EvoMining

Introduction

Enzyme promiscuity on metabolic families, can be looked on enzymes that are over a divergent process.

Gen families expansions on genomes

Pangenomes

Expansions are located on pangenome, Tools to analyse pangenome BPgA

EvoMining

EvoMining looks expansions on prokariotic pangenome.
Biological idea.

EvoMining was available as a consult website with 230 members of the Actinobacteria phylum as genomic data base, 226 unclassified nBGCs, and not interchangable central database 339 queries for nine pathways, including amino acid biosynthesis, glycolysis, pentose phosphate pathway, and tricarboxylic acids cycle. [@cruz-morales_phylogenomic_2016] EvoMining was proved on Actinobacteria Arseno-lipids

Pangenome

The sequenced genome of an individull in some species is just a partil print of the species geneticll repertoire Individualls can gain and loss genes.
[@koonin_turbulent_2015] Pangenome is the total sequenced gene pool in a taxonomically related group. Supergenome all the possible extant genes. About 10 times genomes. there are open, closed pangenomes.Most genomes has a core a shell and a unique genes.
Gene history its a tree history

HGT doubles mutation rate on prokarites.
Maybe HGT is an selected feature, if is the case, so could be np production.
Some archaeas has open pangenome. [@halachev_calculating_2011]

HGT doubles mutation rate on prokarites. [@koonin_turbulent_2015] Maybe HGT is an selected feature, if is the case, so could be np production.
Some archaeas has open pangenome. [@halachev_calculating_2011] Shell trees converge to core trees [@narechania_random_2012]

EvoMining Implementation

EvoMining was expanded from a website (http://evodivmet.langebio.cinvestav.mx/EvoMining/index.html) with limited datasets to an easy to install distribution that allows flexiblibilty on genomic, central and natural product databases. Evomining user distribution was developed on perl on Ubuntu-14.04 but wraped on Docker. Docker is a software containerization platform that allows repetibilty regardless of the environment. Docker engine is avilable for Linux, Cloud, macOS 10.10.3 Yosemite or newer and even 64bit Windows 10.

Dependencies that were packaged at EvoMining docker app are Apache2, muscle3.8.31, newick-utils-1.6,quicktree, blast-2.2.30, Gblocks_Linux64_0.91b perl and from cpan CGI, SVG and Statistics::Basic modules.

Github defines itself as an online project hosting using Git. Its free for open source-code hosting and facilitates team work. Includes source-code browser, in-line editing, and wikis.

Dockerhub is an apps project hosting.

Dockerhub nselem

EvoMining code is open source and it is available at a github repository github/EvoMining

Github and Dockerhub can be coneccted by the use of repositories automatically built. Among the advantages of automated builds are that the DockerHub repository is automatically kept up-to-date with code changes on GitHub and that its Dockerfile is available to anyone with access to the Docker Hub repository. EvoMining is stored on a DockerHub automated build repository linked to github EvoMining repository so that code is always actualized.

To download EvoMining image from docker Hub once Docker engine is installed its necessary to run the following command at a terminal:
`docker pull nselem/newevomining`

To run EvoMining container
`docker run-i -t  -v /home/nelly/docker-evomining:/var/www/html -p 80:80 evomining /bin/bash`

To start evoMining app `perl startEvomining`
“ Detailed tutorial, EvoMining description, pipeline and user guide are available at a wiki on github at EvoMining wiki.

Other genomic apps were containerized to docker images during this work.
- *myRAST* docker- https://github.com/nselem/myrast
RAST is a bacterial and Archaeal genome annotator [@aziz_rast_2008, @overbeek_seed_2014 , @brettin_rasttk:_2015] This app allows myRAST functionality to upload
It allows EvoMining genome database annotation.
-*Orthocores* docker-https://github.com/nselem/orthocore
Helps to obtain genomic core paralog free and construct genomic trees
-*CORASON* docker-https://github.com/nselem/EvoDivMet/wiki
-PseudoCore github- <>
Genomic Core with a reference genome has the advantage of more genomes, but it is not paralog free

-RadiCal docker image
To detect core diferrences on a set of genomes
-BPGA to analize pangenome

EvoMining Dockerization was chosen to avoid future compatibilty problems, for example dependencies unavailabilty, or incompatibility between future versions of its software components. As much as reproducible research was a concerned while developing EvoMining app, reproducibilty is also important on data analysis, for that reason this document was writen using R-markdown and latex template from Reed College [@chesterismay_updated_2016]. While R-markdown allows to write and run R code and interpolate text paragraph to explain scripts and analysis.

EvoMining Databases

Evomining containerized app is a user-interactive genomic tool dedicated to the study of protein function[].

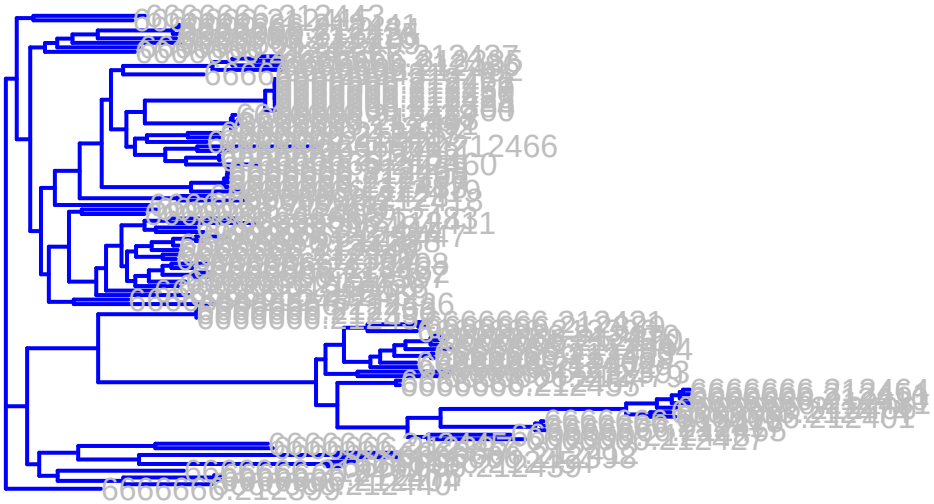
1. Genomes DB
2. Natural Products DB
3. Central Pathways DB

Archaea, *Actinobacteria*, *Cyanobacteria* were used as genome DB, MIBiG was used as Natural Product DB and different Central Pathways were used.

Genome DB

RAST annotation of genomes was done.

Phylogeny



To capture differences on genomes we sort them phylogenetically. Phylogenies can be constructed using different paradigms as Parsimony, Maximum Likelihood, and Bayesian inference. Short descriptions of the main phylogeny methods are included below.

Why is a tree useful {Book reference} why trees are useful for?
* Distance methods
* Parsimony * Maximum Likelihood * Mr bayes

General Trees
Actinobacteria Tree, ArchaeaTree, CyanobacteriaTree.

It's easy to create a list. It can be unordered like

To create a sublist, just indent the values a bit (at least four spaces or a tab). (Here's one case where indentation is key!)

1. Item 1
2. Item 2
3. Item 3
    - Item 3a
    - Item 3b

Central DB

We chose central pathways from [@barona-gomez_what_2012]
* BBH Best Bidirectional Hits with studied enzymes from Central Actinobacterial pathways were selected.

- By abundance
- By expansions on genomes

[largefiles,https://help.github.com/articles/installing-git-large-file-storage/]

Natural Products DB

Natural products was improved from previous version

AntisMASH optional DB

AntiSMASH is [@weber_antismash_2015,@medema_antismash:_2011]
Archaeas Results Archaea is a kingdom of recent discovery were not many natural products has been known. On Actinobacteria, evoMining has proved its value to find new kinds of natural products. The clue to this discovery was that Actinobacteria has genomic expanssions. Now Archaea has genomic expansions, even more has central pathways genomic expansions. Are this expansions derived from a genomic duplication? Has Archaea natural products detected by antismash, and if not, where are this NP's or may Archaea doesn't have NP's.

applying EvoMining to Archaea

Otras estrategias para los clusters Argon context Idea

Argon When you click the **Knit** button above a document will be generated that includes both content as well as the output of any embedded **R** code chunks within the document. You can embed an **R** code chunk like this (`cars` is a built-in **R** dataset):

```
summary(cars)
```

```
##      speed           dist
##  Min.   : 4.0   Min.   :  2.00
##  1st Qu.:12.0   1st Qu.: 26.00
##  Median :15.0   Median : 36.00
##  Mean   :15.4   Mean   : 42.98
##  3rd Qu.:19.0   3rd Qu.: 56.00
##  Max.   :25.0   Max.   :120.00
```

Inline code

If you'd like to put the results of your analysis directly into your discussion, add inline code like this:

> The `cos` of 2π is 1.

Another example would be the direct calculation of the standard deviation:

> The standard deviation of `speed` in `cars` is 5.2876444.

One last neat feature is the use of the `ifelse` conditional statement which can be used to output text depending on the result of an **R** calculation:

> The standard deviation is less than 6.

Note the use of `>` here, which signifies a quotation environment that will be indented.

As you see with `$2 \pi$` above, mathematics can be added by surrounding the mathematical text with dollar signs. More examples of this are in [Mathematics and Science] if you uncomment the code in [Math].

Recomendaciones de Luis

Para evoMining
Probar distintos métodos de filogenia y después hacer la coloración.
maximum likelihood, Protest phyml
Atracción de ramas largas.
raxml
trim all vs Gblocs (Tony Galvadon)

Comparar dos árboles
Para ver si la evolución de los genes concatenados ha sido simultánea
Robinson and foulds
Joe Felsestein
Phylip

2. dist tree
   quarter descomposition
   peter gogarten fendou Mao

Sets de experimentos.
Para el experimento de los streptomyces con ruta centrales el core, analizar el problema de dominios múltiples.
Dominios
Nan Song, Dannie durand
Después del blast

Para obtener
Pablo Vinuesa: Get Homologues

Burkhordelias y su toxina (Preguntar a Beto)
Cianobacterias y la ruta de fijación de nitrógeno.

Servidor Viernes a las 12:00

CORASON: Other genome Mining tools context-based

CORe Analysis of Syntenic Orthologs to prioritize Natural Product-Biosynthetic Gene Cluster

Genome fluidity on Bacteria is source of biosynthetic gene clusters (BGCs) abundance, in fact almost all bacterial genome sequenced contributes with new genes and gene clusters to the Bacterial Pangenome. As a consequence of gene diversity helped by sequence technology advances, researchers often have a large set of genomes that wish to analize in search of a particular gene cluster variation. Answering BGCs analysis needs CORASON allows users to find and visualice variations of a given gene cluster sorting them according to the conserved core phylogeny.

To find cluster variations, given a query protein sequence that belongs to a reference cluster, CORASON will search on a Bacterial genome database all gene clusters that contains orthologues of the query-protein and at least another sequence from the reference cluster. Orthologues on variation clusters are coloured within a gradient according to its identity percentage with the reference cluster sequences.

The cluster core attempts to identify a set of functions conserved on this particular biosynthetic BGC. The core genome on a taxonomical group is the set of coding sequences that are shared between all group members, this definition may be adapted to the cluster core by using a set of gene clusters instead of a set of genomes. A report about gene function will be provided whenever a cluster core exists also core sequences will be concatenated to construct a phylogenetic tree and sort variation clusters accordingly.

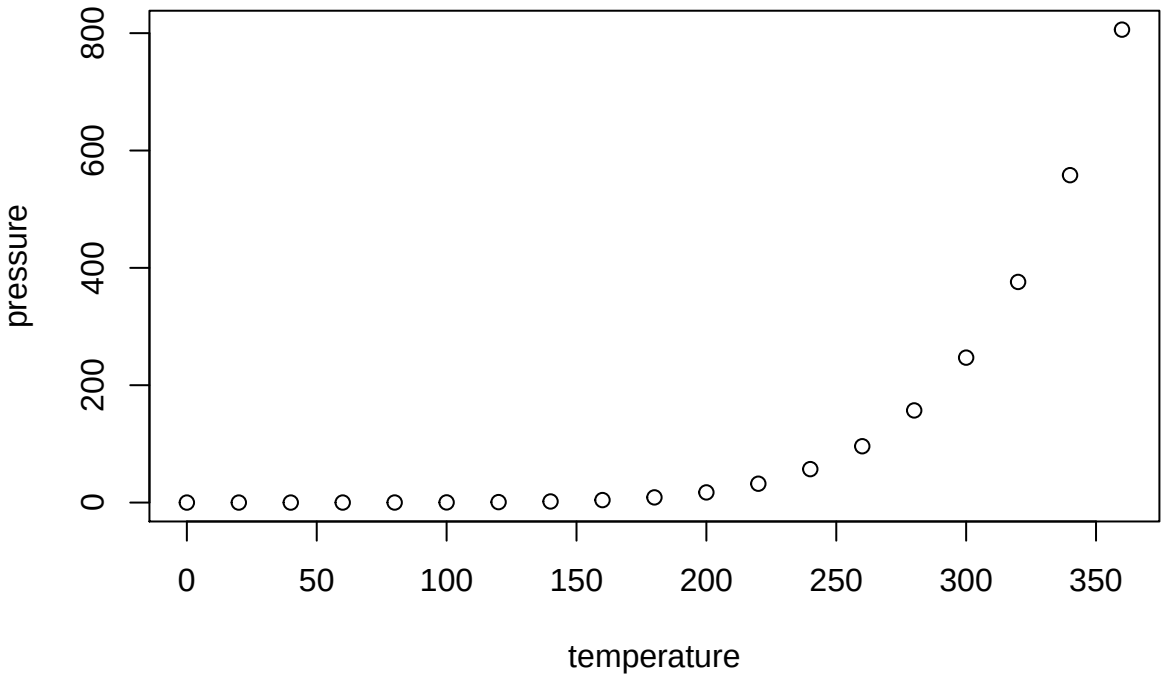
Functional annotations are provided by RAST annotation service due to that CORASON genomic databases must be RAST files. Any archaeal or bacterial genome can be RAST annotated either on the website or by command line using myrast server.

Finally, in order to provide an easy to install distribution CORASON was packaged on docker containerization platform. Software dependencies such as BLAST 2.2.30, muscle3.8.3, GBlocksLinux64_0.91b, quicktree, newick-utils-1.6, and CORASON code were wrapped together on CORASON docker container. Tutorial and software are available at nselem/github.

CORASON inputs are a genomic database, a reference cluster and an enzyme inside this cluster, outputs are newick trees, core functional report and a cluster variation SVG file. SVG format among being high quality scalable graphics, also allow to display metadata such as gene function and genome coordinates just by mouse over figures on a browser facilitating genomic analysis.

In conclusion CORASON is an easy to install comparative genomic visual tool on a customizable genome database that allows users to visualice variations of a reference gene cluster identifing its core functions and finally sorting variations according to their evolutionary history helping to prioritize clusters that may be involved on chemical novelty.

You can also embed plots. For example, here is a way to use the base **R** graphics package to produce a plot using the built-in `pressure` dataset:



Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the **R** code that generated the plot. There are plenty of other ways to add chunk options. More information is available at http://yihui.name/knitr/options/.

Another useful chunk option is the setting of `cache = TRUE` as you see here. If document rendering becomes time consuming due to long computations or plots that are expensive to generate you can use knitr caching to improve performance. Later in this file, you'll see a way to reference plots created in **R** or external figures.

Loading and exploring data

Included in this template is a file called `flights.csv`. This file includes a subset of the larger dataset of information about all flights that departed from Seattle and Portland in 2014. More information about this dataset and its **R** package is available at http://github.com/ismayc/pnwflights14. This subset includes only

Portland flights and only rows that were complete with no missing values. Merges were also done with the `airports` and `airlines` data sets in the `pnwflights14` package to get more descriptive airport and airline names.

We can load in this data set using the following command:

```
flights <- read.csv("data/flights.csv")
```

The data is now stored in the data frame called `flights` in **R**. To get a better feel for the variables included in this dataset we can use a variety of functions. Here we can see the dimensions (rows by columns) and also the names of the columns.

```
dim(flights)
```

```
## [1] 52808    16
```

```
names(flights)
```

```
##  [1] "month"        "day"          "dep_time"     "dep_delay"
##  [5] "arr_time"     "arr_delay"    "carrier"      "tailnum"
##  [9] "flight"       "dest"         "air_time"     "distance"
## [13] "hour"         "minute"       "carrier_name" "dest_name"
```

Another good idea is to take a look at the dataset in table form. With this dataset having more than 50,000 rows, we won't explicitly show the results of the command here. I recommend you enter the command into the Console ***after*** you have run the **R** chunks above to load the data into **R**.

```
View(flights)
```

While not required, it is highly recommended you use the `dplyr` package to manipulate and summarize your data set as needed. It uses a syntax that is easy to understand using chaining operations. Below I've created a few examples of using `dplyr` to get information about the Portland flights in 2014. You will also see the use of the `ggplot2` package, which produces beautiful, high-quality academic visuals.

We begin by checking to ensure that needed packages are installed and then we load them into our current working environment:

```
# List of packages required for this analysis
pkg <- c("dplyr", "ggplot2", "knitr", "devtools")
# Check if packages are not installed and assign the
# names of the packages not installed to the variable new.pkg
new.pkg <- pkg[!(pkg %in% installed.packages())]
# If there are any packages in the list that aren't installed,
# install them
if (length(new.pkg))
  install.packages(new.pkg, repos = "http://cran.rstudio.com")
# Load packages
library(dplyr)
library(ggplot2)
library(knitr)
```

The example we show here does the following:

- Selects only the `carrier_name` and `arr_delay` from the `flights` dataset and then assigns this subset to a new variable called `flights2`.
- Using `flights2`, we determine the largest arrival delay for each of the carriers.

```
flights2 <- flights %>% dplyr::select(carrier_name, arr_delay)
max_delays <- flights2 %>% group_by(carrier_name) %>%
  summarize(max_arr_delay = max(arr_delay, na.rm = TRUE))
```

We next introduce a useful function in the `knitr` package for making nice tables in *R Markdown* called `kable`. It produces the LaTeX code required to make the table and is much easier to use than manually entering values into a table by copying and pasting values into Excel or LaTeX. This again goes to show how nice reproducible documents can be! There is no need to copy-and-paste values to create a table. (Note the use of `results = "asis"` here which will produce the table instead of the code to create the table. You'll learn more about the `\\label` later.) The `caption.short` argument is used to include a shorter version of the title to appear in the List of Tables at the beginning of the document.

```
kable(max_delays, col.names = c("Airline", "Max Arrival Delay"),
      caption = "Maximum Delays by Airline \\label{tab:max_delay}",
      caption.short = "Max Delays by Airline")
```

Table 1: Maximum Delays by Airline

Airline	Max Arrival Delay
Alaska Airlines Inc.	338
American Airlines Inc.	1539
Delta Air Lines Inc.	651
Frontier Airlines Inc.	575
Hawaiian Airlines Inc.	407
JetBlue Airways	273
SkyWest Airlines Inc.	421
Southwest Airlines Co.	694
United Air Lines Inc.	472
US Airways Inc.	347
Virgin America	366

We can further look into the properties of the largest value here for American Airlines Inc. To do so, we can isolate the row corresponding to the arrival delay of 1539 minutes for American in our original `flights` dataset.

```
flights %>% dplyr::filter(arr_delay == 1539,
                          carrier_name == "American Airlines Inc.") %>%
  dplyr::select(-c(month, day, carrier, dest_name, hour,
            minute, carrier_name, arr_delay))
```

```
##   dep_time dep_delay arr_time tailnum flight dest air_time distance
## 1     1403      1553     1934  N595AA   1568  DFW      182     1616
```

We see that the flight occurred on March 3rd and departed a little after 2 PM on its way to Dallas/Fort Worth. Lastly, we show how we can visualize the arrival delay of all departing flights from Portland on March 3rd against time of departure.

```
flights %>% dplyr::filter(month == 3, day == 3) %>%
  ggplot(aes(x = dep_time, y = arr_delay)) +
  geom_point()
```