Code for QSS tidyverse Chapter 1: Introduction

Kosuke Imai and Nora Webb Williams

First Printing

Introduction

Overview of the Book

How to Use this Book

Introduction to R and the tidyverse

Arithmetic Operations: R as a Calculator

```
5 + 3
```

```
## [1] 8
```

```
5 - 3
```

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

```
5 / 3
```

```
## [1] 1.666667
```

```
5 ^ 3
```

```
## [1] 125
```

```
5 * (10 - 3)
```

```
## [1] 35
```

```
sqrt(4)
```

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

R Scripts

```r
## This is the start of an R Script
## The heading provides some information about the file

## File name: testing_arithm.R
## Author: Kosuke Imai and Nora Webb Williams
## Purpose: Practicing basic math commands and commenting in R
##

5 - 3 # What is 5 minus three?
5 / 3
5 ^ 3
5 * (10 - 3) # A bit more complex
sqrt(4) # This function will take the square root of a number
```

Loading Packages

```r
install.packages("devtools") # install the package
library(devtools) # load the package
## install a package from github
devtools::install_github("kosukeimai/qss-package", build_vignettes = TRUE)
## You may need to allow R to update/install additional packages
```

```r
## load the qss package
library("qss")
```

```r
library(tidyverse) #if this command does not work, remember to install the package
```

Objects

```r
result <- 5 + 3
result
```

```
## [1] 8
```

```r
print(result)
```

```
## [1] 8
```

```r
result <- 5 - 3
result
```

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

```r
Result
```

```
## Error in eval(expr, envir, enclos): object 'Result' not found
```

```
kosuke <- "instructor"
kosuke
```

```
## [1] "instructor"
```

```
kosuke <- "instructor and author"
kosuke
```

```
## [1] "instructor and author"
```

```
Result <- "5"
Result
```

```
## [1] "5"
```

```
Result / 3
```

```
## Error in Result/3: non-numeric argument to binary operator
```

```
sqrt(Result)
```

```
## Error in sqrt(Result): non-numeric argument to mathematical function
```

```
result
```

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

```
class(result)
```

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

```
Result
```

```
## [1] "5"
```

```
class(Result)
```

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

```
class(sqrt)
```

```
## [1] "function"
```

Vectors

```r
world.pop <- c(2525779, 3026003, 3691173, 4449049, 5320817, 6127700, 6916183)
world.pop
```

```
## [1] 2525779 3026003 3691173 4449049 5320817 6127700 6916183
```

```r
pop.first <- c(2525779, 3026003, 3691173)
pop.second <- c(4449049, 5320817, 6127700, 6916183)
pop.all <- c(pop.first, pop.second)
pop.all
```

```
## [1] 2525779 3026003 3691173 4449049 5320817 6127700 6916183
```

```r
world.pop[2]
```

```
## [1] 3026003
```

```r
world.pop[c(2, 4)]
```

```
## [1] 3026003 4449049
```

```r
world.pop[c(4, 2)]
```

```
## [1] 4449049 3026003
```

```r
world.pop[-3]
```

```
## [1] 2525779 3026003 4449049 5320817 6127700 6916183
```

```r
pop.million <- world.pop / 1000
pop.million
```

```
## [1] 2525.779 3026.003 3691.173 4449.049 5320.817 6127.700
## [7] 6916.183
```

```r
pop.rate <- world.pop / world.pop[1]
pop.rate
```

```
## [1] 1.000000 1.198047 1.461400 1.761456 2.106604 2.426063
## [7] 2.738238
```

```r
pop.increase <- world.pop[-1] - world.pop[-7]
percent.increase <- (pop.increase / world.pop[-7]) * 100
percent.increase
```

```
## [1] 19.80474 21.98180 20.53212 19.59448 15.16464 12.86752
```

```
percent.increase[c(1, 2)] <- c(20, 22)
percent.increase
```

```
## [1] 20.00000 22.00000 20.53212 19.59448 15.16464 12.86752
```

Functions

```
length(world.pop)
```

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

```
min(world.pop)
```

```
## [1] 2525779
```

```
max(world.pop)
```

```
## [1] 6916183
```

```
range(world.pop)
```

```
## [1] 2525779 6916183
```

```
mean(world.pop)
```

```
## [1] 4579529
```

```
sum(world.pop) / length(world.pop)
```

```
## [1] 4579529
```

```
year <- seq(from = 1950, to = 2010, by = 10)
year
```

```
## [1] 1950 1960 1970 1980 1990 2000 2010
```

```
seq(to = 2010, by = 10, from = 1950)
```

```
## [1] 1950 1960 1970 1980 1990 2000 2010
```

```
seq(from = 2010, to = 1950, by = -10)
```

```
## [1] 2010 2000 1990 1980 1970 1960 1950
```

```r
2008:2012
```

```
## [1] 2008 2009 2010 2011 2012
```

```r
2012:2008
```

```
## [1] 2012 2011 2010 2009 2008
```

```r
names(world.pop)
```

```
## NULL
```

```r
names(world.pop) <- year
names(world.pop)
```

```
## [1] "1950" "1960" "1970" "1980" "1990" "2000" "2010"
```

```r
world.pop
```

```
##    1950    1960    1970    1980    1990    2000    2010
## 2525779 3026003 3691173 4449049 5320817 6127700 6916183
```

```r
myfunction <- function(input1, input2, ..., inputN) {

    DEFINE 'output' USING INPUTS

    return(output)
}
```

```r
my.summary <- function(x){ # function takes one input, x
  s.out <- sum(x)
  l.out <- length(x)
  m.out <- s.out / l.out
  out <- c(s.out, l.out, m.out) # define the output
  names(out) <- c("sum", "length", "mean") # add labels
  return(out) # end function by calling output
}
z <- 1:10 # z is a vector from 1 to 10
my.summary(z) # run my.summary function on z
```

```
##    sum length   mean
##   55.0   10.0    5.5
```

```r
my.summary(world.pop) # run my.summary function on world.pop
```

```
##      sum   length     mean
## 32056704        7  4579529
```

Data Files: Loading and Subsetting

```r
getwd() # Check what your current working directory is
setwd("qss/INTRO") # Set your working directory with a path
getwd() # Check that you changed your working directory
```

```r
## If your working directory is where the .csv file is stored
UNpop <- read_csv("UNpop.csv")
class(UNpop) # What type of object is UNpop?
```

```r
load("UNpop.RData")
```

```r
## Specifying a relative path to find and read in UNpop.csv
## Will overwrite previously loaded UNpop object
UNpop <- read_csv("INTRO/UNpop.csv")
class(UNpop) # what type of object is UNpop?
```

```r
## Load the package
library(qss)
## Load the UN pop data
## Will overwrite previously loaded UNpop object
data(UNpop, package = "qss")
```

```r
names(UNpop)
```

```
## [1] "year"      "world.pop"
```

```r
nrow(UNpop)
```

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

```r
ncol(UNpop)
```

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

```r
dim(UNpop)
```

```
## [1] 7 2
```

```r
UNpop$world.pop
```

```
## [1] 2525779 3026003 3691173 4449049 5320817 6127700 6916183
```

```r
## subset all rows for the column called "world.pop" from the UNpop data
UNpop[, "world.pop"]
```

```
## [1] 2525779 3026003 3691173 4449049 5320817 6127700 6916183
```

```
## subset the first three rows (and all columns)
UNpop[c(1, 2, 3),]
```

```
##   year world.pop
## 1 1950   2525779
## 2 1960   3026003
## 3 1970   3691173
```

```
## subset the first three rows of the "year" column
UNpop[1:3, "year"]
```

```
## [1] 1950 1960 1970
```

```
## Subset the first three rows of UNpop with tidyverse
slice(UNpop, n = 1:3)
```

```
##   year world.pop
## 1 1950   2525779
## 2 1960   3026003
## 3 1970   3691173
```

```
## Extract/subset the world.pop variable (column)
select(UNpop, world.pop)
```

```
##   world.pop
## 1   2525779
## 2   3026003
## 3   3691173
## 4   4449049
## 5   5320817
## 6   6127700
## 7   6916183
```

```
## Base R subset the first three rows of the year variable
UNpop[1:3, "year"]
```

```
## [1] 1950 1960 1970
```

```
## or in tidyverse, combining slice() and select()
select(slice(UNpop, 1:3), year)
```

```
##   year
## 1 1950
## 2 1960
## 3 1970
```

```
UNpop %>% # take the UNpop data we have loaded, and then...
  slice(1:3) %>% # subset the first three rows, and then...
  select(year) # subset the year column
```

```
##   year
## 1 1950
## 2 1960
## 3 1970
```

```
UNpop %>%
  slice(seq(1, n(), by = 2)) %>% # using a sequence from 1 to n()
  select(world.pop)
```

```
##   world.pop
## 1   2525779
## 2   3691173
## 3   5320817
## 4   6916183
```

```
UNpop %>%
  filter(row_number() %% 2 == 1) %>%
  select(world.pop)
```

```
##   world.pop
## 1   2525779
## 2   3691173
## 3   5320817
## 4   6916183
```

```
pop.1970 <- UNpop %>% # take the UNpop data and then....
  filter(year == 1970) %>% # subset rows where the year variable is equal to 1970
  select(world.pop) %>% # subset just the world.pop column
  pull() # return a vector, not a tibble

## Print the vector to the console to see it
print(pop.1970)
```

```
## [1] 3691173
```

Data Files: Adding Variables

```
UNpop.mill <- UNpop %>% # create a new tibble from UNpop
  mutate(world.pop.mill = world.pop / 1000) %>% # create a new variable, world.pop.mill
  select(-world.pop) # drop the original world.pop column
```

```
## Adding a nonsense variable to the UNpop.mill data
UNpop.mill <- UNpop.mill %>%
  mutate(nonsense.var = world.pop.mill / year)
```

```
## Adding a variable with if_else
UNpop.mill <- UNpop.mill %>%
  mutate(after.1980 = if_else(year >= 1980, 1, 0))
```

```
## Creating a vector of the years of interest
specific.years <- c(1950, 1980, 2000)

## Adding a variable with if_else and %in%
UNpop.mill <- UNpop.mill %>%
  mutate(year.of.interest = if_else(year %in% specific.years, 1, 0))
```

Data Frames: Summarizing

```
summary(UNpop.mill)
```

```
##       year      world.pop.mill  nonsense.var
##  Min.   :1950   Min.   :2526   Min.   :1.295
##  1st Qu.:1965   1st Qu.:3359   1st Qu.:1.709
##  Median :1980   Median :4449   Median :2.247
##  Mean   :1980   Mean   :4580   Mean   :2.305
##  3rd Qu.:1995   3rd Qu.:5724   3rd Qu.:2.869
##  Max.   :2010   Max.   :6916   Max.   :3.441
##    after.1980     year.of.interest
##  Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.0000   1st Qu.:0.0000
##  Median :1.0000   Median :0.0000
##  Mean   :0.5714   Mean   :0.4286
##  3rd Qu.:1.0000   3rd Qu.:1.0000
##  Max.   :1.0000   Max.   :1.0000
```

```
mean(UNpop.mill$world.pop.mill)
```

```
## [1] 4579.529
```

```
## Add a row where values for all columns is NA
UNpop.mill.wNAs <- UNpop.mill %>%
  add_row(year = NA, world.pop.mill = NA,
          nonsense.var = NA, after.1980 = NA,
          year.of.interest = NA)
## Take the mean of world.pop.mill (returns NA)
mean(UNpop.mill.wNAs$world.pop.mill)
```

```
## [1] NA
```

```
## Take the mean of world.pop.mill (ignores the NA)
mean(UNpop.mill.wNAs$world.pop.mill, na.rm = TRUE)
```

```
## [1] 4579.529
```

```
UNpop.mill %>%
  summarize(mean.pop = mean(world.pop.mill),
            median.pop = median(world.pop.mill))
```

```
##   mean.pop median.pop
## 1 4579.529   4449.049
```

```
UNpop.mill %>%
  group_by(after.1980) %>% # create subset group for each value of after.1980
  summarize(mean.pop = mean(world.pop.mill)) # calculate mean for each group
```

```
## # A tibble: 2 x 2
##   after.1980 mean.pop
##        <dbl>    <dbl>
## 1          0    3081.
## 2          1    5703.
```

Saving Objects

```
save.image("qss/INTRO/Chapter1.RData")
```

```
save(UNpop, file = "Chapter1.RData")
save(world.pop, year, file = "qss/INTRO/Chapter1.RData")
```

```
load("Chapter1.RData")
```

```
write_csv(UNpop, path = "INTRO/UNpop.csv")
```

Loading Data in Other Formats

```
## install packages -- note the syntax for multiple
## packages at once
install.packages(c("foreign", "haven", "rio"))
```

```
library("foreign") # load package
library("haven")
library("rio")
```

```
read.dta("UNpop.dta")
read.spss("UNpop.sav")
```

```
UNpop_dta_url <- "https://github.com/kosukeimai/qss/raw/master/INTRO/UNpop.dta"

UNpop <- read_dta(UNpop_dta_url)

## reading in with import; note that each UNpop <- will override the prior object
UNpop <- import("https://github.com/kosukeimai/qss/raw/master/INTRO/UNpop.csv")

UNpop <- import("https://github.com/kosukeimai/qss/raw/master/INTRO/UNpop.RData")

UNpop <- import("https://github.com/kosukeimai/qss/raw/master/INTRO/UNpop.dta")
```

```
write.dta(UNpop, file = "UNpop.dta")
write_dta(UNpop, "UNpop.dta")
```

Programming and Learning Tips

```
##
## File: UNpop.R
## Author: Kosuke Imai and Nora Webb Williams
## The code loads the UN population data, adds a variable,
## and saves the data as a STATA file
##

## Load the necessary packages
library(haven)
library(tidyverse)
library(qss)

## Load the UN pop data
data(UNpop, package = "qss")

## Replace the raw population with the population in millions
UNpop <- UNpop %>%
  mutate(world.pop = world.pop / 1000 )

## Save the data as a .dta file
write_dta(UNpop, path = "UNpop.dta")
```

```
source("UNpop.R")
```