Handling Missing Values

STAT 133

Gaston Sanchez

Department of Statistics, UC–Berkeley

gastonsanchez.com
github.com/gastonstat/stat133

Course web: gastonsanchez.com/teaching/stat133

Missing Values

Missing Values are very common

- "no answer" in a questionnaire / survey
- data that are lost or destroyed
- machines that fail
- experiments/samples that are lost
- things not working

The best thing to do about missing values is not to have any

Gertrude Cox

Missing Values

Missing Values in R

- Missing values in R are denoted with `NA`
- `NA` stands for **Not Available**
- `NA` is actually a **logical** value
- Do not confuse `NA` with `"NA"` (character)
- Do not confuse `NA` with `NaN` (not a number)

Missing Values Functions in R

```
# NA is a logical value
is.logical(NA)

## [1] TRUE

# NA is not the same as NaN
identical(NA, NaN)

## [1] FALSE

# NA is not the same as "NA"
identical(NA, "NA")

## [1] FALSE
```

Function is.na()

- is.na() indicates which elements are missing
- is.na() is a generic function (i.e. can be used for vectors, factors, matrices, etc)

```
x <- c(1, 2, 3, NA, 5)
x
```

```
## [1]  1  2  3 NA  5
```

```
is.na(x)
```

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

Function `is.na()`

`is.na()` on a factor

```
g <- factor(c(letters[rep(1:3, 2)], NA))
g

## [1] a    b    c    a    b    c    <NA>
## Levels: a b c

is.na(g)

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

Notice how missing values are denoted in factors

Function is.na()

is.na() on a matrix

```
m <- matrix(c(1:4, NA, 6:9, NA), 2)
m

##      [,1] [,2] [,3] [,4] [,5]
## [1,]    1    3   NA    7    9
## [2,]    2    4    6    8   NA

is.na(m)

##       [,1]  [,2]  [,3]  [,4]  [,5]
## [1,] FALSE FALSE  TRUE FALSE FALSE
## [2,] FALSE FALSE FALSE FALSE  TRUE
```

Function is.na()

is.na() on a data.frame

```
d <- data.frame(m)
d

##   X1 X2 X3 X4 X5
## 1  1  3 NA  7  9
## 2  2  4  6  8 NA

is.na(d)

##         X1    X2    X3    X4    X5
## [1,] FALSE FALSE  TRUE FALSE FALSE
## [2,] FALSE FALSE FALSE FALSE  TRUE
```

Function is.na()

If you're reading a data table with missing values codified differently from NA, you can specify the parameter na.strings

```
url <- "http://www.esapubs.org/archive/ecol/E084/094/MOMv3.3.txt"

df <- read.table(file = url, header = FALSE,
                 sep = "\t", na.strings = -999)
```

Computing with `NA`s

Computing with NA's

Numerical computations using NA will normally result in NA

```
2 + NA

## [1] NA

x <- c(1, 2, 3, NA, 5)
x + 1

## [1]  2  3  4 NA  6
```

Computing with NA's

```
sqrt(x)

## [1] 1.000000 1.414214 1.732051       NA 2.236068

mean(x)

## [1] NA

max(x)

## [1] NA
```

Argument `na.rm`

Most arithmetic/trigonometric/summarizing functions provide the argument `na.rm = TRUE` that removes missing values before performing the computation:

- `mean(x, na.rm = TRUE)`
- `sd(x, na.rm = TRUE)`
- `var(x, na.rm = TRUE)`
- `min(x, na.rm = TRUE)`
- `max(x, na.rm = TRUE)`
- `sum(x, na.rm = TRUE)`
- *etc*

Argument `na.rm`

```r
x <- c(1, 2, 3, NA, 5)

mean(x, na.rm = TRUE)

## [1] 2.75

sd(x, na.rm = TRUE)

## [1] 1.707825

median(x, na.rm = TRUE)

## [1] 2.5
```

Argument `na.rm`

```r
x <- c(1, 2, 3, NA, 5)

y <- c(2, 4, 7, 9, 11)

var(x, y, na.rm = TRUE)

## [1] 6.666667
```

Correlations with NA

```r
# default correlation
cor(x, y)

## [1] NA

# argument 'use'
cor(x, y, use = 'complete.obs')

## [1] 0.9968896
```

`NA` Actions

Argument `na.rm`

Additional functions for handling missing values:

- `anyNA()`
- `na.omit()`
- `complete.cases()`
- `na.fail()`
- `na.exclude()`
- `na.pass()`

Checking for missing values

A common operation is to check for the presence of missing values in a given object:

```
x <- c(1, 2, 3, NA, 5)

any(is.na(x))

## [1] TRUE

# alternatively
anyNA(x)

## [1] TRUE
```

Checking for missing values

Another common operation is to calculate the number of missing values:

```
y <- c(1, 2, 3, NA, 5, NA)

# how many NA's
sum(is.na(y))

## [1] 2
```

Excluding missing values

Sometimes we want to “remove” missing values from a vector or factor:

```
x <- c(1, 2, 3, NA, 5, NA)

# excluding NA's
x[!is.na(x)]

## [1] 1 2 3 5
```

Excluding missing values

Another way to "remove" missing values from a vector or factor is with `na.omit()`

```
x <- c(1, 2, 3, NA, 5, NA)

# removing NA's
na.omit(x)

## [1] 1 2 3 5
## attr(,"na.action")
## [1] 4 6
## attr(,"class")
## [1] "omit"
```

Excluding missing values

There's also the `na.exclude()` function that we can use to "remove" missing values

```
x <- c(1, 2, 3, NA, 5, NA)

# removing NA's
na.exclude(x)

## [1] 1 2 3 5
## attr(,"na.action")
## [1] 4 6
## attr(,"class")
## [1] "exclude"
```

Excluding rows with missing values

Applying `na.omit()` on matrices or data frames will exclude the rows containing any missing value

```
DF <- data.frame(x = c(1, 2, 3), y = c(0, 10, NA))
DF

##   x  y
## 1 1  0
## 2 2 10
## 3 3 NA

# how many NA's
na.omit(DF)

##   x  y
## 1 1  0
## 2 2 10
```

Function `complete.cases()`

Likewise, we can use `complete.cases()` to get a logical vector with the position of those rows having complete data:

```
DF <- data.frame(x = c(1, 2, 3), y = c(0, 10, NA))

# how many NA's
complete.cases(DF)

## [1]  TRUE  TRUE FALSE
```

Function `na.fail()`

`na.fail()` returns the object if it does not contain any missing values, and signals an error otherwise

```
x <- c(1, 2, 3, NA, 5)
na.fail(x)  # fails

## Error in na.fail.default(x):  missing values in object

y <- c(1, 2, 3, 4, 5)
na.fail(y)  # doesn't fail

## [1] 1 2 3 4 5
```

Handling Missing Values

Dealing with missing values

What to do with missing values?

- Correct them (if possible)
- Deletion
- Imputation
- Leave them as is

Correcting

Correcting NAs

- Perhaps there is more data now
- Go back to the original source
- Look for additional information

Deleting NAs

- How many NA's (counts, percents)?
- Can you get rid of them?
- What type of consequences?
- How bad is it to delete NA's?

Deletion

Deleting NAs

- `x[!is.na(x)]`
- `na.omit(DF)`
- `na.exclude(DF)`
- Some functions-methods in R delete NA's by default; e.g. `lm()`

Imputation

Imputing NAs

- Try to fill in values
- Several strategies to fill in values
- No magic wand technique

Imputation

Imputing with measure of centrality

One option is to filling values with some measure of centrality

- mean value (quantitative variables)
- median value (quantitative variables)
- most common value (qualitative variables)

These options require to inspect each variable individually

Imputation

If a variable has a **symmetric** distribution, we can use the mean value

```
# mean value
mean_x <- mean(x, na.rm = TRUE)

# imputation
x[is.na(x)] <- mean_x
```

Imputation

If a variable has a **skewed** distribution, we can use the median value

```
# median value
median_x <- median(x, na.rm = TRUE)

# imputation
x[is.na(x)] <- median_x
```

Imputation

For a qualitative variable we can use the mode value—i.e. most common category—(if there is one)

```
# mode
g <- factor(c('a', 'a', 'b', 'c', NA, 'a'))
mode_g <- g[which.max(table(g))]

# imputation
g[is.na(g)] <- mode_g
```

Imputing with correlations

Explore correlations between variables and look for "high" correlations

```
cor(x, y, use = "complete.obs")
```

What is a "high" correlation?

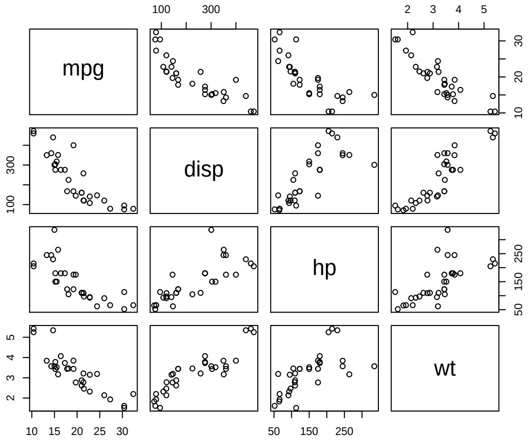
High correlated variables

```r
# subset of 'mtcars'
df <- mtcars[ ,c('mpg', 'disp', 'hp', 'wt')]
head(df)

##                    mpg disp  hp    wt
## Mazda RX4         21.0  160 110 2.620
## Mazda RX4 Wag     21.0  160 110 2.875
## Datsun 710        22.8  108  93 2.320
## Hornet 4 Drive    21.4  258 110 3.215
## Hornet Sportabout 18.7  360 175 3.440
## Valiant           18.1  225 105 3.460
```

```r
# missing values in 'mpg'
df$mpg[c(5,20)] <- NA
mpg <- df$mpg
```

High correlated variables

```r
# scatterplot matrix
pairs(df)
```



High correlated variables

```r
# matrix of correlations
cor(df, use = 'complete.obs')

##             mpg       disp         hp         wt
## mpg   1.0000000 -0.8584325 -0.7729303 -0.8656222
## disp -0.8584325  1.0000000  0.7825073  0.8909678
## hp   -0.7729303  0.7825073  1.0000000  0.6386074
## wt   -0.8656222  0.8909678  0.6386074  1.0000000
```

`mpg` is most correlated with `wt`

Regression analysis with lm()

```
# matrix of correlations
regression <- lm(mpg ~ wt, data = df)

summary(regression)


##
## Call:
## lm(formula = mpg ~ wt, data = df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.3073 -2.0725 -0.2766  1.5742  7.4297
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   36.000      1.860  19.351  < 2e-16 ***
## wt            -5.013      0.548  -9.148 6.62e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.883 on 28 degrees of freedom
##   (2 observations deleted due to missingness)
## Multiple R-squared:  0.7493, Adjusted R-squared:  0.7403
## F-statistic: 83.69 on 1 and 28 DF,  p-value: 6.615e-10
```

Regression analysis with `lm()`

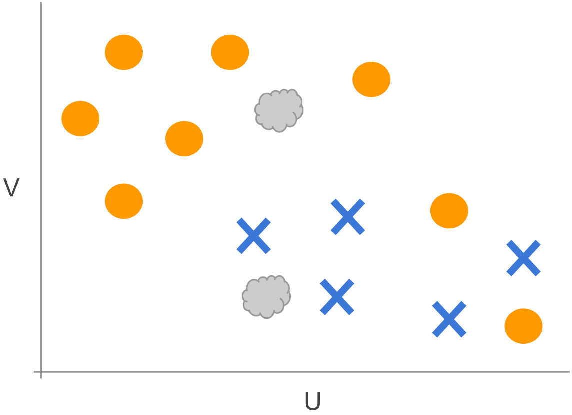
```
# prediction
predict(regression, newdata = df[c(5,20),-1])

## Hornet Sportabout    Toyota Corolla
##          18.75370          26.80021

# compare with true values
mtcars$mpg[c(5,20)]

## [1] 18.7 33.9
```

Nearest Neighbors

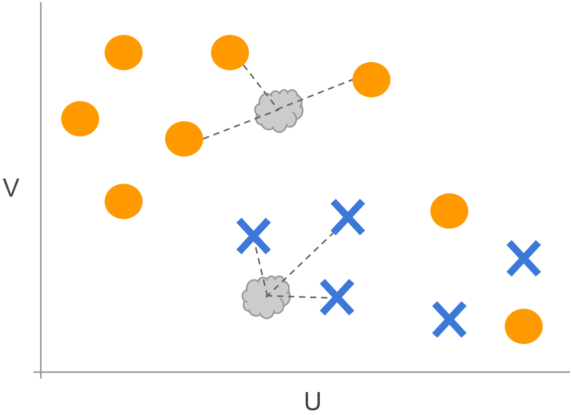
Imputing with similarities

We can calculate distances or similarities between two or more observations

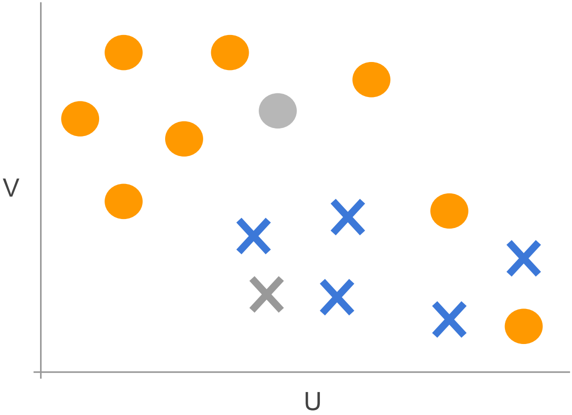
Nearest Neighbors Idea



Nearest Neighbors Idea



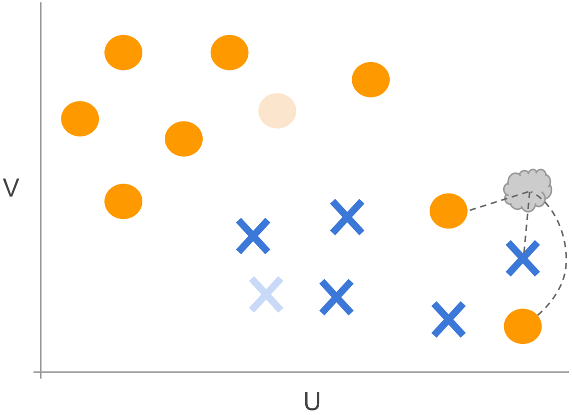
Nearest Neighbors Idea



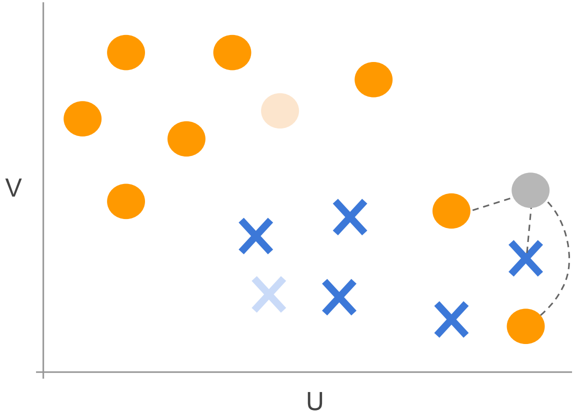
Nearest Neighbor Imputation

- observations near each other in (u, v) space will have similar values (circles, crosses)
- find the $k = 3$ nearest points in (u, v) to the missing value
- let the circles and crosses vote
- if the neighbors have 2/3 circles or all circles, then assign circle, (cross otherwise)

Nearest Neighbors Idea



Nearest Neighbors Idea



Nearest Neighbor Imputation

Questions

- How to choose k?
- How to choose $u, v, ...$? (predicting variables)
- What type of distance/similarity measure?

Nearest Neighbor Imputation

Function `knn()` from package `"class"`

```
knn(train, test, cl, k = 1, l = 0, use.all = TRUE)
```

- `train` matrix or data frame of training set cases
- `test` matrix or data frame of test set cases
- `cl` factor of true classifications of training set
- `k` number of neighbors

Function knn()

```
# subset of 'mtcars'
df <- mtcars[ ,c('mpg', 'disp', 'hp', 'wt')]
head(df)

##                    mpg disp  hp    wt
## Mazda RX4         21.0  160 110 2.620
## Mazda RX4 Wag     21.0  160 110 2.875
## Datsun 710        22.8  108  93 2.320
## Hornet 4 Drive    21.4  258 110 3.215
## Hornet Sportabout 18.7  360 175 3.440
## Valiant           18.1  225 105 3.460
```

```
# missing values in 'mpg'
df$mpg[c(5,20)] <- NA
mpg <- df$mpg
```

Function knn()

```
library(class)

df_aux <- df[ ,-1]                    # data without mpg
df_ok <- df_aux[!is.na(mpg), ]   # train set
df_na <- df_aux[is.na(mpg), ]    # test set

# 1 nearest neighbor
nn1 <- knn(
  train = df_ok,
  test = df_na,
  cl = mpg[!is.na(mpg)],
  k = 1)
```

Function knn()

```
# imputed values
nn1

## [1] 19.2 32.4
## 23 Levels: 10.4 13.3 14.3 14.7 15 15.2 15.5 15.8 16.4 17.3 17.

# compared to real values
mtcars$mpg[c(5,20)]

## [1] 18.7 33.9
```

Function knn()

```
# 3 nearest neighbor
nn3 <- knn(
  train = df_ok,
  test = df_na,
  cl = mpg[!is.na(mpg)],
  k = 3)

# imputed values
nn3

## [1] 19.2 30.4
## 23 Levels: 10.4 13.3 14.3 14.7 15 15.2 15.5 15.8 16.4 17.3 17.

# real values
mtcars$mpg[c(5,20)]

## [1] 18.7 33.9
```

R Packages
VIM and missMDA

Vim and missMDA

- package "VIM" by Templ et al
- package "missMDA" by Francois Husson and Julie Josse

```
install.packacges(c("VIM", "missMDA"))
```

```
library(VIM)
library(missMDA)
```

Data ozone

Data ozone (in "missMDA"): daily measurements of meteorological variables and ozone concentration:

```
data(ozone)

head(ozone, n = 5)


##          maxO3   T9  T12  T15 Ne9 Ne12 Ne15     Vx9    Vx12    Vx15 maxO3v
## 20010601    87 15.6 18.5 18.4   4    4    8  0.6946 -1.7101 -0.6946     84
## 20010602    NA 17.0 18.4 17.7   5    5    7 -4.3301 -4.0000 -3.0000     87
## 20010603    92 15.3 17.6 19.5   2    5    4  2.9544  1.8794  0.5209     82
## 20010604   114 16.2 19.7 22.5   1   NA    0  0.9848      NA      NA     92
## 20010605    94 17.4 20.5 20.4   8    8    7 -0.5000 -2.9544 -4.3301    114
##           vent pluie
## 20010601  Nord   Sec
## 20010602  Nord   Sec
## 20010603   Est  <NA>
## 20010604  <NA>   Sec
## 20010605 Ouest   Sec
```

Data ozone

Number of missing values in each variable:

```
num_na <- sapply(ozone, function(x) sum(is.na(x)))
num_na[1:7]; num_na[8:13]

## max03     T9    T12    T15    Ne9   Ne12   Ne15
##     13      9     16     19      6     11     13
##    Vx9   Vx12   Vx15 max03v   vent  pluie
##      6     14     14      5     11      8
```

Looking at missing values

```r
# aggregation for missing values
oz_aggr <- aggr(ozone, prop = TRUE,
       combined = TRUE, plot = FALSE)

# summary
res <- summary(oz_aggr)
```

Looking at missing values

```
# variables sorted by number of missings
res$missings[order(res$missings[,2]), ]

##         Variable Count
## maxO3v   maxO3v     5
## Ne9         Ne9     6
## Vx9         Vx9     6
## pluie     pluie     8
## T9           T9     9
## Ne12       Ne12    11
## vent       vent    11
## maxO3     maxO3    13
## Ne15       Ne15    13
## Vx12       Vx12    14
## Vx15       Vx15    14
## T12         T12    16
## T15         T15    19
```

Looking at missing values

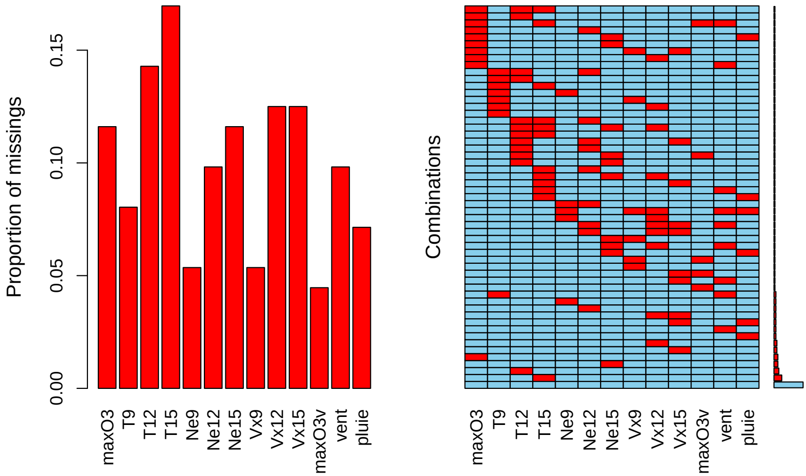
```
# combinations
head(res$combinations, n = 10)

##                  Combinations Count    Percent
## 1  0:0:0:0:0:0:0:0:0:0:0:0:0    30 26.7857143
## 2  0:0:0:0:0:0:0:0:0:0:0:0:1     2  1.7857143
## 3  0:0:0:0:0:0:0:0:0:0:0:1:0     2  1.7857143
## 4  0:0:0:0:0:0:0:0:0:0:1:0:0     1  0.8928571
## 5  0:0:0:0:0:0:0:0:0:1:0:0:0     3  2.6785714
## 6  0:0:0:0:0:0:0:0:0:1:0:0:1     2  1.7857143
## 7  0:0:0:0:0:0:0:0:0:1:0:1:0     1  0.8928571
## 8  0:0:0:0:0:0:0:0:0:1:1:0:0     1  0.8928571
## 9  0:0:0:0:0:0:0:0:1:0:0:0:0     3  2.6785714
## 10 0:0:0:0:0:0:0:0:1:1:0:0:0     2  1.7857143
```

Looking at missing values

```
# combinations
tail(res$combinations, n = 10)

##                   Combinations Count   Percent
## 46 1:0:0:0:0:0:0:0:0:0:0:0:0     4 3.5714286
## 47 1:0:0:0:0:0:0:0:0:0:0:1:0     1 0.8928571
## 48 1:0:0:0:0:0:0:0:1:0:0:0:0     1 0.8928571
## 49 1:0:0:0:0:0:0:1:0:1:0:0:0     1 0.8928571
## 50 1:0:0:0:0:0:1:0:0:0:0:0:0     1 0.8928571
## 51 1:0:0:0:0:0:1:0:0:0:0:0:1     1 0.8928571
## 52 1:0:0:0:0:1:0:0:0:0:0:0:0     1 0.8928571
## 53 1:0:0:1:0:0:0:0:0:0:1:1:0     1 0.8928571
## 54 1:0:1:0:0:0:0:0:0:0:0:0:0     1 0.8928571
## 55 1:0:1:1:0:0:0:0:0:0:0:0:0     1 0.8928571
```

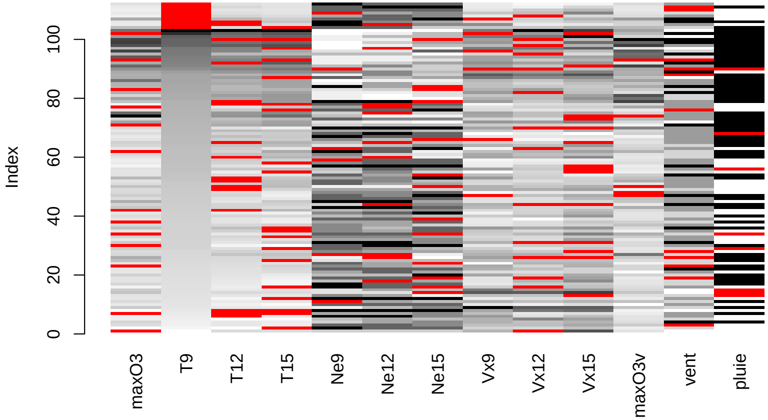
Looking at missing values

```r
# visualizations
plot(oz_aggr)
```



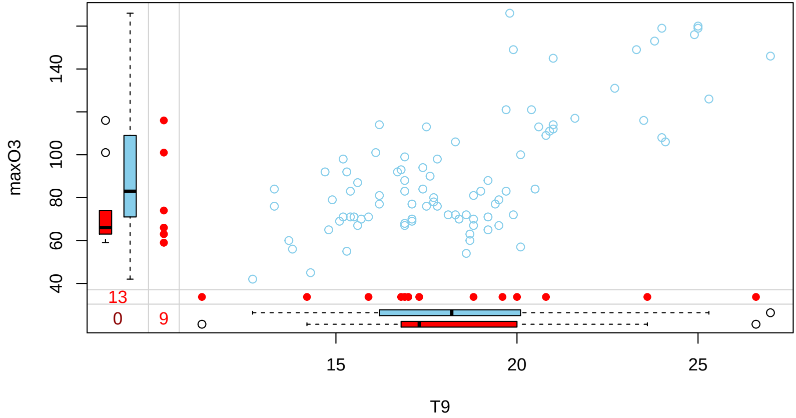
Looking at missing values

```r
# visualizations
matrixplot(ozone, sortby = 2)
```



Looking at missing values

```r
# visualizations
marginplot(ozone[ ,c('T9', 'maxO3')])
```



More info ...

- Is there a pattern of missing values?
- Is there a mechanism leading to missing values?
  - purely random?
  - probability model for missing values?
- There are more sophisticated options:
  "Missing Data: Our View of the State of the Art" (Schafer & Graham, 2000)
- Bayesian imputation
- Multiple imputation
- *etc*