Introduction to Statistics and R

R Crash Course

Eric Stemmler

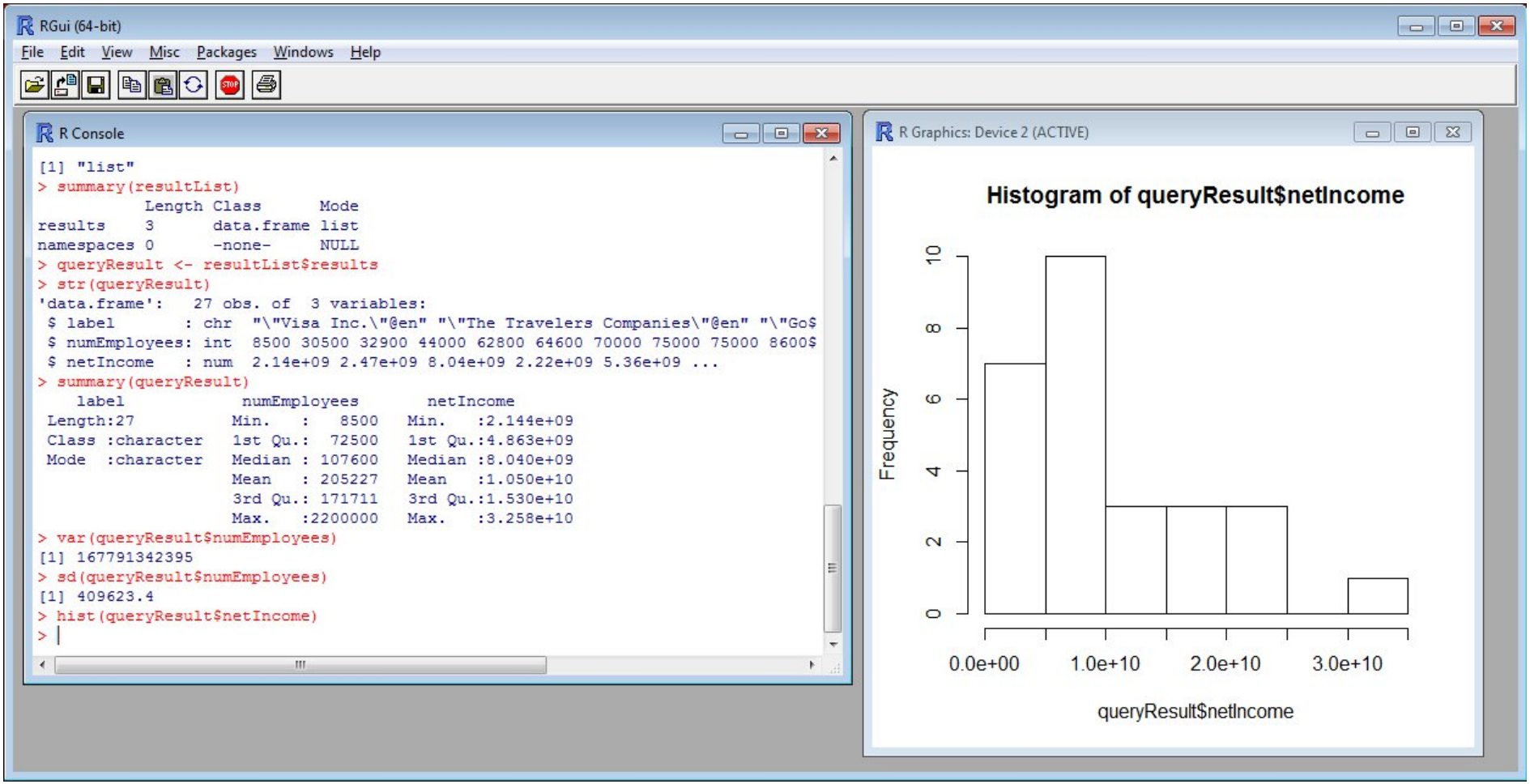
Khovd University

10.02.2021

Section 1

What is R?

What is R?


RGui (64-bit)
File Edit View Misc Packages Windows Help
R Console
[1] "list"
> summary(resultList)
Length Class Mode
results 3 data.frame list
namespaces 0 -none- NULL
> queryResult <- resultList$results
> str(queryResult)
'data.frame': 27 obs. of 3 variables:
$ label : chr "\"Visa Inc.\"@en" "\"The Travelers Companies\"@en" "\"Go$
$ numEmployees: int 8500 30500 32900 44000 62800 64600 70000 75000 75000 8600$
$ netIncome : num 2.14e+09 2.47e+09 8.04e+09 2.22e+09 5.36e+09 ...
> summary(queryResult)
label numEmployees netIncome
Length:27 Min. : 8500 Min. :2.144e+09
Class :character 1st Qu.: 72500 1st Qu.:4.863e+09
Mode :character Median : 107600 Median :8.040e+09
Mean : 205227 Mean :1.050e+10
3rd Qu.: 171711 3rd Qu.:1.530e+10
Max. :2200000 Max. :3.258e+10
> var(queryResult$numEmployees)
[1] 167791342395
> sd(queryResult$numEmployees)
[1] 409623.4
> hist(queryResult$netIncome)
R Graphics: Device 2 (ACTIVE)
Histogram of queryResult$netIncome
Frequency
0
2
4
6
8
10
0.0e+00
1.0e+10
2.0e+10
3.0e+10
queryResult$netIncome

Section 2

R Commands

R Commands

```
3+4*12
```

```
## [1] 51
```

- Place cursor into console
- Write command
- Press ↵
- R interprets the command and returns it's computes output
- Repeat previous command: ↑

R Commands

```
a <- 3+4*12
a
```

```
## [1] 51
```

R Commands

```
b <- c(1, 2, 3, 4, 5)
d <- 1:5
b
```

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

```
b == d
```

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

```
a*b
```

```
## [1]  51 102 153 204 255
```

```
a^b
```

```
## [1]         51       2601     132651    6765201 345025251
```

Section 3

Working Principles

Working Principles

- R commands can be send via console or script
- Console:
  - Experiment with commands
  - R documentation: e.g. `?range`
  - Look up history of commands (↑ or ↓)
  - Installing packages: `install.packages("ggplot2")`
- R-Scripts: File » New Script
  - Make your analysis shareable and replicable
  - Documentation of analysis
  - Using comments: `#here is a comment`
  - Saving analysis: File » Save or Ctrl + s and save as `*.R`

Subsection 1

Using the RGui

Using the RGui

- Chose clean windowing layout: Windows » Tile Vertically
- Select code with cursor and press Ctrl + R executes the code
- . . . or right-click on selection and chose "Run line or selection"
- Clear console: Ctrl + L

Section 4

R Data Types

R Data Types

```
s <- "Hello World!"
s
```

```
## [1] "Hello World!"
```

```
class(s)
```

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

R Data Types

```
a
```

```
## [1] 51
```

```
class(a)
```

```
## [1] "numeric"
```

R Data Types

```
m <- matrix(data = c(1, 2, 3, 4),
            nrow = 2,
            ncol = 2)
m
```

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

```
class(m)
```

```
## [1] "matrix"
```

R Data Types

```
v <- c(2, 2)
m %*% v
```

```
##      [,1]
## [1,]    8
## [2,]   12
```

R Data Types

```
a <- runif(15)
summary(a)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.02798 0.15701 0.36035 0.35852 0.53089 0.83603
```

```
length(a)
```

```
## [1] 15
```

R Data Types

```
violation <- a > 0.8
```

```
class(violation)
```

```
## [1] "logical"
```

```
summary(violation)
```

```
##    Mode   FALSE    TRUE
## logical      14       1
```

```
mean(violation)
```

```
## [1] 0.06666667
```

R Data Types

```
my_list <- list(measurements = a,
        violations = a > 0.8,
        percentage = mean(a > 0.8))
```

```
class(my_list)
```

```
## [1] "list"
```

R Data Types

```
# This is a comment: you can describe something here
my_list
```

```
## $measurements
##  [1] 0.13535976 0.03934248 0.02798057 0.16330318
##  [5] 0.64890211 0.39063200 0.58279991 0.15071223
##  [9] 0.47897123 0.47238432 0.19041395 0.36035138
## [13] 0.83602695 0.62169008 0.27897830
##
## $violations
##  [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [9] FALSE FALSE FALSE FALSE  TRUE FALSE FALSE
##
## $percentage
## [1] 0.06666667
```

R Data Types

```
# You can access elements from a list by the $-operator
my_list$percentage
```

```
## [1] 0.06666667
```

```
# ... or by enumeration
my_list[3]
```

```
## $percentage
## [1] 0.06666667
```

```
# ... or by using names
my_list[["percentage"]]
```

```
## [1] 0.06666667
```

Section 5

R Functions

R Functions

- R and it's packages provide many functions for computations, e.g. `mean(x, na.rm = TRUE)`
- Functions can also be defined by the user

```
greet <- function() {
    return("Hello!")
}

greet()
```

```
## [1] "Hello!"
```

R Functions

```
greet <- function(n = 1) {
    return(paste0(rep("Hello!", times = n), collapse = " "))
}
```

```
greet()
```

```
## [1] "Hello!"
```

```
greet(4)
```

```
## [1] "Hello! Hello! Hello! Hello!"
```

R Functions

```
example(mean)
```

```
##
## mean> x <- c(0:10, 50)
##
## mean> xm <- mean(x)
##
## mean> c(xm, mean(x, trim = 0.10))
## [1] 8.75 5.50
```

Section 6

Exercises

Subsection 1

Using R as calculator

Circles

1. Calculate the area of a circle $A = 2\pi r^2$ with $r = 2$ (Qian, 2016)
2. Write the circle area formula as a function with named as `circle` with parameter `r` and and calculate the area for `r <- seq(0, 3, 0.1)`
3. **Extension:** Extend your function to return a named `list` that contains the area and the circumference for a given parameter vector `r`

Normal Probability Density Function

1. Calculate the density of the normal probability distribution function $x \sim \mathcal{N}(2, 1.25)$ (mean and standard deviation) at `x <- seq(0, 4, 0.5)` by using the normal probability density formula $\left(\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}\right)$, and verify your result by using the function `dnorm` (Qian, 2016).

Subsection 2

Solutions

Circles

```
circle <- function(r) {
    result <- list(area = 2 * pi * r^2,
                   circumference = 2 * pi * r)
    return(result)
}

circle(r = seq(0, 3, 0.5))
```

```
## $area
## [1]  0.000000  1.570796  6.283185 14.137167 25.132741
## [6] 39.269908 56.548668
##
## $circumference
## [1]  0.000000  3.141593  6.283185  9.424778 12.566371
## [6] 15.707963 18.849556
```

Normal Probability Density Function

```
npdf <- function(x, avg, stdev) {
    result <- 1.0 / (sqrt(2*pi*stdev^2)) *
        exp(-(x - avg)^2 / (2.0 * stdev^2))
    return(result)
}

npdf(x = seq(0, 4, 0.5), avg = 2, stdev = 1.25)
```

```
## [1] 0.08873667 0.15534884 0.23175324 0.29461611
## [5] 0.31915382 0.29461611 0.23175324 0.15534884
## [9] 0.08873667
```

```
    dnorm(x = seq(0, 4, 0.5), mean = 2, sd = 1.25)
```

```
## [1] 0.08873667 0.15534884 0.23175324 0.29461611
## [5] 0.31915382 0.29461611 0.23175324 0.15534884
## [9] 0.08873667
```

Song S Qian. *Environmental and ecological statistics with R*. CRC press, 2016.