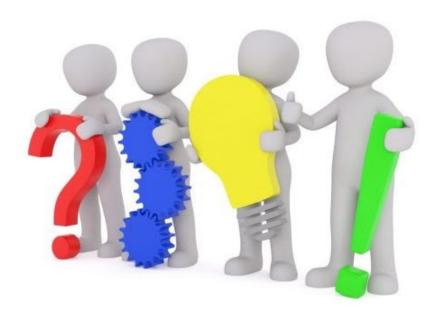
...Data Scientists know that about 80% of a Data Science project consists of preparing the data so that they can be analyzed. Building Machine Learning models is the fun part that only comes afterwards!

This process is called *Data Wrangling* (or *Data Munging*). If you want to use some Base R data wrangling techniques in a fun game to hack a password read on!

First install proton (on CRAN), load it and type proton () to start it:

```
library(proton)
##
##
## Your goal is to find Slawomir Pietraszko's credentials for the Proton server.
## This is the only way for Bit to find the secret plans of Pietraszko's
laboratory.
##
## Enter the `proton()` command in order to start the adventure.
## Remember that at any time you may add `hint=TRUE` argument to the executed
command in order to get additional suggestions.
proton()
## Pietraszko uses a password which is very difficult to guess.
## At first, try to hack an account of a person which is not as cautious as
Pietraszko.
##
## But who is the weakest point? Initial investigation suggests that John
Insecure doesn't care about security and has an account on the Proton server. He
may use a password which is easy to crack.
## Let's attack his account first!
##
## Problem 1: Find the login of John Insecure.
## Bit has scrapped 'employees' data (names and logins) from the www web page of
Technical University of Warsaw. The data is in the data.frame `employees`.
## Now, your task is to find John Insecure's login.
## When you finally find out what John's login is, use `proton(action = "login",
login="XYZ")` command, where XYZ is Insecure's login.
```

Now, try to solve the problem yourself before reading on...



The best way is always to get some overview over the data. That can be achieved by the str() function (for structure):

```
str(employees)
## 'data.frame': 541 obs. of 3 variables:
## $ name : Factor w/ 534 levels "Aaron", "Adam", ..: 272 198 240 442 34 389
433 460 351 16 ...
## $ surname: Factor w/ 541 levels "Abbott", "Adams", ..: 369 274 310 247 251 78
90 462 130 291 ...
## $ login : chr "j.patrick" "gerald.long" "j.mendoza" "rjoh" ...
```

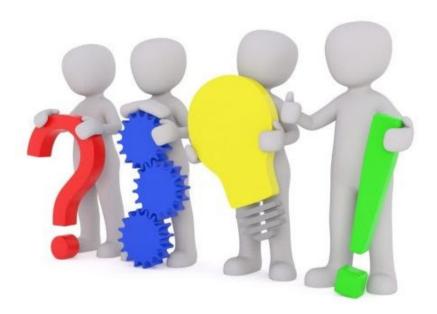
We know that John's surname is "Insecure", so we use that for *subsetting* the data frame:

```
employees[employees$surname == "Insecure", ]
## name surname login
## 217 John Insecure johnins
```

Ok, we can now use this information to get to the next problem:

```
proton(action = "login", login = "johnins", hint = TRUE)
## Congratulations! You have found out what John Insecure's login is!
## It is highly likely that he uses some typical password.
## Bit downloaded from the Internet a database with 1000 most commonly used
passwords.
## You can find this database in the `top1000passwords` vector.
##
## Problem 2: Find John Insecure's password.
##
## Use `proton(action = "login", login="XYZ", password="ABC")` command in order
to log into the Proton server with the given credentials.
## If the password is correct, you will get the following message:
## `Success! User is logged in!`.
## Otherwise you will get:
## `Password or login is incorrect!`.
##
## HINT:
## Use the brute force method.
## By using a loop, try to log in with subsequent passwords from
`top1000passwords` vector as long as you receive:
## `Success! User is logged in!`.
```

Now, try to solve the problem yourself before reading on...



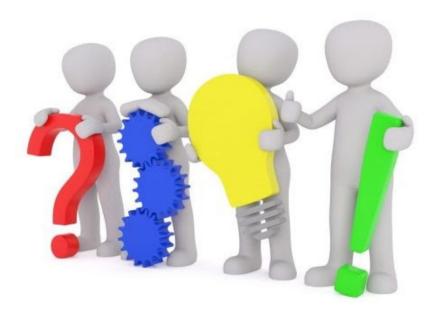
Again, let us gain an overview:

```
str(top1000passwords)
## chr [1:1000] "123456" "password" "12345678" "qwerty" "123456789" ...
```

Ok, *Brute Force* means that we simply try all possibilities to find the right one (a technique that is also used by real hackers... which is the reason why most modern systems only accept a few (e.g. 3) trials before hitting a time limit or deactivating an account altogether). Here, we can use a *for loop* for that:

```
for (pw in top1000passwords) {
 proton(action = "login", login = "johnins", password = pw, hint = TRUE)
}
## Well done! This is the right password!
## Bit used John Insecure's account in order to log into the Proton server.
## It turns out that John has access to server logs.
## Now, Bit wants to check from which workstation Pietraszko is frequently
logging into the Proton server. Bit hopes that there will be some useful data.
##
## Logs are in the `logs` dataset.
## Consecutive columns contain information such as: who, when and from which
computer logged into Proton.
## Problem 3: Check from which server Pietraszko logs into the Proton server
most often.
##
## Use `proton(action = "server", host="XYZ")` command in order to learn more
about what can be found on the XYZ server.
## The biggest chance to find something interesting is to find a server from
which Pietraszko logs in the most often.
##
##
## HINT:
## In order to get to know from which server Pietraszko is logging the most
often one may:
## 1. Use `filter` function to choose only Pietraszko's logs,
## 2. Use `group_by` and `summarise` to count the number of Pietraszko's logs
into separate servers,
## 3. Use `arrange` function to sort servers' list by the frequency of logs.
##
## Use `employees` database in order to check what Pietraszko's login is.
```

There are always different possibilities to achieve a goal and you can, of course, use any of the functions given under "HINT"... yet one of my favorite functions is the table() function and I encourage you to try it out here...



Ok, the following first steps shouldn't come as a surprise by now:

```
str(logs)
## 'data.frame': 59366 obs. of 3 variables:
## $ login: Factor w/ 541 levels "j.patrick", "gerald.long",..: 172 45 42 196
254 390 169 397 469 361 ...
## $ host : Factor w/ 312 levels "193.0.96.13.0",..: 35 124 250 146 157 227 230
69 239 134 ...
## $ data : POSIXct, format: "2014-09-01 03:01:12" "2014-09-01 03:01:51" ...
employees[employees$surname == "Pietraszko", ]
## name surname login
## 477 Slawomir Pietraszko slap
```

As said above we will use the table() function to get the frequencies of server logins:

```
log <- logs[logs$login == "slap", ]
table(as.character(log$host))
##
## 193.0.96.13.20 193.0.96.13.38 194.29.178.108 194.29.178.155 194.29.178.16
## 33 1 74 6 112</pre>
```

We now use the most often used to get to the next and last problem:

```
proton(action = "server", host = "194.29.178.16", hint = TRUE)
## It turns out that Pietraszko often uses the public workstation 194.29.178.16.
## What a carelessness.
##
## Bit infiltrated this workstation easily. He downloaded `bash_history` file which contains a list of all commands that were entered into the server's console.
## The chances are that some time ago Pietraszko typed a password into the console by mistake thinking that he was logging into the Proton server.
##
## Problem 4: Find the Pietraszko's password.
##
## In the `bash_history` dataset you will find all commands and parameters which have ever been entered.
```

Try to extract from this dataset only commands (only strings before space)
and check whether one of them looks like a password.
##
##

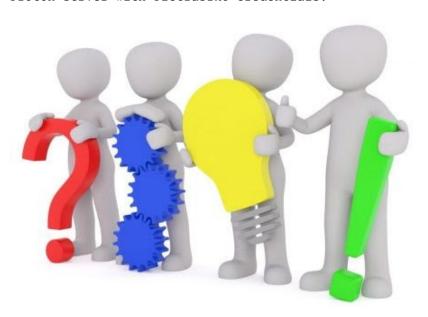
HINT:

Commands and parameters are separated by a space. In order to extract only names of commands from each line, you can use `gsub` or `strsplit` function.

After having all commands extracted you should check how often each command is used.

Perhaps it will turn out that one of typed in commands look like a password?

If you see something which looks like a password, you shall use
`proton(action = "login", login="XYZ", password="ABC")` command to log into the
Proton server with Pietraszko credentials.



After checking the structure again we could use strsplit() (for stringsplit) in the following way

```
str(bash_history)
## chr [1:19913] "mcedit /var/log/lighttpd/*" "pwd" ...
```

table(unlist(strsplit(bash history, " ")))

		<u> </u>
##		
##	/bin	/boot
##	338	338
##	/cdrom	/dev
##	338	338
##	/etc	/home
##	338	338
##	/lib	/lost+found
##	338	338
##	/media	/mnt
##	338	338
##	/opt	/proc
##	338	338
##	/root	/run
##	338	338
##	/sbin	/selinux
##	338	338
##	/srv	/sys
##	338	338
##	/tmp	/usr

	220	220
##	338	338
##	/var	/var/log/apache2/*
##	338	219
##	/var/log/apport.log	/var/log/auth.log
##	219	219
##	/var/log/boot	/var/log/daemon.log
##	219	219
##	/var/log/debug	/var/log/dmesg
##	219	219
##	/var/log/dpkg.log	/var/log/faillog
##	219	219
##	/var/log/fsck/*	/var/log/kern.log
##	219	219
##	/var/log/lighttpd/*	/var/log/lpr.log
##	219	219
##	/var/log/mail.*	/var/log/messages
##	219	219
##	/var/log/mysql.*	/var/log/user.log
##	219	219
##	/var/log/xorg.0.log	~/.bash_history
##	219	133
##	~/.bash_login	~/.bash_logout
##	133	133
##	~/.bash profile	~/.bashrc
##	133	133
##	~/.emacs	~/.exrc
##	133	133
##	~/.forward	~/.fvwm2rc
##	133	133
##	~/.fvwmrc	~/.gtkrc
##	133	133
##	~/.hushlogin	~/.kderc
##	133	133
##	~/.mail.rc	~/.muttrc
##	133	133
##	~/.ncftp/	~/.netrc
##	133	133
##	~/.pinerc	~/.profile
##	133	133
##	~/.rhosts	~/.rpmrc
##	133	133
##	~/.signature	~/.twmrc
##	133	133
##	~/.vimrc	~/.Xauthority
##	133	133
##	~/.Xdefaults-hostname	~/.Xdefaults,
##	133	133
##	~/.xinitrc	~/.Xmodmap
##	133	133
##	~/.xmodmaprc	~/.Xresources
##	133	133
##	~/.xserverrc	~/mbox
##	133	133
##	~/News/Sent-Message-IDs	
	=	alert_actions.conf
##	133	74
##	app.conf	audit.conf
##	74	74

шш	outhortication conf	outhoning conf
##	authentication.conf 74	authorize.conf 74
##	aux	ax
##	100	100
##	cat	cd
##	4341	2520
##	collections.conf	commands.conf
##	74	74
##	ср	crawl.conf
##	1176	74
##	datamodels.conf	default.meta.conf
##	74	74
##	deploymentclient.conf	DHbb7QXppuHnaXGN
##	74	1
##	distsearch.conf	event renderers.conf
##	74	74
##	eventdiscoverer.conf	
##	74	eventtypes.conf 74
	· -	
##	fields.conf	httpd
##	74	20
##	indexes.conf	inputs.conf
##	74	74
##	instance.cfg.conf	limits.conf
##	74	74
##	literals.conf	ls
##	74	1806
##	macros.conf	mc
##	74	112
##	mcedit	multikv.conf
##	1944	74
##	outputs.conf	pdf server.conf
##	74	- <u>-</u> 74
##	procmon-filters.conf	props.conf
##	74	74
##	ps	pubsub.conf
##	420	74
##	pwd	restmap.conf
##	80	74
##		savedsearches.conf
	rm	savedsearches.com
##	1596	
##	searchbnf.conf	segmenters.conf
##	74	74
##	server.conf	serverclass.conf
##	74	74
##	serverclass.seed.xml.conf	service
##	74	20
##	source-classifier.conf	sourcetypes.conf
##	74	74
##	start	tags.conf
##	20	74
##	tenants.conf	times.conf
##	74	74
##	top	transactiontypes.conf
##	150	74
##	transforms.conf	user-seed.conf
##	74	74
##	vi	viewstates.conf
пπ	VΙ	vicwscaces.com

```
##
                                                         74
                          2666
##
                           vim
                                                  web.conf
##
                          2991
                                                        74
##
                                                  wmi.conf
                        whoiam
                                                         74
##
                            90
##
       workflow_actions.conf
##
```

When you look through this list you will find something that looks suspiciously like a password: "DHbb7QXppuHnaXGN". Let's try this to finally hack into the system:

```
proton(action = "login", login = "slap", password = "DHbb7QXppuHnaXGN")
## Congratulations!
##
## You have cracked Pietraszko's password!
## Secret plans of his lab are now in your hands.
## What is in this mysterious lab?
## You may read about it in the `Pietraszko's cave` story which is available at http://biecek.pl/BetaBit/Warsaw
##
## Next adventure of Beta and Bit will be available soon.
## proton.login.pass
## "Success! User is logged in!"
```



This was fun, wasn't it! And hopefully, you learned some helpful data wrangling techniques along the way...