Index Selection

Sweetpotato breeders Meeting, Malawi

Raúl Eyzaguirre

International Potato Center
CIP

June 24, 2014

Outline

1. Two selection indices
2. Computations
3. Example
4. Exercise
5. Conclusions

Outline

1. Two selection indices
2. Computations
3. Example
4. Exercise
5. Conclusions

Why to use a selection index?

- In breeding usually several traits have to be improved simultaneously.
- Very often intuitive procedures are used.
- Intuition usually fails when several correlated traits are involved:
  - Positive correlations: Improving one trait improves the other.
  - Negative correlations: Improving one trait can diminish the other.
- And do not forget the variability:
  - High variability: A lot of room to improve.
  - Low variability: No so much room to improve.

Two selection indices

- Elston, R. C. (1963). A weight-free index for the purpose of ranking or selection with respect to several traits at a time. *Biometrics*. 19(1): 85-97.
- Pesek, J. and R.J. Baker.(1969). Desired improvement in relation to selection indices. *Can. J. Plant. Sci.* 9:803-804.

Elston

The Elston index is a weight free index. Given p traits, for each genotype the Elston index is computed as

$$I_E = \prod_{i=1}^{p} (x_i - k_i)$$

where x_i is the value of the genotype for trait i and k_i is some lower bound. Two options for k:

- $k_i = \min x_i$
- $k_i = \frac{n \min x_i - \max x_i}{n-1}$.

Pesek-Baker

The Pesek Baker is an index where weights are given in the form of desired gains. Given p traits, this index is defined by

$$I_{PB} = \sum_{i=1}^{p} b_i x_i$$

where x_i is the value of the genotype for trait i. The coefficients, b_i , are computed from:

$$\mathbf{b} = \mathbf{V}^{-1}\mathbf{g}$$

with

- $\mathbf{b}$ the vector of index coefficients,
- $\mathbf{V}$ the genetic variance-covariance matrix, and
- $\mathbf{g}$ the vector of desired genetic gains to be specified by the breeder.

Outline

1 Two selection indices

2 Computations

3 Example

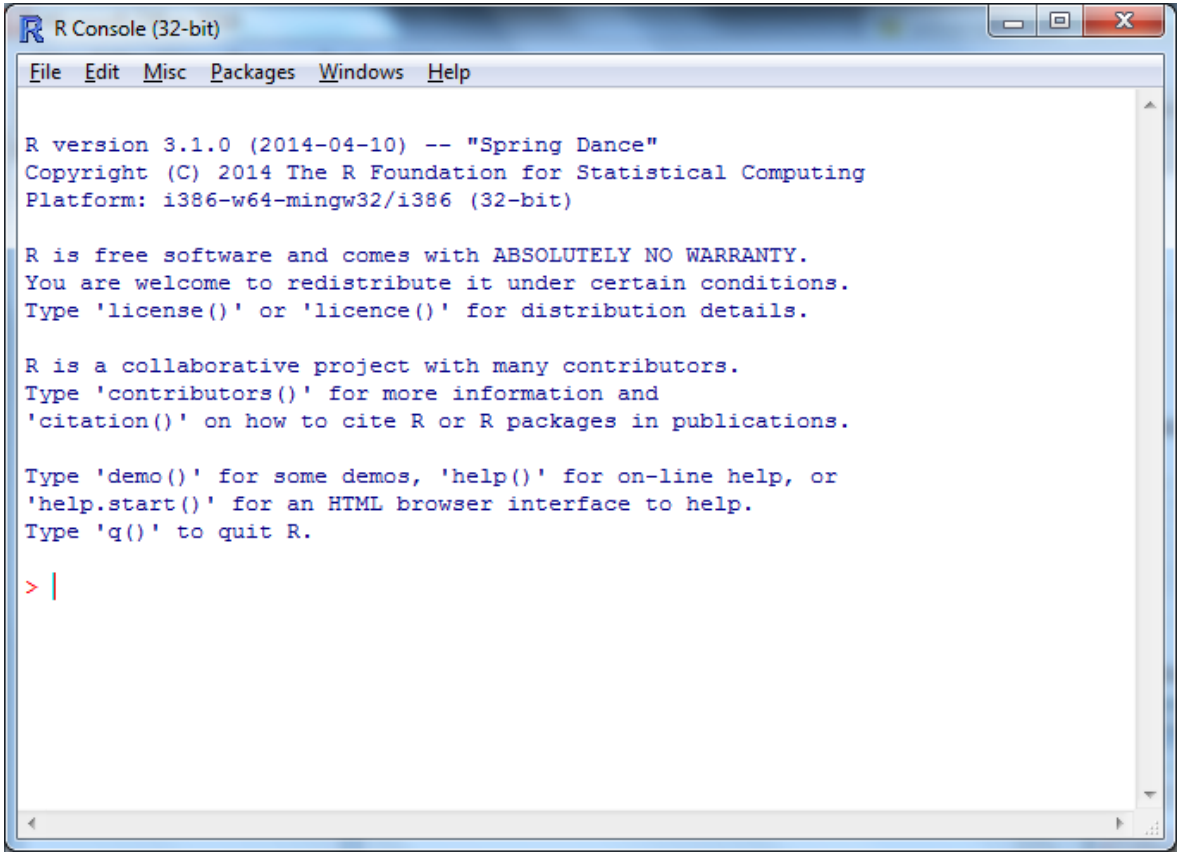
4 Exercise

5 Conclusions

Open R

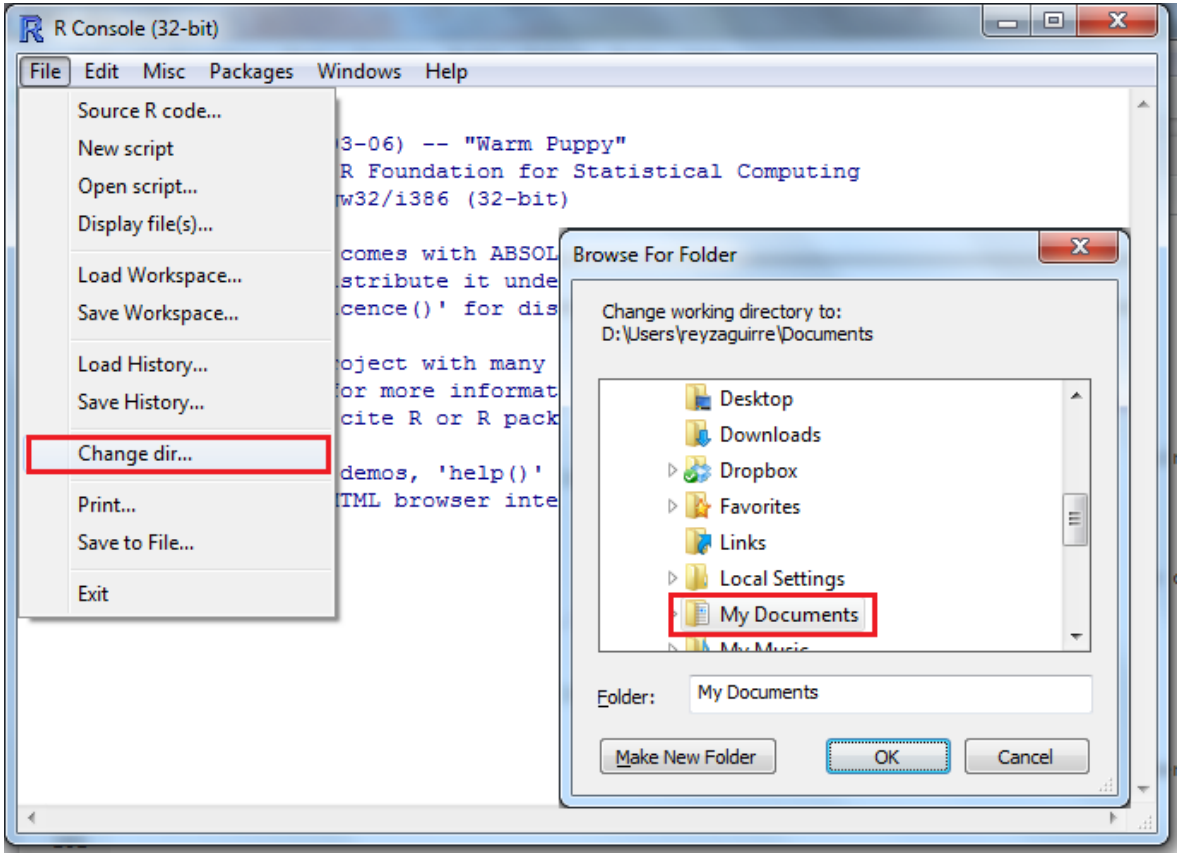
We will use R for the example. If you have Clone Selector, you have R.

Open R.



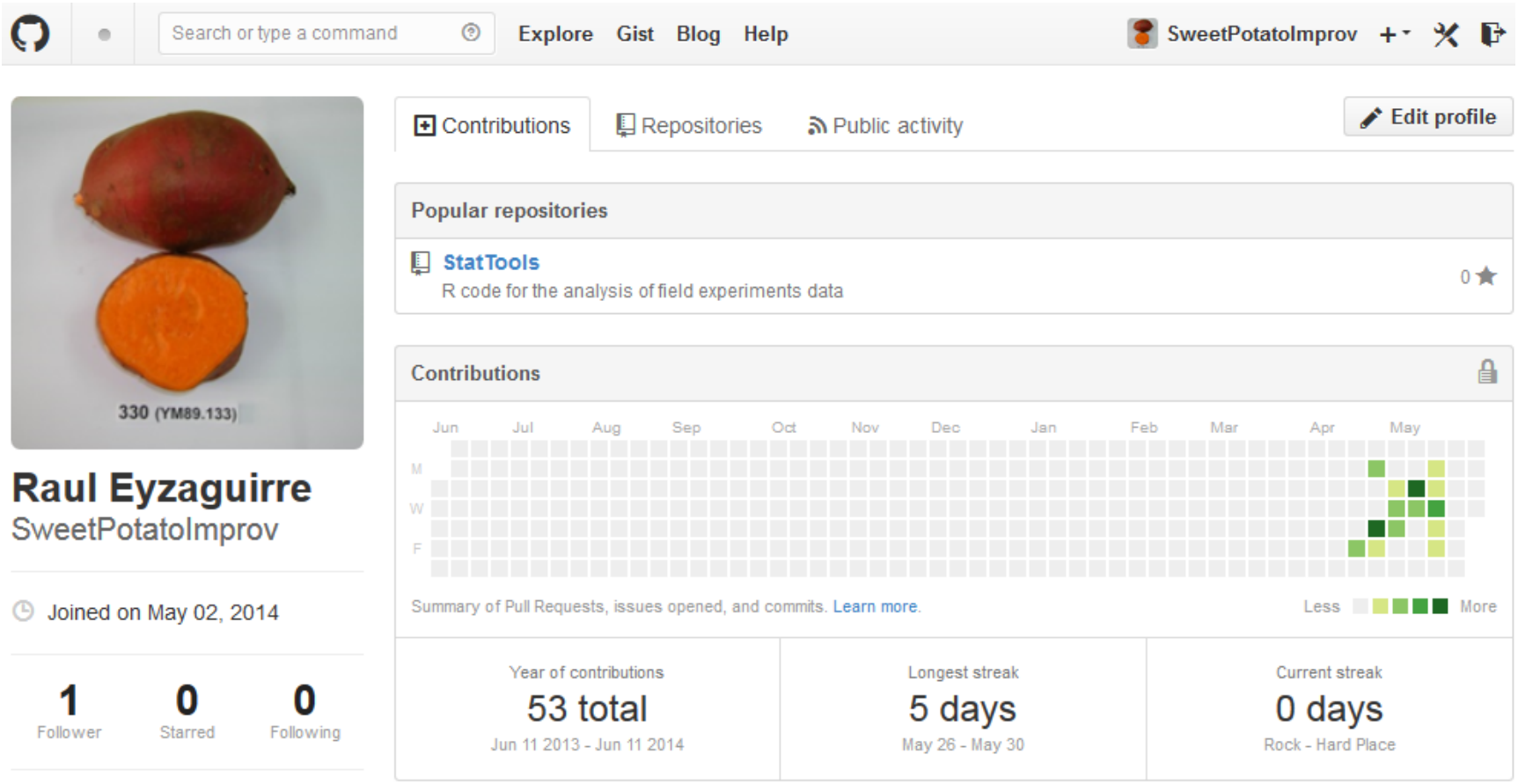
Define your working directory

Define your working directory, *My Documents* by default.



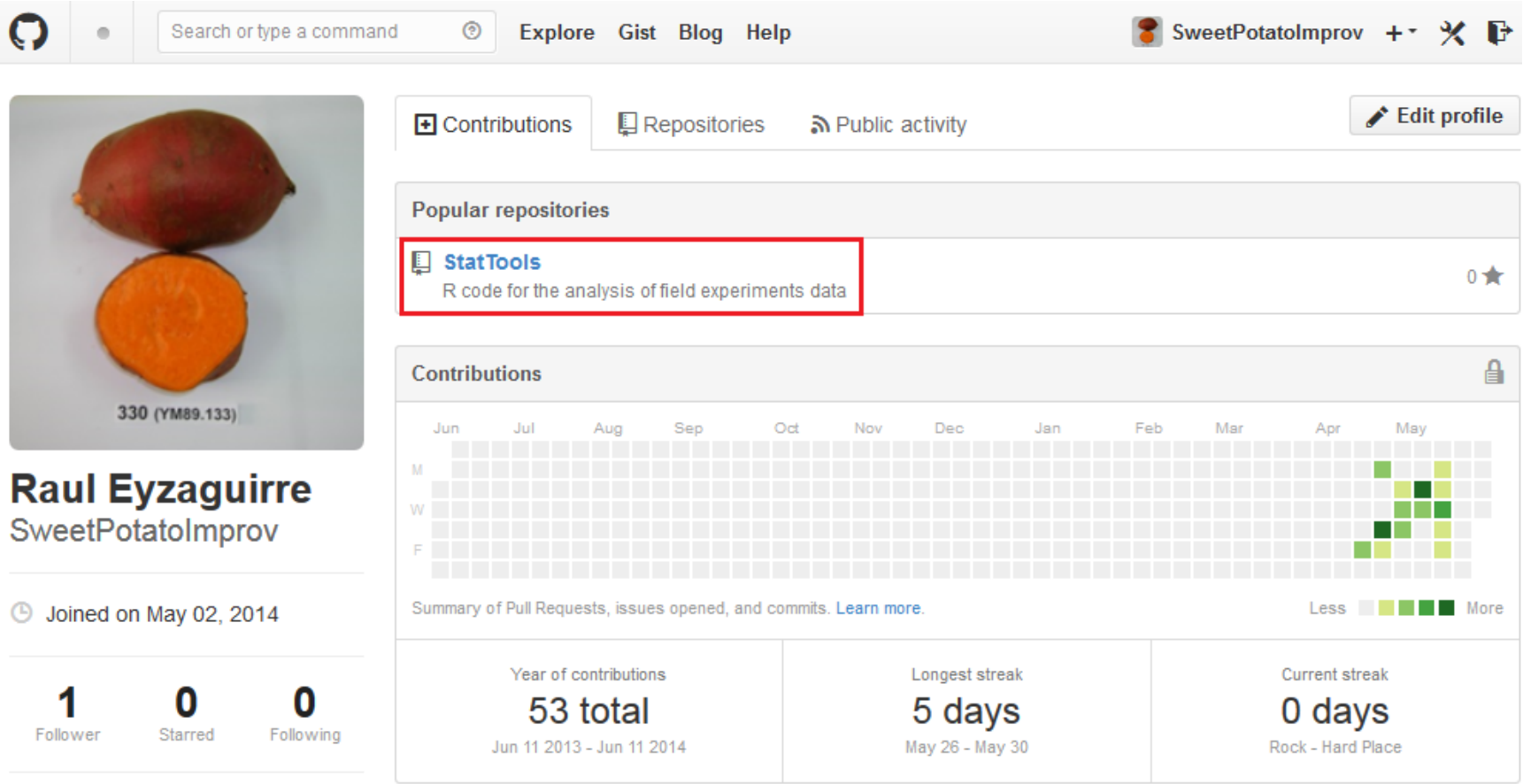
GitHub account

GitHub is a good place to put code.
`https://github.com/SweetPotatoImprov`



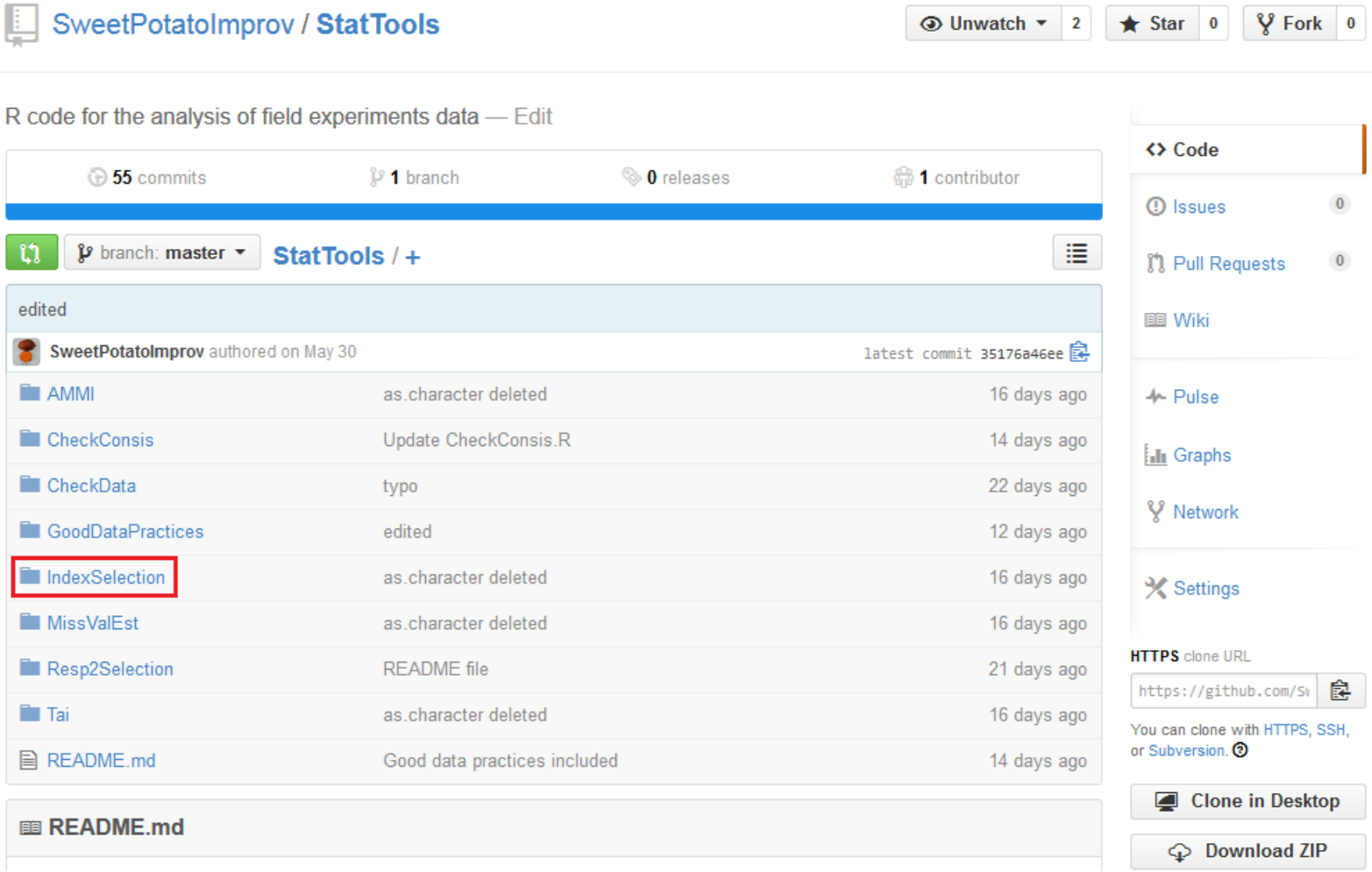
GitHub repo

There is a GitHub repository for the R functions and some other documents: `https://github.com/SweetPotatoImprov/StatTools`



GitHub repo

Go into the *IndexSelection* folder:



Two selection indices

SweetPotatoImprov / StatTools

Unwatch 2 | Star 0 | Fork 0

branch: master | StatTools / IndexSelection / +

presentation

SweetPotatoImprov authored 5 minutes ago — latest commit 1471d5adce

..		
Elston	space	4 days ago
PesekBaker	required packages corrected	2 days ago
Presentation	presentation	5 minutes ago
README.md	new folder	a month ago

README.md

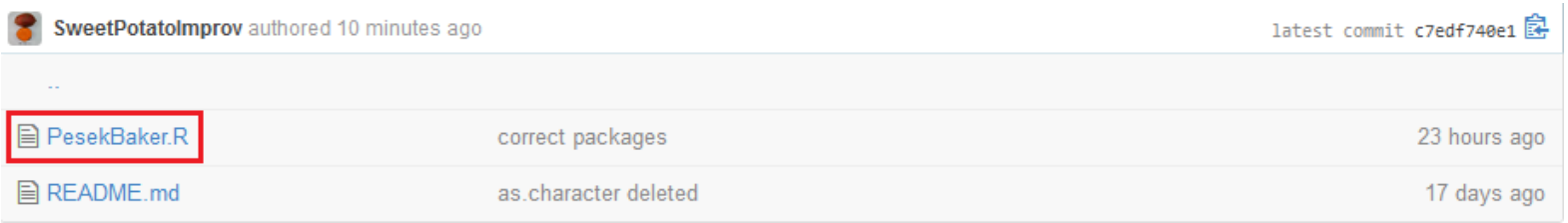
IndexSelection

Proposed Selection Indices for Applied Breeding Programs.

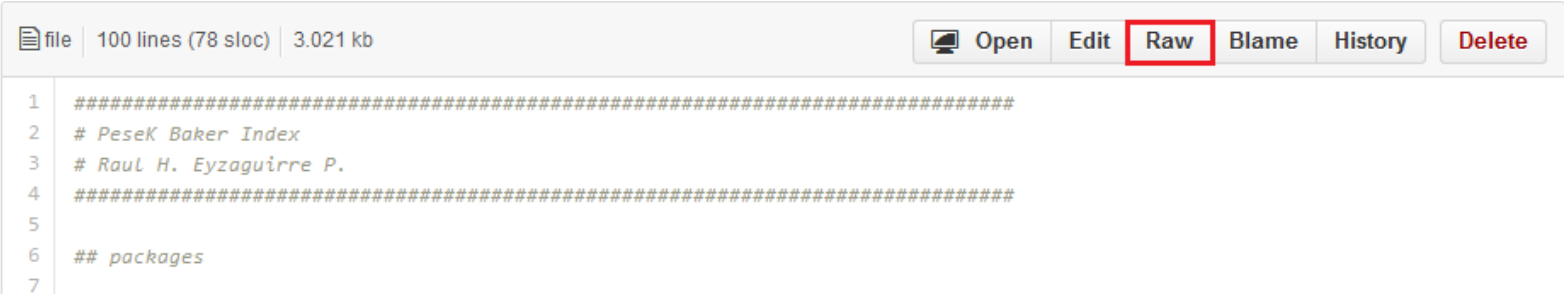
Download the files

Download the R functions (Elston.R and PesekBaker.R) into your working directory.

- Click on the function.

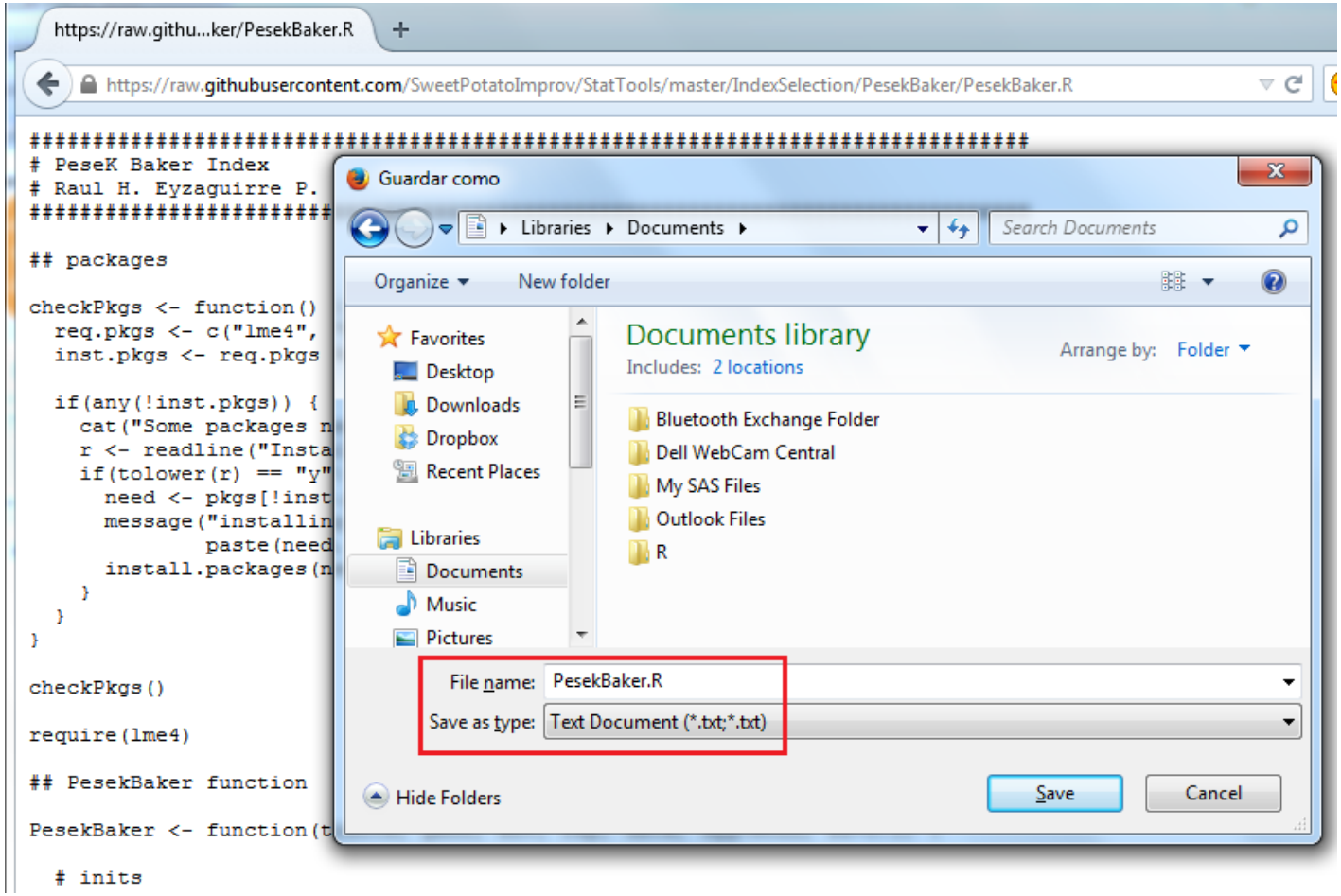


- Click on Raw.



Download the files

- Right click on the mouse and *Save as . . .* or go to *Save web page as . . .* Beware of browsers, some are not so smart. Make sure to save the file with the real name (Elston.R or PesekBaker.R)



Load the functions directly from GitHub

You can also load the R functions directly from GitHub into R. First define the URL of the files:

```
> urlfile1 <- "https://raw.githubusercontent.com/SweetPotatoImprov/StatTools/
+              master/IndexSelection/Elston/Elston.R"
> urlfile2 <- "https://raw.githubusercontent.com/SweetPotatoImprov/StatTools/
+              master/IndexSelection/PesekBaker/PesekBaker.R"
```

This works well if you use RStudio:

```
> source(urlfile1)
> source(urlfile2)
```

This works well directly from R:

```
> require(RCurl)
> file1 <- getURL(urlfile1, ssl.verifypeer = FALSE)
> file2 <- getURL(urlfile2, ssl.verifypeer = FALSE)
> eval(parse(text = file1))
> eval(parse(text = file2))
```

Outline

1. Two selection indices
2. Computations
3. Example
4. Exercise
5. Conclusions

Data

Data is in the file SI_example.csv.

This file contains data for:

- 2 locations: La Molina and Satipo.
- 2 replications in each location.
- 1041 genotypes.
- 25 traits.
- 3 plants per plot.

Small data

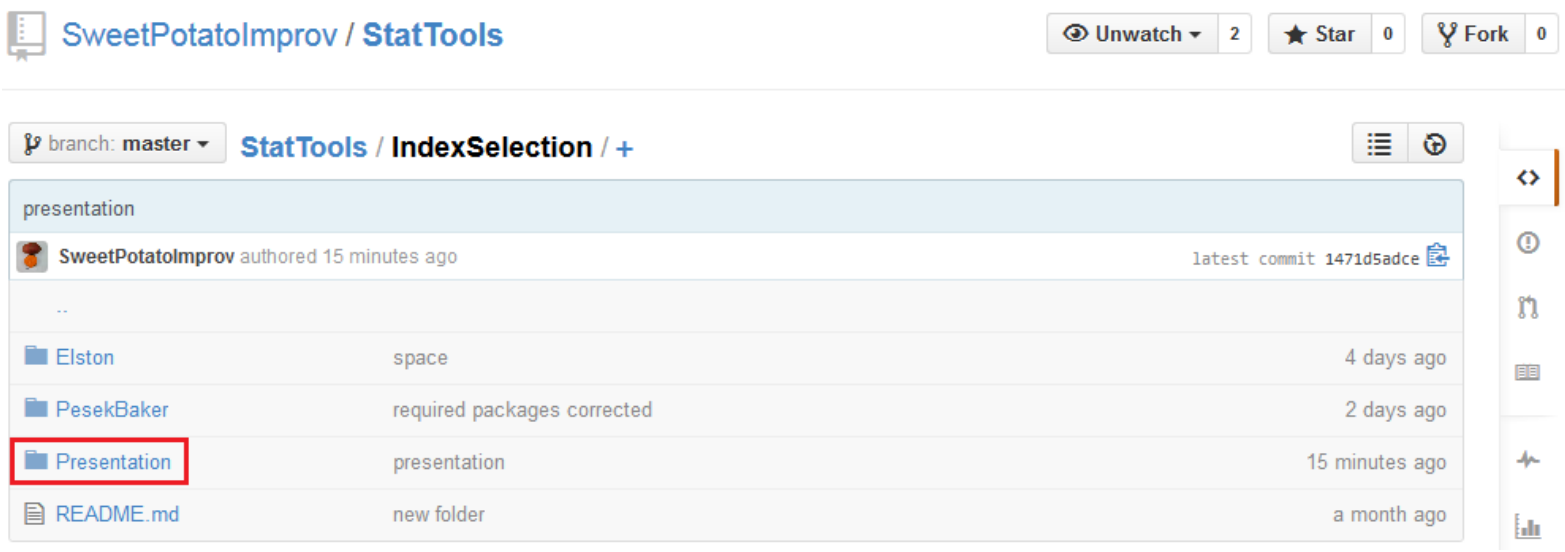
Let us start with a small subset in the file SI_example_small.csv.

This file contains data for:

- 2 locations: La Molina and Satipo.
- 2 replications in each location.
- 8 genotypes.
- 5 traits: RYTHA, BC, DM, STAR, NOCR.
- 3 plants per plot.

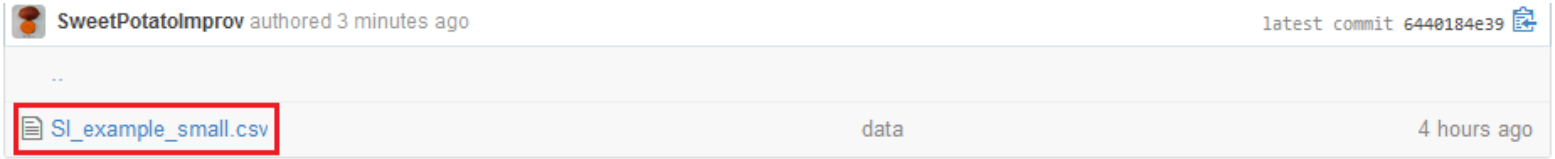
Download the data

Data are on GitHub, so you can proceed in the same way as we did to download the functions. Go into the folder *Presentation*.

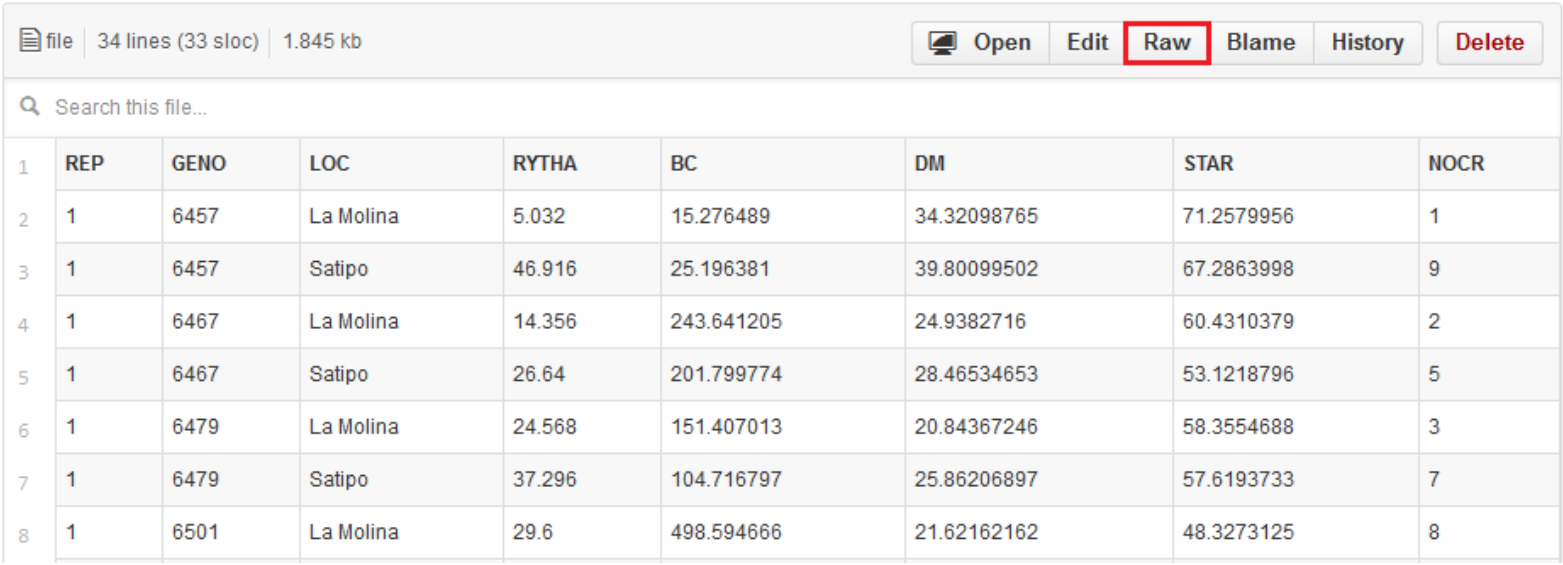


Download the data

- Click on the data file.



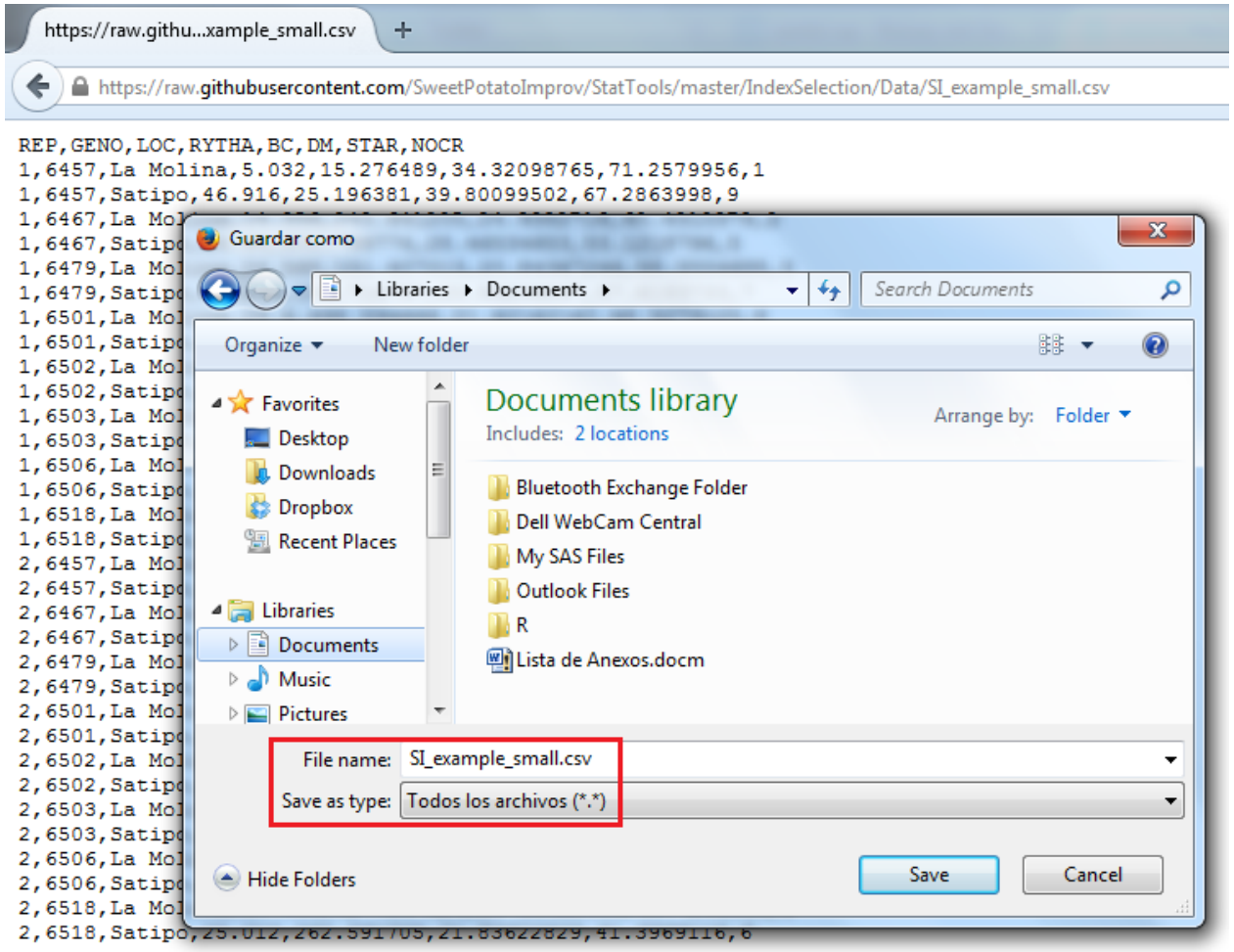
- Click on Raw.



	REP	GENO	LOC	RYTHA	BC	DM	STAR	NOCR
2	1	6457	La Molina	5.032	15.276489	34.32098765	71.2579956	1
3	1	6457	Satipo	46.916	25.196381	39.80099502	67.2863998	9
4	1	6467	La Molina	14.356	243.641205	24.9382716	60.4310379	2
5	1	6467	Satipo	26.64	201.799774	28.46534653	53.1218796	5
6	1	6479	La Molina	24.568	151.407013	20.84367246	58.3554688	3
7	1	6479	Satipo	37.296	104.716797	25.86206897	57.6193733	7
8	1	6501	La Molina	29.6	498.594666	21.62162162	48.3273125	8

Download the data

- Right click on the mouse and *Save as . . .* or go to *Save web page as . . .* Make sure to save the file with the real name and format.



Load the data directly from GitHub

You can also load the data directly from GitHub into R. First define the URL of the file:

```
> urlfile <- "https://raw.githubusercontent.com/SweetPotatoImprov/StatTools/
+             master/IndexSelection/Presentation/SI_example_small.csv"
```

This works well if you use RStudio:

```
> mydata <- read.csv(urlfile)
```

This works well directly from R:

```
> require(RCurl)
> mydata <- getURL(urlfile, ssl.verifypeer = FALSE)
> mydata <- read.csv(textConnection(mydata))
```

Load the data and functions

If you have the files (SI_example_small.csv, Elston.R, PesekBaker.R) in your working directory, proceed like this:

```
> mydata <- read.csv("SI_example_small.csv")
> mydata$REP <- factor(mydata$REP)
> mydata$GENO <- factor(mydata$GENO)
> str(mydata)
```

```
'data.frame':        32 obs. of  8 variables:
 $ REP  : Factor w/ 2 levels "1","2": 1 1 1 1 1 1 1 1 1 1 ...
 $ GENO : Factor w/ 8 levels "6457","6467",..: 1 1 2 2 3 3 4 4 5 5 ...
 $ LOC  : Factor w/ 2 levels "La Molina","Satipo": 1 2 1 2 1 2 1 2 1 2 ...
 $ RYTHA: num  5.03 46.92 14.36 26.64 24.57 ...
 $ BC   : num  15.3 25.2 243.6 201.8 151.4 ...
 $ DM   : num  34.3 39.8 24.9 28.5 20.8 ...
 $ STAR : num  71.3 67.3 60.4 53.1 58.4 ...
 $ NOCR : int  1 9 2 5 3 7 8 11 6 7 ...
```

```
> source('Elston.R')
> source('PesekBaker.R')
```

Elston index - Arguments

The arguments for this function are:

```
Elston(traits, geno, data, lb = 1)
```

- `traits` is a list of traits,
- `geno` is the index for genotypes,
- `data` is the data frame containing the data,
- `lb` is the lower bound, 1 for $k_i = \min x_i$ and 2 for $k_i = \frac{n \min x_i - \max x_i}{n-1}$, 1 by default.

Elston index - Computation

```
> Elston(c('RYTHA', 'BC', 'DM', 'STAR', 'NOCR'), 'GENO', mydata)

$Elston.Index
    6457     6467     6479     6501     6502     6503     6506     6518
 0.09313  0.00000  1.18126  4.82440 33.84316  0.28160  0.00000  0.00000

$Sorted.Elston.Index
    6502     6501     6479     6503     6457     6467     6506     6518
33.84316  4.82440  1.18126  0.28160  0.09313  0.00000  0.00000  0.00000
```

Remember that by default it uses $k_i = \min x_i$ for the index

$$I_E = \prod_{i=1}^{p} (x_i - k_i)$$

Elston index - Computation

We can use the second lower bound, $k_i = \frac{n \min x_i - \max x_i}{n-1}$, writing `lb=2`.

```
> Elston(c('RYTHA', 'BC', 'DM', 'STAR', 'NOCR'), 'GENO', lb=2, mydata)

$Elston.Index
  6457   6467   6479   6501   6502   6503   6506   6518
 4.862  0.710  6.790 19.822 84.154 11.965 18.425  3.439

$Sorted.Elston.Index
  6502   6501   6506   6503   6479   6457   6518   6467
84.154 19.822 18.425 11.965  6.790  4.862  3.439  0.710
```

Pesek Baker index

Let us compute the Pesek Baker index using these traits (means between brackets):

- RYTHA (31.3 tons/ha)
- BC (188 ppm)
- DM (30.05%)
- STAR (60.3%)
- NOCR (6.66)

Which are your desired genetic gains?

My guess: 5, 100, 2, 2, 5.

Pesek Baker index - Arguments

The arguments for this function are:

```
PesekBaker(traits, geno, env, rep, data, dgg = NULL, units =
"sdu", sf = 0.1)
```

- `traits` is a list of traits,
- `geno` is the index for genotypes,
- `env` is the index for environments,
- `rep` is the index for replications,
- `data` is the data frame containing the data,
- `dgg` is the vector of desired genetic gains, one standard deviation by default,
- `units` are the units for dgg, "actual" for actual units and "sdu" for standard deviations, "sdu" by default, and
- `sf` is the selected fraction, 0.1 by default.

Pesek Baker index - Output

The PesekBaker function returns the following elements:

- `$Desired.Genetic.Gains`: The desired genetic gains in actual units.
- `$Standard.Deviations`: The estimated standard deviations.
- `$Genetic.Variance`: The estimated genetic variances.
- `$Correlation.Matrix`: The estimated correlation matrix.
- `$Index.Coefficients`: The index coefficients.
- `$Response.to.Selection`: The response to selection.
- `$Std.Response.to.Selection`: The standardized response to selection.
- `$Pesek.Baker.Index`: The Pesek-Baker index value.
- `$Sorted.Pesek.Baker`: The Pesek-Baker index value sorted in descending order.

Pesek Baker index - Computation

```
> output <- PesekBaker(c('RYTHA', 'BC', 'DM', 'STAR', 'NOCR'), 'GENO', 'LOC', 'REP',
+                      mydata, c(5, 100, 2, 2, 5), "actual")
> output$Index.Coefficients

          coef
RYTHA -0.29365
BC     0.01197
DM     0.21252
STAR   0.01919
NOCR   2.24846
```

Why is the RYTHA coefficient negative?

- It does not mean that we are going to select to diminish root yield.
- We must see the correlations.
- We must see the variances.

See the correlations

We want to improve all the traits, but some have a negative correlation.

There are also traits with high positive correlations.

It is difficult to have this kind of correlation structures on mind.

```
> output$Correlation.Matrix

        RYTHA       BC        DM     STAR     NOCR
RYTHA  1.0000  0.26732 -0.231076 -0.15266 0.615558
BC     0.2673  1.00000 -0.775152 -0.85198 0.091521
DM    -0.2311 -0.77515  1.000000  0.89555 0.004778
STAR  -0.1527 -0.85198  0.895548  1.00000 0.077349
NOCR   0.6156  0.09152  0.004778  0.07735 1.000000
```

See the standard deviations

Maybe we want to improve too much a trait with a very low variability.

If a trait has high variability, then there is a lot of room to improve.

Let us remember my desired gains: 5, 100, 2, 2, 5.

```
> output$Standard.Deviations
[1]   5.068 151.583   5.859   8.231   1.663
> output$Response.to.Selection
[1]  1.765 35.298  0.706  0.706  1.765
> output$Std.Response.to.Selection
[1] 0.34827 0.23286 0.12049 0.08577 1.06140
```

Compute the index - second try

I will relax a little bit my ambition about NOCR.

```
> output <- PesekBaker(c('RYTHA', 'BC', 'DM', 'STAR', 'NOCR'), 'GENO', 'LOC', 'REP',
+                      mydata, c(5, 100, 2, 2, 2.5), "actual")
> output$Index.Coefficients
```

```
         coef
RYTHA 0.01853
BC    0.01669
DM    0.20259
STAR  0.15350
NOCR  0.66803
```

```
> output$Response.to.Selection
```

```
[1]  3.605 72.102  1.442  1.442  1.803
```

```
> output$Std.Response.to.Selection
```

```
[1] 0.7114 0.4757 0.2461 0.1752 1.0841
```

Using the defaults

By default the function gives weights so that the desired genetic gains are one standard deviation for each trait. Same relative weight for each trait.

```
> output <- PesekBaker(c('RYTHA', 'BC', 'DM', 'STAR', 'NOCR'), 'GENO', 'LOC', 'REP',
+                      mydata)
> output$Index.Coefficients

          coef
RYTHA  0.12673
BC     0.04457
DM     0.18665
STAR   0.71823
NOCR  -0.28636

> output$Response.to.Selection

[1]  1.8160 54.3195  2.0995  2.9495  0.5959

> output$Std.Response.to.Selection

[1] 0.3583 0.3583 0.3583 0.3583 0.3583
```

Outline

1. Two selection indices
2. Computations
3. Example
4. Exercise
5. Conclusions

Load the data

Load the complete data set.

```
> mydata <- read.csv("SI_example.csv")
> mydata$REP <- factor(mydata$REP)
> mydata$GENO <- factor(mydata$GENO)
> names(mydata)
 [1] "REP"   "GENO"  "LOC"   "PROT"  "BC"    "FE"    "ZN"    "CA"    "MG"
[10] "STAR"  "FRUC"  "GLUC"  "SUCR"  "MALT"  "DM"    "VW"    "NOPS"  "NOCR"
[19] "NONC"  "CRW"   "NCRW"  "RYTHA" "TRW"   "BIOM"  "HI"    "CI"    "FYTHA"
[28] "CYTHA"
```

Choose a set of traits to improve

Choose a set of traits to improve and run Pesek-Baker using defaults. See the correlations, standard deviations and response to selection.

```
> output <- PesekBaker(c('RYTHA', 'BC', 'DM', 'STAR', 'NOCR'), 'GENO', 'LOC', 'REP',
+                      mydata)
> output$Correlation.Matrix

         RYTHA       BC      DM    STAR    NOCR
RYTHA  1.00000  0.06816 -0.1677 -0.0496  0.8300
BC     0.06816  1.00000 -0.5271 -0.6703  0.1144
DM    -0.16766 -0.52711  1.0000  0.8277 -0.1710
STAR  -0.04960 -0.67032  0.8277  1.0000 -0.1026
NOCR   0.83003  0.11437 -0.1710 -0.1026  1.0000

> output$Standard.Deviations

[1]   5.626 145.822   4.631   7.170   1.394

> output$Response.to.Selection

[1]  3.1484 81.5980  2.5916  4.0124  0.7798

> output$Std.Response.to.Selection

[1] 0.5596 0.5596 0.5596 0.5596 0.5596
```

Get your index

With an eye on the correlations and standard deviations, define your own desired genetic gains. Run the index again.

See the response to selection. Not satisfied? Modify your desired genetic gains and run the index again.

Outline

1. Two selection indices
2. Computations
3. Example
4. Exercise
5. Conclusions

Conclusions for the Pesek Baker index

- Keep an eye in the correlations.
- Keep both eyes in the standard deviations.
- Specify your desired genetic gains in relative terms,
- and even better, specify your desired genetic gains in relative terms and in standard deviation units.
- If a trait is important give it a weight a little bit higher than one standard deviation (somewhere between 1 and 2). If a trait is not so important give it a weight lower than 1 standard deviation (between 0 and 1).
- Do not ever try to improve a trait in more than 2 standard deviations (or maybe 1.5 could be a better and more conservative upper bound).
- Play with different values for the desired genetic gains and compare the response to selection that you get with each group of values.