Minilab 2a Worksheet

Tidy Data

In this minilab, we will use R to create some simple datasets in the form of a *tibble* (a "tidy data table" in the tidyverse).

1. Tennis, Tables and Tibbles

Data in R is generally stored as a *data frame* which is a two-dimensional data table with data organised into rows and columns. A data frame is really just a list in which each element of the list is a vector of the same length. Each vector represents a column, not a row. The elements at corresponding indices in the vectors are considered part of the same row. This structure makes sense because each row may have different types of data, such as a person's name (string) and height (number), and vector elements must all be of the same type.

In the tidyverse, a *tibble* (tidy data table) is a data frame with some additional useful behaviour.

(1) We can create a tibble one-column-at-a-time. Copy-and-paste the following R code into a new R Script and run it. *Be careful about the page break and header.*

```
# Top 10 Male and Female Tennis players
# as at 20 January 2020
# Data from: https://www.atptour.com/en/rankings/singles
#            https://www.wtatennis.com/rankings/singles
#            https://www.eurosport.co.uk/tennis/

library(tidyverse)
name = c("Nadal","Djokovic","Federer","Medvedev","Theim",
         "Tsitsipas","Zverev","Berrettini","Bautista Agut","Monfils",
         "Barty","Pliskova","Halep","Osaka","Svitolina",
         "Andreescu","Bencic","Kvitova","Williams","Bertens")
rank = c(1:10,1:10)
age  = c(33,32,38,23,26,21,22,23,31,33,
         23,27,28,22,25,19,22,29,38,28)
height = c(1.85,1.88,1.85,1.98,1.85,1.93,1.98,1.96,1.83,1.93,
           1.66,1.86,1.68,1.80,1.74,1.70,1.75,1.82,1.75,1.82)
weight = c(85,77,85,83,79,89,90,95,75,85,
           62,72,60,69,60,60,63,68,72,74)
gender = c(rep("M",10),rep("F",10))
tennis = tibble(name,rank,age,height,weight,gender)
print(tennis)
```

- This way of creating a tibble (by typing in all the data) emphasises that a data frame is a list-of-vectors.

- However, it is quite difficult to debug since the data belonging to one individual is one element out of each vector.

(2) When entering data by typing in all the data, it is better to create a tibble from text that looks like a table (using the tribble() function, note the "r" in "tribble"). Notice how the columns of data nicely line up. All lines starting with a "#" are a comment.

```
tennis = tribble(
  ~name, ~rank, ~age, ~height, ~weight, ~gender,
  #--------------|---|---|----|---|----|
  "Nadal",            1, 33, 1.85, 85, "M",
  "Djokovic",         2, 32, 1.88, 77, "M",
  "Federer",          3, 38, 1.85, 85, "M",
  "Medvedev",         4, 23, 1.98, 83, "M",
  "Theim",            5, 26, 1.85, 79, "M",
  "Tsitsipas",        6, 21, 1.93, 89, "M",
  "Zverev",           7, 22, 1.98, 90, "M",
  "Berrettini",       8, 23, 1.96, 95, "M",
  "Bautista Agut",    9, 31, 1.83, 75, "M",
  "Monfils",         10, 33, 1.93, 85, "M",
  "Barty",            1, 23, 1.66, 62, "F",
  "Pliskova",       2, 27, 1.86, 72, "F",
  "Halep",          3, 28, 1.68, 60, "F",
  "Osaka",          4, 22, 1.80, 69, "F",
  "Svitolina",      5, 25, 1.74, 60, "F",
  "Andreescu",      6, 19, 1.70, 60, "F",
  "Bencic",         7, 22, 1.75, 63, "F",
  "Kvitova",        8, 29, 1.82, 68, "F",
  "Williams",       9, 38, 1.75, 72, "F",
  "Bertens",       10, 28, 1.82, 74, "F"
)
print(tennis)
```

(3) However a tibble is entered into R, we can extract some useful information from a tibble.

```
nrow(tennis)
ncol(tennis)
colnames(tennis)
summary(tennis)
View(tennis) # with capital V
```

The last command opens the data frame in a spreadsheet-like viewer within RStudio.

(4) How can we access individual elements of a tibble?

- Since a tibble (data frame) is a list-of-vectors, it is possible to use dollar notation (mydata$column_name) or double-bracket notation (mydata[["column_name"]]) to access entire columns.

- However, R also uses a variation of single-bracket notation that allows you to filter for access individual cells in the table.

```
tennis[1,"name"]
tennis[1,"age"]
tennis[1,]
tennis[,"height"]
tennis$height
```

(5) A useful tool for exploratory data analysis is the *boxplot* (or box-and-whisker plot) which gives a quick way to visualise the distribution of data.

```
ggplot(tennis, aes(y=height)) +
  geom_boxplot()
```

The top and bottom of the *box* are the 75th and 25th percentiles, i.e., upper quartile (UQ) and lower quartile (LQ). The median (50th percentile) is shown by the horizontal line in the box. The *whiskers*, extend from the top and bottom to indicate the range for the bulk of the data. There are many variations of a boxplot. By default, ggplot2 extends the whiskers to the furthest point beyond the box, except that it will not go beyond 1.5 times the inter-quartile range ($IQR = UQ - LQ$) above the UQ or below the LQ. Any data points beyond that are called “outliers”.

(6) Boxplots are often used in side-by-side displays to compare distributions. Try to summarise (in words) what you observe in the following plot).

```
ggplot(tennis, aes(x=gender, y=height)) +
  geom_boxplot()
```

Exercise.

(a) Draw boxplots to compare top 10 male and female tennis players on the basis of age (instead of height). *Notice the “outlier” female indicated by a single dot — who is that? Check back to where we entered this dataset using the tribble() function.*

(b) Try adding “coord_flip()” to the plotting command (so that the x-axis is now the vertical axis and the y-axis is now the horizontal axis).

(c) Draw boxplots to compare the *body mass index* (BMI) of male and female tennis players (see https://en.wikipedia.org/wiki/Body_mass_index). Are any of these players “underweight” or “overweight”?

2. Tidy data: wide or narrow

A dataset is a collection of values, usually either numbers (if quantitative) or strings (if categorical or text). Every value belongs to a *variable* and an *observation*.

- It is easier to *describe relationships* between variables than between rows.

- However, it is easier to *make comparisons* between groups of observations than between groups of columns.

A dataset is called "messy" or "tidy" depending on how rows, columns and tables are matched up with observations, variables and types.

In **tidy** data: (1) each variable forms a column, (2) each observation forms a row, and (3) each value must have its own cell. *This is equivalent to Third Normal Form in relational databases.*

Wide data is easier for human eyeballs to digest, but narrow data is easier for computers to process.

(1) How can we take "wide" data and make it "narrow"?

```
W = tribble(
  ~person, ~age, ~weight, ~height,
  "Bob",      32,     128,     180,
  "Alice",    24,      86,     175,
  "Steve",    64,      95,     165
)
print(W)
N = pivot_longer(W,-person,names_to="variable",values_to="value")
print(N)
```

(2) How can we take narrow data and make it wide?

```
B = pivot_wider(N, names_from=variable, values_from=value)
print(B)
```

Exercise. Consider the dataset given in the table below.

Country	**Year**	**Value**
Algeria	2000	7
Algeria	2001	9
Brazil	2000	12
Brazil	2001	14
Colombia	2000	16
Colombia	2001	18

(a) Enter this dataset into R as a tibble using either tibble() or tribble().
(b) Use pivot_wider() to create a tibble with countries as rows and years as columns.
(c) Use pivot_wider() to create a tibble with years as rows and countries as columns.

Notice that we have seen three different ways to represent exactly the same data in a data table.

Summary

In this minilab, we have looked at how to enter a small dataset as a *tibble* (tidy data table in the tidyverse) by typing (entering whole columns or entering whole rows), and how to transform between a wide data format (useful for humans) and a narrow data format (useful for computers to process further).