Statistical Computing with R Exam

Gaspard Gaches 4645251

2025-12-19

Exercise 1

We begin by loading the necessary libraries, and defining the variables we will work with:

```r
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library(ptmixed)
```

```
## Warning: package 'ptmixed' was built under R version 4.5.2
```

```r
df_airquality = airquality
```

1.

We first take a look at the unique months present in the airquality dataframe, and we notice that the months present are May through September. We select a subset of the dataset only containing rows for the months of June, July and August:

```r
unique(df_airquality$Month)  # 5,6,7,8,9
```

```
## [1] 5 6 7 8 9
```

```r
df_airquality = df_airquality |> filter(Month==6|Month==7|Month==8)
```

2.

Next, we create a new Ozone.category variable with 4 levels in order to simplify our analysis:

```
to_cat = \(x) {
  if (is.na(x)) {
    'Na'
  }
  else if (x<50) {
    'Low'
  }
  else if (x >= 50 & x < 100) {
    'Medium'
  }
  else if (x >= 100) {
    'High'
  }
}
Ozone.category = sapply(df_airquality$Ozone, to_cat)
df_airquality$Ozone.category = Ozone.category
head(df_airquality)
```

```
##   Ozone Solar.R Wind Temp Month Day Ozone.category
## 1    NA     286  8.6   78     6   1             Na
## 2    NA     287  9.7   74     6   2             Na
## 3    NA     242 16.1   67     6   3             Na
## 4    NA     186  9.2   84     6   4             Na
## 5    NA     220  8.6   85     6   5             Na
## 6    NA     264 14.3   79     6   6             Na
```

3.

We begin by creating a new Month.name column to make the spaghetti plot more readable:

```
# adding month name column for
month_name = \(x) {
  if (x==6) 'June'
  else if (x==7) 'July'
  else if (x==8) 'August'
}
df_airquality$Month.name = sapply(df_airquality$Month, month_name)
```

Let's now group our dataframe by month AND ozone category and count the proportion of days for each ozone category, for each month:

```
tbl_q3 = df_airquality
tbl_q3 = tbl_q3 |>
  group_by(Month, Ozone.category) |> count()
tbl_q3
```

```
## # A tibble: 11 x 3
## # Groups:   Month, Ozone.category [11]
##    Month Ozone.category     n
##    <int> <chr>          <int>
##  1     6 Low                8
##  2     6 Medium             1
##  3     6 Na                21
##  4     7 High               2
##  5     7 Low               10
##  6     7 Medium            14
##  7     7 Na                 5
##  8     8 High               4
##  9     8 Low               13
## 10     8 Medium             9
## 11     8 Na                 5
```

The month of August has the highest proportion of ‘High’ Ozone days.

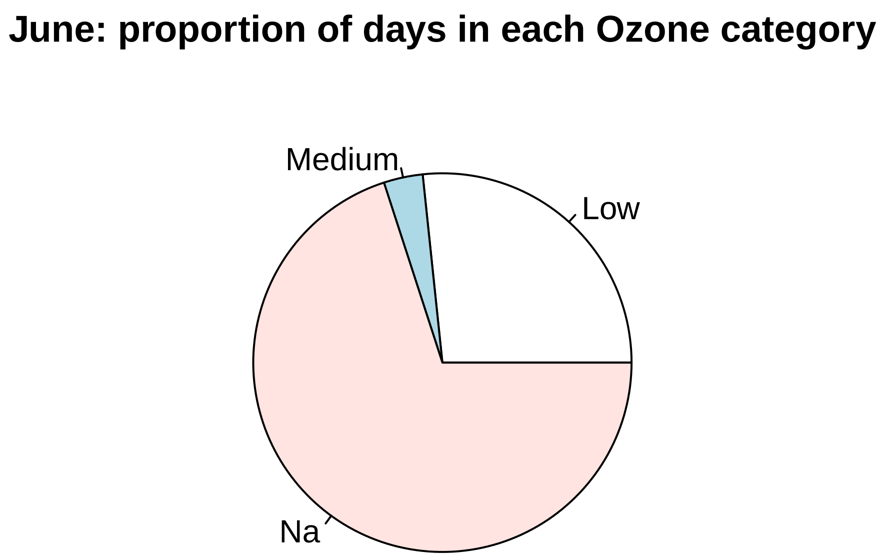
```
(june_q3 = subset(tbl_q3, Month==6, c('Ozone.category', 'n')))
```

```
## # A tibble: 3 x 2
## # Groups:   Ozone.category [3]
##   Ozone.category     n
##   <chr>          <int>
## 1 Low                8
## 2 Medium             1
## 3 Na                21
```

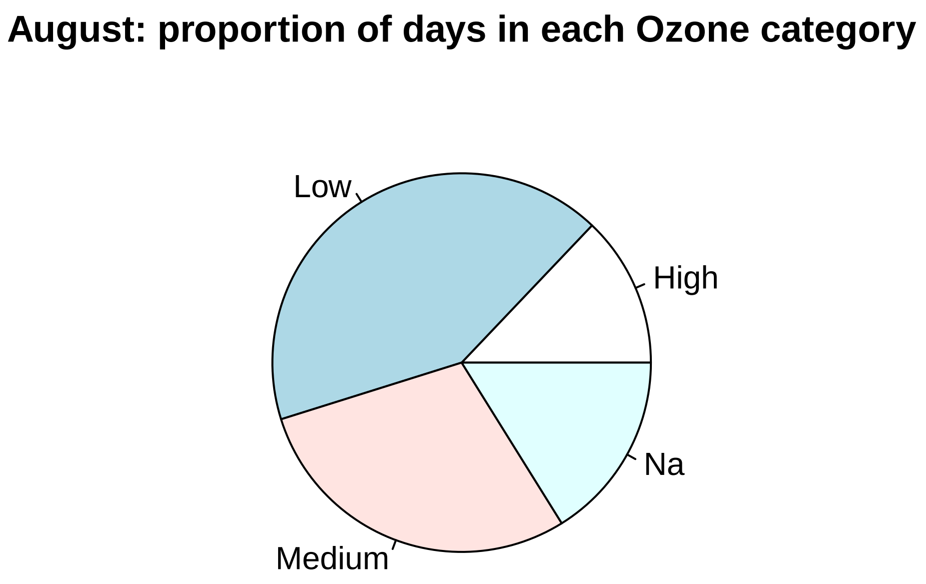
```
(aug_q3 = subset(tbl_q3, Month==8, c('Ozone.category', 'n')))
```

```
## # A tibble: 4 x 2
## # Groups:   Ozone.category [4]
##   Ozone.category     n
##   <chr>          <int>
## 1 High               4
## 2 Low               13
## 3 Medium             9
## 4 Na                 5
```

```
pie(x=june_q3$n, labels=june_q3$Ozone.category
    , main='June: proportion of days in each Ozone category')
```

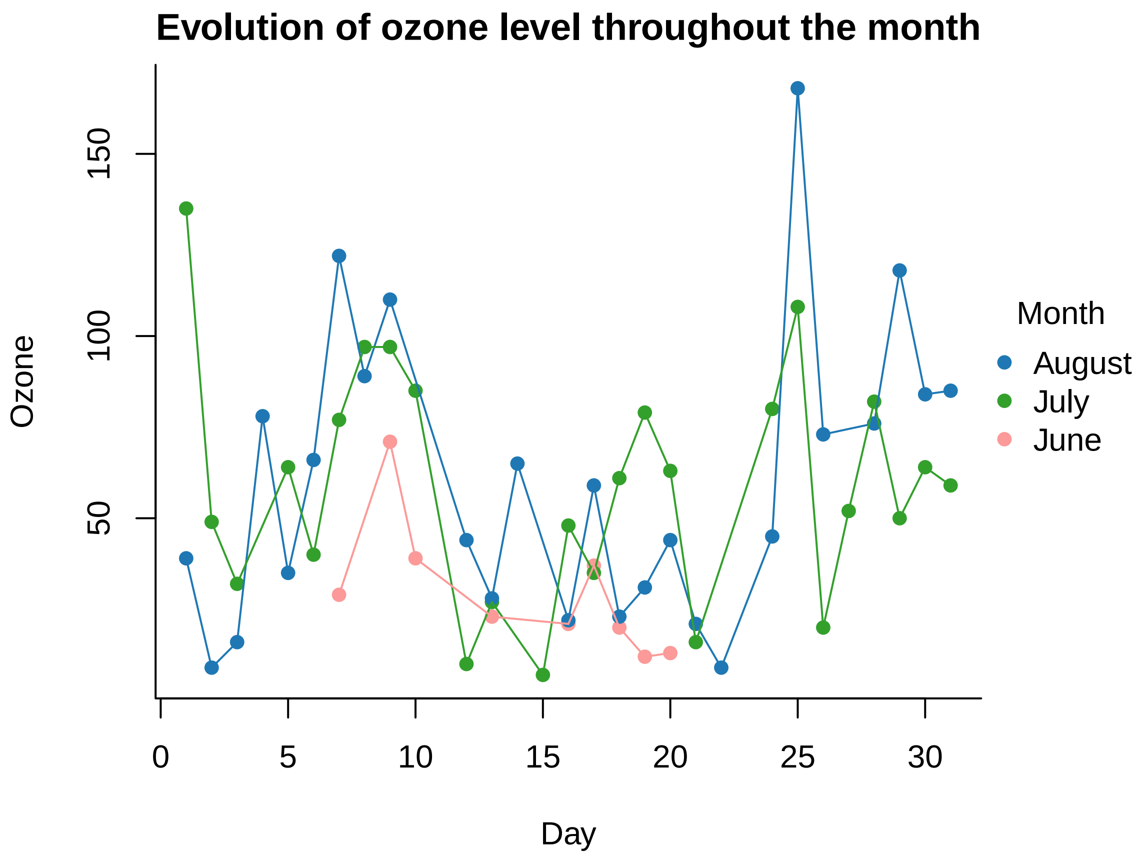


```
pie(x=aug_q3$n, labels=aug_q3$Ozone.category
    , main='August: proportion of days in each Ozone category')
```



The proportion of ‘NA’s dramatically decreased from June to August. The ’Low’ days still dominate the ‘Medium’ days though in higher numbers, and there were approximately as many ‘High’ days in August as ‘NA’s, against no ’High’ days at all in June.

```r
make.spaghetti(Day, Ozone, id=Month.name, group = Month.name, data=df_airquality
               , legend.title='Month', legend.inset=-0.2
               , title='Evolution of ozone level throughout the month')
```



The highest ozone level was reached in late August, while the lowest ozone level was reached in July, though on two August days the ozone level was very close to that extreme low too.

Exercise 2

```
set.seed(1024)
x.obs = rlnorm(120, meanlog=1.2, sdlog=0.55)

loglik = \(mu, sig=0.55) {
  sum(dlnorm(x.obs, meanlog = mu, sdlog = sig, log = T))
}
```

2.

Since mu is the only variabe here, we use the function optimise to find the maximum likelhood estimate of mu:

```
optimise(loglik, c(-5,5), maximum=T)
```

```
## $maximum
## [1] 1.155822
##
## $objective
## [1] -240.6536
```

We find that the MLE of mu is 1.155822, which is roughly 3.7% away from the true mu that we generated x.obs with.

3.

Now, both mu and sigma are unknown parameters. This time we use the function optim, with method 'L-BFGS-B', respecting the lower-bound constraint on sigma which must be strictly positive.

```r
loglik = \(params) {
  mu=params[1]
  sig=params[2]
  -sum(dlnorm(x.obs, meanlog = mu, sdlog = sig, log = T))
}
mle = optim(c(1, 0.5), loglik, method='L-BFGS-B', lower=c(-Inf, 0), upper=c(Inf, Inf))
mle
```

```
## $par
## [1] 1.1558222 0.5654715
##
## $value
## [1] 240.5595
##
## $counts
## function gradient
##       23       23
##
## $convergence
## [1] 0
##
## $message
## [1] "CONVERGENCE: REL_REDUCTION_OF_F <= FACTR*EPSMCH"
```

We notice that the multivariate MLE estimate of mu is extremely close to the one from question 2, with one more precision digit. The MLE estimate of sigma is 0.5654715, which roughly 2.8% away from the true sigma that we generated x.obs with.

4.

See exam paper.

5.

We use the analytical MLE estimates of μ and σ^2 found at question 4 to once again compute their values:

```r
mles = \(x) {
  n = length(x)

  mle_mu = sum(log(x))/n
  mle_sig2 = sum( (log(x) - mle_mu)^2 )/n
  return(c(mle_mu, mle_sig2))
}
result = mles(x.obs)
cat('MLE mu:',result[1], '\n')
```

```
## MLE mu: 1.155822
```

```
cat('MLE sig:', sqrt(result[2]))
```

```
## MLE sig: 0.56547
```

We notice that the numerical and analytical MLEs of μ perfectly match, and the numerical MLE of σ^2 is almost identical to its analytical counterpart with two more precision digits.

Exercise 3

1.

Below we define a function that given an input matrix M and a number minMean returns the submatrix of M whose columns have mean greater or equal than minMean. Additionally, if the input is not of matrix type, the function stops and emits an error message explaining that the input type has to be matrix.

```
matrixColumnFilter = \(M, minMean) {
  if (!is.matrix(M)) stop('Please supply a matrix.')
  col.means = colMeans(M)
  col.subset = which(col.means >= minMean)
  as.matrix(M[,col.subset])
}
```

2.

Modifications were brought to the function so that we can choose to either still consider columns with NAs, or drop them. If ignoreNAs=T, the function will still compute the mean of the columns while ignoring the NA values with na.rm=T. If ignoreNAs=F, the columns containing at least one NA will not be considered.

```
matrixColumnFilter = \(M, minMean, ignoreNAs) {
  if (!is.matrix(M)) stop('Please supply a matrix.')
  if (!ignoreNAs) {
    warning('The function will not ignore NAs.')
    col.means = colMeans(M)
    col.means = col.means[!is.na(colMeans(M))]
  }
  else col.means = colMeans(M, na.rm=T)
  col.subset = which(col.means >= minMean)
  as.matrix(M[,col.subset])
}
```

3.

If no columns remain after applying the function, the function produces a warning:

```
matrixColumnFilter = \(M, minMean, ignoreNAs) {
  if (!is.matrix(M)) stop('Please supply a matrix.')
  if (!ignoreNAs) {
    warning('The function will not ignore NAs.')
    col.means = colMeans(M)
    col.means = col.means[!is.na(colMeans(M))]
  }
  else col.means = colMeans(M, na.rm=T)
  col.subset = which(col.means >= minMean)
  if (length(col.subset) == 0) {
    stop('No columns meet the minimum mean requirement.')
  }
  else {
    as.matrix(M[,col.subset])
  }
}
```

4.

The first error message is slightly modified to also raise an error when the matrix contains characters instead of numbers:

```
matrixColumnFilter = \(M, minMean, ignoreNAs) {
  if (!is.matrix(M) | !is.numeric(M)) stop('Please supply a numeric matrix.')
  if (!ignoreNAs) {
    warning('The function will not ignore NAs.')
    col.means = colMeans(M)
    col.means = col.means[!is.na(colMeans(M))]
  }
  else col.means = colMeans(M, na.rm=T)
  col.subset = which(col.means >= minMean)
  if (length(col.subset) == 0) {
    warning('No columns meet the minimum mean requirement.')
  }
  else {
    as.matrix(M[,col.subset])
  }
}
```

Next, we define the three matrices used in the following two questions:

```
M1 = matrix(c(
  6,NA,4
  ,5,7,3
  ,8,6,NA
), nrow=3, byrow=T)
M2 = matrix(c(
  2,6,8
  ,3,5,7
  ,4,8,9
), nrow=3, byrow=T)
M3 = matrix(c(
  4,5,0
  ,5,5,6
  ,6,5,0
), nrow=3, byrow=T)
```

5.

We apply matrixColumnFilter with minMean=5 and ignoreNAs=T to the three matrices:

```
matrixColumnFilter(M1, minMean=5, ignoreNAs=T)
```

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

```
matrixColumnFilter(M2, minMean=5, ignoreNAs=T)
```

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

```
matrixColumnFilter(M3, minMean=5, ignoreNAs=T)
```

```
##      [,1] [,2]
## [1,]    4    5
## [2,]    5    5
## [3,]    6    5
```

6.

```
matrixColumnFilter(M1, minMean=6, ignoreNAs=F)
```

```
## Warning in matrixColumnFilter(M1, minMean = 6, ignoreNAs = F): The function
## will not ignore NAs.
```

```
##      [,1]
## [1,]    6
## [2,]    5
## [3,]    8
```

```r
matrixColumnFilter(M2, minMean=6, ignoreNAs=F)
```

```
## Warning in matrixColumnFilter(M2, minMean = 6, ignoreNAs = F): The function
## will not ignore NAs.
```

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

```r
matrixColumnFilter(M3, minMean=6, ignoreNAs=F)
```

```
## Warning in matrixColumnFilter(M3, minMean = 6, ignoreNAs = F): The function
## will not ignore NAs.
```

```
## Warning in matrixColumnFilter(M3, minMean = 6, ignoreNAs = F): No columns meet
## the minimum mean requirement.
```