shinemas2R

An R package to visualize outputs from the data base Seed History and Network Management System (SHiNeMaS)

version 0.9.1

January 20, 2016

Pierre Rivière[1,2] Yannick de Oliveira[3]

[1] Réseau Semences Paysannes, 3 avenue de la gare, F-47190 Aiguillon, France
[2] INRA, UMR 0320, Génétique Quantitative et Evolution, DEAP team, Ferme du Moulon F-91190 Gif sur Yvette, France
[3] INRA, UMR 0320, Génétique Quantitative et Evolution, ABI team, Ferme du Moulon F-91190 Gif sur Yvette, France
Contact: pierre@semencespaysannes.org
Contributions: P. Rivière worte the R functions and the vignette, Y. de Oliveira wrote the SQL queries.

Le Réseau Semences Paysannes (the French Farmers' Seeds Network (RSP)), created in 2003, brings together a great diversity of collectives and people who preserve farmers' seeds in fields, orchards, vineyards and gardens. They are involved in supporting the consolidation of local initiatives to maintain and renew cultivated biodiversity through Community Seeds Systems. Over 80 organizations have come together to promote and develop farmers' seeds, and to protect farmers' rights over their seeds.
www.semencespaysannes.org (in french).

The Diversity, Evolution and Adaptation of Populations (DEAP) team led by Isabelle Goldringer is part of INRA UMR 0320 Quantitative Genetic and Evolution. Its work is based on the analysis of the genetic and evolutionary mechanisms underlying evolution and adaptation of crop populations. DEAP develops strategies for on farm management of crop genetic diversity and for plant breeding (evolutionary and/or participatory) adated to organic and low input agriculture. Assessing the benefits of in-field genetic diversity (variety mixtures, populations) and designing / breeding optimized mixtures adapted to local conditions are also key research objectives.
http://moulon.inra.fr/index.php/en/team/deap
The bioinformatics and informatics facility (ABI, Atelier Bioinformatique et Informatique) provides bioinformatics expertise and IT support. The staff includes 6 experts in system administration, software development or bio-analysis, and develops databases and softwares for proteomics, genetics and genomics. ABI offers hardware resources, scientific programming and consulting for DNA, RNA and protein sequence analysis up to genome-wide scale. ABI works in tight collaboration with the Bioinformatics facilities of University Paris-Sud and INRA, and contributes to the future French Bioinformatics Institute.
http://moulon.inra.fr/index.php/en/tranverse-team

Contents

Wheat trials on farm within our participatory plant breeding programme, summer 2012, Auvergne, France. CC-BY-NC-SA. Pierre Rivière.

1 Philosophy of `shinemas2R`

1.1 What is SHiNeMaS?

Started in 2009, DEAP and ABI teams from INRA Le Moulon developped a data base to store data in a reliable way during research programs dedicated to the study of dynamic management of crop diversity within experimental stations and farmer networks (Thomas, 2011).

From 2011 to 2013, this data base was migrated to Seed History and Network Management System (SHiNeMaS) and has evolved in a participatory plant breeding program involving INRA Le Moulon and the Réseau Semences Paysannes (RSP) on bread wheat (Rivière, 2014).

From 2014 to today, farmers organisations' facilitators informed on their specific needs for their data management, involving other species than bread wheat such as maize, tomatos or trees. Specific developments were performed to fit the farmers organisations' needs.

SHiNeMaS stores information related to (Figure 1):

- network relation betweens seed-lots
- data linked to the seed-lots and to the relations between seed-lots

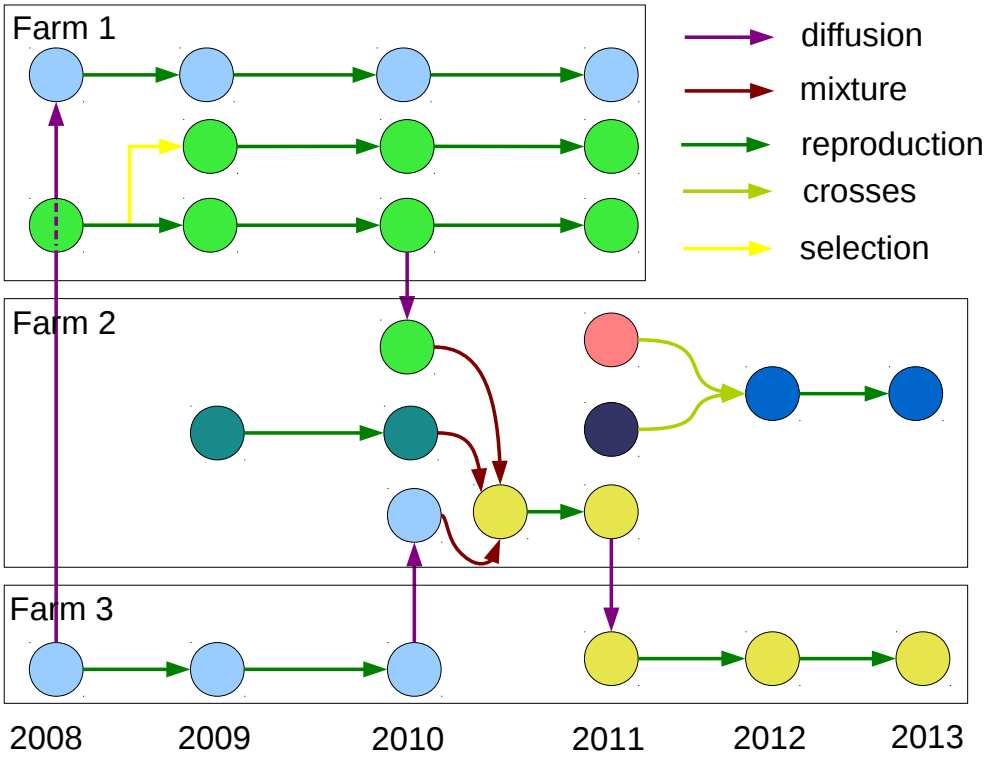


Figure 1: Seed-lots relation in SHiNeMaS. The seed-lots are represented by a circle. The color represents a germplasm (i.e. a variety). The arrows represent the relations between seed-lots (diffusion, mixture, reproduction, crosses and selection). Data are linked to the seed-lots and to the relations between seed-lots.

SHiNeMaS can be download here. More information on the history of SHiNeMaS and its uses with tutorials can be found on the website (De Oliveira et al., 2015).

`shinemas2R` is an `R` package that analyses outputs from SHiNeMaS. It does descriptive analysis, formats data for existing `R` packages to perform statistical analysis as well as compiles results in pdf.

1.2 Function relations in `shinemas2R`

`shinemas2R` is divided into three steps (Figure 2):

1. Get the data from SHiNeMaS with `get.data`. You may encrypt the data with `encrypt.data` or translate it with `translate.data`. There are two types of data:
   - network relation betweens seed-lots (section 3)
   - data linked to the seed-lots and to the relations between seed-lots (section 4)
2. From the data,
   (a) Get descriptive outputs from the data: plots with `get.ggplot` or tables with `get.table`
   (b) Get statistical outputs by formating the data with `format.data` in order to use an existing `R` package (The following packages can be used: `PPBstats`).
3. Get a pdf that compiles the outputs in a pdf document with `get.pdf` (section 5)

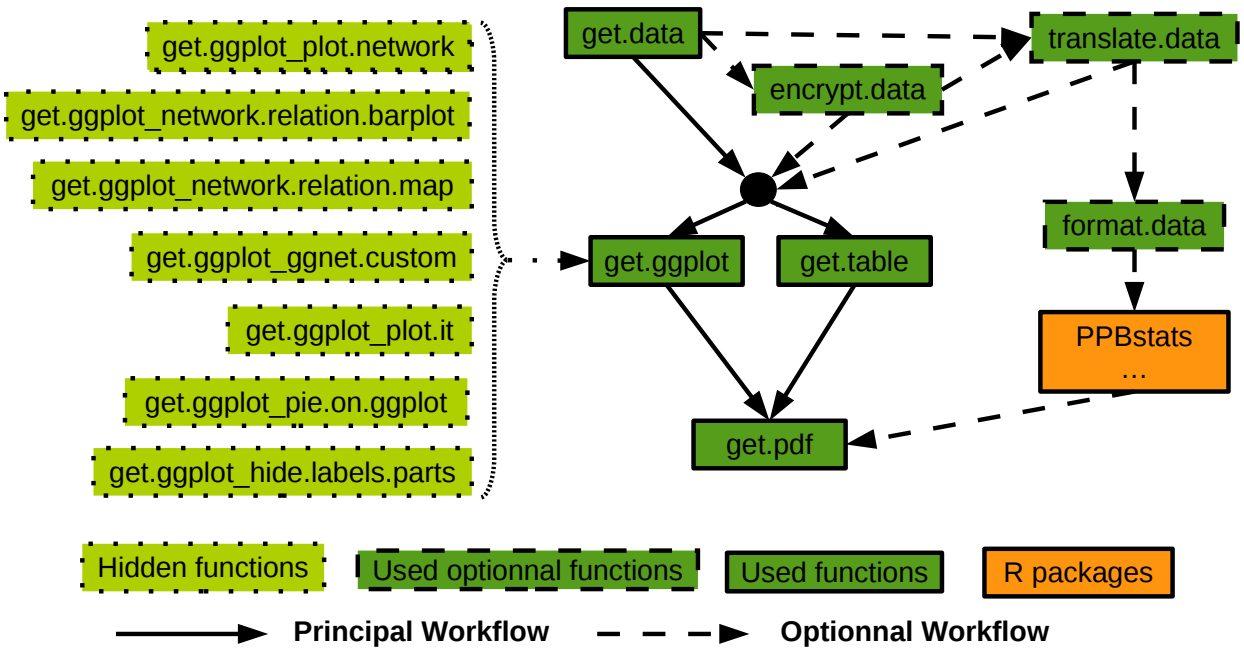


Figure 2: Relations between functions in `shinemas2R`

`get.ggplot` is based on `ggplot2` package. It is therefore easy to custom a plot generated by `get.ggplot` by adding layers. More details can be found on the `ggplot2` documentation website: http://ggplot2.org.

1.3 Let's go!

To continue, load the package:

```
library(shinemas2R)
```

For an easier visualisation, all warning messages are not representing (NAs introduced, rows deleted, etc.)

1.3.1 Test with SHiNeMaS installed

It may be useful to test the `get.data` function on SHiNeMaS directly. Two options:

- SHiNeMaS is installed on a sever where you can have access. You need to ask the admin the following information : `db_user`, `db_host`, `db_name`, `db_password`. Add these information in `info_db`, which is needed in the next steps in `get.data`. For example:

```
info_db = list(
        db_user = "toto",
        db_host = "127.175.166.175",
        db_name = "my_shinemas",
        db_password = "secret"
        )
```

- install a demo of SHiNeMaS. The procedure to do so is explained in appendix A.

As you will use `get.data`, you should set:

```
use.get.data = TRUE
```

1.3.2 Test without installing SHiNeMaS

In this vignette, you can test `shinemas2R` without installing SHiNeMaS. You do not use the `get.data` function:

```
use.get.data = FALSE
```

And you can load the data-sets by downloading it:

```
system("mkdir data")
system("cp -r /home/pierre/Documents/geek/R-stats/Rdev/R_package_shinemas2R/shinemas2R_data/*RData ./
```

and then `load()` it.

2 Raw information on levels and variables

Severals information are stored in SHiNeMaS. To get information on raw information and levels and variable, you must use the argument `query.type` in `get.data`. The following table summarize the different possibilities:

`query.type`	type of information present in SHiNeMaS
"variable"	variable
"person"	person
"year"	year
"project"	projet
"seed.lot"	seed-lots
"selection.person"	persons that did intra-varietal mass selection
"reproduction.type"	reproduction type
"germplasm.type"	germplasm type
"germplasm"	germplasm

For example,

```
if(use.get.data){
        person = get.data(
                db_user = info_db$db_user, db_host = info_db$db_host,# db infos
                db_name = info_db$db_name, db_password = info_db$db_password, # db infos
                query.type = "person" # information on person present in SHiNeMaS
        )
} else {
                load("data/person.RData")
        }

person$data
```

```
##   [1] ""      "ADP"   "AGO"   "ALB"   "ALH"   "ALP"   "ALPO"
##   [8] "ALR"   "ANB"   "ANW"   "AOV"   "ARB"   "ARC"   "AUD"
##  [15] "BAM"   "BED"   "BEN"   "BER"   "BRE"   "CAB"   "CAL"
##  [22] "CEP"   "CER"   "CHD"   "CHH"   "CHP"   "CIG"   "CLB"
##  [29] "CLF"   "CLM"   "CRP"   "DAC"   "DAL"   "DAT"   "DAV"
##  [36] "DET"   "DIF"   "DIT"   "DOB"   "EDD"   "ESS"   "EUK"
##  [43] "FLM"   "FLP"   "FRC"   "FRG"   "FRP"   "FUE"   "GIM"
##  [50] "GUF"   "GUK"   "HEA"   "HEC"   "HEF"   "HEG"   "HEL"
##  [57] "HGA"   "HZE"   "I01"   "ICE"   "INE"   "ISG"   "JAB"
##  [64] "JAR"   "JCM"   "JEF"   "JFB"   "JFM"   "JJG"   "JMC"
##  [71] "JMM"   "JMR"   "JOP"   "JPB"   "JSG"   "JUB"   "JUD"
##  [78] "JUM"   "LAG"   "LAH"   "LAU"   "LOD"   "LUD"   "MAF"
##  [85] "MAI   " "MAS"   "MAT"   "MAV"   "MIR"   "MLN"   "MPH"
##  [92] "NIS"   "OLM"   "OLR"   "P01"   "P02"   "PAD"   "PAF"
##  [99] "PAJ"   "PHC"   "PHG"   "PHL"   "PID"   "PIR"   "PIS"
## [106] "RAB"   "RAL"   "RDR"   "RDZ"   "RIH"   "ROG"   "ROW"
## [113] "SAC"   "STE"   "STP"   "TEH"   "THG"   "TOA"   "VEN"
## [120] "VER"   "VIC"   "VIH"   "VJT"   "YVC"   "YVV"
## attr(,"shinemas2R.object")
## [1] "person"
```

3 Network relations between seed-lots

3.1 Get the data set

To get the data, use the function `get.data` with `query.type = "network"`.

You can set filters to the query with the following argument :

- `filter.in` to choose nothing expect `filter.in`
- `filter.out` to choose everything expect `filter.out`

values of `filter` can be `germplasm`, `germplasm.type`, `year`, `person`, `project`, `seed-lots`, `relation` or `reproduction type`.

It is important to choose if you apply the filter on the `father` or the `son` of a relation. This can be done with the argument `filter.on`. Possibles values are `"father"`, `"son"` or `"father-son"`. It is set by default to `"father-son"`.

It is possible to get the `Mdist` square matrix with the number of reproductions that separate two seed-lots since their last common diffusion (argument `Mdist = TRUE`). This square matrix can be compared to a differenciation distance. It can be put in relation with genetic F_{st} for example (Nei, 1973)[1].

```
if(use.get.data){
        data_network = get.data(
                db_user = info_db$db_user, db_host = info_db$db_host, # db infos
                db_name = info_db$db_name, db_password = info_db$db_password, # db infos
                query.type = "network", # network query
                germplasm.in = "Rouge-du-Roc", # germplasm to keep
                filter.on = "father-son", # filter on father AND son
                Mdist = TRUE
                )
} else {
        # 1. Query SHiNeMaS ...
        # 2. Create network matrix ...
        # 3. Link information to vertex and edges ...
        # 4. Get network information on seed-lots ...
        # 5. Get Mdist square matrix ...

        #data_network = encrypt.data(data_network)
        #The key has been written in /home/pierre/key_network_Tue Nov 24 11:52:47 2015.RData
        load("./data/data_network.RData")
        }
```

The function returns a list with:

- the netwok object

```
n = data_network$data$network
n

##  Network attributes:
##   vertices = 661
##   directed = TRUE
##   hyper = FALSE
##   loops = FALSE
##   multiple = FALSE
##   bipartite = FALSE
##   total edges= 783
```

[1] The `R` package `adegenet` can be used for that.

```
##     missing edges= 0
##     non-missing edges= 783
##
##  Vertex attribute names:
##     germplasm germplasm_type germplasm.type person sex vertex.names year
##
##  Edge attribute names:
##     generation relation
```

Note you can convert this object to an `igraph` object. This may be useful if you like to use the `igraph` package. See http://mbojan.github.io/intergraph/ for more information.

```
n_igraph = intergraph::asIgraph(n)
n_igraph
```

```
## IGRAPH D--- 661 783 --
## + attr: germplasm (v/c), germplasm_type (v/c),
## | germplasm.type (v/c), na (v/l), person (v/c), sex (v/c),
## | vertex.names (v/c), year (v/n), generation (e/c), na
## | (e/l), relation (e/c)
## + edges:
##  [1]  2-> 3  3-> 4  3-> 5  4-> 6  5-> 7  6-> 8  8-> 9  7->10  9->11
## [10] 10->12 11->13 12->14 15->16 11->17 11->18 11->19 11->20 11->21
## [19] 11->22 11->23 11->24 11->25 11->26 11->27 11->28 11->29 13->30
## [28] 14->31 11->32 11->33 11->34 11->35 11->36 12->37 11->38 11->39
## [37] 14->40 13->41 13->42 13->43 21->44 21->45 34->46 34->47 34->48
## + ... omitted several edges
```

- the network.info matrix with information on relation in the network.

```
head(data_network$data$network.info)
```

```
##                                 sl alt  long   lat diffusion
## 1 germplasm-856_person-67_2002_0001 107  0.42 44.40      <NA>
## 2 germplasm-882_person-67_2002_0001 107  0.42 44.40      <NA>
## 3 germplasm-882_person-43_2002_0001  24 -0.60 47.71   receive
## 4 germplasm-882_person-67_2003_0001 107  0.42 44.40      <NA>
## 5 germplasm-882_person-43_2003_0001  24 -0.60 47.71      <NA>
## 6 germplasm-882_person-67_2004_0001 107  0.42 44.40      <NA>
##   id.diff reproduction mixture selection cross.info     germplasm
## 1    <NA>      harvest    <NA>      <NA>       <NA> germplasm-856
## 2    <NA>  harvest-sow    <NA>      <NA>       <NA> germplasm-882
## 3    3229         <NA>    <NA>      <NA>       <NA> germplasm-882
## 4    <NA>  harvest-sow    <NA>      <NA>       <NA> germplasm-882
## 5    <NA>  harvest-sow    <NA>      <NA>       <NA> germplasm-882
## 6    <NA>  harvest-sow    <NA>      <NA>       <NA> germplasm-882
##      person year
## 1 person-67 2002
## 2 person-67 2002
## 3 person-43 2002
## 4 person-67 2003
## 5 person-43 2003
## 6 person-67 2004
```

- the `Mdist` square matrix

```
dim(data_network$data$Mdist)

## [1] 627 627
```

You may want to fill gaps into you network data set regarding diffusion events (if the information is stored in SHiNeMaS!). This may be usefull, regarding a network on one person, to know where do the seed-lots come from. To do so, use `fill.diffusion.gap = TRUE`.

```
if(use.get.data){
        data_network_JAB = get.data(
                db_user = info_db$db_user, db_host = info_db$db_host, # db infos
                db_name = info_db$db_name, db_password = info_db$db_password,  # db infos
                query.type = "network", # network query
                person.in = "JAB", # person to keep
                filter.on = "father-son" # filter on father AND son
                )
} else {
        # 1. Query SHiNeMaS ...
        # 2. Create network matrix ...
        # 3. Link information to vertex and edges ...
        # 4. Get network information on seed-lots ...

        #data_network_JAB = encrypt.data(data_network_JAB)
        # The key has been written in /home/pierre/key_network_Tue Nov 24 11:54:53 2015.RData
        load("./data/data_network_JAB.RData")
        }
```

```
if(use.get.data){
        data_network_JAB_fill_gap = get.data(
                db_user = info_db$db_user, db_host = info_db$db_host, # db infos
                db_name = info_db$db_name, db_password = info_db$db_password, # db infos
                query.type = "network", # network query
                person.in = "JAB", # person to keep
                filter.on = "father-son", # filter on father AND son
                fill.diffusion.gap = TRUE
                )
} else {
        # 1. Query SHiNeMaS ...
        # 2. Create network matrix ...
        # 2.1. Fill diffusion gaps ...
        # 3. Link information to vertex and edges ...
        # 4. Get network information on seed-lots ...

        #data_network_JAB_fill_gap = encrypt.data(data_network_JAB_fill_gap)
        # The key has been written in /home/pierre/key_network_Tue Nov 24 11:55:53 2015.RData
        load("./data/data_network_JAB_fill_gap.RData")
        }
```

3.2 Get the ggplots

Once, you have got the data, you can do plots with the function `get.ggplot`.

```
default_network_ggplot = get.ggplot(data_network)

## As ggplot.type is NULL, ggplot.type is set to network-network, network-reproduction-sown,
network-reproduction-harvested, network-reproduction-positive-inter-selected, network-reproduction-po
network-reproduction-negative-inter-selected, network-reproduction-crossed, network-diffusion-sent,
network-diffusion-received, network-diffusion-relation, network-mixture, network-positive-intra-selec
## As ggplot.display is NULL, ggplot.display is set to c("barplot", "map").  Note that for
ggplot.type == "network-diffusion-relation", ggplot.display is set to "map"
## As x.axis and in.col are NULL, all the combinaisons of x.axis and in.col are done for barplot.

## [1] "A METTRE A JOUR QUAND ON AURA EVENT YEAR"
## [1] "A METTRE A JOUR QUAND ON AURA EVENT YEAR"
```

By default, all the possibles plots or maps are done.

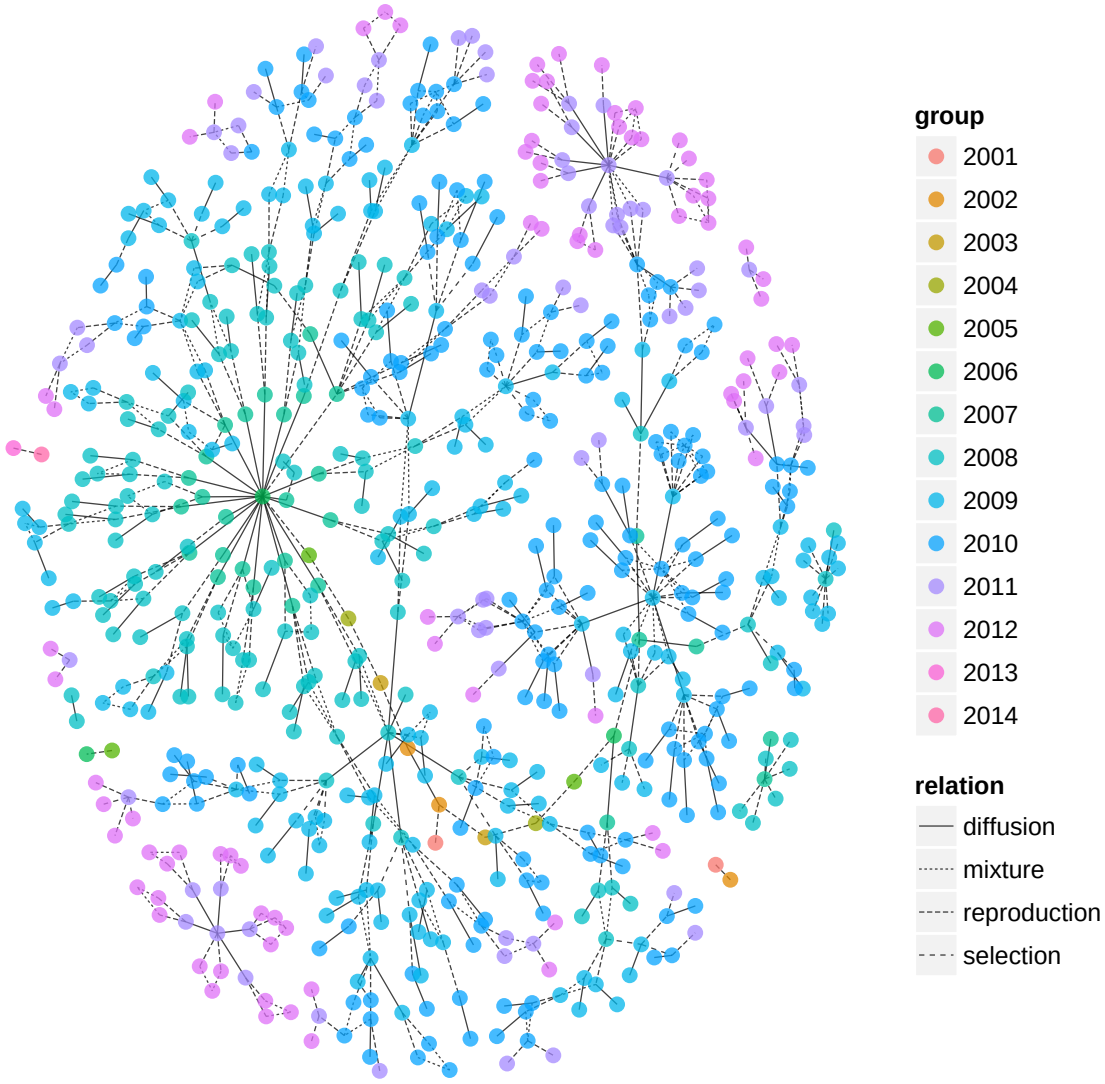
```
names(default_network_ggplot)

##  [1] "network-network"
##  [2] "network-reproduction-sown"
##  [3] "network-reproduction-harvested"
##  [4] "network-reproduction-positive-inter-selected"
##  [5] "network-reproduction-negative-inter-selected"
##  [6] "network-diffusion-sent"
##  [7] "network-diffusion-received"
##  [8] "network-diffusion-relation"
##  [9] "network-mixture"
## [10] "network-positive-intra-selected"
## [11] "network-reproduction-crossed"
```

It is possible to get only one plot by choosing one of the above name with the argument `ggplot.type`. Each `ggplot.type` with default and custom arguments are explained in the following sub-sections.

3.2.1 `ggplot.type = "network-network"`

Default arguments By default, the following network is display:

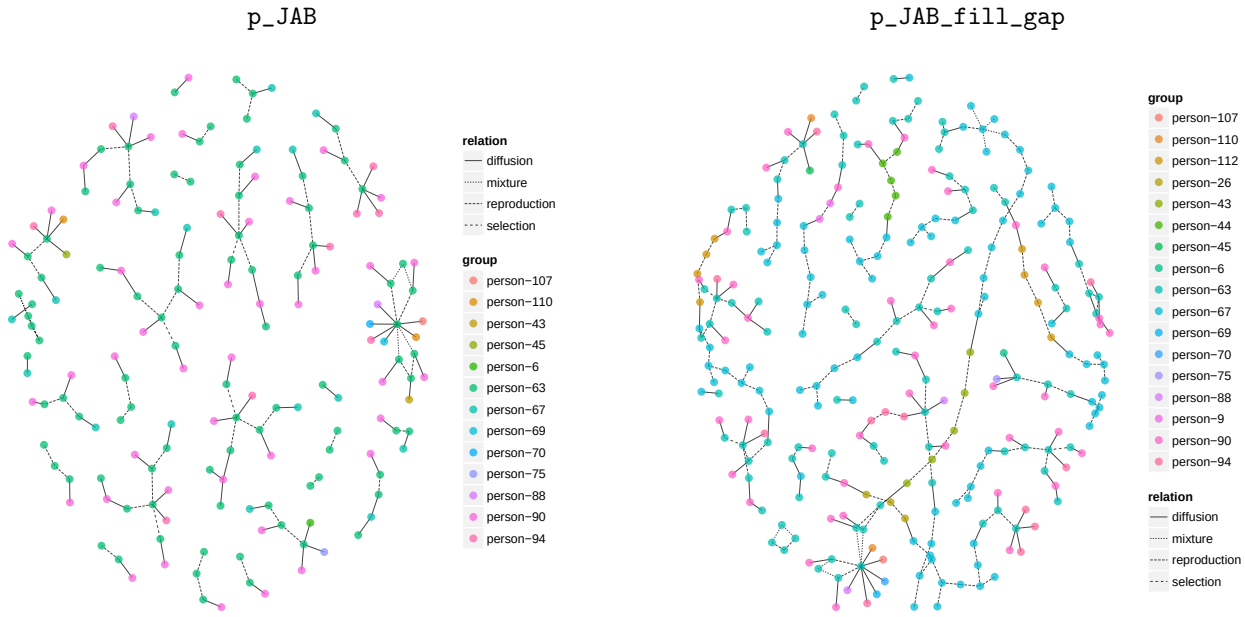
```
p_net_RdR = default_network_ggplot$"network-network"
p_net_RdR
```



Regarding the argument `fill.diffusion.gap` in the function `get.data`:

```
p_JAB = get.ggplot(
        data_network_JAB,
        ggplot.type = "network-network",
        vertex.color = "person"
        )
p_JAB = p_JAB$`network-network`

p_JAB_fill_gap = get.ggplot(
        data_network_JAB_fill_gap,
        ggplot.type = "network-network",
        vertex.color = "person"
        )
p_JAB_fill_gap = p_JAB_fill_gap$`network-network`
```

We can see that extra information are added to know from where come the seed-lots.



Custom arguments It is possible to tune the following arguments (Table 1):

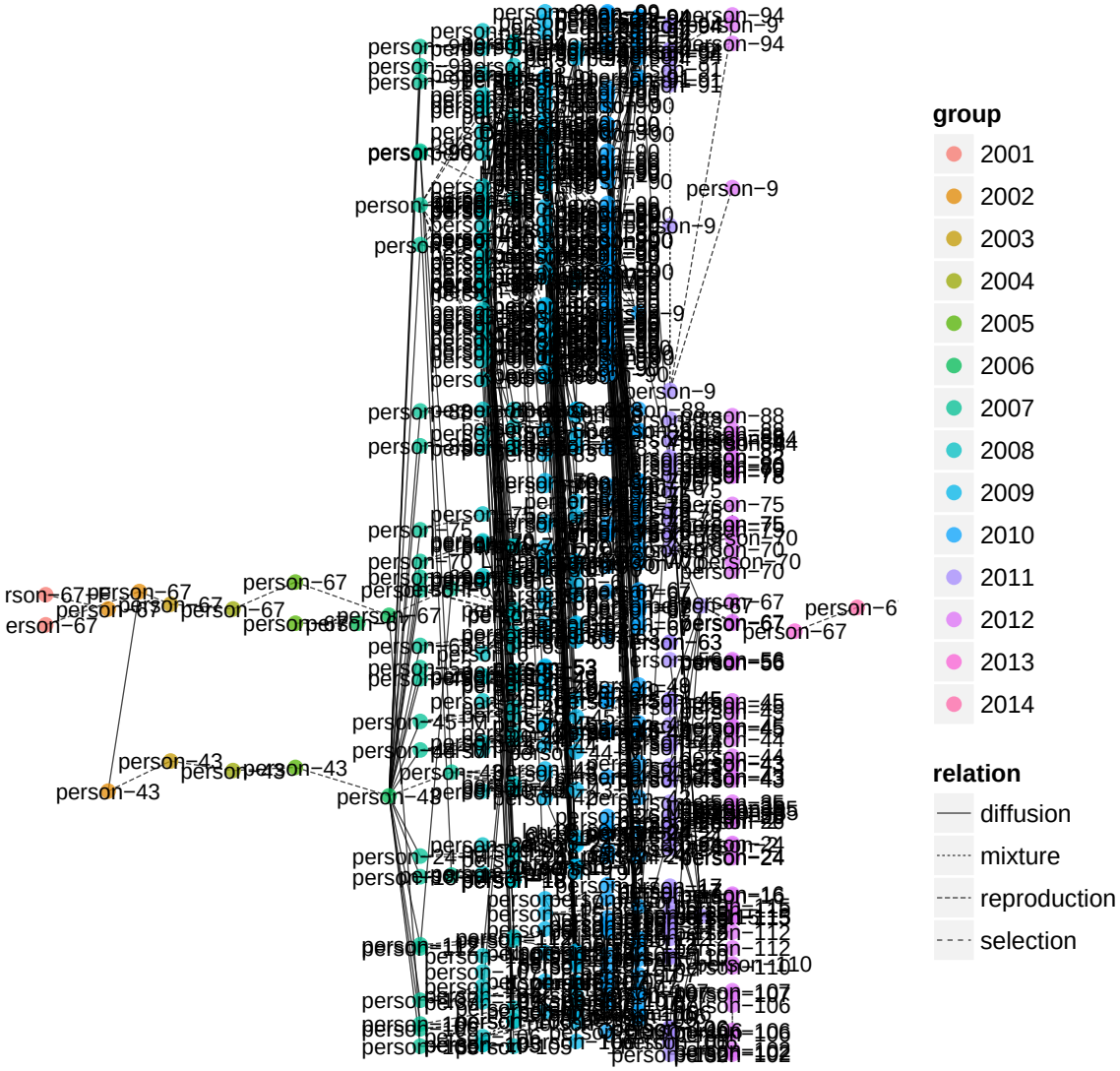
argument	default value	description
`filter.on`	`"father-son"`	it chooses on which seed-lots the filters is applied: `"son"`, `"father"` or `"father-son"`.
`vertex.size`	`3`	size of the vertex
`vertex.color`	`"year"`	color of the vertex. It can be chosen according to `"person"`, `"germplasm"` or `"year"`. If `NULL`, it is in black.
`organise.sl`	`FALSE`	organise seed-lots for an easier visualisation
`hide.labels.parts`	`"all"`	parts of the label hidden: `"germplasm"`, `"person"`, `"year"`, `"person:germplasm"`, `"year:germplasm"`, `"person:year"`, `"all"`. `"all"` means that no labels are dispayed. If `NULL` labels are displayed. Labels are based on seed-lots names under the form `germplasm_year_person_digit`. For easier visualisation, Digit is never display unless you choose `NULL`.
`labels.sex`	`TRUE`	if `TRUE`, display the sex of the seed-lot if it has been used in a cross. Nothing is displayed if hide.labels.parts = `"all"`.
`labels.generation`	`TRUE`	if TRUE, display generation for each reproduction
`labels.size`	`3`	size of the labels

Table 1: Possible arguments to custom arguments regarding network.

For example,

```
get.ggplot(
        data_network,
        ggplot.type = "network-network",
        organise.sl = TRUE,
        hide.labels.parts = "year:germplasm",
        vertex.color = "year"
        )
```

```
## $`network-network`
```



3.2.2 `ggplot.type = "network-reproduction-harvested"`

`ggplot.type = "network-reproduction-harvested"` corresponds to seed-lots that have been harvested after a reproduction.

Two plots are possibles regarding `ggplot.display`.

argument	default value	description
`ggplot.display`	`NULL`	`"barplot"` or `"map"`. It can be a vector of several elements i.e. `c("barplot", "map")`. `NULL` by default: both are done.

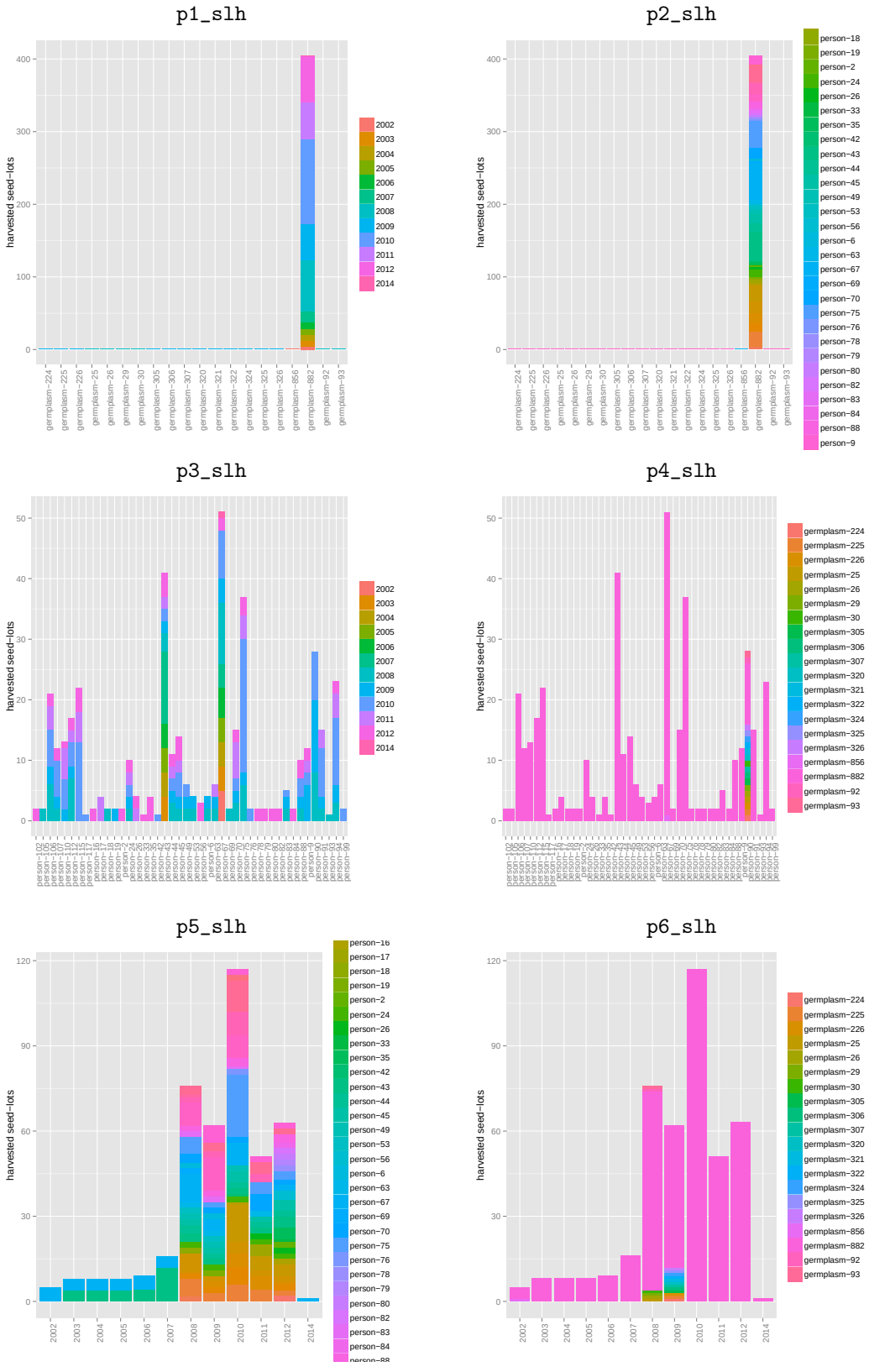
`ggplot.display = barplot` For `ggplot.display = "barplot"`, you may chose what you want in the x axis (`x.axis` argument) and in color (`in.col` argument). The possible values for `x.axis` and `in.col` are: `germplasm`, `year`, `person`. By default all combinaisons of `x.axis` and `in.col` are done with default argument settings. The name of the plot is under the form `x.axis-in.col`.

- Default arguments

```
p_slh = default_network_ggplot$`network-reproduction-harvested`$barplot
p1_slh = p_slh$`germplasm-year`$`x.axis-1|in.col-1`
p2_slh = p_slh$`germplasm-person`$`x.axis-1|in.col-1`
p3_slh = p_slh$`person-year`$`x.axis-1|in.col-1`
p4_slh = p_slh$`person-germplasm`$`x.axis-1|in.col-1`
p5_slh = p_slh$`year-person`$`x.axis-1|in.col-1`
p6_slh = p_slh$`year-germplasm`$`x.axis-1|in.col-1`
```

with

plot	x.axis	in.col
p1_slh	"germplasm"	"year"
p2_slh	"germplasm"	"person"
p3_slh	"person"	"year"
p4_slh	"person"	"germplasm"
p5_slh	"year"	"person"
p6_slh	"year"	"germplasm"



- Custom arguments

  It is possible to tune the following arguments:

argument	default value	description
`x.axis`	`NULL`	factor display on the x.axis of a plot: `"germplasm"`, `"year"` or `"person"` referring to the attributes of a seed-lots. If NULL, all the combinaison are done for `x.axis` and `in.col`.
`in.col`	`NULL`	display in color of a plot: `"germplasm"`, `"year"` or `"person"` referring to the attributes of a seed-lots. If `NULL`, `in.col` is not displayed.
`nb_parameters_per_plot_x.axis`	`NULL`	the number of parameters per plot on `x.axis` argument
`nb_parameters_per_plot_in.col`	`NULL`	the number of parameters per plot for `in.col` argument

Table 2: Possible arguments to custom arguments regarding barplot.

For example, for a given `x.axis` and a given `in.col`:

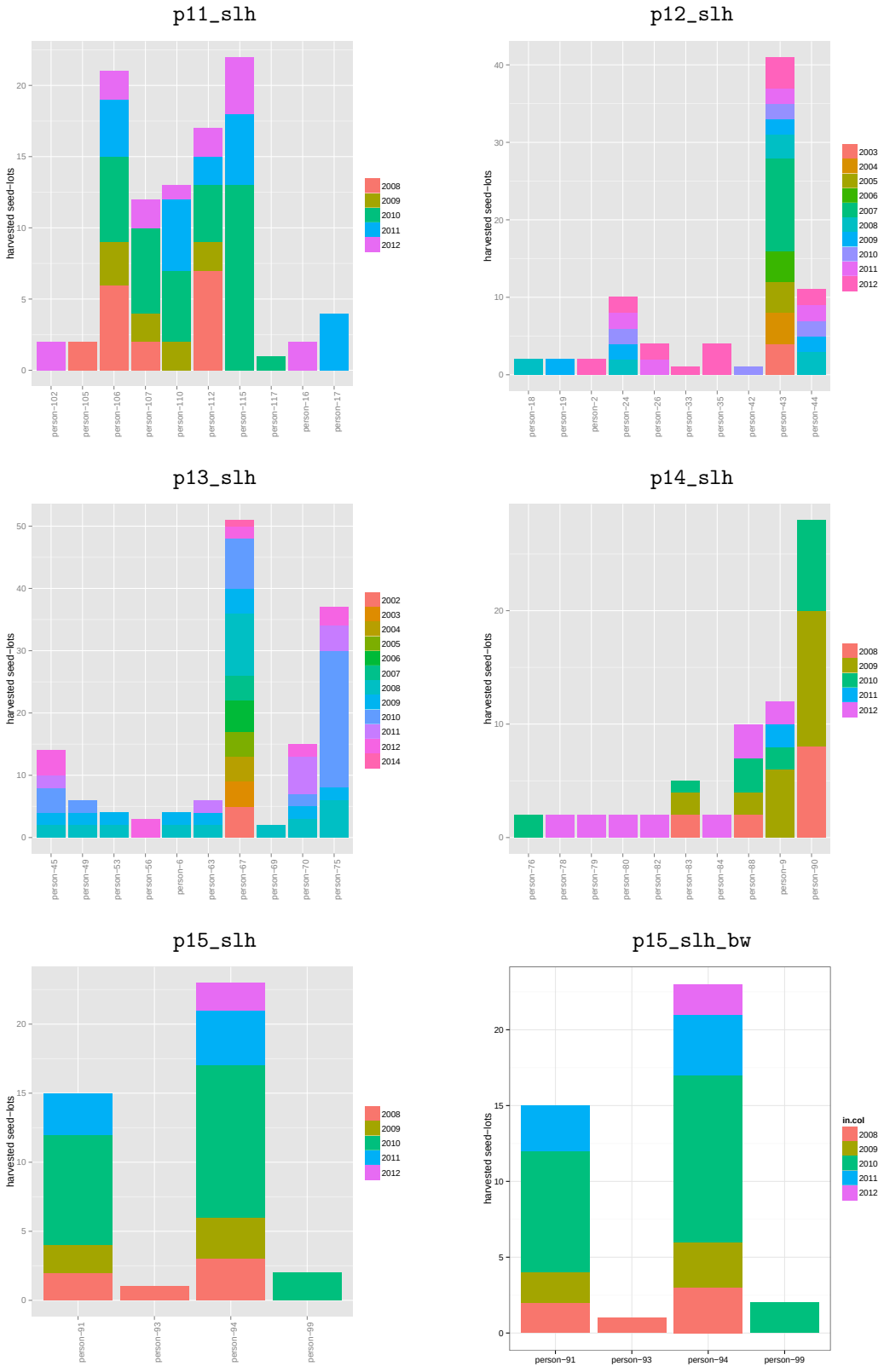
```
p_slh = get.ggplot(
        data_network,
        ggplot.type = "network-reproduction-harvested",
        ggplot.display = "barplot",
        x.axis = "person",
        in.col = "year"
        )
```

It can be useful to separate the plot in several plots regarding `x.axis`:

```
p1_slh = get.ggplot(
        data_network,
        ggplot.type = "network-reproduction-harvested",
        ggplot.display = "barplot",
        x.axis = "person",
        in.col = "year",
        nb_parameters_per_plot_x.axis = 10
        )

p11_slh = p1_slh$`network-reproduction-harvested`$barplot$`person-year`$`x.axis-1|in.col-1`
p12_slh = p1_slh$`network-reproduction-harvested`$barplot$`person-year`$`x.axis-2|in.col-1`
p13_slh = p1_slh$`network-reproduction-harvested`$barplot$`person-year`$`x.axis-3|in.col-1`
p14_slh = p1_slh$`network-reproduction-harvested`$barplot$`person-year`$`x.axis-4|in.col-1`
p15_slh = p1_slh$`network-reproduction-harvested`$barplot$`person-year`$`x.axis-5|in.col-1`
```

Note that you can easily change settings of the plot as it is a ggplot object, for example

```
p15_slh_bw = p15_slh + theme_bw()
```

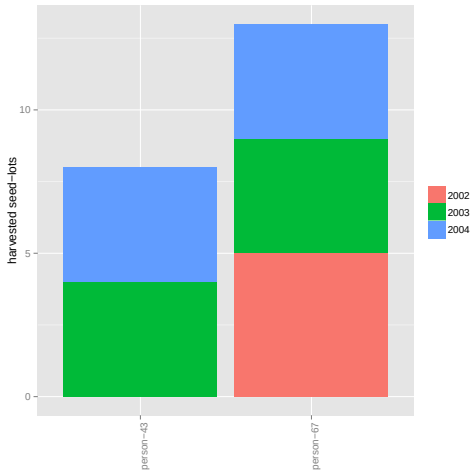


Or for `in.col`:

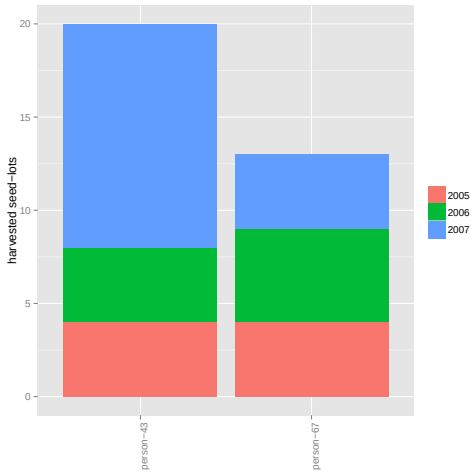
```
p2_slh = get.ggplot(
        data_network,
        ggplot.type = "network-reproduction-harvested",
```

```
        ggplot.display = "barplot",
        x.axis = "person",
        in.col = "year",
        nb_parameters_per_plot_in.col = 3
        )

p21_slh = p2_slh$`network-reproduction-harvested`$barplot$`person-year`$`x.axis-1|in.col-1`
p22_slh = p2_slh$`network-reproduction-harvested`$barplot$`person-year`$`x.axis-1|in.col-2`
p23_slh = p2_slh$`network-reproduction-harvested`$barplot$`person-year`$`x.axis-1|in.col-3`
```

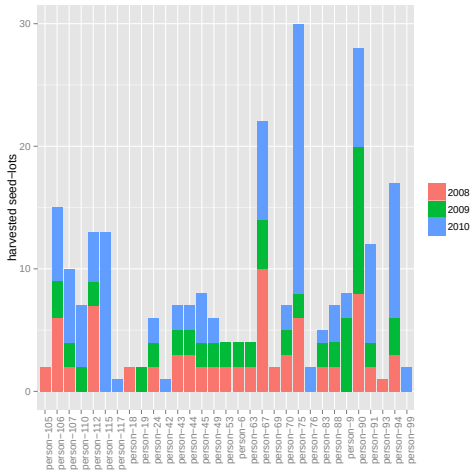
p21_slh



p22_slh



p23_slh



Or for both, for each subset of `x.axis`, you have the set of `in.col`.

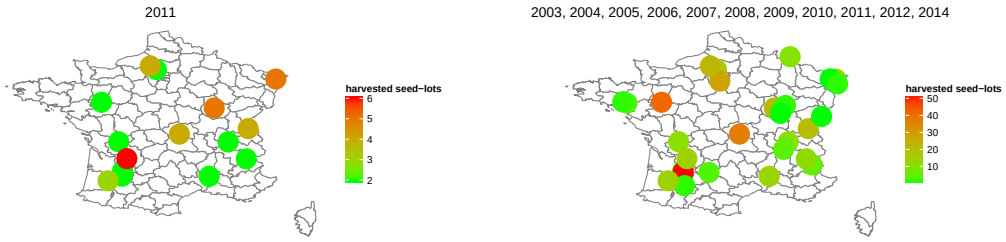
`ggplot.display = map` There are as many maps as years and a map with all the years mixed. The scale represent the number of seed-lots harvected.

- Default arguments

```
m_slh = default_network_ggplot$`network-reproduction-harvested`$map
names(m_slh)
```

```
##  [1] "map-[2002]"
##  [2] "map-[2003]"
##  [3] "map-[2004]"
##  [4] "map-[2005]"
##  [5] "map-[2006]"
##  [6] "map-[2007]"
##  [7] "map-[2008]"
##  [8] "map-[2009]"
##  [9] "map-[2010]"
## [10] "map-[2011]"
## [11] "map-[2012]"
## [12] "map-[2014]"
## [13] "map-[2002, 2003, 2004, 2005, 2006, 2007, 2008, 2009, 2010, 2011, 2012, 2014]"

m4_slh = m_slh$`map-[2011]`
m9_slh = m_slh$`map-[2002, 2003, 2004, 2005, 2006, 2007, 2008, 2009, 2010, 2011, 2012, 2014]`
```

`m4_slh` `m9_slh`



- Custom arguments

  It is possible to tune the following arguments (Table 3):

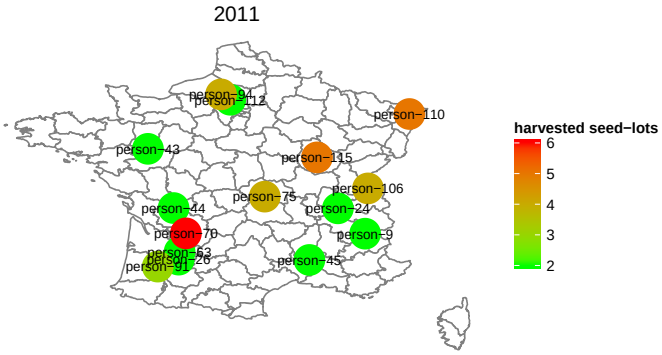
argument	default value	description
`hide.labels.parts`	`NULL`	can be NULL or "all" as only person can be display. `NULL` by default
`labels.size`	`3`	size of the labels
`location.map`	`"france"`	location of the map see `?map` for more details
`pie.size`	`.5`	size of the pie when using pies

Table 3: Possible arguments to custom arguments regarding maps.

For example,

```
p_slh = get.ggplot(
        data_network,
```

```
        ggplot.type = "network-reproduction-harvested",
        ggplot.display = "map",
        hide.labels.parts = NULL
        )

p_slh$`network-reproduction-harvested`$map$`map-[2011]`
```



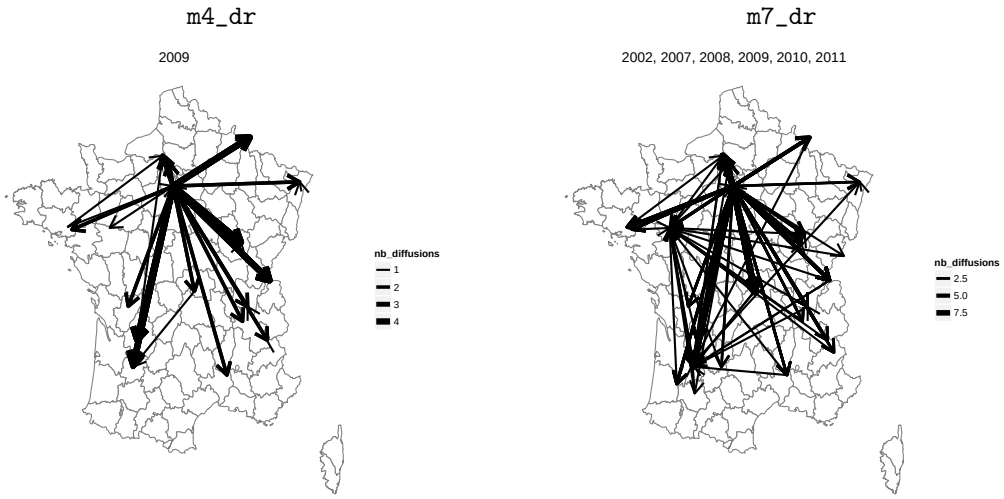
3.2.3 `ggplot.type = "network-diffusion-relation"`

`ggplot.type = "network-diffusion-relation"` corresponds to diffusion of seed-lots on a map. Note that `ggplot.display` is useless here.

There are as many maps as years and a map with all the years mixed. The size of the arrows is proprotionnal to the number of diffusions (`nb_diffusions`).

- Default arguments

```
m_dr = default_network_ggplot$`network-diffusion-relation`$map
names(m_dr)

## [1] "map-[2002]"
## [2] "map-[2007]"
## [3] "map-[2008]"
## [4] "map-[2009]"
## [5] "map-[2010]"
## [6] "map-[2011]"
## [7] "map-[2002, 2007, 2008, 2009, 2010, 2011]"

m4_dr = m_dr$`map-[2009]`
m7_dr = m_dr$`map-[2002, 2007, 2008, 2009, 2010, 2011]`
```



- Custom arguments

  Arguments can be customed related to maps. See Table 3.

3.2.4 `ggplot.type = "network-reproduction-crossed"`

Information on the crosses, their parents (mother and father) and grandparents (grandmother and grandfather). It may be useful to know information on grandparents, especially location, if the cross have been done on another location.

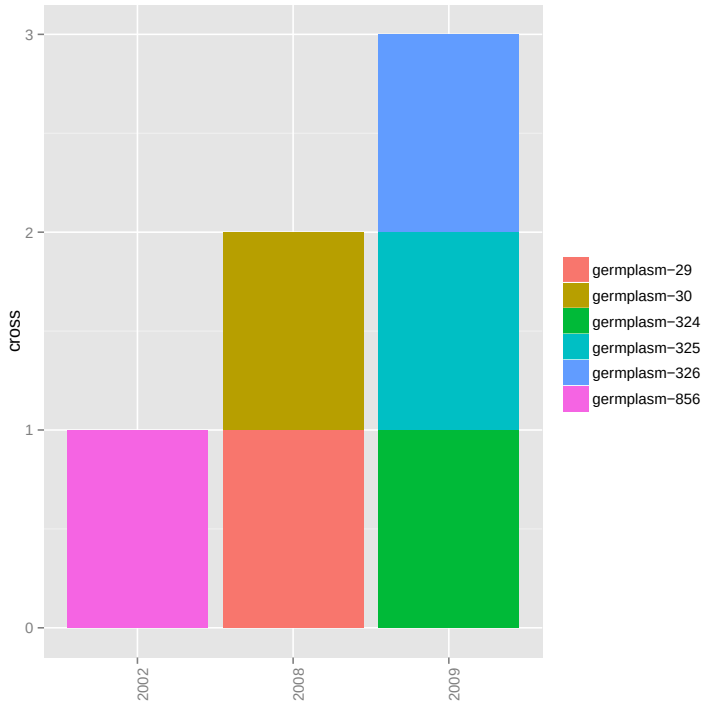
Note that in this example, it represents all the cross where `Rouge-du-Roc` is involved as `"father"` or `"son"` in relation. This is because in `get.data`, the following arguments were set: `germplasm.in = Rouge-du-Roc` and `filter.on = "father-son"`.

- Default arguments

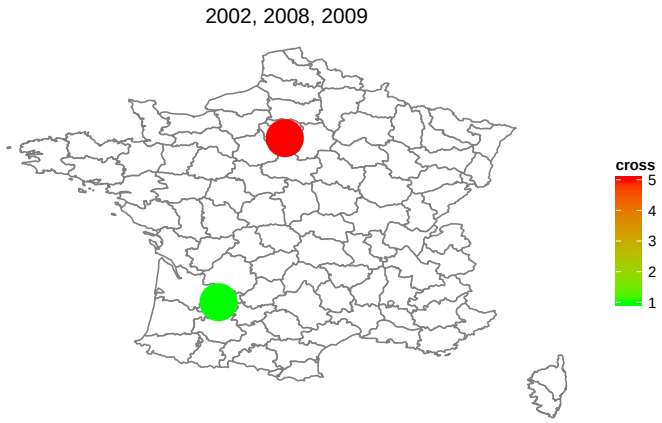
```
cb = default_network_ggplot$`network-reproduction-crossed`$barplot
names(cb)

## [1] "cross"       "father"      "grandfather" "mother"
## [5] "grandmother"

cm = default_network_ggplot$`network-reproduction-crossed`$map
names(cm)

## [1] "cross"       "father"      "grandfather" "mother"
## [5] "grandmother"
```

  For example:

```
cb$cross$barplot$`year-germplasm`$`x.axis-1|in.col-1`
```




```
cm$cross$map$`map-[2002, 2008, 2009]`
```



- Custom arguments

  Arguments can be customed related to barplots (Table 2) or maps (Table 3).

3.2.5 Others `ggplot.type`

For the others `ggplot.type`, as it is exactly the same types of plots than for `ggplot.type = "network-reproduction-harv` no examples are displayed in this vignette.

The following table describe the different `ggplot.type`:

ggplot.type	description
"network-reproduction-sown"	seed-lots that have been sown.
"network-reproduction-positive-inter-selected"	seed-lots that have been harvested after a reproduction and have been sown the next season. It corresponds to positive inter selection.
"network-reproduction-negative-inter-selected"	seed-lots that have been harvested after a reproduction and have NOT been sown the next season. It corresponds to negative inter selection.
"network-diffusion-sent"	seed-lots that have been sent to another location.
"network-diffusion-received"	seed-lots that have been received to a given location.
"network-mixture"	seed-lots that have been mixted into a mixture. Note that mixture of replications are not counted here.
"network-positive-intra-selected"	seed-lots where intra germplasm selection have been done. It corresponds to mass selection.

3.3 Get the tables

It is possible to get the table from the data used for the plot. For example,

```
p = default_network_ggplot$`network-reproduction-crossed`$barplot$cross$barplot
p = p$`year-germplasm`$`x.axis-1|in.col-1`
tab_p = get.table(p)
tab_p

##   mean     germplasm year
## 1    1  germplasm-29 2008
## 2    1  germplasm-30 2008
## 3    1 germplasm-324 2009
## 4    1 germplasm-325 2009
## 5    1 germplasm-326 2009
## 6    1 germplasm-856 2002
```

4 Data linked to the seed-lots and to the relations between seed-lots

4.1 Get the data set

Three types of queries are possibles:

- `query.type = "data-classic"` to study classic variables for each seed-lots or relation between seed-lots (subsection 4.1.1)
- `query.type = "data-S"` to study selection differential (subsection 4.1.2)
- `query.type = "data-SR"` to study response to selection (subsection 4.1.2)

You can set filters to the query with the following argument :

- `filter.in` to choose nothing expect `filter.in`
- `filter.out` to choose everything expect `filter.out`

values of `filter` can be `germplasm`, `germplasm.type`, `year`, `person`, `project`, `seed-lots`, `relation`, `reproduction type` or `variable`.

It is important to choose if you apply the filter on the `father` or the `son` of a relation. This can be done with the argument `filter.on`. Possibles values are `"father"`, `"son"` or `"father-son"`.

You can either query information on relation between seed-lots or on seed-lots. This is set with the argument `data.type`. Possibles values are

- `"relation"` for data linked to relation between seed lots and
- `"seed-lots"` for data linked to seed lots

The function returns a list with

- a data frame with the data set
- a list with data set with individuals that are correlated for a set of variables
- the description of methods used for each variable in with its description and units

Note that for data linked to seed-lots, all the data are correlated as there is one measure for a given seed-lot. Therefore the element of the list for correlated data is always NULL. For data linked to relations, as it can be linked to individual within a seed-lot, data may be correlated (data taken on ther same individual) or not.

A data set with only correlated variables for each individual is useful when doing multivariate analysis such as PCA.

The name of the variables are under the form `variable_name--methods`.

In the following examples, three variables are studied:

```
vec_variables = c("plant_height", "spike_lenght", "spike_weight")
```

4.1.1 `query.type = "data-classic"`

```
if(use.get.data){
        data_classic = get.data(
                db_user = info_db$db_user, db_host = info_db$db_host, # db infos
                db_name = info_db$db_name, db_password = info_db$db_password, # db infos
                query.type = "data-classic", # data-classic query
                person.in = "RAB", # person to keep
                filter.on = "father-son", # filters on father AND son
```

```
                data.type = "relation", # data linked to relation between seed-lots
                variable = vec_variables, # the variables to display
                project.in = "PPB" # the project
                )
} else {
        # 1. Query SHiNeMaS ...
        # 2. Set up data set ...
        # |============================================================| 100%

        #data_classic = encrypt.data(data_classic)
        #The key has been written in /home/pierre/key_data-classic-relation_Tue Nov 24 11:59:59 2015.
        #data_classic = data_classic
        load("./data/data_classic.RData")
}

names(data_classic$data)

## [1] "data"
## [2] "data.with.correlated.variables"
## [3] "methods"

colnames(data_classic$data$data)

##  [1] "son"                          "son_ind"
##  [3] "son_year"                     "son_germplasm"
##  [5] "son_germplasm_type"           "son_person"
##  [7] "son_alt"                      "son_long"
##  [9] "son_lat"                      "father"
## [11] "father_year"                  "father_germplasm"
## [13] "father_germplasm_type"        "father_person"
## [15] "father_alt"                   "father_long"
## [17] "father_lat"                   "reproduction_id"
## [19] "reproduction_type"            "selection_id"
## [21] "selection_person"             "mixture_id"
## [23] "diffusion_id"                 "X"
## [25] "Y"                            "block"
## [27] "project"                      "correlation_group"
## [29] "ID"                           "spike_weight---spike_weight"
## [31] "plant_height---plant_height"  "spike_lenght---spike_length_F"
## [33] "spike_lenght---spike_lenght"

head(data_classic$methods)

## NULL
```

The data with correlated variables :

```
colnames(data_classic$data$data.with.correlated.variables)

## NULL
```

4.1.2 Selection differential (`query.type = "data-S"`), response to selection (`query.type = "data-SR"`) and heritability

For a given trait, selection differential corresponds to the difference between mean of selected spikes and mean of bulk (i.e. spikes that have not been selected). Response to selection correponds to the difference between mean of spikes coming from the selected spikes and the spikes coming from the bulk (Figure 3).

Selection differential (S) and response to selection (R) are linked with the realized heritability (h_r^2):

$$R = h_r^2 \times S$$

See appendix B for more details on the theroy behind this.

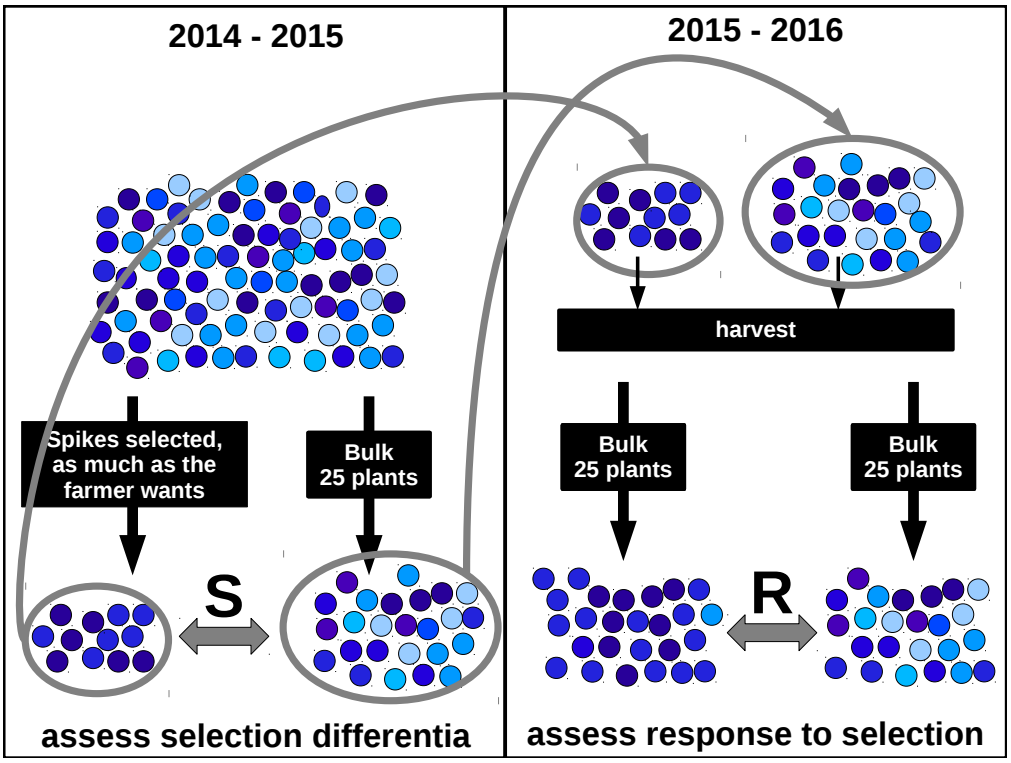


Figure 3: Seletion differential (S) in 2014-2015 and response to selection (R) in 2015-2016. Circles and arrows in gray represent the seed-lots that have been sown in 2015 after harvest in 2015.

`query.type = "data-S"` The data frame returned has a column `"expe"` which corresponds to an id of one selection differential

```
if(use.get.data){
        data_S = get.data(
                db_user = info_db$db_user, db_host = info_db$db_host,
                db_name = info_db$db_name, db_password = info_db$db_password,
                query.type = "data-S",
                person.in = "RAB",
                filter.on = "father-son",
                data.type = "relation",
                variable = vec_variables,
                project.in = "PPB"
                )
} else {
        # 1. Query SHiNeMaS ...
        # 2. Set up data set ...
        # |============================================================| 100%

        #data_S = encrypt.data(data_S)
        #The key has been written in /home/pierre/key_data-S-relation_Tue Nov 24 12:01:25 2015.RData
        load("./data/data_S.RData")
```

```
}

colnames(data_S$data$data)
```

```
##  [1] "son"                           "expe"
##  [3] "sl_statut"                     "expe_name"
##  [5] "expe_name_2"                   "son_ind"
##  [7] "son_year"                      "son_germplasm"
##  [9] "son_germplasm_type"            "son_person"
## [11] "son_alt"                       "son_long"
## [13] "son_lat"                       "father"
## [15] "father_year"                   "father_germplasm"
## [17] "father_germplasm_type"         "father_person"
## [19] "father_alt"                    "father_long"
## [21] "father_lat"                    "reproduction_id"
## [23] "reproduction_type"             "selection_id"
## [25] "selection_person"              "mixture_id"
## [27] "diffusion_id"                  "X"
## [29] "Y"                             "block"
## [31] "project"                       "correlation_group"
## [33] "ID"                            "plant_height---plant_height"
## [35] "spike_weight---spike_weight"   "spike_lenght---spike_lenght"
## [37] "spike_lenght---spike_length_F"
```

`query.type = "data-SR"` The data frame returned has a column `"expe"` which corresponds to an id of one selection differential and the corresponding response to selection

The query takes into account when selection have been done in a seed lot, that this seed lot have been merged and then have been sown. It is the case when selection have been carried out in a replication that have been merge after. Even if this case should not arrise, it may happen.

```
if(use.get.data){
        data_SR = get.data(
                db_user = info_db$db_user, db_host = info_db$db_host,
                db_name = info_db$db_name, db_password = info_db$db_password,
                query.type = "data-SR",
                person.in = "RAB",
                filter.on = "father-son",
                data.type = "relation",
                variable = vec_variables,
                project.in = "PPB"
                )
} else {
        # 1. Query SHiNeMaS ...
        # 2. Set up data set ...
        # |============================================================| 100%

        #data_SR = encrypt.data(data_SR)
        #The key has been written in /home/pierre/key_data-SR-relation_Tue Nov 24 12:02:25 2015.RData
        load("./data/data_SR.RData")
}

colnames(data_SR$data$data)
```

```
##  [1] "son"                           "expe"
##  [3] "sl_statut"                     "expe_name"
```

```
##  [5] "expe_name_2"                   "son_ind"
##  [7] "son_year"                      "son_germplasm"
##  [9] "son_germplasm_type"            "son_person"
## [11] "son_alt"                       "son_long"
## [13] "son_lat"                       "father"
## [15] "father_year"                   "father_germplasm"
## [17] "father_germplasm_type"         "father_person"
## [19] "father_alt"                    "father_long"
## [21] "father_lat"                    "reproduction_id"
## [23] "reproduction_type"             "selection_id"
## [25] "selection_person"              "mixture_id"
## [27] "diffusion_id"                  "X"
## [29] "Y"                             "block"
## [31] "project"                       "correlation_group"
## [33] "ID"                            "plant_height---plant_height"
## [35] "spike_weight---spike_weight"   "spike_lenght---spike_lenght"
## [37] "spike_lenght---spike_length_F"
```

4.2 Get the ggplots

To get the plots, use the function `get.ggplot`.

The name of the variables are under the form `variable_name--methods`:

```
vec_variables = c(
        "plant_height---plant_height",
        "spike_lenght---spike_length_F",
        "spike_weight---spike_weight"
        )
```

4.2.1 ggplots for `data-classic`

- barplot, boxplot, interaction, radar and biplot
  - Default arguments

```
default_data_classic_ggplot = get.ggplot(
        data_classic,
        vec_variables = vec_variables
        )

## As ggplot.type is NULL, ggplot.Type is set to data-barplot, data-boxplot, data-interactio
data-radar, data-biplot
## As x.axis and in.col are NULL, all the combinaisons of x.axis and in.col are done
for data-barplot, data-boxplot and data-interaction.
## As in.col is NULL, each in.col are done for data-radar and data-biplot.

## [1] "A Virer quand les données seront propres dans get.data"
## [1] "A Virer quand les données seront propres dans get.data"
## [1] "A Virer quand les données seront propres dans get.data"

## For data-biplot, hide.labels.parts has been set to NULL instead of "all".
```

By default, the following plots are done:

```
names(default_data_classic_ggplot)

## [1] "data-barplot"     "data-boxplot"     "data-interaction"
## [4] "data-radar"       "data-biplot"
```

which correpond to `ggplot.type` arguments:

ggplot.type	description
"data-barplot"	barplot, there is one barplot per variable
"data-boxplot"	boxplot, there is one boxplot per variable
"data-interaction"	interaction, there is one interation plot per variable
"data-radar"	radar, there is one radar per set of variables (i.e. the whole `vec_variables`)
"data-biplot"	biplot, there is one biplot per pairs of variables

For `ggplot.type = data-barplot`, `data-boxplot` and `data-interaction`, you may chose what you want in the x axis (`x.axis` argument) and in color (`in.col` argument). The posible values for `x.axis` and `in.col` are: `germplasm`, `year`, `person`. By default all combinaisons of `x.axis` and `in.col` are done with default argument settings. The name of the plot is under the form `x.axis-in.col`.

```
names(default_data_classic_ggplot$`data-barplot`)
```

```
## [1] "germplasm-year"   "germplasm-person" "person-year"
## [4] "person-germplasm" "year-person"      "year-germplasm"
```

Knowing a combinaison of `x.axis` and `in.col`, choose the variable you want to get. For example:

* For barplot:

```
p_cal_ph_bar_all = default_data_classic_ggplot$`data-barplot`$`germplasm-year`
p_cal_ph_bar = p_cal_ph_bar_all$`plant_height---plant_height`$`x.axis-1|in.col-1`
```

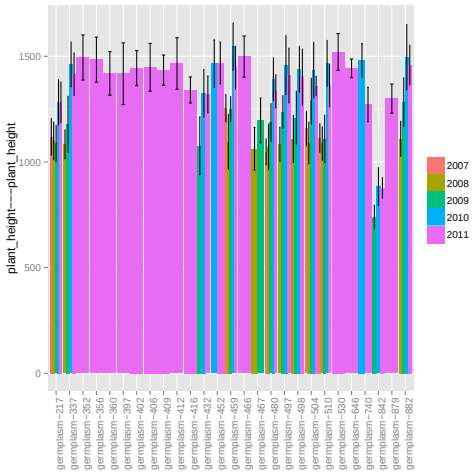
* For bolxplot:

```
p_cal_ph_box_all = default_data_classic_ggplot$`data-boxplot`$`germplasm-year`
p_cal_ph_box = p_cal_ph_box_all$`plant_height---plant_height`$`x.axis-1|in.col-1`
```

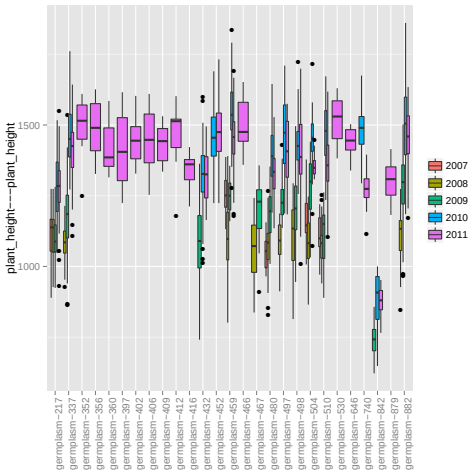
* For interaction plot:

```
p_cal_ph_int_all = default_data_classic_ggplot$`data-interaction`$`year-germplasm`
p_cal_ph_int = p_cal_ph_int_all$`plant_height---plant_height`$`x.axis-1|in.col-1`
```

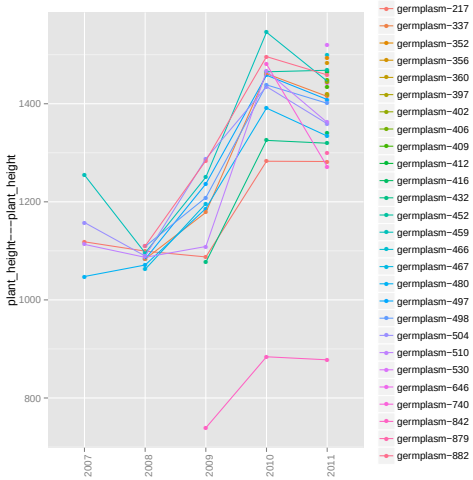
p_cal_ph_bar



p_cal_ph_box



p_cal_ph_int



For `ggplot.type = data-radar`, `data-biplot`, you may chose what you want in color (`in.col` argument). The possible values for `in.col` are: `germplasm`, `year` and `person`.

```
names(default_data_classic_ggplot$`data-radar`)

## [1] "NA-year"      "NA-person"    "NA-germplasm"
```

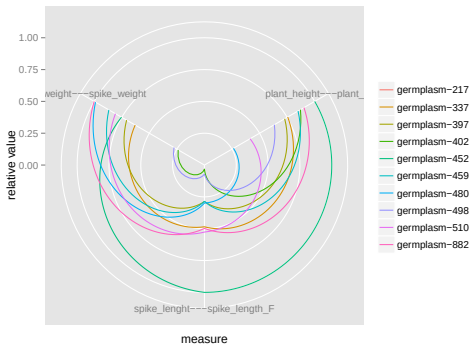
For example,

```
p_cal_rad_all = default_data_classic_ggplot$`data-radar`
p_cal_rad = p_cal_rad_all$`NA-germplasm`$`x.axis-1|in.col-1`
```

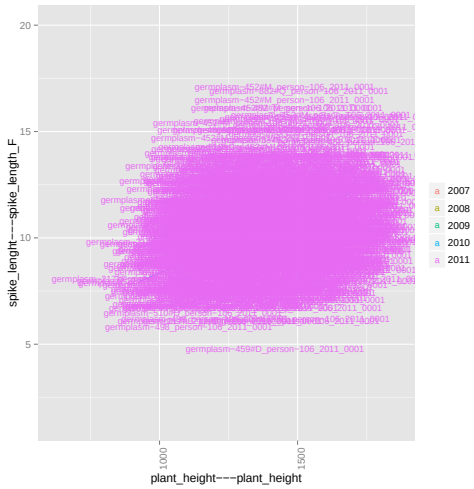
For biplot, you must choose the pair of variables you wish, for example,

```
p_cal_ph_sl_bip_all = default_data_classic_ggplot$`data-biplot`
p_cal_ph_sl_bip = p_cal_ph_sl_bip_all$`NA-year`
p_cal_ph_sl_bip = p_cal_ph_sl_bip$`plant_height---plant_height - spike_lenght---spike_length
p_cal_ph_sl_bip = p_cal_ph_sl_bip$`x.axis-1|in.col-1`
```

`p_cal_rad`



`p_cal_ph_sl_bip`



– Custom arguments

  According to `ggplot.type`, arguments can be customized:

argument	default value	description	data-barplot	data-boxplot	data-interaction	data-radar	data-biplot
ggplot.on	"son"	"father" or "son" depending on which seed-lot you want to plot..	X	X	X	X	X
x.axis	NULL	factor display on the x.axis of a plot: "germplasm", "year" or "person" referring to the attributes of a seed-lots. If NULL, all the combinaison are done for x.axis and in.col.	X	X	X		
in.col	NULL	display in color of a plot: "germplasm", "year" or "person" referring to the attributes of a seed-lots. If NULL, in.col is not displayed. Note it is compulsory for data-biplot and data-radar as in these cases x.axis is not used.	X	X	X	X	X
nb_parameters_per_plot_x.axis	NULL	the number of parameters per plot on x.axis argument	X	X	X		
nb_parameters_per_plot_in.col	NULL	the number of parameters per plot for in.col argument	X	X	X	X	X
hide.labels.parts	"all"	parts of the label hidden: "germplasm", "person", "year", "person:germplasm", "year:germplasm", "person:year", "all". "all" means that no labels are dispayed. If NULL labels are displayed. Labels are based on seed-lots names under the form germplasm_year_person_digit. For "data-biplot", the default value is NULL. For easier visualisation, Digit is never display unless you choose NULL.					X

Table 4: Possible arguments regarding ggplot.type. A cross (X) means that for a given ggplot.type, a given argument can be used

- pies on network
  - Default arguments

    It is possible to add pies on seed-lots represented on a network. The pie represent the distribution of a given variable for a given seed-lot.

    In the following example, as we're working with encrypt data, get.ggplot reverse the transcription to query SHiNeMaS and encrypt again.

    Note that this example is done on a little data set. Indeed, the pies are drawn with a polygon which need lots of points that take memory.

```
if(use.get.data){
        data_classic_bis = get.data(
                db_user = info_db$db_user, db_host = info_db$db_host, # db infos
                db_name = info_db$db_name, db_password = info_db$db_password, # db infos
                query.type = "data-classic", # data-classic query
```

```
                person.in = "LUD", # person to keep
                year.in = 2011, # year to kepp
                filter.on = "father-son", # filters on father AND son
                data.type = "relation", # data linked to relation between seed-lots
                variable = vec_variables, # the variables to display
                project.in = "PPB" # the project
                )

        data_classic_bis = encrypt.data(data_classic_bis)

        pn = get.ggplot(
                data_classic_bis,
                ggplot.type = "data-pie.on.network",
                vec_variables = "spike_weight---spike_weight",
                pie.size = .25,
                hide.labels.parts = "person"
                )

} else {
        # 1. Query SHiNeMaS ...
        # 2. Set up data set ...
        # |============================================================| 100%
        #The key has been written in /home/pierre/key_data-classic-relation_Tue Nov 25 22:29
        load("./data/data_classic_bis.RData")

        # 1. Query SHiNeMaS ...
        # 2. Create network matrix ...
        # 3. Link information to vertex and edges ...
        # The key has been written in /home/pierre/key_network_Wed Nov 25 22:30:16 2015.RDat

        load("./data/pn.RData")
        }
```

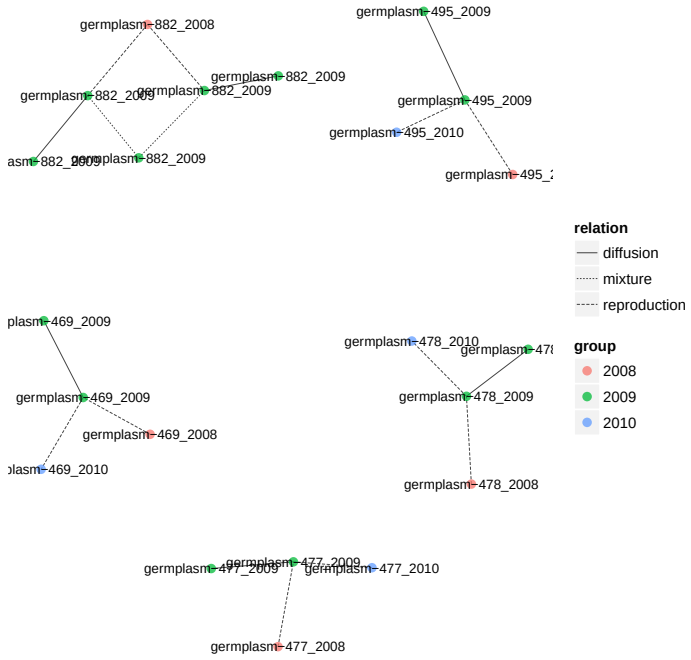
As we're working for one person, we set `hide.labels.parts = "person"`.
The result id divided into two plots: the empty network and the network with the pies.

```
names(pn$`data-pie.on.network`)
```

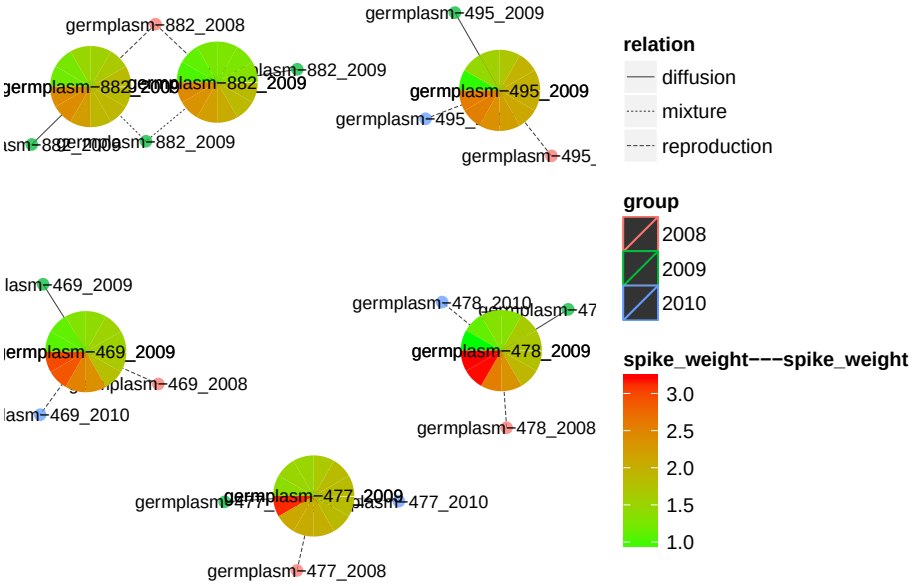
```
## [1] "spike_weight---spike_weight"
```

```
p1 = pn$`data-pie.on.network`$`spike_weight---spike_weight`$network
p2 = pn$`data-pie.on.network`$`spike_weight---spike_weight`$pie.on.network
```

p1


germplasm−882_2008
germplasm−882_2009
germplasm−882_2009
germplasm−882_2009
germplasm−882_2009
germplasm−495_2009
germplasm−495_2009
germplasm−495_2010
germplasm−495_
relation
diffusion
mixture
reproduction
group
2008
2009
2010
germplasm−469_2009
germplasm−469_2009
germplasm−469_2008
germplasm−469_2010
germplasm−478_2010
germplasm−478_2009
germplasm−478_2008
germplasm−477_2009
germplasm−477_2010
germplasm−477_2008

p2


germplasm−882_2008
germplasm−882_2009
germplasm−882_2009
germplasm−882_2009
germplasm−495_2009
germplasm−495_2009
germplasm−495_
relation
diffusion
mixture
reproduction
group
2008
2009
2010
germplasm−469_2009
germplasm−469_2008
germplasm−469_2010
germplasm−478_2010
germplasm−478_2009
germplasm−478_2008
germplasm−477_2009
germplasm−477_2010
germplasm−477_2008
spike_weight−−−spike_weight
3.0
2.5
2.0
1.5
1.0

- Custom arguments
  Arguments can be customed related to network plots (Table 1).

- pies on map
  - Default arguments
    `get.data` is called to get the coordinates data.

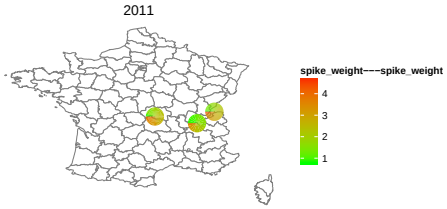
```
if(use.get.data){
        data_classic_ter = get.data(
                db_user = info_db$db_user, db_host = info_db$db_host, # db infos
                db_name = info_db$db_name, db_password = info_db$db_password, # db infos
                query.type = "data-classic", # data-classic query
                germplasm.in = "Rouge-du-Roc", # germplasm to keep
                person.in = c("RAB", "CHD", "JSG"), # person to keep
                filter.on = "father-son", # filters on father AND son
                data.type = "relation", # data linked to relation between seed-lots
                variable = vec_variables, # the variables to display
                project.in = "PPB" # the project
                )

        data_classic_ter = encrypt.data(data_classic_ter)

        pm = get.ggplot(
                data_classic_ter,
                ggplot.type = "data-pie.on.map",
                vec_variables = "spike_weight---spike_weight"
                )
} else {
        load("./data/data_classic_ter.RData")
        load("./data/pm.RData")
        }

names(pm$`data-pie.on.map`)
```

```
## [1] "spike_weight---spike_weight"
```

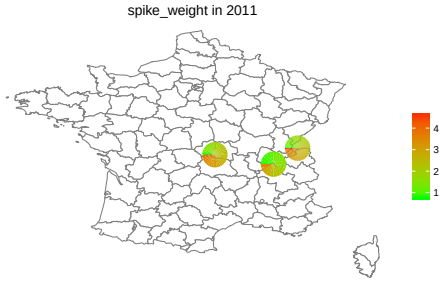
You can get the default ggplot (`p`) or customized it (`p_bis`):

```
p = pm$`data-pie.on.map`$`spike_weight---spike_weight`$`map-[2011]`

p_bis = p +
        ggtitle("spike_weight in 2011") + # change the title
        theme(legend.title=element_blank()) # delete the name of the legend
```

p



p_bis



– Custom arguments

Arguments can be customed related to map plots (Table 3).

For `data-pie.on.map`, it is also possible to choose `x.axis` in order to get a map for each `year` or each `germplasm`.

In the example, as `data_classic` is done on one germplasm (Rouge du Roc), `x.axis = year` has been chosen (default argument).

4.2.2 ggplots for `data-S`

```
default_data_S_ggplot = get.ggplot(
        data_S,
        vec_variables = vec_variables,
        nb_parameters_per_plot_x.axis = 5
        )

## As ggplot.type is NULL, ggplot.Type is set to data-barplot, data-boxplot, data-interaction,
data-radar, data-biplot
## As ggplot.type is NULL and data come from "data-S" or "data-SR", ggplot.type is set to "data-barpl
"data-boxplot", "data-interaction".
## With "data-S" and "data-SR", in.col and x.axis are set automaticaly.

## [1] "A Virer quand les données seront propres dans get.data"
## [1] "A Virer quand les données seront propres dans get.data"
## [1] "A Virer quand les données seront propres dans get.data"
```

By default, the following plots are done with `x.axis` and `in.col` by default (you can not change it):

```
names(default_data_S_ggplot)

## [1] "data-barplot"     "data-boxplot"     "data-interaction"
```

which correpond to ggplot.type arguments as explained in the previous section.

- For barplot:

```
p_S_sw_bar_all = default_data_S_ggplot$`data-barplot`
p_S_sw_bar = p_S_sw_bar_all$`spike_weight---spike_weight`$`x.axis-1|in.col-1`
```

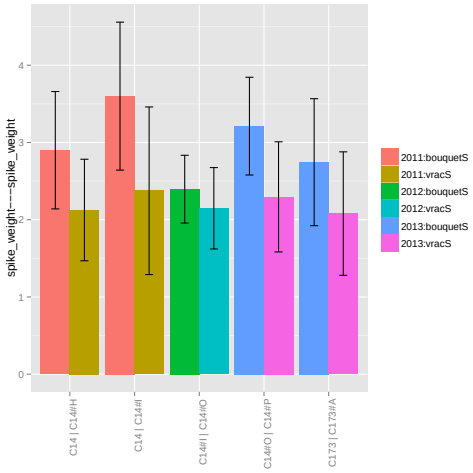
- For boxplot:

```
p_S_sw_box_all = default_data_S_ggplot$`data-boxplot`
p_S_sw_box = p_S_sw_box_all$`spike_weight---spike_weight`$`x.axis-1|in.col-1`
```

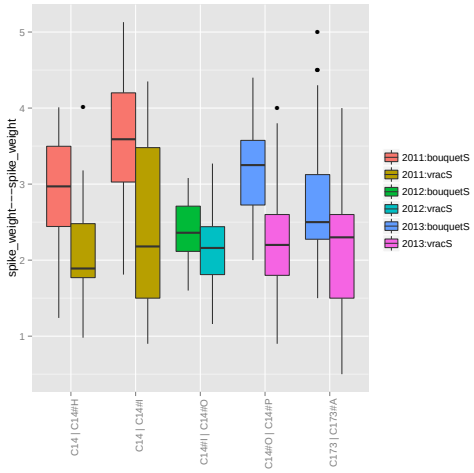
- For interaction plot:

```
p_S_sw_int_all = default_data_S_ggplot$`data-interaction`
p_S_sw_int = p_S_sw_int_all$`spike_weight---spike_weight`$`x.axis-1|in.col-1`
```

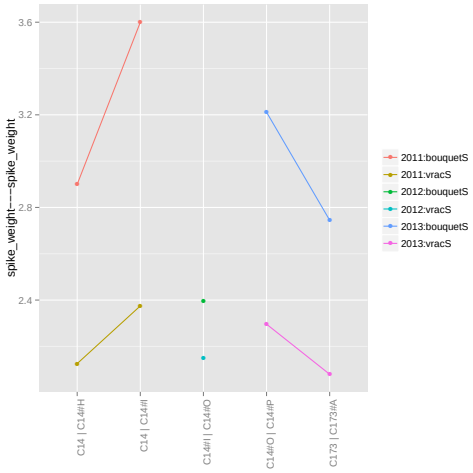
p_S_sw_bar



p_S_sw_box



p_S_sw_int



It is possible to customize the plots as presented in Table 4.

4.2.3 ggplots for `data-SR`

```
default_data_SR_ggplot = get.ggplot(
        data_SR,
        vec_variables = vec_variables,
        nb_parameters_per_plot_x.axis = 5
        )
```

```
## As ggplot.type is NULL, ggplot.Type is set to data-barplot, data-boxplot, data-interaction,
data-radar, data-biplot
## As ggplot.type is NULL and data come from "data-S" or "data-SR", ggplot.type is set to "data-barpl
"data-boxplot", "data-interaction".
## With "data-S" and "data-SR", in.col and x.axis are set automaticaly.

## [1] "A Virer quand les données seront propres dans get.data"
## [1] "A Virer quand les données seront propres dans get.data"
## [1] "A Virer quand les données seront propres dans get.data"
```

By default, the following plots are done with `x.axis` and `in.col` by default (you can not change it):

```
names(default_data_SR_ggplot)

## [1] "data-barplot"     "data-boxplot"     "data-interaction"
```

which correpond to ggplot.type arguments as explained in the previsou section.

- For barplot:

```
p_SR_sw_bar_all = default_data_S_ggplot$`data-barplot`
p_SR_sw_bar = p_SR_sw_bar_all$`spike_weight---spike_weight`$`x.axis-1|in.col-1`
```

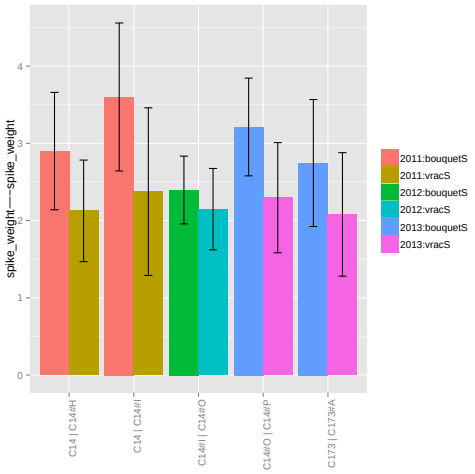
- For boxplot:

```
p_SR_sw_box_all = default_data_S_ggplot$`data-boxplot`
p_SR_sw_box = p_SR_sw_box_all$`spike_weight---spike_weight`$`x.axis-1|in.col-1`
```

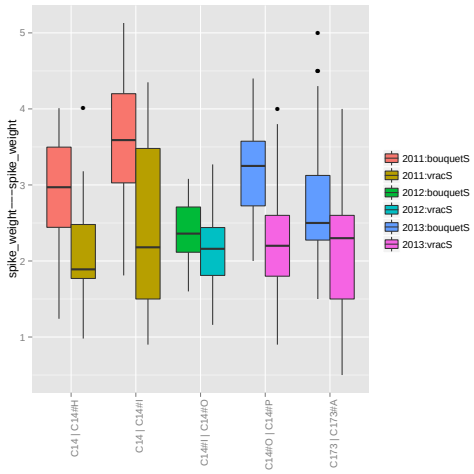
- For interaction plot:

```
p_SR_sw_int_all = default_data_S_ggplot$`data-interaction`
p_SR_sw_int = p_SR_sw_int_all$`spike_weight---spike_weight`$`x.axis-1|in.col-1`
```

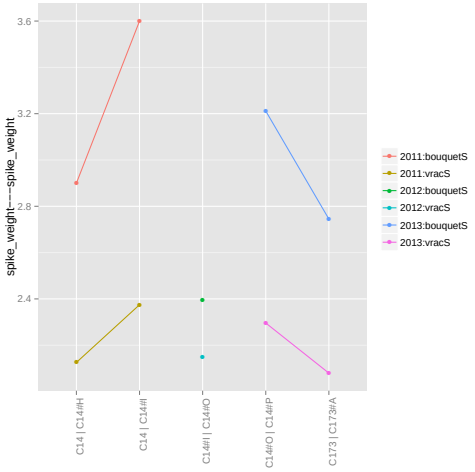
p_SR_sw_bar



p_SR_sw_box



p_SR_sw_int



It is possible to customize the plots as presented in Table 4.

4.3 Get the tables

From the data, to get the tables, use the function `get.table`. Five types of tables can be display according to `table.type` argument :

`table.type`	description
`"raw"`	display raw data. Useful with text for example
`"mean"`	display for each variable columns with mean
`"mean.sd"`	display for each variable columns with mean and standard deviation
`"mean.sd.cv"`	display for each variable columns with mean, standard deviation and coefficient of variation
`"summary"`	display `"Min."`, `"1st Qu."`, `"Median"`, `"3rd Qu."`, `"Max."` of the data

The table always display information with seed-lots on the left side. For information on relation, you must choose if seed-lots displayed in the table are the son or the father with the argument `table.on`. For information on seed-lots, there is no problems.

Then you should set `vec_variables` with the variables displayed in the table.

4.3.1 tables for `data-classic`

– Default arguments

```
tab_class = get.table(
        data_classic,
        table.type = "raw",
        vec_variables = vec_variables
        )
```

the function returns a list with two elements:

```
names(tab_class)

## [1] "duplicated_infos"     "not_duplicated_infos"
```

* `"duplicated_infos"`: lists of two elements with seed-lots involved and variable values

```
tab_class$duplicated_infos$`set-1`
## NULL
```

It is `NULL` as there are no duplicated information here.
An example with duplicated information:

```
if(use.get.data){
        data_classic_4 = get.data(
                db_user = info_db$db_user, db_host = info_db$db_host, # db infos
                db_name = info_db$db_name, db_password = info_db$db_password, # db infos
                query.type = "data-classic", # data-classic query
                person.in = "RAB", # person to keep
                filter.on = "father-son", # filters on father AND son
                data.type = "relation", # data linked to relation between seed-lots
                variable = "sowing_practices", # the variables to display
                project.in = "PPB" # the project
                )

        data_classic_4 = encrypt.data(data_classic_4)
} else {
        load("./data/data_classic_4.RData")
        }

tab_class_4 = get.table(
        data_classic_4,
        table.type = "raw",
        vec_variables = "sowing_practices---sowing_notice_sowing_practices"
        )
```

The information is divided in two:

```
names(tab_class_4$duplicated_infos$`set-1`)
## NULL
```

The seed-lots that are concerned. By default `col_to_display = c("person", "germplasm", "year", "block", "X", "Y")`, therefore, the information is under the form : `person-germplasm-year-bl`

```
tab_class_4$duplicated_infos$`set-1`$`duplicated_infos_seed-lots`
## NULL
```

And the data linked to these seed lots:

```
tab_class_4$duplicated_infos$`set-1`$`duplicated_infos_variables`
## NULL
```

* "not_duplicated_infos": a list with the table of non duplicated information

```
dim(tab_class$not_duplicated_infos$`set-1`)
## [1] 3269   11
```

Note that the threshold up to which the information are duplicated or not is tuned with the argument nb_duplicated_rows (see Custom arguments).
Instead of raw information, some statistics may be useful, for example,

```
tab_class = get.table(
        data_classic,
        table.type = "mean.sd.cv",
        vec_variables = vec_variables
        )
names(tab_class$not_duplicated_infos)

## [1] "set-1"

colnames(tab_class$not_duplicated_infos$`set-1`)

##  [1] "person"
##  [2] "germplasm"
##  [3] "year"
##  [4] "block"
##  [5] "X"
##  [6] "Y"
##  [7] "ID mean"
##  [8] "ID sd"
##  [9] "ID cv"
## [10] "spike_weight---spike_weight mean"
## [11] "spike_weight---spike_weight sd"
## [12] "spike_weight---spike_weight cv"
## [13] "plant_height---plant_height mean"
## [14] "plant_height---plant_height sd"
## [15] "plant_height---plant_height cv"
## [16] "spike_lenght---spike_length_F mean"
## [17] "spike_lenght---spike_length_F sd"
## [18] "spike_lenght---spike_length_F cv"
## [19] "spike_lenght---spike_lenght mean"
## [20] "spike_lenght---spike_lenght sd"
## [21] "spike_lenght---spike_lenght cv"

dim(tab_class$not_duplicated_infos$`set-1`)

## [1] 111  21
```

– Custom arguments
The following arguments can be customized:

argument	default value	description
`nb_row`	`NULL`	the number of rows in the table
`nb_col`	`NULL`	the number of columns for variable in the table. `col_to_display` remains fixed.
`nb_duplicated_rows`	`NULL`	minimum number of duplicated rows for each variable of a table up to which the information is put in only one row.
`col_to_display`	`c("person", "germplasm", "year", "block", "X", "Y")`	columns to display in the table. It can be a vector with `"person"`, `"year"`, `"germplasm"`, `"block"`, `"X"` and `"Y"`. If `NULL`, none of these columns are displayed. The variables follow these columns. For `"data-S"` and `"data-SR"` type, the column `"expe"` and `"sl_statut"` are added by default.
`invert_row_col`	`FALSE`	if `TRUE`, invert row and col in the table. This is possible only for `col_to_display = NULL`.

Table 5: Possible arguments to custom `get.table`.

For examples,

```
tab_class = get.table(
        data_classic,
        table.type = "mean.sd.cv",
        vec_variables = vec_variables,
        nb_row = 30
        )
names(tab_class$not_duplicated_infos)

## [1] "set-1" "set-2" "set-3" "set-4"
```

The information is split into several tables with at least 30 rows.

```
dim(tab_class$not_duplicated_infos$`set-1`)

## [1] 30 21
```

4.3.2 tables for `data-S` and `data-SR`

Tables are the same than for `classic-data`, there is just the column `"expe"` and `"sl_statut"` added by default.

For example,

```
tab_S_msc = get.table(
        data_S,
        table.type = "mean.sd.cv",
        vec_variables = vec_variables,
        nb_col = 3 # Otherwise, the table is too large
        )

colnames(tab_S_msc$not_duplicated_infos$`set-1`)

##  [1] "person"
```

```
##  [2] "germplasm"
##  [3] "year"
##  [4] "block"
##  [5] "X"
##  [6] "Y"
##  [7] "expe"
##  [8] "sl_statut"
##  [9] "plant_height---plant_height mean"
## [10] "plant_height---plant_height sd"
## [11] "plant_height---plant_height cv"
```

It is possible to customize the tables as presented in Table 5.

4.4 Format the data for existing R packages

It is often useful to go beyond descriptive analysis and apply statistical tests to your data. To do so, you need to use existing R packages to perform such analysis.

You need to format your data in order to use these packages. This is done with `format.data` which format your data for the following packages: `PPBstats`.

For example, if you want to use the `PPBstats` package:

```
data_for_PPBstats = format.data(
        data_classic,
        format = "PPBstats"
)

## [1] "Importance d'avoir le meme nombre de colonnes dans toutes les sorties de data ou alors a

head(data_for_PPBstats)

##      year   location     germplasm block    X    Y
## 7702 2009 person-106 germplasm-337     2 <NA> <NA>
## 6415 2009 person-106 germplasm-337     2 <NA> <NA>
## 1529 2009 person-106 germplasm-337     2 <NA> <NA>
## 8908 2009 person-106 germplasm-337     2 <NA> <NA>
## 2718 2009 person-106 germplasm-337     2 <NA> <NA>
## 7703 2009 person-106 germplasm-337     2 <NA> <NA>
##                                           ID
## 7702 ~C14#H_RAB_2011_0001~1~~C14_RAB_2010_0001
## 6415 ~C14#H_RAB_2011_0001~2~~C14_RAB_2010_0001
## 1529 ~C14#H_RAB_2011_0001~3~~C14_RAB_2010_0001
## 8908 ~C14#H_RAB_2011_0001~4~~C14_RAB_2010_0001
## 2718 ~C14#H_RAB_2011_0001~5~~C14_RAB_2010_0001
## 7703 ~C14#H_RAB_2011_0001~6~~C14_RAB_2010_0001
##      spike_weight---spike_weight plant_height---plant_height
## 7702                        2.98                        <NA>
## 6415                        2.73                        <NA>
## 1529                        1.82                        <NA>
## 8908                        3.78                        <NA>
## 2718                        4.01                        <NA>
## 7703                        3.58                        <NA>
##      spike_lenght---spike_length_F spike_lenght---spike_lenght
## 7702                          <NA>                        <NA>
## 6415                          <NA>                        <NA>
```

```
## 1529                          <NA>                          <NA>
## 8908                          <NA>                          <NA>
## 2718                          <NA>                          <NA>
## 7703                          <NA>                          <NA>
```

5 pdf compilation with LaTeX

`get.pdf` aggregates texts and outputs from `get.ggplot` and `get.table`. This is particularly useful in participatory research to give feedback to stackholders such as farmers.

`get.pdf` creates a `.tex` file that is compiled into a `.pdf` file. You need to install LaTeX and the following packages : `longtable`, `lscape`, `graphicx`, `pdfpages`, `float`, `hyperref` and `fancyhdr`. To download LaTeX, go to http://latex-project.org/ftp.html.

5.1 Philosophy of `get.pdf`

The idea of `get.pdf` is to create a list (`LaTeX_body` argument) with several elements that can be:

- `"titlepage"` : a list containing the following elements :
  - `"title"` : the title of the document. Nothing by default.
  - `"authors"` : the authors of the document. Nothing by default.
  - `"email"` : the corresponding email. Nothing by default.
  - `"date"` : the date of the document. Nothing by default.
- `"tableofcontents"` : if `TRUE`, a table of content is display.
- `"chapter"` : name of a chapter
- `"section"` : name of a section
- `"subsection"` : name of a subsection
- `"subsubsection"` : name of a subsubsection
- `"text"` : text to add
- `"includepdf"` : path of a .pdf to include
- `"input"` : path of a .tex to insert
- `"table"` : a list containing the following elements :
  - `"content"` : output from `get.table`
  - `"caption"` : caption of the table
  - `"landscape"` : If `TRUE`, the table is in landscape. `FALSE` by default.
  - `"display.rownames"` : If `TRUE`, display the row names of the table. `FALSE` by default.
- `"figure"` : a list containing the following elements :
  - `"content"` : output from `get.ggplot`
  - `"caption"` : caption of the figure
  - `"layout"` : the layout of the plots. It is a matrix under the form `layout = matrix(c(1:4), ncol = 2, nrow = 2)`. It is `layout = matrix(1, ncol = 1, nrow= 1)` by default.
  - `"width"` : width of the figure in textwidth unit. 1 means as width as the text. It is 1 by default.
  - `"landscape"` : If `TRUE`, the figure is in landscape. `FALSE` by default.

The elements of this list will be in the same order in the final pdf.

You do not have to know LaTeX. `get.pdf` uses it for you. Nevertheless, if you use LaTeX you'll be able to customize the `.pdf` document by creating the tex structure of the document (`LaTeX_head` argument). It is also possible to add specific tex files (`input` element of `LaTeX_body` argument). If you use LaTeX macro, do not forget to use the escape mode (Table 6),

macro	description
`\\textbf{blod}`	**blod**
`\\textit{italique}`	*italique*
`\\underline{underline}`	underline
`\\texttt{type writer}`	`type writer`

Table 6: Basic maros in LaTeX to use in `get.pdf` with escape mode.

Tutorials on LaTeX can be found here: https://en.wikibooks.org/wiki/LaTeX.

See appendix C for some examples.

5.2 Examples

5.2.1 First Information

First, set the directory where you want the pdf to be store:

```
dir = "/home/pierre/"
```

Then, choose a name for you pdf (as well as folder that contain all the raw information to create the pdf)

```
form.name = "my_first_pdf"
```

5.2.2 LaTeX body

Then, create the list with the different elements. The names of plots and tables are coming from sections 3 and 4.

```
OUT = NULL

###################################
# 1. Title page
###################################
out = list(
        "titlepage" = list(
                "title" = "Example of \\texttt{shinemas2R::get.pdf}",
                "authors" = "P. Riviere",
                "date" = "\\today",
                "email" = "pierre@semencespaysannes.org")
        ); OUT = c(OUT, out)


###################################
# 2. Table of contents
###################################
out = list("tableofcontents" = TRUE); OUT = c(OUT, out)


###################################
# 3. Network relations between seed-lots
###################################
out = list(
        "chapter" = "Network relations between seed-lots"
        ); OUT = c(OUT, out)

out = list(
        "section" = "Network for Rouge du Roc"
        ); OUT = c(OUT, out)

out = list(
        "text" = "The following figure display the network for Rouge du Roc"
```

```
        ); OUT = c(OUT, out)

out = list(
        "figure" = list(
                "caption" = "Network for Rouge du Roc",
                "content" = p_net_RdR,
                "width" = 1)
        ); OUT = c(OUT, out)

out = list(
        "section" = "Seed-lots harvested for Rouge du Roc after a reproduction"
        ); OUT = c(OUT, out)

out = list(
        "figure" = list(
                "caption" = "Seed-lots harvested for each person by year for Rouge du Roc",
                "content" = p3_slh,
                "width" = 1)
        ); OUT = c(OUT, out)

out = list(
        "figure" = list(
                "caption" = "Repartition of Rouge du Roc harvested
since the beginning of the project.",
                "content" = m9_slh,
                "width" = 1)
        ); OUT = c(OUT, out)


##################################
# 4. Data linked to the relations between seed-lots
##################################
out = list(
        "chapter" = "Data linked to the relations between seed-lots"
        ); OUT = c(OUT, out)


out = list(
        "section" = "Data related to sowing practices"
        ); OUT = c(OUT, out)

out = list(
        "table" = list(
                "caption" = "Sowing practices for seed-lots at RAB",
                "content" = tab_class_4,
                "landscape" = FALSE
                )
        ); OUT = c(OUT, out)


out = list(
        "section" = "Data on selection differentia"
        ); OUT = c(OUT, out)

out = list(
        "figure" = list(
```

```
                "caption" = paste("Selection differentia for spike weight on RAB farm."),
                "content" = p_S_sw_bar,
                "width" = 1)
        ); OUT = c(OUT, out)

# Here, all the plots in the list are displayed:
# Set "layout" argument is useful as there are more than one plot
out = list(
        "figure" = list(
                "caption" = paste("Selection differentia for spike weight on RAB farm."),
                "content" = p_S_sw_box_all,
                "layout" = matrix(c(1,2), ncol = 1),
                "width" = 1)
        ); OUT = c(OUT, out)


out = list(
        "table" = list(
                "caption" = "Mean, standard deviation and coefficient of variation
for differentia to selection",
                "content" = tab_S_msc,
                "landscape" = TRUE
                )
        ); OUT = c(OUT, out)

out = list(
        "section" = "Data on response to selection"
        ); OUT = c(OUT, out)

out = list(
        "figure" = list(
                "caption" = paste("Selection differentia for spike weight on RAB farm."),
                "content" = p_SR_sw_bar,
                "width" = .5)
        ); OUT = c(OUT, out)
```

5.2.3 Compile the pdf

It is possible to custom the color of the rows of tables with `color1` (color of the head of the table) or `color2` (color of the row, the color of the rows is an alternation of white and `color2`). `compile.tex = TRUE` compiles the pdf.

```
get.pdf(
        dir = dir,
        form.name = form.name,
        LaTeX_head = NULL,
        LaTeX_body = OUT,
        compile.tex = TRUE
        )

## 1.  Get LaTeX head ...
## 2.  Get LaTeX body ...
## 3.  Compilation of .tex ...
## /home/pierre/my_first_pdf.pdf has been compiled.
```

6 Common use rights

The use of a tool such as a data base rise questions on collaborative research. The Réseau Semences Paysannes is a network that brings together a great diversity of collectives. Some collective are known as Community Seed Houses (CSH) (Réseau Semences Paysannes, 2014).

Within RSP, biodiversity is seen as a common. Therefore, common use rights are needed to manage this common, especilly on data management.

Common use rights related to data management within CSHs can be seen at two levels :

1. Internal organisation rules of the group and their relations with juridic, social, agronomic and economic environments:
   - Up to which level dematerialse the information ?
   - Which data to store ?
   - Which access to data ?
2. Biopiracy risk:
   - Which data to store ?
   - Which access to data ?
   - Link of CSHs with research : which protections of CSHs'work, ressources and knowledge ?

More information on this topic can be found in Réseau Semences Paysannes (2015).

To cite shinemas2R

To cite this package and or this vignette:

```
citation("shinemas2R")

##
## To cite package 'shinemas2R' in publications use:
##
##   Pierre Riviere (2015). shinemas2R: An R package to
##   visualize outputs from the data base Seed History and
##   Network Management System (SHiNeMaS). R package version
##   0.9. http://github.com/priviere/shinemas2R
##
## A BibTeX entry for LaTeX users is
##
##   @Manual{,
##     title = {shinemas2R: An R package to visualize outputs from the data base Seed History an
##     author = {Pierre Riviere},
##     year = {2015},
##     note = {R package version 0.9},
##     url = {http://github.com/priviere/shinemas2R},
##   }
```

Aknowledgement

This worked has been first funded by the European Community's Seventh Framework Programme (FP7/9 2007–2013) under the grant agreement n245058-Solibam (Strategies for Organic and Low-input Integrated Breeding and Management). It has been completed by funding from European Union's Horizon 2020 research and innovation programme under grant agreement No 633571 (DIVERSIFOOD project) and Fondation de France.

Thanks to Hadley Wickham for his web site http://r-pkgs.had.co.nz/ that help us a lot in the creation of this package.

Thanks to Gaelle Van Frank for her comments and remarks improving this vignette and the code of the package.

References

Y. De Oliveira, L. Burlot, M. Lefebvre, D. Madi, P. Rivière, and M. Thomas. SHiNeMaS : a database dedicated to seed lots genealogy, phenotyping and cultural practices. Poster at Jobim, Clermont-Ferrand, 2015.

M. Nei. Analysis of gene diversity in subdivided populations. *Proceedings of the National Academy of Sciences*, 70:3321–3323, 1973.

Réseau Semences Paysannes. *Les maisons des semences paysannes : Regards sur la gestion collective de la biodiversité cultivée en France.* Réseau Semences Paysannes, 2014.

Réseau Semences Paysannes. *Éléments de réflexion sur la gestion des données dans les Maisons des Semences Paysannes.* Réseau Semences Paysannes, 2015.

P. Rivière. *Méthodologie de la sélection décentralisée et participative: un exemple sur le blé tendre.* PhD thesis, Université Paris Sud, 2014.

M. Thomas. *Gestion dynamique à la ferme de l'agrobiodiversité : relation entre la structure des populations de blé tendre et les pratiques humaines.* PhD thesis, Université Paris Diderot, 2011.

A Install SHiNeMaS on localhost

A.1 Install PostgreSQL and SHiNeMaS

With Linux,

```
pierre@RSP:~$ sudo apt-get install postgresql
pierre@RSP:~$ sudo -i -u postgres
postgres@RSP:~$ psql
postgres=# ALTER USER postgres with password 'postgres';
postgres=# create user pierre;
postgres=# ALTER ROLE pierre WITH CREATEDB;
postgres=# ALTER USER pierre SUPERUSER;
postgres=# create database shinemas_tuto owner pierre;
postgres=# alter user pierre with encrypted password 'pierre';
postgres=# GRANT ALL PRIVILEGES ON DATABASE shinemas_tuto TO pierre;
```

Once you've done this, download the `shinemas_tuto.sql` here: TODO!, put in your `/home` for example, and install it on postgresql.

Note than `shinemas_tuto.sql` must be accesible in writing and reading.

```
pierre@RSP:~$ psql -h localhost -d shinemas_tuto -U pierre -f /home/shinemas_tuto.sql
```

You can look at it:

```
sudo -i -u postgres
postgres@RSP:~$ psql -l
```

More information on posgresql can be found here: http://doc.ubuntu-fr.org/postgresql

A.2 Set up the information to connect to SHiNeMaS with `get.data`

```
info_db = list(
        db_user = "pierre",
        db_host = "127.0.0.1", # localhost
        db_name = "shinemas_tuto",
        db_password = "pierre"
        )
```

`db_info` is use in `get.data` funtion.

B Therory regarding selection differentia, response to selection and heritability

to do !!!

C `get.pdf` examples

C.1 `LaTeX_head` examples

to do !

C.2 `.tex` examples for `input`

to do !