

metabólico. El caso de Archaea es llamativo porque las copias de metabolismo central llegan en promedio hasta .5 copias por genoma, es decir muchos genomas de Archaea no cuentan con una copia de PriA, y en cambio, contrario a Actinobacteria, un 50% de la copias es marcado en negro, es decir varios de los genomas que tienen al menos una copia de PriA en realidad tienen dos copias. Esta figura muestra que en Actinobacteria PriA constituye un ejemplo de familia promiscua mayoritariamente distribuida con una sola copia por genoma. Esta observación demuestra que para que una familia sea promiscua no es imperativo tener copias extra con marcas de reclutamiento en metabolismo especializado. Aunque las copias extra suelen ser una indicación de promiscuidad, no son una condición necesaria. Tanto EvoMining como CORASON mostraron que existen excepciones de organismos donde PriA tiene doble copia, tanto en Actinobacteria como en otros linajes genómicos.

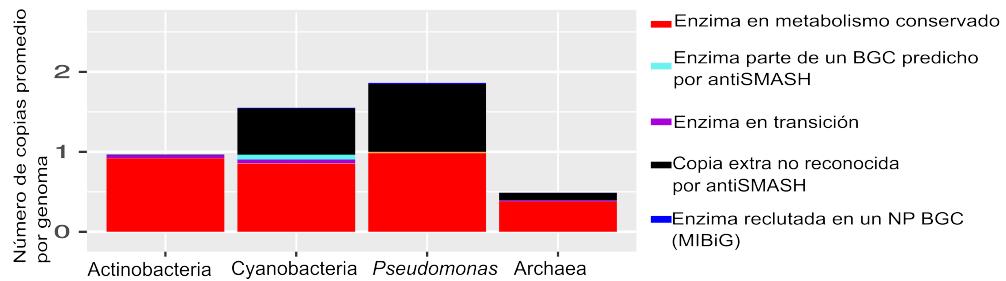


Figure 1: Número promedio de copias por genoma de PriA en Actinobacteria, Cyanobacteria, Pseudomonas y Archaea. Los colores muestran el destino metabólico asignado a cada copia según EvoMining. En rojo están los BBH a las enzimas semilla de metabolismo central. En morado las enzimas de metabolismo central también reconocidas por antiSMASH como parte de un BGC y en negro las copias sin un destino metabólico conocido.

Después del conteo de número de copias promedio, se analizaron los árboles de PriA de EvoMining, tanto los coloreados de acuerdo al número de copias, Figure 2, como los coloreados según el destino metabólico, Figure 3. En Actinobacteria la mayoría de las hojas son verdes reafirmando que existe sólo una copia por genoma en ese organismo. Sin embargo, existen varias hojas de color amarillo, indicando de dos copias en ese genoma. Las Actinobacterias con dos copias son *Ornithinimicrobium pekingense* DSM 21552, *Pseudonocardia* sp. P2, *Sericicoccus marinus* DSM 15273, *Sericicoccus profundi* MCCC 1A05965, *Sphaerobacter thermophilus* DSM 20745 y *Streptomyces* sp. CT34.

En Cyanobacteria, Pseudomonas y Archaea en contraste con Actinobacteria, se muestran una mezcla entre organismos que poseen una (verdes) o dos copias (amarillos) de PriA. Sin embargo al analizar detalladamente los árboles producidos por EvoMining en los distintos linajes, observamos que las copias extra tanto de Cyanobacteria como de Pseudomonas son en realidad miembros de otras familias enzimáticas, que guardan cierta similitud de secuencia con PriA. En Cyanobacteria y Pseudomonas la copia extra está en una rama divergente y muy poblada del árbol. En ambos casos esta segunda copia está en su mayoría anotada por RAST como imidazol glicerol fosfato sintasa ciclase. En Archaea sin embargo, diversas especies de la clase Methanomicrobia sí tienen dos copias de PriA.

Después de explorar cuáles organismos tienen expansiones de PriA, a continuación se muestra en la Figure 3 el posible destino metabólico de las copias extra de la familia. El árbol de Actinobacteria está poblado de hojas rojas, es decir de PriA dedicadas al metabolismo conservado, en este caso a las rutas de Histidina y Triptofano. Sin embargo hay algunas hojas grises, como es el caso de los dos *Sericicoccus* mencionados previamente. Es posible que estas PriA puedan tener funciones alternativas. Además, en Actinobacteria la PriA de *Janibacter Hoilley*, la rama más larga del árbol, es muy divergente. Esto se debe a que existe una fusión de PriA con HisH, que no parece ser un artefacto de anotación ya que hay otros *Janibacter* con PriA ligeramente más grandes que el promedio.

En Cyanobacteria hay pocas hojas verdes (EvoMining predictions) y no están localizadas cerca de saxitoxin, la HisA marcada como BGC de Cyanobacteria. En Pseudomonas hay una gran población de predicciones de EvoMining, pero son falsos positivos, ya que estas no corresponden a una rama dedicada al metabolismo secundario, sino a la rama de la ciclase. En Archaea algunas de las hojas grises, es decir sin destino metabólico conocido serán exploradas en la siguiente sección.

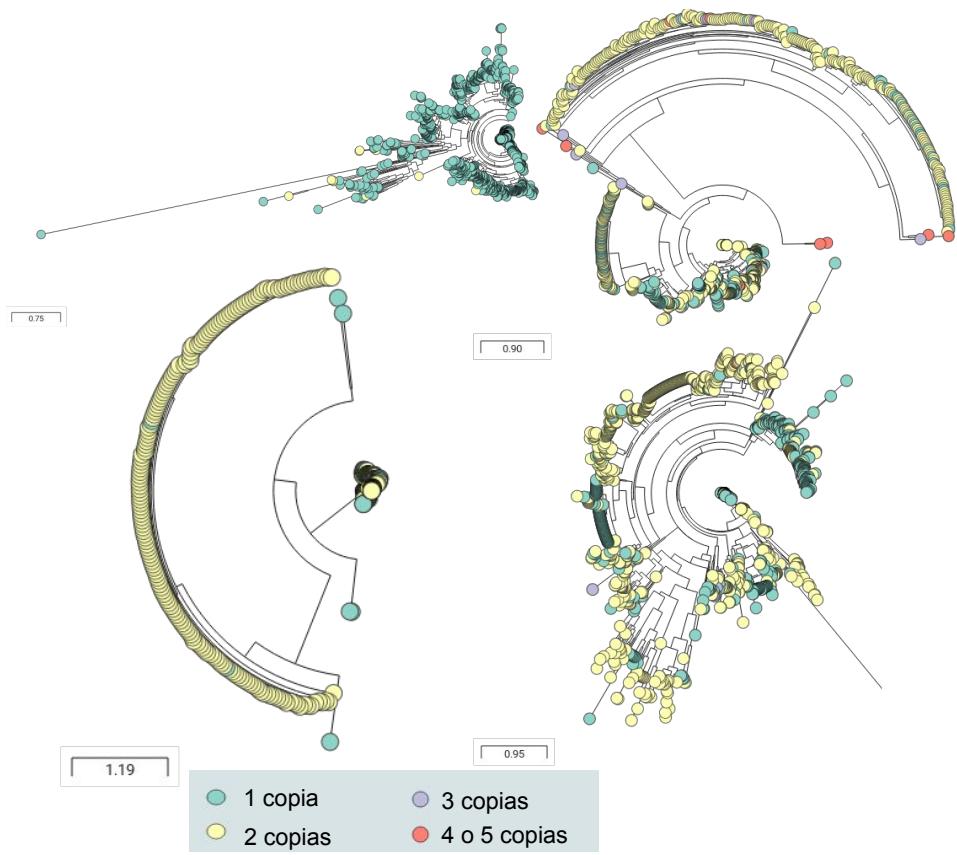


Figure 2:

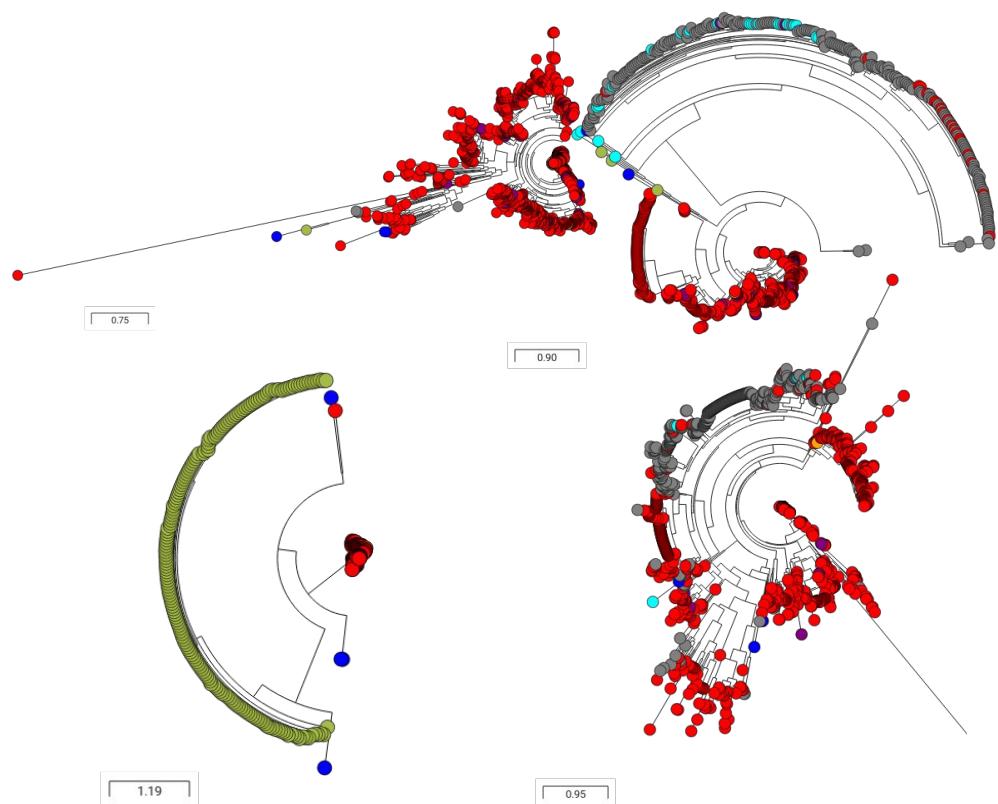


Figure 3: Árboles de destino metabólico de PriA en Actinobacteria, Cyanobacteria, Pseudomonas y Archaea según EvoMining

conservada que prevalezca en todo Archaea. Esto puede deberse a que Archaea es un dominio, no un phylum como Actinobacteria, y por tanto hay una mayor distancia entre los organismos que la conforman.

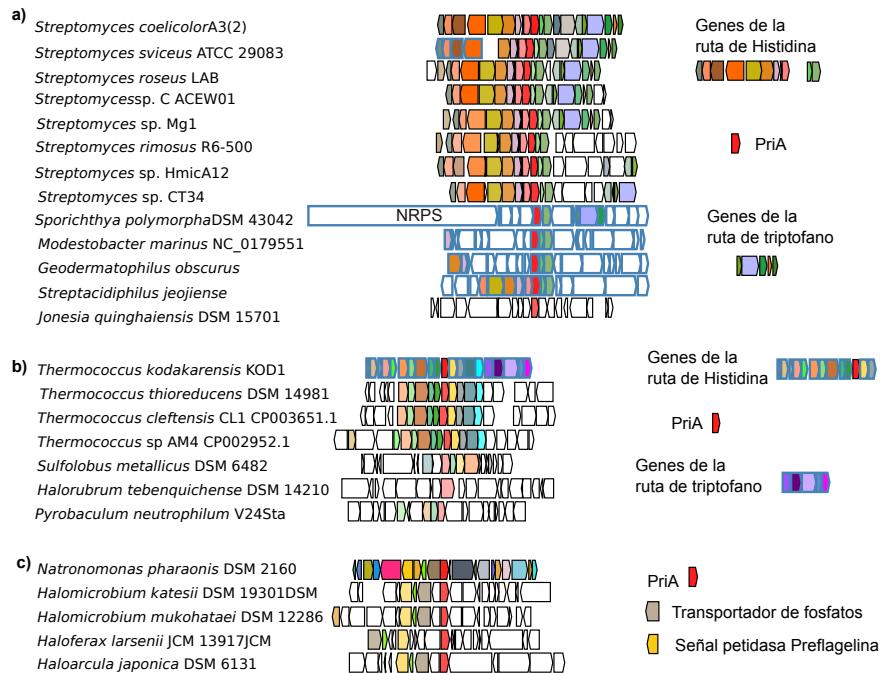


Figure 4: Contextos de PriA en Actinobacteria y Archaea

En cuanto a PriA en Cyanobacteria la visualización de contextos producida por CORASON tomando como referencia el cluster de saxitoxin, muestra que las enzimas biosintéticas de ese no están conservadas en otros organismos. Además en el lado izquierdo de la Figure 5 se muestra que la PriA del BGC saxitoxin no está ubicada en una rama de PriA divergente, al contrario está en la parte más conservada. Por estas razones es posible que PriA esté en la orilla del BGC de saxitosin y más bien no participe en la síntesis de este compuesto.

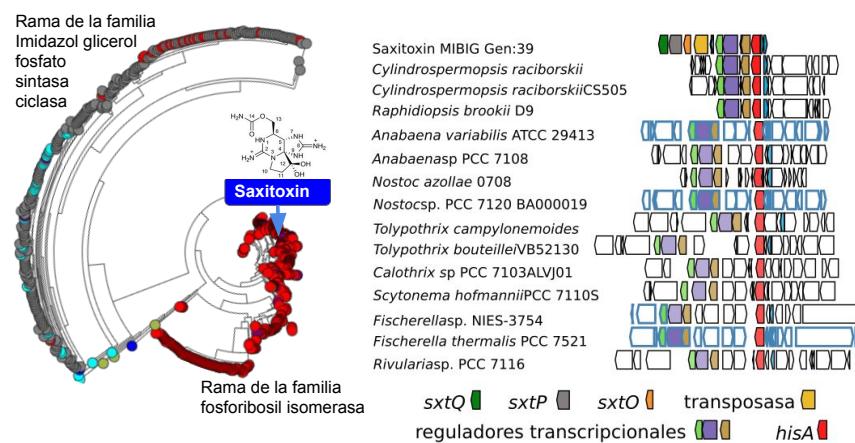


Figure 5: HisA en saxitoxin, un cluster de Cyanobacteria

El monitoreo en cambios de aminoácidos en PriA para estudiar rutas evolutivas y reconstrucciones de la estructura tridimensional.

En esta segunda sección concerniente a la familia PriA se busca información contenida en la secuencia de aminoácidos. En la primera parte se discute cómo en datos de evolución dirigida en el laboratorio no se encontró ninguna trayectoria en donde algún paso incrementara la actividad de PriA en sus dos sustratos nativos. En la segunda parte se muestra una reconstrucción de la estructura tridimensional de PriA basada en la covarianza de sus aminoácidos en secuencias del registro evolutivo.

Al transformar una subHisA en una PriA mediante mutaciones no se observó ninguna trayectoria creciente para ambos sustratos (darwiniana)

En esta sección analizamos cómo cambia la capacidad catalítica de PriA sobre un sustrato mientras se varía la del otro. Para ello se utilizaron datos de mutantes de subHisA de *Corynebacterium diphtheriae*. Estas mediciones de cinéticas enzimáticas fueron obtenidas del trabajo de tesis de Lianet Noda [[@noda_tesis_2012](#)]. A partir de la secuencia original que se mostró es una subHisA, se realizaron mutantes con el objetivo de alcanzar la promiscuidad, es decir de convertir la enzima subHisA en una PriA. Se comenzó con diferentes mutantes puntuales adicionando una mutación cada vez, hasta llegar a una con 9 mutaciones, la que alcanzó mayor actividad PRA isomerasa. En esta colección de mutantes varias ganaron la función de PRA isomerasa, a distintos niveles. En estos datos quedaba pendiente la exploración de las rutas, es decir cómo es el camino desde una mutante sencilla hasta una múltiple ¿cuántas rutas son? ¿Existe alguna tendencia en ciertos momentos de la ruta sobre el incremento/decremento de alguna de las dos funciones?

Así pues se desarrolló un script utilizando recursividad para reconstruir todas las rutas posibles. Las rutas son mostradas en la Figure 6.

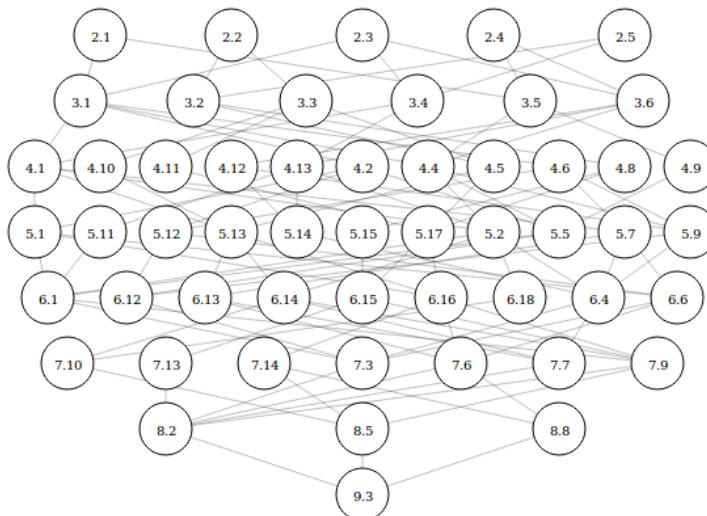


Figure 6: Rutas desde una subHisA hasta una variante con 9 mutaciones. En cada círculo el primer dígito indica el número de mutaciones

Al analizar los cambios sufridos en cada actividad en cada paso de cada ruta se descubrió que no existe en ellas una trayectoria Darwiniana para ambos sustratos. En la Figure 6 se muestran las rutas donde cada mutante mantiene un nivel mínimo de actividad de ProFAS isomerasa ($\frac{K_{cat}}{K_m}$ ProFAR $\geq .004$). En azul se ven los incrementos en ProFAR y en rojo los incrementos en PRA. Es decir al menos en este ejemplo, las

mutaciones puntuales que llevan a una enzima monofuncional a adquirir promiscuidad no mantienen una tendencia no decreciente de principio a fin sobre alguna reacción (isomerización de PRA y de ProFAR). Siempre en algún paso decrece alguna de las actividades.

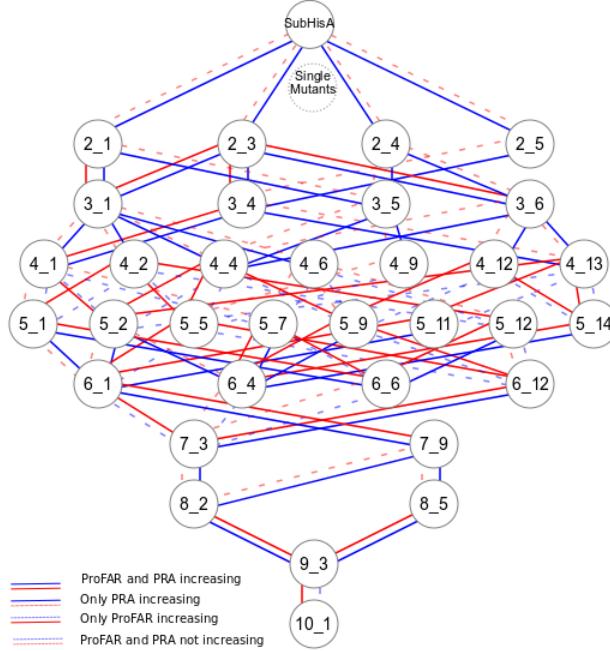


Figure 7:

Los residuos que covarián en el registro evolutivo de PriA permiten una reconstrucción aproximada de su estructura tridimensional

Evcouplings es un método que considera la información contenida en las secuencias genómicas como experimentos exitosos de la naturaleza. Con esta información obtiene la covariación de pares de aminoácidos y reconstruye con ellos la estructura tridimensional de una proteína. Los pares fuertemente relacionados se denominan acoplamientos, estos acoplamientos a menudo están cerca físicamente en la estructura terciaria de la proteína. Se ha demostrado que muchas proteínas contienen suficientes acoplamientos distribuidos en su secuencia para permitir su reconstrucción terciaria [1].

Las diferencias a nivel estructural pueden amplificar la información proporcionada por variaciones a nivel de secuencia. Por este motivo se decidió implementar Evcouplings [REF], para poder aplicarlo a la familia PriA. Este método es de difícil instalación ya que tiene muchas dependencias, por ello desarrollé un contenedor docker donde las dependencias y el software quedan instalados. Este desarrollo fue incluido por los desarrolladores originales como sugerencia de instalación.

El contenedor docker implementa EVcouplings python framework [2] que comprende cinco etapas para estudiar el análisis de coevolución de residuos de una familia de proteínas. Estas etapas son i) Alineado, ii) análisis de acoplamiento, iii) plegamiento basado en acoplamientos iv) análisis de mutación y v) comparación con estructuras conocidas.

EVcouplings fue aplicado a PriA de Streptomyces coelicolor con identificador de Uniprot HIS4_STRCO. Las

secuencias para las alineaciones se recuperaron automáticamente de Uniprot mientras que otros parámetros se dejaron con la configuración inicial del archivo de configuración.

Resultado se modelo PriA con evcouplings y los aminoácidos encontrados coinciden con los encontrados experimentalmente.

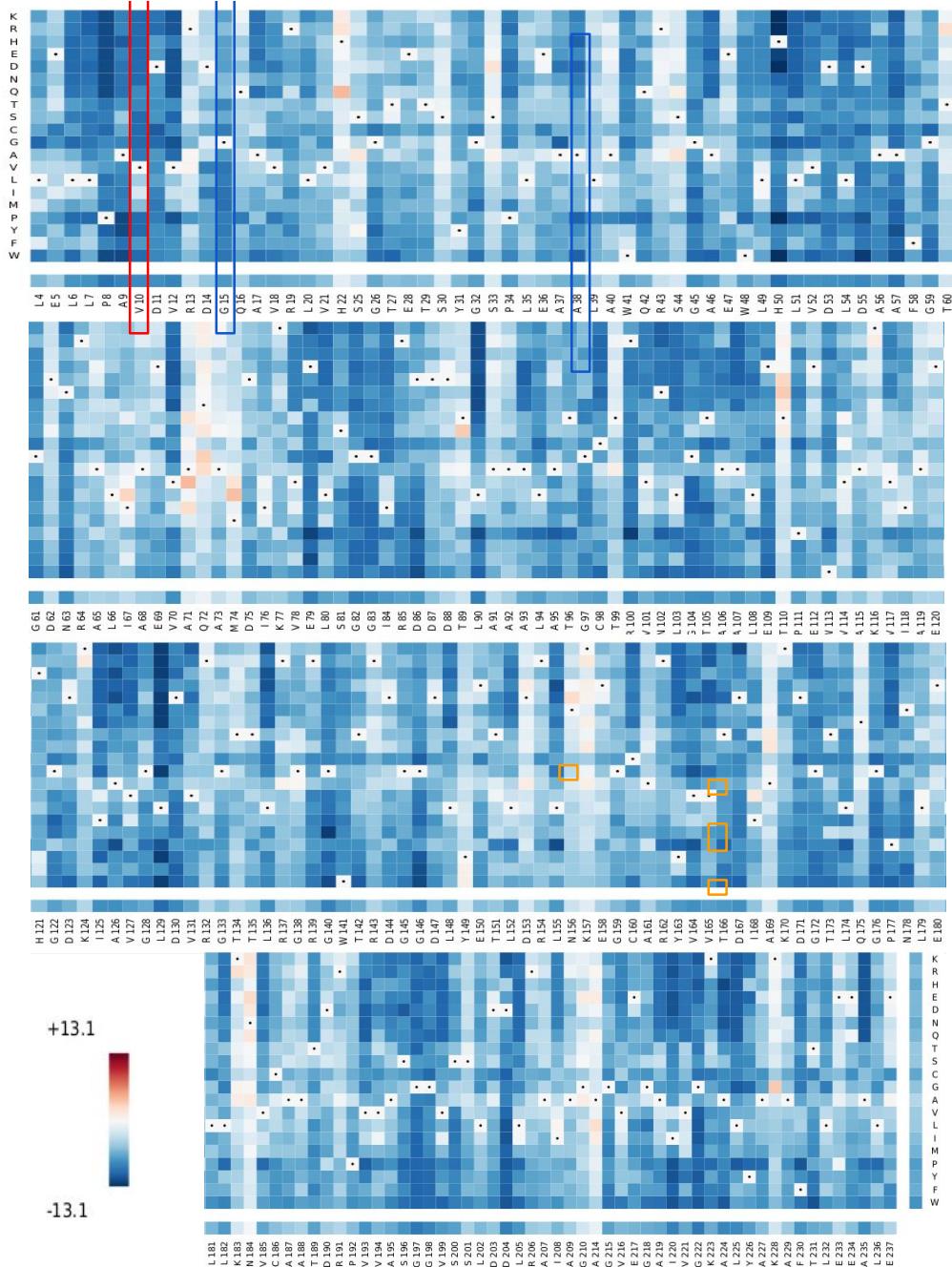


Figure 8:

La estructura reconstruida es parecida, pero para obtener mejor refinamiento se debe seleccionar mejor las secuencias del alineamiento

Finalmente, estos aminoácidos fueron comparados con los provistos por EVCouplings, de covariación.

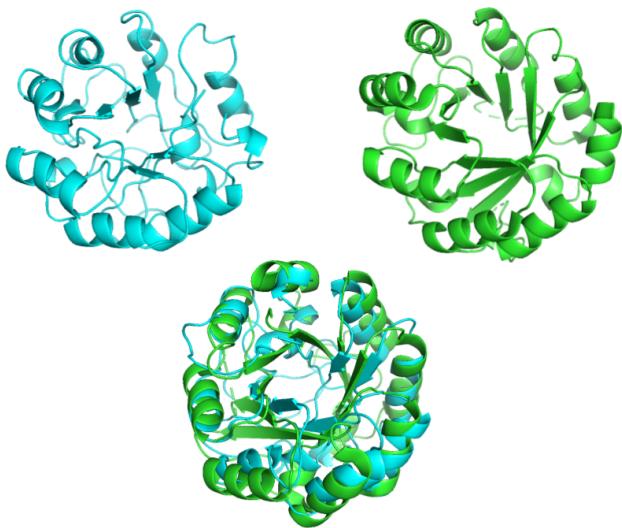


Figure 9:

<i>Corynebacterium</i>	<i>Streptomyces</i>
D20V	21V
L48I	49L
F50L	51L
M66I	67I
T80S	81S
A97C	98C
D127A	128G
A129D	130D
T139L	136L
Y214L	211Y
E230A	227A

Afinidad de enzimas selectas por sustratos químicamente parecidos a PRA y PROFAR

Algunas de estas enzimas se seleccionaron en busca de nuevas funciones promiscuas, las reacciones catalizadas por los candidatos se investigaron mediante la exploración de sustratos químicamente similares a los nativos.

Dado que en *Streptomyces* se han encontrado representantes de las familias PriA y PriB, se seleccionaron 39 secuencias homólogas para realizar el análisis del sustrato de la enzima de acoplamiento. Estas secuencias seleccionadas de PriA / PriB pertenecen a *Streptomyces* uniformemente distribuidas en un árbol de especies RpoB con diferentes condiciones sobre la presencia / ausencia de TrpF. En este estudio se incluyeron otros homólogos de PriA Actinobacterial caracterizados químicamente, y finalmente se agregaron HisA de *Escherichia coli*, *Arthrobacter Aurescens*, *Salmonella enterica* y *Acidimicrobium ferrooxidans* y Actinobacterial TrpF como controles.

Cuando existían estructuras de cristal, de lo contrario, se generaban estructuras homólogas utilizando como plantilla la enzima más cercana disponible con estructura de cristal.

Controls

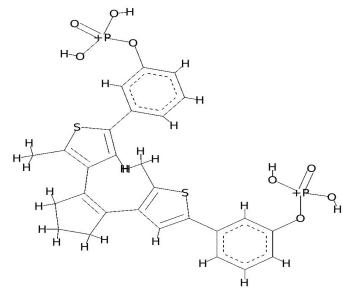
HisA Enterobacteria enzymes from *Salmonella enterica* (PDB:5AHE), *Escherichia coli K12* *Acidimicrobium ferrooxydans* (PDB:4WD0)

TrpF Actinobacteria *Jonesia denitrificans* and *Streptomyces sp Mg1* sequences.

Chemically characterized Actinobacterial enzymes 12

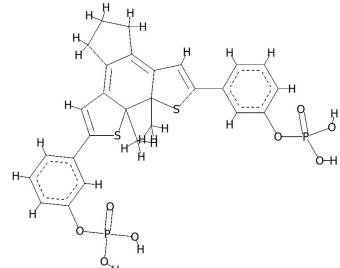
PriA

Mycobacterium tuberculosis (Mtub PDB:2Y88,2Y89,2Y85,3ZS4) *Streptomyces coelicolor* (Scoe PDB:2VEP,2X30,1VZW), *Streptomyces globisporus*, *Actynomyces urogenitalis* 4X2R *Corynebacterium jeikeum*



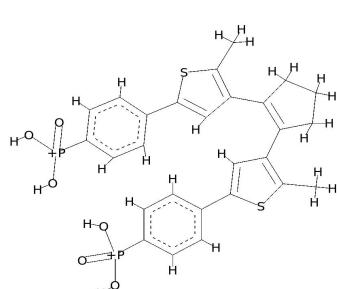
13

DTE-meta-phosphate(dte6_Open form)



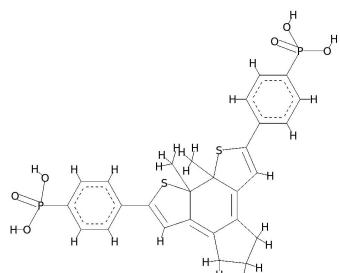
14

DTE-meta-phosphate(dte6_Closed form)



15

DTE-Para-Phosphonate(dte13_Closed form)



16

DTE-para-phosphonate(dte13_Closed form)

Figure 10: Substatos químicamente similares a los sustratos de PriA (parte 1)

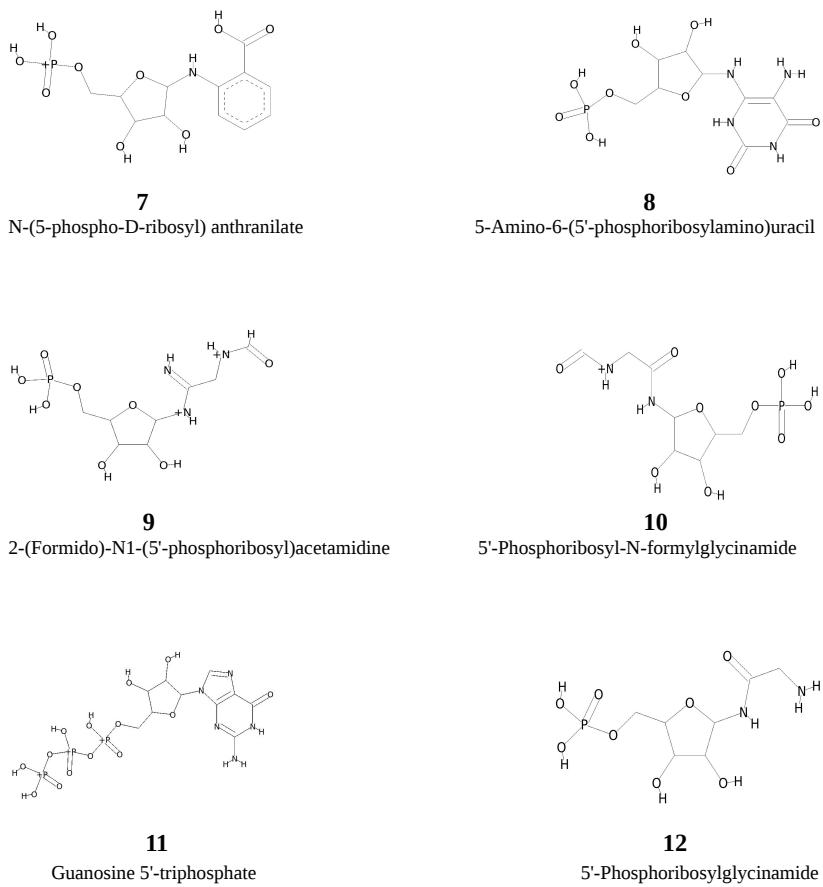
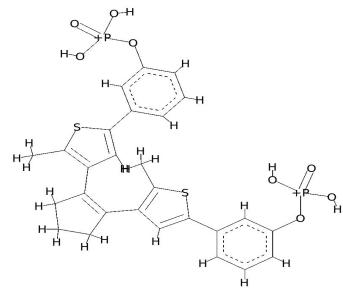
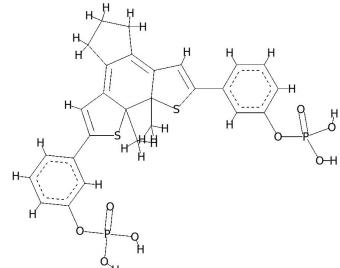


Figure 11: Substatos químicamente similares a los sustratos de PriA (parte 2)



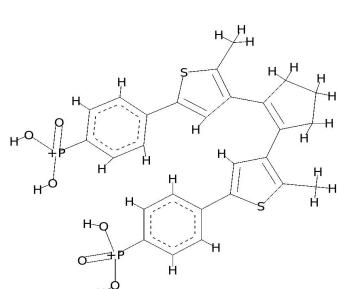
13

DTE-meta-phosphate(dte6_Open form)



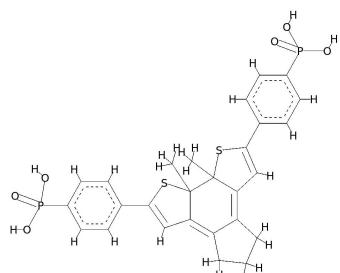
14

DTE-meta-phosphate(dte6_Closed form)



15

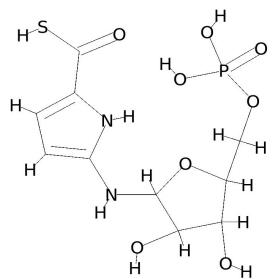
DTE-Para-Phosphonate(dte13_Closed form)



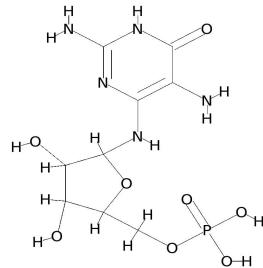
16

DTE-para-phosphonate(dte13_Closed form)

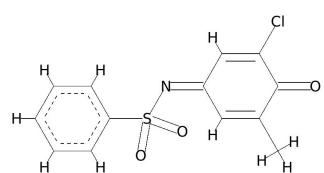
Figure 12: Substatos químicamente similares a los sustratos de PriA (parte 3)



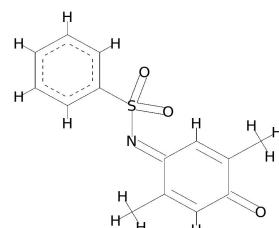
17
4N'-(5'-phosphoribosyl) 4-aminopyrrole
-2-carboxilate



18
2,5-di-amino-6-ribosylamino-4
(3H)-pyrimidinone 5'-phosphate



19
(E)-N-(3-chloro-5-methyl-4-oxocyclohexa-
2,5-dienylidene)benzenesulfonamide



20
2,5 dimethyl-N-(4-oxocyclohexa-
2,5-dienylidene)benzenesulfonamide

Figure 13: Substatos químicamente similares a los sustratos de PriA (parte 4)

Sustratos similares a PRA y PROFAR sugeridos por distancias de Tanimoto o probados previamente

S1, S2, ..., S20 sustratos fueron recolectados de la literatura y las predicciones de la quimioinformática. S3 PRA y S7 PROFAR son sustratos nativos, S13-S16 son sustratos activados por la luz, S17 PRAP, S18 Compuesto V, se encontraron en la literatura, S6 GMP, S11 GTP y otros fueron sugeridos por chemoinformatics.

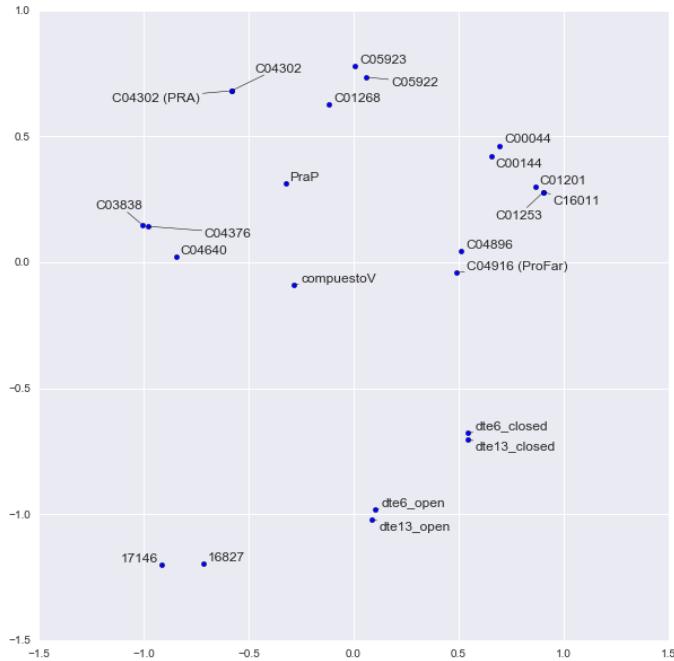


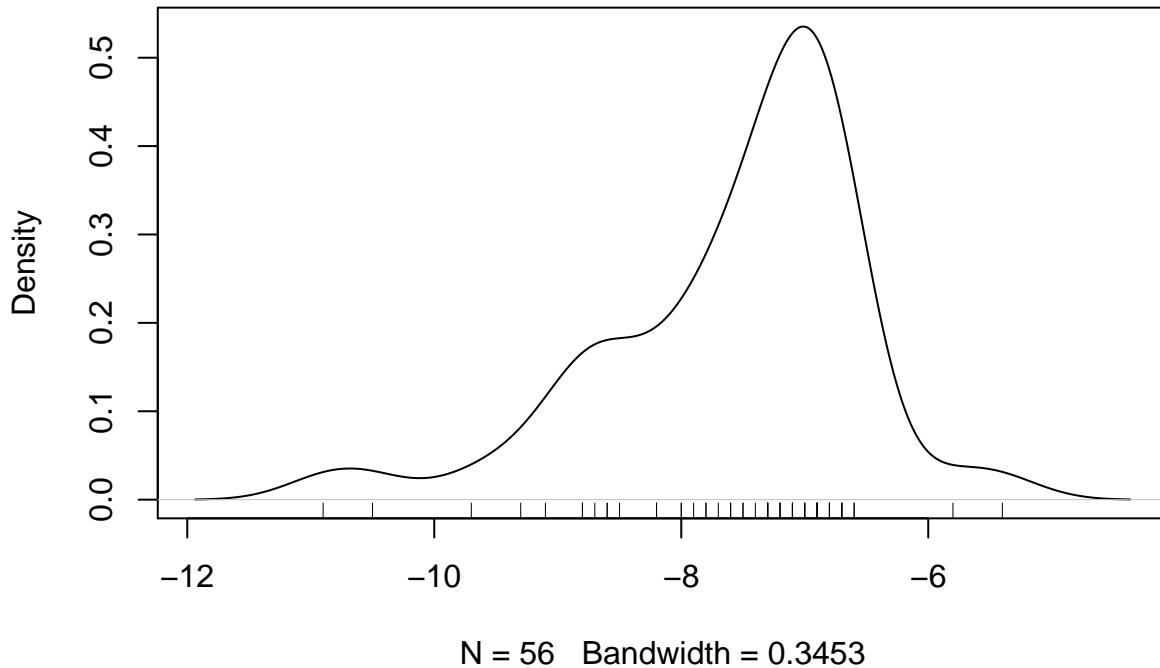
Figure 14:

On next figure we can see their chemical structures.

```
### Docking between PriA enzymes and selected substrates  
Para diversas enzimas PriA de Streptomyces se realizaron simulaciones de docking. Se incluyeron también como controles enzimas TrpF provenientes de Streptomyces Mg1, Jonesia denitrificans. Los procedimientos pueden ser consultados en Docking Protocols
```

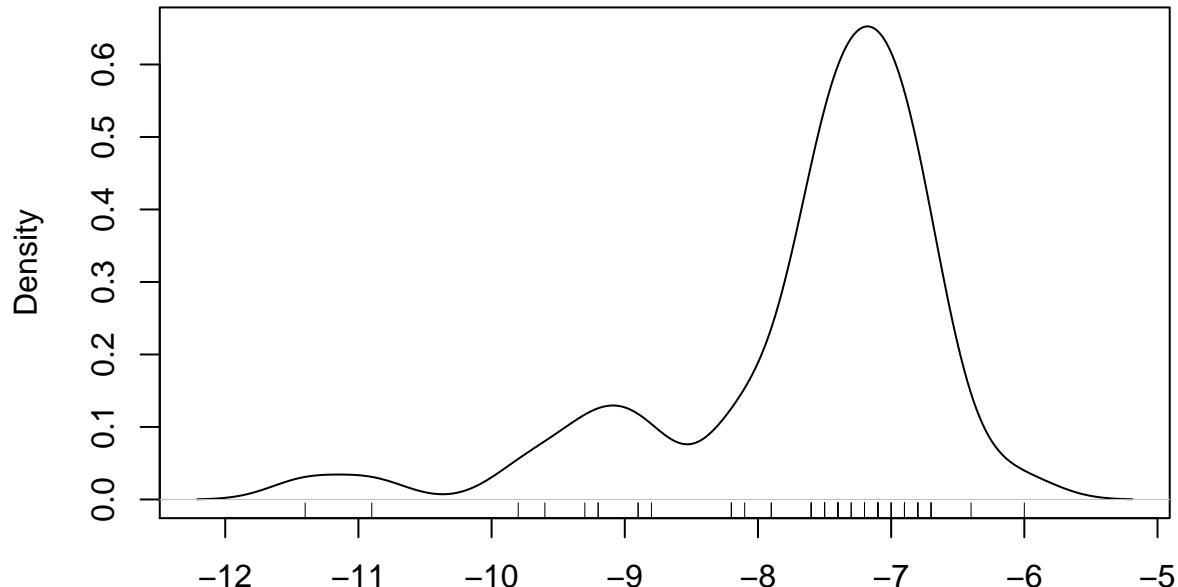
```
## Called from: eval(expr, envir, enclos) ## debug en <text>#4: plot(density(docking[,  
i], na.rm = T))
```

```
density.default(x = docking[, i], na.rm = T)
```



```
## debug en <text>#4: rug(docking[, i]) ## debug en <text>#4: browser() ## debug en  
<text>#4: plot(density(docking[, i], na.rm = T))
```

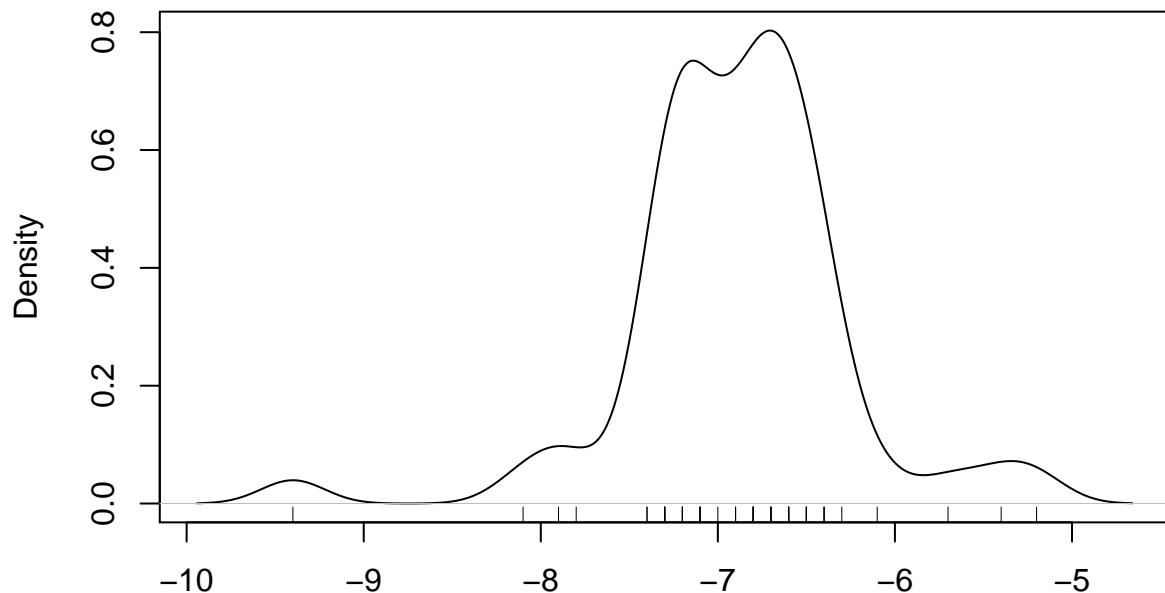
```
density.default(x = docking[, i], na.rm = T)
```



N = 56 Bandwidth = 0.2702

```
## debug en <text>#4: rug(docking[, i]) ## debug en <text>#4: browser() ## debug en
<text>#4: plot(density(docking[, i], na.rm = T))
```

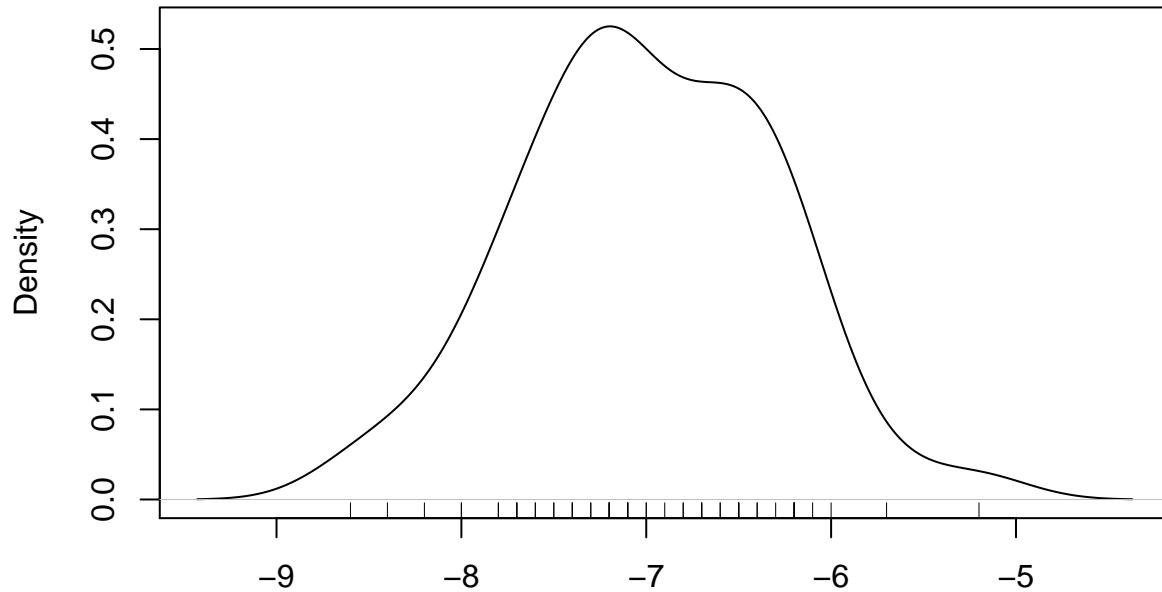
```
density.default(x = docking[, i], na.rm = T)
```



N = 56 Bandwidth = 0.1802

```
## debug en <text>#4: rug(docking[, i]) ## debug en <text>#4: browser() ## debug en
<text>#4: plot(density(docking[, i], na.rm = T))
```

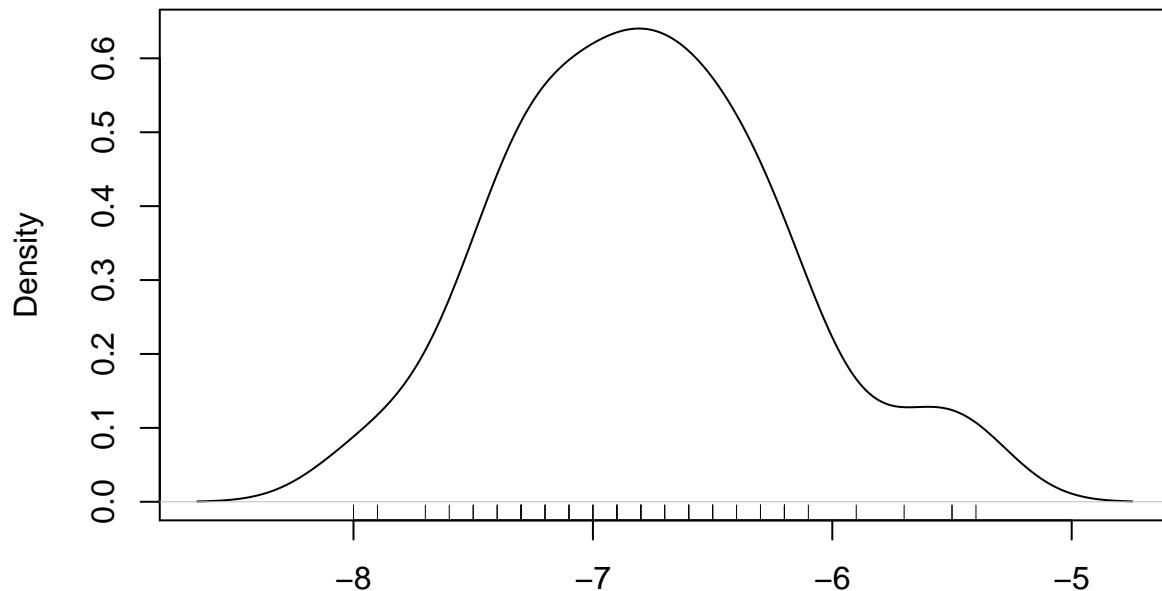
```
density.default(x = docking[, i], na.rm = T)
```



N = 56 Bandwidth = 0.2764

```
## debug en <text>#4: rug(docking[, i]) ## debug en <text>#4: browser() ## debug en
<text>#4: plot(density(docking[, i], na.rm = T))
```

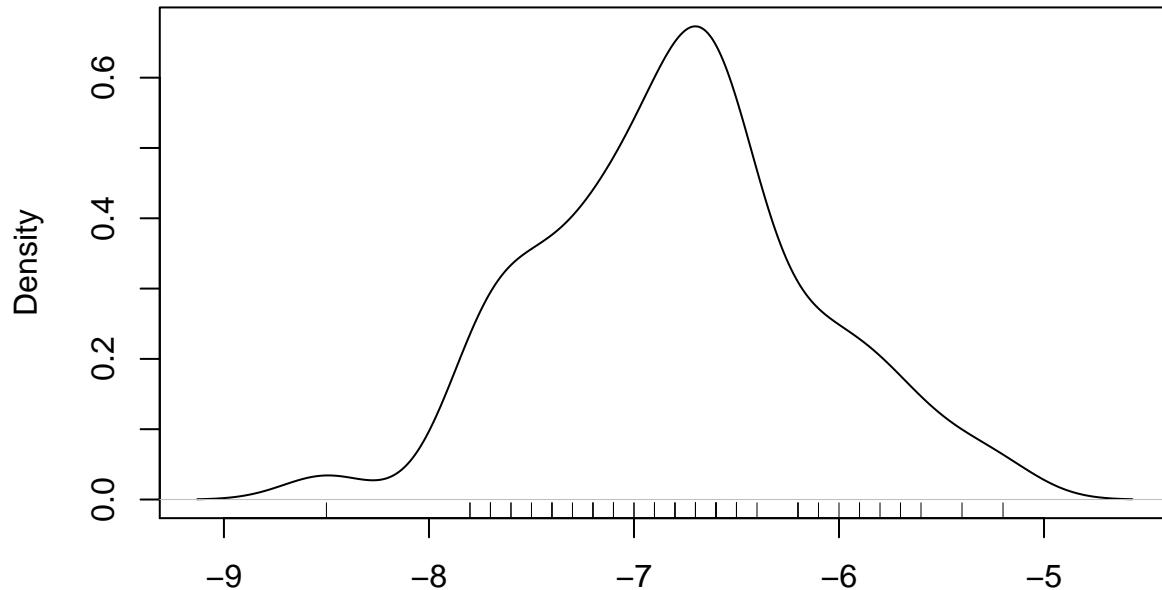
```
density.default(x = docking[, i], na.rm = T)
```



N = 56 Bandwidth = 0.2177

```
## debug en <text>#4: rug(docking[, i]) ## debug en <text>#4: browser() ## debug en
<text>#4: plot(density(docking[, i], na.rm = T))
```

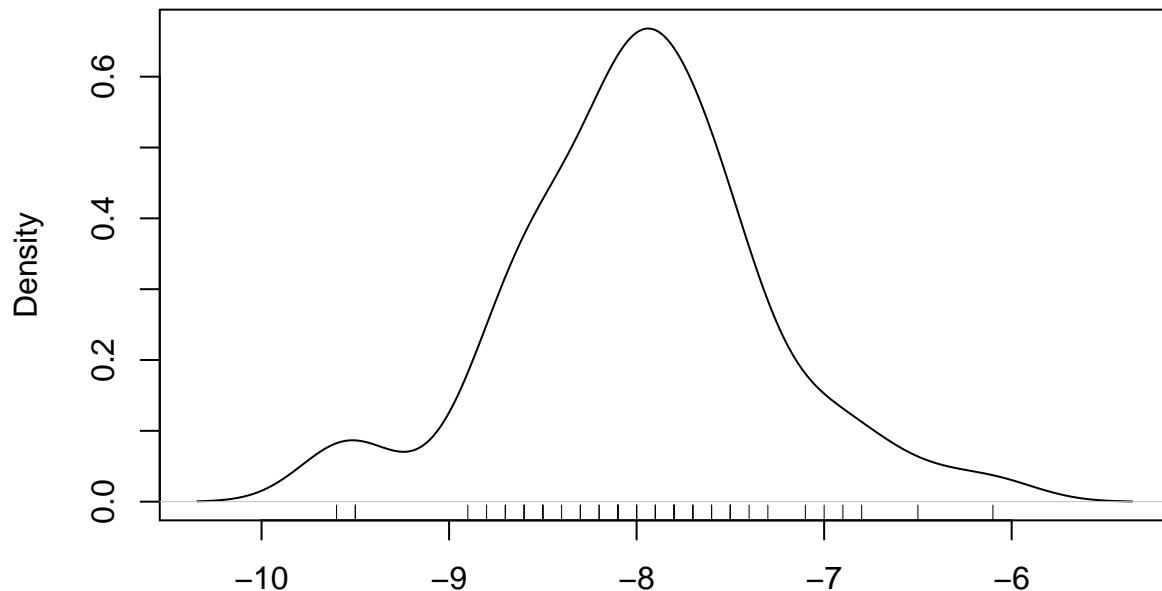
```
density.default(x = docking[, i], na.rm = T)
```



N = 56 Bandwidth = 0.2102

```
## debug en <text>#4: rug(docking[, i]) ## debug en <text>#4: browser() ## debug en
<text>#4: plot(density(docking[, i], na.rm = T))
```

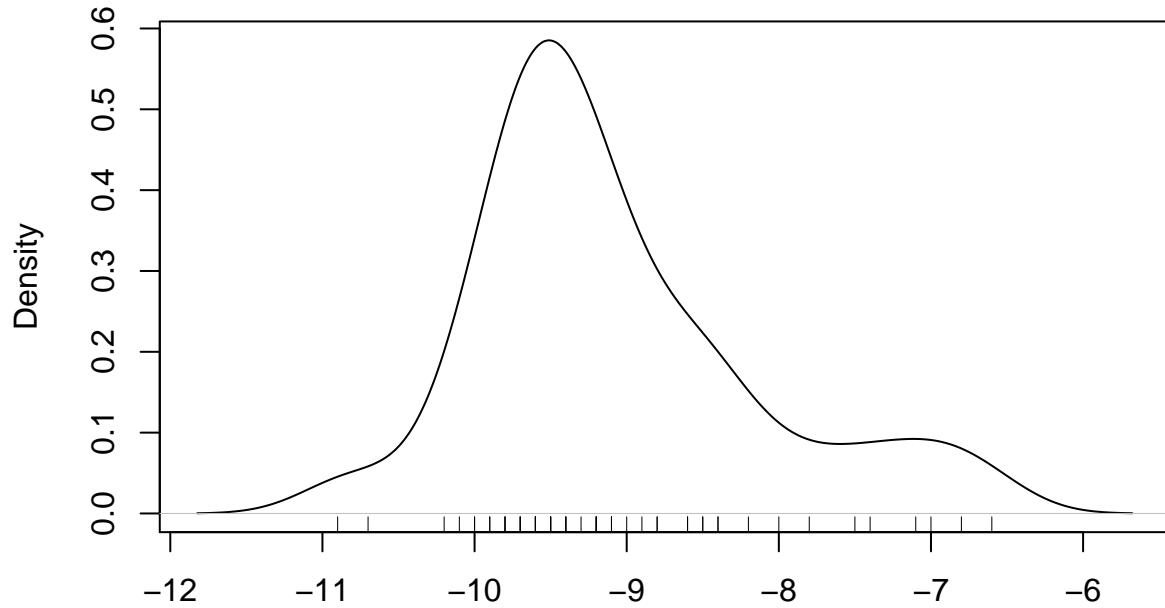
```
density.default(x = docking[, i], na.rm = T)
```



N = 56 Bandwidth = 0.2477

```
## debug en <text>#4: rug(docking[, i]) ## debug en <text>#4: browser() ## debug en
<text>#4: plot(density(docking[, i], na.rm = T))
```

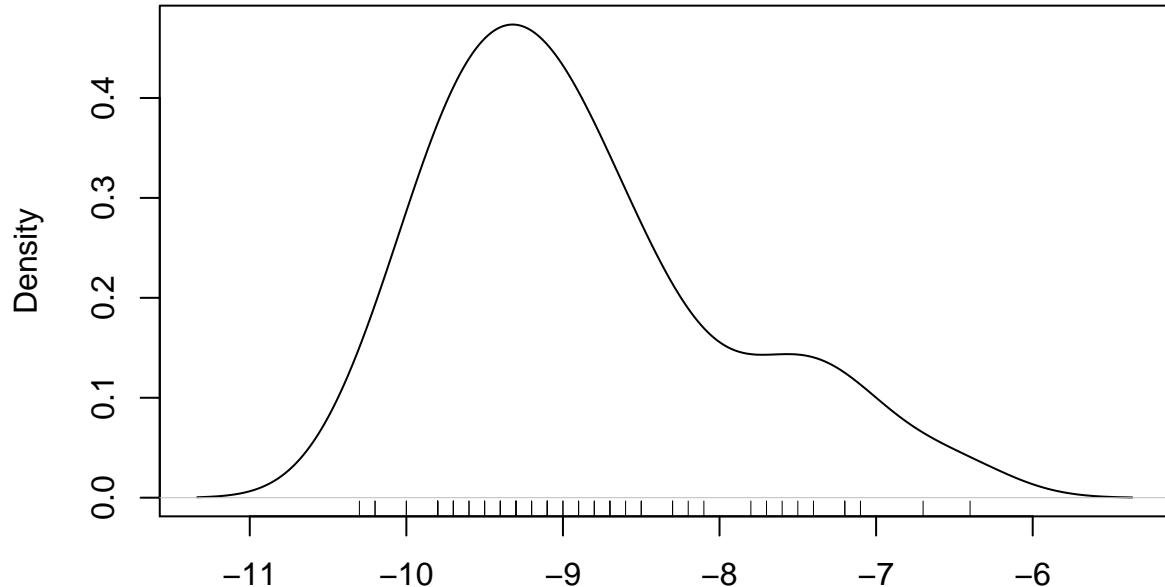
```
density.default(x = docking[, i], na.rm = T)
```



N = 56 Bandwidth = 0.3078

```
## debug en <text>#4: rug(docking[, i]) ## debug en <text>#4: browser() ## debug en
<text>#4: plot(density(docking[, i], na.rm = T))
```

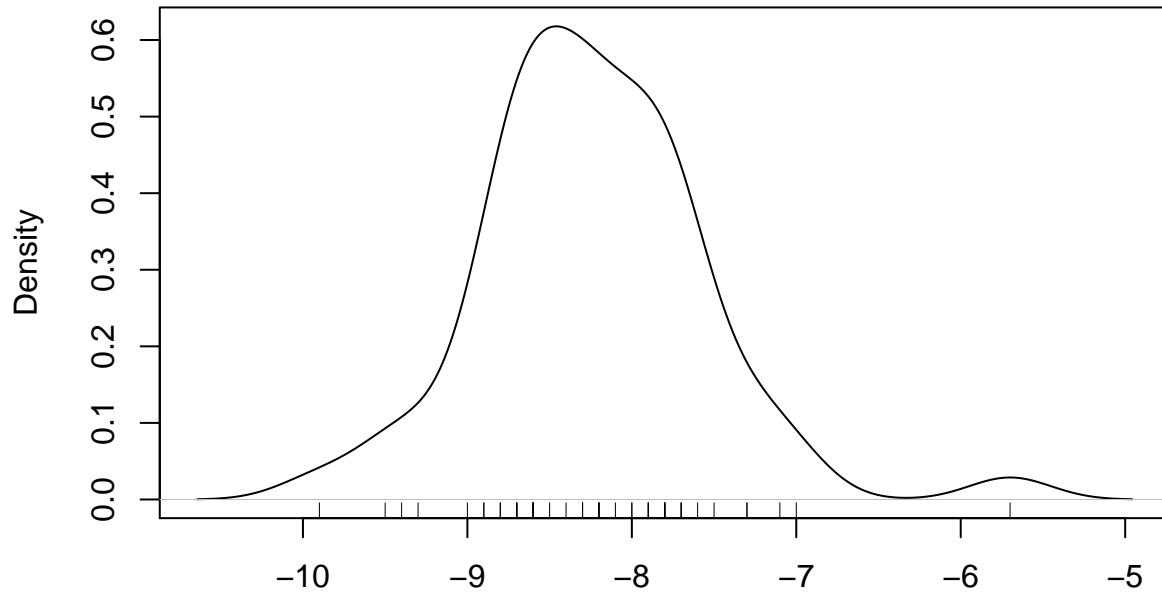
```
density.default(x = docking[, i], na.rm = T)
```



N = 56 Bandwidth = 0.3453

```
## debug en <text>#4: rug(docking[, i]) ## debug en <text>#4: browser() ## debug en
<text>#4: plot(density(docking[, i], na.rm = T))
```

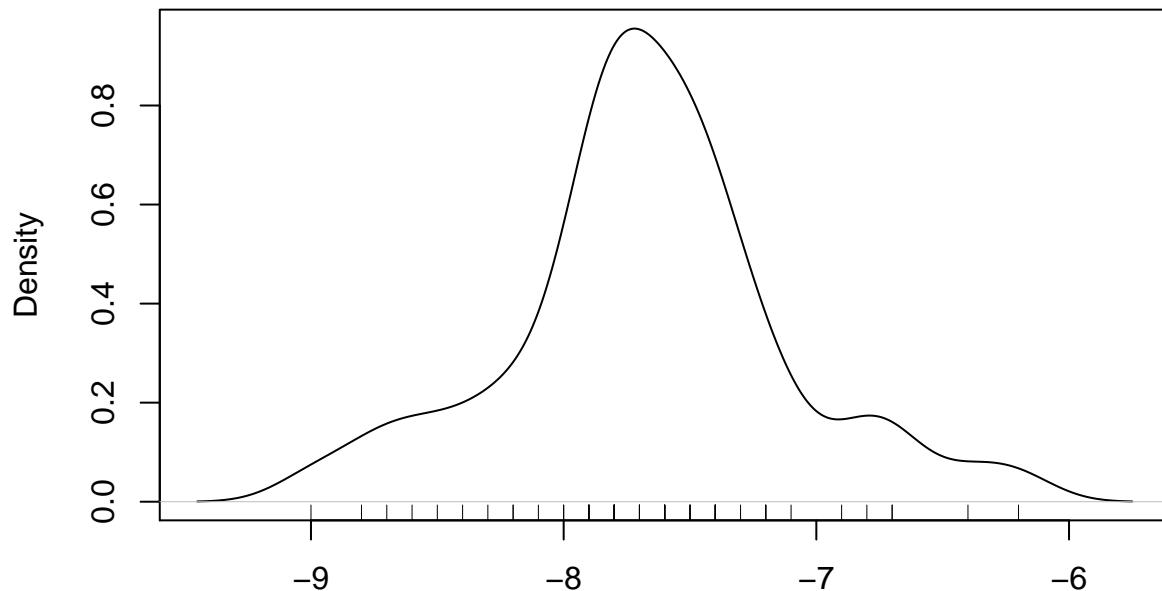
```
density.default(x = docking[, i], na.rm = T)
```



N = 56 Bandwidth = 0.2477

```
## debug en <text>#4: rug(docking[, i]) ## debug en <text>#4: browser() ## debug en
<text>#4: plot(density(docking[, i], na.rm = T))
```

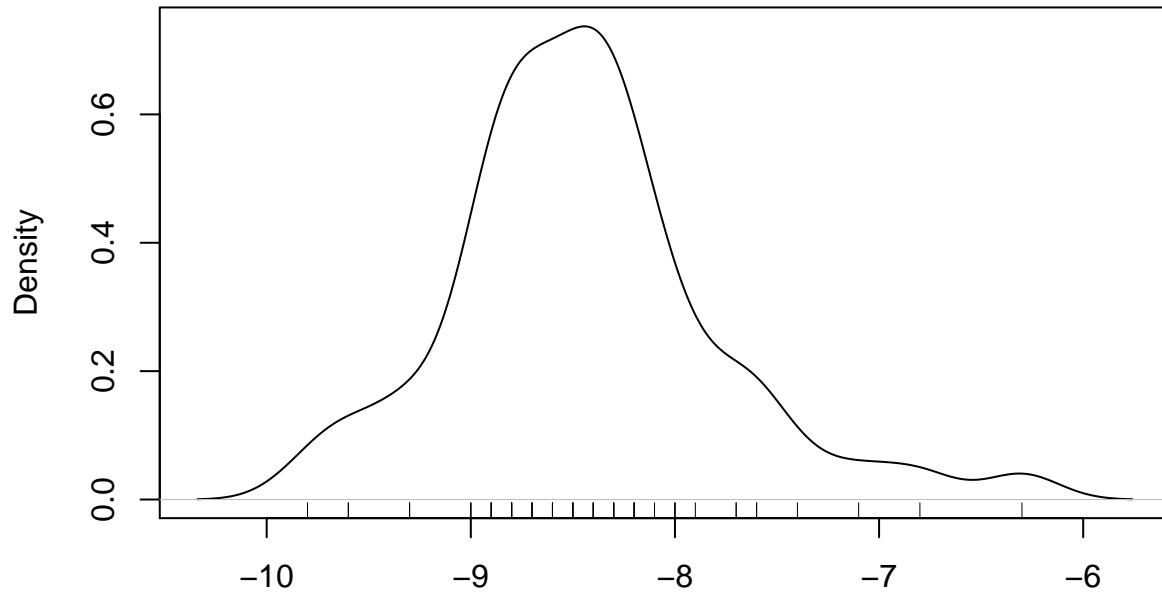
```
density.default(x = docking[, i], na.rm = T)
```



N = 56 Bandwidth = 0.1501

```
## debug en <text>#4: rug(docking[, i]) ## debug en <text>#4: browser() ## debug en
<text>#4: plot(density(docking[, i], na.rm = T))
```

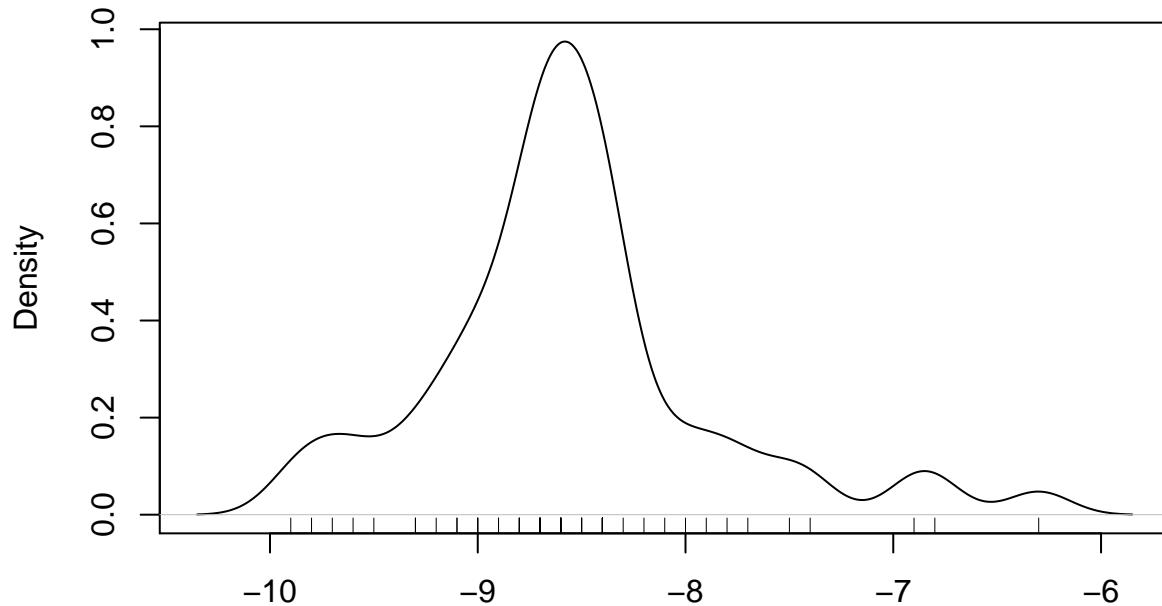
density.default(x = docking[, i], na.rm = T)



N = 56 Bandwidth = 0.1802

```
## debug en <text>#4: rug(docking[, i]) ## debug en <text>#4: browser() ## debug en
<text>#4: plot(density(docking[, i], na.rm = T))
```

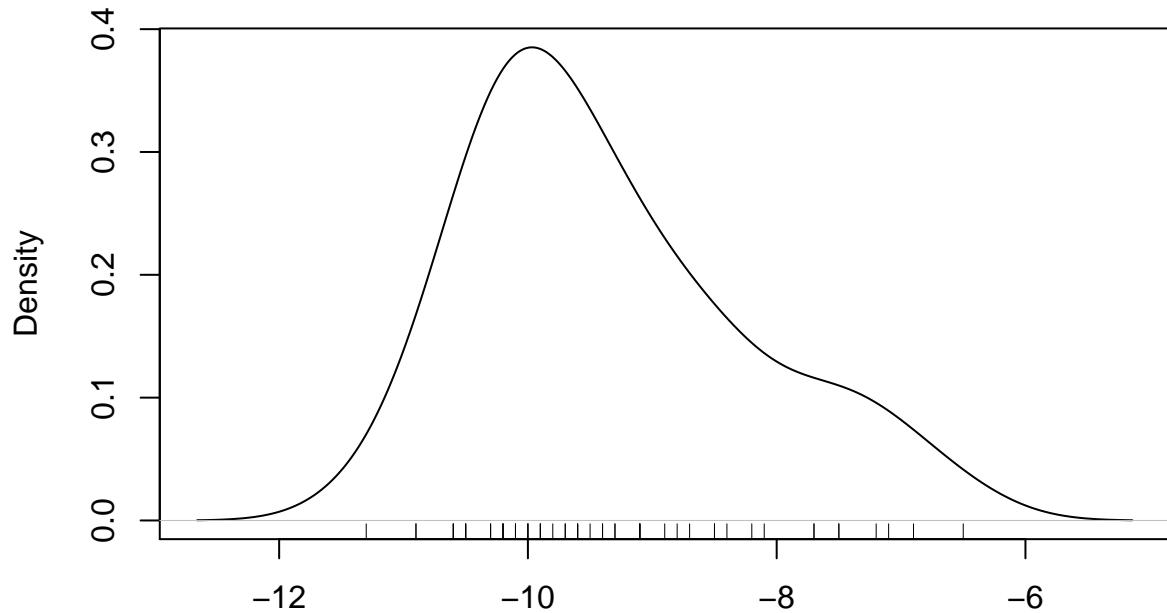
density.default(x = docking[, i], na.rm = T)



N = 56 Bandwidth = 0.1501

```
## debug en <text>#4: rug(docking[, i]) ## debug en <text>#4: browser() ## debug en
<text>#4: plot(density(docking[, i], na.rm = T))
```

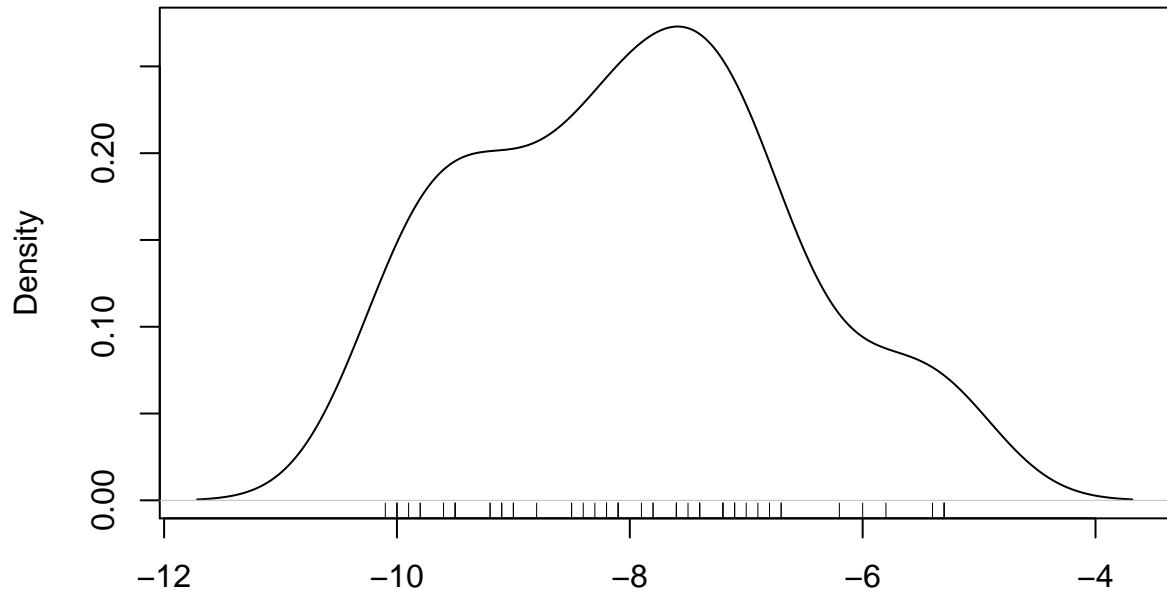
density.default(x = docking[, i], na.rm = T)



N = 56 Bandwidth = 0.453

```
## debug en <text>#4: rug(docking[, i]) ## debug en <text>#4: browser() ## debug en
<text>#4: plot(density(docking[, i], na.rm = T))
```

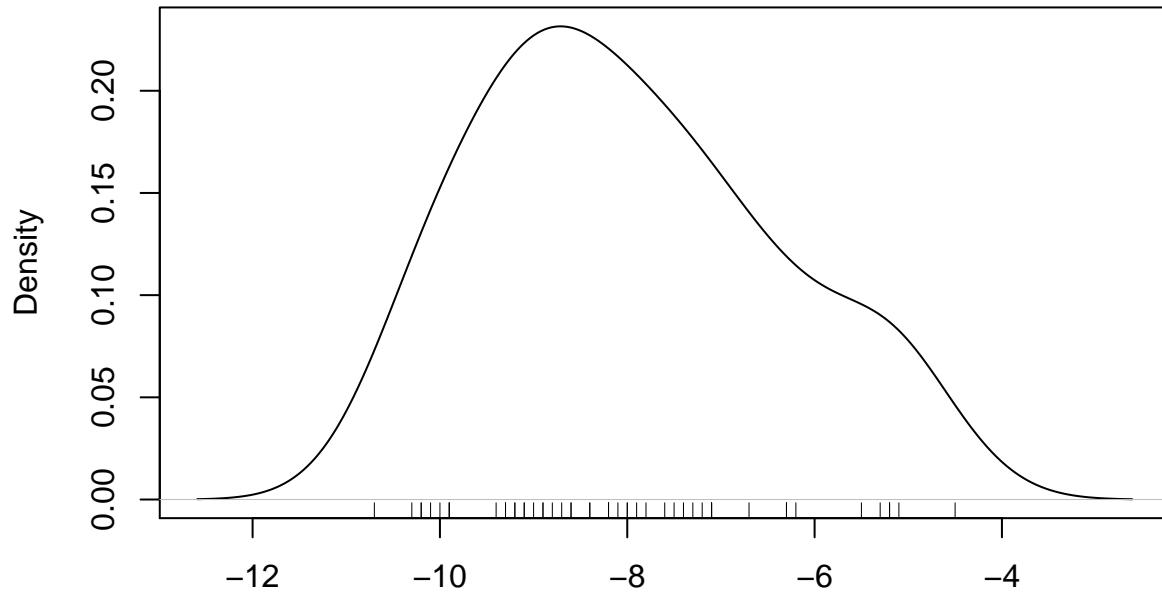
density.default(x = docking[, i], na.rm = T)



N = 56 Bandwidth = 0.5386

```
## debug en <text>#4: rug(docking[, i]) ## debug en <text>#4: browser() ## debug en
<text>#4: plot(density(docking[, i], na.rm = T))
```

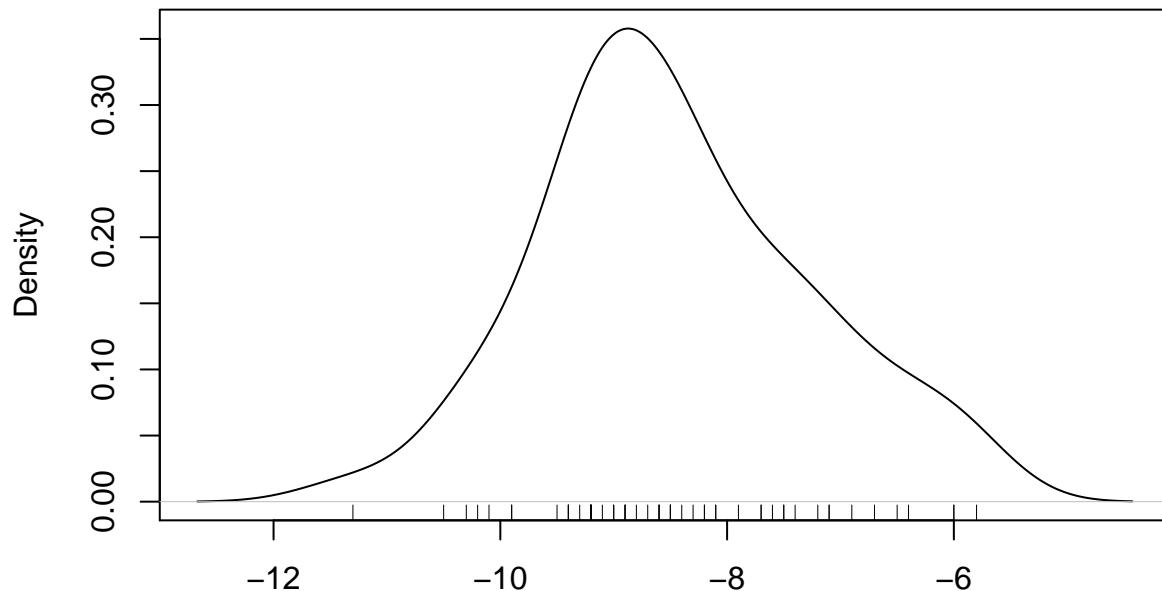
```
density.default(x = docking[, i], na.rm = T)
```



N = 56 Bandwidth = 0.6305

```
## debug en <text>#4: rug(docking[, i]) ## debug en <text>#4: browser() ## debug en
<text>#4: plot(density(docking[, i], na.rm = T))
```

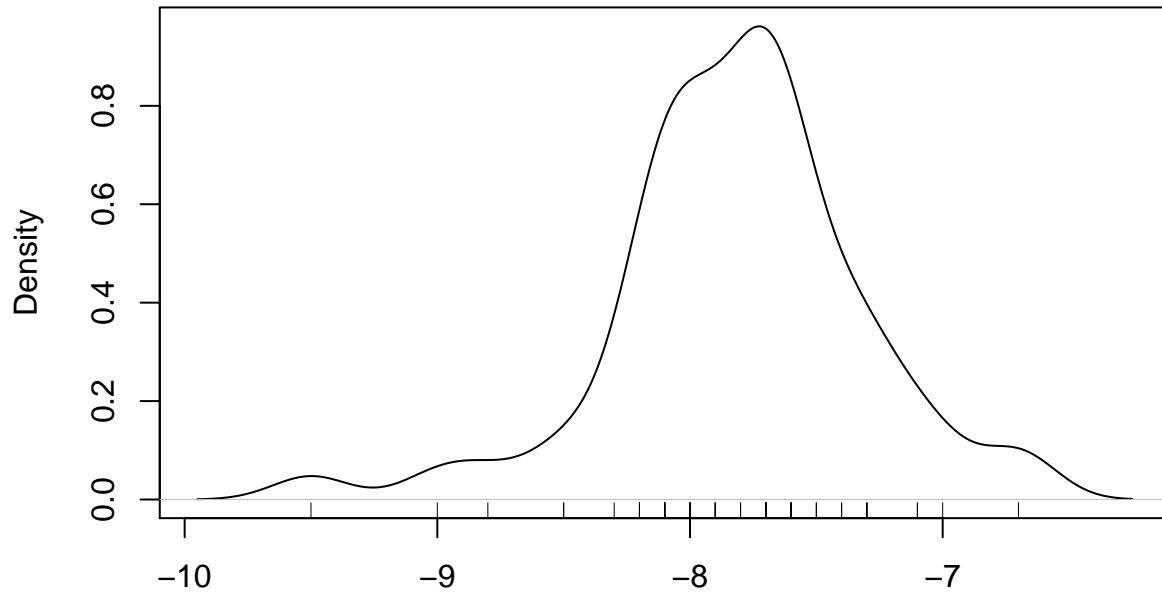
```
density.default(x = docking[, i], na.rm = T)
```



N = 56 Bandwidth = 0.4579

```
## debug en <text>#4: rug(docking[, i]) ## debug en <text>#4: browser() ## debug en
<text>#4: plot(density(docking[, i], na.rm = T))
```

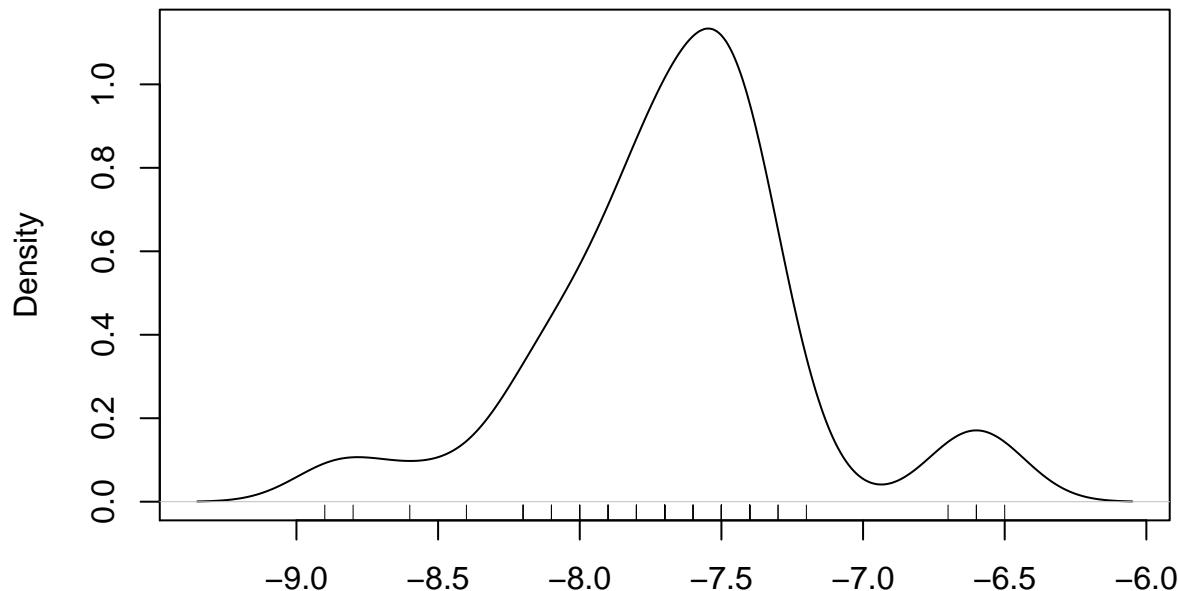
```
density.default(x = docking[, i], na.rm = T)
```



N = 56 Bandwidth = 0.1501

```
## debug en <text>#4: rug(docking[, i]) ## debug en <text>#4: browser() ## debug en
<text>#4: plot(density(docking[, i], na.rm = T))
```

```
density.default(x = docking[, i], na.rm = T)
```



N = 56 Bandwidth = 0.1501

```
## debug en <text>#4: rug(docking[, i]) ## debug en <text>#4: browser()
r plot(cor(docking[-c(5,12,13,38,51),-1]))
```

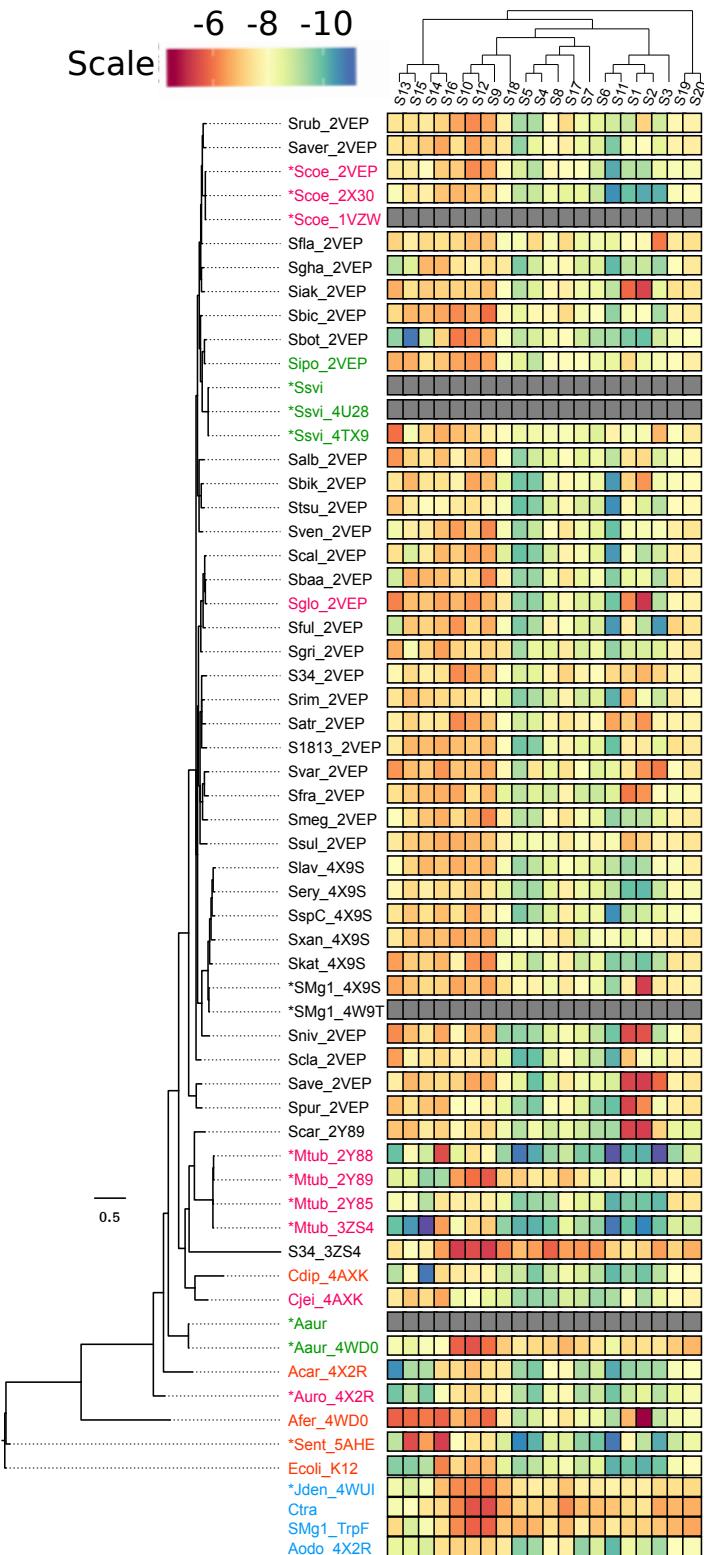



Figure 15:
30

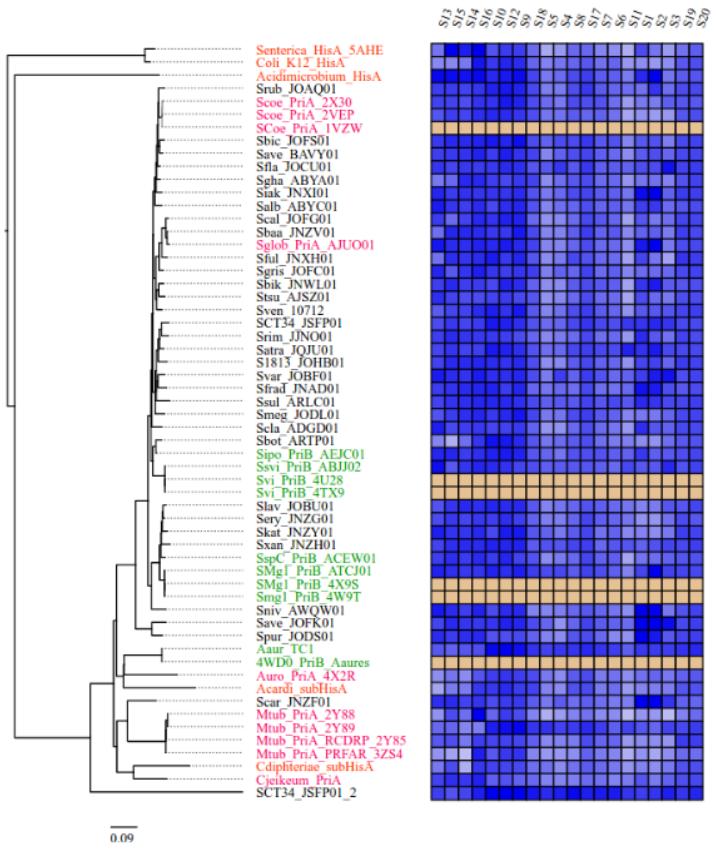
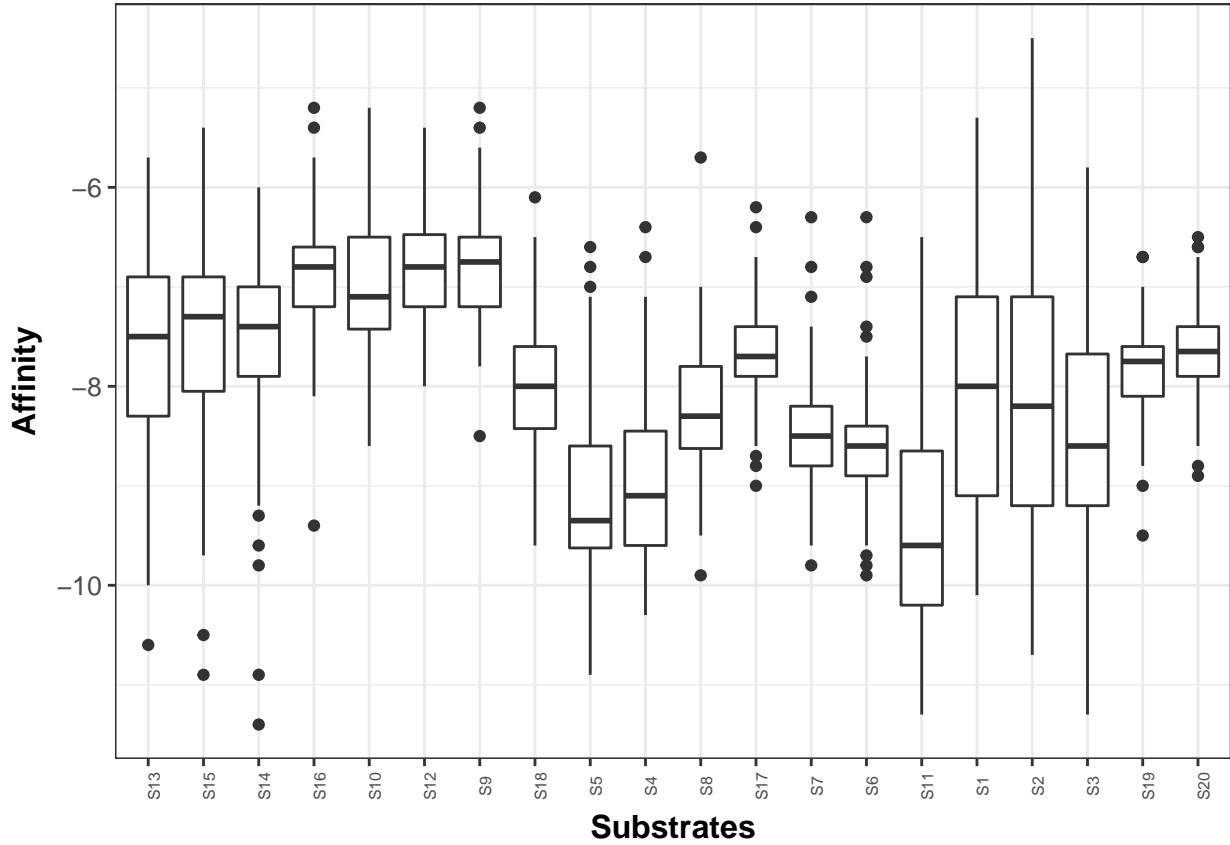


Figure 16:

GTP es el sustrato por el que PriA muestra mayor afinidad

```
## boxplot de los sustratos
ggplot(docking.m, aes(x=variable, y=value)) + labs(x = "Substrates", y = "Affinity",text = element_text)
## Warning: Removed 100 rows containing non-finite values (stat_boxplot).
```




```
#sessionInfo()
```

PriA en cinéticas enzimáticas no tradicionales.

Además de la exploración genómica de PriA se realizaron caracterizaciones experimentales. *i)* Se realizaron cinéticas de PriA en GTP . *ii)* Se avanzó en medir simultáneamente la actividad de PriA sobre ProFAR y PRA.

Es posible que PriA tenga actividad en GTP

Activity was measured fluorometrically in 96-well plates (Nuc 96-Well Optical Botto Plates) in a TECAN infinite M1000 plate reader (excitation at 286 nm and emission at 386 nm)

Preliminary activity assays were performed on an active PriA from *Streptomyces coelicolor* and an inactive mutant D11A.

Enzymes were cloned on coli V68 strains, overexpression were induced and protein were purified.

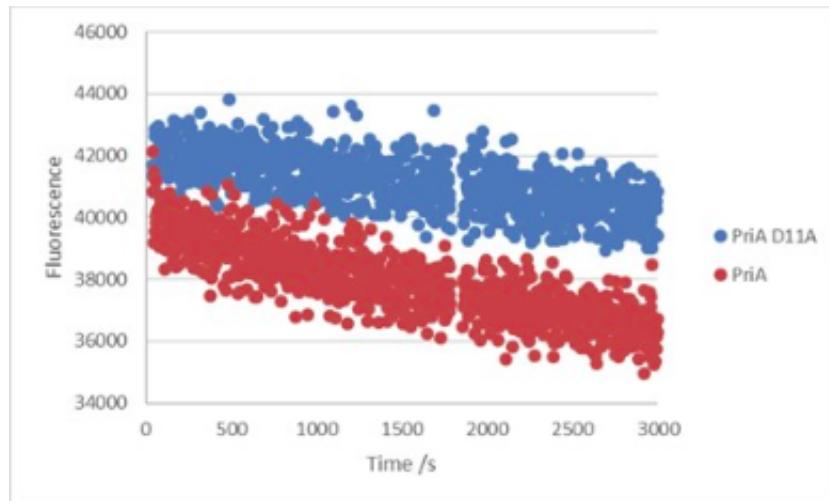


Figure 17:

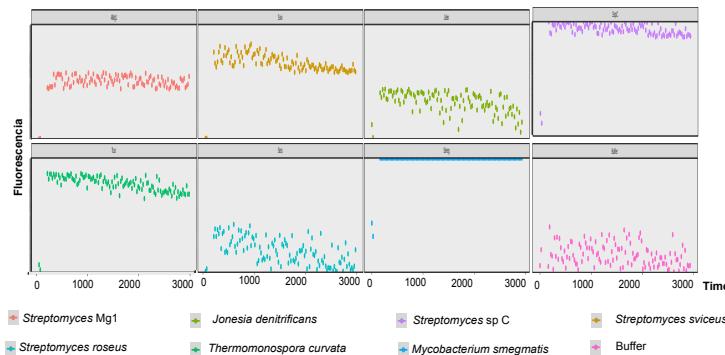


Figure 18:

Enzima	S13	S15	S14	S16	S10	S12	S9	S18	S5	S4	S8	S17	S7	S6	S11	S1
--------	-----	-----	-----	-----	-----	-----	----	-----	----	----	----	-----	----	----	-----	----



Figure 19:

Cinéticas simultáneas

tle: "TrpF_kinetics"
 thor: "NellySelem"
 te: "May 18, 2017"
 tput: pdf_document

```
#First I saved file as scv tablar separated
#perl -p -i -e 's/,/\t/g' Pra_scoe1.csv
#An cut it to obtain just data
#tail -n +39 Pra_scoe1.csv | head -n-3 > Pra_scoe1.data
#perl -p -i -e 's/^\sNr\.\./\s\[.\w*\]\//\.\s\[.*\]\//g' Pra_scoe1.data


```

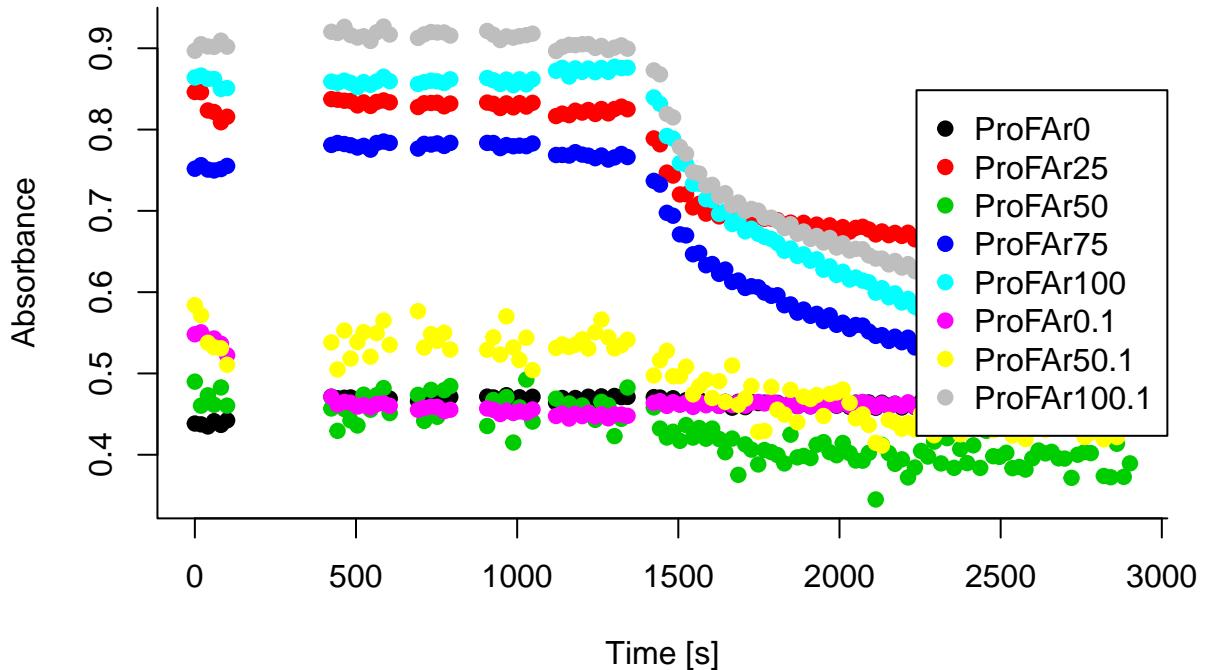
```

## [1] 0.00 19.15 40.50 59.65 81.00 100.15 423.50 442.65
## [9] 464.00 483.15 504.50 523.65 545.00 564.15 585.50 604.65
## [17] 692.40 711.55 732.90 752.05 773.40 792.55 907.00 926.15
## [25] 947.50 966.65 988.00 1007.15 1028.50 1047.65 1120.70 1139.85
## [33] 1161.20 1180.35 1201.70 1220.85 1242.20 1261.35 NA NA
## [41] 1282.80 1301.95 1323.30 1342.45 1424.00 1443.15 1464.50 1483.65
## [49] 1505.00 1524.15 1545.50 1564.65 1586.00 1605.15 1626.50 1645.65
## [57] 1667.00 1686.15 1707.50 1726.65 1748.00 1767.15 1788.50 1807.65
## [65] 1829.00 1848.15 1869.50 1888.65 1910.00 1929.15 1950.50 1969.65
## [73] 1991.00 2010.15 2031.40 2050.55 2071.90 2091.05 2112.50 2131.65
## [81] 2153.00 2172.15 2193.50 2212.65 2234.00 2253.15 2274.50 2293.65
## [89] 2315.00 2334.15 2355.50 2374.65 2396.00 2415.15 2436.50 2455.65
## [97] 2477.00 2496.15 2517.50 2536.65 2558.00 2577.15 2598.50 2617.65
## [105] 2639.00 2658.15 2679.50 2698.65 2720.00 2739.15 2760.50 2779.65
## [113] 2801.00 2820.15 2841.50 2860.65 2882.00 2901.15

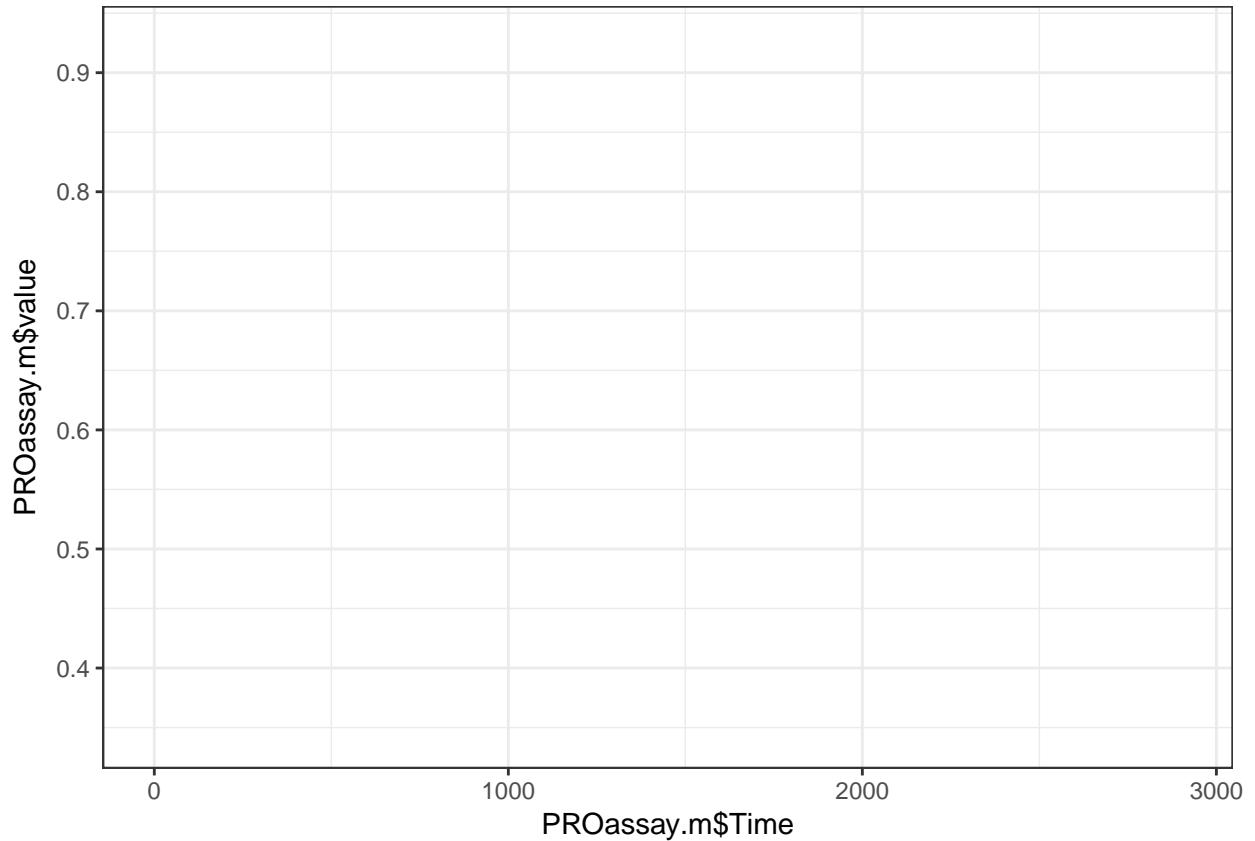
tablePRO<-tablePRO[grep("Pro|Time", colnames(tablePRO))]
#tablePRA<-tablePRA[grep("Pra/Time", colnames(tablePRA))]
```

```
PROassay.m <- melt(tablePRO,id="Time")
```

```
plot(PROassay.m$Time, PROassay.m$value, col=PROassay.m$variable, xlab="Time [s]",ylab="Absorbance", pch=19, bg="white", cex=0.8)
par(xpd = TRUE)
legend("right", legend = (unique(PROassay.m$variable)), col = (unique(PROassay.m$variable)),pch=19,bg="white")
```



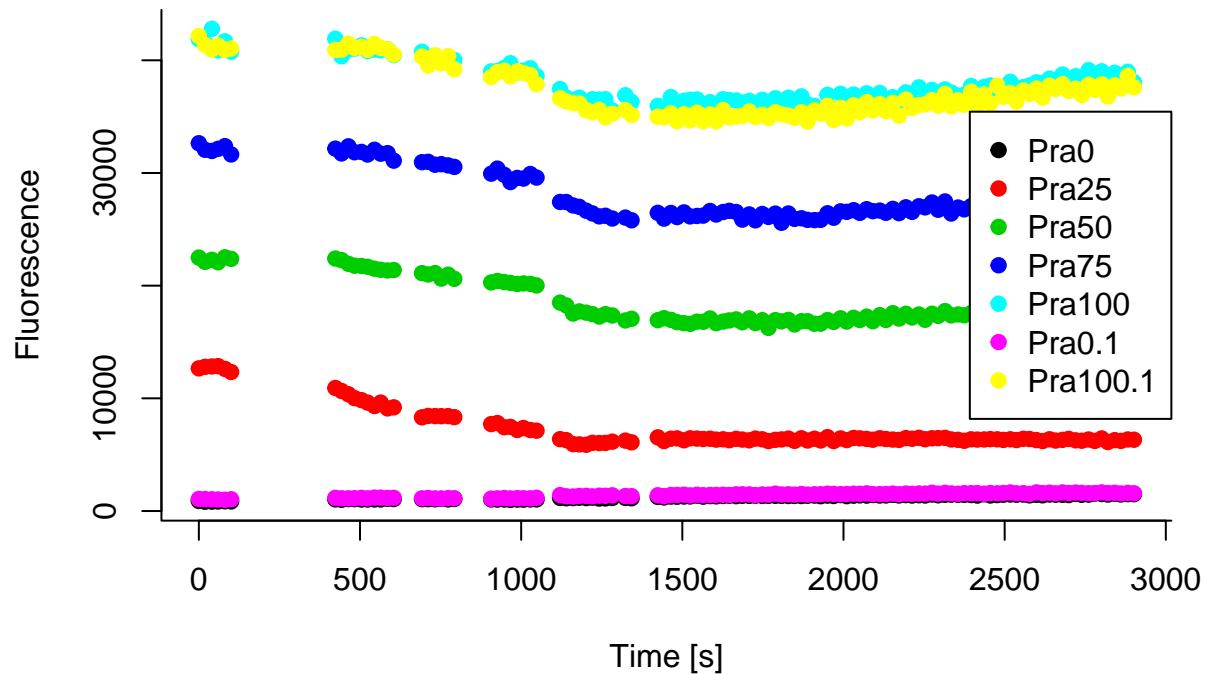
```
ggplot(PROassay.m, aes(x = PROassay.m$Time, y = PROassay.m$value),color="variable") + theme_bw()
```



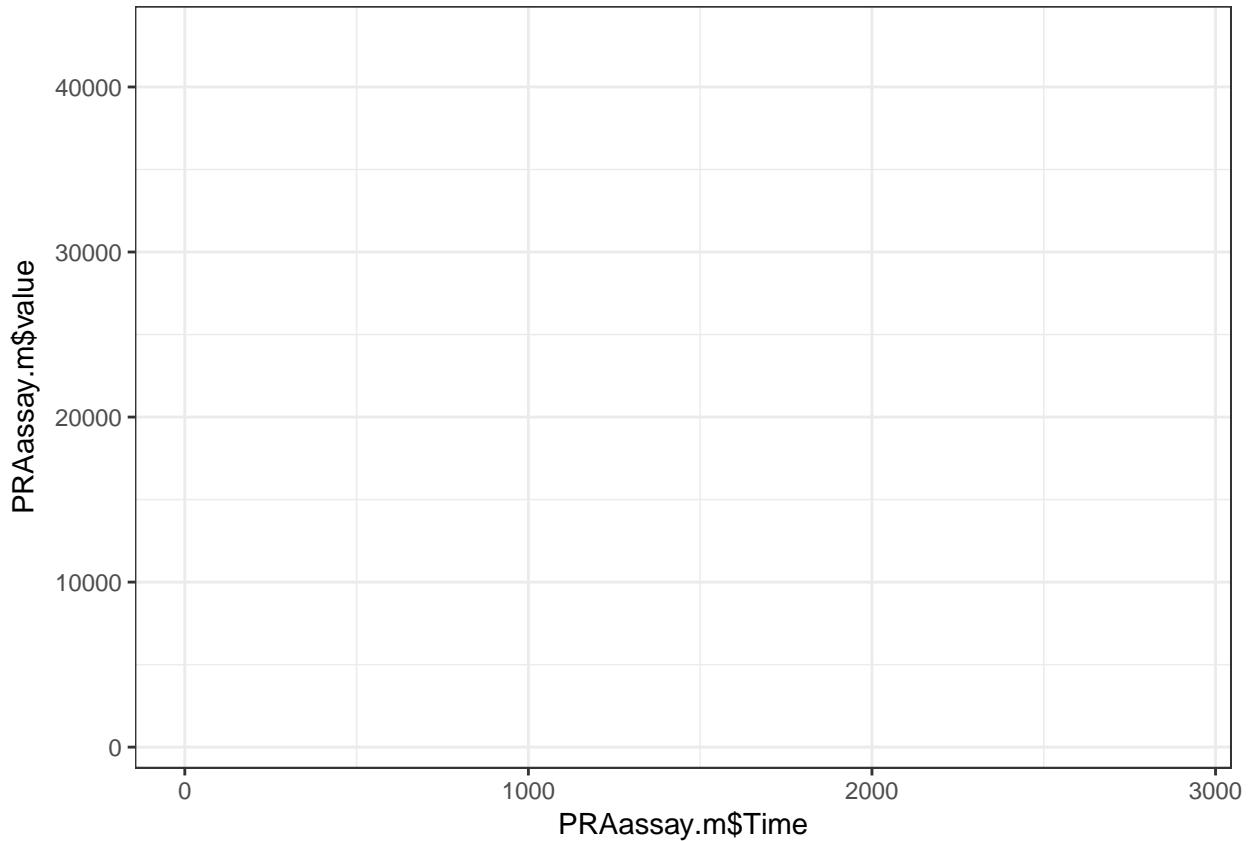
```
qplot(Time,value, data = PROassay.m, colour=variable)+theme_bw() + theme(legend.position  ="bottom", legend.title=element_text(size=10))  
## Warning: Removed 16 rows containing missing values (geom_point).
```



```
PRAassay.m <- melt(tablePRA,id="Time")
plot(PRAassay.m$Time, PRAassay.m$value, col=PRAassay.m$variable, xlab="Time [s]",ylab="Fluorescence",
par(xpd = TRUE)
legend("right", legend = (unique(PRAassay.m$variable)), col = (unique(PRAassay.m$variable)),pch=19,bg="white")
```

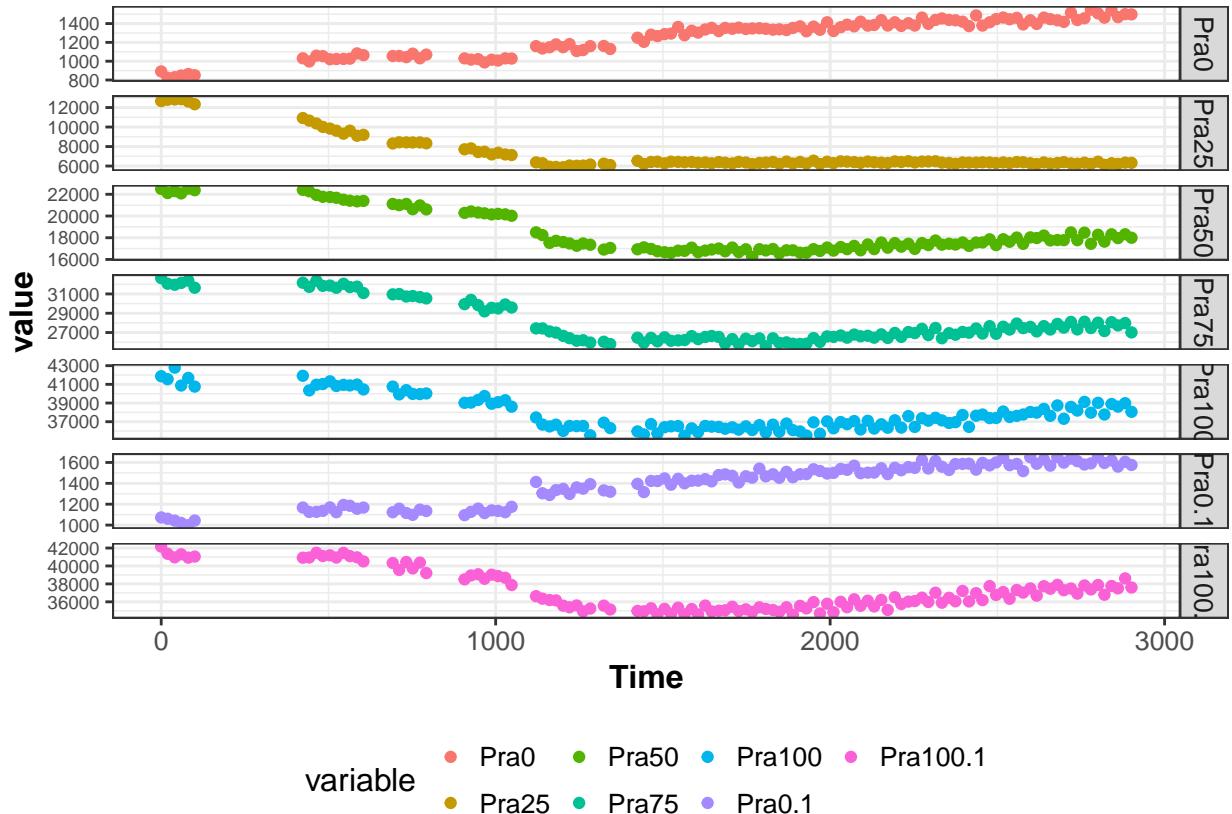


```
ggplot(PRAassay.m, aes(x = PRAassay.m$Time, y = PRAassay.m$value),color="variable") + theme_bw()
```



```
qplot(Time,value, data = PRAassay.m, colour=variable) + theme_bw() + theme(legend.position = "bottom", legend
```

Warning: Removed 7 rows containing missing values (geom_point).



```

##pendientes (slopes) of linear part of the curve
# time when TrpF is added
cuttime<-344.2
maxtime=620

PRAassay2Time<-tablePRA[which(tablePRA$Time >= cuttime & tablePRA$Time <= maxtime),]

## removi columnas que se salian del margen de medicion
#PRAassay2Time<-PRAassay2Time[, !(colnames(PRAassay2Time) %in% c("C62uM"))]

# vector with slopes for each dataset
V0 <- apply(PRAassay2Time, 2, function(x) coefficients(lm(x ~ PRAassay2Time$Time, na.action=na.omit)))[2]

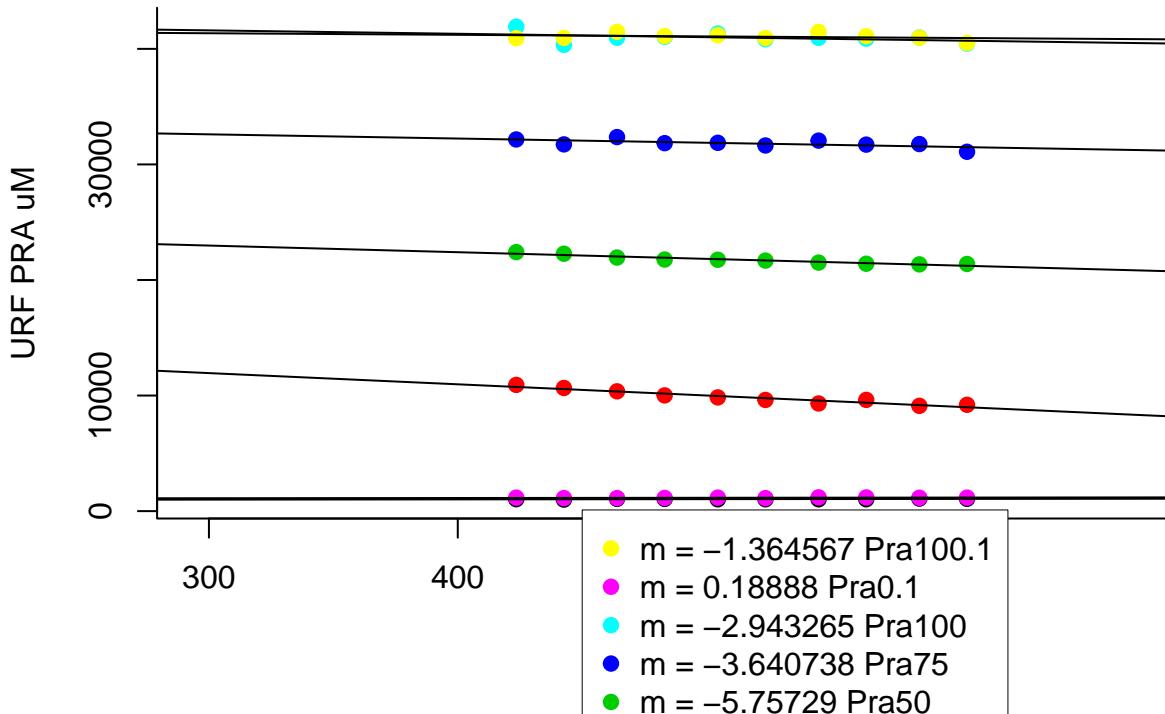
PRAassay2Time.m <- melt(PRAassay2Time,id="Time")
plot(PRAassay2Time.m$Time, PRAassay2Time.m$value, col=PRAassay2Time.m$variable,xlab="Time", ylab="URF P")

V1 <- apply(PRAassay2Time[,2:ncol(PRAassay2Time)], 2, function(x) coefficients(lm(x ~ PRAassay2Time$Time, na.action=na.omit)))[2]
apply(V1, 2, function(x) {abline(x, col = PRAassay2Time.m$variable)})

## NULL

par(xpd = TRUE)
slopeText2="m ="
slopeText<-paste(slopeText2,rev(round(V0[-1],digits=6)),rev(names(V0[-1])))
legend(450,100, legend = slopeText, col = rev(unique(PRAassay2Time.m$variable)), pch=19,box.lwd=0, bg="white")

```



```

## get relevant velocities and get correspondent concentrations
## V0 or slopes are on [PRA]uM/s
# slopes stores the slope, with TRUE , FALSE the desired enzyme can be selected, One more than one is t
slopes=(round(V0[-1],digits=6))[c(TRUE)]
concentration<-as.matrix(PRAassay2Time[1,names(slopes)])
#slopes
#concentration
slopes=as.matrix((round(V0[-1],digits=6))[c(TRUE)])
row.names(slopes) <- NULL
slopes<-c(slopes)# plot on r base

slopes

## [1] 0.196328 -9.695510 -5.757290 -3.640738 -2.943265 0.188880 -1.364567
# in concentration I stored the initial substrate concentration
row.names(concentration) <- NULL
concentration<-c(concentration)
slopes

## [1] 0.196328 -9.695510 -5.757290 -3.640738 -2.943265 0.188880 -1.364567
concentration

## [1] 1031 10920 22410 32161 41918 1168 40922
## interpolar michaelis-menden

https://rpubs.com/RomanL/6752
https://davetang.org/muse/2013/05/17/fitting-a-michaelis-mentens-curve-using/
#S <-concentration
## from highest to lower concentration
S_ProFAR <-c(0,25,50,75,100,0,50,100)
v<-slopes

```

```

#v<-slopes[-length(slopes)]
#v <-slopes[c(TRUE, FALSE)]
v<--1*v
S_ProFAR
v
mm <- data.frame(S_ProFAR,v)
model.drm <- drm(v ~ S, data = mm, fct = MM.2())
summary(model.drm)
## first value equal km
## second value = vm
Km=coefficients(model.drm) [1]
Vmax=2*coefficients(model.drm) [2]
Enzyme=2.5 #2.5uM
Kcat=Vmax*Enzyme
Km
Vmax
Kcat
mml <- data.frame(S = seq(0, max(mm$S), length.out = 100))
mml$v <- predict(model.drm, newdata = mml)

## plot on r base
plot(mm,log=' ',xlim=c(0,max(mm$S)), ylim=c(0,max(mm$v)), xlab="Reads", ylab="Transcripts")

##plot on ggplot
ggplot(mm, aes(x = S, y = v)) + theme_bw() + xlab("Concentration [uM]") + ylab("Speed [d[PRA]uM/s]")
ggsave("mm.pdf", width = 6, height = 4)

```