Statistical Analysis of Coding Sequences

Set-up of an R package
statanacoseq
distributed on GitHub
to carry future implementations of
Codon Bias Indexes.

Master Thesis 2016
presented on August 12 2016
at the Faculty of Science
Institute of Plant Sciences (IPS)
in the master program of Bioinformatics
University of Bern and University of Fribourg

for the master degree in
Bioinformatics and Computational Biology
by

Fredy Siegrist

accepted on the faculty reunion:

Declaration

under Art. 28 Para. 2 RSL 05

Last, first name: Siegrist Fredy

Matriculation number: 99-119-380

Programme: Bioinformatics and Computational Biology

Thesis title: Statistical Analysis of Coding Sequences

Thesis supervisor: Dr. Cannarozzi Gina

I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person, except where due acknowledgement has been made in the text. In accordance with academic rules and ethical conduct, I have fully cited and referenced all material and results that are not original to this work. I am well aware of the fact that, on the basis of Article 36 Paragraph 1 Letter o of the University Law of 5 September 1996, the Senate is entitled to deny the title awarded on the basis of this work if proven otherwise. I grant insprection of my thesis.

Bern, 12.08.2016

University of Bern, 2016

But for a champion, the lasting memory
is always the loss in the race unfinished.
[...] My race is now unfinished.

— **Abebe Bikila**
from *Atletu* (2009)

To my friends...

Acknowledgements

First of all I would like to thank my supervisor Gina Cannarozzi to support my idea to not only transcribed the codon bias functions to a language we students are familiar with but to generate a R package that resembles a complete code. Thanks go to my Boss Zerihun Tadele for accepting a master student in his Teff group that is more interested in working on the bits that defines the teff genome than on the plants itself and more interested to run grasslands than to breed grasses. I thank the whole department to give me a nice introduction into plant science and to leave some food in the kitchen for me to survive the long coding evenings. I would like to thank all my friends who accepted my regression to go back to school and my attitude to no longer join them in drinking and eating out and all those that found other ways to entertain me. Special thanks go to my friends that offered me a bed to bring variation in my training sessions to discover new running tracks and all sportsmen motivating me in the competitions. Almost superfluous is to thank all those contributors who worked in the past and in parallel on open source codes to bring tools and innovation to the scientific community that is nowadays regarded as taken for granted but we only get the full awareness when we have to submit our work, showing our weakness and limits, for public usage.

Bern, August 12 2016 Fredy Siegrist

Preface

A preface is not mandatory. It would typically be written by some other person (eg your thesis director).

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Bern, August 12 2016 Gina Cannarozzi

Abstract

The basic idea of this project was to implement codon bias indicies that have been listed and mathematically described before [**?**] together with important indices that came after and improvements. The standard genetic code, as we know it for most of the species, has 61 sense codons translating into 20 amino acids. These codons are usually decoded by a much larger number of different tRNAs, that in addition can ‘wobble’, that is some tRNAs are used to read more than one codon. For example, yeast has 41 tRNAs to decode its 61 sense codons. The choice of synonymous codon is not at all random, some codons are used frequently and others are hardly used at all [**?**]. In this project, we implemented several functions that compute indices of codon usage and applied them to coding sequences of the grass plant *Eragostis tef*. We set up an R project based on another package seqinr and transcribed the code for standard indicies that measure codon usage and bias such as the codon adaptation index (CAI), the Number of effective codons (Nc) and others from a code-library called CodonIndices of the language Darwin. Most importantly, functions were written for text- and graphical-based visualization of analyses based on codon usage that can be asses by demonstrations of the package. Further work may assess the significance of groups with extreme values of these indices using Gene-Ontology Enrichment analyses.
The package compromises of a set of functions to handle genomic data, to compute some codon indices, as well as tests, demonstrations, sample data and documentation thereof and provides a backbone for further implementation of a complete codon bias analysis instrument.

Key words:
R package; Codon Bias; Codon Usage; Statistical Analysis

Zusammenfassung

Das Grundprinzip dieses Projektes war das Implementieren von Kodon-Vorliebe Indexen die zuvor schon mathematisch aufgelistet wurden [**?**] sowie weitere Indexe die später dazu kamen und Verbesserungen in den schon Bestehenden. Der genetische Standardcode, wie wir ihn in den meisten Spezies antreffen, hat 61 kodierende Trinukleotide die in 20 Aminosäuren übersetzt werden. Diese Kodons werden üblicherweise von einer viel grösseren Anzahl an tRNAs gelesen, die zusätzlich auch noch 'wobbeln' können, das bedeutet, dass einige tRNAs benützt werden können um mehr als ein unterschiedliches Kodon zu lesen. Zum Beispiel reichen in der Hefe 41 tRNA um alle 61 Kodons zu dekodieren. Die Wahl der synonymen Kodons ist in der Tat alles andere als zufällig und einige Kodons werden weit öfters verwendet als Andere, welche zuweilen fast gar nicht benützt werden [**?**]. In diesem Projekt haben wir verschiedenste Funktionen eingebettet, die die Indexe für Kodon-Vorliebe ausrechnen und haben diese zur Berechnung von Gensequenzen der Graspflanze *Eragostis tef* eingesetzt. Wir haben ein R Projekt entwickelt, basiert auf einem Fremdpaket seqinr und haben den Kode für Standardindexe umgeschrieben welche die Kodon-Vorliebe ausrechnen wie zum Beispiel den Kodonadaptionsindex (CAI), die Anzahl effektiver Kodons (Nc) und Weitere von einer Kodesammlung frmit dem Namen CodonIndices der Programmiersprache Darwin.
Hauptsächlich wurden Funktionen geschrieben um die Text- und Bild-basierte Visualisierung der Analysen basierend auf der Kodon-Vorliebe darzustellen, welche durch die Demonstrationsfunktionen des Pakets aufgerufen werden können. Weiterführende Arbeiten können dann die Signifikanz von Gruppen mit extremen Werten für diese Indexe in genontologischen Anreicherungsanalysen untersuchen.
Das vorliegende Paket beinhaltet ein Satz an Funktionen zum Bearbeiten der genomische Daten, um einige Kodonindexe auszurechnen, sowie Test-, Demonstrationsfunktionen und Beispieldaten sowie eine Dokumentationen davon und bildet ein Skelett für weitere Einbindungen um ein vollständiges Kodon-Vorliebe-Analyse-Instrument zu erstellen.

Stichwörter:
R Sammlung; Kodon-Vorliebe; Kodon-Gebrauch; Statistische Analyse

Contents

List of Figures

List of Tables

Introduction

This document is written based on a Generic EPFL Thesis Template in LaTeX v. 5.57 and may be extended in the vignette section as it is distributed through GitHub to annotate the statanacoseq project and this chapter should contain the text elements that are used to build up a proper R vignette of the package. Meaning to give examples for the usage of the different indices and functions for biological interpretation with the code for repeating the *in silico* experiments by applying the code lines step by step or by copy pasting the entire code for a sample experiment which raw data can be found in a public database. At the moment the vignette section just list the included functions and data, as well as a short description as in the package overview documentation.

0.1 Aim

We are aiming to set up a tool to automatically calculate and evaluate statistical properties of codon bias, especially implementing the calculation for different codon bias indices and for genes of organisms where its genome is sequenced for the most part. That means that we have at least a partially sequenced genome and the information on expressed sequences or proteins. The most simple calculation of codon bias is the GC content of the third base of synonymous codons of a single gene or coding DNA sequence (CDS). This codon bias reflects neutral mechanisms like mutational selection for a high or low GC content in an organism. The CG content of the third coding base generally is around 40 percent in low GC-regions but may increase up to 80 percent in high GC-regions [**?**]. This GC3 value can be easily calculated by the coding sequence alone and has been associated with overall GC content of a genome [**?**].

Such mechanisms based on a single gene can be revised by incorporating species independent information of general translation efficiency and RNA folding. Moreover, instead of analyzing a single nucleotide position alone (mononucleotides), information on cytosine (C)-methylation can be taken into account and calculation on dinucleotides (GC) for mammals or quadnucleotide (GATC) or pentnucleotides (CCWGG) which harbors a degenerate nucleotide position that can be A or T. The succession of codons (bicodons or even tricodons) may of course also be of interest but may only refine codon bias indexes where information on the whole genome or even for species is taken into account.

When we aim for introducing a little bit more sophisticated indexes, they will depend on additional information such as, the complete transcriptome, the complete genome, gene expression, tRNA availability and protein properties.

One aspect for some codon bias measurements is the definition of the optimal codon, thus the one with the most desirable quality. This information is used to measure the frequency of optimal codons (Fop) or the P1 index, a measure of the influence of tRNA availability. Which of the codons is optimal can be manifoldly defined, such as predicted using the tRNA gene copy number as a indicator as tRNA levels correlate with gene copy number in general [**?**].

0.2 Teff genome

Eragostis tef is not only a grass but an important crop, that was neglected in scientific studies before the setup of the 'Teff Improvement Project" ten years ago at the University of Bern. Applied science generated improved plants that made it to the farmers in Ethiopia and research is ongoing to discover new break-through plants that show increased seed size, soil acidity, drought and herbic ide tolerance along with the improved yield and to maintain lodging resistance without being semi-dwarf. A valuable resource for plants genetics is also the sequencing and annotation of the tetraploid genome of Teff and the evaluation of its properties [**?**]. Therefore, we used as sample organism here, sequences from this genome to examine what obstacles one encounters by working with a almost totally sequenced but not yet fully annotated genome. By learning from its codon usage, variations generated by technologies already used in the 1960 for screening of genetic variations of agricultural plants combined with DNA sequencing technologies may improve the selection criteria for optimal mutants used for back crossing.

0.3 Biological aspects of Codon usage

The ribosome is coordinating the translation from messenger RNA (mRNA) to proteins in the cells together with tranlsational factors and aminoacyl-transfer RNAs (tRNA) and guanosinetriphosphate as substrate. The efficiency of translation is thereby defined by the mRNA quantity and stability, but also by the composition (concinnity, order and the distribution of codons, the bioavailability of the mRNA and the translational factors and much more. The bias for using particular codons preferentially must be differentiated

Figure 1: Photography of model grass plants of species *E. tef*

between a *codon bias*, which tends to use those codons that will lead to an optimal use of tRNA for having an optimal cost-benefit factor between a fast and a accurate protein synthesis (including aminoacid misincorporations and protein malfolding) and a *codon adaptation*, that is aiming to optimize the use of codons towards a enhanced translational efficiency and this is subject to a exchange trend towards more optimal codons. For estimation of the codon bias we can compute codon usage bu measuring the bias to use adaptive codons for example with the Codon Adaptation Index (CAI) that is implemented in our code. The relative adaptiveness (RA) has thereby be calculated form a separate set of sequences, usually from the usage of highly expressed genes (codon fitness values). For the moment that is the users task, but may be decided on gene expression data in the future in our package. In contrast, the Number of effective Codons (NEC) in our package measures only the degree of deviation from an uniform codon usage based on basic statistics, regardless of which codons are overrepresented [**?**]. Moreover the package could be expanded in the future to compute additional indices such as the S value, that measures the strength of selection for codon adaptation (S), to perform within-group correspondence analysis (WCA) of codon usage, give information on replication strand skew analysis, the GC skew index (GCSI), single statistics (S) or PCA to estimate the relative selection strength within a genome.

0.3.1 Genetic Code

Here is shown a table with the standard genetic code (from verified ORF from *E. teff*), generated with the first of the following functions:

```r
require(seqinr)
tablecode(latexfile="codontable.tex")
# to plot a codon table with ratios and frequencies with the overwriting function
  from statanacoseq
require(statanacoseq)
tablecode(mylist(whatout=1))
```

```r
require(statanacoseq)
require(seqinr)
verified <- mylist("/windows/R/Round3.all.maker.transcripts.verified.fasta", "/
  windows/R/Round3.all.maker.proteins.verified.fasta", whatout=1)
# the process is killed when using to many sequences, so it is divided in four
  parts and added together later
tabl <- uco(verified[[1]], 0, "eff")
for (n in 2:10000) { #length(verified)
 tabl <- tabl + uco(verified[[n]], 0, "eff")
}
tabl1 <- uco(verified[[10001]], 0, "eff")
for (n in 10002:20000) {
 tabl1 <- tabl + uco(verified[[n]], 0, "eff")
}
tabl2 <- uco(verified[[20001]], 0, "eff")
for (n in 20002:30000) {
 tabl2 <- tabl + uco(verified[[n]], 0, "eff")
}
tabl3 <- uco(verified[[30001]], 0, "eff")
for (n in 30002:41988) {
 tabl3 <- tabl + uco(verified[[n]], 0, "eff")
}
tabla <- tabl+tabl1+tabl2+tabl3
perc <- round(tabla/nuu*100,2)
(rbind(tabla, perc))
```

As the verified CDS from E. teff have their stop codon already chopped off, we don't count them in the codon table below.

TTT	Phe	231800	1.44%	TCT	Ser	231219	1.43%	TAT	Tyr	169761	1.05%	TGT	Cys	99497	0.62%
TTC	Phe	363171	2.25%	TCC	Ser	236162	1.46%	TAC	Tyr	250318	1.55%	TGC	Cys	181252	1.12%
TTA	Leu	101045	0.63%	TCA	Ser	234348	1.45%	TAA	Stp	0	0%	TGA	Stp	0	0%
TTG	Leu	239454	1.48%	TCG	Ser	155902	0.97%	TAG	Stp	0	0%	TGG	Trp	197407	1.22%
CTT	Leu	289697	1.79%	CCT	Pro	221214	1.37%	CAT	His	182484	1.13%	CGT	Arg	106245	0.66%
CTC	Leu	356286	2.21%	CCC	Pro	166959	1.03%	CAC	His	189362	1.17%	CGC	Arg	190954	1.18%
CTA	Leu	119917	0.74%	CCA	Pro	230207	1.43%	CAA	Gln	208855	1.29%	CGA	Arg	82144	0.51%
CTG	Leu	356616	2.21%	CCG	Pro	205050	1.27%	CAG	Gln	332504	2.06%	CGG	Arg	156854	0.97%
ATT	Ile	257976	1.60%	ACT	Thr	188031	1.16%	AAT	Asn	266263	1.65%	AGT	Ser	158479	0.98%
ATC	Ile	308634	1.91%	ACC	Thr	210048	1.30%	AAC	Asn	304586	1.89%	AGC	Ser	246403	1.53%
ATA	Ile	155768	0.96%	ACA	Thr	208202	1.29%	AAA	Lys	284060	1.76%	AGA	Arg	169467	1.05%
ATG	Met	378621	2.35%	ACG	Thr	151392	0.94%	AAG	Lys	549104	3.40%	AGG	Arg	226367	1.40%
GTT	Val	289284	1.79%	GCT	Ala	345407	2.14%	GAT	Asp	431250	2.67%	GGT	Gly	246669	1.53%
GTC	Val	291192	1.80%	GCC	Ala	377503	2.34%	GAC	Asp	412788	2.56%	GGC	Gly	374322	2.32%
GTA	Val	118050	0.73%	GCA	Ala	306736	1.90%	GAA	Glu	369243	2.29%	GGA	Gly	250072	1.55%
GTG	Val	361307	2.24%	GCG	Ala	289771	1.79%	GAG	Glu	580359	3.60%	GGG	Gly	227638	1.41%

Table 1: Table with standard genetic code

1 Setup of data and project

In this chapter we will discuss the setup of the project. The first step was to read in the teff genomic data in R in a way it can be processed by the *seqinr* package and be converted to work as a string sequences as required for compatibility with Darwin processes. The next step was to read in tRNA gene count for different genomes to estimate the tRNA availability in different genomes and to decide on the optimality of the codons used. To conclude the setup phase we implemented a graphical representation over genes to show a moving average of where in the gene optimal codons are used based on the fraction of codon tRNA gene counts to the maximal count for the amino acid to be translated.

1.1 Getting familiar with the data

Before the project was set-up the initial task was to gather information for calculating one of the first defined and most basic codon bias index called frequency of optimal codons (Fop). The optimality of a codon can be estimated by the usage in the given gene, the transcriptome or by accessing the numbers of tRNA genes found in the genome. As we have already the data of tRNA genes for the teff genome available, this was the choice to work with and to import the information for closely related species for mays and millet.

1.1.1 tRNA gene statistics and optimality of codons

First we addressed a public database containing the tRNA gene counts, after detecting them by tRNAscan and filtering the non-confident hits. Later tRNA gene numbers for all available eukariotic, bacterial, archaea and one viral genome was extracted from tRNAscan-SE v.1.3 run statistics of the GtRNAdb 2.0 (available at Genomic tRNA

Database 2.0, http://gtrnadb.ucsc.edu/GtRNAdb2/genomes/). [?] The first script for extracting the numbers was:

```
#03.01.2016 Master project read in stat files from view-source:http://gtrnadb.ucsc
  .edu/GtRNAdb2/genomes/eukaryota/

#reading the files
setwd("E:/R")
readstats <- function(fileno=2, chunksize=1) {
  statfile=dir(pattern=".stats$")[fileno]
  con <- file(statfile, "r", blocking = FALSE)
  for(i in seq(1,3500,chunksize)){
        d=scan(con,what="a",nlines=chunksize,sep="|", quote="",quiet=TRUE)
    if (length(d)>0 && d[1]==""){
      print(substr(d[2:(length(d)-1)], 2, 15) )
      reps <- as.numeric(gsub("[^0-9]","", d[2:(length(d)-1)]))
      tRNAs <- substr(d[2:(length(d)-1)], 2, 4)
      codons <- substr(d[2:(length(d)-1)], 6, 8)
        closeAllConnections()
      return(data.frame(tRNAs, codons, reps))
    }
  }
}
(Zmays <- readstats() )
eins <- function(x=c("Val","UUU","1")){optfactor <- as.numeric(x[3])/max(Zmays[
  tRNAs==x[1],3])}
optfact <- apply(Zmays, 1, eins)
isopt <- (optfact == 1)
Zmays.out <- data.frame(Zmays, optfact, isopt)
Zmays.out[Zmays.out$isopt==TRUE,-4]
```

The statistic file from a html page for mays is given below as demonstration:

```
tRNAscan-SE v.1.3 (March 2011) scan results (on host aero.cse.ucsc.edu)
Started: Thu Jul 26 22:25:36 PDT 2012

------------------------------------------------------------
Search Mode:                 Eukaryotic
Searching with:              tRNAscan + EufindtRNA -> Cove
Covariance model:            TRNA2-euk.cm
tRNAscan parameters:         Strict
EufindtRNA parameters:       Relaxed (Int Cutoff= -32.1)

Reporting HMM/2' structure score breakdown
------------------------------------------------------------

First-pass (tRNAscan/EufindtRNA) Stats:
---------------
Sequences read:         13
Seqs w/at least 1 hit:  13
Bases read:             2066432718 (x2 for both strands)
Bases in tRNAs:         2747160
tRNAs predicted:        28967
Av. tRNA length:        94
Script CPU time:        417.21 s
Scan CPU time:          1253.04 s
Scan speed:             3298.3 Kbp/sec

First pass search(es) ended: Thu Jul 26 22:54:15 PDT 2012

Cove Stats:
-----------
Candidate tRNAs read:     28967
Cove-confirmed tRNAs:     2249
Bases scanned by Cove:  3210632
% seq scanned by Cove:  0.1 %
Script CPU time:          117.99 s
Cove CPU time:            2336.55 s
Scan speed:               1374.1 bp/sec

Cove analysis of tRNAs ended: Fri Jul 27 00:06:48 PDT 2012

Summary
--------
Overall scan speed: 1001957.8 bp/sec

tRNAs decoding Standard 20 AA:              1455
Selenocysteine tRNAs (TCA):                 4
Possible suppressor tRNAs (CTA,TTA):        7
tRNAs with undetermined/unknown isotypes:   12
Predicted pseudogenes:                      771
                                            -------
Total tRNAs:                                2249

tRNAs with introns:         54

| Thr-GGT: 1 | Val-AAC: 3 | Val-CAC: 1 | Val-TAC: 1 | Ser-GCT: 1 | Arg-TCT: 1 | Leu-TAA: 2 | Asn-ATT: 1 | Ile-AAT: 1 | Met-CAT: 24 | Tyr-ATA: 1 | Tyr-GTA: 16 | Cys-ACA: 1 |

Isotype / Anticodon Counts:

Ala   : 122   AGC: 79      GGC:         CGC: 28      TGC: 15
Gly   : 55    ACC:         GCC: 33      CCC: 11      TCC: 11
Pro   : 59    AGG: 15      GGG: 1       CGG: 9       TGG: 34
Thr   : 58    AGT: 21      GGT: 13      CGT: 7       TGT: 17
Val   : 61    AAC: 25      GAC: 15      CAC: 16      TAC: 5
Ser   : 68    AGA: 13      GGA: 23      CGA: 7       TGA: 11      ACT:         GCT: 14
Arg   : 90    ACG: 45      GCG:         CCG: 6       TCG: 7       CCT: 12      TCT: 20
Leu   : 79    AAG: 30      GAG:         CAG: 14      TAG: 9       CAA: 20      TAA: 6
Phe   : 27    AAA: 2       GAA: 25
Asn   : 53    ATT: 2       GTT: 51
Lys   : 350                             CTT: 206     TTT: 144
Asp   : 45    ATC: 1       GTC: 44
Glu   : 59                              CTC: 30      TTC: 29
His   : 33    ATG: 4       GTG: 29
Gln   : 39                              CTG: 14      TTG: 25
Ile   : 79    AAT: 73      GAT:                      TAT: 6
Met   : 90                              CAT: 90
```

```
Tyr    : 28    ATA: 2      GTA: 26
Supres: 7                              CTA: 6      TTA: 1
Cys    : 30    ACA: 2      GCA: 28
Trp    : 30                            CCA: 30
SelCys: 4                                          TCA: 4
```

However this code had to be adapted because it is only reading in the tRNA genes with introns. Therefore, another function was implemented to read in the total tRNA gene count. Interestingly, by browsing some of the output data from tRNAscan for other species we discovered some that have a huge amount of the alternative usage of the opal stop codon "UGA" that is messing up with the display of the overview in a hierarchical plot for all the tRNA numbers even when standardized for each tRNA, we can see that some of the species show a much higher total number of tRNAs.

However, there is some inconsistency in the data base in order that a few of the genomes contain so many tRNA genes, that these are overcrowding the graphical representation in a heat map for example (see fig. 1.1).

If we plot the log values of the tRNA gene count and normalize by species (column) then we can see that some species have very low numbers of certain codons that is usually compensated by using another block of codons that are rare in the other kingdoms (see fig. 1.2). We also observe some species that have a high number of tRNAs where the isoacceptor remains unknown (codon 65).

We can observe that those run statistics, from vertebrates especially non-primate mammals, are littered with tRNA-derived repetitive elements with primary sequences very similar to real tRNAs. So they apply a non-unveiled post filter after the tRNAscan-SE on those genomes before depositing the predictions to GtRNAdb. Therefore, these had to be rebuild from scanning the headers of the fasta files by implementing yet an alternative function, that seems to be consistent with the new version of tRNAscan 2.0 and gave the best results. For that purpose the fasta file name had to be scanned in the index.html file and the fasta files were renamed according to the directory name to facilitate automation. This needed a helper function to find those fasta files on the GtRNAdb2 server. One issue was that genomes with names containing # or have to be treated as special cases, as even the browser fails to fetch the files. Therefore # was replaced with %23.

To read in the output file of tRNAscan for tef, a function readtRNAout was created that converts this file to a table resembling the ones created from GtRNAdb2.

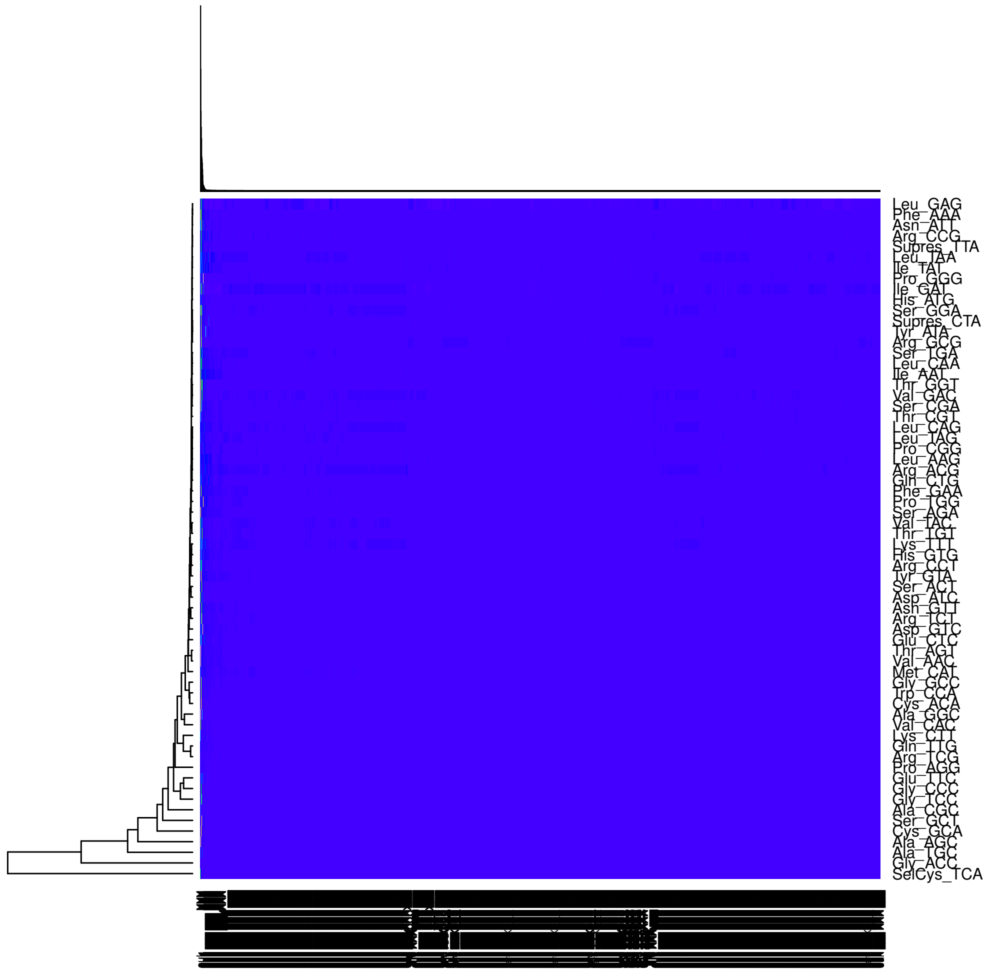

Figure 1.1: Heat map of tRNA gene counts normalized per row in all species

In contrast to other codon usage software, the statistics is now enhanced by an 65th line containing undefined aminoacid all other anticodons with degenerated base information are counted to the undefined species as they do in the summary page.

1.2 First functions and analyses

The project had now the first lines of code that were producing something interesting, however the project should be based on three pillars, namely the code itself, the testing functions and the documentation, including demonstration scripts. Therefore, the R package *statanacoseq* was initiated and the code that produced the first indices and

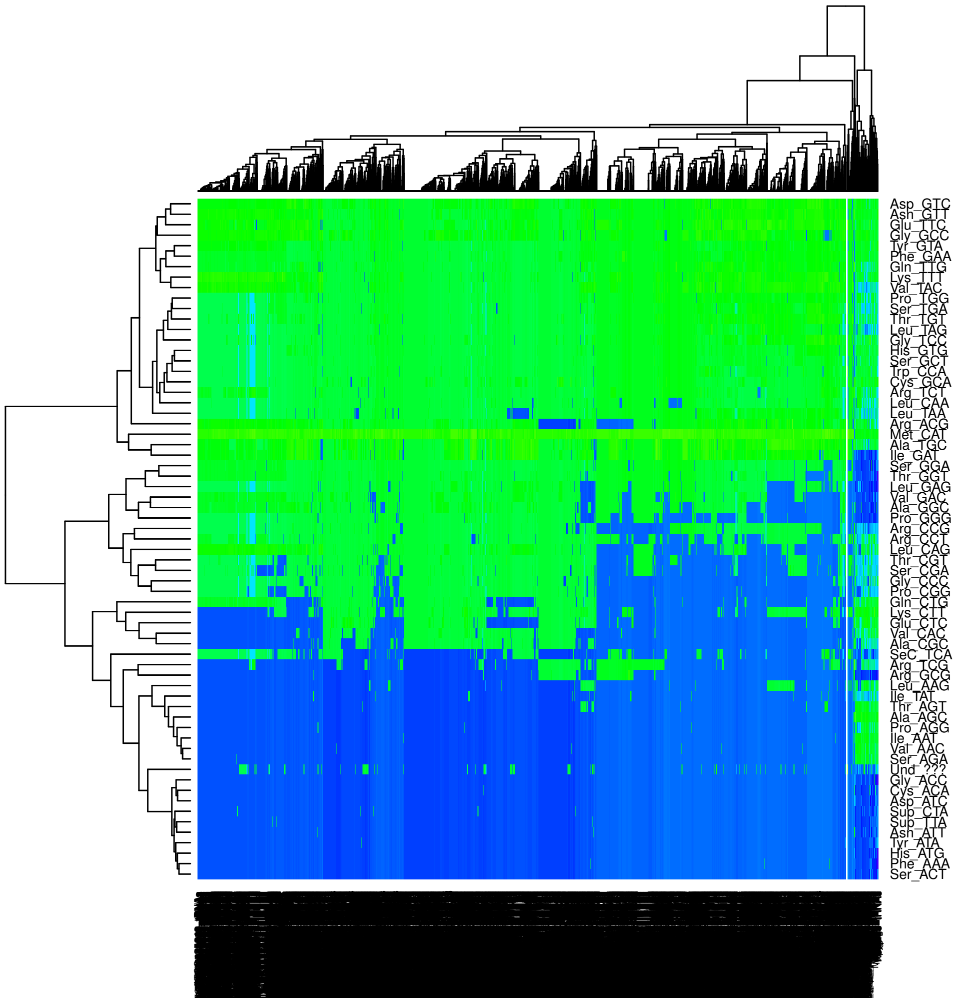

Figure 1.2: Heat map of logged(tRNA gene counts) normalized per column in all species

graphical representation rewritten to work as standalone package as well as handling local files. Before we introduce the setup of the package we show some results from the first analyses of the sample data.

1.2.1 Optimality of used codons

The first index to be implemented was the frequency of optimal codons. When the decision on which codons are optimal is made, the calculation of this index is quite basic. The decision can be made by taking the codon usage of the gene alone or the full genome or what we intended to implement first was to make the decision on the availability of tRNAs estimated from the number of tRNA genes that we find in the genome of *E. tef*. For generalization of the function we scanned available information on tRNA gene counts from a public database. However, the retrieved tRNA database turned out to make some problems as discussed above. In conclusion, some of the species had a much higher number of tRNA genes, so that a normalization was necessary to get an overview on the used codons.

To decide on optimal codons the number of genes for the given anticodon was divided by the maximal number of genes for a anticodon for the same amino-acid to give fraction to the optimal codon. The codon was accepted as optimal if this fraction was 1. In a list for every genome with available run statistic data, a frame with amino-acid, codon, number of genes, fraction to optimal codon, fraction to all codons, and the decision if it is a optimal codon was stored for the four superkingdoms separately. To get an overview of the optimality of the codon used in a gene the function plotOCU was implemented to show a moving average on the used codon optimality (see fig. 1.4).

One of the expected attributes of the moving average line for the optimality of the used codons is the so called codon ramp [**?**]. The codon ramp describes the phenomenon that rare codons are used at the start of genes. This hypothesis first mainly was proposed for genes with strong translation, but then it has been shown that rare codons at the N terminus increased expression in a general manner. However, when we analyzed some genes from teff, we couldn't observe a overrepresentation of rare codons at the start of the protein translation. When analyzing the moving average of the optionality factor of the codons for the tRNA genes, we don't really detect a codon ramp at the 5' terminus, but the first codon was always optimal. No wonder because there is only one codon for methionine, the start codon. Therefore we should only analyse the aminoacids that have a choice of anti-codons to use. However, the refinement of this code needs still to be implemented. Other issues during codon usage analysis was found in the data files of teff that is labeled as validated protein-transcript fasta files: first one of the proteins (Et_s4372-0.17-1) was truncated and didn't startet with methionine and was not corresponding to the truncated transcript, secondly the number of transcripts matched the number of proteins, but there was one orphan entry on each side (Et_s2692-0.26-1, Et_s14755-0.7-1), that had to be filled up with the corresponding entry of the other datafile. The files were then corrected and stored as a local copy, however this lead to the condensation of those huge fasta files to a sample data-set incorporated in the package itself that could be used further for demonstration of the functions without running in further issues.

1.2.2 GC content at codon sites

Another basic statistic that can be applied without looking at the full genome is the GC content at the different positions of the codons. If we print the mean GC content of the first two bases against the one at the third codon position for the sample sequences of teff, we can observed that they tend to follow each other there (see fig. 1.4). This function was implemented at last but shows exemplary how basic analyzes on custom genomes can be implemented in simple codes for comparing the genomic information.

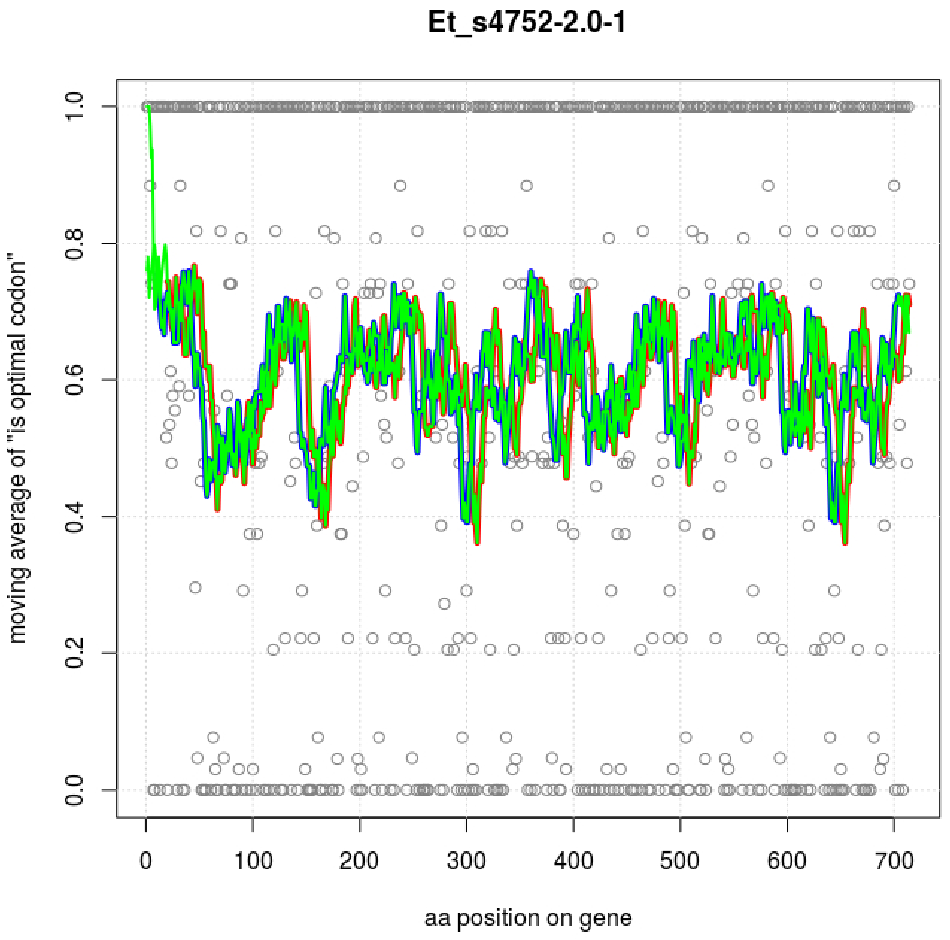

Figure 1.3: A sample figure (from plot OCU *demo(plotOCU)*, of statanacoseq, available at https://github.com/fredysiegrist/statanacoseq) showing the optimality score of each codon used for coding Similar to Eukaryotic peptide chain release factor subunit 1-3 (eRF1-3) gene of teff.

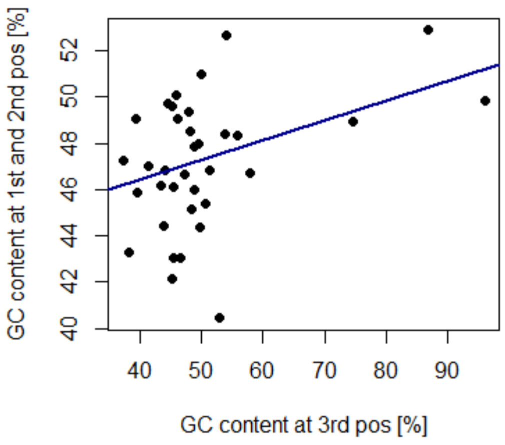

Figure 1.4: Mean GC content at first and second compared to third site of codon trinucleotide for sample sequences from mylist

2 R package setup and maintenance

In this chapter we will describe the generation of the R package structure, implementation in R Studio and version numbers on GitHub.

2.1 Generation of R package

The first intention was to generate a package named *statanacoseq* in the Linux version of R. The name is pretty long but has not been found to be used as R package name anywhere else, as far as a search in Google resulted.
Setting up of the R package has been done with the help of *devtools*, some few lines of code generate a backbone that then can be filled with the content and basically be packed after having put the first function in to an R file.

```r
# Construction of the first package
# set the working directory to the shared partition
setwd("E:/R/")
# setwd("/windows/R/")
# install.packages("devtools")
library("devtools")
# devtools::install_github("klutometis/roxygen")
library(roxygen2)
# Creation of empty package
create("statanacoseq")
# save function as "E:/R/statanacoseq/R/countcodonfreq.R"
setwd("./statanacoseq")
document()
setwd("..")
install("statanacoseq")
sessionInfo()
detach("package:statanacoseq", unload=TRUE)
```

In this case a simple function with the misleading name was *countcodonfreq* but was used to check the 'validated' fasta files by printing out the fraction of codons to aminoacids for the nucleotide and aminoacids with the same entry name.

```
countcodonfreq <- function(no=1, wdir="/windows/R/") {
    require(seqinr)
    list(myseq, myaa) <- mylist(wdir)
    print(length(myseq[[no]])/3/length(myaa[[attr(myseq[[no]], "name")]]))
}
```

As we meantioned in the chapter before, the fasta files had already undergone some error fixing and was now correctly processed by this function, what left this function more or less abandoned.
Another advantage of this function was, to check the correct interpretation of the roxygen documentation, that can be checked by addressing the help of the function in a internal window or as html style:

```
options(help_type = "text") # open help internally
options(help_type = "html") # open help in browser
?countcodonfreq
```

2.1.1 Editing of elementary files

Next, the DESCRIPTION file was manually edited, mentioning the copyright, version number and more basic information about the package. Besides of *devtools* packages other software, such as *Rtools* from CRAN and *roxygen* was needed to produce the first version of a functional package. However after some reformatting steps, the description of the package itself gave an error out and needs to be fixed later in order that the help call ?statanacoseq is working properly.

2.1.2 Generation of sample sequence data file

The *statanacoseq* software package is improved with a small set of sample sequences that are used for testing and demonstration for people who want just to see what it is actually doing and do not (yet) have sequences of there own or the idea of on which sequences from public datasets they want to work on. The *seqinr* package offers here extensive methods for fetching public sequenced and will not be further discussed here. To give the package a touch of our own or better said of the model species that we are working on in our research group, we decided to pack some teff genes and their translation as exemplary data. There are thousands of sequences from teff available that may feed in to a sample data set of about 20 genomic and matched protein sequences.

So we did a BLASTP 2.2.25 of the coding sequences from teff to the rice (*Oryza sativa Japonica*) proteome. The report of this search is given here:

```
  Database: IRGSP-1.0_protein_2011-12-05.fasta
    Posted date:  Mar 4, 2013  10:59 PM
  Number of letters in database: 13,848,002
  Number of sequences in database:  42,124

Lambda     K      H
   0.318    0.135    0.408

Gapped
Lambda     K      H
   0.267   0.0410    0.140


Matrix: BLOSUM62
Gap Penalties: Existence: 11, Extension: 1
Number of Sequences: 42124
Number of Hits to DB: 468,858,621,970
Number of extensions: 439567939
Number of successful extensions: 147953882
Number of sequences better than 1.0e-180: 18307
Number of HSP's gapped: 147741103
Number of HSP's successfully gapped: 21415
Length of query: 395
Length of database: 13,848,002
Length adjustment: 102
Effective length of query: 293
Effective length of database: 9,551,354
Effective search space: 2798546722
Effective search space used: 2798546722
Neighboring words threshold: 11
Window for multiple hits: 40
X1: 16 ( 7.3 bits)
X2: 38 (14.6 bits)
X3: 64 (24.7 bits)
S1: 41 (21.7 bits)
S2: 1622 (629.4 bits)
```

Then, we picked out manually some of the sequences that showed no variation in sequence length (indels) in both sequences and had about the same total entry length. Furthermore, the sequences should be fairly similar in the sequence (generally > 90% amino acid similarity). At the first look, there were not that many sequences that fitted those criteria, and the original idea of inserting only identical sequences had to be discarded soon, still a complete search for it has not been performed. The 37 sample sequences were then condensed in two new fasta file and the found similarity to rice protein was used to give a name to their sequences accordingly to the better studied model organism *O. sativa*.

These sequences can be used for testing of the functions that are implemented, however

it should be considered to also test on the sequences form the articles that describe the indices for the first time, to show that the implementation in this package give the same values, if there are improvements of the indices described in later publications, these should be made available by setting the attributes to the different descriptions of the indices.
For some of the functions there have been tests implemented that show an overview of different testing procedures, as testing for the availability and integrity of the GtRNAdb2 internet source, the correct computation of some index values or if warnings in codon sequences can be overcome by correcting those sequences. The tests can be run by the following two lines:

```
library(devtools)
test()
```

2.1.3 Maintenance with R Studio

To be more efficient in managing the project the project initiated with *devtools* was imported into R Studio, a software package that brings many functions for assisting with R code writing, documentation and communication with the version control system [**?**]. One of the first functions to be implemented there was ComputeCAI, where the original Darwin code is mentioned in the description and should be compared to the implementation in *seqinr*. Future improvements of the code should be made to maintain the possibility to change base data (tRNA mapping and frequencies) and make it universally usable even for species with a non-standard codon table to be maximal flexible in case existing implementations (in Darwin, *seqinr* or other 3rd party sofware) are not flexible enough.

2.1.4 Publishing on GitHub - Description of versions

After the package got the first functions to work independently on sample data from the package it was submitted to GitHub, a hosting service for software projects, for version control and to make it available for revision.
The origin is placed at https://github.com/fredysiegrist/statanacoseq.git. Since no version has been generated where the indices that are not implemented in Darwin has been drawn up, the version remained at 0.0.0.900x, marking it as under development. At this moment of initiation of writing this text the version is 0.0.0.9001 and will be set to 0.0.0.9002 for the evaluation of this work and 0.0.0.9003 after the corrections. Any further version will mark improvements to the code made post thesis submission.

The version 0.0.1 should mark the mile stone of achieving better performance than the

Darwin package, including the calculation of not-implemented functions in that code.

Version 0.1.0 should be the version that is distributed to the scientific community and following versions may be described in articles.

Version 1.0.0 will be defined after distribution to CRAN or bioconductor, if a bug free code that is consistent in it-self is created and will have some back-references from other packages.

3 Transcription of Darwin Code

In this chapter we summarize the transcription of Darwin codes and the adaptation of some function to the R environment.

3.1 Transcription of Darwin functions

One of the tasks was to set up a system that can handle the functions implemented in Darwin in the source compilation *CodonIndices* and process genetic files given from local fasta files, sequence files from public databases or sample files from the R package if no custom sequence is entered. The remaining problem is to understand how the variable *Entries* is build up from the database *db* and to mirror that function as we built up the R package on the existing R project SeqInR *seqinr* that uses a distinct different sequence database handling. Some other functions, where indices are calculated on the sequence information itself, where adaptable without dealing with the sequence database problem and were transcribed 1:1 to a function R code, however the loops are not acknowledging the functionality of the R environment.

3.2 R environement adaptation

To upvalue the code transcibed from Darwin, the function *ComputeNEC* was simplified to R language by eliminating the loops and introducing table and sapply function instead. The source code now looks pretty much like a R code that was setup in that language at first. Below we list this code for demonstration, the original code in Darwin can be found in the appendix.

```
ComputeNEC <- function(cds) {
  if(!(checkCDS(cds))) {stop("non valid CDS)", call.=FALSE)}
  else {
    cds <- tolower(cds)
    cod <- rep(0, times=64)
    names(cod) <- sapply(as.character(Tef$codons), reversecomplement)
    cod <- cod[c(-59, -60, -64)]
    aa <- rep(0, times=20)
    names(aa) <- levels(Tef[,1])[c(-16,-18)]
    cods <- count(s2c(cds), word = 3, by=3)
    cod[names(cod)] <- as.vector(cods[tolower(names(cod))])
    AA <- table(translate(s2c(cds)))
    aa[names(aa)] <- AA[a(names(aa))]
    NC <- rep(1, times=length(aa))
    names(NC) <- names(aa)
    Acods <- sapply(names(NC), function(i) sapply(as.character(Tef[,2]),
      reversecomplement)[Tef[,1]==i])
    contributors <- names(aa)[sapply(Acods, length)>1&aa>1]
    Nc <- sapply(contributors, function(i) {
     n <- sum(cod[unlist(Acods[i])])
     S <- sum(sapply(cod[unlist(Acods[i])], function(x) (x/n)^2))
     F <- (n*S-1) / (n-1)
     return(1/F)
     }
    )
    NC[contributors] <- Nc[contributors]
    return(sum(NC))
  }
}
```

4 Vignette

This document is a vignette for using the R package statanacoseq:

4.1 statanacoseq

4.2 Demos

Fop   Computes the Frequency of Optimal Codons in the Eragostis tef Sample Set
NEC   Computes the Overall Number of Effective Codons in the Eragostis tef Sample Set
plotOCU   Plots the Codon Optimality Score for Each Aminoacid in the Eragostis tef Sample Set with Moving Averages

4.3 Tests

checkCDNA   Tests if the sample CDS throws a warning for the missing stop codon and whether this can be rescued by adding a stop codon.
Fop   Tests if Frequency of optimal codons are calculated (with added opal stop codon) as initially set by comparing with a vector of correct Fop for the sample data.
NEC   Tests if the output of ComputeNEC is the same as for the 3rd party function CUB from *vhica* package.
NucleotideContent   Tests if NucleotideContent calculates the same value for a random sequence as would be expected intuatively.
species   Tests the external database if all species have all codons defined or if there is a change in GtRNAdb2.

4.4 Functions

4.4.1 Indices

ComputeCAI	Codon Adaptation Index
ComputeCAIVector	Codon Adaptation Index Vector
ComputeCarboneRA	Compute Carbone Relative Adaptiveness
ComputeFop	Frequency of Optimal Codons
ComputeGC3syn	GC content 3rd position of synonymous codons
ComputeGC12syn	GC content 1st and 2nd position of synonymous codons
ComputeNEC	Effective number of codons
RelativeAdaptiveness	Compute Relative Adaptiveness

Intermediator functions for indices

RSCU Relative	Synonymous Codon Usage

Void indices

ComputeCBI	Codon Bias Index
ScaledChiSquare	Scaled Chi-Square
SilentSiteComposition	Base Composition at Silent Sites

Database Function from Darwin left untranscribed

Entries	Entries

4.4.2 Helperfunctions

anticodoncount	Calculate Global Anticodon Usage
areopts	Optimality Decision for Codons of Custom Genome

checkCDS Checks	Integrity of CDS Sequence
CodonProb	Codon Probability table
CodonProbabilities	Codon Probabilities
CodonUsage	Table for Codon Usage and Optimality of Codons
FindHighlyExpressedGenes	Find Highly Expressed Genes
NucleotideContent	Nucleotide Content
reversecomplement	Reverse Complement Anticodon
SetupRA	Relative Synonymous Codon Usage

Graphical functions

plotOCU	Plot of Optimality Score for Codons in a Single Protein

3rd party functions

movingAverage	Calculating a Moving Average

Obsolete functions

countcodonfreq	Count Codon Frequency

Data Handling

mylist	My List of Nucleotide-Sequences and Aminoacid-Sequences
readfasta	Processed tRNA Containing Fasta File Headers to Count Anticodons
readintronic	Read tRNA Intronic Gene Count Statistics
readstats	Read tRNA Gene Count Statistics
readtRNAout	Read tRNA Gene Count Statistics from custom file
searchfafile	Search and Return a Fasta File Name from Partial Name

4.5 Sampledata

4.5.1 External Data

Etef.sample.protein.fasta	Selected Teff Protein sequences, matching selected CDS
Etef.sample.transcript.fasta	Selected Teff CDS, closely related to rice sequences
seq30469.out	tRNAscan-SE 2.0 online output file for teff tRNA sequences is used as example imput for readtRNAout

4.5.2 Datasets

aa_ac	Aminoacids and Anticodons
GtRNAdb2species	Viruses, Eukaryota, Archaea, Bacteria Species Names on GtRNAdb2
Tef	Optimal Codon Table for Eragostis tef Decided on tRNAscan-SE count of tRNA genes in sequenced genome
veab	Viruses, Eukaryota, Archaea, Bacteria Species File Names on GtRNAdb2
veabfa	Viruses, Eukaryota, Archaea, Bacteria Species Fasta File Names on GtRNAdb2

5 Outlook

In this chapter we will give a brief overview on which indices are meant to be implemented in the future, what results could be generated by applying them and what the future for this package looks like.

5.1 Codon Bias Indices

There are dozens of codon bias indices that yet have to be implemented in the package. The original Darwin code has alredy defined some of the most wanted indices yet to be implemented. Among them are as example normalized translational efficiency (nTE) index or the tRNA adaptation index (tAI) [**?**].
Table 5.1 gives an overview on the implemented, the ones which have a void function backbone and the non-implemented functions tab:CodonBiasIndices. The indices are taken from a review [**?**] and can be found in other reviews of codon indices [**?**].

To give those indices more power, attributes should be set to choose from different version of those indices that have been improved over time. Test should be done with the results from the original papers to confirm the down-gradability of the index functions.

5.2 Biological meaning and statistics

The basic idea of a statistical analysis package for codon analysis is to combine the most frequently used codon indices in one software to evaluate which of them is best performing regarding to the availability (quality) and the biological properties for the given species. By putting it into the environment of the statistical open source program R following analysis steps as correlation to Gene Ontology (GO) tags can be done without

having to deal with data handling to other software. The open structure of R codes also allows to use other implementation of algorithms already developed for R as we here used the functions RSCU from *seqinr* package for example. That allows other scientist to work with there own interpretation of new indices or better implementation of algorithms to calculate them. Furthermore, it is designed to adapt collaborators or third party code in the package, because every function has its own file and comes with the description of the function.
The basic biological problem to solve, as soon as all the major indices are implemented, is to find correlations of indices or clusters with other biological information such as gene expression (that is already part of the calculation of some indices), intracellular location, stress responses, enzymatic families, pathways, development stages and many more. One possible approach is to correlate the index properties to GO attributes and to find out whether there are indices that are significantly representing some of them, in order that a prediction of the function or other properties of unknown proteins can be predicted.

Such GO enrichment analyses then may be directly used to find the most appropriate index for the given species that may be used for interpretation or prediction of unknown properties of new genes. It may also be used to estimate the impact on translation efficiency for a given mRNA or the amount of functional proteins produced in a context of mutational discrepancies among newly bred species. Furthermore, indices may be found to find new members to sets of genes that can be found in stress resistant or intolerant species and maybe in a distant future predict beneficiary mutations that lead to stress tolerance.

5.3 Package development

The future development of the package is heavily dependent of the scientific interest in a package that computes all the codon bias indices and to find the best correlation to the phenotypic attribute to be investigated on. Once a working package is created it may be submitted to an official R package depository at CRAN or Bioconductor. However, at the moment first the missing indexes have to be implemented, some bugs eliminated and the vignette completed to attract interest in the scientific community.

Table 5.1: Measures of codon bias

name	implemented	backbone	missing
Relative codon frequencies			
g_c / g_{ac} global codon frequency	x*		
f_{ac} frequency of codon c encoding amino acid a	x*		
r_{ac} relative synonymous codon usage (RSCU) for codon c and amino acid a	x*		
w_{ac} relative adaptiveness	x		
Measures based on reference			
Fop Frequency of optimal codons	x		
CBI Codon Bias Index		x	
B / E Codon Usage Bias (expression)			x
CEC Codon-Enrichment Correlation			x
Measures based on the geometric mean			
P Codon Preference			x
CAI Codon Adaptation Index	x		
rCAI Relative Codon Adaptation Index			x
RCB Relative Codon Usage Bias			x
RCA Relative codon adaptation			x
FFT Autocorrelation			x
GCB Iterative approach to determining codon bias			x
tAI tRNA adaptation index			x
nTE Normalized translational efficiency			x
Measures based on deviation from an expected distribution			
CPB Codon-Preference Bias Measure			x
MCB Maximum-likelihood Codon Bias			x
B_a Bias of an individual amino acid			x
χ^2 Scaled **chi squared** statistic		x	
Ew Weighted sum of relative entropy			x
SCUO Synonymous codon usage order			x
Measures focusing on tRNA interaction			
P1 Influence of tRNA availability			x
P2 Bias for anticodon–codon interactions of intermediary strength			x
TPI tRNA-pairing index			x
Measures based on intrinsic properties of codon usage			
GC3s Base composition at silent sites	x*		
Nc Effective number of codons	x		
MILC Measure independent of length and composition			x
MELP MILC Expression Level Prediction			x
ICDI Intrinsic codon bias index			x
HK Multivariate statistical method by Hey and Kliman			x
S2str Strength of mRNA secondary structure			x
ER Evolutionary rate			x
v(c) Codon volatility			x
PLS Partial least squares regression			x
SUMBLE Synonymous codon usage bias maximum-likelihood estimation			x
SEMPPR Stochastic evolutionary model of protein production rate			x
Measures for total codon usage - genome wide			
tot. Codon using			x
D_{mean} **Mean** dissimilarity index			x
BC6 base composition of 6mers *(new)*			x
Indices measuring amino acid usage			
GRAVY Grand averages of hydropathy			x
AROMA Aromaticity score			x
k_s Rate of protein decay			x

* needs proper implementation

6 R package on GitHub

This chapter gives a short summary on how the software can be installed on a system with a recent installation of the software R. The project is hosted on GitHub and freely available for the scientific community interested in it: https://github.com/fredysiegrist/statanacoseq

To install the R package on a R version > 3.3.0 the following lines of codes are sufficient to load the package:

```r
# install 3rd party software for communicating with GitHub
install.packages("devtools")
library(devtools)
# install software package of master thesis
install_github("fredysiegrist/statanacoseq")
library(statanacoseq)
```

To run a short demo on some of the functionalities one can now enter the following lines to test the package installation.

```r
demo(plotOCU)
demo(Fop)
demo(NEC)
```

A Appendix - the Darwin code for codon indices

In this appendix we list the CodonIndices file of the Darwin software package that was used to transcribed most of the functions of statanacoseq package.
Data Analysis and Retrieval With Indexed Nucleotide/peptide sequences homepage

```
#
# Several indices for codon usage.
#
#                           Alexander Roth (2005-2007)


# Frequency of optimal codons (Ikemura 1981).  AR (April 2007)
ComputeFop := proc(d:string)
  cu:=CodonUsage();
  aviod:={op(AToCodon('$')),
          op(AToCodon('M')),
          op(AToCodon('W'))};
  xop:=0; xnon:=0;
  for i to length(d) by 3 do
    c := d[i..i+2];
    if not member(c,aviod) then
      if c=cu[CodonToInt(c),1,1] then
        xop:=xop+1;
      else
        xnon:=xnon+1;
      fi;
    fi;
  od;
  xop/(xop+xnon);
end:


# Effective number of codons* (Wright 1990, *Fuglsang 2004).  AR (April 2007)
ComputeNEC := proc(d:string)
  cod:=CreateArray(1..64);
  aa:=CreateArray(1..20);
  aviod:={op(AToCodon('$'))};
  count:=0;
  for i to length(d) by 3 do
    c := d[i..i+2];
    if not member(c,aviod) then
      ai:=CodonToInt(c);
      ci:=CodonToCInt(c);
      cod[ci]:=cod[ci]+1;
      aa[ai]:=aa[ai]+1;
      count:=count+1;
    fi;
  od;

  Nc:=0;
  for i to 20 do
    Acods := IntToCInt(i);
    k := length(Acods);
    if k<2 then Nc := Nc + 1; next; fi;
    n := sum([seq(cod[Acods[x]], x=1..k)]);
    S := sum([seq((cod[Acods[x]]/n)^2, x=1..k)]);
    F := (n*S-1) / (n-1);
    Nc := Nc + 1/F;
  od;
  Nc;
end:

#one:=''; all:='';
#for x to 3 do
#for i to 20 do for j to length(IntToCodon(i)) do
#    all:=all.IntToCodon(i)[j];
#    one:=one.IntToCodon(i)[1];
#od od od;

# Nucleotide content.  AR (2006)
NucleotideContent := proc( ; tD:{string, Entry}, pos=[1,2,3]:list(posint)) -> list
  ;
  o := CreateArray(1..4);
  n:=0;
  if not assigned(tD) then
    for z to DB[TotEntries] do
      d:=SearchTag(DNA, Entry(z));
      for i1 to length(d)-max(pos) by 3 do for i2 in pos do
          i:=i1+i2;
          n:=n+1;
          o[BToInt(d[i])] := o[BToInt(d[i])]+1
      od od;
    od;
  else
    if type(tD, Entry) then d:=SearchTag('DNA', tD)
    else d:=tD fi;
    for i1 to length(d)-max(pos) by 3 do for i2 in pos do
        i:=i1+i2;
        n:=n+1;
        o[BToInt(d[i])] := o[BToInt(d[i])]+1
    od od;
  fi;
  return(o/n);
end:


# G+C content 3rd position of synonymous codons.  AR (April 2007)
ComputeGC3syn:= proc(td:string)
  if member(td[-3..-1], AToCodon('$')) then d:=td[1..-4] else d:=td fi; # remove stop codon
  o := CreateArray(1..4);
  n:=0;
  for i to length(d) by 3 do
    c:=d[i..i+2];
    if length(IntToCInt(CodonToInt(c)))>1 then
      n:=n+1;
      oi:=BToInt(c[3]);
      o[oi] := o[oi]+1
    fi;
  od;
  o:=o/n;
  return(o[2]+o[3]);
end:


# Base composition at silent sites.
SilentSiteComposition := proc(d:string)
  # to be implemented
end:


# Scaled Chi-Square.
ScaledChiSquare := proc(d:string)
  # to be implemented
end:


# Codon Bias Index (CBI) (Bennetzen and Hall 1982).
ComputeCBI :=  proc(d:string)
  # to be implemented
end:


# Relative synonymous codon usage.  AR (2007)
RSCU := proc(;d:string)
  if nargs>0 then
    cc:=CodonCount(d);
  else
    cc := CodonCount();
  fi;
  rscu := CreateArray(1..64);
  for i to 64 do
    s:=0;
    syn:=IntToCInt(CIntToInt(i));
    l:=length(syn);
    for j in syn do
      s:=s+cc[j];
    od;
    if s=0 then next fi;
    rscu[i]:=cc[i]/(s/l);
  od;
  rscu;
end:


# Compute CAI, the Codon Adaptation Index (Sharp and Li 1987).
# Markus Friberg and Alexander Roth (Dec 2005)
ComputeCAI := proc(DNA:{string, Entry})
  # check global variables and scan arguments
  if not assigned(RA) then
    error('Error in ComputeCAI: RA not assigned, use e.g. SetupRA(yeast);') fi;
  if type(DNA, Entry) then dna:=copy(SearchTag('DNA', DNA))
  else dna:=DNA fi;
  UseCodonProb := false;
  for i from 2 to nargs do
    if length(args[i]) = 2 and args[i, 1] = 'UseCodonProb' then
      UseCodonProb := args[i, 2]
    else
      error('Unknown argument ', args[i]);
    fi;
  od;
  # compute cai
  w := 0;
  n := length(dna)/3;
  for j to length(dna) by 3 do
    cint := CodonToCInt(dna[j..j+2]);
    codprob := If(UseCodonProb, CodonProb[cint], 1);
    if CIntToA(cint) <> '$' then    # don't consider stop codons
    w := w + ln(codprob * RA[cint]) fi;
  od;
  exp(1/n * w)
end:

ComputeCAIVector := proc(e:Entry)
  if not assigned(RA) then
    error('Error in ComputeCAI: RA not assigned, use e.g. SetupRA(yeast);') fi;
  dna := SearchTag('DNA', e);
  wa := CreateArray(1..20);
  na := CreateArray(1..20);
  for j to length(dna) by 3 do
    cint := CodonToCInt(dna[j..j+2]);
    a := CIntToInt(cint);
    if a <= 20 then
      wa[a] := wa[a] + ln(RA[cint]);
      na[a] := na[a]+1;
    fi;
  od;
  res := CreateArray(1..21);
  for i to 20 do
    res[i] := If(na[i]=0, 'NA', exp(1/na[i] * wa[i])) od;
  res[21] := exp(1/sum(na) * sum(wa));
  res
end:

```

```
SetupRA := proc(s:string)
  global RA, CodonProb;
  CodonProb := [0.9910, 0.9750, 0.9793, 0.9691, 0.9318, 0.9268, 0.8389, 0.9636,
                0.9622, 0.8830, 0.8633, 0.9223, 0.9179, 0.9598,      1, 0.9825,
                0.9720, 0.8660, 0.8883, 0.9223, 0.9530, 0.8176, 0.7371, 0.9253,
                0.5895, 0.5874, 0.4657, 0.8154, 0.9370, 0.7825, 0.8817, 0.9173,
                0.9832, 0.9475, 0.9284, 0.9727, 0.9341, 0.9249, 0.8082, 0.9614,
                0.8887, 0.8914, 0.8059, 0.9475, 0.9074, 0.9249, 0.9072, 0.9719,
                     0, 0.9460,      0, 0.9436, 0.9328, 0.9347, 0.8408, 0.9737,
                     0, 0.7542, 0.8870, 0.8534, 0.9722, 0.9748, 0.9819, 0.9703]:
  if s = 'yeast' then  #based on the original CAI paper by Shart and Li
    RA := [0.135,     1,     1, 0.053, 0.012,     1, 0.006, 0.921,
               1, 0.031, 0.003, 0.021, 0.003,     1,     1, 0.823,
               1,     1, 0.007, 0.245,     1, 0.009, 0.002, 0.047,
           0.002, 0.002, 0.002, 0.137, 0.039, 0.003, 0.003, 0.006,
               1,     1, 0.016, 0.554, 0.015, 0.316, 0.001,     1,
           0.002, 0.020, 0.004,     1, 0.002, 0.831, 0.018,     1,
               1,     1,     1, 0.071, 0.036, 0.693, 0.005,     1,
               1, 0.077,     1,     1, 0.117,     1,     1, 0.113]:
  elif s = 'yeast2perc' then
    RA := [0.1277, 1, 1, 0.08078603, 0.02564103, 0.9501, 0.01, 1, 1, 0.0313253,
           0.00159109, 0.03253012, 0.00383632, 1, 1, 0.8325, 1, 1, 0.00572082,
           0.2407, 1,  0.00333704, 0.01, 0.08676307, 0.01, 0.01, 0.01, 0.1687,
           0.04752971, 0.01, 0.00125078, 0.00562852, 1, 1, 0.01611863, 0.6806,
           0.00955593, 0.3007, 0.00337268, 1, 0.00286369, 0.01890034, 0.00229095,
             1,
           0.00169635, 0.7625, 0.01526718, 1, 1, 1, 1, 0.08910891, 0.02409639,
             0.6892,
           0.00120482, 1, 1, 0.05555556, 1, 1, 0.1451, 1, 1, 0.1765]:
  elif s = 'yeast1perc' then
    RA := [0.07619048, 1, 1, 0.04887218, 0.01160093, 1, 0.01, 0.9722, 1,
           0.02690583, 0.01, 0.02690583, 0.00233645, 1, 1, 0.7664, 1, 1, 0.01,
           0.2192, 1, 0.00190114, 0.01, 0.07984791, 0.01, 0.01,
           0.01, 0.1402, 0.03326180, 0.01, 0.00107296, 0.00536481, 1, 1,
           0.01156069, 0.6220, 0.00392542, 0.2826, 0.00098135, 1, 0.01,
             0.01452282,
           0.00207469, 1, 0.01, 0.8253, 0.02028081, 1, 1, 1, 1, 0.06722689,
           0.01345291, 0.7691, 0.01, 1, 1, 0.08333333, 1, 1, 0.1159, 1, 1,
             0.1405]:
  elif s = 'yeast05perc' then
    RA := [0.06239168, 1, 1, 0.025, 0.004329, 1, 0.01, 0.8095, 1,
           0.01877934, 0.01, 0.00938967, 0.00452489, 1, 1, 0.7285, 1, 1, 0.01,
           0.1574, 1, 0.01, 0.01, 0.04961832, 0.01, 0.01,
           0.01, 0.08362369, 0.02471910, 0.01, 0.01, 0.00449438, 1, 1,
           0.01518987, 0.5362, 0.01, 0.2367, 0.00189394, 1, 0.01, 0.00852878,
           0.00426439, 1, 0.01, 0.8644, 0.01261830, 1, 1, 1, 1, 0.05263158,
           0.00938967, 0.7793, 0.01, 1, 1, 0.1132, 1, 1, 0.07865169, 1, 1,
           0.08415842]:
  elif s = 'yeasttop24protexpr' then
    RA := [0.3403, 1, 1, 0.3230, 0.1646, 0.6951, 0.04268293, 1, 1, 0.09117647,
           0.04307692, 0.1059, 0.07407407, 0.6931, 1, 1, 1, 1, 0.1014, 0.7321, 1,
           0.03353659, 0.00914634, 0.1982, 0.01, 0.00923077, 0.01, 0.2308,
           0.1548, 0.01092896, 0.04007286, 0.07103825, 1, 0.9475, 0.1169, 1,
             0.1366,
           0.3707, 0.02764228, 1, 0.03382353, 0.06470588, 0.02647059, 1,
             0.07712766,
           0.7048, 0.09574468, 1, 1, 1, 1, 0.4009, 0.1324, 0.6029, 0.02941176, 1,
             1,
           0.07865169, 1, 1, 0.357, 1, 1, 0.3927]:
  elif s = 'yeasttop24mrnaexpr' then
    RA := [0.1286, 1, 1, 0.08292683, 0.04761905, 1, 0.04761905, 0.9864, 1,
           0.03783784, 0.00904977, 0.02702703, 0.00621118, 1, 1, 0.677, 1, 1,
           0.00625, 0.17, 1, 0.00485437, 0.01941748, 0.09223301, 0.01, 0.00452489,
           0.01, 0.2081, 0.04885057, 0.00862069, 0.01, 0.01436782, 1, 1,
           0.00914634, 0.6516, 0.01081081, 0.327, 0.01, 1, 0.0080429, 0.02144772,
           0.00268097, 1, 0.01754386, 0.8816, 0.01754386, 1, 1, 1, 1, 0.08163265,
           0.07027027, 0.6757, 0.00540541, 1, 1, 0.125, 1, 1, 0.1782, 1, 1,
             0.1019]:
  elif s = 'carbone' then
    RA := ComputeCarboneRA();
  else
    error('Error in SetupRA: not yet implemented for that organism')
  fi;
end:


ComputeCarboneRA := proc( ; t=0.01:nonnegative, initfrac=1:nonnegative, iterfrac
  =0.5:nonnegative, mode:string)
  global RA;
  if not assigned(DB) then error('DB must be assigned') fi;
  x := 1;  # fraction of the sequences used to compute RA in this iteration
  AllGenes := [seq(i, i=1..DB[TotEntries])]:
  genes := Shuffle(AllGenes)[1..round(initfrac * DB[TotEntries])]:
  bestCorr := 0;
  cai := CreateArray(1..DB[TotEntries]):
  while length(genes) / DB[TotEntries] > t do
    RA := RelativeAdaptiveness(genes);
    for i to DB[TotEntries] do
      dna:=SearchTag('DNA',Entry(i));
      if SearchString('X', dna)<>-1 then next fi;
      cai[i] := ComputeCAI(dna) od;
    x := x * iterfrac;
    res := transpose([AllGenes, cai]):
    if mode='reverse' then
      res := transpose(sort(res, res -> res[2])):
    else
      res := transpose(sort(res, res -> -res[2])):
    fi;
    genes := res[1][1..round(x * DB[TotEntries])]:
  od;
  RA
end:


RelativeAdaptiveness := proc(entries:list(posint))
  CodonCounts := CreateArray(1..64);
  for i in entries do
    dna := SearchTag('DNA', Entry(i));
    for j to length(dna) by 3 do
      cod := CodonToCInt(dna[j..j+2]);
      if cod=0 then next fi;   # to avoid XXX
      CodonCounts[cod] := CodonCounts[cod]+1;
    od;
  od;
  RA := CreateArray(1..64);
  aa := 1;
  for aa to 20 do
    codons := IntToCInt(aa);
    counts := [seq(CodonCounts[i], i=codons)];
    freqs := counts / sum(counts);
    for i to length(codons) do
      cod := codons[i];
      RA[cod] := freqs[i] / max(freqs);
    od;
  od;
  for i to length(RA) do       # set minimum RA value to 0.01
    if RA[i] = 0 then
      RA[i] := 0.01 fi od;
  for i in AToCInt('$') do     # set RA value of stop codons to 1
    RA[i] := 1; od;
  RA
end:

# for each codon, compute the probability that it occurs at least once in a gene
CodonProbabilities := proc()
  res := CreateArray(1..64);
  for e in Entries() do
    occurs := CreateArray(1..64);
    dna := SearchTag('DNA', e);
    for c to length(dna) by 3 do
      cint := CodonToCInt(dna[c..c+2]);
      occurs[cint] := 1;
    od;
    res := res + occurs;
  od;
  res / DB[TotEntries]
end:


FindHighlyExpressedGenes := proc( ; n=100:integer, tag='PROTEXPR':string)
  # tags: 'PROTEXPR' 'MRNAEXPR'
  expr := CreateArray(1..DB[TotEntries]);
  for i to DB[TotEntries] do
    ex := sscanf(SearchTag(tag, Entry(i)), '%f');
    if ex <> [] then
      expr[i] := op(ex) fi;
  od;
  sorted := sort(expr);
  limit := sorted[length(sorted)-n+1];
  genes := [];
  for i to DB[TotEntries] do
    if expr[i] >= limit then
      genes := append(genes, i) fi od;
  genes
end:
```

Fredy SIEGRIST
Schnbergweg 11
CH-3006 Bern
+41 31 511 0512

Objective: Master in Bioinformatics and Computational Biology

EDUCATION

2016 (ongoing), Universities of Bern and Fribourg, Switzerland, Master in Bioinformatics, English.

2011, Apr 13, University of Basel, Switzerland, PhD Cell Biology, English.

2003, Oct 24, University of Bern, Switzerland, Master Biochemisty, German.

1999, Jun 17, Gymnasium Bern-Kirchenfeld, Switzerland, Matura type C, German.

ACTIVITIES

Editorial Acitivities and Teaching

Served as referee for: BMC Research Notes (BioMed Central), International Journal of Interferon, Cytokine and Mediator Research (Dove Press).

Modular course in Clinical Medicine for bachelor students (University of Basel) Tracking down genes modern genetics in clinical research

Administration

Administrative Board Member, Moosbad Immobilien AG, Emmenmatt, Switzerland

Publication Record

h index / Sum of the Times Cited (Web of science)

5 / 317

Computer Experience

Programming Langugaes (OO): R, PHP5, Python, Java. Programming Langugages (procedural, script): Unix-Bourne-Shell / bash, DOS-Batch, GW-BASIC. Operating Systems: Windows, Ubuntu, MAC-OS, MS-DOS. Software: MS Office / LibreOffice, Oracle VM Virtual Box CMS: OpenText, Joomla, Adobe Experience Manager, LSF / SGE Networks: Windows Server, Novell NetWare Databases: SQL / MySQL, MS Access

Work Experience

Publisher (CMS), Webeditor, IT-Support, Brewery Assistant.

EMPLOYMENT

Publisher, Federal Food Safety and Veterinary Office, Liebefeld, Switzerland 2016, Feb - Mai

Scientific Associate, Clinic of Nephrology, Bern University Hospital, Switzerland 2014, Feb - Jul

Post-doc, Laboratory of Virology, University Hospitals of Geneva, Switzerland 2011, Mar - 2013, Feb

PhD, Molecular Medicine and Toxicology, Roche, Basel, Switzerland 2006, Mar - 2010, Sep

Trainee, Functional Genomics, Novartis, Basel, Switzerland 2004, Apr - 2005, Mar

SCIENTIFIC AFFILIATIONS

2016 International Society for Computational Biology
2008 - 2014 International Society for Interferon and Cytokine Research

PROJECT SUMMARIES

Short-term scientific associate, University Hospital of Bern *Department of Nephrology, Hypertension and clinical Pharmacology; Prof B. Frey Department of medical Oncology; Dr M. Zweifel* Genome-mutations in the androgen-receptor ligand-binding domain in mamma- and ovarian carcinomas
Mutations in the androgen-receptor ligand-binding domain is addressed by next-generation sequencing in mamma-, ovarian- and prostate-cancer samples. Driver mutations will be analysed in-vitro for sensitivity to dihydrotestosteron derivatives in reporter assays. *Department of Nephrology; Prof U. Huynh-Do* Genomic analyses of chronic hypoxia exposed fetal kidneys
Kidney from mice embryos hold in a hypoxia chamber were analysed by microarrays and significant genes are verified by qPCR, in-situ hybridization and immunohistochemistry to validate expression. Epigenetic DNA status (MetC and hMetC) of candidate genes will be assessed by MeDIP-qPCR.

Post-doctoral position, University Hospitals of Geneva *Laboratory of Virology; Prof L. Kaiser and Dr C. Tapparel* Small RNA sequencing of rhinovirus infections
Deep sequencing data from small RNA Illumina libraries were analysed in rhinovirus infected cell culture samples, small viral RNAs are detected and human miRNA quantified. HeLa cells were infected with different rhinovirus types. Viral RNA fragments and human miRNAs were analysed with northern blots, primer extension and rapid amplification of cDNA ends assays. Impact of RNA fragments on viral replication and translation was addressed with quantitative PCR, Luciferase and immuno-fluorescence.

PhD Thesis, F. Hoffmann-La Roche AG *Molecular Medicine Laboratories; Prof U. Certa* SOCS proteins in IFN silencing
Function of SOCS Proteins in IFN signalling was studied by gene expression analysis with semi quantitative RT- qPCR. Selected candidates were cloned in a mammalian expression vector and fusion proteins with SNAP-tag (Covalys) were cloned and analysed with fluorescence microscopy. Stable SOCS expressing cell lines were generated and characterized for their interferon response by gene and miRNA expression microarrays and statistical analysis with R/Bioconductor. Bimolecular fluorescence complementation with eYFP and STAT1/2 fusion proteins was insensitive to STAT activation by IFN, but localization of STAT fusion proteins as for untagged dimers.
Cell-to-cell transfer of a proliferation control protein (IFITM3)
The transfer of IFITM3 proteins from a generator cell to a recipient cell was assessed by fluorescence microscopy, eYFP marked cell lines were sorted by FACS and analysed for protein transfer by immunoblotting. Protein transfer and proliferation assay (metabolite calorimetry, BrdU ELISA, FACS) excluded transfer of cytostatic effect. Phylogenetic analyses demonstrated recent IFITM gene development.

Traineeship, Novartis Pharma AG, NIBR, Basel *Functional Genomics Group; Dr F. Natt* siRNA stability in biological fluids
Obstacles for siRNA therapeutics are siRNA delivery and the short half-life of non-modified oligonucleotides. I have established a method for rapid analysis of degradation and demonstrated benefit of novel siRNA modifications (especially at their 3 overhang) for serum stability. I synthesized siRNA derivatives together with hydrolysis-stable MOE-RNA standard marker and analysed degradation of differently modified siRNA with HPLC, CE and Gel- electrophoresis. Furthermore, I have specified mechanism of degradation using LC-MS.

Diploma Thesis, University of Bern *Department of Chemistry and Biochemistry; Prof R. Haener* Construction and analysis of a cis-acting Ribonucleasemimic
Ribozyme mimics are generally linked to a nucleic-acid-backbone for specific recognition of the targeted gene transcripts. Efficient substitutes for the big catalytic domain are metal complexes
44 like Cutrpy for example. I have demonstrated self-cleavage of a RNA-Cutrpy-based ribozyme mimic at a specific phosphordiester bond. Main work involved the RNA backbone design (single bulge cleaving site), RNA modification (functional and radioactive / fluorescence labelling) and purification. Finally, triggering of cleavage and analysis of RNA degradation (PAGE).

PUBLICATIONS

Original articles published or accepted in peer reviewed journals (IF 2012)

Scott R, Siegrist F, Foser S, Certa U. Interferon-alpha induces reversible DNA demethylation of the IFITM3 core promoter in human melanoma cells. J Interferon Cytokine Res 2011 Aug 11; 31(8):601-8. IF: 3.3

Cortzar D, Kunz C, Selfridge J, Lettieri T, Saito Y, MacDougall E, Wirz A, Schuermann D, Jacobs A, Siegrist F, Steinacher R, Jiricny J, Bird A, Schr P. Embryonic Lethal Phenotype Reveals a Function of TDG in Maintaining Epigenetic Stability. Nature 2011 Feb 17; 470 (7334): 419-423. IF: 38.6

Siegrist F, Singer T, Certa U. MicroRNA Expression Profiling by Bead Array Technology in Human Tumor Cell Lines Treated with Interferon-Alpha-2a. Biol Proced Online 2009 Dec; 11 (1): 113-29. IF: 1.0

Hallen LC, Burki Y, Ebeling M, Broger C, Siegrist F, Oroszlan-Szovik K, Bohrmann B, Certa U, Foser S. Antiproliferative Activity of the Human IFN-alpha-Inducible Protein IFI44. J Interferon Cytokine Res 2007 Aug; 27 (8): 675-80. IF: 3.3

Reviews published or accepted in peer reviewed journals (IF 2012)

Tapparel C, Siegrist F, Petty TJ, Kaiser L. Picornavirus and enterovirus diversity with associated human diseases. Infect Genet Evol. 2013; 14: 282-293. IF: 2.8

Siegrist F, Ebeling M, Certa U. The small interferon induced transmembrane genes and proteins. J Interferon Cytokine Res 2011 Jan; 31 (1): 183-97. IF: 3.3

Original articles, reviews, editorials, letters, published or accepted in non-peer reviewed journals

Siegrist F, Certa U. Micro RNA Induction by Interferon Alpha and a Potential Role to Interfere with SOCS. In 7th Joint Conference Montral, Qubec, Canada, October 12-16, 2008 Editor: John Hiscott. Medimont International Proceedings 2008: 93-97.

Thesis

Siegrist F. Transcriptional responses of tumor cell lines to interferon-alpha. PhD Thesis, cell biology, University of Basel 2011.

Siegrist F. Auf der Spur eines artifiziellen, selbstspaltenden RNA-Molekls. Diplomathesis, Master of biochemistry, University of Bern 2003.

Abstracts presented at international and national meetings

Siegrist F, Otten-Hernandez P, Thomas Y, Farinelli L, Kaiser L and Tapparel C. Viral genome sequencing and small RNA detection by next generation sequencing. Cytokines 2012 Sep; 59(3): 565.

Siegrist F, Otten P, Thomas Y, Farinelli L, Kaiser L and Tapparel C. Viral genome sequencing and small RNA detection by next generation sequencing. 3rd Swiss Funtamental Virology Workshop 2011 Aug.

Urfer PM, Siegrist F, Noreen F, Weis S, Certa U, Truninger K, Schr P. Integrating transcriptome and epigenome analyses to identify DNA methylation changes associated with colorectal carcinogenesis. BioValley Science Day 2010 Sep.

Cortazar D, Kunz C, Selfridge J, Lettieri T, Wirz A, Schrmann D, Jacobs A, Siegrist F, Jiricny J, Bird A, Schr P. Embryonic lethality of TDG-deficient mice reveals a function of TDG in the maintenance of epigenetic stability. BioValley Science Day 2010 Sep.

Siegrist F, Certa U. Suppression of interferon alpha mediated gene expression by SOCS1 and SOCS3. FEBS-Special Meeting: Jak-Stat Signalling: from Basics to Disease 2010 Feb.

Siegrist F, Ebeling M, Certa U. Phylogenetic analysis of interferon inducible transmembrane gene family and functional aspects of IFITM3. Cytokine 2009 Oct-Nov; 48 (1-2): 87.

Siegrist F, Certa U. Micro RNA induction by interferon alpha and their potential role to interfere in the negative feedback pathway. Cytokine, 2008 Sep; 43 (3): 284-285.

Scott RW, Siegrist F, Burki Y, Foser S, Certa U. Methylation Status Influence On Interferon-alpha Sensitivity In Human Melanoma Cells. 3rd Swiss Meeting on Genome Stability DNA Dynamics and Epigenetics. 2007 Oct.

PUBLISHED DATASETS

GSE16421, GSE20693, GSE21158, GSE22801, GSE37872, GSE37873, CY079542.2-CY079549.