# ADS2 Practical 6: Getting and cleaning Data. SOLUTION

Dmytro Shytikov and Xushen Xiong

2023-11-02

In this practical session, you will work on two real-world datasets that are modified to suit our practical purpose. You could work through this guide alone or in groups. Facilitators are here to help. The time it takes to complete this practical can vary between individuals - this is OK. Do not worry if you do not finish within the session.

## Learning Objectives

- Getting data in R
- Use different data types and data structures in R
- Explain the advantages of tidy datasets
- Clean a real-world dataset according to tidy principles

**Help tips:**

Any functions that you want to know more about/better, use `? Function_name`. Many useful libraries in R make data cleaning much more efficient and easier.

## Practice 1. West Nile Virus (WNV) Mosquito Test Background

### Getting the data

Input the `WNV_mosquito_test_results.csv` using `read.csv ()`.

This part must be straightforward. Remember to set the working directory `setwd()` and specify the path to the file.

```
wnv <- read.csv("Week_7_WNV_mosquito_test_results.csv" )
```

### Screen and diagnosis.

- Is this a long or wide form?
- Are the variable names informative and precise? (**Hint**: The name of the first variable `SEASON.YEAR` is not accurate. Please change it to `YEAR`.)
- Are there missing values? `anyNA()`?
- Are there duplicated values?
- Are there any strange patterns? You will need to make different diagnostic plots to spot any strange patterns. Try to be creative!

```r
head(wnv, 10)
```

```
##    SEASON.YEAR TEST.ID                  BLOCK TRAP TRAP_TYPE
## 1         2019   49933    62XX N MCCLELLAN AVE T236    GRAVID
## 2         2019   49952       17XX N PULASKI RD T039    GRAVID
## 3         2019   49966      11XX W CHICAGO AVE T049    GRAVID
## 4         2019   49984          63XX W 64TH ST T155    GRAVID
## 5         2019   50009          17XX W 95TH ST T094    GRAVID
## 6         2019   49929       71XX N HARLEM AVE T233    GRAVID
## 7         2019   49918     41XX N OAK PARK AVE T002    GRAVID
## 8         2019   49998 64XX S STONY ISLAND AVE T077    GRAVID
## 9         2019   49936       58XX N PULASKI RD T027    GRAVID
## 10        2019   49995      39XX S ASHLAND AVE T074    GRAVID
##              TEST.DATE NUMBER.OF.MOSQUITOES               SPECIES
## 1  09/26/2019 12:09:00                    3         CULEX RESTUANS
## 2  09/26/2019 12:09:00                    2 CULEX PIPIENS/RESTUANS
## 3  09/26/2019 12:09:00                   12 CULEX PIPIENS/RESTUANS
## 4  09/26/2019 12:09:00                    4 CULEX PIPIENS/RESTUANS
## 5  09/26/2019 12:09:00                    6 CULEX PIPIENS/RESTUANS
## 6  09/26/2019 12:09:00                   23 CULEX PIPIENS/RESTUANS
## 7  09/26/2019 12:09:00                   35 CULEX PIPIENS/RESTUANS
## 8  09/26/2019 12:09:00                   34 CULEX PIPIENS/RESTUANS
## 9  09/26/2019 12:09:00                    8 CULEX PIPIENS/RESTUANS
## 10 09/26/2019 12:09:00                   11 CULEX PIPIENS/RESTUANS
##                                  LOCATION
## 1   (41.99496630402897, -87.77083721987879)
## 2   (41.91356758228873, -87.72630030176042)
## 3  (41.896131092623506, -87.65676212387862)
## 4   (41.77600539167921, -87.77940766760916)
## 5   (41.72128749967918, -87.66523570170051)
## 6    (42.0106432736568, -87.80679730045945)
## 7  (41.956298856118664, -87.79751744482932)
## 8  (41.778128857884745, -87.58624503516381)
## 9  (41.986319851449004, -87.72837845617912)
## 10  (41.82085850772701, -87.66510809467968)
```

```r
summary(wnv)
```

```
##   SEASON.YEAR      TEST.ID         BLOCK               TRAP
##  Min.   :2007   Min.   :20000   Length:29489       Length:29489
##  1st Qu.:2009   1st Qu.:27718   Class :character   Class :character
##  Median :2012   Median :35150   Mode  :character   Mode  :character
##  Mean   :2013   Mean   :35156
##  3rd Qu.:2016   3rd Qu.:42641
##  Max.   :2019   Max.   :50029
##   TRAP_TYPE          TEST.DATE         NUMBER.OF.MOSQUITOES   SPECIES
##  Length:29489       Length:29489       Min.   : 1.00        Length:29489
##  Class :character   Class :character   1st Qu.: 2.00        Class :character
##  Mode  :character   Mode  :character   Median : 5.00        Mode  :character
##                                        Mean   :12.35
##                                        3rd Qu.:16.00
##                                        Max.   :77.00
```

```
##    LOCATION
##  Length:29489
##  Class :character
##  Mode  :character
##
##
##
```

```
str(wnv)
```

```
## 'data.frame':    29489 obs. of  9 variables:
##  $ SEASON.YEAR         : int  2019 2019 2019 2019 2019 2019 2019 2019 2019 2019 ...
##  $ TEST.ID             : int  49933 49952 49966 49984 50009 49929 49918 49998 49936 49995 ...
##  $ BLOCK               : chr  "62XX N MCCLELLAN AVE" "17XX N PULASKI RD" "11XX W CHICAGO AVE" "63XX W
##  $ TRAP                : chr  "T236" "T039" "T049" "T155" ...
##  $ TRAP_TYPE           : chr  "GRAVID" "GRAVID" "GRAVID" "GRAVID" ...
##  $ TEST.DATE           : chr  "09/26/2019 12:09:00" "09/26/2019 12:09:00" "09/26/2019 12:09:00" "09/1
##  $ NUMBER.OF.MOSQUITOES: int  3 2 12 4 6 23 35 34 8 11 ...
##  $ SPECIES             : chr  "CULEX RESTUANS" "CULEX PIPIENS/RESTUANS" "CULEX PIPIENS/RESTUANS" "CU
##  $ LOCATION            : chr  "(41.99496630402897, -87.77083721987879)" "(41.91356758228873, -87.7263
```

```
class(wnv)
```

```
## [1] "data.frame"
```

```
class(wnv$TEST.DATE)
```

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

From this step, you can see that your data frame follows a wide format and has a generally appropriate structure. Anyway, there are the following issues:

- The `SEASON.YEAR` column would be more correct if it was called just `YEAR`.
- The `TEST.DATE` column is encoded as a character variable. Thus, it cannot be analysed as a **date**.
- The `LOCATION` variable is encoded wrongly including the latitude and longitude at the same time.
- We need to check for the presence of duplicated rows and `NA` values.

### Treat and document

Our plan is:

- Rename the `SEASON.YEAR` column.
- Reformat the data set so all the columns have appropriate types of data. For instance, `TEST.DATE` must be treated as a **date**.
- Separate `LOCATION` to `LONGITUDE` and `LATITUDE`.
- Check for the presence of duplicated rows and `NA` values.

```
anyNA(wnv)
```

```
## [1] FALSE
```

```
# Seems there are no NAs

duplicated(wnv) %>% sum()
```

```
## [1] 0
```

```
# Seems, no duplicated rows.

# Let's check the `TEST.DATE` column
class(wnv$TEST.DATE)
```

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

```
# The format is wrong
```

So, there are no duplicated rows or `NA` (at least, we did not detect them). But the title of the `SEASON.YEAR` column is not exactly correct, and the class of `wnv$TEST.DATE` is also wrong. Let's correct these issues:

```
# Let's make names in our data frame appropriate and make sure that `TEST.DATE`
# is formatted correctly.
wnv <- wnv %>%
  rename(YEAR = SEASON.YEAR) %>%
  relocate(1, 6) %>%
  mutate(TEST.DATE = as.POSIXct(wnv$TEST.DATE,
                                format = "%m/%d/%Y %H:%M:%S",
                                tz="America/Chicago"))
head(wnv, 10)
```

```
##    YEAR           TEST.DATE TEST.ID                   BLOCK TRAP TRAP_TYPE
## 1  2019 2019-09-26 12:09:00   49933    62XX N MCCLELLAN AVE T236    GRAVID
## 2  2019 2019-09-26 12:09:00   49952       17XX N PULASKI RD T039    GRAVID
## 3  2019 2019-09-26 12:09:00   49966      11XX W CHICAGO AVE T049    GRAVID
## 4  2019 2019-09-26 12:09:00   49984          63XX W 64TH ST T155    GRAVID
## 5  2019 2019-09-26 12:09:00   50009          17XX W 95TH ST T094    GRAVID
## 6  2019 2019-09-26 12:09:00   49929       71XX N HARLEM AVE T233    GRAVID
## 7  2019 2019-09-26 12:09:00   49918     41XX N OAK PARK AVE T002    GRAVID
## 8  2019 2019-09-26 12:09:00   49998 64XX S STONY ISLAND AVE T077    GRAVID
## 9  2019 2019-09-26 12:09:00   49936       58XX N PULASKI RD T027    GRAVID
## 10 2019 2019-09-26 12:09:00   49995      39XX S ASHLAND AVE T074    GRAVID
##    NUMBER.OF.MOSQUITOES               SPECIES
## 1                     3         CULEX RESTUANS
## 2                     2 CULEX PIPIENS/RESTUANS
## 3                    12 CULEX PIPIENS/RESTUANS
## 4                     4 CULEX PIPIENS/RESTUANS
## 5                     6 CULEX PIPIENS/RESTUANS
## 6                    23 CULEX PIPIENS/RESTUANS
## 7                    35 CULEX PIPIENS/RESTUANS
## 8                    34 CULEX PIPIENS/RESTUANS
## 9                     8 CULEX PIPIENS/RESTUANS
## 10                   11 CULEX PIPIENS/RESTUANS
##                                  LOCATION
```

```
## 1   (41.99496630402897, -87.77083721987879)
## 2   (41.91356758228873, -87.72630030176042)
## 3  (41.896131092623506, -87.65676212387862)
## 4   (41.77600539167921, -87.77940766760916)
## 5   (41.72128749967918, -87.66523570170051)
## 6    (42.0106432736568, -87.80679730045945)
## 7  (41.956298856118664, -87.79751744482932)
## 8  (41.778128857884745, -87.58624503516381)
## 9  (41.986319851449004, -87.72837845617912)
## 10  (41.82085850772701, -87.66510809467968)
```

We have renamed the `SEASON.YEAR` column to `YEAR` to be more precise and reformatted the `TEST.DATE` column to the `Date`.

OK. Now, our data frame looks better. Let's check the `TEST.DATE` column more precisely and experiment with it. For instance, let's change its attributes.

```
attributes(wnv$TEST.DATE)
```

```
## $class
## [1] "POSIXct" "POSIXt"
##
## $tzone
## [1] "America/Chicago"
```

```
# Let's try to change attributes of some values
dat1 <- wnv$TEST.DATE[1]
attributes(dat1)
```

```
## $class
## [1] "POSIXct" "POSIXt"
##
## $tzone
## [1] "America/Chicago"
```

```
attributes(dat1)$tzone <- "America/Los_Angeles"
```

Now, let's have a closer look at our data frame:

```
# Let's have a glimpse at columns and check for anything interesting
colnames(wnv)

# Let's check all the columns for anything strange:
for(i in c(1, 2, 4:ncol(wnv))){
  writeLines(colnames(wnv)[i])
  print(table(wnv[, i]))
  writeLines("\n")
}

# This code will produce a very long output.  Thus, to save space, I will not evaluate it.
# You can do it by yourself.
```

Looks like we found something interesting:

- NUMBER.OF.MOSQUITOES is mostly relatively low except for one case where the trap captured over 70 mosquitoes at once. Need to check it.
- The LOCATION variable consists of two elements LATITUDE and LONGITUDE. In addition to the poor outlook, the LOCATION column has more than 4400 cells with nothing inside.

Let's treat these issues:

```
# One outlier. Which is it?
wnv[wnv$NUMBER.OF.MOSQUITOES> 70, ]
```

```
##       YEAR           TEST.DATE TEST.ID                 BLOCK TRAP TRAP_TYPE
## 10872 2014 2014-08-07 12:08:00   39090 51XX N MONT CLARE AVE T223    GRAVID
##       NUMBER.OF.MOSQUITOES               SPECIES
## 10872                   77 CULEX PIPIENS/RESTUANS
##                                      LOCATION
## 10872 (41.974522761157274, -87.80458946950488)
```

So far, nothing special. Still, let's check it in more detail:

- Let's check its BLOCK
- Let's check the surrounding rows.

```
wnv_out <- which(wnv$NUMBER.OF.MOSQUITOES> 70)
wnv_out_block <- wnv %>%
  slice(wnv_out) %>%
  select(BLOCK) %>% unlist()

# Let's see values around this block
wnv %>%
  filter(BLOCK == wnv_out_block)

# Let's see values around this row
wnv %>%
  slice((wnv_out-4):(wnv_out+4))

# This code will produce a very long output.  Thus, to save space, I will not evaluate it.
# You can do it by yourself.
```

We checked this unusual case, but did not find anything special about it except the number of mosquitoes caught was unusually high.

What about the LOCATION column? Let's check it:

```
table(wnv$LOCATION) %>% head(3) %>% dimnames()
```

```
## [[1]]
## [1] ""
## [2] "(41.644720066326094, -87.60185152802353)"
## [3] "(41.64831068933974, -87.55963204714429)"
```

The LOCATION column has 4416 cells with "" value inside. Generally, it would be appropriate to check and try to recover these values. For example, you can try to determine the location of these traps by the trap ID. But for the sake of speed, we will just substitute these values with NAs and separate them into LONGITUDE and LATITUDE.

```r
wnv$LOCATION[wnv$LOCATION == ""] <- NA

# As you can see, each value is in brackets. We do not need them.
wnv$LOCATION <- wnv$LOCATION %>%
  gsub(pattern= "[()]", replacement = "", wnv$LOCATION, perl = T)

# Now, we can separate our `LOCATION` column into two more columns
wnv <- separate(data = wnv, col = LOCATION,
                into = c("LATITUDE", "LONGITUDE"),
                sep = ",", remove = F, fill = "left", convert = T)
```

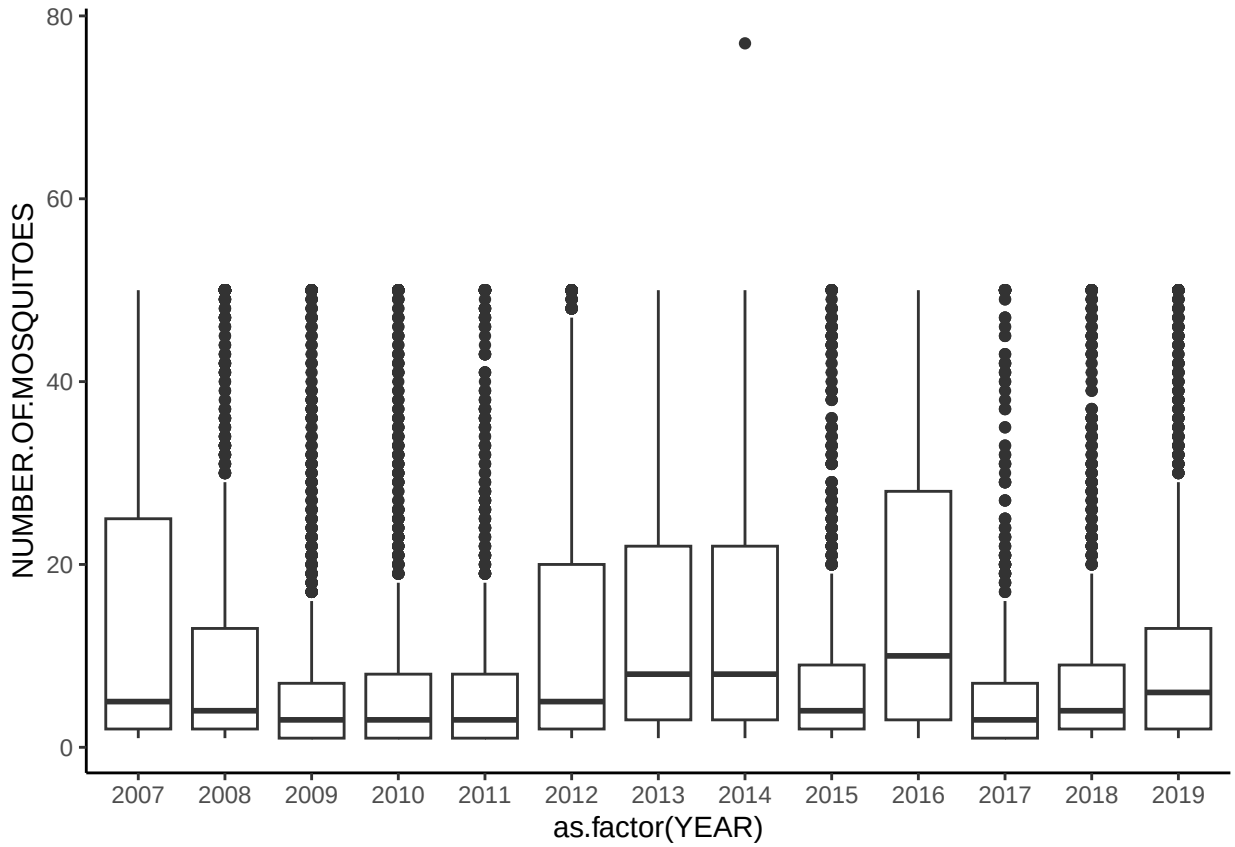
We separated the `LOCATION` column into `LATITUDE` and `LONGITUDE`.

**Summarize your data:**

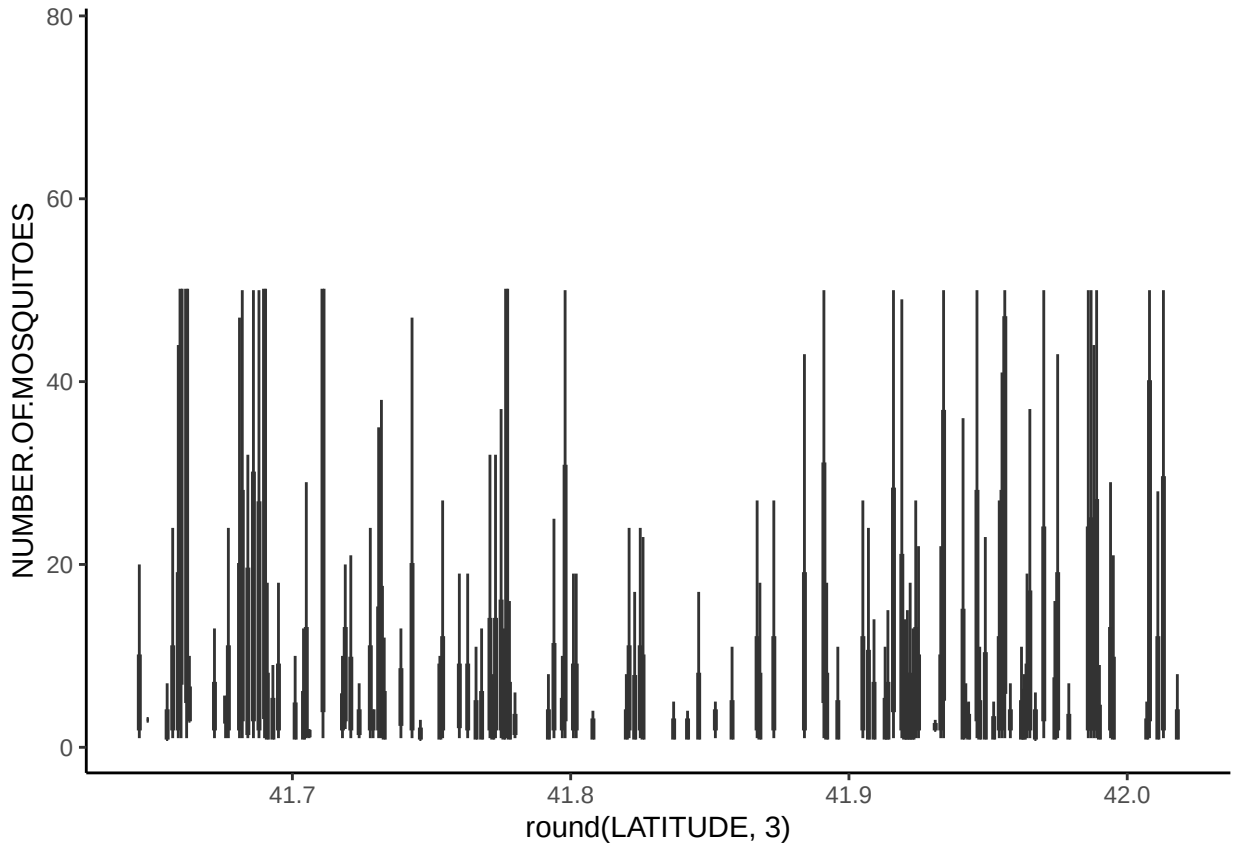
- Is there any relationship between the number of mosquitoes caught and the year?
- Is there any relationship between the location, longitude, or latitude, and the number of mosquitoes caught?
- Some types of traps may catch particular species of mosquitoes more often. Can you see it from the data?
- Formulate any other hypothesis.

Try an appropriate graph type to fit your data best. Use either in-built plotting functions (`graphics` package) or those from `ggplot2`.

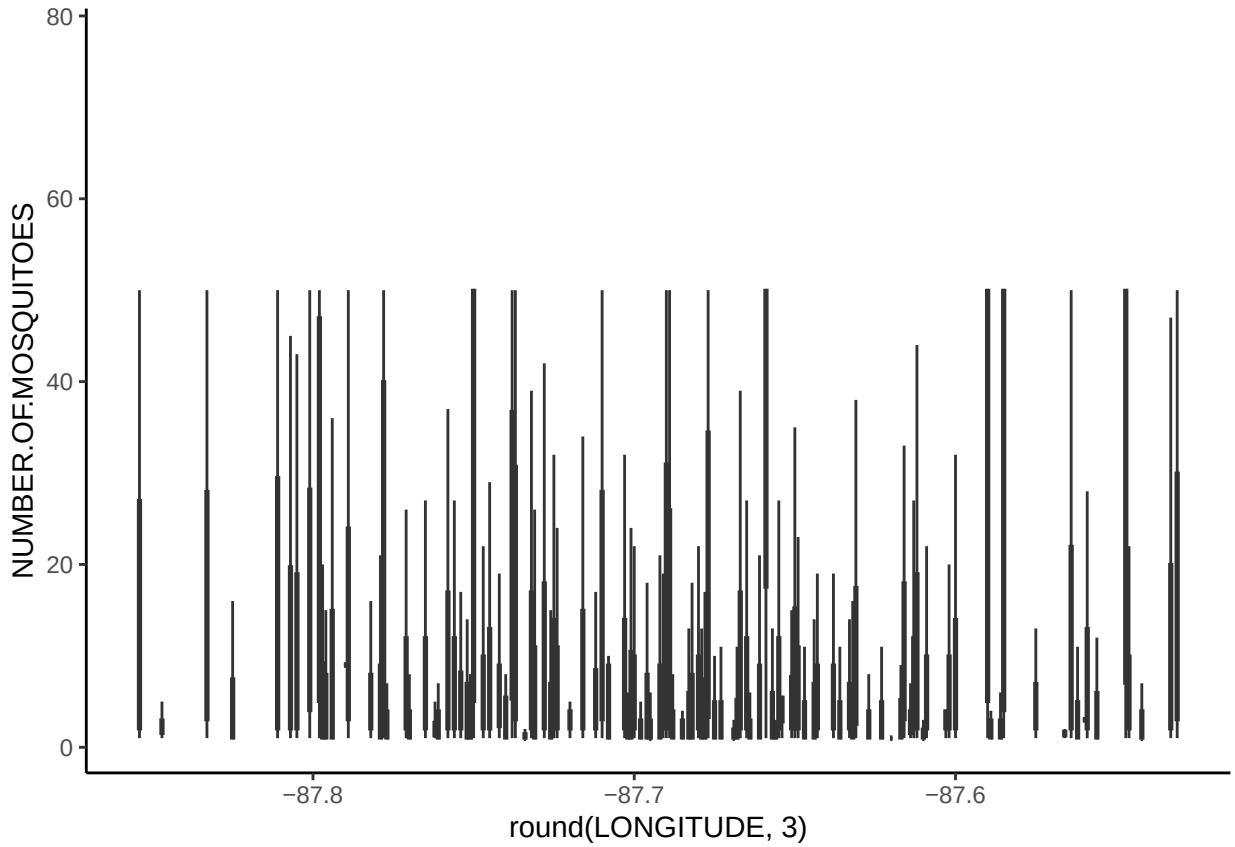
```r
# ggplot2----
wnv_plot <- ggplot(data = drop_na(wnv)) +
  theme_classic()

# By Year
wnv_plot +
  geom_boxplot(mapping = aes(x = as.factor(YEAR),
                             y = NUMBER.OF.MOSQUITOES))
```



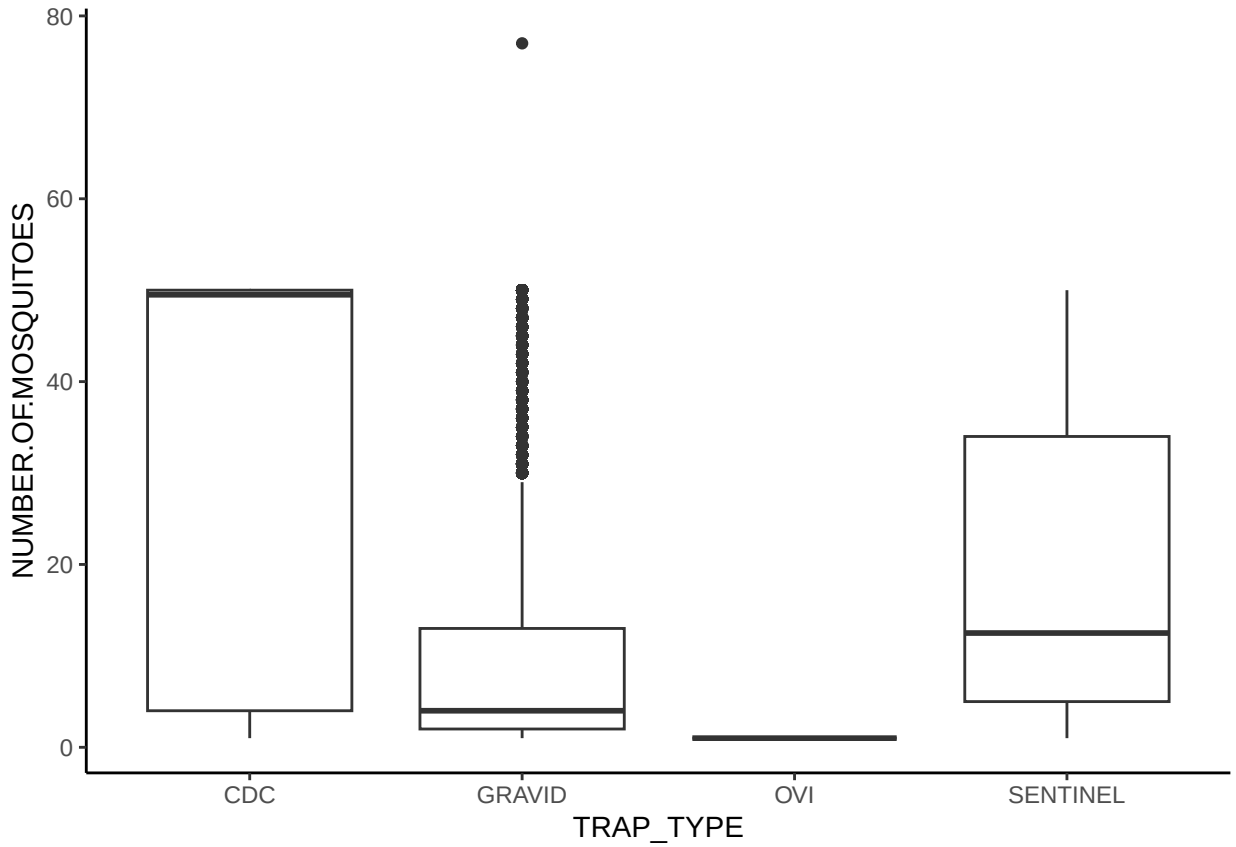

```
# By Latitude
wnv_plot +
  geom_boxplot(mapping = aes(x = round(LATITUDE, 3),
                             y = NUMBER.OF.MOSQUITOES,
                             group = round(LATITUDE, 3)),
               position = "identity",
               outlier.shape = NA) +
  scale_x_continuous()
```




```
# By Longitude
wnv_plot +
  geom_boxplot(mapping = aes(x = round(LONGITUDE, 3),
                             y = NUMBER.OF.MOSQUITOES,
                             group = round(LONGITUDE, 3)),
               position = "identity",
               outlier.shape = NA) +
  scale_x_continuous()
```




```
# By Trap type
wnv_plot +
  geom_boxplot(mapping = aes(x = TRAP_TYPE,
                             y = NUMBER.OF.MOSQUITOES,
                             group = TRAP_TYPE),
               position = "identity")
```




You can also try to plot the number of mosquitoes on the map or plot against `BLOCK`. As your data is clean now, you can formulate and investigate your own ideas. You can also try to look for other data around. For example, what is the effect of weather during the given years on the number of mosquitoes and species in the region? . ### Document

- The `wnv.csv` data frame had 29489 x 9 rows and columns, follows a wide format, and has a generally appropriate structure. After our cleaning, it included 29489 x 11 rows and columns.
- The title of the `SEASON.YEAR` column was changed to `YEAR`.
- The `TEST.DATE` column was recoded a **date**.
- `NUMBER.OF.MOSQUITOES` is mostly relatively low except for one case where the trap captured over 70 mosquitoes at once (`TEST.ID` is 39090). We did not find anything special about it.
- We removed brackets from the `LOCATION` variable and separated it into `LATITUDE` and `LONGITUDE`.
- 4416 cells in the `LOCATION` column were empty with nothing inside. We converted them to `NA`. Later, it is possible to estimate the location by comparing its `TRAP` ID.

## Practice 2. Tests for antibodies to trachoma PGP3 antigen

A short summary of the data:

This set includes data used in a latent class model to compare testing platforms for detection of antibodies against the *Chlamydia trachomatis* antigen *Pgp3*.

Source: data.cdc.gov

The dataset is trimmed from the original dataset.

### Getting the data

Input the `Tests_PGP3.txt` (**Hint**: Try different input functions or parameter settings. Do the input data formats look the same?)

```
pgp3 <- read.table(file = "Week_7_Tests_PGP3.txt", sep = "\t")
```

### Screen and diagnosis

- Is this a long or wide form? Is it tidy?
- Do we need to convert it? Try to stick to the "tidy data" principle.
- Are the variable names informative and precise? (**Hint**: The variable `sex` is not readable, convert it to a more readable format. Men are coded as `1` and women are coded as `2`).
- Are there missing values? Pay attention to the types of missing values.
- Are there duplicated values? Try `which( duplicated() )` . `which()` returns the index. Try to set the parameter `fromLast` as `TRUE` in the function `duplicated()`
- Are there any strange patterns?

```
head(pgp3, 10)
```

```
##    SampleID     measured value
## 1         1          sex  <NA>
## 2         1        age.f
## 3         1     elisa.od 0.097
## 4         1 elisa.pre.od 0.062
## 5         2          sex  <NA>
## 6         2        age.f
## 7         2     elisa.od 0.127
## 8         2 elisa.pre.od 0.099
## 9         2          sex  <NA>
## 10        3          sex  <NA>
```

```
anyNA(pgp3)
```

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

```
sum(duplicated(pgp3)) # Some cells are duplicated
```

```
## [1] 4
```

```
pgp3[which(duplicated(pgp3)),]
```

```
##     SampleID measured value
## 9          2      sex  <NA>
## 30         7      sex  <NA>
## 187       46    age.f
## 808      201      sex     2
```

```
# Let's check the respective samples
pgp3[which(pgp3$SampleID %in% c(2,7,46,201)),]
```

```
##     SampleID     measured   value
## 5          2          sex    <NA>
## 6          2        age.f
## 7          2     elisa.od   0.127
## 8          2 elisa.pre.od   0.099
## 9          2          sex    <NA>
## 26         7          sex    <NA>
## 27         7        age.f
## 28         7     elisa.od   0.186
## 29         7 elisa.pre.od   0.209
## 30         7          sex    <NA>
## 183       46          sex    <NA>
## 184       46        age.f
## 185       46     elisa.od   0.246
## 186       46 elisa.pre.od   0.132
## 187       46        age.f
## 804      201          sex       2
## 805      201        age.f (20,30]
## 806      201     elisa.od    0.32
## 807      201 elisa.pre.od   0.224
## 808      201          sex       2
```

```
# I see that some rows are just duplicated and there is nothing more serious.
```

Ok, the data set requires some effort:

- The format is long. It means it is not tidy.
- Some cells have `NA` values, some cells are empty, but not `NAs`, and there are some duplicated rows.

Firstly, let's reshape our dataset.

```
# Remove duplicated rows
pgp3 <- pgp3 %>%
  .[-which(duplicated(.)),] %>%
  # And convert to the wide format
  spread(key = measured, value = value)
head(pgp3)
```

```
##   SampleID age.f elisa.od elisa.pre.od  sex
## 1        1          0.097        0.062 <NA>
## 2        2          0.127        0.099 <NA>
## 3        3          0.517        0.332 <NA>
## 4        4          0.052        0.031 <NA>
## 5        5          0.163        0.219 <NA>
## 6        6          0.181        0.188 <NA>
```

```
colnames(pgp3)
```

```
## [1] "SampleID"     "age.f"        "elisa.od"     "elisa.pre.od" "sex"
```

```
str(pgp3)
```

```
## 'data.frame':    580 obs. of  5 variables:
##  $ SampleID    : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ age.f       : chr  "" "" "" "" ...
##  $ elisa.od    : chr  "0.097" "0.127" "0.517" "0.052" ...
##  $ elisa.pre.od: chr  "0.062" "0.099" "0.332" "0.031" ...
##  $ sex         : chr  NA NA NA NA ...
```

```
for(i in 1:ncol(pgp3)){
  if(i == 1){
    writeLines(colnames(pgp3)[i])
    writeLines("NAs in the column:")
    sum(is.na(pgp3[, i])) %>% print()
    writeLines("\n\n")
  }else{
    writeLines(colnames(pgp3)[i])
    table(pgp3[, i]) %>% print()
    writeLines("NAs in the column:")
    sum(is.na(pgp3[, i])) %>% print()
    writeLines("\n\n")
  }
}

# This code will produce a very long output.  Thus, to save space, I will not evaluate it.
# You can do it by yourself.
```

We removed duplicated rows and reshaped the dataset to the wide format.

We see the following issues:

- The column order is counter-intuitive. For instance, it is better to keep independent variables *prior to* the dependent ones. And among columns with measurements from several time points (measurements taken *before* and *after* a certain event), it is better to keep the ones for the earlier measurements before the latter ones. In our case, it would be better to rearrange our columns into this order: `SampleID, age.f, sex, elisa.pre.od, elisa.od`. It is a minor issue, but it is an issue.
- 107 cells of the `age.f` column are empty, but not `NAs`. 103 cells of the `sex` column have `NAs`.
- `Gender` is encoded as `1` (men) or `2` (women) while better encoding is preferable.
- The data is still not tidy: each row includes two measurements stored in the `elisa.pre.od` and `elisa.od` columns. It would be better to arrange the data in a way that allows paired comparisons (the antigen was measured at 2 time points in the same subject).
- The data format is completely wrong.

**Treat and document.**

The plan is to:

- Substitute the "" values inside the empty cells with NAs.
- Substitute values that designate gender with more intuitive ones.
- Reformat columns.
- Reshape the dataset to the wide format.

```
# Change codes for `sex`
pgp3$sex[which(pgp3$sex == 1)] <- "Male"
pgp3$sex[which(pgp3$sex == 2)] <- "Female"

# Substitute "" with NAs
pgp3$age.f[pgp3$age.f == ""] <- NA

# Reformat and reshape the dataset
pgp3 <- pgp3 %>%
  mutate_at(.vars = c(1,2,5), .funs = as.factor) %>%
  mutate_at(.vars = c(3,4), .funs = as.numeric) %>%
  relocate(1,5,2,4,3) %>%
  gather(key = "Time.point", value = "OD", c(4,5)) %>%
  drop_na(c(2,3)) %>%
  mutate(Time.point = factor(Time.point, levels = c("elisa.pre.od", "elisa.od"), ordered = T)) %>%
  droplevels()
str(pgp3)
```

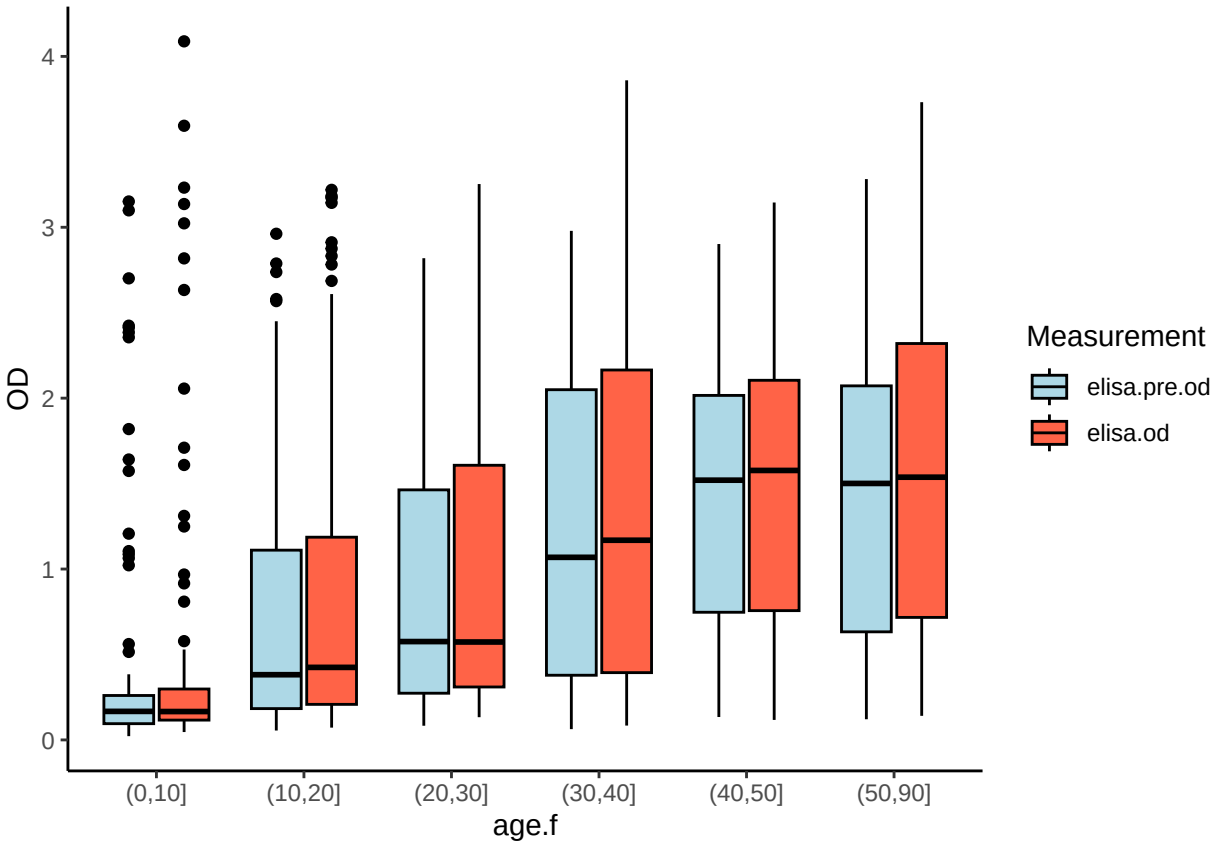
```
## 'data.frame':    944 obs. of  5 variables:
##  $ SampleID  : Factor w/ 472 levels "82","83","84",..: 1 2 3 4 5 6 7 8 9 10 ...
##  $ sex       : Factor w/ 2 levels "Female","Male": 2 2 2 1 2 2 2 1 2 2 ...
##  $ age.f     : Factor w/ 6 levels "(0,10]","(10,20]",..: 4 2 6 6 2 4 4 6 4 6 ...
##  $ Time.point: Ord.factor w/ 2 levels "elisa.pre.od"<..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ OD        : num  0.201 1.045 1.497 0.505 0.147 ...
```

We changed codes for sex, substituted empty cells in age.f with NAs, and reshaped the data.

**Hint**: it would be better to check these empty cells in the age.f column more precisely.

**Summarize your data:**

```
ggplot(data = pgp3, aes(x = age.f,
                        y = OD,
                        fill = Time.point)) +
  geom_boxplot(color = "black") +
  theme_classic() +
  scale_fill_manual(labels = c("elisa.pre.od", "elisa.od"),
                    name = "Measurement",
                    values = c("#ADD8E6", "#FF6347"),
                    breaks = c("elisa.pre.od", "elisa.od")) +
  theme(legend.position = "right") +
  guides(color = guide_legend(override.aes = list(size = 3) ) )
```




**Document**

- The `pgp3` data frame has 2324 x 3 rows and columns and follows a long format. Several rows were duplicated. We removed these duplicated rows and converted our table to a wide format. Finally, it included 944 x 5 rows and columns.
- The column order after reshaping was still counter-intuitive. For instance, it was `SampleID, age.f, elisa.od, elisa.pre.od`, and `sex`. We rearranged columns as follows: `SampleID, age.f, sex, elisa.pre.od, elisa.od`.
- 107 cells of the `age.f` column were empty, but not `NAs`. 103 cells of the `sex` column had `NAs`. We substituted the `""` values inside the empty cells of the `age.f` column with `NAs`.
- `Gender` was encoded as `1` (men) or `2` (women) while better encoding is preferable. We substituted these values with more intuitive ones.
- After that step, we further gathered the `elisa.pre.od` and `elisa.od` columns into another two columns: `Time.point` (the grouping variable) and `OD` (the actual values).
- The data format was completely wrong with `age.f, sex, elisa.pre.od`, and `elisa.od` columns set to contain the **character** data while `SampleID` was set to contain **integers**. We changed `SampleID, age.f`, and `sex` to **factors**, and `elisa.pre.od` and `elisa.od` were changed to the **numeric** data.

## Further notes

**1. Regardless of the solution shown here, it does not mean that you should act exactly in this way. For example, the `wnv` dataset is still not clean enough: we did not try to treat `NA` cases properly, we did not convert all the columns to the appropriate types of data. There are plenty of hypotheses to test as well.**

**2. Whenever you get some data, be curious about it. Check it. Interesting features or problems may not be seen clearly at first.**

---

Originally, created by Chaochen Wang in 2019.

Updated by D Shytikov in 2023.