Workshop 5
Jointly summarising two variables

Workshop

The exercises today have fewer commands for you to blindly type in, than in previous weeks. For aspects covered in previous workshops/lectures consult the relevant teaching material. In other places hints are given.

Start R-Studio in the usual way and open a word document to include answers to any questions, graphics, etc.

Exercise 1 Contingency tables

In the package `carData` there is a data frame called `TitanicSurvival`. Load this package and read the help page for this data frame.[1] Note that there are missing data in the variable `age`

Obtain a frequency table to find out how many passengers survived and how many died.

Obtain the relative frequencies for passenger survival: Hint `prop.table(table(???))`.

Obtain a contingency table with the frequencies for `survived` and `passengerClass`.

Obtain the overall relative frequencies, the column and row relative frequencies for passenger survival and class: Hint see notes for Lecture 5.

Use the results from the commands above to answer the following questions.

(a) What proportion of passengers were 1st class and survived?

(b) Of the 1st class passengers what proportion survived?

(c) Of the 3rd class passengers what proportion survived?

(d) Of those passengers who survived what proportion were 3rd class passengers? Note that (c) and (d) are not the same thing.

[1] NB: There are several data sets with very similar Titanic data. This one has data on the passengers only, others include data on the crew as well.

Exercise 2 Bar charts

Plot a bar chart of passenger survival. Hint `barplot(table(???))`

Plot a bar chart of passenger survival and class.

Bar plot accepts the arguments `beside=TRUE` and `legend=TRUE`.
Investigate the effect of these two arguments.

What effect doies reversing the order of `survived` and `passengerClass` have on the diagram?

Exercise 3 `attach`, `detach` and the search path

You will come back to the Titanic data a bit later. This and the next exercise use the Prestige data which you have worked with over the last couple of weeks.

So far we have always referred to a variable using `objectname$variablename`. There is a way to get round typing `objectname` all the time, but you do need to be careful.

```
> mean(Prestige$income)
> attach(Prestige)
> mean(income)
```

The **`attach`** command means the variables in `Prestige` can be accessed without needing to name of the data frame.

To find out what does attach actually does, type:

```
> search()
 [1] ".GlobalEnv"         "Prestige"           "package:rsample"    "package:tidyr"
 [5] "package:xtable"     "package:carData"   "package:MASS"   "package:rpart.plot"
 [9] "package:rpart"      "package:naivebayes" "package:e1071"      "tools:rstudio"
[13] "package:stats"      "package:graphics"   "package:grDevices"  "package:utils"
[17] "package:datasets"   "package:methods"    "Autoloads"          "package:base"
```

Your list of packages will be similar but not the same. Notice that the second element in the *search path* is called `Prestige` the name of the data frame we have just attached. In first place is `.GlobalEnv` which contains all the objects in your Environment Window (notice that it says `Global Environment` near the top of this window). This list of environments, packages and objects is called the *search path*.

When you type a command with `income` in it, R searches for an object with this name within `.GlobalEnv`. If none is found R then searches for an object called `income` within `Prestige`, and in this case it is found.

Example

```
> pi
[1] 3.141593
> find("pi")
[1] "package:base"
```

This tells us that the constant `pi` is stored in the very last position in the search path, in R's Base Package. We can overwrite the value of `pi`!

```
> pi<-3.0
> find("pi")
[1] ".GlobalEnv"   "package:base"
> 2*pi
[1] 6
```

So now objects called `pi` are in two places, in the Global Environment and in the Base Package. When we access `pi` the first is used, so the user defined version is takes preference.

Please now remove the fool's version of pi!

```
> rm(pi)
> pi
[1] 3.141593
> find("pi")
[1] "package:base"
```

The `attach` command provides a very convenient way to access columns of a data frame.
HOWEVER, indiscriminate use of `attach` can lead to problems. For example, suppose you have a data frame with a column called `income` *and* an object called `income` in your workspace. Every time you ask R to use `income`, it will use the first one it finds, which may not the one you want! The problem gets even worse if you've `attached` several data frames simultaneously. As a rule, only `attach` one data frame at a time and, when you've finished with it, use
`> detach(data.frame.name)`
which un-attaches that object. Detach does not delete the data frame itself so the data are still available.

If you get confused over what is and isn't attached, type
`> search()`
to list all the locations that R searches.

Exercise 4 Jointly summarising a qualitative and a quantitative variable
You have already learnt the best graphical method of comparing a quantitative variable across different levels of a qualitative variable, a box plot, in Workshop 3.

Create a box plot for `education` split by `type`.

To obtain statistics for `education` split by `type` we use the function `tapply()`

```
> table(type)
> tapply(education,type,median)
```

Observe that the values for the median in each group matches with the box plot. We read the above `tapply` command as „apply the function `median()` to the variable `education` split by `type`.

Many functions take a data variable as the first argument and then optional arguments to specify settings.

```
> mean(education)
> mean(education,trim=0.05)
```

Read the help page for mean to find out what a trimmed mean is.

To apply a function with optional arguments specify them as an argument to `tapply()`, which will pass the argument on to the function.

```
> tapply(education,type,mean)
> tapply(education,type,mean,trim=0.05)
```

Find the mean and standard deviation for `prestige` split by `type` (two commands).

The function `quantile(x,c(0.25,0.75))` returns the lower and upper quartiles for the variable `x`. Obtain the lower and upper quartiles for `education` split by `type`. Note the slightly different output format. Observe that the values for the quantile in each group matches with the box plot.

Exercise 5 Jointly summarising two quantitative variables
Plot the two variables `education` and `prestige` in a scatter plot.

Is there a linear relationship between these two variables? If yes, is it positive or negative? is it strong, weak or middling? Have a guess at estimating the correlation coefficient before you calculate it.

The covariance and (Pearson's) correlation between two variables is found using the functions `cov()` and `cor()` respectively. Obtain the covariance and correlation for `education` and `prestige`. How good was your estimate?

Plot a scatter plot of `education` and `income` and obtain the Pearson correlation for these two. Notice that there are a couple of outliers in the variable `income`.

The function `cor()` takes an argument `method="spearman"`, to calculate Spearman's rank correlation coefficient. Find this for `education` and `income`. What do you conclude?

Exercise 6 empirical cumulative distribution function

To get the R's default plot of the empirical cumulative distribution function (ecdf) use:

```
> plot(ecdf(income))
```

Note that the function `ecdf()` does the calculation and the `plot()` function displays it.

I don't particularly like the default options. To get rid of the big black dots use the argument `do.points=FALSE`.

To draw the jumps as steps use the argument verticals=TRUE.

Notice that the ecdf for `income` rises steeply between 2 thousand and 10 thousand dollars, and is then shallow for values above that. This tells us that the variable is skewed (not symmetric).

Compare this with the shape of the ecdf for `education`, which is roughly symmetric.

Exercise 7 detach

We're finished with the `Prestige` data set. you should now detach it.

Exercise 8 Missing values

In the `TitanicSurvival` data set, there is a variable called `age`. What is the problem if you try to calculate the mean age of the titanic passengers?

To see how many missing values there are use

```
> table(is.na(TitanicSurvival$age))}
```

Many of the R functions, which return a statistic, accept an argument called `na.rm`. The default value is `FALSE`, but if set to `TRUE`, the missing values are simply ignored. To obtain the mean age for those passengers with known age, use

```
> mean(???,na.rm=TRUE)
```

Find the mean age of the passengers who survived and who died, using `tapply`

```
prop.table(table(is.na(TitanicSurvival$age),TitanicSurvival$passengerClass),1)
```

What does this table tell you.

Warning! Be very careful when just ignoring missing values. If the "missingness" is dependent on another variable, you can easily bias the results of an analysis. You will learn how to deal with missing data properly in Machine Learning 2.

Exercise 9 Tidying up
At the end of the workshop: tidy up your script file, add comments to make the code readable and save your files.

Have a great weekend!

Exercise to do at home

Exercise 10 Calculating the correlation coefficient
For the following two (small!) samples X and Y, plot the variables in a scatter plot (using pencil and paper).

X	Y
10	20
60	140
70	130
20	30
30	60

Calculate the mean, variance and standard deviation of X and Y and the covariance and correlation coefficient for X and Y. Use the following table to hel with the calculations.

x	y	$(x_i - \overline{x})$	$(x_i - \overline{x})^2$	$(y_i - \overline{y})$	$(y_i - \overline{y})^2$	$(x_i - \overline{x})(y_i - \overline{y})$
10	20					
60	140					
70	130					
20	30					
30	60					
190	380	0		0		