Year", timevar = "month", direction = "wide")
> colnames(nottem.dt) <- gsub("temp.", "", colnames(nottem.dt))
```

Now we create a summer.avg column and assign by reference the result of taking the mean of `.SD` (subset of data table) with columns defined by `.SDcols`. The columns defined by `.SDcols` are the columns named from Jun to Sep. The mean is applied to each row using `by = .I` which is the row index. `rowMeans` is used to calculate the mean across the Row. Finally, we caluclate the the sum of `summer.avg`.

```
> nottem.dt[, summer.avg := rowMeans(.SD), by = .I,
+               .SDcols=colnames(nottem.dt[, Jun:Sep])]
> sum(nottem.dt$summer.avg)
[1] 1184.7
```

Create the new variable and assign by reference the mean of the month column. Group the column by the left three characters of the year string. To show the first year of each decade, convert the year to a numeric type and select where the modulus of 10 is 0.

```
> nottem.dt[, jun.decade.avg := mean(Jun), by=substr(Year, 0, 3)]
> nottem.dt[as.numeric(Year)%%10 == 0, .(Year, jun.decade.avg)]
   Year jun.decade.avg
1: 1920          56.99
2: 1930          59.09
```