



Fødevaredatanalyse

Debugging in R

Berit Østeby `scg776@alumni.ku.dk`

Version 1.0

Contents

Debugging	1
Errors relating to the structure of the dataset	2
> <u>NA(not available); Missing values in the dataset</u>	2
> <u>Error: Unexpected symbol in "..."</u>	
Renaming variables	5
Renaming one (or a few) variable name(s)	5
Substituting all spaces in, and adding prefixes to, all variable names	6
Data types	7
Identifying data type in your dataset	7
Treating numerical data as factors	7
> <u>Error in ...: ... : 'x' must be numeric</u>	
Changing characters (factors) to numerical values	9
Coding in R	10
> <u>Error: "File.xlsx" does not exist in current working directory ('C:/../')</u>	10
> <u>Error: could not find function "..."</u>	10
> <u>Error in ... : object 'X' not found</u>	10
<u>Error</u> when trying to run a function	11

Debugging

Errors in R can coarsely be related to either the structure of the dataset or the commands written into the R worksheet. R gives clues about whats wrong in its error message, and often, if you still don't understand what's wrong, it's easiest to google it. Here, we have provided a few of the most common errors expected in this course, and have offered a few solutions.

Errors relating to the structure of the dataset

`View` gives an intuitive idea of differing lengths or missing values.

> NA (not available); Missing values in the dataset

It happens that datasets have missing values for given variables. While these are seen in excel files as empty cells, R reads them as missing values in a matrix.

The method to exclude numbers, depends on how you are going to use the data.

	P	Q	R	S
1	<u>homa</u>	<u>mat</u>	<u>tg</u>	<u>chol</u>
2			0,91	6,1
3	1,232056	7,323452	0,67	5,7
4	1,06295		1,07	6,2
5	1,053039	8,112415	1,02	6,1
6			0,65	4,7
7	2,742857	5,235512	0,6	4,6
8	2,705691		0,6	4,1
9	1,605575	8,55614	0,57	4,4
10			0,69	4,9

R Data: Diet				
	homa	mat	tg	chol
1	NA	NA	0.91	6.1
2	1.232056	7.323452	0.67	5.7
3	1.06295	NA	1.07	6.2
4	1.053039	8.112415	1.02	6.1
5	NA	NA	0.65	4.7
6	2.742857	5.235512	0.6	4.6
7	2.705691	NA	0.6	4.1
8	1.605575	8.55614	0.57	4.4
9	NA	NA	0.69	4.9

Figure 1: How missing values are displayed in a spreadsheet vs. with `View` in R.

- Localizing NA's

By using `colSums(is.na(Dataset))`, we get any presence of NA's in the different columns.

```
library(readxl)

Diet <- read_excel("NewNordicDiet.xlsx")
colSums(is.na(Diet))
```

```
homa      mat      tg      chol      hdl
148      300       1       1       1
ldl      vldl  energyday  proEpro  fatpro
1         1      441      441      441
```

Figure 2: Overview from R of location of the NA's

- Exclude all rows with any missing values

If you only want to use complete rows, the rows including NA can be excluded from your dataset by the use of `complete.cases`.

```
library(readxl)

Diet <- read_excel("NewNordicDiet.xlsx")
nrow(Diet)

Diet_clean <- Diet[complete.cases(Diet),]
nrow(Diet_clean)
```

```
> nrow(Diet)
[1] 588
> Diet_clean <- Diet[complete.cases(Diet),]
> nrow(Diet_clean)
[1] 139
```

Figure 3: Only using complete cases dramatically reduces the amount of rows from 588 to 139.

- Calculating means/median etc. within a column

We have a column of values, but we are missing the values for some of the measurements. When trying to calculate the mean of these numbers, R replies with a missing (NA) value, telling us that our data is containing a missing value.

By using the argument `na.rm=TRUE`, we tell R to ignore any missing values in the column.

```
mean(Diet$bm)

sum(is.na(Diet$bm))
mean(Diet$bm, na.rm=TRUE)
```

```
> mean(Diet$bm)
[1] NA
>
> sum(is.na(Diet$bm))
[1] 150
> mean(Diet$bm, na.rm=TRUE)
[1] 2792.456
```

Figure 4: by using the argument `na.rm=TRUE`, R ignores any cells containing missing values.

- 'Invisible' missing values

It might be hard to see in an excel sheet whether there are any additional rows of missing values, if they are placed at the end.

	E	F	G	H	I	Fat_Protein	cage_no	bw_w_minus1	bw_w0	bw_w1
38	HF whey	8	10,7	14,7	14,6	HF whey	8	14.0	17.0	16.7
39	HF whey	8	13,2	16,4	16,1	HF whey	8	10.7	14.7	14.6
40	HF whey	8	11,3	15,3	16,2	HF whey	8	13.2	16.4	16.1
41	HF whey	8	11,4	16,3	16,5	HF whey	8	11.3	15.3	16.2
42						HF whey	8	11.4	16.3	16.5
43						NA	NA	NA	NA	NA
44						NA	NA	NA	NA	NA
45						NA	NA	NA	NA	NA
46						NA	NA	NA	NA	NA

Figure 5: The left picture shows a screenshot from the original spreadsheet, while the picture to the right is a screenshot from Rstudio.

The solution here is to extract the rows that you are using from the original dataset:

```
Mouse <- read_excel('Mouse_diet_intervention.xlsx')
View(Mouse)

# Extracting the 40 rows of the dataset containing values
MouseN <- Mouse[c(1:40),]
# Checking that it looks good
View(MouseN)
```

> Error: Unexpected symbol in "..."

Renaming variables

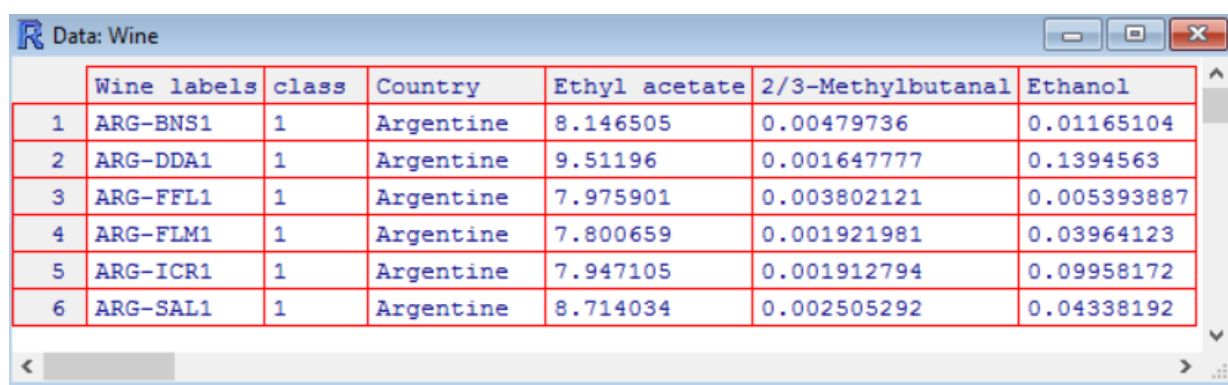
Valid variable names

- starts with a letter or a dot
- do not contain any spaces

Invalid variable names

- starts with a number
- contain spaces
- contain math operators such as /, which occasionally will be interpreted as math and not characters

Occasionally your dataset will contain invalid variable names, and R will complain.



	Wine labels	class	Country	Ethyl acetate	2/3-Methylbutanal	Ethanol
1	ARG-BNS1	1	Argentina	8.146505	0.00479736	0.01165104
2	ARG-DDA1	1	Argentina	9.51196	0.001647777	0.1394563
3	ARG-FFL1	1	Argentina	7.975901	0.003802121	0.005393887
4	ARG-FLM1	1	Argentina	7.800659	0.001921981	0.03964123
5	ARG-ICR1	1	Argentina	7.947105	0.001912794	0.09958172
6	ARG-SAL1	1	Argentina	8.714034	0.002505292	0.04338192

Figure 6: View shows a space in one of the variable names

We try to make a `qplot`, but get an error message

```
Wine <- read_excel('Wine.xlsx')
View(Wine)

p1 <- qplot(data=Wine, Furfural, Ethanol)
p2 <- qplot(data=Wine, Ethyl acetate, ethanol)
```

```
> p1 <- qplot(data=Wine, Furfural, Ethanol)
> p2 <- qplot(data=Wine, Ethyl acetate, ethanol)
Error: unexpected symbol in "p2 <- qplot(data=Wine, Ethyl acetate"
```

Figure 7: Error message: R console tells us there's something wrong in the command

Renaming one (or a few) variable name(s)

Starting on an exercise asking you to code with one of the variable with spaces in, it might seem easiest to change the single variables that you need in excel, and save it as a new. And this works perfectly fine. You can also change variable names in R as shown below

```
#Changing the name of the variable in column 4
names(Wine)[4] <- 'Ethyl.2.methyl.propanoate'
```

Substituting all spaces in, and adding prefixes to, all variable names

When you realise that Morten keeps asking for new variables to compare, several of them containing spaces ..you'll look for a way to efficiently remove or replace all the spaces in the variable names. We can use `gsub`.

```
# Substituting all spaces in variable names with a dot
names(Wine) <- gsub(" ", ".", names(Wine))
```

A prefix can also be added to all the variable names, to avoid R complaining about names starting the wrong way.

```
# Defining all variable names to start with a_
colnames(Wine) <- paste("a", colnames(Wine[,]), sep = "_")
```

	Wine labels	class	Country	Ethyl acetate	2/3-Methylbutanal	Ethanol
1	ARG-BNS1	1	Argentina	8.146505	0.00479736	0.01165104
2	ARG-DDA1	1	Argentina	9.51196	0.001647777	0.1394563
3	ARG-FFL1	1	Argentina	7.975901	0.003802121	0.005393887

	Wine.labels	class	Country	Ethyl.acetate	2/3-Methylbutanal	Ethanol
1	ARG-BNS1	1	Argentina	8.146505	0.00479736	0.01165104
2	ARG-DDA1	1	Argentina	9.51196	0.001647777	0.1394563
3	ARG-FFL1	1	Argentina	7.975901	0.003802121	0.005393887

	a_Wine.labels	a_class	a_Country	a_Ethyl.acetate	a_2/3-Methylbutanal	a_Ethanol
1	ARG-BNS1	1	Argentina	8.146505	0.00479736	0.01165104
2	ARG-DDA1	1	Argentina	9.51196	0.001647777	0.1394563
3	ARG-FFL1	1	Argentina	7.975901	0.003802121	0.005393887

Figure 8: Top figure shows the original variable names. In the middle, the spaces in variable names are replaced with dots. At the bottom, a prefix has been added to all the variable names

Data types

Identifying data type in your dataset

```
#Finding the structure of our object  
str(object)
```

Treating numerical data as factors

Decimal values are called numerics in R, and is the default computational type. However, sometimes we would prefer our numbers not to be treated as a continuous range of decimal values, but rather like factors.

```
library(readxl)  
  
CoffeeP <- read_excel("Results Panel.xlsx")  
str(CoffeeP)
```

It makes sense that the response variables can take on any decimal number in a given range for the assessors to choose from. It does not make sense to treat the different judge, who are each assigned by a number, as decimal numbers, as we only have integers for the different judges. We would prefer to treat the numbers designated for the judges, as factors. This can be done 2 ways, either it can be defined in the start, with the command `as.factor` or you can define it later in the function, by surrounding the relevant variable with `factor(variable)`.

Visual explanation of the differences - (Figure 9)

```
library(gdata) #needed by rename.vars  
library(gridExtra) #use for grid.arrange  
library(readxl)  
  
CoffeeP <- read_excel("Results Panel.xlsx")  
  
CoffeeAG <- aggregate (CoffeeP,by=list(CoffeeP$Assessor,CoffeeP$Sample),FUN ="mean")  
# renaming some variables  
CoffeeAG <- rename.vars(CoffeeAG,c("Group.1","Group.2"),c("Judge","Temp"))  
  
p1 <- qplot(data=CoffeeAG,Temp,Intensity,group=Judge,color=Judge)+geom_line()  
p2 <- qplot(data=CoffeeAG,Temp,Intensity,group=Judge,color=factor(Judge))+geom_line()  
grid.arrange(p1, p2, ncol=2)
```

Telling R to treat numerical string as factor from the start - (Figure 10)

```
library(gdata) #needed by rename.vars  
library(gridExtra) #use for grid.arrange  
library(readxl)  
  
CoffeeP <- read_excel("Results Panel.xlsx")  
# Telling R to treat Assessor as factors  
CoffeeP$Assessor <- as.factor(CoffeeP$Assessor)  
  
CoffeeAG <- aggregate (CoffeeP,by=list(CoffeeP$Assessor,CoffeeP$Sample),FUN ="mean")  
# renaming some variables  
CoffeeAG <- rename.vars(CoffeeAG,c("Group.1","Group.2"),c("Judge","Temp"))  
head(CoffeeAG)  
  
p1 <- qplot(data=CoffeeAG,Temp,Intensity,group=Judge,color=Judge)+geom_line()  
p2 <- qplot(data=CoffeeAG,Temp,Intensity,group=Judge,color=factor(Judge))+geom_line()  
grid.arrange(p1, p2, ncol=2)
```

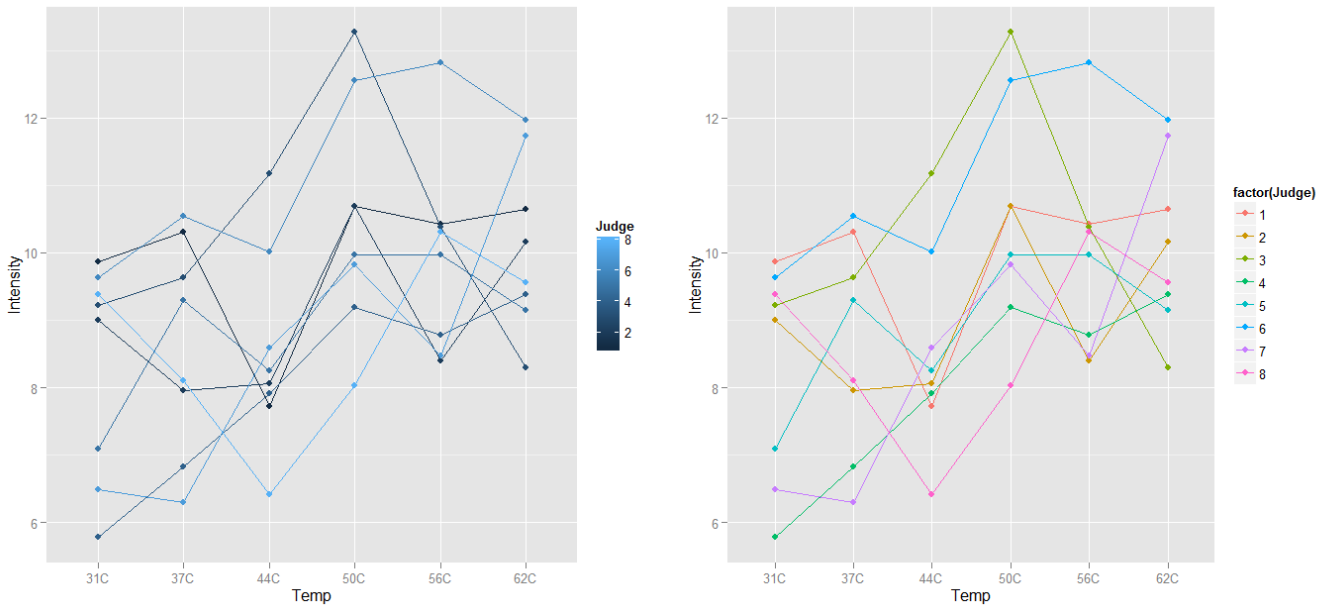



Figure 9: In the figure to the left, the numbers representing the judges are treated as a continuous scale, whereas on the figure to the left, the numbers are treated as factors.

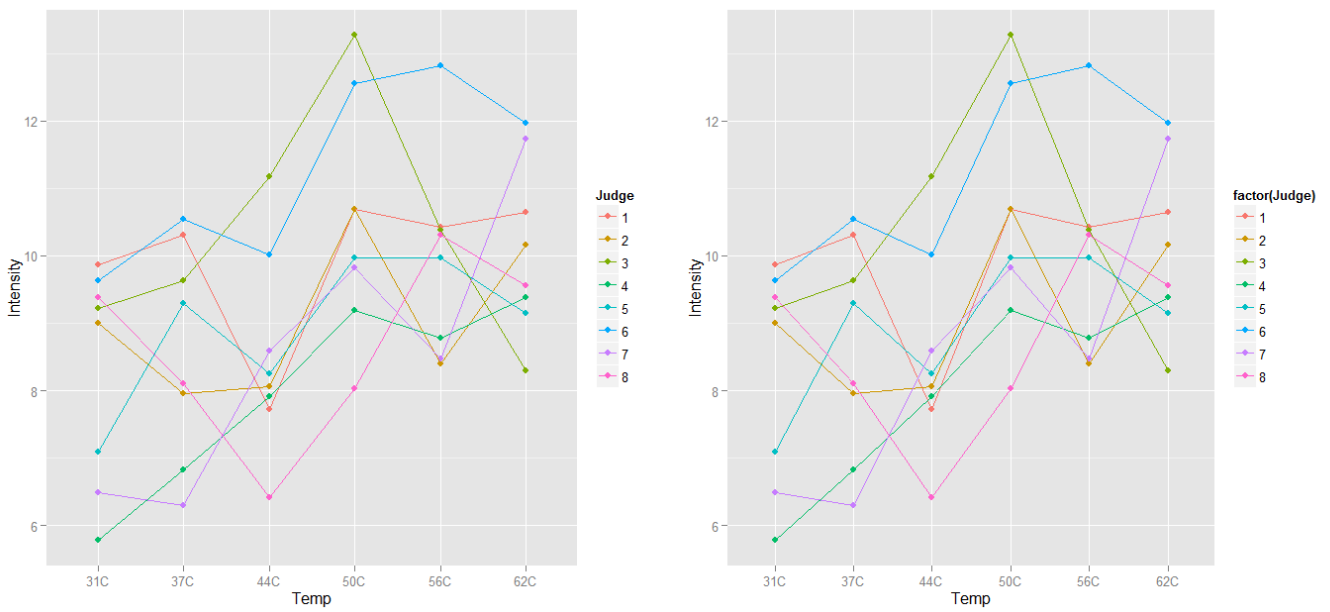


Figure 10: As we defined assessors as factors in the start, R will treat them as factors all along, and therefore we see no difference in these plots.

> Error in ...: ... : 'x' must be numeric

Changing characters (factors) to numerical values

Values that we want to treat as numerical, may be styled as characters in a dataset. If you get the error message up, that x should be numeric, you can change factors into numerical values similar to what we did above.

OBS! Whenever you change from character or factor, check that R has done it correctly to avoid losing information from your dataset. Do this by saving the numeric version onto a new string, and compare this with the original one.

```
CoffeeP$Assessor.num <- as.numeric(CoffeeP$Assessor)
```

```
CoffeeP$Assessor
```

```
CoffeeP$Assessor.num
```

Might come in handy in week 4, task 3

Coding in R

```
> Error: "File.xlsx" does not exist in current working directory ('C:/../')
```

Check that you have told R to work from the right folder. Go to **Session** → **Set working directory** → **Choose directory...**, and find the folder where you have the relevant datasets and saved R-files.

```
> Error: could not find function "..."
```

Check:

- is the function written correctly ?
- has the library necessary for the function been called for?

```
> CoffeeP <- read_excl("Results Panel.xlsx")
Error: could not find function "read_excl"
> CoffeeP <- read_Excel("Results Panel.xlsx")
Error: could not find function "read_Excel"
> CoffeeP <- read_excel("Results Panel.xlsx")
> |
```

Figure 11: R won't run on typos

```
> Error in ... : object 'X' not found
```

Check for typos in your function.

A good way to ensure that the object exists is to "tab" out the name.

If you want to call the variable `CoffeeP@Assessor`, write `Coffee` and tab to get the rest of the object. Continue with `CoffeeP@As` and hit tab to get the rest of the variable.

```
> CoffeeP <- read_excel("Results Panel.xlsx")
> CoffeeAG <- aggregate (CoffeP,by=list(CoffeeP$Assessor,CoffeeP$Sample),FUN ="sd")
Error in aggregate(CoffeP, by = list(CoffeeP$Assessor, CoffeeP$Sample), :
  object 'CoffeP' not found
> |
```

Figure 12: R won't run on typos

Error when trying to run a function

For a given R-function to run, you need to feed it with some required information. If something is missing, for R to be able to run the command, you can use `help` for the relevant function. This will give you the information you need about the setup of this function. We try with the function `aggregate`:

```
help(aggregate)
```

What happens is that this explanation pops up in the bottom right corner (given that you've spelled the function correctly):

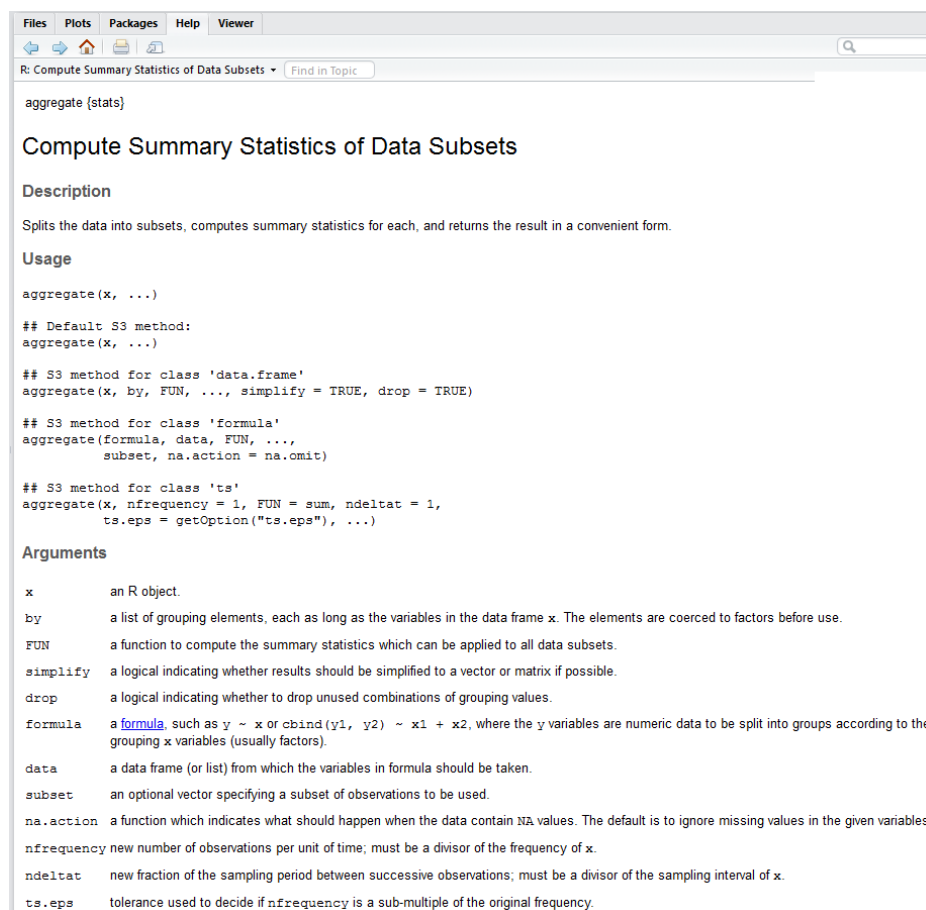


Figure 13: Rstudio can give detailed information about the functions