For those who prefer the "$" notation, i.e. not to use pipes ("|>"), the following 2 statements are equivalent - they assume the dataset has been loaded onto Consumption).

Consumption |> select (Oil) |> summary()

summary(Consumption$Oil)

Similarly, the following are equivalent

Consumption |> pull (Oil) |> mean()

mean(Consumption$Oil)

---

title: "Lab 6 - Solutions"

author: "CMM020 class"

date: ""

output: pdf\_document

---

```{r}

rm(list = ls())

library(tidyverse)

```

Loading the data. Two files loaded. The first one is needed later on.

```{r}

browserdata <- read.csv("browserdata.csv", header=T,stringsAsFactors = T)

Consumption <- read.csv("consumptionTransposed.csv", header=T,stringsAsFactors = T)

```

# Checking the data

To get the dimensions of the consumption dataset.

````{r}

dim(Consumption)

```

To get a summary of the consumption dataset.

````{r}

summary(Consumption)

```

To display the structure of the consumption dataset.

```{r}

str(Consumption)

```

To display the class of the consumption dataset.

```{r}

class(Consumption)

```

To display the first elements of the consumption dataset.

```{r}

head(Consumption)

```

To display the last 4 elements of the consumption dataset.

```{r}

tail(Consumption, 4)

```

To get a summary.

```{r}

summary(Consumption)

```

### Applying some of the functions to the Oil column.

To check column Oil in the consumption dataset we first select the values of column Oil and put them in variable o.

```{r}

Consumption |> select (Oil)

```

To get a summary of the the Oil column.

````{r}

Consumption |> select (Oil) |> summary()

```

To display the structure of the the Oil column.

```{r}

Consumption |> select (Oil) |> str()

```

To display the class of the the Oil column.

```{r}

Consumption |> select (Oil) |> class()

```

To display the first elements of the the Oil column.

```{r}

# this is equivalent to head(o,6)

Consumption |> select (Oil) |>head()

```

To display the last 4 elements of the the Oil column.

```{r}

Consumption |> select(Oil) |>tail(4)

```

## Statistics

To calculate the mean (average) for gas

```{r}

Consumption |> pull(Gas) |> mean()

```

Note that for convenience the whole Gas column as been stored in variable gas.

To obtain the median for gas

```{r}

Consumption |> pull(Gas) |> median()

```

To obtain the sample standard deviation

```{r}

Consumption |> pull(Gas) |> sd()

```

To obtain the population standard deviation we first store the values in a variable.

```{r}

gas <- Consumption |> pull(Gas)

sd(gas)\*sqrt((length(gas)-1)/length(gas))

```

Note that length() returns the number of instances (rows)

To check the variance

```{r}

Consumption |> pull(Gas) |> var()

```

For minimum and maximum values

```{r}

Consumption |> pull(Gas) |> min()

Consumption |> pull(Gas) |> max()

```

And to obtain the range in one go

```{r}

Consumption |> pull(Gas) |> range()

```

To check the inter quartile range (3rd quantile minus first quantile)

```{r}

Consumption |> pull(Gas) |> IQR()

```

To obtain all the quantiles

```{r}

Consumption |> pull(Gas) |> quantile()

```

To get the fivenum values

```{r}

Consumption |> pull(Gas) |> fivenum()

```

There is a difference in the way the 1st and 3rd quartiles (2nd and 4th numbers in the 4 number summary) are calculated for quantiles and fivenum. When the median of the lower half is between two values, quantiles averages them whereas fivenums takes the larger value. Try

```{r}

# put numbers zero to 11 (in increments of 2) in y

y = seq(1, 11, by = 2)

y

```

```{r}

quantile(y)

```

```{r}

fivenum(y)

```

Quantile sorts the data in ascending order to find Q2.

For Q1, it takes the element 1+(n-1)/100.

For Q3 it calculates position p as follows:

p = 1+(n-1)\*3/100

If p is an integer, it takes the value at position p. If it isn't it aggregates the values of the 2 nearest positions(lp and hp) with values (lvalue and hvalue) as follows:

lvalue + (p - lp(hvalue-lvalue))

For the example, when calculating Q1 we have

p=2.25

lp = 2 with lvalue = 3

hp = 3 with lvalue = 5

Q1 = 3 + (2.25 - 2)(5 - 3) = 3.5

Fivenum sorts the data in ascending order

To find Q2: If the data set has an odd number of data, then the Q2 (the median) is the middle data point, otherwise is the average of the 2 middle points.

Q1 is the median for the data below Q2. Q3 is the median of the data above Q2.

## Exercise 1

The answer to this exercise depends on what the student focuses on.

For the whole dataset

```{r}

summary(browserdata)

dim(browserdata)

str(browserdata)

class(browserdata)

head(browserdata, 5)

tail(browserdata, 7)

```

We select a specific column, MinsPerDay, for the summary statistics below. Other columns can be selected.

```{r}

browserdata |> pull(MinsPerDay) |> mean()

browserdata |> pull(MinsPerDay) |> median()

browserdata |> pull(MinsPerDay) |> sd()

browserdata |> pull(MinsPerDay) |> var()

browserdata |> pull(MinsPerDay) |> range()

browserdata |> pull(MinsPerDay) |> IQR()

browserdata |> pull(MinsPerDay) |> min()

browserdata |> pull(MinsPerDay) |> max()

browserdata |> pull(MinsPerDay) |> mean()

browserdata |> pull(MinsPerDay) |> quantile()

browserdata |> pull(MinsPerDay) |> fivenum()

```

##Frequency counts

```{r}

counts <- browserdata |> pull(Browser) |> table()

counts

dim(browserdata)

```

In percertages (proportions)

```{r}

prop.table(counts)

```

Two-way tables

```{r}

freqBrowser <- xtabs(~ Browser+Membership, data= browserdata)

freqBrowser

```

To get probatbilities

```{r}

a <- prop.table(freqBrowser,1) # each row ads up to 1t

a

b <- prop.table(freqBrowser,2) # each column adds up to 1

b

```

## Bar plots

```{r}

# freqBrowser contains data from the two-way table obtained earlier.

# convert data to dataframe first

e <- as.data.frame(freqBrowser)

```

```{r}

# create plot

p <- ggplot(e, aes(x=Membership,fill=Browser))

p <- p + geom\_bar(aes(y=Freq), stat="identity")

p

```

Adding labels and changing the colour scheme.

```{r}

## With title and labels

p <- ggplot(e, aes(x=Membership, y=Freq, fill=Browser))

p <- p + geom\_bar(stat="identity")

p <- p + labs(title = "Browser use by membership type", x = "Membership type", y = "Frequency count")

p <- p + scale\_fill\_brewer(palette="Set1")

p <- p + theme\_classic()

p

```

With columns side by side instead of stacked.

```{r}

p <- ggplot(e, aes(x=Membership, y=Freq, fill=Browser) )

p <- p + geom\_bar(stat="identity", position=position\_dodge())

p <- p + labs(title = "Browser use by membership type", x = "membership type", y="Frequency count")

p <- p + scale\_fill\_brewer(palette="Dark2")

p <- p + theme\_classic()

p

```

## Histograms

```{r, warning=F}

p <- ggplot(browserdata, aes(x= MinsPerDay))

p <- p + geom\_histogram(aes(y=..density..), binwidth=20 )

p

```

Adding colour.

```{r}

p <- ggplot(browserdata, aes(x= MinsPerDay))

p <- p + geom\_histogram(col="red", fill="red" , binwidth=20)

p <- p + labs(x="Dataset 2", y="Counts")

p

```

Adding a density line plus labels

```{r}

p <- ggplot(browserdata, aes(x= MinsPerDay))

p <- p + geom\_histogram(aes(y=..density..),col="red", fill="red" , binwidth=20)

p <- p + labs(x="Average time spent online (mins/day)", y="Density")

p <- p + geom\_density(col="blue")

p <- p + theme\_classic()

p

```

## Dot plots

```{r}

p <- ggplot(browserdata, aes(x= Use))

p <- p + geom\_dotplot(binwidth=0.05)

p

```

Adding colour and a label

```{r}

p <- ggplot(browserdata, aes(x= Use))

p <- p + geom\_dotplot(col="red", fill="red" , binwidth=0.05)

p <- p + labs(x="Use level", y="")

p <- p + theme\_classic()

p

```

# Box plots

```{r}

p <- ggplot(browserdata, aes(y= MinsPerDay))

p <- p + geom\_boxplot()

p

```

Adding colour and labels.

```{r}

p <- ggplot(browserdata, aes(y= MinsPerDay))

p <- p + geom\_boxplot(col="blue", fill="lightblue")

p <- p + labs(title="User average minutes/day online", x="",y="")

p <- p + theme\_classic()

p

```

A boxplot per Browser type.

```{r}

p <- ggplot(browserdata, aes(x=Browser, y= MinsPerDay))

p <- p + geom\_boxplot(aes(col=factor(Browser)) )

p <- p + labs(title="User average minutes/day online by browser", x="",y="Average minutes per day")

p <- p + theme\_classic()

p

```

## Exercises 2, 3 and 4

The solutions to these exercises depend on the datasets selected by the student.

## Plots with functions

Previous labs have covered how to plot data but sometimes functions are plotted.

Try

```{r}

xlimits <- c(-pi,pi) ## pi = 3.14...

xlimits

```

To plot it

```{r}

p <- ggplot(data.frame(xlimits),aes(x=xlimits))

p <- p + stat\_function(fun=cos)

p

```

Where cos is the cosine function which is already predefined in R. You may wish to define your own functions. To define a function which returns the square of a number

```{r}

mysquare <- function(xval){xval\*xval}

# to test it with number 3

mysquare(3)

```

To plot it for numbers between 0 and 1.

```{r}

xlimits <- c(0,1)

p <- ggplot(data.frame(xlimits),aes(x=xlimits))

p <- p + stat\_function(fun=mysquare,col="steelblue")

p <- p + labs(x="x", y ="x squared", title="The Square Function")

p

```

## Plotting probability distributions

To plot a normal distribution

```{r}

p <- ggplot(data.frame(x=c(-3,3)),aes(x=x))

p <- p + stat\_function(fun=dnorm)

p <- p + labs(y="Density", title = "Normal Distribution")

p

```

To plot it cumulatively

```{r}

p <- ggplot(data.frame(x=c(-3,3)),aes(x=x))

p <- p + stat\_function(fun=pnorm)

p <- p + labs(y= "Cumulative Density", title = "Cumulative Normal Distribution")

p

```

To plot several distributions with different means and standard deviations.

```{r}

p <- ggplot(data.frame(x=c(-6,6)),aes(x=x))

p <- p + stat\_function(fun=dnorm,args=list(mean=0,sd=3), col="blue")

p <- p + stat\_function(fun=dnorm,args=list(mean=1,sd=2.3), col="red")

p <- p + stat\_function(fun=dnorm,args=list(mean=2,sd=2), col="green")

p <- p + stat\_function(fun=dnorm,args=list(mean=-0.5,sd=1.3), col="orange")

p <- p + labs(x="", y="")

p

```

The function rnorm returns randomly generated numbers. Each call returns different numbers unless the seed is the same.

```{r}

rnorm(10)

rnorm(10)

```

To fix the seed use set.seed.

```{r}

rnorm(10)

## this returns same ones as previous time seed was set to 123

set.seed(123)

rnorm(10)

```

```{r}

set.seed(123)

rnorm(10)

## this returns different ones

```

## Visualising univariate data

### Histogram

#### from a normal sampled distribution

If height of bars is density

```{r}

set.seed(123)

normal.sampled <- rnorm(1000)

p <- ggplot(NULL,aes(x=normal.sampled,y=..density..))

p <- p +geom\_histogram()

p

```

If height of vars is count

```{r}

p <- ggplot(NULL,aes(x=normal.sampled))

p <- p + geom\_histogram()

p

```

Using a frequency polygon instead of histogram

```{r}

p <- ggplot(NULL,aes(x=normal.sampled,y=..density..))

p <- p + geom\_freqpoly()

p

```

```{r}

p <- ggplot(NULL,aes(x=normal.sampled,y=..density..))

p <- p + geom\_density()

p

```

Using a density function instead.

```{r}

poisson50 <- rpois(n=50,lambda=3)

p50.df <- data.frame(poisson50)

p <- ggplot(p50.df,aes(x=poisson50))

p <- p + geom\_histogram(binwidth=1)

p

```

### Dotplot

```{r}

p <- ggplot(p50.df,aes(x=poisson50))

p <- p + geom\_dotplot(binwidth=0.5)

p

```

### Boxplots

Two different distributions. Differences are obvious from boxplots

```{r}

normal.m1.s1 <- rnorm(1000,mean=1,sd=1)

normal.m2.s3 <- rnorm(1000,mean=2,sd=3)

norm.compare <- data.frame(normal.m1.s1,normal.m2.s3)

head(norm.compare)

```

```{r}

bplot.data <- stack(norm.compare)

head(bplot.data)

```

```{r}

p <- ggplot(bplot.data,aes(x=ind,y=values,fill=ind))

p <- p + geom\_boxplot()

p

```