CSSCR R Workshop

Dadmehr Didgar

2024-1-22

Introduction to R using R Studio

R is a versatile programming language widely used for statistical analysis and data visualization. Its comprehensive libraries enable effective data manipulation, making it essential for researchers and data scientists. This workshop provides a foundational understanding of R's syntax and functions, focusing on data handling and graphical representation.

Simple Calculation

```r
3+4
```

```
## [1] 7
```

```r
2/3
```

```
## [1] 0.6666667
```

```r
5*2
```

```
## [1] 10
```

`x = 1` vs. `x <- 1`

In R, `x = 1` and `x <- 1` both assign the value `1` to `x`. The traditional `<-` operator is preferred for variable assignments due to its clarity and readability, while `=` is commonly used for specifying function arguments. Though functionally similar in most cases, `<-` is the conventional choice in R scripting for assignment operations.

Data Type & Assign Values

```r
2.1
```

```
## [1] 2.1
```

```r
F
```

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

```r
"Happy"
```

```
## [1] "Happy"
```

```r
"2"
```

```
## [1] "2"
```

```
a <- 3
## assign the character 2.1 to object called b
b <- "2.1"
## assign the character hello to object called bb
c <- "happy"
## assign the value of object a to object called c
d <- a
```

Built-in Mathematical Functions

R provides a variety of built-in mathematical functions. Here are a few examples:

- **Square Root**: The `sqrt()` function computes the square root of a number. For example, `sqrt(16)` will give 4.
- **Exponential**: The `exp()` function calculates the exponential of a number. For instance, `exp(1)` computes e^1, which is approximately 2.7182818.
- **Logarithm**: The `log()` function computes logarithms. `log(10)` gives the natural logarithm of 10, equal to 2.3025851.
- **Trigonometry**: Functions like `sin()`, `cos()`, and `tan()` are used for trigonometric calculations.

These functions exemplify the simplicity and power of R for mathematical computations.

```
pi
```

```
## [1] 3.141593
```

```
sqrt(4)
```

```
## [1] 2
```

```
exp(1) # Euler's number
```

```
## [1] 2.718282
```

```
log(1)
```

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

```
sin(3.1415)
```

```
## [1] 9.265359e-05
```

```
tan(3.1415)
```

```
## [1] -9.265359e-05
```

Types of Objects

```
# vector
numbers <- c(1,4,2)

colors <- c("lightgreen", "pink", "blue")

# data frame
demo_data <-    data.frame(
  gender = c("Male", "Male","Female"),
  height = c(152, 171.5, 165),
```

```r
  weight = c(81,93, 78),
  Age    = c(42,38,26)
  )

# list
mylist <- list(2.1, c(1,3,7), c("abc", "def"), demo_data)
```

If Clause in R

```r
x <- 5

if (x > 0) {
  print("x is positive")
} else {
  print("x is not positive")
}
```

```
## [1] "x is positive"
```

For Loop in R

```r
for (i in 1:5) {
  print(i)
  #print(paste("Value of i is", i))
}
```

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

Census Data

CSSCR_data is a small random sample (0.005%) of the census data for 2022

```r
# sampled_data <- data[sample(nrow(data), nrow(data) * 0.00005), ]

CSSCR_data <- data.frame(AGE = c(36, 66, 48, 84, 76, 61, 69, 33, 95, 61, 83, 69, 29, 73, 28, 80, 49, 48


CSSCR_data$INCTOT <- c(100000, 11600, 105000, 79400, 2300, 73000, 138000, 15000, 24000, 96900, 9300, 35
 14000, 45000, 46000, 65000, 39000, 36600, 75000, 56000, 25000, 170000, 16800, 18300, 400, 40000, 1900,
```

Data Analysis Essentials

The R code provided performs fundamental data analysis operations on the `CSSCR_data` dataset.

```r
dim(CSSCR_data)
```

```
## [1] 124   2
```

```r
mean(CSSCR_data$INCTOT)
```

```
## [1] 50146.37
```

```r
median(CSSCR_data$INCTOT)
```

```
## [1] 30550
```

```r
mean(CSSCR_data$AGE)
```

```
## [1] 51.5
```

```r
median(CSSCR_data$AGE)
```

```
## [1] 53
```

Count

```r
#install.packages("dplyr")
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
```

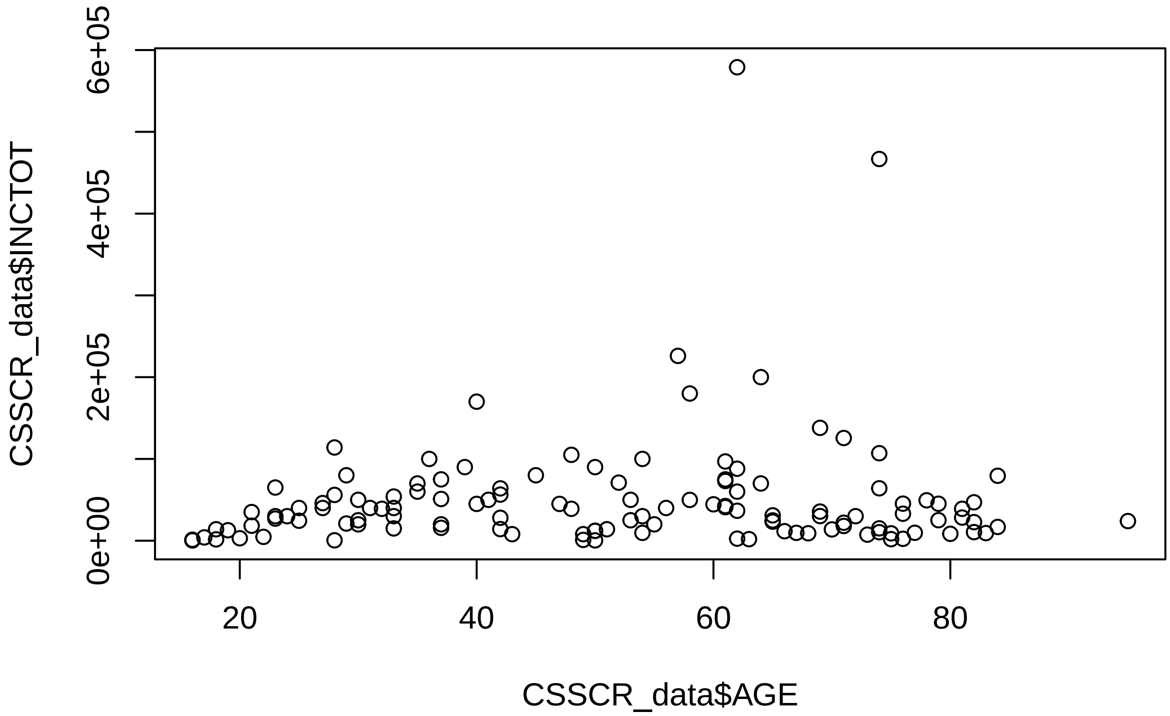
```r
count(CSSCR_data,AGE > 30)
```

```
##   AGE > 30  n
## 1    FALSE 27
## 2     TRUE 97
```

```r
help("count")
?count
```

Simple Plot

```r
plot(CSSCR_data$AGE, CSSCR_data$INCTOT)
```



Less Simple Plot

```
plot(CSSCR_data$AGE, CSSCR_data$INCTOT,
     main = "Age vs. Income",
     xlab = "Age",
     ylab = "Income",
     ylim = c(0,150000),
     xlim = c(18,85))
```


```
?plot
```

```
## Help on topic 'plot' was found in the following packages:
##
##   Package               Library
##   graphics              /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library
##   base                  /Library/Frameworks/R.framework/Resources/library
##
##
## Using the first match ...
```

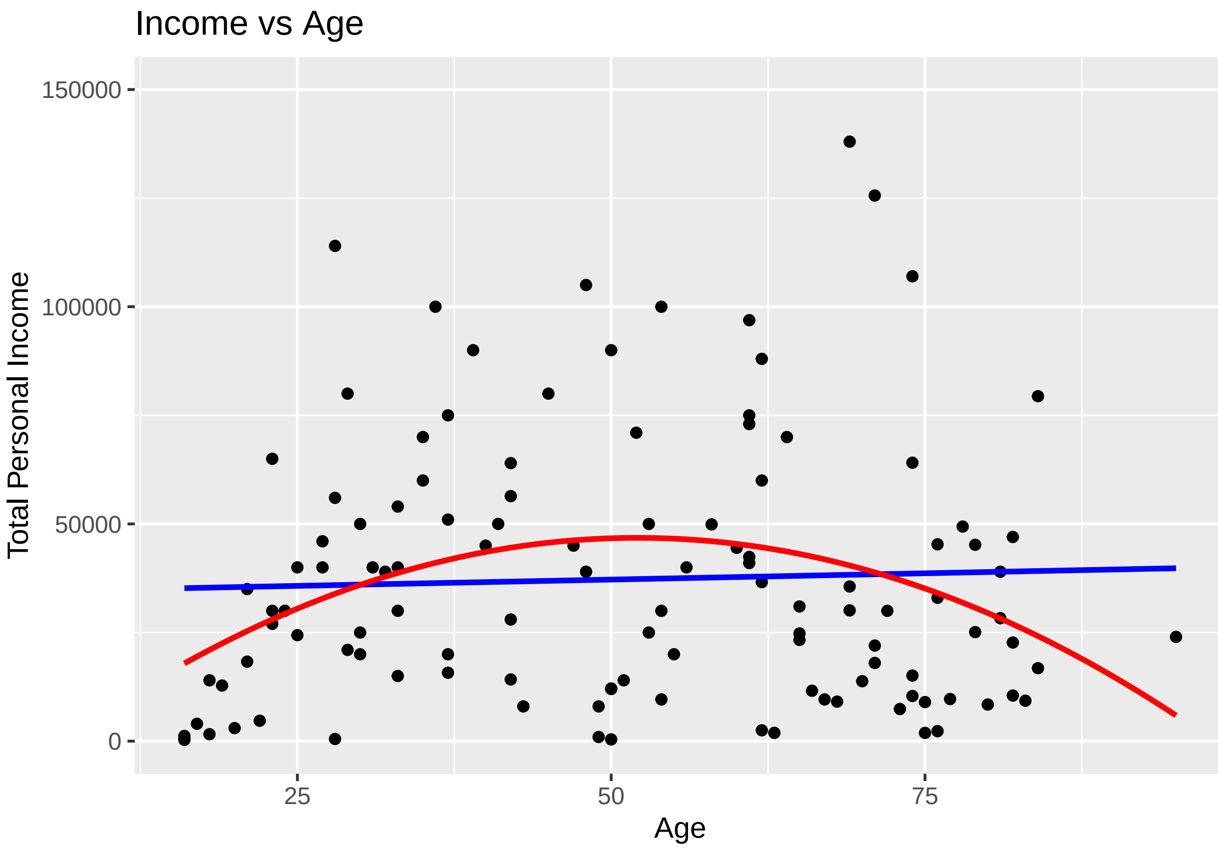
Fancy Plot

```
# Install and load the ggplot2 package
#install.packages("ggplot2")

library(ggplot2)

# Plotting
ggplot(CSSCR_data, aes(x = AGE, y = INCTOT)) +
  geom_point() +  # Scatter plot
  geom_smooth(method = "lm", se = FALSE, color = "blue") +  # Linear fit
  geom_smooth(method = "lm", formula = y ~ poly(x, 2), se = FALSE, color = "red") +  # Quadratic fit
  ylim(0, 150000) +
  labs(title = "Income vs Age", x = "Age", y = "Total Personal Income")
```

```
## `geom_smooth()` using formula = 'y ~ x'
```

```
## Warning: Removed 6 rows containing non-finite values (`stat_smooth()`).
## Removed 6 rows containing non-finite values (`stat_smooth()`).
```

```
## Warning: Removed 6 rows containing missing values (`geom_point()`).
```



Download Data, `setwd` & `read.csv`

National Obesity By State Data Analysis

The dataset on national obesity by state can be found at Data.gov. Follow the link to download the CSV file and save it in your working directory.

```r
getwd()
```

```
## [1] "/Users/dadmehr/R"
```

```r
#setwd("/Users/dadmehr/R")

# Destination file path (where you want to save the file)
obesity_data <- read.csv("National_Obesity_By_State.csv")

# View the first few rows of the dataset
head(obesity_data)
```

```
##   FID       NAME Obesity SHAPE_Length   SHAPE_Area
## 1   1      Texas    32.4     15408322 7.672329e+12
## 2   2 California    24.2     14518698 5.327809e+12
## 3   3   Kentucky    34.6      6346699 1.128830e+12
## 4   4    Georgia    30.7      5795596 1.652980e+12
## 5   5  Wisconsin    30.7      6806782 1.567816e+12
## 6   6     Oregon    30.1      7976011 3.178446e+12
```

```r
tail(obesity_data, n = 10)
```

```
##    FID          NAME Obesity SHAPE_Length   SHAPE_Area
## 43  43       Arizona    28.4      8044184 3.562686e+12
## 44  44    New Mexico    28.8      8075167 3.622933e+12
## 45  45      Maryland    28.9      5850363 3.039432e+11
## 46  46      Delaware    29.7      1383604 5.908110e+10
## 47  47  Pennsylvania    30.0      5024348 1.288452e+12
## 48  48        Kansas    34.2      6540498 2.340366e+12
## 49  49       Vermont    25.1      2653732 2.789313e+11
## 50  50    New Jersey    25.6      2599119 2.246065e+11
## 51  51  North Dakota    31.0      5872756 2.013152e+12
## 52  52 New Hampshire    26.3      2674767 2.705294e+11
```

```r
?tail
summary(obesity_data)
```

```
##       FID            NAME              Obesity       SHAPE_Length
##  Min.   : 1.00   Length:52          Min.   :20.20   Min.   :       0
##  1st Qu.:13.75   Class :character   1st Qu.:26.25   1st Qu.: 5022132
##  Median :26.50   Mode  :character   Median :29.80   Median : 6445438
##  Mean   :26.50                      Mean   :29.29   Mean   : 6294282
##  3rd Qu.:39.25                      3rd Qu.:31.48   3rd Qu.: 7747383
##  Max.   :52.00                      Max.   :36.20   Max.   :15408322
##    SHAPE_Area
##  Min.   :0.000e+00
##  1st Qu.:8.300e+11
##  Median :1.492e+12
##  Mean   :1.724e+12
##  3rd Qu.:2.246e+12
##  Max.   :7.672e+12
```

```r
?summary
```

Pipe Operator %>%

The pipe operator `%>%` from the `magrittr` package, which is integrated into `dplyr`, allows for clearer and more intuitive syntax when chaining together multiple data manipulation commands. It passes the result of one expression as the first argument to the next expression, making your code more readable and concise.

Why Use the Pipe Operator?

- **Readability**: Code is more readable and easier to understand.
- **Simplification**: Reduces the need for intermediate variables.
- **Flow of Operations**: Reflects the logical flow of operations, making complex operations easier to follow.

With Pipe

```r
# Load the dplyr package
library(dplyr)

obesity_data %>%
  filter(Obesity > 34 ) %>%
  arrange(desc(Obesity))
```

```
##   FID          NAME Obesity SHAPE_Length   SHAPE_Area
## 1   9     Louisiana    36.2      7383857 1.355094e+12
## 2  25   Mississippi    35.6      5834202 1.327853e+12
## 3  31 West Virginia    35.6      5374280 6.851674e+11
## 4  38       Alabama    35.6      5750658 1.442807e+12
## 5   3      Kentucky    34.6      6346699 1.128830e+12
## 6  24      Arkansas    34.5      5707634 1.488699e+12
## 7  48        Kansas    34.2      6540498 2.340366e+12
```

Without Pipe

```r
arrange(filter(obesity_data, Obesity > 34), desc(Obesity))
```

```
##   FID          NAME Obesity SHAPE_Length   SHAPE_Area
## 1   9     Louisiana    36.2      7383857 1.355094e+12
## 2  25   Mississippi    35.6      5834202 1.327853e+12
## 3  31 West Virginia    35.6      5374280 6.851674e+11
## 4  38       Alabama    35.6      5750658 1.442807e+12
## 5   3      Kentucky    34.6      6346699 1.128830e+12
## 6  24      Arkansas    34.5      5707634 1.488699e+12
## 7  48        Kansas    34.2      6540498 2.340366e+12
```