My Thesis or Dissertation Title

A Thesis Presented for the

Doctor of Philosophy

Degree

The University of Tennessee, Knoxville

Cedric Landerer

December 2018

To my mother

Acknowledgments

First and foremost I want to thank my Adviser, Michael Gilchrist for his long lasting patience, his availability and his teachings; always sharpening my focus and providing a new angle to a problem.

The faculty and students in EEB allowing me to broaden knowledge and insights with always stimulating discussions.

Abstract

Abstract text goes here...

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Chapter 2

AnaCoDa: Analyzing Codon Data with Bayesian mixture models

This chapter is a lightly revised version of a paper by the same name published in Bioinformatics and co-authored with Alexander Cope, Russell Zaretzki, and Michael A. Gilchrist.

C. Landerer, A. Cope, R. Zaretzki, M.A. Gilchrist, AnaCoDa: analyzing codon data with Bayesian mixture models, Bioinformatics, 34, 2018, 2496-2498

Abstract

AnaCoDa is an R package for estimating biologically relevant parameters of mixture models, such as selection against translation inefficiency, nonsense error rate, and ribosome pausing time, from genomic and high throughput datasets. **AnaCoDa** provides an adaptive Bayesian MCMC algorithm, fully implemented in C++ for high performance with an ergonomic R interface to improve usability. **AnaCoDa** employs a generic object-oriented design to allow users to extend the framework and implement their own models. Current models implemented in **AnaCoDa** can accurately estimate biologically relevant parameters given either protein coding sequences or ribosome foot-printing data. Optionally, **AnaCoDa** can utilize additional data sources, such as gene expression measurements, to aid model fitting and parameter estimation. By utilizing a hierarchical object structure, some parameters can vary between sets of genes while others can be shared. Genes may be assigned to clusters or membership may be estimated by **AnaCoDa**. This flexibility allows users to estimate the same model parameter under different biological conditions and categorize genes into different sets based on shared model properties embedded within the data. **AnaCoDa** also allows users to generate simulated data which can be used to aid model development and model analysis as well as evaluate model adequacy. Finally, **AnaCoDa** contains a set of visualization routines and the ability to revisit or re-initiate previous model fitting, providing researchers with a well rounded easy to use framework to analyze genome scale data.

Availability:

AnaCoDa is freely available under the Mozilla Public License 2.0 on CRAN (https://cran.r-project.org/web/packages/AnaCoDa/).

2.1 Introduction

AnaCoDa is an open-source software implemented in R [**?**] that allows researchers to analyze genome-scale data like coding sequences and ribosome footprinting data using evolutionary or analytical models in a Bayesian framework. **AnaCoDa** was developed to analyze selection on synonymous codon usage in the form of ribosome overhead cost ([**?** **?** **?**]). However, other codon metrics like the codon adaptation index [**?**] or the effective number of codons [**?**] are also provided as reference. In addition, three currently unpublished models to analyze coding sequences for evidence of selection against nonsense errors and estimate ribosome pausing times from ribosome footprinting data are included. **AnaCoDa** implements an adaptive Gibbs sampler within a Metropolis-Hastings Monte Carlo Markov Chain (MCMC). This allows for the incorporation of prior knowledge such as observed gene expression levels and easy sampling from the posterior distribution to estimate parameter values and quantify degree of uncertainty. **AnaCoDa** provides a mixture distribution option to all implemented models, combining genes into sets by estimating the posterior probabilities of set membership based on gene-set specific parameters shared by all genes assigned to a given set. **AnaCoDa** provides a generic, mixture distribution option to all implemented models, allowing for the estimation of condition specific parameters or the automatic categorization of data into different sets based on differences in their posterior probabilities of set membership. In addition to the four models provided, **AnaCoDa** provides a modular infrastructure such that additional genome scale or even phylogenetic models can be integrated.

The **AnaCoDa** framework works with **AnaCoDa** requires gene specific data such as codon frequencies obtained from coding sequences or position specific footprint counts. Conceptually, **AnaCoDa** allows for three different types of parameters. The first type are gene specific parameters such as protein synthesis rate or relative functionality. The second type are gene-set specific parameters, such as mutation bias terms or translation error rates. These parameters are shared across genes within a set and can be exclusive to a single set or shared with other sets. While the number of gene sets must be pre-defined by the user, set assignment of genes can be pre-defined or estimated as part of the model fitting. Estimation

of the set assignment provides the probability of a gene being assigned to a set allowing the user to asses the uncertainty in each assignment. The third type are hyperparameters allowing for the construction and analysis of hierarchical model. Hyperparameters control the prior distribution for gene and gene-set specific parameters such as mutation bias or protein synthesis rate.

2.2 Features

AnaCoDa provides an interface written in R, a freely available programming language noted for its ease of use for even inexperienced programmers. As a result, **AnaCoDa** is accessible to researchers with minimal computational experience.

The interface of **AnaCoDa** is designed for quick and efficient data analysis. Generally, the only input needed for fitting a model to the data are protein-coding codon sequences in the form of a FASTA file or a flat-file containing codon counts obtained from ribosome foot-printing experiments. **AnaCoDa** also provides visualization functionality, including plotting functions to compare parameter estimates for different mixture distributions and display codon usage patterns. In addition, diagnostic functions such as those for calculating and visualizing the degree of autocorrelation in the parameter tracess are provided.

Robust and efficient model fitting

AnaCoDa has built-in features designed to improve the robustness and performance of the implemented MCMC approach. For example, the implemented MCMC automatically adapts the proposal width for sampled parameters such that an user defined acceptance range is met, improving sampling efficiency of the MCMC and computational performance. Even though **AnaCoDa** is written in C++, analysis of large datasets and/or complex models can be very computationally intensive. To protect users from computer failures or aid in the collection of additional MCMC samples, **AnaCoDa** can periodically produce output checkpoint files, which can be used to restart an MCMC chain from a previous time point. In addition, **AnaCoDa** automatically thins all parameter traces - meaning only every k^{th} sample is kept - increasing the effective number of samples and reducing its memory foorprint.

Although **AnaCoDa** is provided as an R package, the main computational work is implemented in C++. Because R does not provide native C++ support, Rcpp was employed to expose whole C++ classes as modules to R [**?**]. Using Rcpp eliminates time consuming data transfers between the R environment and the C++ core during model fitting, resulting in improved computational performance and allows for a fully object-oriented code design [**?**]. As expected, the runtime of **AnaCoDa** scales linearly with genome size and number of iterations, and scales polynomially with the number of mixture distributions in the data set. The polynomial increase in runtime with the number of mixture distributions is due to the necessity to condition the gene assignment on the estimation of gene specific parameters, such as, protein synthesis rate.

Data Simulation

In addition to fitting the models to datasets, **AnaCoDa** can be used to generate simulated data sets as well. On their own, simulated datasets are useful for model development and analysis. Simulating data under different conditions allows the user to explore model behavior and explore theoretical scenarios. Different conditions can include the addition or elimination of parameters, or simply allowing a set of parameter values to vary. Fitting models to simulated data can provide insight into potential pitfalls or shortcomings when fitting observational data and can serve as the basis for evaluating model adequacy of a model fit to observational data [**?**]. Significant differences between simulated and observational data suggests the current set of parameters or the model as a whole fail to include or adequately represent biological mechanisms underlying the observed data.

Available models

AnaCoDa currently provides codon models for analyzing genome scale data. The ROC model implements and extends the codon usage bias (CUB) models developed by **? ? ?**], which can reliably estimate the strength of selection on ribosome overhead cost, mutation bias and allows for the inference of protein synthesis rates. This model allows for the separation of effects of mutation and selection based on gene ordering by protein synthesis rate, and the addition of a mixture distribution allows for gene clustering based on mutation bias and

selection for translation efficiency. In addition to identifying the most efficient codons, ROC provides estimates of mutation bias allowing the approximation of mutation ratios between codons [**?** **?**].

The ability to estimate protein synthesis rates in the absence of empirical data is useful for investigating CUB of non-model organisms for which such data is lacking and enables the usage of protein synthesis rate in comparative frameworks or other analyses requiring protein synthesis rate information [**?**]. Use of the mixture model allows for the investigation of CUB heterogeneity at the genome or gene level. Following the same framework, additional models included in **AnaCoDa** provide estimates of codon-specific nonsense errors rates (FONSE) and ribosome pausing times (PA and PANSE).

Parameters estimated with the evolutionary models ROC and FONSE represent evolutionary averages and do not depend on experimental conditions. In contrast, PA and PANSE estimate the distribution of biologically relevant parameters like ribosome pausing times along a gene from experimental data such as ribosome footprinting data. The distribution can be dependent (PANSE) or independent (PA) of evidence for nonsense errors in the data.

2.3 Supplementary Material

AnaCoDa allows for the estimation of biologically relevant parameters like mutation bias or ribosome pausing time, depending on the model employed. Bayesian estimation of parameters is performed using an adaptive Metropolis-Hasting within Gibbs sampling approach. Models implemented in AnaCoDa are currently able to handle gene coding sequences and ribosome footprinting data.

2.3.1 The AnaCoDa framework

The AnaCoDa framework works with gene specific data such as codon frequencies or position specific footprint counts. Conceptually, AnaCoDa uses three different types of parameters.

- The first type of parameters are **gene specific parameters** such as gene expression level or functionality. Gene-specific parameters are estimated separately for each gene and can vary between potential gene categories or sets.

- The second type of parameters are **gene-set specific parameters**, such as mutation bias terms or translation error rates. These parameters are shared across genes within a set and can be exclusive to a single set or shared with other sets. While the number of gene sets must be pre-defined by the user, set assignment of genes can be pre-defined or estimated as part of the model fitting. Estimation of the set assignment provides the probability of a gene being assigned to a set allowing the user to asses the uncertainty in each assignment.

- The third type of parameters are **hyperparameters**, such as parameters controlling the prior distribution for mutation bias or error rate. Hyperparameters can be set specific or shared across multiple sets and allow for the construction and analysis of hierarchical models, by controlling prior distributions for gene or gene-set specific parameters.

Analyzing protein coding gene sequences

AnaCoDa always requires the following four objects:

- **Genome** contains the codon data read from a fasta file as well as empirical protein synthesis rate in the form of a comma separated (.csv) ID/Value pairs.

- **Parameter** represents the parameter set (including parameter traces) for a given genome. The parameter object also hold the mapping of parameters to specified sets.

- **Model** allows you to specify which model should be applied to the genome and the parameter object.

- **MCMC** specifies how many samples from the posterior distribution of the specified model should be stored to obtain parameter estimates.

2.3.2 AnaCoDa setup

Application of codon model to single genome

In this example we are assuming a genome with only one set of gene-set specific parameters, hence **num.mixtures** = 1. We assign all genes the same gene-set, and provide an initial value for the hyperparameter sphi (s_ϕ). s_ϕ controls the lognormal prior distribution on the gene specific parameters like the protein synthesis rate ϕ. To ensure identifiability the expected value of the prior distribution is assumed to be 1.

$$E[\phi] = \exp\left(m_\phi + \frac{s_\phi^2}{2}\right) = 1 \tag{2.1}$$

Therfore the mean m_ϕ is set to be $-\frac{s_\phi^2}{2}$. For more details see **?**].

After choosing the model and specifying the necessary arguments for the MCMC routine, the MCMC is run

```
genome <- initializeGenomeObject(file = "genome.fasta")
parameter <- initializeParameterObject(genome = genome, sphi = 1,
                        num.mixtures = 1,
                        gene.assignment = rep(1, length(genome)))
model <- initializeModelObject(parameter = parameter, model = "ROC")
mcmc <- initializeMCMCObject(samples = 5000, thinning = 10,
                        adaptive.width=50)
runMCMC(mcmc = mcmc, genome = genome, model = model)
```

runMCMC does not return a value, the results of the MCMC are stored automatically in the mcmc and parameter objects created earlier.

> **Please note that AnaCoDa utilizes C++ obejct orientation and therefore employs pointer structures. This means that no return value is necessary for such objects as they are modified within the the runMCMC routine. You will find that after a completed run, the parameter object will contain all necessary information without being directly passed into the MCMC routine. This might be confusing at first as it is not default R behaviour.**

Application of codon model to a mixture of genomes

This case applies if we assume that parts of the genome differ in their gene-set specific parameters. This could be due to introgression events or strand specific mutation difference, horizontal gene transfers or other reasons. We make the assumption that all sets of genes are independent of one another. For two sets of gene-set specific parameter with a random gene assignment we can use:

```
parameter <- initializeParameterObject(genome = genome,
                          sphi = c(0.5, 2), num.mixtures = 2,
                          gene.assignment = sample.int(2,
                                  length(genome), replace = T))
gene.assignment = sample.int(2, length(genome), replace = T))
```

To accommodate for this mixing we only have to adjust **sphi**, which is now a vector of length 2, **num.mixtures**, and **gene.assignment**, which is chosen at random here.

Empirical protein synthesis rate values

To use empirical values as prior information one can simply specify an observed.expression.file when initializing the genome object.

```
genome <- initializeGenomeObject(file = "genome.fasta",
                observed.expression.file = "synthesis_values.csv")
```

These observed expression or synthesis values (Φ) are independent of the number of gene-sets. The error in the observed Φ values is estimated and described by sepsilon (s_ϵ). The csv file can contain multiple observation sets separated by comma. For each set of observations an initial s_ϵ has to be specified.

```
# One case of observed data
sepsilon <- 0.1
# Two cases of observed data
sepsilon <- c(0.1, 0.5)
# ...
# Five cases of observed data
```

```
sepsilon <- c(0.1, 0.5, 1, 0.8, 3)
parameter <- initializeParameterObject(genome = genome, sphi = 1,
                        num.mixtures = 1,
                        gene.assignment = rep(1, length(genome)),
                        init.sepsilon = sepsilon)
```

In addition one can choose to keep the noise in the observations (s_ϵ) constant by using the **fix.observation.noise** flag in the model object.

```
model <- initializeModelObject(parameter = parameter, model = "ROC",
fix.observation.noise = TRUE)
```

Fixing parameter types

It can sometime be advantages to fix certain parameters, like the gene specific parameters. For example in cases where only few sequences are available but gene expression measurements are at hand we can fix the gene specific parameters to increase confidence in our estimates of gene-set specific parameters.

We again initialize the **genome**, **parameter**, and **model** objects.

```
genome <- initializeGenomeObject(file = "genome.fasta")
parameter <- initializeParameterObject(genome = genome, sphi = 1,
                        num.mixtures = 1,
                        gene.assignment = rep(1, length(genome)))
model <- initializeModelObject(parameter = parameter, model = "ROC")
```

To fix gene specific parameters we will set the **est.expression** flag to **FALSE**. This will estimate only gene-set specific parameters, hyperparameters, and the assignments of genes to various sets.

```
mcmc <- initializeMCMCObject(samples, thinning=1,
                        adaptive.width=100, est.expression=FALSE,
                        est.csp=TRUE, est.hyper=TRUE, est.mix=TRUE)
```

If we would like to fix gene-set specific parameters we instead disable the **est.csp** flag.

```
mcmc <- initializeMCMCObject(samples, thinning=1,
```

```
                        adaptive.width=100, est.expression=TRUE,
                        est.csp=FALSE, est.hyper=TRUE, est.mix=TRUE)
```

The same applies to the hyper parameters (**est.hyper**),

```
mcmc <- initializeMCMCObject(samples, thinning=1,
                        adaptive.width=100, est.expression=TRUE,
                        est.csp=TRUE, est.hyper=FALSE, est.mix=TRUE)
```

and gene set assignment (**est.mix**).

```
mcmc <- initializeMCMCObject(samples, thinning=1,
                        adaptive.width=100, est.expression=TRUE,
                        est.csp=TRUE, est.hyper=TRUE, est.mix=FALSE)
```

We can use these flags to fix parameters in any combination.

Combining various gene-set specific parameters to a gene-set description.

We distinguish between three simple cases of gene-set descriptions, and the ability to customize the parameter mapping. The specification is done when initializing the parameter object with the **mixture.definition** argument.

We encounter the simplest case when we assume that all gene sets are independent.

```
parameter <- initializeParameterObject(genome = genome,
                        sphi = c(0.5, 2), num.mixtures = 2,
                        gene.assignment = sample.int(2,
                                length(genome), replace = T),
                        mixture.definition = "allUnique")
```

The **allUnique** keyword allows each type of gene-set specific parameter to be estimated independent of parameters describing other gene sets.

In case we want to share mutation parameter between gene sets we can use the keyword **mutationShared**

```
parameter <- initializeParameterObject(genome = genome,
                        sphi = c(0.5, 2), num.mixtures = 2,
                        gene.assignment = sample.int(2,
```

```
                                    length(genome), replace = T),
                            mixture.definition = "mutationShared")
```

This will force all gene sets to share the same mutation parameters.

The same can be done with parameters describing selection, using the keyword **selectionShared**

```
parameter <- initializeParameterObject(genome = genome,
                            sphi = c(0.5, 2), num.mixtures = 2,
                            gene.assignment = sample.int(2,
                                    length(genome), replace = T),
                            mixture.definition = "selectionShared")
```

For more intricate compositions of gene sets, one can specify a custom $n \times 2$ matrix, where n is the number of gene sets, to describe how gene-set specific parameters should be shared. Instead of using the **mixture.definition** argument one uses the **mixture.definition.matrix** argument.

The matrix representation of **mutationShared** can be obtained by

```
# [,1] [,2]
#[1,] 1 1
#[2,] 1 2
#[3,] 1 3
def.matrix <- matrix(c(1,1,1,1,2,3), ncol=2)
parameter <- initializeParameterObject(genome = genome,
                            sphi = c(0.5, 2, 1), num.mixtures = 3,
                            gene.assignment = sample.int(3,
                                    length(genome), replace = T),
                            mixture.definition.matrix = def.matrix)
```

Columns represent mutation and selection, while each row represents a gene set. In this case we have three gene sets, each sharing the same mutation category and three different selection categories. In the same way one can produce the matrix for three independent gene sets equivalent to the **allUnique** keyword.

```
# [,1] [,2]
```

```
#[1,] 1 1
#[2,] 2 2
#[3,] 3 3
def.matrix <- matrix(c(1,2,3,1,2,3), ncol=2)
```

We can also use this matrix to produce more complex gene set compositions.

```
# [,1] [,2]
#[1,] 1 1
#[2,] 2 1
#[3,] 1 2
def.matrix <- matrix(c(1,2,1,1,1,2), ncol=2)
```

In this case gene set one and three share their mutation parameters, while gene set one and two share their selection parameters.

Checkpointing

AnaCoDa does provide checkpointing functionality in case runtime has to be restiricted. To enable checkpointing, one can use the function **setRestartSettings**.

```
# writing a restart file every 1000 samples
setRestartSettings(mcmc, "restart_file", 1000, write.multiple=TRUE)
# writing a restart file every 1000 samples
# but overwriting it every time
setRestartSettings(mcmc, "restart_file", 1000, write.multiple=FALSE)
```

To re-initialize a parameter object from a restart file one can simply pass the restart file to the initialization function:

```
initializeParameterObject(init.with.restart.file="restart_file.rst")
```

Load and save parameter objects

AnaCoDa is based on C++ objects using the Rcpp [**?**]. This comes with the problem that C++ objects are by default not serializable and can therefore not be saved/loaded with the default R save/load functions.

AnaCoDa however, does provide functions to load and save parameter and mcmc objects. These are the only two objects that store information during a run.

```
#save objects after a run
runMCMC(mcmc = mcmc, genome = genome, model = model)
writeParameterObject(parameter = parameter, file = "parameter.Rda")
writeMCMCObject(mcmc = mcmc, file = "mcmc_out.Rda")
```

As **genome**, and **model** objects are purely storage containers, no save/load function is provided at this point, but will be added in the future.

```
#load objects
parameter <- loadParameterObject(file = "parameter.Rda")
mcmc <- loadMCMCObject(file = "mcmc_out.Rda")
```

2.3.3 File formats

protein coding sequence

Protein coding sequences are provided by fasta file with the default format. One line containing the sequence id starting with > followed by the id and one or more lines containing the sequence. The sequences are expected to have a length that is a multiple of three. If a codon can not be recognized (e.g AGN) it is ignored.

```
>YAL001C
TTGGTTCTGACTCATTAGCCAGACGAACTGGTTCAA
CATGTTTCTGACATTCATTCTAACATTGGCATTCAT
ACTCTGAACCAACTGTAAGACCATTCTGGCATTTAG
>YAL002W
TTGGAACAAAACGGCCTGGACCACGACTCACGCTCT
TCACATGACACTACTCATAACGACACTCAAATTACT
TTCCTGGAATTCCGCTCTTAGACTCAACTGTCAGAA
```

Empirical gene expression

Empirical expression or gene specific parameters are provided in a csv file format. The first line is expected to be a header describing each column. The first column is expected be the gene id, and every additional column is expected to be represent a measurement. Each row corresponds to one gene and contains all measurements for that gene, including missing values.

```
>YAL001C
ORF,DATA_1,DATA_2,...DATA_N
YAL001C,0.254,0.489,...,0.156
YAL002W,1.856,1.357,...,2.014
YAL003W,10.45,NA,...,9.564
YAL005C,0.556,0.957,...,0.758
```

Ribosome foot-printing counts

Ribosome foot-printing (RFP) counts are provided in a csv file format. The first line is expected to be a header describing each column. The columns are expected in the following order gene id, position, codon, rfpcount. Each row corresponds to a single codon with an associated number of ribosome footprints.

```
GeneID,Position,Codon,rfpCount
YBR177C, 0, ATA, 8
YBR177C, 1, CGG, 1
YBR177C, 2, GTT, 8
YBR177C, 3, CGC, 1
```

2.3.4 Analyzing and Visualizating results

Parameter estimates

After we have completed the model fitting, we are interested in the results. AnaCoDa provides functions to obtain the posterior estimate for each parameter. For gene-set specific

parameters or codon specific parameters we can use the function **getCSPEstimates**. Again we can specify for which mixture we would like the posterior estimate and how many samples should be used. **getCSPEstimates** has an optional argument filename which will cause the routine to write the result as a csv file instead of returning a **data.frame**.

```
csp_mat <- getCSPEstimates(parameter = parameter, CSP="Mutation",
                           mixture = 1, samples = 1000)
head(csp_mat)
# AA Codon Posterior 0.025% 0.975%
#1 A GCA -0.2435340 -0.2720696 -0.2165220
#2 A GCC 0.4235546 0.4049132 0.4420680
#3 A GCG 0.7004484 0.6648690 0.7351707
#4 C TGC 0.2016298 0.1679025 0.2387024
#5 D GAC 0.5775052 0.5618199 0.5936979
#6 E GAA -0.4524295 -0.4688044 -0.4356677
getCSPEstimates(parameter = parameter, filename = "mutation.csv",
                           CSP="Mutation", mixture = 1, samples = 1000)
```

To obtain posterior estimates for the gene specific parameters, we can use the function **getExpressionEstimatesForMixture**. In the case below we ask to get the gene specific parameters for all genes, and under the assumption each gene is assigned to mixture 1.

```
phi_mat <- getExpressionEstimates(parameter = parameter,
                           gene.index = 1:length(genome),
                           samples = 1000)
head(phi_mat)
# PHI log10.PHI Std.Error log10.Std.Error 0.025 0.975 log10.025 ...
#[1,] 0.2729446 -0.6188447 0.0001261525 2.362358e-04 0.07331819 ...
#[2,] 1.4221716 0.1498953 0.0001669425 5.194123e-05 1.09593642 ...
#[3,] 0.7459888 -0.1512764 0.0002313539 1.529267e-04 0.31559618 ...
#[4,] 0.6573082 -0.2030291 0.0001935466 1.400333e-04 0.31591233 ...
#[5,] 1.6316901 0.2098120 0.0001846631 4.986347e-05 1.28410352 ...
#[6,] 0.6179711 -0.2286806 0.0001744928 1.374863e-04 0.28478950 ...
```

However we can decide to only obtain certain gene parameters. in the first case we sample 100 random genes.

```
# sampling 100 genes at random
phi_mat <- getExpressionEstimates(parameter = parameter,
                          gene.index = sample(1:length(genome), 100),
                          samples = 1000)
```

Furthermore, AnaCoDa allows to calculate the selection coefficient s for each codon and each gene. We can use the function **getSelectionCoefficients** to do so. Please note, that this function returns the $log(sNe)$.

getSelectionCoefficients returns a matrix with $log(sNe)$ relative to the most efficient synonymous codon.

```
selection.coefficients <- getSelectionCoefficients(genome = genome,
                          parameter = parameter, samples = 1000)
head(selection.coefficients)
# GCA GCC GCG GCT TGC TGT GAC GAT ...
#SAKL0A00132g -0.1630284 -0.008695144 -0.2097771 0 -0.1014373 ...
#SAKL0A00154g -0.8494558 -0.045305847 -1.0930388 0 -0.5285367 ...
#SAKL0A00176g -0.4455753 -0.023764823 -0.5733448 0 -0.2772397 ...
#SAKL0A00198g -0.3926068 -0.020939740 -0.5051875 0 -0.2442824 ...
#SAKL0A00220g -0.9746002 -0.051980440 -1.2540685 0 -0.6064022 ...
#SAKL0A00242g -0.3691110 -0.019686586 -0.4749542 0 -0.2296631 ...
```

We can compare these values to the weights from the codon adaptatoin index (CAI) citepSharp1987 or effective number of codons (N_c) [**?**] by using the functions **getCAIweights** and **getNcAA**.

```
cai.weights <- getCAIweights(referenceGenome = genome)
head(cai.weights)
# GCA GCC GCG GCT TGC TGT
#0.7251276 0.6282192 0.2497737 1.0000000 0.6222628 1.0000000
nc.per.aa <- getNcAA(genome = genome)
head(nc.per.aa)
```

```
# A C D E F G ...
#SAKL0A00132g 3.611111 1.000000 2.200000 2.142857 1.792453 ...
#SAKL0A00154g 1.843866 2.500000 2.035782 1.942505 1.986595 ...
#SAKL0A00176g 5.142857 NA 1.857143 1.652174 1.551724 3.122449 ...
#SAKL0A00198g 3.800000 NA 1.924779 1.913043 2.129032 4.136364 ...
#SAKL0A00220g 3.198529 1.666667 1.741573 1.756757 2.000000 ...
#SAKL0A00242g 4.500000 NA 2.095890 2.000000 1.408163 3.734043 ...
```

We can also compare the distribution of selection coefficients to the CAI values estimated from a reference set of genes.

```
selection.coefficients <- getSelectionCoefficients(genome = genome,
                          parameter = parameter, samples = 1000)
s <- exp(selection.coefficients)
cai.weights <- getCAIweights(referenceGenome = ref.genome)
codon.names <- colnames(s)
h <- hist(s[, 1], plot = F)
plot(NULL, NULL, axes = F, xlim = c(0,1),
                          ylim = range(c(0,h$counts)),
                          xlab = "s", ylab = "Frequency",
                          main = codon.names[1], cex.lab = 1.2)
lines(x = h$breaks, y = c(0,h$counts), type = "S", lwd=2)
abline(v = cai.weights[1], lwd=2, lty=2)
axis(1, lwd = 3, cex.axis = 1.2)
axis(2, lwd = 3, cex.axis = 1.2)
```

Diagnostic Plots

A first step after every run should be to determine if the sampling routine has converged. To do that, AnaCoDa provides plotting routines to visualize all sampled parameter traces from which the posterior sample is obtained. First we have to obtain the **trace** object stored within our **parameter** object. Now we can simply plot the **trace** object. The argument **what** specifies which type of parameter should be plotted. Here we plot the

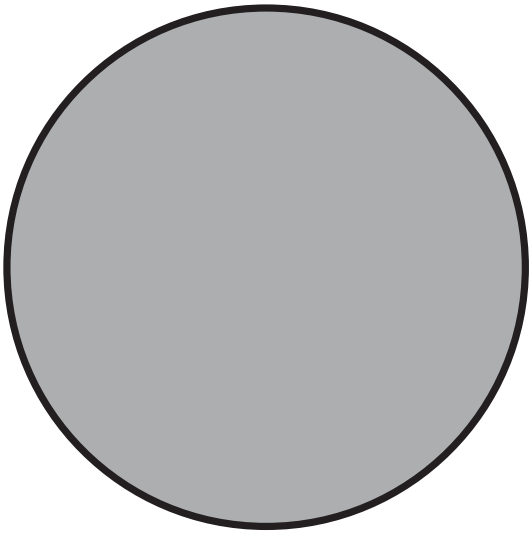

Figure 2.1: Distribution of s for codon GCA for amino acid alanine. Dashed line indicates the CAI weight for GCA. The comparisson provides a more nuanced picture as we can see that the selection on GCA varies across the genome.

selection parameter $\Delta\eta$ of the ROC model. These parameters are mixture specific and one can decide which mixture set to visualize using the argument **mixture**.

```
trace <- getTrace(parameter)
plot(x = trace, what = "Mutation", mixture = 1)
```

A special case is the plotting of traces of the protein synthesis rate ϕ. As the number of traces for the different ϕ traces is usually in the thousands, a **geneIndex** has to be passed to determine for which gene the trace should be plotted. This allows to inspect the trace of every gene under every mixture assignment.

```
trace <- parameter$getTraceObject()
plot(x = trace, what = "Expression", mixture = 1, geneIndex = 669)
```

We can find the likelihood and posterior trace of the model fit in the **mcmc object**. The trace can be plotted by just passing the **mcmc** object to the **plot** routine. Again we can switch between log(*likelihood*) and log(*posterior*) using the argument **what**. The argument **zoom.window** is used to inspect a specified window in more detail. It defaults to the last 10 % of the trace. The log(*posterior*) displayed in the figure title is estimated over the **zoom.window**.

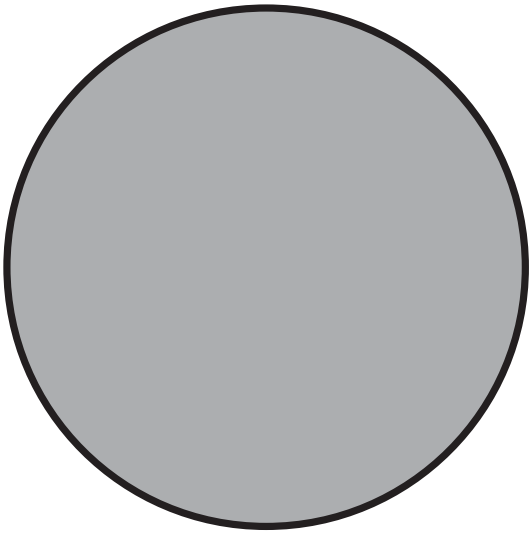

Figure 2.2: Trace plot showing the traces of all 40 codon specific selection parameters organized by amino acid.

```
plot(mcmc, what = "LogPosterior", zoom.window = c(9000, 10000))
```

Model visualization

We can visualize the results of the model fit by plotting the **model** object. For this we require the model and the **genome** object. We can adjust which mixture set we would like to visualize and how many samples should be used to obtain the posterior estimate for each parameter. For more details see **?**].

```
# use the last 500 samples from mixture 1 for posterior estimate.
plot(x = model, genome = genome, samples = 500, mixture = 1)
```

As AnaCoDa is designed with the idea to allow gene-sets to have independent gene-set specific parameters, AnaCoDa also provides the option to compare different gene-sets by plotting the parameter object. Here we compare the selection parameter estimated by ROC for seven yeast species.

```
# use the last 500 samples from mixture 1 for posterior estimate.
plot(parameter, what = "Selection", samples = 500)
```

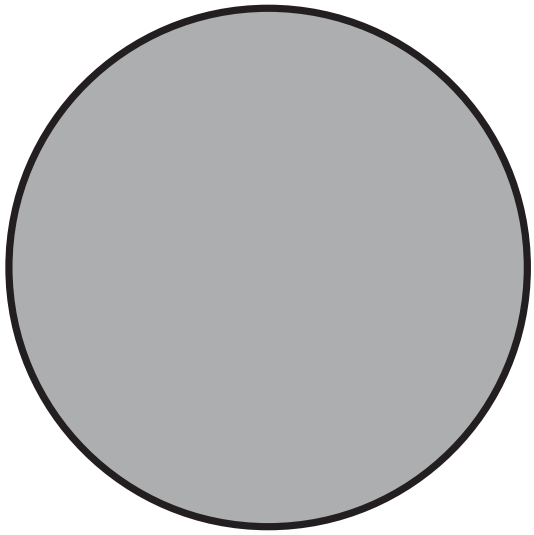

Figure 2.3: Trace plot showing the protein synthesis trace for gene 669.

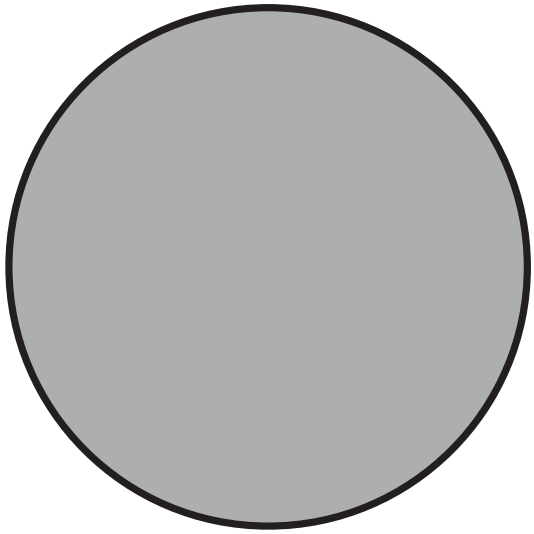

Figure 2.4: Trace plot showing the log(*posterior*) trace for the current model fit

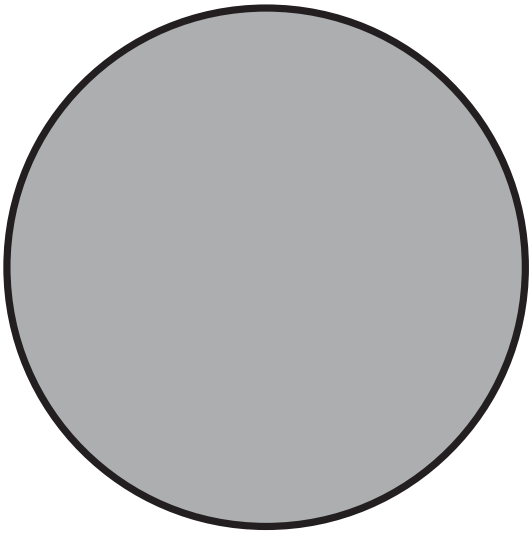

Figure 2.5: Fit of the ROC model for a random yeast. The solid line represent the model fit from the data, showing how synonymous codon frequencies change with gene expression. The points are the observed mean frequencies of a codon in that synthesis rate bin and the wisks indicate the standard diviation within the bin. The codon favored by selection is indicated by a "*". The bottom right panel shows how many genes are contained in heach bin

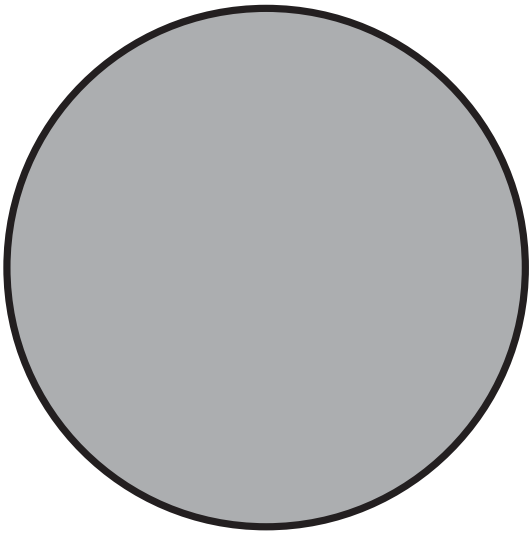

Figure 2.6: Comparisson of the selection parameter of seven yeast species estimated with ROC-SEMPPR.

Chapter 3

Decomposing mutation and selection to identify mismatched codon usage

This chapter is a lightly revised version of a paper to be submitted to Genome Biology and Evolution and co-authored with Michael A. Gilchrist.

C. Landerer, M.A. Gilchrist, Decomposing mutation and selection to identify mismatched codon usage

Abstract

Codon usage has been used as a measure for adaptation of genes to their cellular environment for decades. The introgression of genes from one cellular environment to another may cause well adapted genes to suddenly be less adapted due to them having evolved in a different environment. As a result, we expect that transferred genes result in a large fitness burden for the new host organism. Here we examine the yeast *Lachancea kluyveri* which has experienced a large introgression, replacing the left arm of chromosome C ($\sim$ 10% of its genome). The *L. kluyveri* genome provides an opportunity to study the evolution of introgressed genes to a novel cellular environment and estimate the fitness cost such a transfer imposes. We quantified the effects of mutation bias and selection against translation inefficiency on the codon usage pattern of the endogenous and introgressed exogenous genes using a Bayesian mixture framework. We found substantial differences in codon usage between the endogenous and exogenous genes, and show that these differences can be largely attributed to a shift in mutation bias from A/T ending codons in the endogenous genes to C/G ending codons in the exogenous genes. Recognizing the two different signatures of mutation and selection bias improved our ability to predict protein synthesis rate by 17% and allowed us to accurately assess codon preferences. In addition we utilize the estimates of mutation bias and selection against translation inefficiency to determine *Eremothecium gossypii* as potential source lineage, estimate the time since introgression to be on the order of 6×10^8 and assess the fitness burden across introgressed loci, showing the advantage of mechanistic models when analyzing codon data.

3.1 Introduction

Mutation, selection and genetic drift can be used to quantify the environment a genome has evolved in. Mutation bias is purely determined by the cellular environment, while the strength and efficacy of selection relative to drift is determined by the cellular environment, e.g. tRNA abundance, and the natural environment e.g. gene expression. A lineages effective population size determines the efficacy of selection relative to drift. Synonymous codon usage, the non-uniform usage of codons encoding the same amino acid, is a reflection of both, the cellular and the natural environment. Decomposing codon usage, therefore, provides us the necessary information to describe the environment a genome has evolved in in terms of its codon usage.

In general, the strength of selection on codon usage increases with gene expression [**?** **?**]. Conversely, the impact of mutation bias on codon usage declines with gene expression. Thus, we can easily imagine that with increasing gene expression, codon usage shifts from a process dominated mutation to a process dominated by selection. Together, the mutation process favoring specific synonymous codons - or mutation bias - and the selection for translation efficiency scaled by gene expression and effective population size - or selection bias - shape codon usage in a genome. This mutation-selection-drift balance model allows us to explicitly describe the environment in which genes evolve with respect to mutation and selection bias. Here we show that estimating the influence of mutation bias and selection bias on a gene's codon usage allows us to not only predict protein synthesis rate ϕ, but also, to infer its history and make predictions about its future with respect to these forces.

Most studies implicitly assume that synonymous codon usage of a genome is shaped by a single cellular environment. However, it is easy to think about the influence of multiple cellular environments within a cell, as genes are horizontally transferred, introgress, or as species hybridize. Genes introduced via horizontal gene transfer, introgression, or hybridization may carry the signature of a different, foreign cellular environment. These transferred genes may be less adapted to their new cellular environment, potentially imposing large fitness burdens to the organism. We expect a greater fitness burden of transferred genes if donor and recipient environment differ greatly in their selection bias, making such

transfers less likely. More practically, if transferred genes are unaccounted for, they may distort parameters by biasing estimates. This can lead to the conclusion of the wrong codon preference for an amino acid when analyzing a genome that has experienced such transfer events.

In this study, we analyze the synonymous codon usage of the genome of *Lachancea kluyveri*, the earliest diverging lineage of the Lachancea clade. The Lachancea clade diverged from the Saccharomyces clade prior to the whole genome duplication, about 100 Mya ago. Since its divergence from the other Lachancea, *L. kluyveri* has experienced a large introgression of exogenous genes. The introgression replaced the left arm of the C chromosome and displays a 13% higher GC content than the endogenous *L. kluyveri* genome [**?** **?**]. These characteristics make *L. kluyveri* an ideal model to study the effects of an introgressed cellular environment and the resulting mismatch codon usage.

Using ROC SEMPPR, a population genetics Bayesian model, allows us to quantify the cellular environment in which genes have evolved by separating and estimating effects of mutation bias and selection bias, and predicting protein production rate [**?**]. We use ROC SEMPPR to describe two cellular environments reflected in the *L. kluyveri* genome, a native endogenous and an introgressed exogenous environment. Our results indicate that the difference in GC content between endogenous and exogenous genes mostly to differences in mutation bias. Recognizing the differences in codon usage between the endogenous and exogenous gene sets also improves our ability to predict protein synthesis rate from the sequence data alone.

With our improved model fits, we obtained more reliable estimates of mutation bias, selection bias and protein synthesis rate, allowing us to address more refined questions of biological importance. First we determine a potential source lineage of the exogenous genes using a combination of information in codon usage and gene synteny. We compared estimates of mutation bias (ΔM) and selection bias ($\Delta\eta$) for the exogenous genes to 38 yeast lineages and further investigated candidate lineages using synteny. Second, we estimate the time since introgression and the persistence of the signal of the exogenous cellular environment from our estimates of ΔM using an exponential model of decay. Third, we estimate the selective cost of the mismatched codon usage for the introgression, using our estimates of $\Delta\eta$

and protein synthesis rate ϕ. Thus, in addition to being able to estimate codon preference and gene expression to describe codon usage patterns, we also gain insights into the evolution of genes that have been transferred between lineages.

3.2 Results

Model selection and validation confirmed that the *L. kluyveri* genome contains signatures of at least two cellular environments. We compared model fits of ROC SEMPPR to the homogeneous *L. kluyveri* genome and the separated sets of endogenous and exogenous genes of 4864 and 497 genes respectively, using AnaCoDa [**?**]. We compared estimates of the cellular environment to describe differences in endogenous and exogenous codon usage. Furthermore, we utilize the differences in model fit and parameters estimated from the endogenous and exogenous genes to explore the evolution of the exogenous gene set.

AIC indicates that parameter estimates for mutation bias (ΔM) and selection bias ($\Delta\eta$) differ greatly between exogenous and endogenous gene sets. As a result, the partitioning of the *L. kluyveri* genome into an endogenous and exogenous gene set is clearly favored by model selection. The inclusion of 81 additional parameters (40 for ΔM, 40 for $\Delta\eta$, and one for s_ϕ) necessary to describe both gene sets separately improves our model fit by $\sim 75,000$ AIC units (5,311,060 for the combined gene set vs 5,235,598 for the separated gene sets).

In addition to model selection, we utilized independent information on gene expression to evaluate model fit. Recognizing differences in ΔM and $\Delta\eta$ for the endogenous and exogenous gene sets substantially improves our ability to predict protein synthesis rate ϕ ($\rho = 0.69$ vs. $\rho = 0.59$ for the full genome; Figure 3.1).

3.2.1 Differences in the Endogenous and Exogenous Codon Usage

As our estimates of parameters for a codon family coding for an amino acid are relative to a reference codon, changes in the reference codon will change the order between sets. To better compare our estimates of of mutation bias (ΔM) and selection bias ($\Delta\eta$) obtained from fitting ROC SEMPPR between the endogenous and exogenous gene sets, we express our estimates relative to the mean for each codon family. As We find larger differences between

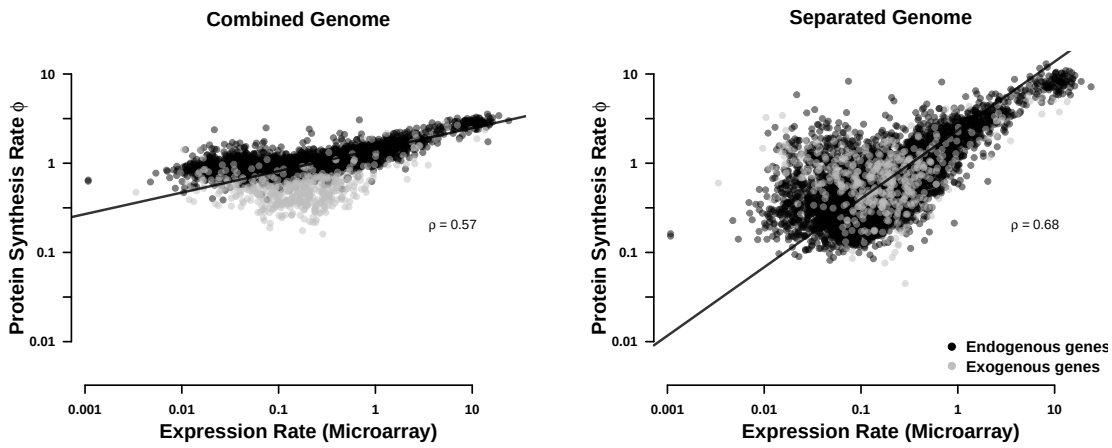

Figure 3.1: Comparison of predicted protein synthesis rate ϕ to Microarray data from **?**] for (a) the combined genome and (b) the separated endogenous and exogenous genes. Endogenous genes are displayed in black and exogenous genes in gray. Black line indicates type II regression line.

ΔM than $\Delta\eta$ (Figure 3.2). Estimates of ΔM in the endogenous genes negatively correlate with the ΔM estimates for the exogenous genes ($\rho = -0.49$) indicating strong discordance in the mutation environment between *L. kluyveri* and the donor lineage of the exogenous genes. For example, $\sim$ 95% of codon families show mutation preference for A/T ending codons, in contrast, the exogenous genes display an equally strong mutation bias towards C/G ending codons. Only the two codon amino acid Phenylalanine (Phe, F) shows complete concordance between endogenous and exogenous genes in their ΔM values.

Our estimates of $\Delta\eta$ for the endogenous and exogenous genes were positively correlated ($\rho = 0.69$), indicating increased concordance of $\sim$ 53% between the two selection environments (Figure 3.2). Nevertheless, the endogenous genes only show a selection preference for C and G ending codons in $\sim$ 58% of the codon families. In contrast, the exogenous genes display a strong preference for A and T ending codons in $\sim$ 89% of the codon families.

The difference in codon preference between endogenous and exogenous genes is striking. Fits to the complete *L. kluyveri* genome reveal that the relatively small exogenous gene set has a disproportional effect on the model fit. We find that the complete *L. kluyveri* genome is estimated to share the mutational preference with the exogenous genes in $\sim$ 78% of codon

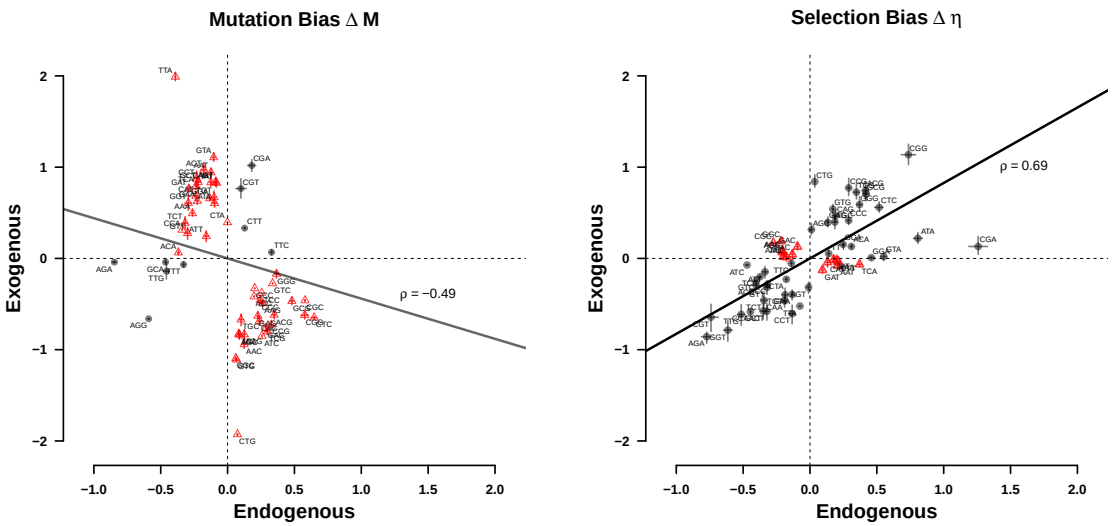

Figure 3.2: Comparison of (a) mutation bias ΔM and (b) selection bias $\Delta\eta$ of endogenous and exogenous genes. Estimates are relative to the mean for each codon family. Black dots indicate parameters with sign concordance, red dots indicate parameters with sign discordance between endogenous and exogenous genes. Black line shows the type II regression. Dashed lines mark quadrants.

families with discordance between endogenous and exogenous genes. In two cases, Isoleucine (Ile, I) and Arginine (Arg, R), the strong discrodance in mutation preference results in a estiamted codon preference in the complete *L. kluyveri* genome that is not reflected by either endogenous nor exogenous genes.

The impact of the small exogenous gene set on the fit to the complete *L. kluyveri* genome is less prevalent in our estimates of selection bias $\Delta\eta$ but still strong. We find that the complete *L. kluyveri* genome is estimated to share the selection preference with the exogenous genes in $\sim 60\%$ of codon families with discordance between endogenous and exogenous genes. Therefore, it is important to recognize and treat endogenous and exogenous genes as separate sets to avoid the inference of incorrect synonymous codon preferences.

3.2.2 Determining Source of Exogenous Genes

We combined our estimates of mutation bias (ΔM) and selection bias ($\Delta\eta$) with synteny information and searched for potential source lineages of the introgressed region. We examined 38 yeast lineages of which two (*Eremothecium gossypii* and *Candida dubliniensis*)

showed a strong positive correlation in codon usage (Figure 3.3a). The endogenous

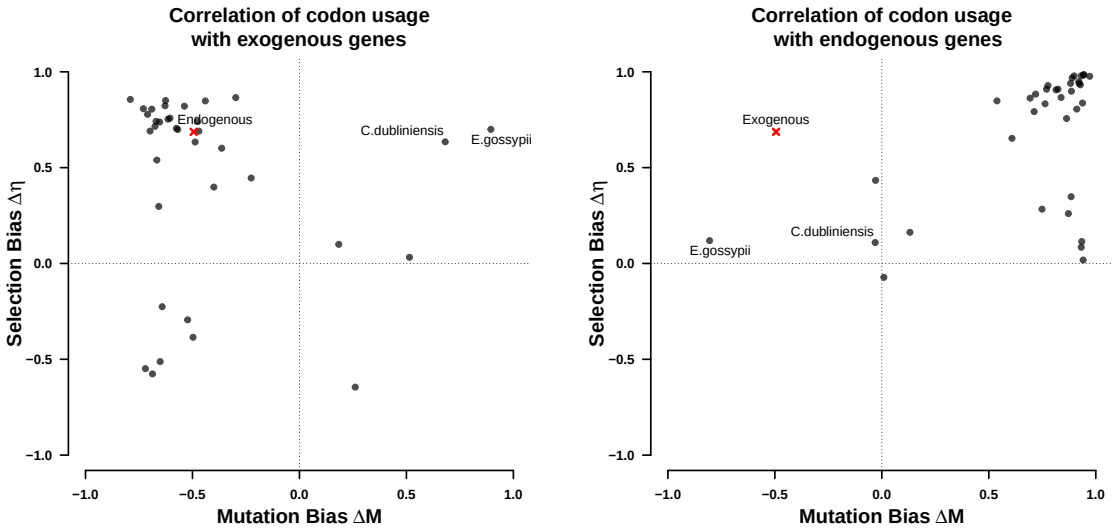

Figure 3.3: Correlation of ΔM and $\Delta\eta$ of the (a) exogenous and (b) endogenous genes with 38 examined yeast lineages. Dots indicate the correlation of ΔM and $\Delta\eta$ of the lineages with the endogenous and exogenous parameter estimates. All regressions were performed using a type II regression.

L. kluyveri genome exhibits codon usage very similar to most yeast lineages examined, indicating little variation in codon usage among the examined yeasts (Figure 3.3b). Four lineages show a positive correlation for ΔM and $\Delta\eta$ with the exogenous genes and have a weak to moderate positive correlation in selection bias with the endogenous genes; but, like the exogenous genes, tend to have a negative correlation in ΔM with the endogenous genes.

Comparing synteny between the exogenous left arm of chromosome C, and *E. gossypii* and *C. dubliniensis* as well as closely related yeast species we find that *E. gossypii* displays the highest synteny coverage (Figure S3a,b). *C. dubliniensis*, even though it displays similar codon usage does not show synteny with the exogenous region. Furthermore, the synteny relationship between the exogenous region and other yeasts appears to be limited to the Saccharomycetacease group(Figure S3b). Given these results, we conclude that the *E. gossypii* lineage is the most likely source of the introgressed exogenous genes.

3.2.3 Estimating Introgression Age

We estimated the introgression age using an exponential model of decay for mutation bias, by assuming that *E. gossypii* is still representative of the mutation bias of its ancestral source lineage at the time of the introgression. We utilize the ΔM estimates for all two codon amino acids and infer the age of the introgression to be on the order of $6.2 \pm 1.2 \times 10^8$ generations. We assume a mutation rate of 3.8×10^{-10} per nucleotide per generation, a value in line with other estimates [**?** **?**]. *L. kluyveri* experiences between one and eight generations per day, we therefore expect the introgression to have occurred about $205,000$ to $1,600,000$ years ago which is longer than previous estimates of **?**]. However, our estimates are likely overestimates as they assume a purely neutral decay.

Furthermore, we estimated the persistence of the signal of the foreign cellular environment. Assuming that differences in mutation bias will decay more slowly than differences in selection bias, we predict that the ΔM signal of the source cellular environment will have decayed to be within one percent of the *L. kluyveri* environment within about $5.4 \pm 0.2 \times 10^9$ generations.

3.2.4 Fitness Burden of the Exogenous Genes

Estimates of selection bias for the exogenous genes show that, while well correlated with the endogenous genes, only nine amino acids share the optimal codon. We therefore expect that the introgressed genes represent a significant reduction in fitness, or genetic load for *L. kluyveri*, and even more so at the time of introgression. As the introgression occurred before the diversification of *L. kluyveri* and has fixed since then throughout the various populations, we are left without the original chromosome arm [**?**]. However, using our estimates of ΔM and $\Delta\eta$ from the endogenous genes, we can estimate the genetic load of the exogenous genes relative to an expected gene set. We define genetic load as the difference between the fitness of an expected, replaced endogenous gene and the inferred introgressed gene relative to drift $sN_e \propto \phi\Delta\eta$ (See Methods for details).

We estimate the genetic load of the exogenous genes at the time of introgression (Figure 3.4a) and currently (Figure 3.4b). These estimates are dependent on three key assumptions.

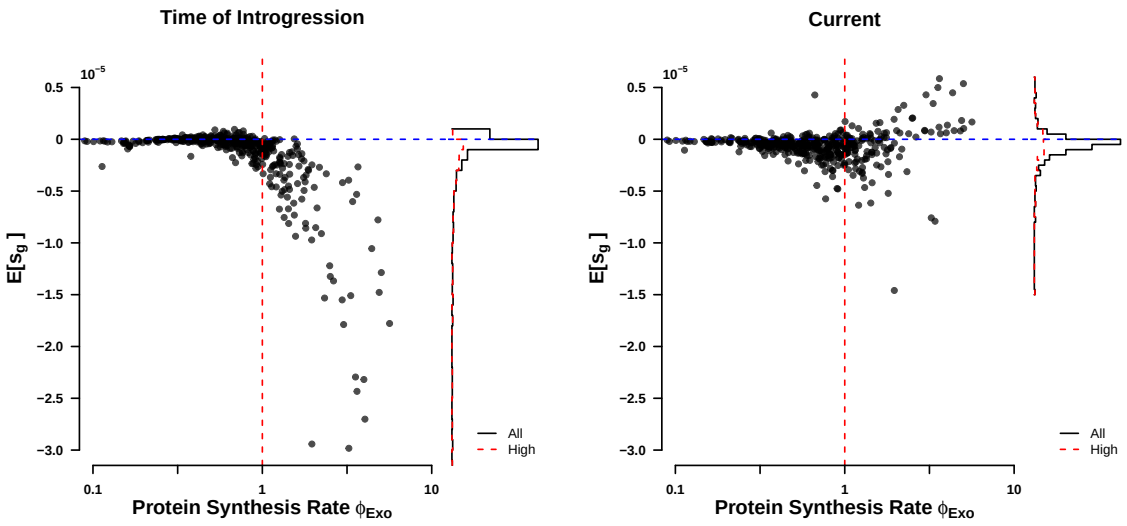

Figure 3.4: Genetic load Δs (left) at the time of introgression ($\kappa = 5$), and (right) currently ($\kappa = 1$).

First, we assume again that the current cellular environment of *E. gossypii* is reflective of the ancestral environment. Second, we assume that the current amino acid composition of the exogenous genes is the same as in the replaced endogenous genes. Third, we assume that the difference in the efficacy of selection between *E. gossypii* and *L. kluyveri* can be described with a simple scaling term we call κ (Figure S4b). As $\Delta\eta$ is defined as $\Delta\eta = 2N_e q(\eta_i - \eta_j)$, we can not distinguish if κ is scaling protein synthesis rate ϕ, effective population size N_e, or the value of an ATP q [**?**].

At the time of the introgression, we predict that only a few genes were weakly exapted (Figure 3.4a) with all high expression genes ($\phi > 1$) being maladapted to the novel cellular environment. However, these highly expressed genes show the greatest rate of adaptation to the *L. kluyveri* cellular environment (Figures 3.4a, S5).

3.3 Discussion

Using ROC SEMPPR we show that the *L. kluyveri* genome contains two distinct signatures of cellular environments, its own endogenous and a foreign exogenous one obtained by an introgression event ($\Delta AIC = 78,000$). Following **?**], who defined the boundary of the

anomalous chromosome region based on its elevated GC content, we partitioned the *L. kluyveri* genome into an endogenous and an exogenous gene set using gene location. We estimated the codon usage of the entire *L. kluyveri* genome and the separated endogenous and exogenous gene sets (Figure S6). Both, Mutation bias and selection bias differ between endogenous and exogenous genes. The endogenous genes show a strong mutation bias towards A/T ending codons, while the exogenous genes show mutation is bias towards G/C ending codons. We observed the reversed to be true in selection bias, leading to a strong mismatch in codon usage between the gene sets, supporting our notion of two distinct signatures of codon usage.

Only half of the codon families share the same optimal codon in the endogenous and exogenous gene sets. However, we find that the strength of selection within a codon family differs between gene sets, causing a change in rank order. Nevertheless, we find a high correlation for our estimates of selection bias $\Delta\eta$ between the two gene sets. Our estimates of the optimal codon differ in nine cases between endogenous and exogenous genes. Interestingly, when the difference in codon usage is ignored, we find that in seven out of these nine cases the exogenous codon preference is inferred as optimal(Table S2). We find even greater discordance in our estimates of ΔM between endogenous and exogenous gene sets(Table S1). Without recognizing this difference in codon preference our estimates would not have been reflective of the actual codon usage of the *L. kluyveri* genome but of a relatively small introgressed gene set. This shows that a small number of exogenous genes ($\sim$ 9% of genes) can have a disproportional impact on our estimates of ΔM and $\Delta\eta$ when fitting ROC SEMPPR to the entire *L. kluyveri* genome. While this is surprising, it highlights the importance to recognize differences in codon usage within a genome. Our results also indicate that we can attribute the higher GC content in the exogenous genes mostly to differences in mutation bias favoring G/C ending codons rather than a novel selective force.

Separating the endogenous and exogenous genes improves our estimates of protein synthesis rate ϕ by 17% relative to the full genome estimate ($\rho = 0.59$ vs. $\rho = 0.69$, respectively). Furthermore, we find that the variation in our estimates of ϕ is more consistent with the current understanding of gene expression (compare Figure 3.1a and b). Small variation in ϕ estimates may serve as an indicator for the presents of the signature of multiple

cellular environments in future work. In the case of the *L. kluyveri* genome, finding a severe mismatch in ΔM causes ϕ values for low expression genes ($\phi < 1$) to increase towards the inflection point where the dominance of mutation gives way to selection. In the case of the two codon amino acids, the inflection point represents the point at which mutation and selection are contributing equally to the probability of a codons occurrence. We find this inflection point around $\phi = 1$ for most amino acids (Figure S6). However, ROC SEMPPR assumes that estimates of ϕ follow a log-normal distribution with an expected value $E[\phi] = 1$. This assumption allows us to interpret $\Delta\eta$ as the strength of selection relative to drift (sN_e) for a codon in a gene with the average protein synthesis rate $\phi = 1$. However, tying the mean and standard deviation of the prior distribution together. Therefore, an increase in ϕ for low expression genes has to be meet with a decrease of ϕ for high expression genes, reducing the overall variance in ϕ (see **?**] for details).

Having shown that the introgressed exogenous genes reflect a foreign cellular environment, we used the quantitative estimates of mutation bias ΔM and selection bias $\Delta\eta$ from ROC SEMPPR to identify potential source lineages. The comparison of the endogenous and exogenous ΔM and $\Delta\eta$ estimates to 38 other yeast lineages revealed that most yeasts examined share similarity in mutation bias (Figure 3.2). Similar, we find strong similarities in selection bias between examined yeasts, potentially indicating stabilizing selection on codon usage. However, the exogenous genes do not share this commonality (Figure 3.2a), as their mutation bias strongly deviates from the endogenous genes and most other yeast species examined. This large difference in mutation bias between endogenous and exogenous genes allowed us to limit our candidate list to only two likely lineages, *C. dubliniensis* and *E. gossypii*. Interestingly, we did not find *Lachancea thermotolerance*, a thermophilic lineage closely related to *L. kluyveri*, as a potential candidate. While *L. thermotolerance* does have a strong synteny relationship with *L. kluyveri*, it does not show similarity in codon usage with the exogenous genes and does not share their high GC content.

Inference of synteny relationships between the exogenous region and *C. dubliniensis* and *E. gossypii* as well as closely related species showed that synteny relationship is limited to the Saccharomycetaceae clade (Figure S3b). *E. gossypii* showed the highest syntenty coverage and is the only species with similar codon usage. Furthermore, *E. gossypii* is the only species

examined with a GC content $> 50\%$ like it is observed in the exogenous region. The synteny coverage extends along the whole exogenous regions with the exception to the very 3' and 5' end of the region. The lack of synteny at the ends of the region also coincides with a drop in GC content, potentially indicating remains of the original replaced region or increased adaptation. The ancestral introgressed region may have also broken up in *E. gossypii* as we find non overlapping synteny with chromosomes *VI* and *V* as well as have indication that the C chromosome of *L. kluyveri* very robust to recombination events [**?** **?**].

With *E. gossypii* identified as potential source lineage of the introgressed region, we inferred the time past since the introgression occurred using our estimates of mutation bias ΔM. The ΔM estimates are well suited for this task as they are free of the influence of selection and unbiased by N_e and other scaling terms, which is in contrast to our estimates of $\Delta\eta$ [**?**]. We estimated the time since introgression to be on the order of 6×10^8 generations, which is ~ 10 times longer time than a previous estimate by **?**] of a minimum of 5.6×10^7 generations . However, our estimate implicitly assumes all mutations are neutral, it is therefore a conservative estimate, potentially overestimating the time since introgression. Our estimate also depend on the assumption that the *E. gossypii* cellular environment reflects the ancestral environment at the time of the introgression. If the the ancestral mutation environment was more similar to the *L. kluyveri* environment at the time of the introgression than the *E. gossypii* environment is today, we would overestimate this time. On the other hand, we would underestimate the time since introgression if the two cellular environments were more dissimilar. We could have attempted to reconstruct the ancestral state of *E. gossypii*, however, as methods for ancestral state reconstruction are phenomenological, assumptions would be unclear.

The estimates of mutation bias ΔM also allow us the infer the time until the signature of the exogenous cellular environment will have decayed to be indistinguishable at about one percent difference. Our estimate of decay is an order of magnitude greater than our estimate of the time since introgression (5×10^9 and 6×10^8 generations). Estimates of decay based on ΔM are more conservative as we expect differences in $\Delta\eta$ to decay before due to selection favoring the decay.

As we have determined that the introgression event has a long persisting exogenous signature, it is important to understand the fitness consequences of such an event. We estimated the genetic load that the exogenous genes represent assuming that the replaced endogenous genes and the new exogenous genes had the same amino acid composition. This assumption, along with the assumption that the current *L. kluyveri* cellular environment is reflective of the cellular environment at the time of the introgression is necessary to estimate the expected endogenous sequence that was replaced. Our results show that individual low expression genes contribute little to the genetic load, and show less adaptation to the novel cellular environment (Figure 3.4, S5). A small number of low expression genes even appear exapted, likely due to the mutation bias in the endogenous genes matching the selection bias in the exogenous genes for G/C ending codons. Highly expressed genes on the other hand have greatly adapted to the *L. kluyveri* cellular environment. This, however, does not mean that these genes show a higher rate of evolution, but that small changes in their sequence have large impacts on the fitness burden these sequences represent. To this day, the exogenous genes represent a significant fitness burden on *L. kluyveri*. However, our estimates are conservative as we do not account for potential changes in the codon usage of *E. gossypii*. While divergent evolution in codon usage between *E. gossypii* and *L. kluyveri* would cause us to overestimate the genetic load, convergent evolution, on the other hand, would cause us to underestimate the genetic load. However, as the introgression appears to have reached fixation [**?**], the genetic load relative to the replaced chromosome arm is only of theoretical interest.

The large genetic load the exogenous genes represented at the time of the introgression indicates that the fixation of the introgression was a very unlikely event in a population with a large N_e as it is typical for yeasts. It is hard to contextualize the probability of this introgression being fixed as we are not aware of any estimates of the frequency at which such large scale introgressions of genes with very different signatures of codon usage occur. One example is *Saccharomyces bayanus*, a hybrid of *Saccharomyces uvarum*,*Saccharomyces cerevisiae*, and *Saccharomyces eubayanus*. However, unlike with *L. kluyveri* and *E. gossypii* it appears that the donor lineages show similar codon usage. *Saccharomyces cerevisiae* and *Saccharomyces eubayanus* show a very strong correlation between selection bias $\Delta\eta$ of

$\rho = 0.98$ and a strong correlation between mutation bias ΔM of $\rho = 0.83$ We were unable to identify codon usage for *Saccharomyces uvarum*. However, *L. kluyveri* diverged about 85 Mya ago from the rest of the Lachancea clade. This represents between 10^{10} to 10^{11} generations. Assuming for yeasts typical effective population size on the order of 10^8, we are left with 10^{18} to 10^{19} opportunities for such an event to occur. In addition, the strong mutation bias towards G/C ending codons in the exogenous genes may have contributed to the fixation of this introgression (include figure of ΔM v $\Delta\eta$). It is, on the other hand, also possible that despite their mismatch in codon usage, the exogenous genes have represented a fitness increase due to external environmental factors resulting in the fixation of the introgression.

In conclusion, our results show the usefulness of the separation of mutation bias and selection bias and the importance of recognizing the presence of multiple cellular environments in the study of codon usage. We also illustrate how a mechanistic model like ROC SEMPPR and the quantitative estimates it provides can be used for more sophisticated hypothesis testing in the future. In contrast to other approaches used to study codon usage like CAI [**?**] or tAI [**?**], ROC SEMPPR is sensitive to differences in mutation bias. We highlight potential pitfalls when estimating codon preferences, as estimates can be biased by the signature of a second, historical cellular environment. In addition, we show how quantitative estimates of mutation bias and selection relative to drift can be obtained from codon data and used to infer the fitness cost of an introgression as well as its history and potential future.

3.4 Materials and Methods

3.4.1 Separating endogenous and exogenous genes

A GC-rich region was identified by **?**] in the *L. kluyveri* genome extending from position 1 to 989,693 of chromosome C. This region was later identified as an introgression by **?**]. We obtained the *L. kluyveri* genome from SGD Project http://www.yeastgenome.org/download-data/ (last accessed: 09-27-2014) and the annotation for *L. kluyveri* NRRL Y-12651 (assembly ASM14922v1) from NCBI (last accessed: 12-09-2014). We assigned 457

genes located on chromosome C with a location within the $\sim 1Mb$ window to the exogenous gene set. All other 4864 genes of the *L. kluyveri* genome were assigned to the exogenous genes. All genes could be uniquely assigned to one or the other gene set.

3.4.2 Model Fitting with ROC SEMPPR

ROC SEMPPR was fitted to each genome using AnaCoDa (0.1.1) [**?**] and R (3.4.1). ROC SEMPPR was run from multiple starting values for at least 250,000 iterations, every 50th sample was collected to reduce autocorrelation. After manual inspection to verify that the MCMC had converged, parameter posterior means were estimated from the last 500 samples.

3.4.3 Comparing codon specific parameter estimates

Choice of reference codon does reorganize codon families coding for an amino acid relative to each other, therefore all parameter estimates are relative to the mean for each codon family.

$$\Delta M^{c}_{i,a} = \Delta M_{i,a} - \Delta \bar{M}_{a} \tag{3.1}$$

$$\Delta \eta^{c}_{i,a} = \Delta \eta_{i,a} - \Delta \bar{\eta}_{a} \tag{3.2}$$

Comparison of codon specific parameters (ΔM and $\Delta \eta$) was performed using the function lmodel2 in the R package lmodel2 (1.7.3) and R version 3.4.1. Type II regression was performed with re-centered parameter estimates, accounting for noise in dependent and independent variable.

3.4.4 Synteny

We obtained complete genome sequences from NCBI (last accessed: 02-05-2017). Genomes were aligned and checked for synteny using SyMAP (4.2) with default settings [**?** **?**]. We assessed Synteny as percentage non-overlapping coverage of the exogenous gene region (Figure S3b).

3.4.5 Determining introgression timeline

We modeled the change in codon frequency over time using an exponential model for all two codon amino acids, and describing the change in codon c_1 as

$$\frac{dc_1}{dt} = -\mu_{1,2}c_1 - \mu_{2,1}(1 - c_1) \tag{3.3}$$

where $\mu_{i,j}$ is the rate at which codon i mutates to codon j and c_1 is the frequency of the reference codon. Our estimates of ΔM_{endo} can be directly related to the steady state of equation 3.3.

$$\frac{\mu_{2,1}}{\mu_{1,2} + \mu_{2,1}} = \frac{1}{1 + \exp(\Delta M_{endo})} \tag{3.4}$$

Solving for $\mu_{1,2}$ gives us $\mu_{1,2} = \Delta M_{endo} \exp(\mu_{2,1})$ which allows us to rewrite and solve equation 3.3 as

$$c_1(t) = \frac{\exp(-t(1 + \Delta M_{endo})\mu_{2,1}) \exp(t(1 + \Delta M_{endo})\mu_{2,1}) + (1 + \Delta M_{endo})K}{1 + \Delta M_{endo}} \tag{3.5}$$

where K is

$$K = \frac{-1 + c_1(0) + c_1(0)\Delta M_{endo}}{1 + \Delta M_{endo}} \tag{3.6}$$

Equation 3.5 was solved over time with a mutation rate $m_{2,1}$ of 3.8×10^{-10} per nucleotide per generation [**?**]. Initial codon frequencies $c_1(0)$ for each codon family where taken from our estimates of ΔM_{gos} from *E. gossypii*. Current codon frequencies for each codon family where taken from our estimates of ΔM from the exogenous genes. Mathematica (9.0.1.0) [**?**] was used to calculate the time t_{exo} it takes for the initial codon frequencies $c_1(0)$ for each codon family to change to the current exogenous codon frequencies. The same equation was used to determine the time t_{endo} at which the signal of the exogenous cellular environment has decayed to within 1% of the endogenous environment.

3.4.6 Estimating fitness burden

To estimate the fitness burden, we made three key assumptions. First, we assumed that the current exogenous amino acid composition of genes is representative of the replaced

endogenous genes. Second, we assume that the currently observed cellular environment of *E. gossypii* reflects the cellular environment that the exogenous genes experienced before transfer to *L. kluyveri*. Lastly, we assume that the difference in the efficacy of selection between the cellular environments of the source lineage and *L. kluyveri* can be expressed as a scaling constant and that protein synthesis rate ϕ has not changed between the replaced endogenous and the introgressed exogenous genes.

We calculated the fitness burden each gene represents assuming additive fitness effects as

$$(sN_e)_g = \sum_{i}^{C} -\kappa\phi_g \Delta\eta_i n_{g,i} \tag{3.7}$$

where $(sN_e)g$ is the selection against translation inefficiency relative to drift. ϕ_g is the estimated protein synthesis rate for gene g in the exogenous gene set. $n_{g,i}$ is the codon count of each codon i in the codon set C for each gene g. κ is a constant, scaling the efficacy of selection between cellular environments. Like stated previously, our $\Delta\eta$ are relative to the mean of the codon family. We find that the fitness burden of the introgressed genes is minimized at $\kappa \sim 5$ (Figure S4b). Thus, we set $\kappa = 1$ if we calculate the $(sN_e)g$ for the endogenous and the current exogenous genes, and $\kappa = 5$ for $(sN_e)g$ for the fitness burden at the time of introgression. Since we are unable to observe codon counts for the replaced endogenous genes and for the exogenous genes at the time of introgression, we calculate expected codon counts

$$E[n_{g,i}] = \frac{\exp(-\Delta M_i - \Delta\eta_i\phi_g)}{\sum_j^C \exp(-\Delta M_j - \Delta\eta_j\phi_g)} \times m_{a_i} \tag{3.8}$$

m_{a_i} is the number of occurrences of amino acid a that codon i codes for.

We report the fitness burden of the introgression as $\Delta sN_e = (sN_e)_{Intro} - (sN_e)_{Endo}$ where $(sN_e)_{Intro}$ is either the fitness burden at the time of the introgression or presently.

3.5 Acknowledgments

This work was supported in part by NSF Awards MCB-1120370 (MAG and RZ) and DEB-1355033 (BCO, MAG, and RZ) with additional support from The University of Tennessee Knoxville. CL received support as a Graduate Student Fellow at the National Institute for Mathematical and Biological Synthesis, an Institute sponsored by the National Science Foundation through NSF Award DBI-1300426, with additional support from UTK. The authors would like to thank Brian C. O'Meara and Alexander Cope for their helpful criticisms and suggestions for this work.

Amino Acid	*E. gossypii*	Endogenous	Exogenous	*L. kluyveri*
Ala A	GCG	GCA	GCG	GCG
Cys C	TGC	TGT	TGC	TGC
Asp D	GAC	GAT	GAC	GAC
Glu E	GAG	GAA	GAG	GAG
Phe F	TTC	TTT	TTT	TTT
Gly G	GGC	GGT	GGC	GGC
His H	CAC	CAT	CAC	CAC
Ile I	ATC	ATT	ATC	ATA
Lys K	AAG	AAA	AAG	AAA
Leu L	CTG	TTG	CTG	CTG
Asn N	AAC	AAT	AAC	AAT
Pro P	CCG	CCA	CCG	CCG
Gln Q	CAG	CAA	CAG	CAG
Arg R	CGC	AGA	AGG	CGG
Ser_4 S	TCG	TCT	TCG	TCG
Thr T	ACG	ACA	ACG	ACG
Val V	GTG	GTT	GTG	GTG
Tyr Y	TAC	TAT	TAC	TAC
Ser_2 Z	AGC	AGT	AGC	AGC

Table S1: Synonymous codon preference in the various data sets based on our estimates of ΔM

3.6 Supplementary Material

Supporting Materials for *Fitness consequences of mismatched codon usage* by Landerer *et al.*.

Amino Acid	*E. gossypii*	Endogenous	Exogenous	*L. kluyveri*
Ala A	GCT	GCT	GCT	GCT
Cys C	TGT	TGT	TGT	TGT
Asp D	GAT	GAC	GAT	GAT
Glu E	GAA	GAA	GAA	GAA
Phe F	TTT	TTC	TTC	TTC
Gly G	GGA	GGT	GGT	GGT
His H	CAT	CAC	CAT	CAT
Ile I	ATA	ATC	ATT	ATT
Lys K	AAA	AAG	AAA	AAG
Leu L	TTA	TTG	TTG	TTG
Asn N	AAT	AAC	AAT	AAC
Pro P	CCA	CCA	CCT	CCA
Gln Q	CAA	CAA	CAA	CAA
Arg R	AGA	AGA	AGA	AGA
Ser_4 S	TCA	TCC	TCT	TCT
Thr T	ACT	ACC	ACT	ACT
Val V	GTT	GTC	GTT	GTT
Tyr Y	TAT	TAC	TAT	TAC
Ser_2 Z	AGT	AGT	AGT	AGT

Table S2: Synonymous codon preference in the various data sets based on our estimates of $\Delta\eta$

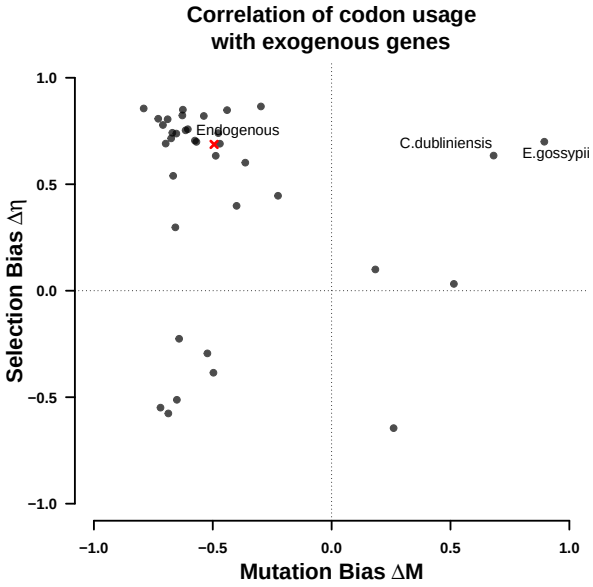

Figure S1: Correlation of ΔM and $\Delta\eta$ of the endogenous genes with 38 examined yeast lineages. Dots indicate the correlation of ΔM and $\Delta\eta$ of the lineages with the endogenous and exogenous parameter estimates. All regressions were performed using a type II regression.

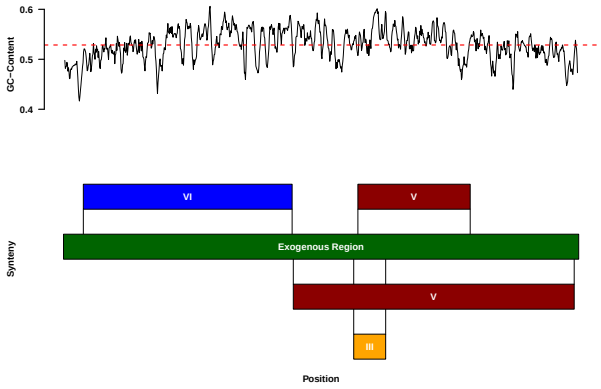

Figure S2: Synteny relationship of *E. gossypii* and the exogenous genes

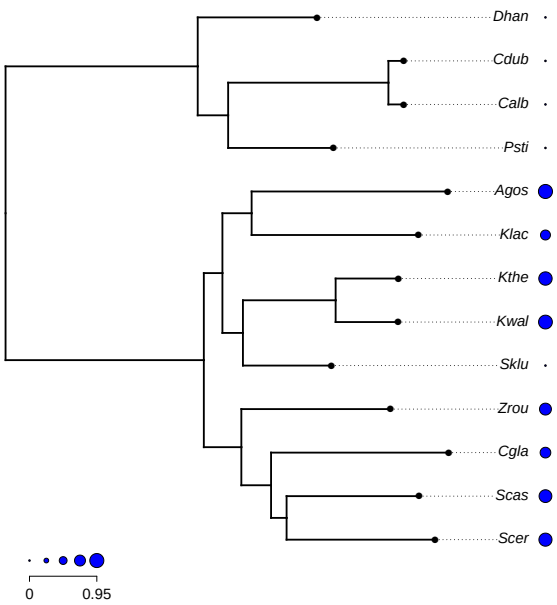

Figure S3: Amount of synteny for each species (Units of std dev) checked for synteny.

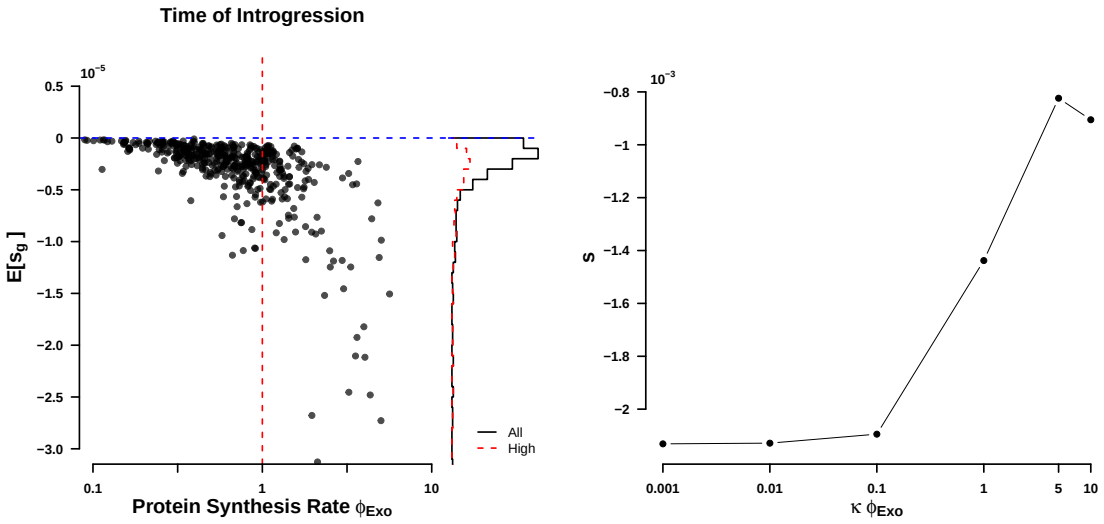

Figure S4: Suppl Fig: Fitness burden (left) without scaling of ϕ, and change of total fitness burden with scaling κ

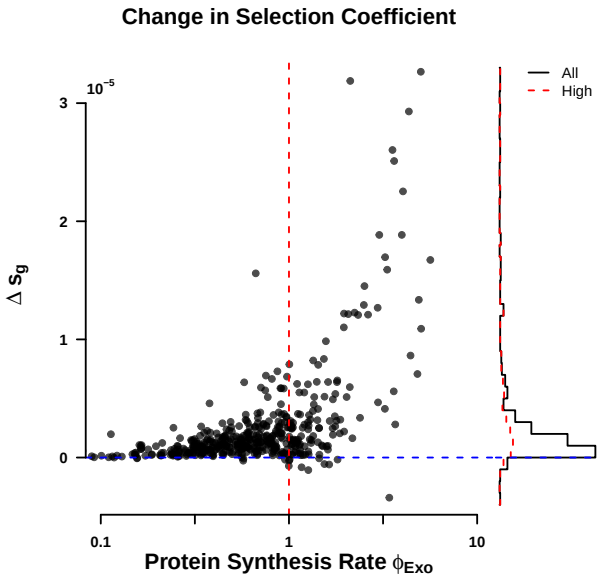

Figure S5: Total amount of adaptation between time of introgression and now

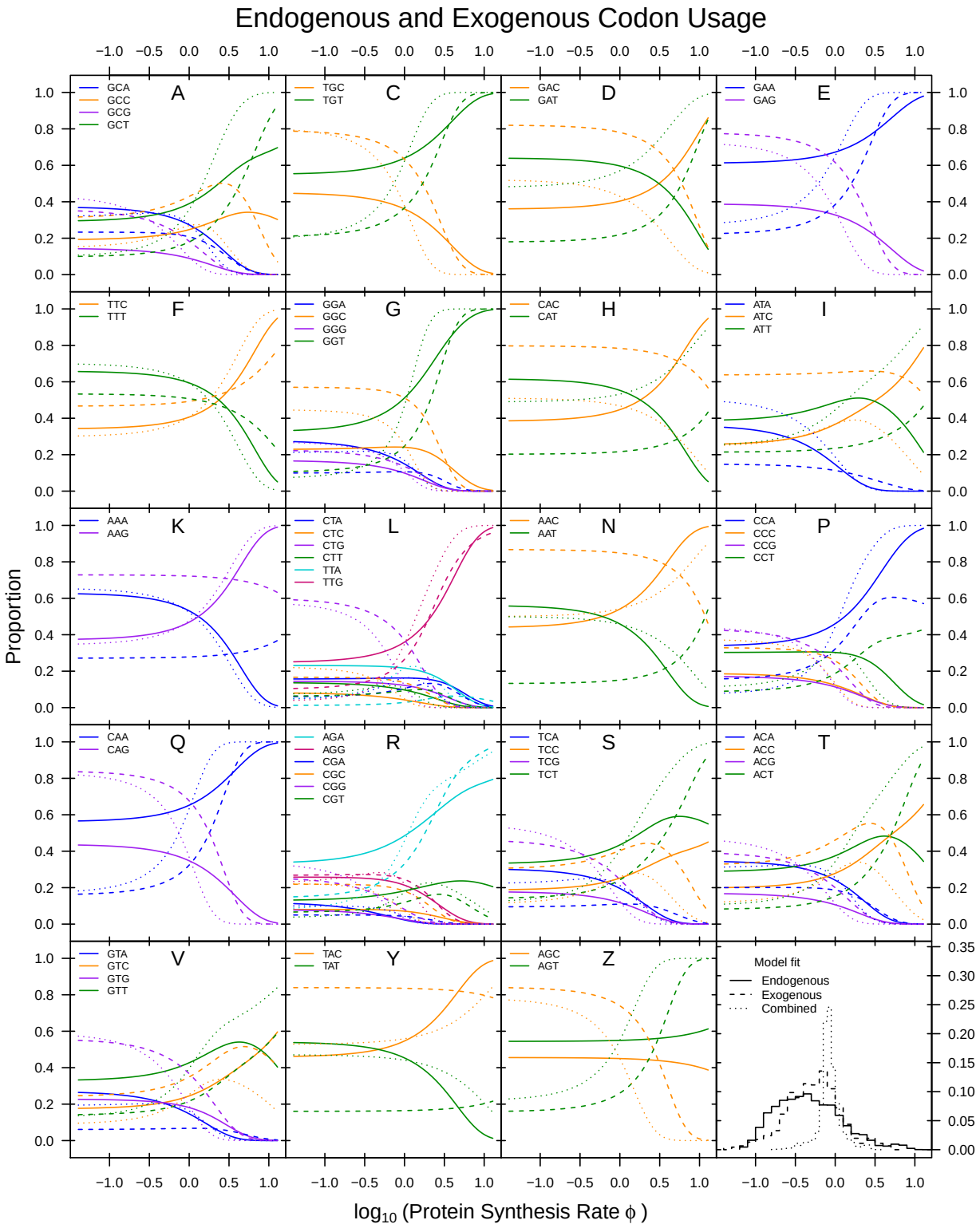

Figure S6: Suppl Fig