12-2002

Analysis of Protistan Grazing on Bioremediative Bacteria Using IN WVO Fluorescent Protein Expression and Flow Cytometry

Yutao Fu

ANALYSIS OF PROTISTAN GRAZING ON BIOREMEDIATIVE BACTERIA USING *IN VIVO* FLUORESCENT PROTEIN EXPRESSION AND FLOW CYTOMETRY

By

Yutao Fu

B.S. NanKai University, 1998

A THESIS

Submitted in Partial Fulfillment of the

Requirements for the Degree of

Master of Science

(in Biochemistry)

The Graduate School

The University of Maine

December, 2002

Advisory Committee:

Daniel L. Distel, Associate Professor of Biochemistry, Microbiology and Molecular Biology, Advisor

Katherine J. Boettcher, Assistant Professor of Microbiology

John T. Singer, Professor of Microbiology

ANALYSIS OF PROTISTAN GRAZING ON BIOREMEDIATIVE BACTERIA USING *IN VIVO* FLUORESCENT PROTEIN EXPRESSION AND FLOW CYTOMETRY

By Yutao Fu

Thesis Advisor: Dr. Daniel L. Distel

An Abstract of the Thesis Presented
in Partial Fulfillment of the Requirements for the
Degree of Master of Science
(in Biochemistry)
December, 2002

Protistan bacterivory can influence the size distribution, cell structure and composition of natural bacterial communities and is of significant concern for design of bioremediation efforts, yet adequate methods for observation and modeling are lacking. In this investigation, fluorescent protein expression and flow cytometry were used to study protistan grazing on genetically modified strains of several bacterial species that have been considered for use in bioremediation. Broad-host-range plasmids were constructed and used to introduce genes encoding GFP (green fluorescent protein) or RFP (red fluorescent protein) to prey species. A heterotrophic flagellate *Paraphysomonas imperforata* (Hflag) served as a model predator. Predator-prey interactions were observed and quantified using particle counts and individual optical signals recorded by FACScan flow cytometry. Result files were parsed by Windows Multiple-Document flow cytometry Interface (WinMDI v2.8) and analyzed by GR, a Perl (Practical Extraction and

Report Language) program described herein. Grazing preference of Hflag was influenced by prey type, size and predator culturing conditions. Hflag showed strong preference for bacterial cells over algal cells of a similar size, as well as for bacterial cells of different dimensions. However, size preferences were observed among cells within individual bacterial prey species. Significant preference was also observed for cells labeled by GFP and for unstained cells as compared to cells stained by a traditional DTAF-staining (5-(4,6-Dichloro-Triazin-2-yl)-Amino Fluorescein hydrochloride) method. No difference was observed between cells labeled by GFP and unstained cells. These results show that the methods described here provide a viable approach to observing protistan bacterivory that is superior in some respects to currently used methods.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

I. INTRODUCTION

Bioremediation, the use of naturally occurring or introduced organisms as a means to reduce the harmful effects of environmental contaminants, has been studied widely and applied with some success in field trials. For example, genetically modified bacteria have been used to transform or immobilize toxic chemicals such as polyaromatic hydrocarbons (PAH), chlorinated hydrocarbon, tri/di-chloroethene (TCE/DCE), and heavy metal ions (Ellis *et al.*, 2001; Arai *et al.*, 1999; Tan *et al.*, 1999; Tso and Taghon, 1997; Focht *et al.*, 1996; EPA 1995; Daubaras *et al.*, 1992). Success of such bioremediation efforts is dependent on growth of bacteria involved, and so is influenced by bacterivorous protists – indigenous predators of bacteria.

Protists are known to graze selectively on different types of bacteria and so may strongly influence bacterial community composition, physiology and metabolism. (Hahn and Höfle, 1998; Klaus *et al.*, 1999; Kinner *et al.*, 1998; Simek *et al.*, 1997). Factors proposed to influence prey selection include predator and prey specific factors such as predator digestive capacity, prey type, nutrient content (C:N ratio), motility and cell size; the latter being the most frequently studied (Hahn and Höfle, 1999; Klaus *et al.*, 1999; Kinner *et al.*, 1998; Simek and Chrzanowski, 1992, Verity, 1991). Protistan bacterivory may have both positive and negative impacts on bacterial populations. Selective grazing may reduce the population size of some bacteria while increasing that of others due to removal of competitors or by recycling nutrients in nutrient-limited environments (Hahn and Höfle, 1999; Tso and Taghon, 1997; Sherr *et al.*, 1986; Sherr *et al.*, 1982). Therefore, quantitative observation of protistan bacterivorous activity is crucial to achieving a

complete understanding of chemical and energetic transformations within ecosystems and so will be essential to the design of successful *in situ* bioremediation protocols.

To assess the potential effects of protistan bacterivory on bacterial populations, it will be necessary to develop methods to observe predator prey relationship realiably. A key question has been whether grazing rates, preferences and behaviors are altered by the methods that are used to observe and quantify them. Labeling methods for prey cells have employed fluorescent dyes, e.g. AO (Acridine Orange), CTC (cyanoditolyl tetrazolium chloride), immunofluorescence, and even radio-labeled bacteria (Christoffersen *et al.*, 1997; Epstein and Rossel, 1995; Nagaard and Hessen, 1990; Sherr *et al.*, 1987). The most commonly used method utilizes DTAF (5-(4,6-Dichloro-Triazin-2-yl)-Amino Fluorescein hydrochloride, 492 nm excitation maximum and 513 nm emission maximum) to label bacterial cells (Sherr *et al.*, 1987). Advantages of the DTAF staining method include its ability to stain a wide variety of cell types, and that preparations can be made from collected natural bacterial assemblages. However, the method results in prey cell death and chemical alteration of prey cell surfaces. Therefore, it is possible that DTAF staining may influence grazing by protistan predators. It was proposed that DTAF staining does not affect predator grazing rates based on the observation that a ubiquitous marine ciliate *Uronema marina* and a mixed culture of flagellates both grew at the same rate whether feeding on DTAF-stained FLB (Fluorescence-Labeled Bacteria) or on unstained natural bacteria (Sherr *et al.*, 1987). However, in these experiments grazing was observed on stained and unstained cells in separate experiments thus predator preference was not assessed. Moreover, observations were made only at a single prey concentration. Therefore, it could not be determined

whether predator growth was limited by prey availability or prey suitability. It remains to be determined whether chemical staining methods alter grazing by predators and thus may contribute to incorrect estimations of protistan bacterivory (see Chapter III).

The use of traditional microscopic methods has also placed some limitations on bacterivory studies. Traditional microscopy utilizes fixation steps that may skew estimation of bacterivory by altering behavior of the grazing organisms. For example, it has been shown that the rate of food vacuole egestion may be influenced by various fixation methods (Sieracki *et al.*, 1987). Also, manual cell enumeration using epifluorescence microscopy is a time-consuming and error-prone process that is not amenable to generation of large data sets suitable for statistical analyses.

In order to better measure protistan bacterivory, we have genetically modified prey bacteria to express fluorescent proteins *in vivo* and have used flow cytometry to automate detection and enumeration of predator and prey cells in microcosm studies. Using this approach, hereafter referred to as LIVE (Labeling by *In Vivo* Expression) Flow Cytometry, it is possible to observe grazing on live prey that have not been chemically-stained or fixed and it is possible to generate large numbers of observations in real time.

Two fluorescent proteins were used: *Aequoria victoria* Green Fluorescence Protein (GFP) and *Discosoma spp.* Red Fluorescence Protein (RFP). GFP is a 27 kDal protein with an encircled Ser-Tyr-Gly tripeptide fluorophore (wild type characteristics are a 395 nm excitation and a 509 nm emission maximum). Its compact molecular structure confers GFP chemical stability and high quantum yield (Yang *et al.*, 1996). GFP fluorescence is strongly correlated with host viability, which makes it superior to DTAF-staining for monitoring metabolically active and viable bacteria within populations. For

example, Lowder *et al.* (2000) used the chromosomally GFP-tagged *P. fluorescens* treated under different conditions to show that starved cells and VBNC (Viable But Non-Culturable) cells remain fluorescent in contrast to UV or heat-killed cells, most of which had lost GFP fluorescence, probably due to membrane breakdown. RFP (558 nm excitation and 583 nm emission) from the commercial plasmid pDsRed (Clontech, Palo Alto, CA) has been used in conjunction with GFP for multiple-color imaging (Bloemberg *et al.*, 2000). Due to the high similarity in the amino acid sequence to GFP, RFP has been assumed to have a similar β-can structure (Frandkov *et al.*, 2000; Geoffrey *et al.*, 2000; Yang *et al.*, 1996). By introducing genes encoding either red or green fluorescent proteins, it is possible to monitor two prey types simultaneously and determine grazing preference directly (see Chapter II).

A disadvantage of GFP labeling is that oxygen is required for development of the GFP fluorophore. Therefore GFP is not suitable for application in strictly anaerobic bacteria (Yang *et al.*, 1996). Oxygen suppression can lead to GFP bleeding through to longer emission wavelengths and crosstalk with the RFP signal (Elowitz 1997). The major drawbacks for RFP are its long maturation half-time (*ca.* 27 h) and the fact that the premature form fluoresces weakly in the green region of the spectrum (475 nm excitation and 499 nm emission) due to slow post-translational tetramerization (Geoffrey *et al.*, 2000; Gross *et al.*, 2000; Ahmed *et al.*, 2000).

A great advantage of live cell GFP/RFP labeling is that it is compatible with the application of automated real-time flow cytometry. Flow cytometers allow particles in a liquid suspension to traverse a laser beam, one at a time, to generate light signals for various PMT (Photo Multiplier Tubes) detectors. These signals include: forward scattered

light (FSC, related to particle size), side scatter light (SSC, related to surface granularity) and fluorescence intensity (FL, which may be observed in several wavelength ranges, FL1/FL2/FL3) (Shapiro, 1994). These signals can be used to distinguish predator and prey cells from other cells and inanimate particles.

Protist cells are generally larger than bacteria, therefore giving higher FSC signals. Labeled bacterial prey cells can be separated from other species or inanimate debris particles according to their fluorescent signals. After recording cell densities and detector signals of different particle groups along a time series, grazing rates can be calculated from the loss of prey densities divided by predator density and elapsed time (see Chapter III).

In order to steamline the tedious and repetitive data analysis of flow cytometry bacterivory experiments, a Perl program, entitled GR (Grazing Rates) was created to handle the large quantities of data generated by WinMDI. Perl programming has markedly accelerated analysis of flow cytometry grazing data with its excellent performance in text pattern matching and data structure management on different operating systems (Wall, 1996). The functions of GR include organization of sample parameters, management of data modification, time-related analysis, and statistical comparison between samples or sample sets. GR produces formatted results of grazing rates and calculates signal kinetics by comparing prey composition at different time points. Taken together, fluorescent protein labeling, flow cytometry and *in silico* information processing provide a promising tool kit for examination of protistan bacterivory.

II. VECTOR CONSTRUCTION AND BACTERIA TRANSFORMATION

Abstract

To induce *in vivo* expression of fluorescence in experimental prey cells for flow cytometric analyses of protistan bacterivory, genes encoding green fluorescent protein (GFP) and red fluorescent protein (RFP) were introduced on broad-host-range plasmids to a variety of bacteria. Plasmids were constructed based on broad-host-range expression vectors pBMMB67EH and pBBR122. Bacteria expressing these fluorescent proteins can easily be detected using either epifluorescence microscopy or flow cytometry.

Introduction

Several species of bacteria that have been proposed for use in bioremediation were chosen for use as model prey for assessing vulnerability of bioremediative bacteria to protistan grazing. These bacteria, which are capable of aerobic growth and are relatively easy to grow in pure culture, include *Pseudomonas putida*, *Rhodobacter sphaeroides, Klebsiella pneumoniae* and *Enterobacter aerogenes. P. putida* can oxidize catechol to dicarboxylic acid and eventually acetyl-CoA (Dayna, 1992; EPA Air Toxics Website: http://www.epa.gov/ttn/atw/hlthef/pyrocate.html). *R. sphaeroides* can reduce DMSO (Dimethyl Sulfoxide), a paper mill waste chemical, to hydrogen sulfide. (Temple, 2001; Ellis *et al.*, 2001). *K. pneumoniae* and *E. aerogenes* can both dechlorinate DDT (1,1,1-Trichloro-2,2-bis-(4'-chlorophenyl)ethane) to DDE (1,1-Dichloro-2,2-bis(4'-chlorophenyl)ethylene) by either aerobic or anaerobic degradation (Ellis *et al.*, 2001; Mendel and Walton, 1966). In addition, *Escherichia coli* was used for the ease with which it can be manipulated genetically. Each of these species can be cultivated in LB

(Luria-Bertani) medium at 37 °C or 30 °C, and none have significant autofluorescence overlapping with GFP or RFP spectra.

Exogenous genes can be introduced into bacteria either by integration into the chromosome or as autonomously replicating plasmids. In these experiments fluorescent protein genes were maintained in experimental prey cells on plasmids. Though less stable than chromosome labeling, plasmids achieve higher copy number, potentially increasing expression and improving detectability of the fluorescence signal. Plasmid-encoded genes are believed to account for the majority of biodegradation pathways (Tan, 1999; Dayna, 1992; Eberhard, 1990). Therefore, GFP/RFP plasmid labeling is a strategy that parallels many bioremediation-related genetic modifications. By taking advantage of established codon usage information, transformation protocols and protein expression conditions, GFP/RFP plasmid labeling can better mimic bioremediative bacteria populations under protistan grazing.

Commercial GFP and RFP plasmids (pQBIT7GFP and pDsRed, see materials and methods section) cannot be used to transform most species of bacteria directly, because they lack appropriate replication and expression machinery for many hosts (see Chapter II discussion). For example, pQBIT7GFP is a GFP encoding plasmid in which expression is controlled under a T7 promoter. Thus the presence of a T7 polymerase gene in the host genome is required. Unfortunately, the T7 polymerase gene is rare in naturally occuring wild type strains. Both pQBIT7GFP and pDsRed lack proper replication origin sequences and genes required for plasmid maintenance in many hosts. Furthermore, *E. coli* transformation procedures, including $CaCl_2$-heat shock and routine electroporation are not suitable for many bacterial species. Therefore, it was essential to construct broad-

host-range expression vectors and to establish transformation methods for each prey species.

Materials and Methods

Luria-Bertani (LB, powder from Fisher Scientific, Pittsburgh, PA) liquid medium was prepared and sterilized (15 min, 121 °C, 15 pounds per square inch) according to the manufacturers recommended protocol. SOC broth (2% Tryptone, 0.5% yeast extract, 10 mM NaCl, 2.5 mM KCl, 10 mM $MgCl_2$, 10 mM $MgSO_4$ and 20 mM glucose) and $CaCl_2$ (0.1 M) solutions were prepared according to Sambrook *et al.* (1989) and sterilized as above. *Rhodobacter sphaeroides* transformation buffers (0.2 M $CaCl_2$ - 0.1 M Tris, pH 7.2 and 40% PEG6000 (polyethylene glycol) – 0.1 M Tris, pH 7.2) were made by first preparing 10 ml Tris buffer (0.1 M, pH 7.2) then dissolving 4 g PEG6000 (Polyethylene glycol, Sigma, St. Louis, MO) or 0.22 g $CaCl_2$ in the Tris followed by filter-sterilization (0.2 _m). All restriction endonucleases, T4 DNA ligase and enzyme buffers were purchased from New England Biolabs (Beverly, MA) except that XbaI and its buffer were purchased from Life Technologies Gibco BRL (now Invitrogen, Carlsbad, CA). L1 and f/2 mineral medium was prepared according to Guillard *et al.* (1993). Organic protist media was prepared by mixing 5 parts filtered seawater and 2 parts dH_2O and adding yeast extract to a final concentration of 0.01% (w/v) before sterilization.

GFP plasmid pQBIT7GFP (catalog number AFP2242) was obtained from Quantum Biotechnologies Inc. (now QBiogene, Carlsbad, CA). RFP plasmid pDsRed was from Clontech (catalog number 6923-1). Broad-host-range vector pMMB67EH (ATCC# 37622) and pBBR122 (order number PBBR01) were from the American Type

Culture Collection (ATCC, Manassas, VA) and MoBiTec (Goettingen, Germany) respectively (Furste, 1986).

All bacterial species were obtained from ATCC unless indicated ot

herwise. Stock cultures of *Escherichia coli* BL21DE3 (Cat No. 69450, Novagen, Madison, WI), *Enterobacter aerogenes* NCDC 819-56, *Klebsiella pneumoniae* subsp. pneumoniae (Schroeter) (ATCC number 211), *Pseudomonas putida* KT2442, *P. putida* SM1396 (from Dr. Søren Molin's Lab at Technical University of Denmark) and *Rhodobacter sphaeroides* ATH2.4.1 cells were inoculated from frozen stocks (-80°C) into 2 ml liquid LB medium and were grown with shaking (250rpm) for 12 h at 37°C (*E. coli*, *E. aerogenes* and *K.pneumoniae)* or at 30°C (*P. putida* and *R .sphaeroides*) prior to use in transformation experiments detailed below. Transformed strains are hereafter referred to by three letter acronyms, the first two characters of which refer to the first letter of the genus and species names respectively, followed by the letters G or R indicating transformation with a GFP or RFP encoding plasmid (e.g. EcG = *E. coli* transformed with a GFP encoding plasmid, see Table 2). Transformed bacteria were cultivated in the presence of appropriate antibiotics (50 μg/ml ampicillin for EcG and EaG, or 10 μg/ml kanamycin for EcR, EaR, PpR, KaG and RsG. Culturing PpG required 250 μg/ml (5×) ampicillin plus IPTG (1μg/ml) in the medium and overnight growth on LB ampicillin agar plate at 37 °C solid medium to achieve homogeneous size and fluorescence distribution. EcR also required extended (48 h) shaking at 37 °C and a 24-h incubation at 4 °C to enhance fluorescence development.

Molecular recombination procedures were according to Sambrook *et al.* (1989). Restriction digestions were done in 50 μl volumes with 0.5 unit of enzyme (37 °C for 2

h). Ligation was performed overnight with an insert to vector ratio of 3:1 in a 20μl mixture containing 2U of T4 DNA ligase (16 °C for sticky-ends and room temperature for blunt-ends). Digested fragments were separated by electrophoresis (70V for 45 min) and recovered using QIAquick gel extraction kit according to product protocols. The GFP coding sequence on pQBIT7GFP (from position 30 to 1082, numbered relative to the start of its multiple cloning site) was excised using *HindIII/XbaI* digestion yielding a fragment of 1052 bps which was then inserted using the *HindIII/XbaI* sites of pMMB67EH broad-host-range expression plasmid (at positions 8804 and 8828) to construct pMMBGFP. The *lacIq* fragment was disrupted by *PvuII* digestion (cut at pMMB67EH positions 7357 and 7450 bp) to give pMMBGFPβ. The GFP coding sequence together with its Ptac promoter on pMMBGFP was amplified by a PCR reaction of 30 cycles with a left primer: ATCCGGGCTTATCGACTG; and a right primer: GCTGTAGGCATAGGCTTGG; Cycling conditions are: 94 °C for 45 sec (5 min for the first round), 52 °C for 1 min and 70 °C for 2 min (8 min at final cycle). The amplified fragment was then inserted into pBBR122 (Antoine and Locht 1992) to partially replace a portion of its chloramphenicol resistance gene between two *DraI* sites at 4361 and 4699 bp to construct pBBRGFP. The RFP coding sequence from pDsRed plasmid was blunt-end excised by *PvuII/StuI* digestion (cut at 55 and 1041 bp) and the resulting 986 bp fragment was inserted to pBBR122 as done with GFP. The orientation of insertion or presence/absence of sequences was verified by diagnostic restriction digestions followed by electrophoresis (see result section). See Figure 1 & 2.

E. coli (Ec) was transformed by calcium heat shock as described by Sambrook *et al.* (1989). Briefly, competent cells were prepared by washing early-log phase LB culture

(optical density, O.D., 0.4~0.6 at 600 nm) twice with 1/5 volume of ice-cold 0.1 M $CaCl_2$ and resuspending cells in 0.1 M $CaCl_2$ to 1/10 the original volume. After overnight incubation on ice, 0.1 ml competent cells were mixed with 0.5 μg plasmid DNA, heated-shocked at 42 °C for 90 sec then mixed with 0.9 ml SOC medium. After a 1-h incubation at 37°C with shaking (250 rpm) the transformed cells were spread on LB plates amended with the appropriate antibiotic.

E. aerogenes (Ea) was transformed by inoculating wild-type Ea into 5 ml LB liquid medium with 1 mM $MgCl_2$ and incubating at 30 °C with shaking (250 rpm) until mid-log phase (O.D. ~0.6). Cells were then pelleted by 8000×g centrifugation for 10 min at 4 °C. The cell pellet was then washed twice with 1 ml of ice-cold 0.01 M $CaCl_2$ and resuspended in 0.5 ml cold $CaCl_2$ and allowed to incubate on ice for 2 h. Cells were pelleted again (5000×g for 3 min at 4°C) and washed twice with 1 ml ice-cold 10% glycerol. After resuspension in 1 ml ice-cold 10% glycerol, 0.1ml competent cells were mixed with 1 _g plasmid. After a 10-min incubation on ice, the transformation mixture was frozen at –80 °C for 5 min then thawed on ice and transferred to an electroporation cuvette. The transformation mixture was electroporated (15 kV, 200 Ω resistance and 25 μF capacitance $\tau \approx 4.6$) using a Gene Pulser electroporation device (Biorad, Hercules, CA). Following electroporation, cells were gently transferred to a culture tube and allowed to stand on ice for 1 min followed by a 3-min incubation at 37 °C. SOC medium (0.9 ml) was then added and the cells were incubated at 30°C for 1 h. Cells were then spread on LB agar plates amended with the appropriate antibiotic.

Rs transformation: Cells were grown in 20 ml LB liquid culture until reaching 10^8 cell/ml (O.D. ~0.5) and centrifuged at 6000×g for 5 min at 4 °C. The pellet was washed with 10

ml ice-cold 0.5 M Tris twice and resuspended in 1.2 ml Tris-0.2 M $CaCl_2$. Plasmid DNA was added to 0.2 ml competent cells. An equal volume of ice-cold 40% PEG6000 in 100 mM pH 7.2 Tris was added to the transformation mixture, gently shaken to mix and incubated on ice for 10 min. Cells were heat-shocked at 35 °C for 2 min, added to 1 ml LB with 1 mM $MgCl_2$, incubated at 35 °C for 6.5 h and plated on agar.

The remaining species were transformed by routine electroporation (Dower, 1988). Briefly, LB-cultured cells were washed and resuspended in ice-cold 10% glycerol. Cells (0.1 ml) were mixed with DNA and electroporated at 12.5 Kv/cm, 200 Ω and 25 μF. Outgrowth and plating procedures were as described previously for heat shock methods.

FP-expressing bacteria were mounted to a Nikon Labophot-2 epifluorescence microscope and observed under both fluorescence and transmitted light with a 2.5× eyepiece and a 100× oil object lens (Plan Fluor 100/1.30 ph4 DLL 160/0.17), resulting in the total magnification of 250. Transformed and wild type bacteria cells were measured with a stage micrometer (Spencer Lens Co., Buffalo, NY) and enumerated with a hemacytometer (Fisher Scientific). The following filter sets were used for microscopy:

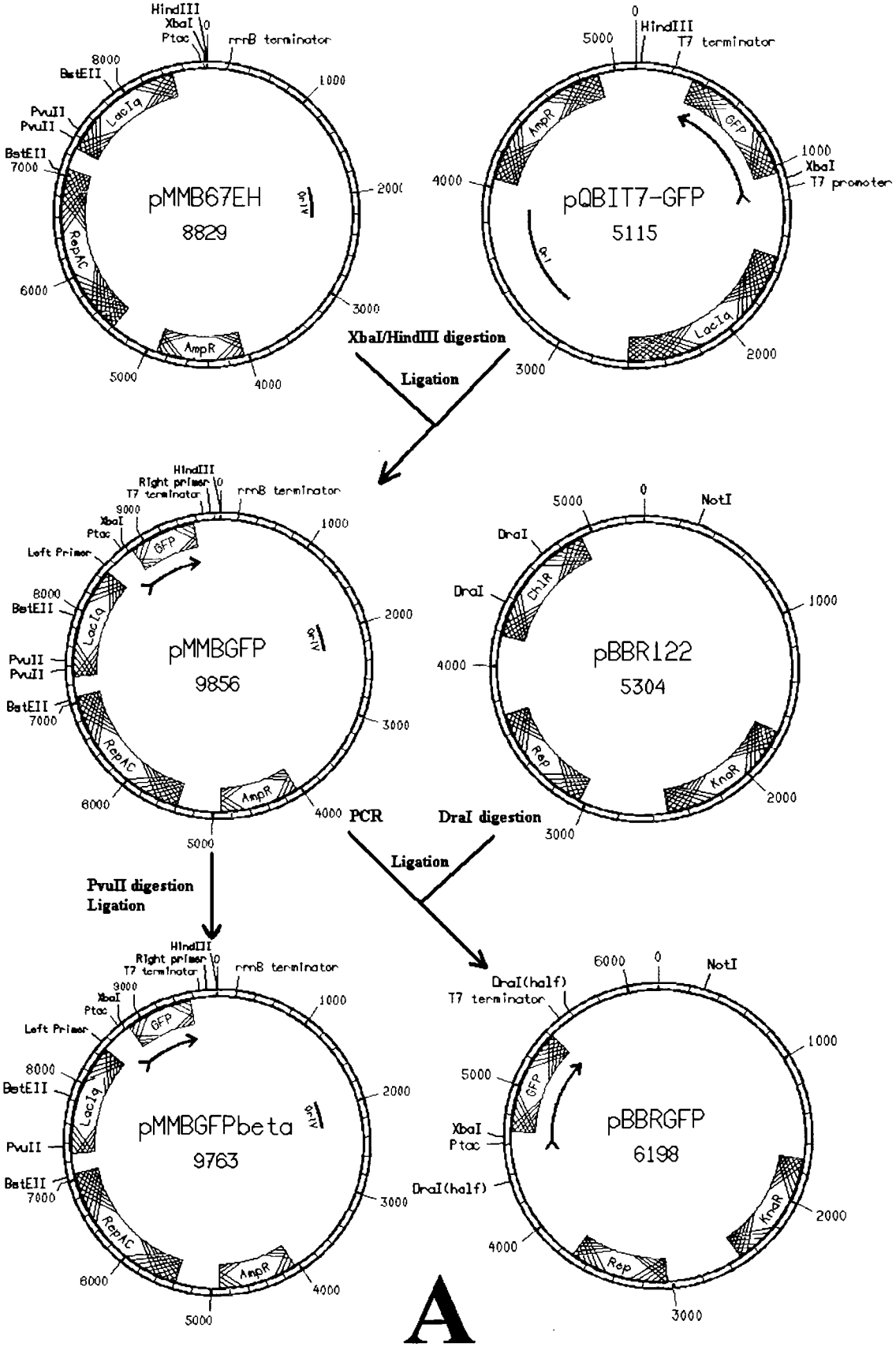
Filter set	Texas Red	TRITC/FITC	GFP
Exciter	540-580 nm	450-490	450/50
Dichroic Mirror	595	505	Q480LP
Barrier	600-660	520	510/50
Manufacturer	Nikon	Nikon	Chroma Corp.

Table 1. Filter sets installed for epifluorescence microscopy of GFP/RFP expressing bacteria.

Protistan bacterivory was started by adding bacteria prey to protist cultures (2 week-old L1 mineral medium culture) in 100 ml tissue culture flasks to a starting concentration of 10^7 prey/ml and ~10^4 predators/ml. The flasks were kept at room tempearture in the dark without shaking. To monitor fluorescence in food vacuoles of predators, protist cells in a grazing sample were first immobilized using 0.5% $NiSO_4$, then 5 μl of sample was applied onto a glass slide, covered with 25 mm × 25 mm ×0.17 mm coverslip, and the numbers of bacteria cells in the protist food vacuole were counted manually. Fluorescent images were taken with a SPOT-2 charge-coupled digital camera (Diagnostic Instrument Inc, Sterling Heights, MI). Density of predators was monitored by applying 20 μl grazing media onto a hemacytometer and manually counting the number of protists under the microscope for a time series of 0, 4, 8, 12, 20 and 28 h.

Results

Three GFP and two RFP broad-host-range expression plasmids were constructed (Figure 1). GFP expression on both pMMBGFP, pMMBGFPβ and pBBRGFP was controlled by the Ptac promoter from pMMB67EH and RFP expression on pBBRRFP(-) controlled by Plac from pDsRed. PCR-amplified Ptac-GFP DNA was inserted into pBBR122 in a single direction while *PvuII/StuI*-excised RFP fragment was ligated in both directions, giving pBBRRFP and pBBRRFP-.


pMMB67EH
8829
HindIII
XbaI
Ptac
rrnB terminator
LacIq
RepAC
AmpR
BstEII
PvuII
pQBIT7-GFP
5115
HindIII
T7 terminator
GFP
AmpR
LacIq
XbaI
T7 promoter
ori
XbaI/HindIII digestion
Ligation
pMMBGFP
9856
HindIII
Right primer
T7 terminator
rrnB terminator
XbaI
Ptac
GFP
Left Primer
LacIq
RepAC
AmpR
BstEII
PvuII
pBBR122
5304
NotI
DraI
CmR
Rep
KnaR
PCR
DraI digestion
Ligation
PvuII digestion
Ligation
pMMBGFPbeta
9763
pBBRGFP
6198
DraI(half)
T7 terminator
A

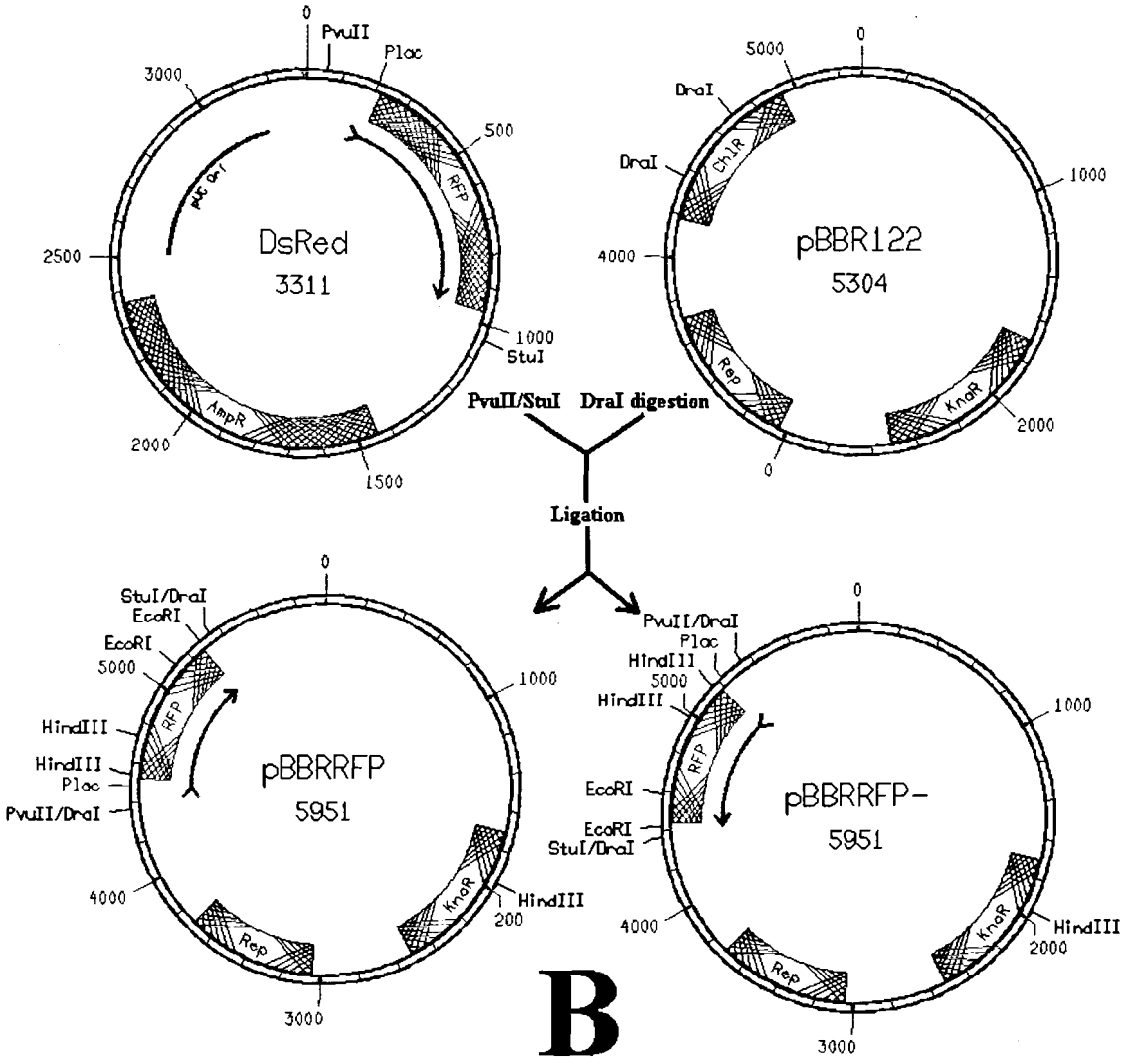


Figure 1. Construction of broad-host-range expression plasmids with GFP (A) or RFP (B). The GFP coding sequence without the T7 promoter was excised from pQBIT7GFP by *XbaI/HindIII* combined digestion and ligated downstream to the Ptac promoter of PMMB67EH. The resulting pMMBGFP was used as the template for PCR-amplification of Ptac-GFP. The amplified Ptac-GFP was blunt-ended by Klenow treatment and ligated with *DraI*-linearized pBBR122. (pBBR122α, which derived from pBBR122 with most of its chloramphenicol resistance gene removed by a single *DraI* digestion, was also obtained in this step). The RFP coding sequence with the Plac promoter of pDsRed was excised by *PvuII/StuI* digestion and ligated to *DraI*-digested pBBR122, giving pBBRRFP or pBBRRFP- depending on the orientation of the insertion.

Orientation of GFP coding sequence in pBBRGFP was examined by *NotI/XbaI* digestion (Figure 1 and 2), which generated two bands of 4.4 and 1.8 kb for all clones. If GFP was inserted in the same orientation as CAT is trancribed, the two bands should have been 5.4 and 0.8 kb, respectively. *HindIII/EcoRI* digestion was used to analyze pBBRRFP clones. Two different patterns were observed (3.2 kb + 2.6 kb and 3.0 kb + 2.8 kb), which were consistent with the two orientations of RFP insertions into pBBR122.

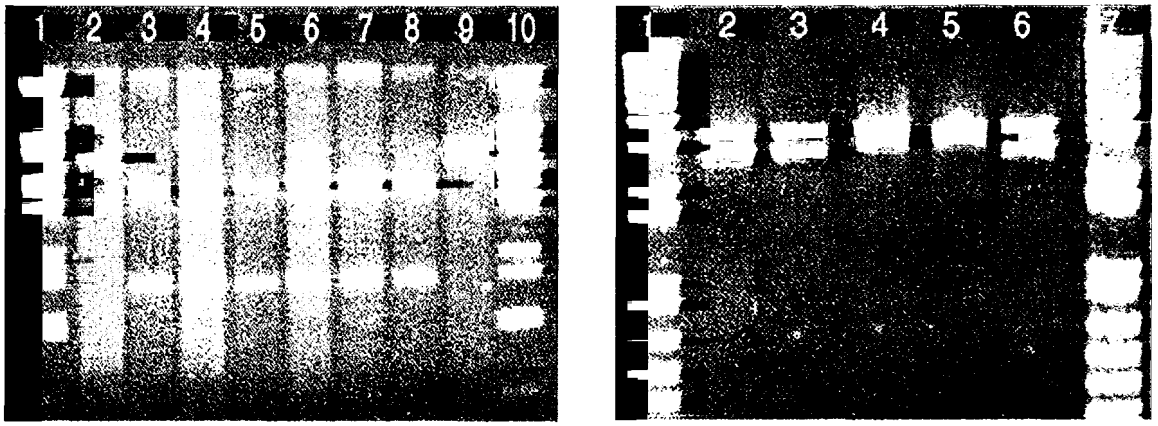


Figure 2. Electrophoresis band patterns of *NotI/XbaI*-digested pBBRGFP (A) and *HindIII/EcoRI*-digested pBBRRFP (B). Lanes in panel A: 1 and 10, *BstEII*-digested λ-DNA ladder; 3-8, *NotI/XbaI* -digested pBBRGFP clone 1-6, sequentially; 2 and 9, *XbaI* -digested pBBRGFP clone 1 and 6, respectively. Panel B: 1 and 7, 1kb DNA ladder; 2-6, *HindIII/EcoRI* -digested pBBRRFP(-) clone 1-5.

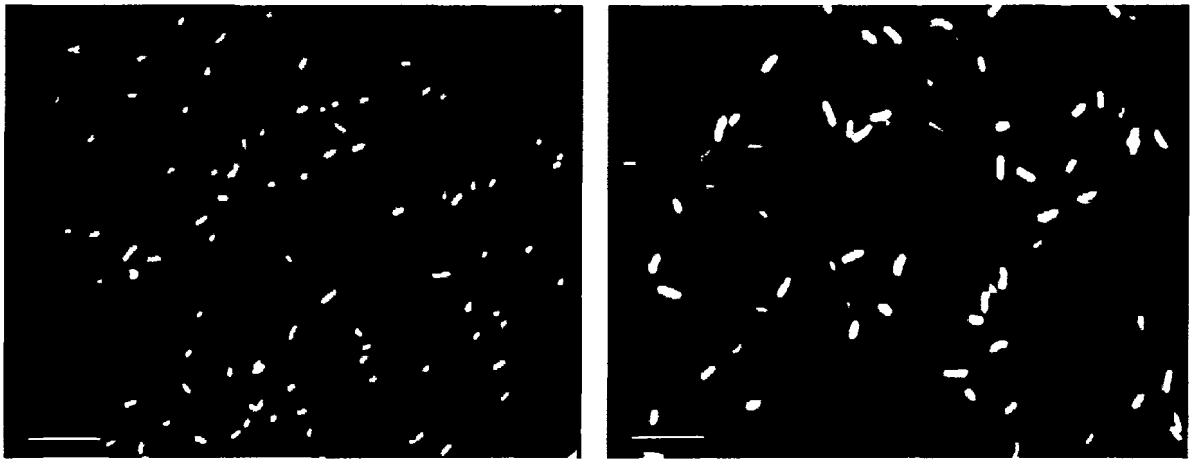
Five species of bacteria were labeled successfully with GFP plasmids, 3 of them also were transformed with RFP plasmids (Table 2). *P. putida* KT2442 was successfully transformed with pMMBGFP, but transformation of *P. putida* SM1396 with pQBIT7GFP failed.

Transformed *E. aerogenes, E. coli* and *P. putida* cells cultivated in LB liquid are rods, of 0.6 × 1.2 μm, 0.9 × 2.4 μm, 1.3 × 4.2 μm, respectively (average of 100 cells). The size differences among these bacterial species are clearly evident in Figure 3.

Species / Plasmids	*E. coli*	*E. aerogenes*	*R. sphaeroides*	*K .pneumoniae*	*P. putida*
pQBIT7GFP	+	--	--	--	--
pMMBGFP	+ (EcG)	+ (EaG)	--	--	+ (PpG)
pBBRGFP	+	+	+ (RsG)	+ (KpG)	--
pDsRed,	+	--	--	--	--
pBBRRFP	+ (EcR)	+ (EaR)	--	--	+ (PpR)

Table 2. Transformation of bioremediative bacteria species using various GFP and RFP plasmids. '+' indicates expression of GFP or RFP fluorescence, '--' indicates failure to express fluorescence. Acronyms for transformed strains used in this investigation appear in parentheses.

Fluorescent signals produced by transformed bacteria were sufficiently intense to allow direct observation by epifluorescence microscopy, especially for larger cells like PpG. No significant photobleaching or crosstalk between GFP and RFP was observed during normal observations, although the Texas-Red filter sets used are not optimized for RFP detection. However, when PpG and KpG cells were intensely illuminated by the blue light, photoactivation-induced bleed-through to the RFP channel was observed when switching wavelength from GFP to RFP excitation wavelengths (Elowitz *et al.*, 1997).



A: a mixture of EcR and EaG. **B: a mixture of EcR and PpG.**

Figure 3. GFP/RFP labeled EcR, EaG and PpG showed high fluorescent intensity and significant size differences. Scale bars represent 10 μm.

Predation by Hflag on GFP/RFP transformed bacteria could be observed directly by epifluorescence microscopy. When Hflag was exposed to bacteria expressing GFP or RFP, individually or in combination, accumulation of green and/or red fluorescent signal was observed in the food vacuoles (Figure 3). Fluorescence could be detected in food vacuoles within 1 min after mixing of prey and predators. For example in Figure 4 at least 6 bacterial cells can be distinguished in a single Hflag cell. Here Hflag cells were presented with EcR and EaG. Bacteria visible in the upper part of the cell appear to be undergoing cell lysis and digestion by the grazer. These cells are in vacuoles that appear overlapped in the image but that may or may not be coalescent. Four bacterial cells are visible in the lower portion of the cell, two of which exhibit red and two of which exhibit green fluoresence. The alternating positions of red and green labeled prey cells indicate grazing on both EcR and EaG.

In long-term predation experiments statistically indistinguishable growth of Hflag populations was observed when Hflag was exposed to wild type Ea cells as compared to GFP or RFP expressing transformants EaG and EaR (Figure 5). Hflag densities recorded at 0, 4, 8, 12, 20 and 28 h (Figure 5) showed no significant difference whether feeding on Ea, EaG or EaR. ANOVA (Aanalysis of Variation) for log-phase growth rates for Hflag of the same starting densities presented with Ea, EaG and EaR, and for the plateau densities both produced $p>0.5$.

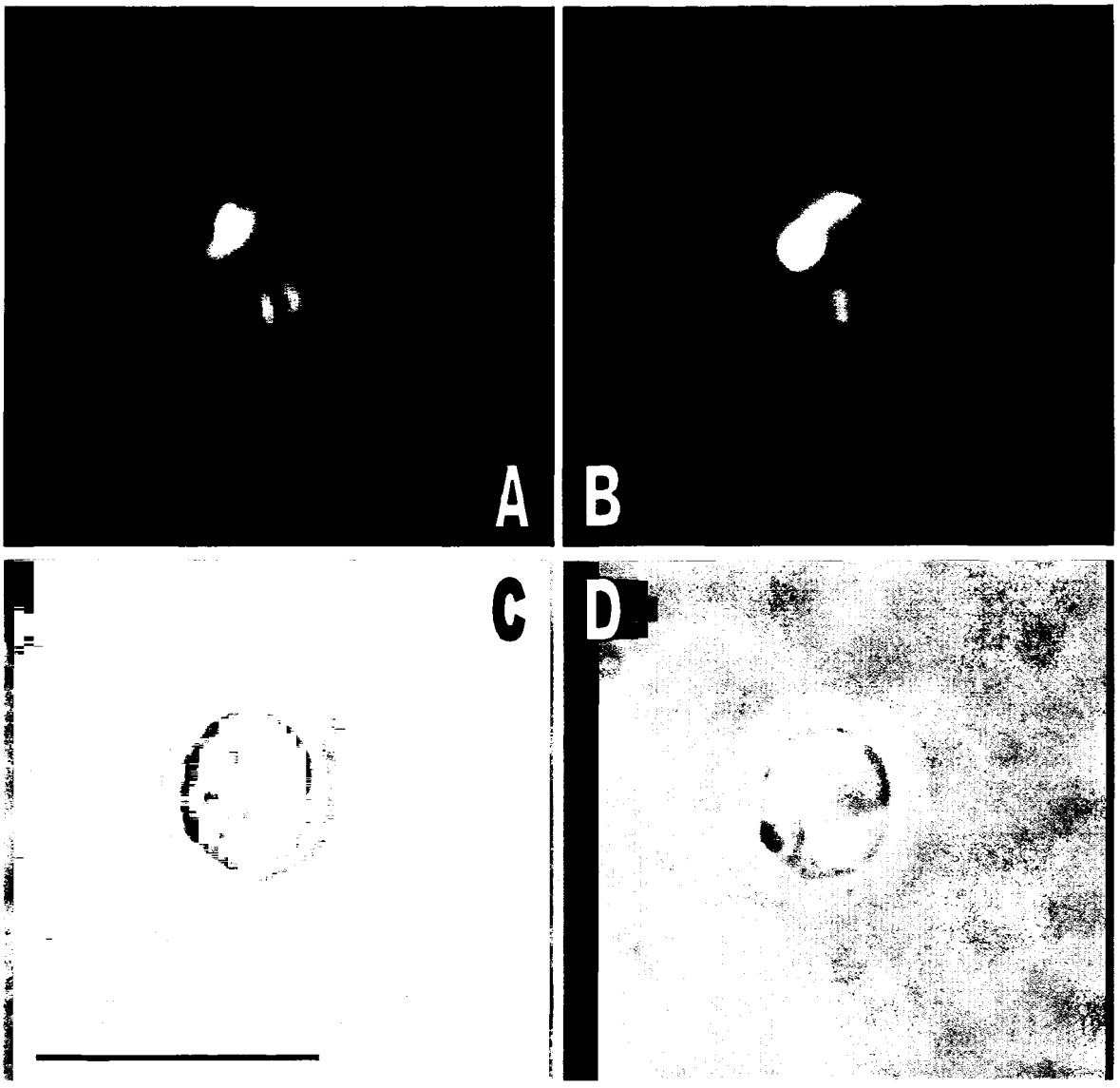


Figure 4. After ingesting EcR and EaG, fluorescent signals in red and green color can be detected in protist food vacuoles. Images were taken 3 minutes after mixing protist and bacteria culture. The flagellum of the Hflag is out of the plane of focus and cannot be seen in this image. A: Epifluorescent image with Texas-Red filter set; B: Epifluorescent image with GFP filter set; C: Transmistted light, phase contrast image; D: A+B+C. Scale bar: 10 μm.

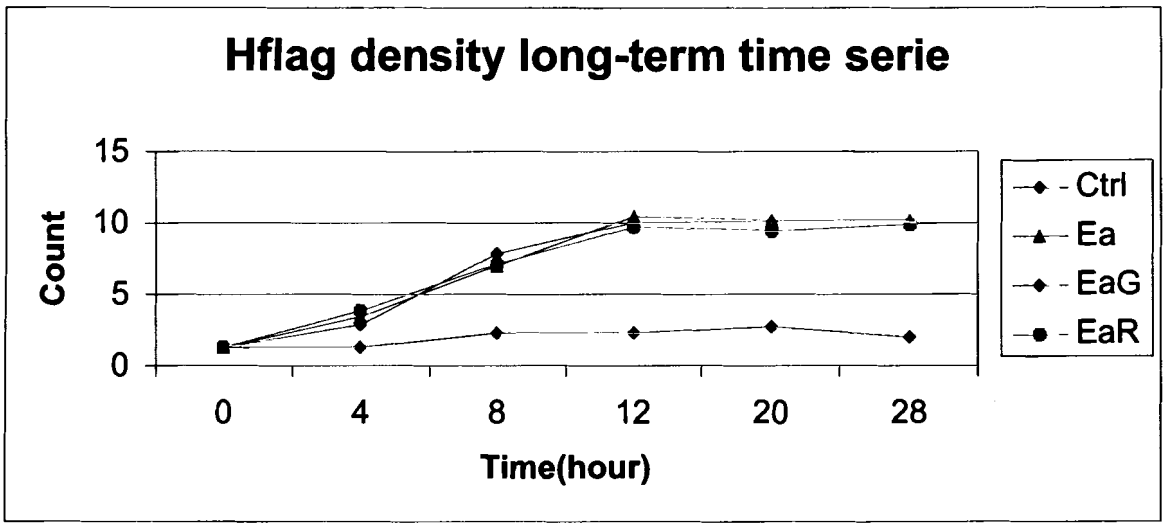


Figure 5. Protists (Hflag) feeding on Ea, EaG and EaR showed similar rate of growth. Error bars were not presented to avoid mixups

Discussion

A critical step toward the application of GFP and RFP fluorescence as biomarkers for monitoring bacterivory is the successful introduction and expression of genes encoding these fluorescent proteins to wild type prey species. This requires construction of broad-host-range fluorescent protein expression vectors. To construct a GFP expression vector, the GFP coding sequence from pQBIT7GFP was inserted downstream of the Ptac promoter of a broad-host-range vector pMMB67EH, by taking advantage of the presene of *HindIII* and *XbaI* sites in its directional multiple cloning sites. Ptac is a hybrid of the -35 box of Ptrp and the -10 box of the PlacUV5, and has shown strong expression in various bacteria with IPTG induction (Amman *et al.* 1983). The resulting pMMBGFP was introduced into *E.coli*, *E. aerogenes* and *P. putida* by electroporation and GFP fluorescence was detected by epifluorescence microscopy. Surprisingly, expression was observed without IPTG induction despite the presence of a complete *LacIq* gene on pMMBGFP. In principle a functional *lacIq* gene should have suppressed

GFP expression by producing an inhibitor protein, and Ptac has been known to be tight in expression control (Furste *et al*, 1986). Therefore, although IPTG induction did increase fluorescence intensity it is not a prerequisite for GFP expression. Whether this is due to Ptac promoter leakage in the stains of bacteria used in this study, or to alteration of *LacIq/LacO* interaction caused by recombination procedures, needs further investigation.

IPTG-independent GFP expression is desirable for the labeling and monitoring of bioremediative bacteria, since the requirement for unnecessary additional chemicals, like IPTG, that lack bioremediative functions, would be impractical in a large scale application in natural environments. To investigate whether it is possible to achieve a satisfactory level of IPTG-independent expression of GFP in this system, attempts were made to disrupt the function of the *lacIq* gene, thereby further increasing baseline expression of GFP without IPTG. *PvuII* digestion was used to remove a 93 bp fragment of *lacIq*. However, the resulting pMMBGFPβ did not show significantly higher constitutive GFP expression. An IPTG induction effect was still evident by comparing induced and uninduced EaG with epifluorescence microscopy, indicating that the absence of the 94bp did not inactivate *lacIq* completely. Additional efforts to excise a *BstEII* fragment of 879bp (from 7037 to 7916bp) from the *lacIq* gene (7286 to 8492 bp, see Figure 1) was also unsuccessful due to the failure of the construct to produce transformants. This may have been due to removal of 56bp from the terminus of the RepAC operon, which is required for plasmid replication (RepAC is located from 5416 to 7093 bp, the 3' end of which overlaps with the *BstEII* fragment). *BstEII* digestion was used to confirm the absence of the 93 bp fragment on pMMBGFPβ, which gives a band of 786 bp compared to an 879 bp fragment from *BstEII*-digested pMMBGFP.

While electroporation with pMMBGFP was successful in producing flourescent derivatives of *E.coli, E aerogenes*, and *P. putida,* no transformants of *K. pneumoniae* and *R. sphaeroides* were observed after electroporation with this plasmid. These species were not transformed with pMMB67EH by electroporation. As an alternative, a second GFP expression plasmid was constructed based on pBBR122 (a broad-host-range vector with a smaller size (2/3) and a lower copy number compared to pMMB67EH). This vector has been used successfully in more than 20 species including *R. sphaeroides*. (http://www.mobitec-germany.com/products/vector_sys/pbbr122.html). To construct this plasmid, here referred to as pBBRGFP, the GFP coding sequence and the associated Ptac promoter were PCR-amplified from pMMBGFP and inserted into the chloramphenicol acetyl transferase (CAT, or chlR in Figure 1) gene of pBBR122. Two *DraI* sites (at 4361 and 4699bp) in CAT were used to reduce vector size, to disable CAT for reverse selection and to provide blunt ends for ligation with PCR products.

It is interesting to note that this PCR fragment inserted to pBBR122 in only one orientation, opposite to that of CAT transcription. This may have been due to the expression of a toxic chimera of reverse-sense GFP and residual CAT. In contrast, the *StuI/PvuII* blunt-end restriction fragment containing the RFP coding sequence and its promoter was incorporated into pBBR122 in both orientations, producing pBBRRFP and pBBRRFP-, each with similar fluorescent strength. Though the copy number of pBBRRFP is lower than that of pDsRed, red fluorescence developed more quickly and cells displayed greater homogeneity with regard to fluorescence in pBBRRFP-transformed *E. coli.* Further investigation will be required to determine whether this is

due to differences in plasmid copy number, in expression efficiency, or in choice of antibiotic resistance.

Contrasting results were obtained for bacteria transformation with pQBIT7GFP and pMMBGFP, proving the necessity of constructing broad-host-range GFP expression vectors. *P. putida* SM1396 was derived from *P. putida* KT2442 by inserting a T7 polymease gene into the genome (Christensen *et al.*, 1996; Nieto *et al.*, 1990). Therefore if pQBIT7GFP could be introduced into *P. putida* SM1396, GFP should be transcribed by T7 polymerase. But transformation of *P. putida* SM1396 with pQBIT7GFP was unsuccessful. In contrast, *P. putida* KT2442 was transformed with pMMBGFP to produce green fluorescence. These results indicated that pMMB67EH-derived construct is necessary and sufficient for expressing GFP in *P. putida.*

Many wild type bacteria are not easily transformed by standard protocols used successfully for laboratory strains. In many cases additional and often more extreme manipulations are required to achieve reasonable transformation efficiencies. In these investigations *E. aerogenes* and *R. sphaeroides* proved to be the most difficult species to transform. The observed transformation efficiencies were on the order of 10^2 CFU/μg DNA for EaR and 10^3 for RsG, compared to 10^9 for *E. coli* electroporation. Improved transformation efficiency was obtained for Ea with pMMBGFP by adding a –80°C pre-chilling incubation before and a heat-shock step after electroporation (see methods). Profound temperature changes can cause altered distribution of membrane lipid, which may improve uptake of plasmid DNA. Post-electroporation heat shock may help to inhibit host restriction modification systems (Van der Rest *et al.*, 1999; Farinha and Kropinski, 1990). Improved transformation efficiency was observed for *R. sphaeroides*

by adding PEG6000 to coprecipitate DNA (Fornari and Kaplan, 1982) during $CaCl_2$-heat shock procedures. It is noteworthy that pMMBGFP is mobilizable, due to the presence of MobABC genes on pMMB67EH (Furste, 1986). Therefore, more bioremediative bacterial species may be transformed in the future by conjugation instead of electroporation.

A second critical concern in measuring protistan bacterivory is whether the biomarkers used to label bacterial prey cells alter the suitability of those organisms as food for the grazing protists. In a previous investigation Sherr *et al.* (1987) compared growth rates of protistan predators fed on a diet of DTAF labeled vs. unlabeled prey cells (Sherr et al, 1987). Similar growth rates were interpreted to indicate equivalent suitability of different prey categories as food for the grazing protists. In this investigation, long-term monitoring of changes in predator density demonstrated that, at least at the predator and prey densities observed, fluorescent protein expression labeled EaG and EaR supported growth rates for Hflag that were statistically indistinguishable from those produced by wild type Ea (Figure 5). While this type of experiment demonstrates that labeled and unlabeled prey are equally suitable food sources under the given conditions, it does not directly measure prey preference or factors that may be related to predator and prey density. For example, when prey density is growth limiting or when prey are available in large excess, differences in prey suitability or predator preference on predator growth rates may be masked. Measurement of clearance rates of prey cells at various levels of dilution provides a more direct and comprehensive approach to assessing prey preference and suitability for grazing protists.

In this investigation we have attempted to address these issues by using multiple-color *in vivo* labeling of prey species combined with real-time monitoring of prey clearance rates at various prey densities. This allows simultaneous monitoring of differently labeled prey cells of the same species, giving direct estimates of the effect of labeling on prey suitability and predator preference. The same methods can be used to assess differences in predator preferences for differently labeled prey of different species. Reciprocal labeling experiments, e.g. comparison of the labeled prey combinations such as EaG + EcR vs. EcG + EaR, can also help to reduce artifacts resulting from differences in detection efficiency of differently labeled prey types. To our knowledge, only two species have been reported previously to express RFP (*E. coli* and *P. fluorescens*).

The LIVE methods developed here may have broad applications including bacteria dispersion control, community composition analysis, symbiosis studies and many other tracking purposes aside from monitoring the effects of predation on bioremediative bacteria. For instance, Bloemberg *et al.* (2000) used chromosome labeling to express GFP, RFP, YFP (Yellow Fluorescent Protein) and BFP (Blue Fluorescent Protein) in *P. fluorescens* for multi-color imaging to monitor movement of bacteria of different origins along tomato seedling roots (Bloemberg *et al.*, 2000).

BFP and YFP are potentially useful for study protistan bacterivory by microscopy in addition to GFP and RFP. However, it is noteworthy that BFP requires UV excitation, which is not the common wavelength range for single-laser flow cytometers. The spectral separation between GFP and YFP is less than that between GFP and RFP; therefore more signal crosstalk should be expected for flow cytometry.

LIVE method may be used in the future to link directly monitoring of the fate and effectiveness of bioremediation bacteria. As more pathways of biodegradation become known, the genes and regulatory pathways linked with bioremediation will be determined. Fluorescent protein expression can be engineered to reflect regulation or expression of bioremediative pathways. For example GFP coding sequences could be inserted downstream from the promoter that is specifically activated during bioremediation, or fused with enzymes responsible for a given reaction. In this way it may be possible to monitor protistan grazing and biodegradative metabolism simultaneously, providing a better understanding of the fate of bioremediative bacteria in the environment.

III. FLOW CYTOMETRY BACTERIVORY ASSAY

Abstract

To evaluate these methods, ten grazing trials were conducted using Hflag as a model predator in microcosm experiments in which predator and prey cells were introduced in known quantities and monitored by flow cytometry over time courses ranging from 0-180 min. Various bacterial and eukaryotic prey types were added individually or in combination. These included both unlabeled cells of several prey species and prey cells that were fluorescence-labeled either by a traditional chemical method (DTAF) or by LIVE GFP or RFP plasmids. Clearance rates and size distribution were estimated for prey cells and cell density and accumulation of fluorescence in food vacuoles was monitored by optical properties observed using flow cytometry.

Hflag was found to prefer LIVE-labeled bacteria to *Micromonas pusilla* prey and the preference is not due to difference in prey cell size. Grazing preference within a single prey species is size-related, as the shift in size distribution suggested. Comparisons of protistan grazing on DTAF-stained bacteria, LIVE-labeled bacteria and unstained wild type bacteria suggested that Hflag select against DTAF-stained bacteria.

Introduction

LIVE-FCM (Labeling by *In Vivo* Expression and Flow Cytometry) has numerous applications for studies of microbial ecology and modeling of protistan bacterivory. However, the applications developed and described here have been limited by the narrow range of excitation wavelengths that can be generated and emission wavelengths that can be detected by the available flow cytometer. The argon laser employed by the FACScan

(488 nm blue laser) can only modestly excite wild-type GFP (Lybarger *et al.*, 1999). However, the GFP gene on pQBIT7GFP (sgAFP) is a red-shifted variant (474 nm excitation maximum and 509 nm emission), which is efficiently excited by the 488 nm blue laser. In addition, sgAFP is claimed to be the brightest available GFP variant and the only red-shifted variant with a Stoke's shift greater than 30 nm – the minimum separation of emission from excitation wavelength recommended for easy visualization (QBiogene product online literature, available at http://www.qbiogene.com/literature/maps/pdf/map-pQBI-T7-GFP.pdf). Finally, GFP emission (509 nm) matches the FACScan FL1 channel (510-525 nm). Therefore, the spectral properties of GFP are suitable for flow cytometry detection. For example, it has been demonstrated that GFP-labeled *Pseudomonas fluorescence* A506 cells can be detected using flow cytometry (Tombolini *et al.*, 1997).

The FACScan is less well suited for detection of RFP. Strong RFP fluorescence can be picked up by flow cytometer channel FL2 (585/21 nm) but its excitation maximum of 558 nm is distant from the 488 nm blue laser of FACScan. An alternative light source such as a green laser would better excite the RFP fluorophore resulting in more sensitive detection. This is especially necessary for species with lower RFP expression levels and species in which fluorophore development is slow.

Flow cytometry analysis and data processing can be easily automated. Compared with microscopy, the power of flow cytometry analysis comes from its capability to measure multiple parameters simultaneously and from the high operating speed of flow cytometers for particle analysis. The FACScan is capable of detecting 10^3 particles/sec and rapidly generates data files, which may be analyzed later with computer programs available for different operating systems. For example, WinMDI can be used on a PC for

batch processing of the large amount of data with features such as colored 2D regions for easier gate specification, log normal statistics, file name increment, and plain text output of collective statistical data for each sample (http://facs.scripps.edu/software.html).

Various parameters of protistan bacterivory can be calculated based on flow cytometry data. The dominant hypothesis currently used to describe bacterivory states that the frequency of prey encounters is the speed-limiting factor for protistan grazing on bacteria, i.e., the extent and magnitude of bacterivory by a single predator depends directly on prey density at a given moment. Prey density time curves, therefore, should be simulated to the kinetics of first-order reactions (Vazquez-Dominguez *et al.*, 1999; Kinner *et al.*, 1998; Sherr 1986). In this investigation the grazing rate (GR) was calculated according to the following formula:

$$\mathbf{GR=\ln(N_0/N_T)/MT}$$

where N is density of prey cells, M is density of predator cells and T is elapsed time. For prey species that display rapid production or high mortality it is necessary to subtract background prey growth and non-grazing death from the apparent clearance rates to obtain accurate estimates of clearance due to bacterivory. These values can be estimated from prey densities in control flasks lacking predator cells.

The heterotrophic flagellate *Paraphysomonas imperforata* (Hflag) was selected as the model predator in the grazing experiments because Hflag cultures can be easily maintained in the lab and the size of Hflag cells make them easily distinguishable from prey cells. Choice of prey was based on several factors. *K. pneumoniae* and *E. aerogenes* are commonly used in laboratory settings as food organisms for protists like Hflag. (O'Kelly, personal communication). *Micromonas pusilla* (abbreviated Mp hereafter) was

chosen to represent a non-bacteria prey since previous experiments showed that Hflag preferred *M. pusilla* to 2 other species (*Synechococcus spp. a*nd *Pycnococcus provasolii*) and fluorescent microspheres (Sieracki and Cucci, unpublished).

Materials and Methods

Filtered Sea Water (FSW) was obtained from the Center for Culture of Marine Phytoplankton in Boothbay Harbor, Maine. Phosphate Buffered Saline (PBS) was prepared as follows: 8.0 g NaCl, 0.2 g KCl, 1.44 g Na_2PO_4 and 0.24 g KH_2PO_4 were dissolved in 800 ml distilled H_2O; and adjusted to pH to 7.4 with HCl. The volume was adjusted to 1 L with distilled H_2O followed by sterilization. Carbonate-bicarbonate buffer (0.1 M, pH 9.5) was prepared by dissolving 2.93 g $NaHCO_3$ 1.59 g Na_2CO_3 in 1000 ml dH_2O, adjusting pH to 9.5 followed by sterilization.

A FACScan™ flow cytometer (Becton Dickinson Inc., Franklin Lakes, NJ) equipped with a 15mW argon laser at 488 nm and three-color fluorescence (510-525 nm; 560-590 nm; >650 nm emissions) is driven by CellQuest software installed on a Power Macintosh 8500.

Micromonas pusilla IB4 was maintained in f/2 mineral media at 21°C with 12 h light-dark cycling. Hflag was maintained either in L1 mineral media or organic media at 21°C in the dark. Cultures were transferred to new media and fed with *E. aerogenes* every 30 days. Ten-day old protist cultures were used in bacterivory experiments.

DTAF staining was preformed according to Vazquez-Dominguez *et al.* (1999). Bacterial cells in mid-log phase culture (O.D. ~0.6) were harvested by centrifugation at 6000×g for 3 min, and washed twice by a pipetting basic phosphate buffered saline (0.05 M Na_2HPO_4-0.85% NaCl adjusted to pH 9.0) of equal volume. Cells were then resuspend

in PBS at 10^9 cells/ml, added with DTAF to the final concentration of 200μg/ml, and incubated in 60°C water bath for 2 h. Stained cells were washed by centrifugation at 3600×g for 3 min and resuspension with 0.1 M pH 9.5 carbonate-bicarbonate buffer for 6 times. Cell clumping was minimized by vigorous vortexing.

Bacterial cells in mid-log phase culture (O.D. ~0.6) were harvested and washed twice (by centrifugation at 6000×g for 3 min and pipetting filtered seawater (FSW) of equal volume), and diluted to 1:1000 with FSW. One milliliter diluted bacterial cells were mixed with 50 μl 37% paraformaldehyde, kept in a 1.5ml eppendorf centrifuge tube at 4 °C in the dark for 30 min, then mixed with 10 μl Pico Green stock solution (Molecular Probes, Inc, Eugene, OR) and stained in the dark at room temperature for 15 min.

Before grazing analysis, bacteria density was determined and detector settings were optimized for each bacterial strain in pure culture using the FACScan. Briefly, cultured bacteria were washed twice with PBS and twice with FSW by centrifugation at 6000×g for 3 min, resuspended into FSW of equal volume, transferred to 10ml glass tubes which were then mounted onto FACScan. For all the following experiments, the threshold was set to SSC at 150. The voltages (amp gain) were set to E00 for FL1 (1.0), 300 for SSC (1.0), 550 for FL1, 520 for FL2, and 550 for FL3. Sample flow rate was 10 μl/sec.

Bacterivory assays were performed in 25 ml filtered seawater in 100 ml tissue culture flasks with the following steps:

1. One milliliter 10-days old Hflag culture was pipetted to a 10ml FACScan sample tube and the density of Hflag was counted on FSC-SSC scattergrams. Dilutions for treatment flasks were calculated to reach 10^5 Hflag/ml in experimental flasks.

2. Prey cells were washed with FSW for 5 times (by centrifugation at 5000×g for 3 min followed by resuspension), diluted with FSW (1:100 for Pico green stained bacteria, 1:100000 for LIVE-labeled bacteria, and 1:10000 for DTAF-stained bacteria) and counted on FL1-FSC scattergrams. Dilution rates were calculated to reach ~10^6 prey cells/ml in experimental flasks.

3. Flask setup (in order):

Treatment	FSW	Hflag	Prey
Experimental flask	+	+	+
No prey control	+	+	--
No predator control	+	--	+

Table 3. Composition of microcosms for flow cytometetric bacterivory experiments.

4. Each experiment was done in at least triplicate. Bacteria prey cells were added to the experimental flasks at time 0. The mixture was incubated in the dark at room temperature for time intervals specified in individual experiments. At each sampling time point the flask was mixed thoroughly and aliquots of 0.5 ml were transferred from each flask to a 10 ml FACScan sample tube. Tubes were weighed before and after mounting onto FACScan to determine the flow-through volume. Each tube was sampled was for 3 min to record light scatter and fluorescence signals for particles in the sample.

Flow cytometry data were saved as FCS2.0 list mode files by CellQuest, transferred to a PC and read by WinMDI Flow cytometry Interface (Joseph Trotter of the Scripps Research Institute, unpublished). Numbers of all cell population at each sampling time was exported to Excel files and plotted with the charter tools therein. Prey and

predator regions were specified on scattergrams and used to gate histograms to produce plain text readouts. WinMDI output was analyzed by GR to calculate grazing rates and signal kinetics (see Chapter IV). ANOVA was used to calculate probability (P) value for statistically significant difference in grazing rates in dilution experiments, and Student's t-test (paired) was used for non-dilution experiments if there were more than one prey species.

Results

The GFP signals for all green transformants tested were easily detected by the FACScan flow cytometer using the FL1 channel. However, of the three RFP transformants confirmed by epifluorescence microscopy (EaR, EcR, and PpR), only EcR produced sufficiently strong signal to be distinguished from background noise in FL2 channel.

Among experimental prey types tested, those that produced detectable fluorescent signal could be distinguished from other, differently labeled prey types and from predator cells by analysis of scattergrams comparing fluorescence, forward scattered and side scattered light signals. For example, Figure 6 demonstrates resolution of a mixed population of Hflag, EcR and EaG. The Hflag population could be distinguished from the bacteria prey populations by examination of the FSC/SSC signals. This reflects the significant difference in sizes and surface properties observed between predator and prey cells. Prey cell types could also be distinguished from one another. EcR has higher light scatter signals than EaG, in accordance with its larger size. However, these two populations overlapped in size distribution and so could not be distinguished by FSC/SSC

separation alone. However, they were easily distinguished by their different fluorescence signals.

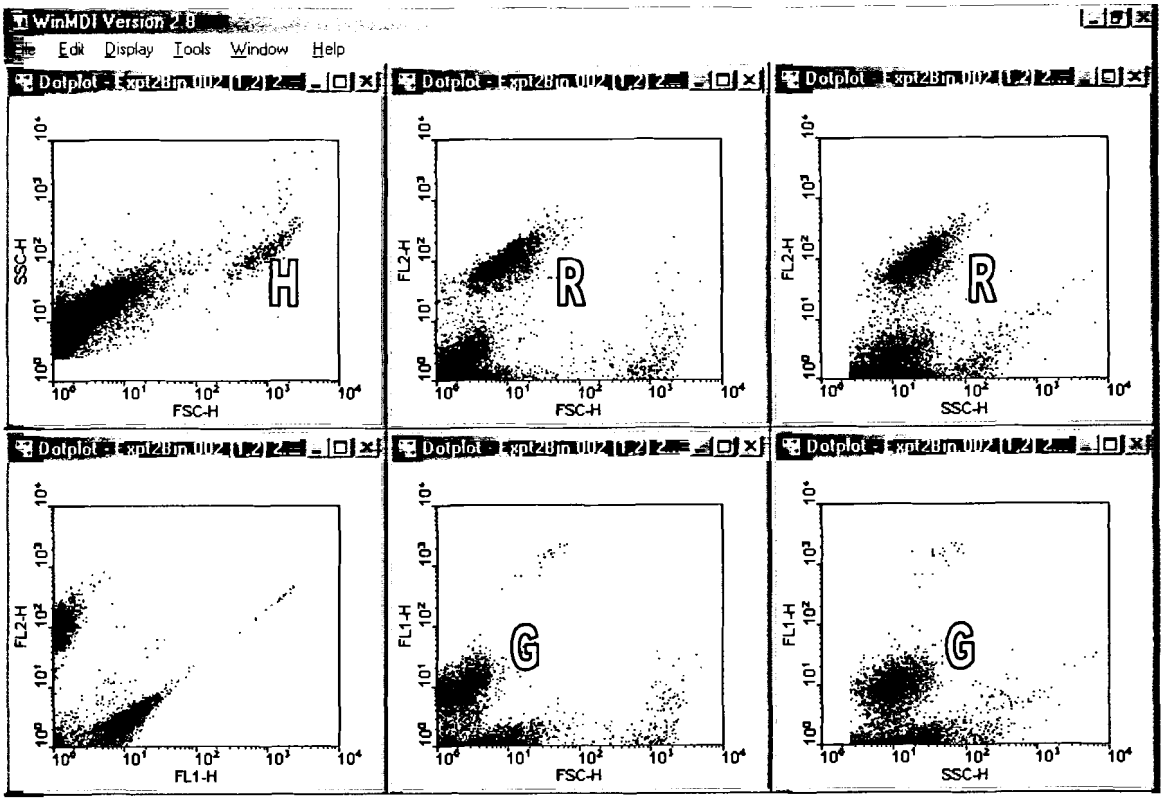


Figure 6. Flow cytometry data scatter gram generated by FACScan and displayed by WinMDI flow cytometry interface. Three populations of microbes in the same sample were labeled H for Hflag, R for EcR and G for EaG

The prey cell density counts of EaG and EcR were shown in Figure 7. Densities of the prey species were plotted against grazing time. The control densities in flasks without Hflag remained stable (less than 5% variation from initial values) during the experiment time course. In contrast, the density of EcR and EaG showed exponential decay over time with grazing by Hflag. For the experiment represented by Figure 7, the grazing rates for EaG and EaR were 0.203 and 0.271 nl·Hflag^{-1}·h^{-1}, respectively (see Table 4).

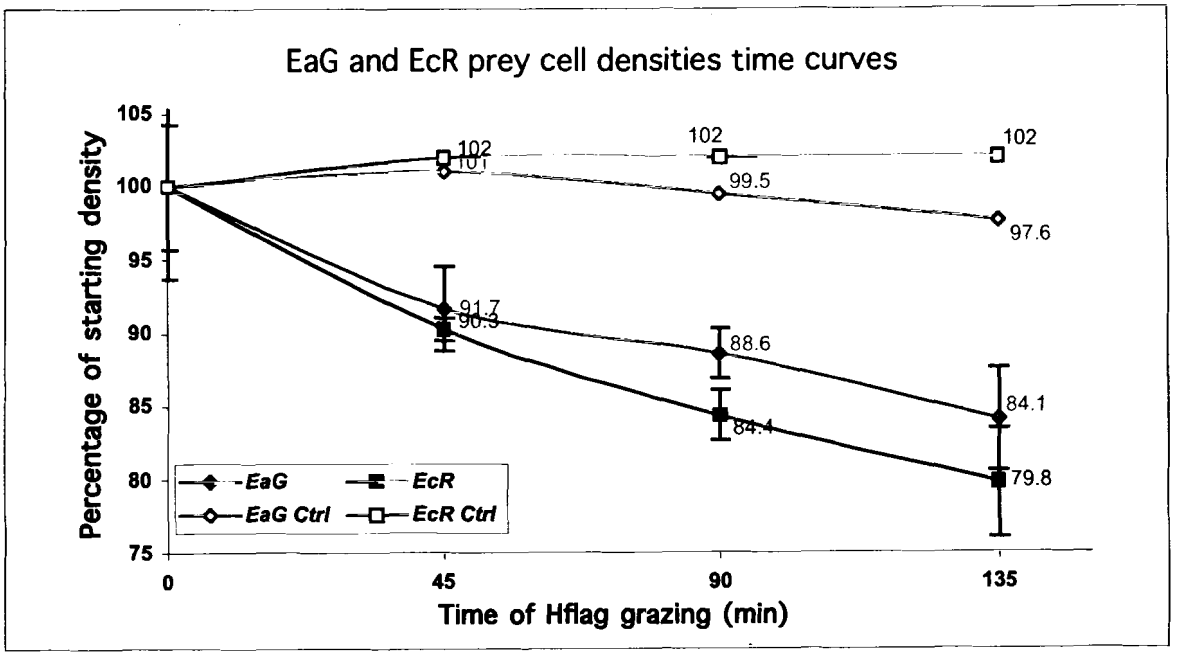


Figure 7. Relative densities of EaG and EcR under Hflag grazing for a time series.

Various experiments were performed to examine grazing by Hflag on different prey species (discussed in detail in the next section). In all experiments significant grazing effects were observed. Grazing rates were calculated according to the aforementioned formula for each flask and time interval. Grazing rates remained consistent within individual experiments, i.e., within a given sample, grazing rates remained statistically unchanged for all time intervals examined up to 180 min. The average grazing rates and standard deviations are listed in the following table:

Trial	Hflag batch	Prey species											P
		Mp	EaG	10%G	1%G	PpG	EcG	EcR	EcRD	EaD	10%D	1%D	
1	1	–	0.281 (0.055)	–	–	–	–	–	–	–	–	–	–
2	1	0.162 (0.110)	–	–	–	–	–	–	–	–	–	–	–
3	1	0.042 (0.025)	0.273 (0.039)	–	–	–	–	–	–	–	–	–	0.002
4	2	–	–	–	–	0.168 (0.067)	–	–	–	–	–	–	–
5	2	0.071 (0.038)	–	–	–	0.226 (0.094)	–	–	–	–	–	–	0.027
6	2	–	0.203 (0.119)	–	–	–	–	0.271 (0.044)	–	–	–	–	0.150
7	3	–	–	–	–	–	0.664 (0.166)	0.660 (0.161)	–	–	–	–	0.487
8	4	–	–	–	–	–	–	0.554 (0.113)	0.389 (0.098)	–	–	–	0.007
9*	5	–	–	–	–	–	–	–	–	0.841 (0.290)	1.33 (0.40)	1.55 (0.54)	0.035
10*	6	–	0.594 (0.15)	0.592 (0.261)	0.609 (0.285)	–	–	–	–	–	–	–	0.990

Table 4. Grazing rates calculated from various bacterivory experiments. The numbers in brackets are stand deviations. Experiments labeled with an asterisk were dilution experiments. For dilution experiments, each column showed the dilutions of tracer prey species used for different flasks. For all other (non-dilution) experiments each row showed the prey species present in the same grazing flasks. P stands for probabilities of prey indiscrimination. Abbreviations for prey species are the same as mentioned in Chapter I, in addition, 10%G stands for EaG diluted to 10:100 with wild type *E. aerogenes*; 1%G, EaG diluted to 1:100 with wild type *E. aerogenes*; EcRD, DTAF-stained EcR; EaD, DTAF-stained wild type *E. aerogenes*; 10%D, EaD diluted to 10:100 with by wild type *E. aerogenes*; 1%D, EaD diluted to 1:100 with wild type *E. aerogenes*.

The change in fluorescent intensity from an average protist food vacuole during experiment time courses was examined (Figure 8). EaD, PpG and EcR fluorescence showed significant accumulation during the time series. DTAF-fluorescence continued to increase during grazing for 135 min, while the signals from samples with GFP and RFP appeared to reach an asymptote by about 90 min. EaG, EcG and Mp fluorescence from Hflag food vacuoles did not show such an increasing trend, probably due to low GFP expression levels and small cell volumes.

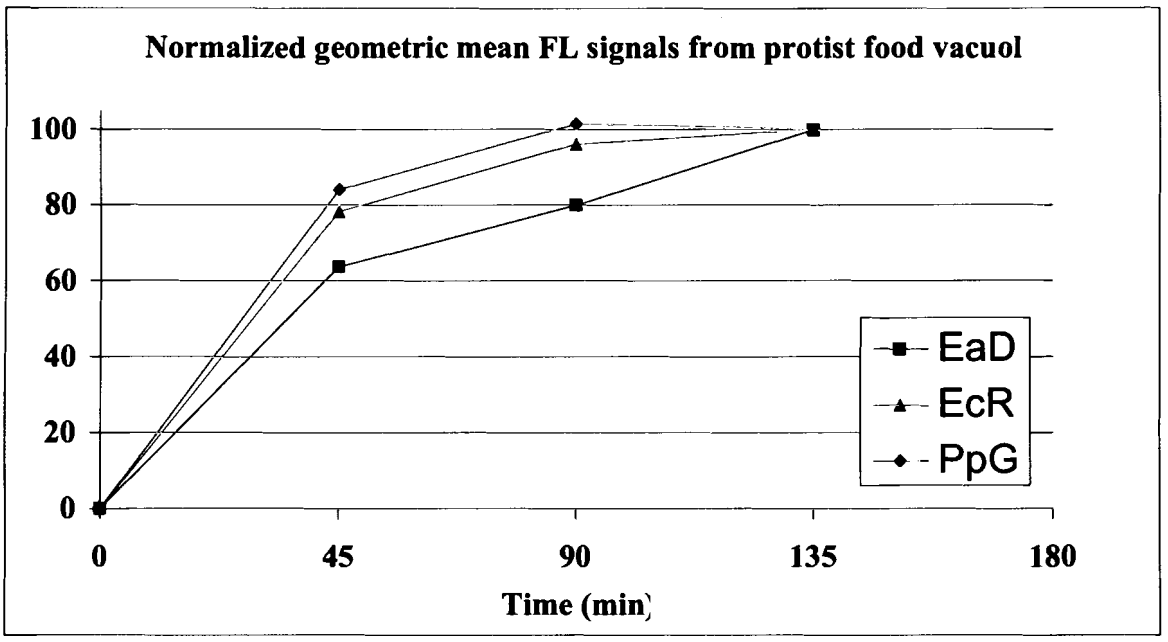


Figure 8. Time-curve of protist food vacuole fluorescence. Normalized FL signals (FL1 for EaD and PpG, FL2 for EcR, each using 135 min readings as 100% for each fluorophore) were plotted against time.

Effects of protistan grazing on size distribution prey populations in real-time were significant. An example is illustrated in Figure 9 in which the effect of grazing by Hflag on prey size disribution for two prey species was examined. Changes in average prey cell size could be inferred for the remaining prey population after grazing as reflected by changes in FSC over given time intervals. For PpG a significant shift toward larger cell sizes was observed in the surviving prey population during each grazing interval. The mean size for consumed prey cells as inferred by the shift in FSC signals were significantly smaller than the mean size of remaining prey cells for all intervals. And consumed prey for the first two intervals (0-45min and 45-90 min) was significantly smaller than that for the third interval (90-135 min). No significant changes in inferred size distribution were observed after grazing by Hflag on *E. aerogenes* or other prey species.

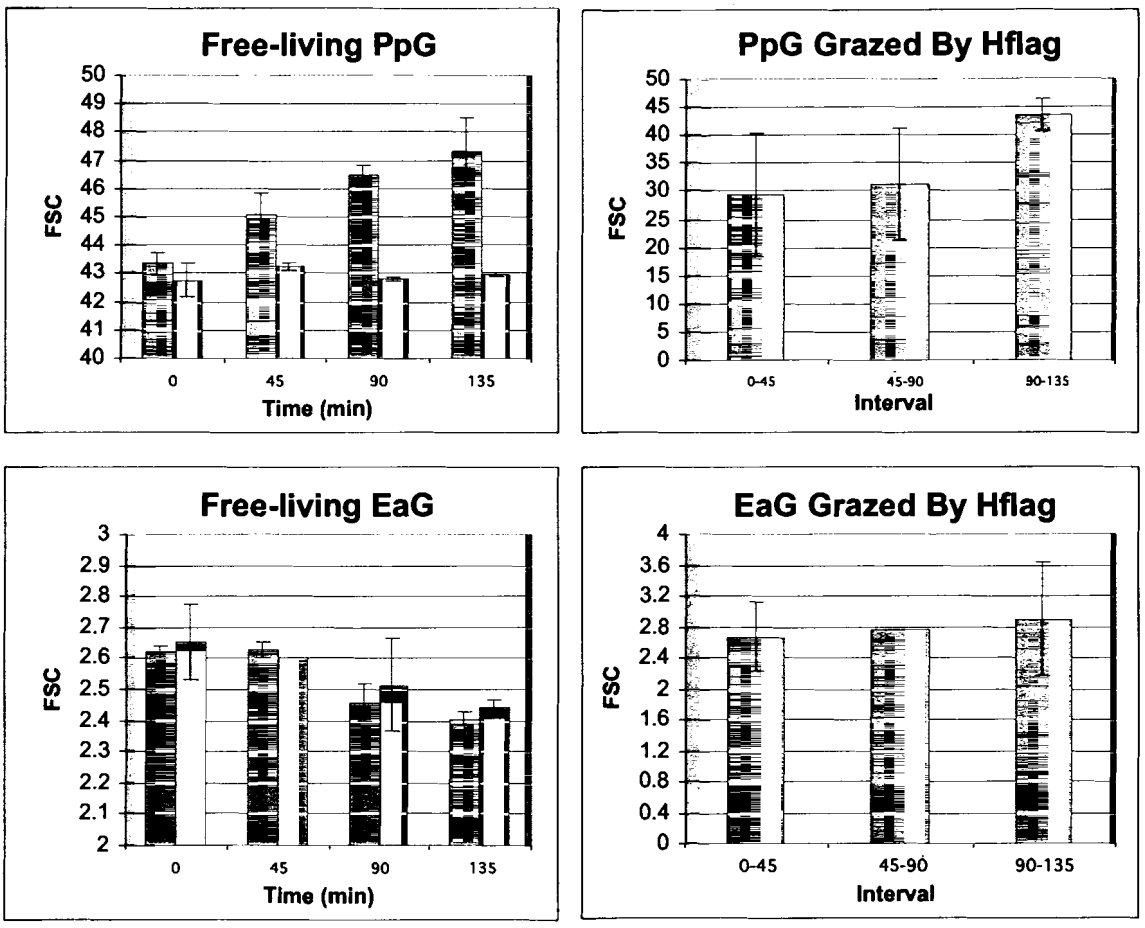


Figure 9. Time kinetics histograms of PpG and EaG forward scatter signals. Left panels showed mean FSC signals of prey population for a time series with (green bars) or without (blue) Hflag grazing. Red bars in the right panels showed the inferred mean FSC signals of prey cleared by Hflag between time points.

Discussion

In order to obtain accurate estimates of protistan bacterivory on bioremediative bacteria, it must be demonstrated that 1) predators successfully consume labeled prey, 2) labeled prey can be distinguished from unlabeled cells, differently labeled cells, inanimate particles and non-prey cells, 3) changes in predator and prey populations can be determined quantitatively and in real time, and 4) labeled prey are viable and no comparative preference is observed for labeled vs. unlabeled (wild-type) prey cells.

Furthermore, to realistically model predator-prey relationships, such as prey preference, it must be possible to differentially label prey types and to obtain simultaneous determinations predator and prey densities and other parameters in controlled microcosm experiments containing predators and mixed prey types. LIVE-FCM was used to perform grazing trials (Table 4) in attempt to more closely satisfy these ideal requirements for a bacterivory assay system.

The conclusions of the trials were discussed briefly here in the order of their appearance in Table 4, and explained in rest of discussion in the order of their importance regarding the Hflag grazing study. Trial 1 and 2 demonstrated that it is possible to measure grazing by Hflag on two type of prey cells, bacteria genetically modified to express GFP *in vivo* (EaG) and a eucaryotic prey species that expresses a detectable endogenous red fluoresence due to the presence of naturally occurring pigments (Mp), quantitatively and in real time using LIVE-FCM. Trial 3 showed that grazing on these two prey types can be introduced in combination to be measured simultaneously and that the predator Hflag demonstrates significant preference for the bacterial prey over the eucayotic prey cells of similar size. Trial 4 and 5 demonstrated preference for a bacterial prey species over the eukaryotic prey regardless of the fact that the average size distribution of the bacterial prey is larger than the preferential prey size for this predator, which caused a significant shift in overall size distribution of prey cells. Trial 6 indicates that GFP and RFP LIVE plasmids can be used to distinguish bacterial species and to determine grazing rates for each in mixed prey microcosms. Trial 7 showed that no difference is observed in grazing rates of Hflag for a single bacterial species (*E.coli*) whether LIVE-labeled with GFP or RFP. Trial 8 and 9 suggested that a commonly used

chemical labeling method (DTAF) results in decreased grazing rates when compared to either LIVE-labeled or unlabeled wild-type prey cells of the same species. Last but not least, Trial 10 indicated that no difference is observed between protistan grazing rates for GFP LIVE-labeled prey cells and unlabeled wild-type prey cells of the the same species.

Information obtained from these trials can be classified into the following aspects that elucidate important features of Hflag bacterivory:

1) LIVE *vs.* DTAF labeling

Trial 10 was designed to examine whether GFP labeled EaG could be ingested as fast as wild type *E. aerogenes* prey. In the set of dilution experiments, the portion of fluorescently-labeled bacteria prey varied from 1% to 100% of total. Because grazing rates are equivalent to the volume of medium a single protist cell clears within a unit of time, the dilution experiments should give similar results for different ratios of labeled prey, if there is no preference for or against it. EaG dilution experiments confirmed that Hflag does not differentiate between Ea and EaG, since the grazing rates for 100%, 10%, 1% tracer experiments were the similar (0.594+/-0.15, 0.592+/-0.261 and 0.609+/-0.285, respectively, ANOVA P=0.99).

However, EaD dilution experiments (Trial 9) showed that Hflag prefers Ea to EaD (DTAF-stained *E. aerogenes*) with apparently higher grazing rates for lower tracer proportion. The following is the relevant GR output (each dilution treatment is presented in the following format: number of replicates * mean grazing rates +/- c.v.):

1% : 6 * 1.55 +/- 0.542 vs. 100% : 6 * 0.841 +/- 0.291 Student's t-test P=0.019
10% : 5 * 1.33 +/- 0.405 vs. 100% : 6 * 0.841 +/- 0.291 Student's t-test P=0.045
ANOVA : FP=0.03549

This result suggested that Hflag selected against DTAF-stained *E. aerogenes*, which may cause underestimation of protistan bacterivory.

More evidence for Hflag selection against DTAF-stained bacteria comes from the direct comparison of grazing rates in Trial 8 (Table 4). The dilution experiments were designed because there is no efficient flow cytometry method to enumerate wild type bacteria cells in real time within a dense background of non-living particles. If all the prey species are fluorescent, they can be separated, counted and compared against each other directly. Pairwise t-test used in such experiments is more powerful than pooled t-test comparison and ANOVA used in dilution experiments. For example, comparison between EcR and EcG (Trial 7, $p=0.48$), and between EcR and EaG (Trial 6, $p=0.15$) in the same grazing systems gave no significant difference in grazing rates. However, when EcR and EcRD (DTAF-stained EcR, Trial 8) were mixed as prey, Hflag again showed clear preference of the former ($p=0.007$). We chose EcR for simultaneous comparison of grazing on FP-labeled and DTAF-labeled tracer, because active EcR cells have a strong FL2 signal, while DTAF-stained EcR (DTAF-EcR) fluoresce in green but a majority of the cells were killed by heat-treatment and maintained weakened red fluorescence. This pair provided prey of exactly the same strain with different spectrum.

Sherr *et al.* (1987) showed that DTAF-stained FLB could support predator's growth similar to unstained bacteria. This is from the grazer's point of view, however. There is no guarantee that chemically stained and unstained bacteria are consumed at the same rates, which is a more important issue concerning the effect of bioremediation. Unfortunately, in several cases artifacts of chemical staining have been observed. Christoffersen *et al.* (1997) examined protist food vacuoles with immunofluorescence

labeling after bacteria ingestion and found ingestion rates were substantially higher for living bacteria than for FLB. This is in accordance with the hypothesis that selective grazing of the more actively growing cells helps explain the ability of slow-growing cells to persist in bacterioplankton assemblages (Sherr *et al.*, 1992). Therefore, observation of bacteria in predator food vacuole and bacteria at large suggest that DTAF-staining increases the proportion of inactive prey cells and artificially lowers overall bacterivory measurement. Moreover, compared to DTAF-stained FLB, bacterivorous ciliates showed significantly higher grazing rates on bacteria stained with CTC, which stains vital bacteria at room temperature (Epstein and Rossel, 1995). Combined CTC-DTAF staining also proved that DTAF-staining heat-kills most bacteria (Vishvesh *et al.*, 1999). These observations, as well as our results, make the use of DTAF in bacterivory experiments questionable when bioremediation is involved, which depends on the growth and active metabolism of bioremediative bacteria.

There are several possible explanations for the lowered grazing rates of DTAF-stained bacteria. DTAF can stain dead bacteria but GFP/RFP expression is restricted in living-cells, which were more motile and more frequently grazed upon (Verity, 1991). DTAF-staining procedure may also chemically alterred prey cells to the extent that affected Hflag physiologically to cause selection against DTAF-stained prey. DTAF-staining may boost the nutrient value of prey cells and fulfill protists' need for digestible elements. Further investigation is needed to examine these possibilities.

Figure 8 showed that DTAF, as a covalently bound chemical stain; accumulates faster and stayed fluorescence longer than fluorescent proteins. Apparently, fluorescence from *in vivo*-expressed proteins is affected by protist digestion. And digestion capacity

does contribute to grazing preference, such as differentiation in environmental elimination of enteric bacteria (Iriberri *et al.*, 1994). RFP is more resistant to photobleaching, and more stable in a low pH environment (Frandkov *et al.*, 2000; Geoffrey *et al.*, 2000; Yang *et al.*, 1996). This is consistent with our observation that RFP fluoresced longer than GFP-labeled prey in Hflag food vacuoles. Though less suitable for long time experiments without sample fixation, fluorescent proteins are less likely to cause adversary effects for grazers than chemical stains that is not as readily digested.

2) Prey type *vs.* prey cell size

Analysis of how prey cell size affect protistan bacterivory was based on FSC signals, which is proportional to particle size and can be used to infer comparative differences in cell sizes for prey cell populations. By correlating the FSC signal with the fluorescence signal for each particle counted, it is possible to determine size distributions for each differently labeled prey types in a mixed prey population. Changes in prey size distribution over a time series can then be used to infer the mean size of the cells that were consumed during a given time interval.

In order to use LIVE-FCM to distinguish between effects of prey type and prey size on grazing preference it is important to determine that this labeling method does not alter prey size. In these experiments no difference was observed between GFP/RFP-expressing bacteria and wild type bacterial cells of the same species. This is in agreement with the results of Tombolini *et al* (1997), who expressed GFP in *Pseudomonas fluoresces* A506 and found no change in its size and surface granularity as indicated by FSC and SSC signals, respectively.

Prey size and shape have been considered to be the primary determinants for protistan grazing preference (Hahn and Höfle, 1999; Kinner *et al.*, 1998; Simek and Chrzanowski ,1992). Two different roles of prey cell size concerning protistan grazing preference between prey species and within prey species were evident in Trial 3 and 5.

The results indicated that, at least for Hflag, prey type may supercede prey size as a criterion for determing prey preference. Both EaG, which is similar in size to *M. pusilla*, and PpG, which is larger than *M. pusilla*, are consumed preferentially to *M. pusilla.* Hflag grazing on *M. pusilla* was suppressed by coexistance with either EaG or PpG, as indicated by the weakly significant difference in Mp grazing rates between Trial 2 and 3 (P=0.10). On the other hand, no significant difference existed for Hflag grazing on EaG (P=0.825) and PpG (P=0.263) in the presence or absence of *M. pusilla.*

Flow cytometry bacterivory experiments recorded detector signals along a time series, therefore it's possible to subtract information obtained at one time point from another, producing time kinetics of the corresponding property. In grazing trials 4 and 5, analysis of changes in prey cell size distribution over a time series demonstrated that PpG cells consumed by Hflag were significantly smaller than the average for the total PpG population. Therefore, the mean size of consumed cells and the average size of the PpG cell population remaining after predation increased significantly over the time course of the grazing experiment. No significant size preferences were observed for the smaller prey species Ea and Ec, suggesting that the size range of PpG included cells larger than the prefered size range for Hflag thus smaller PpG cells were preferentially removed by grazing pressure.

3) Predator conditions

Grazing rates by Hflag on EaR showed great amount of variance between different batches and culturing media. Hflag batch 1 & 2 in table 4 are cultured with organic media, which gave much lower grazing rates than Hflag cultured on mineral media (comparing Trial 1 vs 10, and 6 vs 7). Variations in Hflag grazing may also be attributed to possible nutrient and antibiotic carry-over from prey enriching media or staining buffer to the grazing system. Heteronanoflagellates are sensitive to various antibioitics such as penicillin and chloramphenicol (Sherr 1986). Therefore it's important to wash cultured bacteria exhaustively before mixing them with grazers. Sufficient washing also helps reduce nutrient carry-over to the oligotrophic grazing system (Vazquez-Dominguez *et al.*, 1996). Considering the potential effect of pollutants and the procedures of bioremediation on protists, the variables influencing bacterivory on the part of protists need to be studied in the future as thoroughly as those of prey bacteria.

Overall, bioremediation-related bacterivory experiments by LIVE labeling and flow cytometry showed several advantages over previous methods. First of all, a flow cytometer sample takes at most 3 min to run through, much faster than any microscopic or imaging analysis. Second, a flow cytometer detects and correlate scattered light signals and fluorescence and multiple wavelength simultaneously. Third, no sample fixation or addition of secondary reagent is required for GFP/RFP flow cytometry, allowing real time examination of live prey. Furthermore, information processing for flow cytometry results is much more easily automated (Chapter IV). Therefore, LIVE-FCM method is

promising for extensive protistan bacterivory analysis, including factors rarely studied such as temperature, light intensity, liquid flow, etc.

There are certain pitfalls in application of flow cytometry for bacterivory analysis. Flow cytometry methods are not applicable for measuring consumption rates of attached bacteria. And it is noteworthy that in some cases attached bacteria or biofilms are selected for bioremediation purposes (Holman *et al.*, 2002; Campell *et al.*, 2001). Another limiting factor is that prey and predator must be be of sufficiently different size (FSC/SSC signals), if they don't have distinct fluorescence properties. The light sources of flow cytometers are of limited ranges of wavelength, impeding detection of RFP in our experiments, as well as GFP variants of other colors that may be used in the future.

IV. INFORMATION PROCESSING USING PERL

After discussing the results of data analysis the GR program provided, this chapter will concentrate on the functional features of the program itself. GR was originally created with Perl 5.0 on Solaris. Since Perl interpreters for many operating systems are available, GR can be easily ported to other platforms such as Linux, Windows[1], Macintosh, etc. The purpose of GR is to manipulate statistics data created by flow cytometry software (see Appendix I for program source code).

File Types Used by GR

Two types of files are required to use GR:

1. An experiment file (.exp suffix is optional) is an original statistics file created by flow cytometry software. At this time the preferred way to create such a file is through a histogram window within WFI (Figure 10). WFI has the powerful feature of filename increment/decrement that allow easy scanning through data series for the same experiment and combining the statistic output. In order to keep the original experiment readouts, experiment files should be set read-only and not modified after scanning through the file series. The following is an example of the plain text read-outs:

Multiple Document Interface for Flow Cytometry
WinMDI Version 2.8 - Windows 3.95/DOS 7.10

[1] There are two problems with GR running on a Windows PC: 1). Perl programs conflicts with McAfee on a PC with a floppy drive. The first GR user input triggers visiting of the floppy drive and returns hardware failure messages. Therefore McAfee should be terminated before starting GR; 2). System complains with "Bad command or file name" but GR functions normally when PATH environment variable does not cover all system command directories.

Mon Oct 01 15:45:43 2001

Gates: R1

Project: Experiment:

File: Expt1Bin.001 Sample: Control#1 T=0min

Date: 18-Oct-0 Parameters: 5

Total Events 14880 Gated Events 7 0.05%

System: Log Parameter Means: Arithmetic

Param name	Events	%Total	%Gated	Median	Mean	CV	Peak,Value
FSC-	7	0.05	100.00	259.46	293.18	122.45	1,582.942
SSC-	7	0.05	100.00	203.51	359.21	83.03	1,991.046
FL1-	7	0.05	100.00	1.19	1.26	2728.38	2,1
FL2-	7	0.05	100.00	1.10	1.32	3093.35	1,1.91095
FL3-	7	0.05	100.00	1.81	4.10	3920.11	1,9.64662

2. A Volume file (.vol suffix is mandatory) is a read-only file recording volume of the sample used in the flow cytometry run-through. Volume files are necessary for flow cytometers that cannot measure sample liquid volume automatically. Lines in the file should include a sample name and its volume data separated by a blank space.

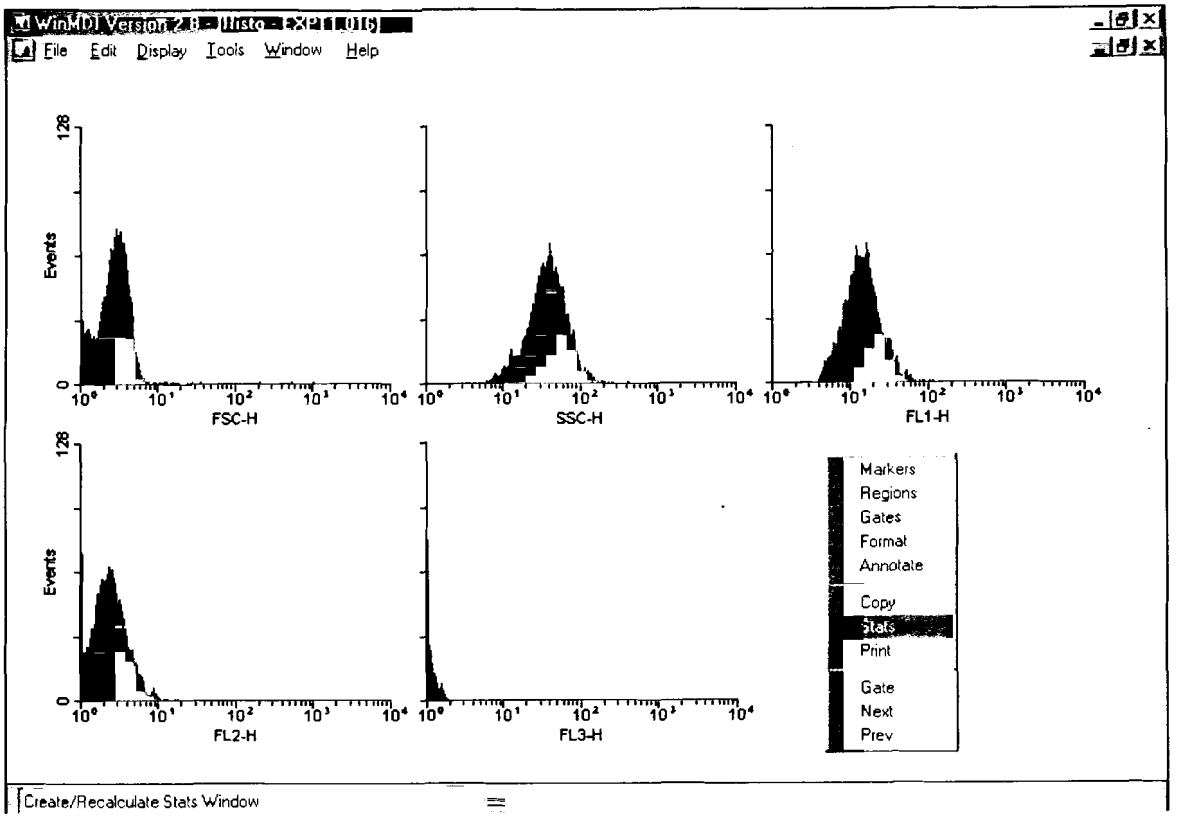


Figure 10. Experiment files were created within WFI histogram window.

Four types of files can be generated by GR:

1. GR Data file (.grd) is saved during a GR session if modifications on the data need to be kept permanently (since the original experiment files are read-only). Both modified experiment sample readouts statistics and volume data can be saved in data files that are rewritable. The format is similar to that of an experiment file without excessive headlines. A user can open a data file in a text editor and add comment lines starting with an "#".

2. Log file (grlog) is generated during the first use of GR and appended every time GR is used. Grlog is used to record data modifications and program output, which should not be tampered by users. In some cases grlog format is modified from actual GR to

accommodate text cut-and-paste from grlog. Handling of grlog is operating system specific in order to keep grlog read-only.

3. GR rate file (.grr) contains the results of grazing rates calculation and can be reloaded to GR for further statistical comparison. Rate files can also be generated manually. Each line contains a calculation condition (before a colon) and a grazing rate , as shown by the following example:

SAMPLE_EAGA_PREY_EAG_PREDATOR_PREDATOR_FROM_0_TO_45_MIN:0.2
SAMPLE_EAGA_PREY_EAG_PREDATOR_PREDATOR_FROM_45_TO_90_MIN:0.383
SAMPLE_EAGA_PREY_EAG_PREDATOR_PREDATOR_FROM_90_TO_135_MIN:0.315

4. GR macro file (grmacro) records all user input in a GR session, which can be or slightly modified and used for input redirection for repeated data analysis. For example, the following is the macro used to calculate grazing rates in Trial 8 in Table 4:

```
1
grd/e5.grd
2
d
ldbactcontrolat=180

y
q
4
5
lda
ldbt
ldc
ldd
lde
0-90
3
ldbactcontrola
ldbactcontrolb
ldbactcontrolc

e

2
```

```
ecr
dtaf

ld
6
```

Instructions for Using GR

GR provides context-sensitive help information. By typing '!' at a user input prompt, GR will give specific directions. Typing 'gr –h' will print the content of this chapter as the general help information.

GR is started by typing "gr" on the command line (see next section for optional arguments). The first time GR is used, it tries to determine what operating system it is running on and what choices the user might have to run GR (whether running a pre-compiled version or using Perl interpreter). Then GR enters the normal main function loop by showing the following menu:

Select:
1 to load flow cytometry readout or grazing rates data;
2 to annotate/analyze loaded data;
3 to do statistic and time kinetic analysis;
4 to calculate grazing rates;
5 to compare pre-computed grazing rates;
6 to quit.
:

The term "session" is used hereafter to indicate the interval between starting GR and terminating it (by choosing '6' on the main menu, pressing "Ctrl-C", or issuing Unix 'kill' commands). After starting a GR session, the first step a user should always take is to choose 1 from the menu to load data files. If extension names are omitted, files are

matched in the following order: GR data files (.grd), experiment files (.exp), volume files (.vol) and GR rate files (.grr). Volume files should be loaded after the corresponding sample readout files (.exp or .grd). Using Perl associative array, GR stores sample data into a hierarchy of Sample => Gate (including 'VOLUME') => Parameter (detector channels) => Statistics term (Figure 11). To confirm data loading and avoid redundancy, GR will report how many sample/gate combinations or sample volumes it reads from loaded files. The bacterivory assay is based on a time-series of samples, therefore GR requires the sample names to be suffixed with a time label "T=xxxMIN", such as "EAGMPT=45MIN" (grazing sample for EaG *and M. pusilla* at 45 min). Non-time-labeled sample names will generate warnings, but still read into memory during loading step.

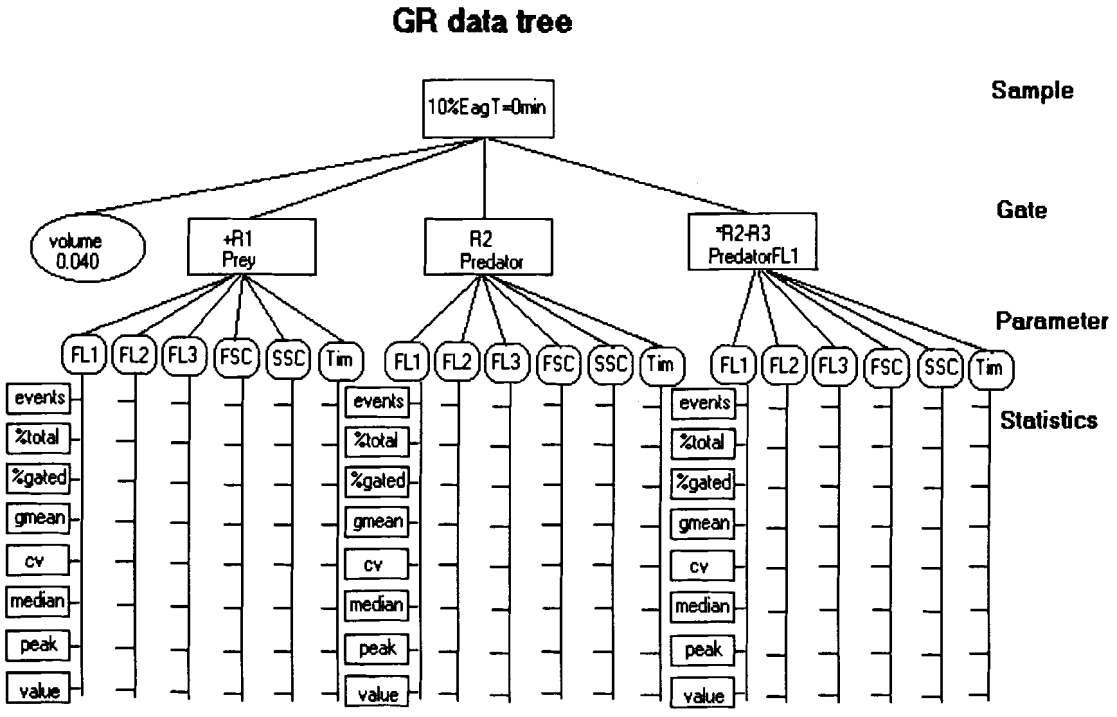


Figure 11. GR internal organization of flow cytometry sample readout data.

The user can now restructure data by choosing 2 from the main menu, which will give a data submenu:

Select again:
A: alias sample/gate;
B: browse/midify data;
D: delete data;
L: list sample names;
S: save modified data;
U: undo modifications;
Q: quit to upper level.
:

These menu items are self-explanatory. If the user does not yet have an overall picture of data quality, he/she can choose 'B' to browse data. GR will prompt for sample name pattern, gate name pattern, parameter name and statistical term to match data item(s) be displayed. A full sample name allows GR to determine and display all of its available gates. Alternatively, partial sample pattern even Perl Regular Expression (RegExp) can be used to indicate a pool of samples (Wall *et al.*, 1996). For example, using '.' to match all available samples. A prompt for a gate pattern can be similarly handled with the caution that RegExp control characters such as "+" and "*" should be escaped by prepending a backslash ("\"). Parameter names and statistical terms must be exact matches. If only one data item matches user specification, GR will give the option to change the value of that item. Data modifications are necessary only when volume is

not properly recorded with data, or as a result of WFI malfunction in rare cases, therefore GR does not allows massive data modification.

To use more intuitive terms, the user may also choose to alias certain sample or gate name(s). The original sample/gate patterns and new patterns specification can use RegExp. After aliasing the original data can be deleted by choosing 'D' from data submenu to reduce the size of GR data files. GR keeps tracks of all data modifications for the current session. All modified or deleted data can be recovered using the undo function before this GR session is terminated.

Comparison of flow cytometry signals between samples of different treatment or different time points can reveal important features of protistan grazing. Designed for such comparisons, GR main menu item '3' will generate the statistics submenu:

Select again:
M: detector signal comparison;
T: time kinetic analysis;
Q: quit to upper level.
:

Choose 'M' to compare the mean values of a parameter (detector) between samples, which should be specified in the same way as data browsing steps ('2'->'B') mentioned above, except that the statistic term is fixed to be 'MEAN'. Two-tailed Student's t-test and ANOVA (Analysis of Variation) are used to detect significant difference. Comparisons are done at two levels: first a sample mean comparison is performed between each pair of samples, and then user-defined sample sets are compared to each other. The degree of freedom is determined by 'EVENTS' terms of the

corresponding samples for sample mean comparison, and by the number of samples in each set for sample set comparison. Detector signal time kinetics is calculated by choosing 'T' in the sub menu, for which GR will ask for sample pattern, gate pattern and time ranges. This calculation uses 'MEAN' and 'EVENTS' terms for samples at different time points to find shifting in signal distributions as a result of protistan grazing during the interval.

After data reorganization and statistical analysis, the user can start rates calculation by choosing '4' on the main menu. GR will prompt for number of treatment replicate, treatment sample time-series names (i.e., name without time labels shared by all samples of the same time series) and time ranges to be used for rates calculation. Next GR asked for specification of prey control flasks in the same manner, if the user specifies that the number of prey control replicates is larger than 0. Then the user is asked to designate calculation model, predator gate and prey gate names. These names are either exact matches, or set to default "exponential", "PREDATOR" and "PREY", respectively, if they are omitted by pressing enter. If all time ranges are accepted, GR will handle deleted outlier time point samples by extending rate calculation according to availability of the time points before and after the deleted time point.

The grazing rates are displayed in a table format and can be saved to a GR rate file. Average values and stand deviations of each row and column are appended, such as the following example:

Preygate PREY

	AG100%A	AG100%B	AG100%C	SAMPLE_AVE+/-S.D.
0--50Min	0.714	0.671	>	---> 0.692+/-0.0301
50--103Min	0.504	0.423	0.613	---> 0.513+/-0.0957
103--156Min	0.414	0.567	0.849	---> 0.61+/-0.221
TIME AVE.	0.544	0.554	0.731	---> 0.594+/-0.15
+/-S.D.	0.154	0.125	0.167	

By choosing '5' on the main menu, the user can compare rates either loaded previously or calculated during the current session. The grouping and statistical methods used are similar to sample mean statistics mentioned above.

Command Line Arguments and Environmental Variables

GR can take the following command line arguments:

l <file> immediately load the file after starting GR, the next word being the file name;

p<os> specify current Operating System, must be the last argument in the first word;

v perform perl detection and display execution recommendations;

w print warning information during data loading and manipulation operations;

h print help information

m write user input to grmacro for later user redirection.

s<decimal> use a statisticly-significant probability level other than 0.05.

GR gathers information from these optional environment variables:

USER used only on Microsoft operating systems for logging purpose;

GROS	operating system specification, may be overwritten by command line option p;
PATH	used as the last resort to discern current operating system;

REFERENCES

Amman E, J. Brosius, and M. Ptashne. 1983. Vectors bearing a hybrid *trp-lac* promoter useful for regulated expression of cloned genes in *Escherichia coli*. Gene *25*: 167- 178.

Antoine, R. and C. Locht. 1992. Isolation and molecular characterization of a novel broad-host-range plasmid from *Bordetella bronchiseptica* with sequence similarities from Gram-positive organisms. Mol. Microbiol. 6(13):1785-1799.

Arai, H., S. Akahira S, T. Ohishi, T. Kudo. 1999. Adaption of *Comamonas testosteroni* TA441 to utilization of phenol by spontaneous mutation of the gene for a trans-acting factor. Mol. Microbiol. 33(6): 1132-1140.

Baird, Geoffrey S., D. A. Zacharias and R. Y. Tsien. 2000. Biochemistry, mutagenesis, and oligomerization of DsRed, a red fluorescent protein from coral. Proc. Natl. Acad. Sci. USA 97:11984-11989.

Bhupathiraju, V. K., M. Hernandez, D. Landfear, and L. Alvarez-Cohen. 1999. Application of a tetrazolium dye as an indicator of viability in anaerobic bacteria. J. Microbiol Methods 37(3):231-243.

Bloemberg, GV, A. H. M Wijfjes, G.E.M. Lamers, N. Stuurman, and B.J.J. Lugtenberg. 2000. Simultaneous imaging of Pseudomonas fluorescens WCS365 populations expressing three different autofluorescent proteins in the rhizosphere: new perspectives for studying microbial communities. Mol. Plant Microbe Interact. 13: 1170-1176.

Campbell, S.C., T. R. Clark, and G. McFeters. 2001. Biogenic Production of Cyanide and Its Application to Gold Recovery. J. Industrial Microbiol. & Biotechnol., 26:134-139.

Christensen, B. B,, C. Sternberg, and S. Molin. 1996. Bacterial plasmid conjugation on semi-solid surfaces monitored with the green fluorescent protein (GFP) from *Aequorea victoria* as a marker. Gene 173(1Spec No):59-65.

Christoffersen K, O. Nybroe, K. Jürgens, and M. Hansen. 1997. Measurement of bacterivory by heterotrophic nanoflagellates using immunofluorescence labeling of ingested cells. Aquatic. Mairine Ecol. 13:127-134.

Cubitt, A. B, R. Heim, S. R. Adams, A. E. Boyd, L. A. Gross and R. Y. Tsien. 1995 Understanding, improving and using green fluorescent proteins. Trends Biochem. Sci. 20:448-455.

Daubaras, Dayna & A.M. Chakrabarty. (1992). The environment, microbes and bioremediation: microbial activities modulated by the environment. Biodegradation 3:125-135.

Dower, William J., J. F. Miller and C. W. Ragsdale. 1988. High efficiency transformation of *E.coli* by high voltage electroporation. Nucleic Acids Research 16:6127-6145.

Eberhard W. G. 1990. Evolution in bacterial plasmids and levels of selection. Q. Rev. Biol. 65:3-22.

Elowitz, Miahael .B., M. G. Surette, P. Wolf, J. B. Stock, and S. Leibler. 1997. Photoactivation turns green fluorescent protein red. Current Biology 7: 809-812.

Epstein S. S. and J. Rossel. (1995) Methodology of in situ grazing experiments: evaluation of a new vital dye for preparation of fluorescently labeled bacteria. Mar. Ecol. Prog. Ser. 128:143-150.

Farinha M.A and A.M. Kropinski. 1990. High efficiency electroporation of *Pseudomonas aeruginosa* using frozen cell suspensions. FEMS Microbiol Lett 58(2):221-5.

Focht DD., D. B. Searles, and S. C. Khi (1996) Genetic exchange in soil between introduced chlorobenzoate degraders and indigenous bihenyl degraders. Appl Environ Microbil 62:3910-3913.

Fornari, C. S., and S. Kaplan. 1982. Genetic transformation in *Rhodopseudomonas sphaeroides* by plasmid DNA. J. Bacteriol. 152:89-97.

Frandkov, A. F., Y. Chen, L. Ding, E. V. Barsova, M. V. Matz and S. A. Lukyanov. 2000. Novel fluorescent protein from *Discosoma* coral and its mutants possesses a unique far-red fluorescence. FEBS Letters 479:127-130.

Furste, J. P., W. Pansegrau, R. Frank, H. Blocker, P. Scholz, M. Bagdasarian and E. Lanka. (1986) Molecular cloning of the plasmid RP4 primase region in a multi-host-range tacP expression vector. Gene 48:119-131.

Gross, L. A., S.B. Geoffrey, R.C. Hoffman, K.K. Baldridge, and R.Y. Tsien. 2000. The structure of the chromophore within DsRed, a red fluorescent protein from coral Proc. Natl. Acad. Sci. USA, 97(22):11990-11995.

Guillard R.R.L and P. Hargraves, P (1993). Stichochrysis immobilis is a diatom, not a chrysophyte. Phycologia 32: 234-236.

Hahn, M. W. and M.G. Höfle. 1998. Grazing Pressure by a Bacterivorous Flagellate Reverses the Relative Abundance of Comamonas acidovorans PX54 and Vibrio Strain CB5 in Chemostat Cocultures. Appl. Environ. Microbiol. 64: 1910-1918.

Hahn, M. W. and M.G. Höfle. 1999. Predation on a Bacterial Model Community: Interplay of Size-Selective Grazing, Specific Bacterial Cell Size, and Bacterial Community Composition Appl. Environ. Microbiol. 65(11):4863-4872.

Heikal, A. A., T. H. Samuel, S. B. Geoffrey, R. Y. Tsien, and W.W. Webb. Molecular spectroscopy and dynamics of intrinsically fluorescent proteins: Coral red (DsRed) and yellow (Citrine). 2000. Proc. Natl. Acad. Sci. USA 97(22):11996-12001.

Holman, H.-Y. N., K. Nieman, D.L. Sorensen, C.D. Miller, M.C. Martin, T. Borch, W.R. McKinney, and R.C. Sims. 2002. Catalysis of PAH Biodegradation by Humic Acid Shown in Synchrotron Infrared Studies. Environ. Sci. Technol. 36(6):1276-1280.

Iriberri, J., I. Azúa, A. Labirua-Iturburu, I. Artolozaga, and I. Barcina. 1994. Differential elimination of enteric bacteria by protists in a freshwater system. J. of Appl. Bacteriol. 77:476-483.

Jurgens, K., J. Pernthaler, S. Schalla, and R. Amann. Morphological and compositional changes in a planktonic bacterial community in response to enhanced protozoan grazing. 1999. Appl. Environ. Mcrobiol. 65:1241-1250.

Kinner, N.E., R. W.Harvey, K. Blakeslee, G. Novarino, and L. D. Meeker. 1998. Size-selective predation on groundwater bacteria by nanoflagellate in an organic-contaminated aquifer. Appl. Environ. Microbiol. 64:618-625.

Lowder, M., A. Unge, N. Maraha, J. K. Jansson, J. Swiggett, and J. D. Oliver . 2000. Effect of Starvation and the Viable-but-Nonculturable State on Green Fluoresccent Protein (GFP) Fluorescence in GFP-tagged *Pseudomonas fluoresces* A506. Appl. Environ. Microbiol. 66: 3160-3165.

Lybarger, Lonnie and R. Chervenak . 1999. Fluorescent proteins in single- and multicolor flow cytometry. Green Fluorescent Protein. Methods Enzymol. 302:189-199.

Mendel, JL, and M.S. Walton. 1966. Conversion of p,p' -DDT to p,p' -DDD by intestinal flora of the rat. Science 151(717):1527-1528

Nagaard, K. and D. O. Hessen. 1990. Use of ^{14}C-protein-labelled-bacteria for estimating clearance rates by heterotrophic and mixotrophic flagellates. Mar. Ecol. Prog. Ser. 40:185-193.

Nieto C., E. Fernandez-Tresguerres, N. Sanchez, M. Vicente, and R. Diaz. 1990. Cloning vectors, derived from a naturally occurring plasmid of *Pseudomonas savastanoi*, specifically tailored for genetic manipulations in Pseudomonas. Gene 87: 145-149.

Sambrook, J., E. Fritsch, and T. Maniatis. 1989. Molecular Cloning: A Laboratory Manual (Second edition). Cold Spring Harbor Press, Cold Spring Harbor, NY.

Shapiro, H. M. 1994. Practical Flow Cytometry, 3rd Edition. Wiley Liss, New York. ISBN 0-471-30376-3.

Sherr, B. F., E. B. Sherr, and R. D.Fallon. 1987. Use of monodispersed fluorescently labeled bacteria to estimate in situ protozoan bacterivory. Appl. Environ..Microbiol. 53(5):958.

Sherr, B. F., E. B. Sherr, T.L. Andrew, R. D. Fallon and S. Y. Newell. 1986. Trophic interactions between heterotrophic protozoa and bacterioplankton in estuarine water analysed with selective metabolic inhibitors. Mar. Ecol. Prog. Ser. 32:169-179.

Sherr, B. F., E. B. Sherr and J. McDaniel. 1992. Effect of Protistan Grazing on the Frequency of Dividing Cells in Bacterioplankton Assemblages. Appl. Environ. Microbiol. 58:2381-2385.

Sherr B.F., E. B. Sheer and T. Berman. 1982. Decomposition of organic detritus: a selective role for microflagellate protozoa. Limnol. Oceanogr. 27(4): 765-769.

Sieracki., M. E., L. W. Haas, D. A. Caron and E. J. Lessard. 1987. Effect of fixation on particle retention by microflagellates: underestimation of grazing rates. Mari. Ecol. Prog. Ser. 38:251.

Sime-Ngando, T., S. Demers, and S. K. Juniper. 1999. Protozoan bacterivory in the ice and the water column of a cold temperate lagoon. Microbial Ecology 37:95.

Simek, K., Pernthaler, J. Posch, T. Vrba, J. Amann, and R. Psenner. 1997. Contrasting bacterial strategies to coexist with a flagellate predator in an experimental microbial assemblage. Appl. Environ. Microbiol. 63 (2) 596-601.

Simek,K and T. H. Chrzanowski. 1992. Direct and indirect evidence of size-delective grazing on pelagic bacteria by freshwater nonoflagellates. Appl Environ Microbiol 58:3715:1157-1163, and references therein.

H.-M.Tan. Bacterial catabolic transposons. Appl Microbiol Biotechnol (1999) 51:1-12.

Temple C. A., G. N. George, J. C. Hilton, M. J. George, R. C. Prince, M. J. Barber, and K.V. Rajagopalan. 2000. Structure of the Molybdenum Site of Rhodobacter sphaeroides Biotin Sulfoxide Reductase. Biochem. 39:4046-4052.

Tombolini, Riccardo, A. Unge, M. E. Davey, F. J. de Bruijn, and J. K. Jansson.(1997). Flow cytometric and microscopic analysis of GFP-tagged *Pseudomonas fluoresces* bacteria. FEMS Microbiology. Ecology. 22: 17-28.

Tso, S. F and G. L. Taghon. 1997. Factors Affecting Predation by *cyclidium* sp. and *Euplotes* sp. on PAH-Degrading and Nondegrading Bacteria. Microb. Ecol. 37:3-12.

U.S. Environmental Protection Agency Office of Research and Development (1995). Bioremediation of Hazardous Wastes: Research, Development and Field Evaluations. Publication Document Number: EPA 540-R-95-532. Available online at http://207.86.51.66/download/remed/biosym.pdf.

Van der Rest, M. E., C. Lange, and D. Molenaar. 1999. A heat shock following electroporation induces highly efficient transformation of *Corynebacterium glutamicum* with xenogeneic plasmid DNA. Appl. Microbiol. Biotechnol. **52:** 541-545.

Verity P. G. 1991. Feeding in planktonic protozoans: Evidence for non-random acquisition of prey. J. Protozool. 38:69-76.

Yang, F., L. G. Moss, and G. N. Phillips, Jr. 1996. The Molecular Structure of Green Fluorescent Protein. Nature Biotechnology 14:1246-1251.

APPENDIX: Source code of GR

```
#!/usr/local/bin/perl
use POSIX qw(floor log10 atan);
@options=@ARGV;
undef @ARGV;
#Some pecularities occurred during user input if @ARGV persists because
Perl considers arguments to be filenames by default.
if($options[0]=~/h/i) {&help();}
$os="Unknown";
if($options[0]=~/p([^\s]+)$/)
{$os=$1;$options[0]=~s/p$os//;}
if(! -f 'grlog')
 {if($options[0]!~/c/i) {$options[0]="c".$options[0];}
 if($options[0]!~/v/i) {$options[0]="v".$options[0];}
 }
if($options[0]=~/v/i)
 {$perlver=`perl -v`;
 if($perlver eq '')
  {print "Perl interpreter detection failed. You have to use this
compiled GR program.\n";
  }
 else
  {$perlver=~/This is perl,([version\ \d\.\_]+)built for
([^\n]+)/;$os=$2;
  print "You have Perl$1($2) installed. GR recommends you to get the
source code and interpret it by Perl for better performance and maximal
flexibility.\n";
  }
 if($options[0]!~/c/i){exit;}
 }
if($os eq 'Unknown')
 {$perlver=`perl -v`;
 if($perlver=~/This is perl,([version\ \d\.\_]+)built for
([^\n]+)/){$os=$2;}
 }
if($os eq 'Unknown' && $ENV{'GROS'}!='') {$os=$ENV{'GROS'};}
if($os eq 'Unknown')
 {$path=$ENV{'PATH'};
 if($path=~/\\/) {$os="MSWin";}
 else {$os="Unix";}
 }
#Four ways to determine operating system with their priority order:
option p; perl -v output; GROS environment variable; PATH variable
format. OS determination is important for proper file privilege
settings.
@grtime=localtime(time);
$grtime[5]+=1900;
$grtime[4]+=1;
if($grtime[2]<10) {$grtime[2]="0".$grtime[2];}
if($grtime[1]<10) {$grtime[1]="0".$grtime[1];}
if($grtime[0]<10) {$grtime[0]="0".$grtime[0];}
if($grtime[8]) {$DST="DST";}
$fdlogtemp="LOGGED AT $grtime[2]".":".$grtime[1].":".$grtime[0]." $DST
ON ".$grtime[4]."/".$grtime[3]."/".$grtime[5]."\n";
if ($os=~/MSWin/)
 {if(-f "grlog") {system("attrib -r grlog");}
```

```
 $gruser=$ENV{'USER'};
 if($gruser ne '') {$gruser=" BY $gruser";};
 $grpath=`cd`;chomp $grpath;
 if($grpath ne '') {$grpath=" IN $grpath";};
 $fdlogtemp.="$gruser$grpath";
 }
else
 {if(-f "grlog") {system("chmod u+w grlog");}
 $gruser=`who am i`;$gruser=~s/[\ \t]{1,}/\ /g;
 chomp $gruser;
 $grpath=`pwd`;
 $fdlogtemp.="BY $gruser IN $grpath";
 }
if($options[0] ne ''){$fdlogtemp.=" WITH OPTIONS: @options";}
$fdlogtemp.="\n";
if(!open(fdlog,">>grlog")) {print "Unable to open logfile!\n";exit;}
else {print fdlog $fdlogtemp;}
#Writing a session head to grlog.
if($options[0]=~/1/i) {$select='L';}
print "\'GR -h\' for general help information. '!' for context-
sensitive help.\n";
if($options[0]=~/m/i)
 {if(! open(fdmacro,">grmacro"))
  {printoutput("Unable to write to grmacro!\n");
  $macro=0;
  }
 else {$macro=1;}
 }
while(1)
 {if($select ne 'L')
  {printoutput("\nSelect:\
1 to load flow cytometry readout or grazing rates data;\
2 to annotate/analyze loaded data;\
3 to do statistic and time kinetic analysis;\
4 to calculate grazing rates;\
5 to compare pre-computed grazing rates;\
6 to quit.\n:");
  $select=<>;
  print fdlog $select;
  printoutput("\n");
  if($macro) {print fdmacro $select};
  }
 if($select==1 || $select eq "L") {&load();$select=1;}
 if($select==2) {&data();}
 if($select==3) {&anal();}
 if($select==4) {&calc();}
 if($select==5) {%comparestatdata=%ratestatdata;&comp();}
 if($select==6)
  {print fdlog "\n";
  close fdlog;
  if($macro) {close fdmacro;}
  if($os=~/MSWin/) {system('attrib +r grlog');}
  else {system('chmod a-w grlog');}
#Keeping GRlog protected since users are not supposed to manually edit
it.
  exit;
  }
```

```
 }

sub load()
{
if($select ne 'L')
 {printoutput("Input file name (.grd, .exp, .vol or .grr):");
 $filename=<>;
 while($filename=~/\!/)
  {print "Enter a name of your GR data file, experiment file, volume
data file or precomputed rate file. If you omit the extension name, GR
will try to match it in the above order. Volume data take effect only
for samples already loaded. Wildcards are not supported.\n";
  print "Input file name (.grd, .exp, .vol or .grr):";
  $filename=<>;
  }
 print fdlog $filename;
 if($macro){print fdmacro $filename;}
 }
else {$filename=$options[1];}
chomp $filename;
printoutput("\n");
if ($filename eq '') {return;}
if(! -f $filename)
 {if($filename!~/\.grd/ && -f $filename.".grd") {$filename.=".grd";}
 if($filename!~/\.exp/ && -f $filename.".exp") {$filename.=".exp";}
 if($filename!~/\.vol/ && -f $filename.".vol") {$filename.=".vol";}
 if($filename!~/\.grr/ && -f $filename.".grr") {$filename.=".grr";}
 }
#GR tries to match extension names in order when they are omitted.
if(!open(fd1,"<".$filename))
 {printoutput("Unable to open $filename!\n"); return;
 }
if($filename=~/\.vol/) {&loadvolume();return;}
if($filename=~/\.grr/) {&loadrate();return;}
#Two variables to resolve statistic terms in an experiment file. This
allows GR to adapt to different statfile.txt format.
undef $statcount;
undef %loadstat;
$samplegatenum=0;
$newsamplegatenum=0;
$replacedsamplegatenum=0;
$linenum=0;
$paramindex=0;
$newsamplenum=0;
$paramnum=100;
#Initializing various counters and data storage variables.
while(<fd1>)
 {$linenum++;
 chomp;
 if(/^Volume\ of\ Sample\ (\S+)\ :([\d\.]+)/)
  {$volumesample=$1;
  ${"Sample_".$volumesample}{'VOLUME'}=$2;
  next;
  }
 if(/^Gates:\s+(\S+)/) {$gate=$1;$gate=~tr/a-z/A-
Z/;$gate=~s/\s//g;next;}
 if(/Parameters:\ (\d+)/) {$paramnum=$1;$paramindex=0;next;}
```

```
 if(/Sample:\ (.+)/)
  {$sample=$1;$sample=~s/\s+//g;$sample=~tr/a-z/A-Z/;$samplegatenum++;
  if($sample!~/T=\d+\s*MIN/ && $options[0]=~/w/)
   {printoutput("\nWarning: Sample-$sample:Gate-$gate is not time-
labled!");
   }
#Sample names need to be time-labeled for kinetic analysis.
 if(! defined %{"Sample_".$sample."_Gate_".$gate})
{$newsamplegatenum++;}
  else
   {if($options[0]=~/w/)
    {printoutput("\nWarning: Sample-".$sample.":Gate-".$gate." is
replaced.");
    }
   $replacedsamplegatenum++;
   }
#Optional prompt for replicated sample:gate combinations.
  next;
  }
 if(/^Param name\s+[^\s\,]+[\s\,]+/)
  {if(defined $statcount){next;}
  s/\%/PC/g;s/\,/ /g;tr/a-z/A-Z/;
  do
   {s/(PARAM NAME\s+)([^\s]+)/$1/;
   $statcount++;
   $loadstat{$statcount}=$2;
   if($loadstat{$statcount} eq "GMEAN"){$loadstat{$statcount}="MEAN";}
   }while(/^PARAM NAME\s+[^\s]+/);
  next;
  }
#Resolving statistics terms such as percent total, mean, cv etc.
 if($statcount && /^(\w{3})\-*[\s\,]+[\d\.]+[\s\,]+[\d\.]+/)
  {$paramindex++;
  $paramname=$1;
  s/\,/ /;
  s/$paramname\-*//;
  $paramname=~tr/a-z/A-Z/;$paramname=~s/\s//g;
  for($statindex=1;$statindex<=$statcount;$statindex++)
   {s/\s+([\d\.]+)//;

${"Sample_".$sample."_Gate_".$gate."_Param_".$paramname}{$loadstat{$sta
tindex}}=$1;
   }

${"Sample_".$sample."_Gate_".$gate}{$paramname}=\%{"Sample_".$sample."_
Gate_".$gate."_Param_".$paramname};
  if($paramindex==$paramnum)
   {${"Sample_".$sample}{$gate}=\%{"Sample_".$sample."_Gate_".$gate};
   $paramnum=100;
   if (! defined $samples{$sample})
    {$samples{$sample}=\%{"Sample_".$sample};
    $newsamplenum++;
    }
   }
  next;
  }
 if($linenum!=1 && (/Multiple/|| eof fd1) && !$paramindex)
```

```
  {${"Sample_".$sample."_Gate_".$gate}{"void"}=1;
  ${"Sample_".$sample}{"void"}=1;
  if($options[0]=~/w/) {printoutput("\nWarning: Sample-$sample:Gate-
$gate have void parameters!");}
#Handling empty sample:gate combination.
  }
 if(/^\#/ || /^Multiple\ Document\ Interface\ for\ Flow\ Cytometry/
||/^WinMDI\ Version/ || /^\w{3}\ \w{3}\ \d{2}\ \d{2}:\d{2}:\d{2}\
\d{4}/ || /^Project:/ || /^Total\ Events/ || /^System:/){next;}
 else {printoutput("Unresolved data at line $linenum: ".$_."\n");}
 }
printoutput("\nRead $linenum lines from $filename including\
$samplegatenum sample:gates\nof\
$newsamplenum new samples\nand\
$newsamplegatenum new sample:gates\nand\
$replacedsamplegatenum modified sample:gates.\n");
close fd1;
}

sub loadvolume()
{$linenum=0;
while(<fd1>)
 {$linenum++;
 chomp;
 if(/^([^\t]+)\s+([\d\.]+)/)
  {$volumesamplename=$1;
  $volume=$2;
  $volumesamplename=~tr/a-z/A-Z/;
  $volumesamplename=~s/\s+//g;
  if(! defined $samples{$volumesamplename})
   {printoutput("$volumesamplename $volume is invalid at this
time!\n");
   next;
   }
  $samples{$volumesamplename}{'VOLUME'}=$volume;
#Volume data should not precede its FCM data loading.
  }
 else
  {if (/^\#/) {next;}
  else{printoutput("Unresolved volume data at line $linenum:
".$_."\n");}
  }
 }
printoutput("Read $linenum lines from $filename.\n");
close fd1;
}

sub loadrate()
{$linenum=0;
while(<fd1>)
 {$linenum++;
 chomp;
 if(/(SAMPLE_.+?_PREY_.+?_PREDATOR_.+?_FROM_\d+_TO_\d+_MIN):([\-
\d\.e]+)/)
  {if(defined $ratestatdata{$1})
   {printoutput("Rate data for $1 ignored.\n");
   next;
```

```
   }
#Replicated rate data discarded.
  else{$ratestatdata{$1}=$2;}
  }
 else
  {if (/^\#/) {next;}
  else{printoutput("Unresolved grazing rate data at line $linenum:
".$_."\n");}
  }
 }
printoutput("Read $linenum lines from $filename.\n");
close fd1;
}

sub data()
{while(1)
 {printoutput("\nSelect again:\
A: alias sample/gate;\
B: browse/modify data;\
D: delete data;\
L: list sample names;\
S: save modified data;\
U: undo modifications;\
Q: quit to upper level.\n:");
 $select=<>;
 while($select=~/\!/)
  {print "You can list existent sample names, choose a sample to list
available gates, then browse or modify specific data. You might want to
alias sample patterns for easy identification, and gate names to
default gates such as \"prey\" or \"predator\". You can also restore
any changes with the 'U' option to selected data before quitting GR or
further modifying them.\nYour choice:";
  $select=<>;
  }
#Context-sensitive help loop.
 print fdlog $select;
 if($macro){print fdmacro $select;}
 chomp $select;$select=~tr/a-z/A-Z/;
 printoutput("\n");
 if($select eq 'L')
  {printoutput("Sample name pattern:");
  $samplepattern=<>;
  while($samplepattern=~/\!/)
   {&help1();
   print "Sample name pattern:";
   $samplepattern=<>;
   }
  print fdlog $samplepattern;
  if($macro){print fdmacro $samplepattern;}
  chomp $samplepattern;
  printoutput("\n");
#User input should be free of white spaces.
  if($samplepattern eq ''){next;}
  foreach $samplename (sort keys %samples)
   {if($samplename=~/$samplepattern/i)
    {print $samplename."\t";
    print fdlog $samplename."\n";
```

```
    }
#Two different formats for display and storage output. GR tries to
contain display to one screen and facilitate block copy of sample names
from grlog.
   }
  printoutput("\n");
  }
 if($select eq 'A') {&aliasing();}
 if($select eq 'B') {&browsemodifyitem();}
 if($select eq 'D') {&delete();}
 if($select eq 'S') {&save();}
 if($select eq 'U') {&undo();}
 if($select eq 'Q') {return;}
 }
}

sub browsemodifyitem()
{&samplegatepattern();
if($sgmatch==0) {return;}
if($gatepattern!~/VOL/i)
 {printoutput("Parameter name: (FSC,SSC,FL1,FL2,FL3,Tim):");
 $param=<>;
 while($param=~/\!/ || $param ne "\n" && $param!~/\w{3}/)
  {print "Parameter name must be an exact full match. Press enter to
abort.\n";
  print "Parameter name: (FSC,SSC,FL1,FL2,FL3,Tim):";
  $param=<>;
  }
 print fdlog $param;
 if($macro){print fdmacro $param;}
 chomp $param;$param=~tr/a-z/A-Z/;$param=~s/\s//g;
 printoutput("\n");
 if($param eq '') {return;}
 printoutput("Parameter statistics
(events,percenttotal,percentgated,median,mean,cv,peak,value):");
 $stat=<>;
 while($stat=~/\!/ || $stat ne "\n" && $stat!~/\w+/)
  {print "Statistics term name must be an exact full match. If you
specified mutiple mean data, you can perform statistics analysis on
them. Press enter to abort.\n";
  print "Parameter statistics
(events,percenttotal,percentgated,median,mean,cv,peak,value):";
  $stat=<>;
  }
 if($macro){print fdmacro $stat;}
 print fdlog $stat;
 chomp $stat;
 printoutput("\n");
 $stat=~tr/a-z/A-Z/;$stat=~s/\s//g;
 if($stat eq '') {return;}
 foreach $samplename (sort keys %samples)
  {if($samplename=~/$samplepattern/i)
   {foreach $gatename (sort keys %{$samples{$samplename}})
    {if ($gatename=~/$gatepattern/i && defined
$samples{$samplename}{$gatename}{$param}{$stat} )
```

```
{printoutput("$samplename:$gatename:$param:$stat=$samples{$samplename}{
$gatename}{$param}{$stat}\n");
     }
    }
   }
  }
 }
else
 {foreach $samplename (keys %samples)
  {if($samplename=~/$samplepattern/i)
   {if (! defined $samples{$samplename}{$gatepattern})
    {printoutput("$samplename:$gatepattern is not present.\n");
    return;
    }
   else
{printoutput("$samplename:Volume=$samples{$samplename}{'VOLUME'}\n");}
   }
  }
 }
if($sgmatch==1)
 {printoutput("Enter a new value if desired:");
 $newvalue=<>;
 while($newvalue=~/\!/ || $newvalue ne "\n" && $newvalue!~/[\d\.]+/)
  {print "This value can only contain digits or a decimal dot. Press
enter to skip.\n";
  print "Enter a new value if desired:";
  $newvalue=<>;
  }
 if($macro){print fdmacro $newvalue;}
 print fdlog $newvalue;
 chomp $newvalue;
 printoutput("\n");
 if ($newvalue=~/[^\d\.]/ || $newvalue eq '')
  {printoutput("Original status retained\n");
  return;
  }
 else
  {if($gatepattern!~/VOL/i)

{$modified{'Sample_'.$samplenametemp.'_Gate_'.$gatenametemp.'_Param_'.$
param.'_Stat_'.$stat}=$samples{$samplenametemp}{$gatenametemp}{$param}{
$stat};
   $samples{$samplenametemp}{$gatenametemp}{$param}{$stat}=$newvalue;
   }
  else

{$modified{'Sample_'.$samplenametemp.'_Gate_VOLUME'}=$samples{$samplena
metemp}{'VOLUME'};
   $samples{$samplenametemp}{'VOLUME'}=$newvalue;
   }
  }
 }
#Modification is limited to one datum per operation.
}

sub aliasing()
```

```
{&samplegatepattern();
if($smatch==0){return;}
printoutput("Sample name pattern aliased as:");
$aliassamplepattern=<>;
while($aliassamplepattern=~/\!/)
 {&help1();
 print "Sample name pattern aliased as:";
 $aliassamplepattern=<>;
 }
print fdlog $aliassamplepattern;
if($macro){print fdmacro $aliassamplepattern;}
chomp $aliassamplepattern;
printoutput("\n");
$aliassamplepattern=~tr/a-z/A-Z/;
$aliassamplepattern=~s/\s+//;
if($gatepattern ne '')
 {printoutput("Gate pattern aliased as:");
 $aliasgatepattern=<>;
 while($aliasgatepattern=~/\!/)
 {&help2();
 print "Gate pattern aliased as:";
 $aliasgatepattern=<>;
 }
 print fdlog $aliasgatepattern;
 if($macro){print fdmacro $aliasgatepattern;}
 chomp $aliasgatepattern;
 printoutput("\n");
 $aliasgatepattern=~tr/a-z/A-Z/;$aliasgatepattern=~s/\s+//;
 }
#If gatepattern input is omitted, the user must want to alias certain
sample names only.
foreach $samplename (keys %samples)
 {if($samplename=~/$samplepattern/i)
  {$samplenametemp=$samplename;
  if($aliassamplepattern ne '')
   {if($aliassamplepattern=~/\$1/)
    {$patterntemp1=$`;
    $patterntemp2=$';
    $samplenametemp=~s/$samplepattern/$patterntemp1$1$patterntemp2/ig;
    }
#Perl cannot properly handle "$1" within a variable. May extend to
other RegExp predefinitions.
   else {$samplenametemp=~s/$samplepattern/$aliassamplepattern/ig;}
   if($select2 ne 'A')
    {printoutput("Alias $samplename as $samplenametemp ?(y for yes, a
for
all, any other key for no):");
    $select2=<>;
    if($macro){print fdmacro $select2;}
    print fdlog $select2;
    chomp $select2;$select2=~tr/a-z/A-Z/;
    printoutput("\n");
    }
   if($select2 eq 'Y' || $select2 eq 'A')
    {if(defined $samples{$samplenametemp})
     {printoutput("This sample name $samplenametemp already exist!\n");
     next;
```

```
      }
     $modified{'Sample_'.$samplenametemp}='NULL';
#Save modification information for possible recovery.
     $samples{$samplenametemp}=$samples{$samplename};
     }
    }
   if($gatepattern ne '' && $aliasgatepattern ne '')
    {foreach $gatename (keys %{$samples{$samplenametemp}})
     {if($gatename=~/$gatepattern/i)
      {$gatenametemp=$gatename;
      $gatetnametemp=~s/$gatepattern/$aliasgatepattern/ig;
#Gate pattern input must be RegExp-escaped!
      if($select2 ne 'A')
       {printoutput("Alias $samplenametemp:$gatename as
$samplenametemp:$gatenametemp ?(y for yes, a for all, any other key for
no):");
       $select2=<>;
       if($macro){print fdmacro $select2;}
       print fdlog $select2;
       chomp $select2;$select2=~tr/a-z/A-Z/;
       printoutput("\n");
       }
      if($select2 eq 'Y' || $select2 eq 'A')
       {if(defined $samples{$samplenametemp}{$gatenametemp})
        {printoutput("This sample gate $samplenametemp:$gatenametemp
already exist!\n");
        next;
        }

$modified{'Sample_'.$samplenametemp.'_Gate_'.$gatenametemp}='NULL';

$samples{$samplenametemp}{$gatenaametemp}=$samples{$samplenametemp}{$ga
tename};
       }
      }
     }
    }
   }
  }
undef $select2;
}

sub delete()
{&samplegatepattern();
if($samplepattern eq ''||$smatch==0){return;}
foreach $samplename (keys %samples)
 {if($samplename=~/$samplepattern/i)
  {if($gatepattern eq '')
   {if($select3 ne 'A' && $gatepattern eq '')
    {printoutput("Delete sample $samplename?(y for yes, a for all, any
other key for no):");
    $select3=<>;
    if($macro){print fdmacro $select3;}
    print fdlog $select3;
    chomp $select3;$select3=~tr/a-z/A-Z/;
    printoutput("\n");
    }
```

```
    if($select3 eq 'Y' || $select3 eq 'A')
     {$modified{'Sample_'.$samplename}=$samples{$samplename};
     delete $samples{$samplename};
     }
    }
    else
    {foreach $gatename (keys %{$samples{$samplename}})
     {if($gatename=~/$gatepattern/i)
      {if($select3 ne 'A')
       {printoutput("Delete $samplename:$gatename?(y for yes, a for all,
any other key for no):");
       $select3=<>;
       if($macro){print fdmacro $select3;}
       print fdlog $select3;
       chomp $select3;$select3=~tr/a-z/A-Z/;
       printoutput("\n");
       }
      if($select3 eq 'Y' || $select3 eq 'A')

{$modified{'Sample_'.$samplename."_Gate_".$gatename}=$samples{$samplena
me}{$gatename};
       delete $samples{$samplename}{$gatename};
       }
      }
     }
    }
   }
  }
undef $select3;
}

sub undo()
{if(! defined %modified) {return;}
printoutput("Are you sure you want to undo the following changes?\n");
foreach $modifieditem (keys %modified)
 {if($select4 ne 'A')
  {$modifiedtemp=$modifieditem;
  $modifiedtemp=~s/_Gate_|_Param_|_Stat_/\:/g;
  $modifiedtemp=~s/Sample_//;
  printoutput("Restore $modifiedtemp=$modified{$modifieditem} ? (y for
yes, a for all, any other key for no):");
  $select4=<>;
  if($macro){print fdmacro $select4;}
  print fdlog $select4;
  chomp $select4;$select4=~tr/a-z/A-Z/;
  printoutput("\n");
  }
 if($select4 eq 'Y' || $select4 eq 'A')
  {$restorevalue=$modified{$modifieditem};
  if($modifieditem=~/(.+?)_Gate_(.+?)_Param_(.+?)_Stat_(.+)/)
   {$restoretemp=$samples{$1}{$2}{$3}{$4};
   $samples{$1}{$2}{$3}{$4}=$restorevalue;
   if($restorevalue eq 'NULL') {delete $samples{$1}{$2}{$3}{$4};}
   }
  else
   {if($modifieditem=~/(.+?)_Gate_(.+)/)
    {$restoretemp=$samples{$1}{$2};
```

```
    $samples{$1}{$2}=$restorevalue;
    if($restorevalue eq 'NULL') {delete $samples{$1}{$2};}
    }
   else
    {$restoretemp=$samples{$modifieditem};
    $samples{$modifieditem}=$restorevalue;
    if($restorevalue eq 'NULL') {delete $samples{$modifieditem};}
    }
   }
  $modified{$modifieditem}=$restoretemp;
  }
 }
}
#Recovery is carried out according to different hierarchy of involved
data items.

sub samplegatepattern()
{printoutput("Sample name pattern:");
$samplepattern=<>;
while($samplepattern=~/\!/)
 {&help1();
 print "Sample name pattern:";
 $samplepattern=<>;
 }
if($macro){print fdmacro $samplepattern;}
print fdlog $samplepattern;
chomp $samplepattern;
printoutput("\n");
if($samplepattern eq '') {return;}
$smatch=0;$sgmatch=0;
for $samplename (keys %samples)
 {if ($samplename=~/$samplepattern/i)
  {if ($options[0]=~/w/i) {printoutput("Matching $samplename\t");}
  $smatch++;$samplenametemp=$samplename;
  }
 }
if($smatch==0) {printoutput("Unmatched sample pattern!\n");return;}
if($smatch==1)
 {printoutput("Gate pattern(");
 foreach $gatename (keys %{$samples{$samplenametemp}})
  {if ($gatename ne "void" && $gatename ne 'VOLUME')
   {printoutput($gatename." ");
   }
  }
 printoutput(")");
 if(defined $samples{$samplenametemp}{'VOLUME'}){printoutput(" or 'vol'
for volume data:");}
 else {print ":";print fdlog ":";}
 }
#List gates if only one sample name matches.
if($smatch>1){printoutput("Gate pattern('vol' for volume data if
applicable):");}
$gatepattern=<>;
while($gatepattern=~/\!/)
 {&help2();
```

```
 print "If your sample and gate names match only one present
combination, you may be given the choice to alter it. However GR will
not allow massive modification of loaded data at one time.\n";
 print "Gate pattern :";
 $gatepattern=<>;
 }
if($macro){print fdmacro $gatepattern;}
print fdlog $gatepattern;
chomp $gatepattern;
printoutput("\n");
if($gatepattern=~/VOL/i) {$gatepattern='VOLUME';}
foreach $samplename (sort keys %samples)
 {if($samplename=~/$samplepattern/i)
  {foreach $gatename (sort keys %{$samples{$samplename}})
   {if ($gatename=~/$gatepattern/i && $gatename ne 'VOLUME' ||
$gatename eq 'VOLUME' && $gatepattern eq 'VOLUME')
    {if($options[0]=~/w/i)
     {print "Matching $samplename:$gatename\t";
     print "Matching $samplename:$gatename\t";
     }
    $sgmatch++;$gatenametemp=$gatename;
    }
   }
  }
 }
}

sub getparam()
{&samplegatepattern();
if($sgmatch<2||$gatepattern=~/VOL/i)
 {printoutput("Designated patterns must match at least 2
sample/gates!\n");
 return;
 }
printoutput("Parameter name: (FSC,SSC,FL1,FL2,FL3,Tim):");
$param=<>;
while($param=~/\!/ || $param ne "\n" && $param!~/\w{3}/)
 {print "Parameter name must be an exact full match. Press enter to
abort.\n";
 print "Parameter name: (FSC,SSC,FL1,FL2,FL3,Tim):";
 $param=<>;
 }
print fdlog $param;
if($macro){print fdmacro $param;}
chomp $param;$param=~tr/a-z/A-Z/;$param=~s/\s//g;
printoutput("\n");
if($param eq '') {$sgmatch=0;}
}

sub anal()
{while(1)
 {printoutput("\nSelect again:\
M: detector signal comparison;\
T: time kinetic analysis;\
Q: quit to upper level.\n:");
 $select=<>;
 while($select=~/\!/)
```

```
  {print "Choose M to compare parameter readouts statisticly between
certain sample/gate. Choose T to calculate changes between two time
points in the same sample-time series by subtration. Both are based on
mean, cv and events readout for different sample/gate
combinations.\n:";
  $select=<>;
  }
 print fdlog $select;
 if($macro){print fdmacro $select;}
 chomp $select;$select=~tr/a-z/A-Z/;
 printoutput("\n");
 if($select eq 'M') {&meancomparison();}
 if($select eq 'T') {&timekinetics();}
 if($select eq 'Q') {return;}
 }
}


sub meancomparison()
{&getparam();
if($sgmatch<2||$gatepattern=~/VOL/i) {return;}
undef %statdata;
$stat='MEAN';
foreach $samplename (sort keys %samples)
 {if($samplename=~/$samplepattern/i)
  {foreach $gatename (sort keys %{$samples{$samplename}})
   {if ($gatename=~/$gatepattern/i && defined
$samples{$samplename}{$gatename}{$param}{$stat} )

{printoutput("$samplename:$gatename:$param:$stat=$samples{$samplename}{
$gatename}{$param}{$stat}\n");

$statdata{"$samplename:$gatename:$param:$stat"}=$samples{$samplename}{$
gatename}{$param}{$stat};
    }
   }
  }
 }
&ave();
printoutput("\nAverage+/-S.D.: %-6.2f+/-%-6.2f\n",$ave,$sd);
printoutput("\nIndividual T-test:\n");
&ttest();
printoutput("\nANOVA: ");
&vtest();
%comparestatdata=%statdata;
&comp();
undef %statdata;undef %comparestatdata;
}

sub timekinetics()
{&getparam();
if($sgmatch<2||$gatepattern=~/VOL/i){return;}
$tkparam=$param;
printoutput("Timepoints (<t1> <t2> ...):");
$tp=<>;
while($tp!~/\d+\s+\d+/||$tp=~/\!/)
```

```
 {print "Use the numbers in the sample name time labels.\nTimepoints
(<t1> <t2> ...):";
 $tp=<>;
 }
print fdlog $tp;
if($macro){print fdmacro $tp;}
chomp $tp;
printoutput("\n");
@tp=split(/\ /,$tp);
printoutput("Time\t");
for($i=0;$i<=$#tp;$i++) {print "$tp[$i]\t\t";}
printoutput("\n");
undef %tk;undef @tkstartsample;undef @tkmean; undef @tkinterval;
foreach $samplename (keys %samples)
 {if($samplename=~/$samplepattern/i)
  {$samplenametemp=$samplename;
  $samplenametemp=~s/T=(\d+)MIN//;$tptemp=$1;
  for($i=0;$i<=$#tp&&$tp[$i]!=$tptemp;$i++){};
  if($i>$#tp) {next;}
  if(defined $tk{$samplenametemp}) {next;}
  $tksamplename=$samplenametemp;
  if($samplenametemp=~/.(.{7})$/){$tksamplename=$1;}
  $tkgatematch=0;
  foreach $gatename (keys %{$samples{$samplename}})
   {if($gatename=~/$gatepattern/i)
    {$tkgatematch++;
    printoutput("\n$tksamplename\t");
    for($i=0;$i<=$#tp;$i++)
     {$tksample[$i]=$samplenametemp."T=".$tp[$i]."MIN";
     if(! defined $samples{$tksample[$i]}{$gatename}{$tkparam}{'MEAN'})
      {printoutput("N/A\t\t");next;
      }
     $tkmean=$samples{$tksample[$i]}{$gatename}{$tkparam}{'MEAN'};
     $tkmean[$i].="$tkmean ";
     printoutput("$tkmean\t\t");
     }
    printoutput("\nEvents\t");
    for($i=0;$i<=$#tp;$i++)
     {if(! defined
$samples{$tksample[$i]}{$gatename}{$tkparam}{'EVENTS'})
      {printoutput("N/A\t\t");next;
      }

printoutput("$samples{$tksample[$i]}{$gatename}{$tkparam}{'EVENTS'}\t\t
");
     }
    printoutput("\nInterval\t");
    for($i=0;$i<$#tp;$i++)
     {$tkinterval="NULL";
     if(! defined
$samples{$tksample[$i]}{$gatename}{$tkparam}{'EVENTS'} ||! defined
$samples{$tksample[$i+1]}{$gatename}{$tkparam}{'EVENTS'})
      {printoutput("%-6s\t\t","N/A");
      next;
      }

if(($samples{$tksample[$i+1]}{$gatename}{$tkparam}{'EVENTS'}/$samples{$
```

```
tksample[$i+1]}{'VOLUME'}-
$samples{$tksample[$i]}{$gatename}{$tkparam}{'EVENTS'}/$samples{$tksamp
le[$i]}{'VOLUME'})<0)

{$tkinterval=(log($samples{$tksample[$i+1]}{$gatename}{$tkparam}{'MEAN'
})*$samples{$tksample[$i+1]}{$gatename}{$tkparam}{'EVENTS'}/$samples{$t
ksample[$i+1]}{'VOLUME'}-
log($samples{$tksample[$i]}{$gatename}{$tkparam}{'MEAN'})*$samples{$tks
ample[$i]}{$gatename}{$tkparam}{'EVENTS'}/$samples{$tksample[$i]}{'VOLU
ME'})/($samples{$tksample[$i+1]}{$gatename}{$tkparam}{'EVENTS'}/$sample
s{$tksample[$i+1]}{'VOLUME'}-
$samples{$tksample[$i]}{$gatename}{$tkparam}{'EVENTS'}/$samples{$tksamp
le[$i]}{'VOLUME'});
      $tkinterval=exp($tkinterval);

$tkintervalratio=($samples{$tksample[$i+1]}{$gatename}{$tkparam}{'MEAN'
}+$samples{$tksample[$i]}{$gatename}{$tkparam}{'MEAN'})/2/$tkinterval;
      if($tkintervalratio<0.5 ||
$tkintervalratio>5){$tkinterval="NULL";}
#artificial thresholds for outliers
      else {printoutput("%6.4g",$tkinterval);}
      }
      printoutput("\t\t");
      $tkinterval[$i].="$tkinterval ";
     }
    printoutput("\n");
    }
   }
  if($tkgatematch){$tk{$samplenametemp}++;}
  }
 }
printoutput("\nAverage for
$samplepattern:$gatepattern:$tkparam\nTime\t");
for($i=0;$i<=$#tp;$i++) {print "$tp[$i]\t\t";}
printoutput("\nMean\t");
for($i=0;$i<=$#tp;$i++)
 {undef %statdata;undef %tkmean;
 @tktemp=split(/\ /,$tkmean[$i]);
 for($j=0;$j<=$#tktemp;$j++)
  {$tkmean{$j}=$tktemp[$j];
  }
 %statdata=%tkmean;
 &ave();
 printoutput("%-6.4g\t\t",$ave);
 }
printoutput("\n+/-S.D.\t");
for($i=0;$i<=$#tp;$i++)
 {undef %statdata;undef %tkmean;
 @tktemp=split(/\ /,$tkmean[$i]);
 for($j=0;$j<=$#tktemp;$j++)
  {$tkmean{$j}=$tktemp[$j];
  }
 %statdata=%tkmean;
 &ave();
 printoutput("%-4.2g\t\t",$sd);
 }
printoutput("\nInterval\t");
```

```
for($i=0;$i<$#tp;$i++)
 {undef %statdata;undef %tkinterval;
 @tktemp=split(/\ /,$tkinterval[$i]);
 for($j=0;$j<=$#tktemp;$j++){if($tktemp[$j] ne 'NULL' )
{$tkinterval{$j}=$tktemp[$j];}}
 %statdata=%tkinterval;
 &ave();
 printoutput("%-6.4g\t\t",$ave);
 }
printoutput("\n+/-S.D.\t\t");
for($i=0;$i<$#tp;$i++)
 {undef %statdata;undef %tkinterval;
 @tktemp=split(/\ /,$tkinterval[$i]);
 for($j=0;$j<=$#tktemp;$j++){if($tktemp[$j] ne 'NULL' )
{$tkinterval{$j}=$tktemp[$j];}}
 %statdata=%tkinterval;
 &ave();
 printoutput("%-4.2g\t\t",$sd);
 }
printoutput("\n");
}

sub save()
{printoutput("Save to data file (.grd):");
$filename=<>;
while($filename=~/\!/)
  {print "You may omit the .grd extension. In Unix you can also skip
the file name and save it to a hidden file.\n";
  print "Save to data file (.grd):";
  $filename=<>;
  }
if($macro){print fdmacro $filename;}
print fdlog $filename;
chomp $filename;
printoutput("\n");
if($filename!~/\.grd/i) {$filename.=".grd";}
if(!open(fdl,">$filename"))
 {printoutput("Unable to write to $filename!\n");
 return;
 }
undef %entries; undef @entriesnt;
foreach $samplename (keys %samples)
 {if(defined $samples{$samplename})
  {if($samplename=~/T=\d+MIN$/)
   {$samplename=~s/T=(\d+)MIN$//;
   $entries{$samplename}.="$1 ";
   }
  else {push(@entriesnt,$samplename);}
  }
 }
undef @orderlist;
foreach $entryname (sort keys %entries)
 {@entrytime=sort {$a<=>$b} split(/\ /,$entries{$entryname});
foreach $entrytp (@entrytime)
  {push(@orderlist,$entryname."T=".$entrytp."MIN");}
 }
push(@orderlist,@entriesnt);
```

```
#Make ordered sample name list to facilite future examination of GR
data files.
foreach $sampleentry (@orderlist)
 {if ($sampleentry eq 'void') {next;}
 foreach $gateentry (keys %{$samples{$sampleentry}})
  {if ($gateentry eq 'void' || ! defined
$samples{$sampleentry}{$gateentry}) {next;}
  if($gateentry!~/VOL/i)
   {print fd1 "Gates: $gateentry\n";
   }
  else
   {print fd1 "Volume of Sample $sampleentry
:$samples{$sampleentry}{'VOLUME'}\n";
   next;
   }
#Corresponding to statements in load()
  print fd1 "Sample: $sampleentry\n";
  $paramnum=0;
  foreach (keys %{$samples{$sampleentry}{$gateentry}}) {$paramnum++;}
  print fd1 "Parameters: $paramnum\n";
  print fd1 "Param name \t";
  foreach $statentry (keys
%{$samples{$sampleentry}{$gateentry}{'FSC'}})
   {print fd1 $statentry."\t";
   }
  print fd1 "\n";
  foreach $paramentry (sort keys %{$samples{$sampleentry}{$gateentry}})
   {print fd1 $paramentry."-\t\t";
   foreach $statentry (keys
%{$samples{$sampleentry}{$gateentry}{$paramentry}})
    {print fd1
$samples{$sampleentry}{$gateentry}{$paramentry}{$statentry}."\t";
    }
   print fd1 "\n";
   }
  }
 }
}

sub calc()
{foreach $flask ('','PREY_CONTROL')
 {printoutput("Number of $flask replicates:");
 $replicatetemp=<>;
 while($replicatetemp=~/\!/ || $replicatetemp ne "\n" && $replicatetemp
!~/\d+/)
   {print "Please enter an integer. Or press enter to abort:";
   $replicatetemp=<>;
   }
 if($macro){print fdmacro $replicatetemp;}
 print fdlog $replicatetemp;
 chomp $replicatetemp;
 printoutput("\n");
 ${$flask.'replicate'}=$replicatetemp;
#Control replicate input can be omitted for uncontrolled grazing
experiment
 if($replicatetemp ne '' && $replicatetemp !=0 ||$flask eq '')
  {
```

```
there: for($i=1;$i<=$replicatetemp;$i++)
   {if(! $flask) {printoutput("Replicate $i sample name pattern:");}
   else{printoutput("$flask Replicate $i sample name pattern:");}
   $samplepattern=<>;
   while($samplepattern=~/\!/)
    {&help1();print"Make sure this pattern matches only one sample name
deprived of time label.\n";
    print "$flask Replicate $i sample name pattern:";
    $samplepattern=<>;
    }
   if($macro){print fdmacro $samplepattern;}
   print fdlog $samplepattern;
   chomp $samplepattern;
   printoutput("\n");
   if($samplepattern eq '') {&myundef(); return;}
   $samplenametemp='';
   foreach $samplename (keys %samples)
    {if($samplename=~/$samplepattern/i)
     {if(! defined $samples{$samplename}{'VOLUME'})
      {print "Volume data for $samplename is absent!\n";
      &myundef();return;
      }
#Volume data is necessary to calculate densities.
     $samplename=~s/T=(\d+)MIN$//;
     $time=$1;
     if($samplenametemp eq '') {$samplenametemp=$samplename;}
     if($samplename!~/$samplenametemp/)
      {printoutput("$samplename and $samplenametemp share this pattern,
please be more specific!\n");
      &myundef();$i=0;
      next there;
      }
#Examination for replicate sample name redundancy.
     else
      {split(/\ /, ${$flask.'sampletimeseries'}{$samplenametemp});
      for($_=0;$_<=$#_;$_++) {if($_[$_]==$time){next there;}}
      ${$flask.'timesampleaggregate'}{$time}++;
      ${$flask.'sampletimeseries'}{$samplenametemp}.="$time ";
      }
     }
    }
    if($samplenametemp eq '')
    {printoutput("Unmatched $flask sample pattern, please try
again!\n");
    &myundef();$i=0;
    next there;
    }
   }
  foreach $samplename (keys %{$flask.'sampletimeseries'})
   {split(/\ /,${$flask.'sampletimeseries'}{$samplename});
   ${$flask.'sampletimeseries'}{$samplename}=join(' ', sort {$a<=>$b}
@_);
   ${$flask.'sampletimeseries'}{$samplename}='
'.${$flask.'sampletimeseries'}{$samplename}.' ';
   }
  printoutput("Available $flask time points:");
```

```
  @{$flask.'timepoint'}=sort{$a<=>$b} keys
%{$flask.'timesampleaggregate'};
  foreach $tp (@{$flask.'timepoint'}){printoutput($tp."Min ");}
  printoutput("\n");
  do
   {printoutput("Pick $flask time range (<t1>-<t2>, press enter for
all):");
   $timerange=<>;
   while($timerange=~/\!/ || $timerange ne "\n" && $timerange!~/\d+\s*-
\s*\d+/)
    {print "Please give two time values connected by a hyphen. Grazing
rates calculation will be between these two time points only. Press
enter to keep all time points.\n";
    print "Pick $flask time range (<t1>-<t2>, press enter for all):";
    $timerange=<>;
    }
   if($macro){print fdmacro $timerange;}
   print fdlog $timerange;
   chomp $timerange;
   printoutput("\n");
   if($timerange ne '')
    {if($timerange=~/(\d+)\s*-\s*(\d+)/)
     {@timepointtemp=sort{$a<=>$b}($1,$2);
     foreach $samplename (keys %{$flask.'sampletimeseries'})

{if(${$flask.'sampletimeseries'}{$samplename}!~/\s$timepointtemp[0]\s/|
|${$flask.'sampletimeseries'}{$samplename}!~/\s$timepointtemp[1]\s/)
       {printoutput("$samplename does not fit this range!\n");
       $timerange='NULL';
       }
      }
     }
    else {$timerange='NULL';}
    }
   }while($timerange eq "NULL");
#If valid, ONLY those two selected time points are used in
calculations. Invalid input causes maintenance of all available time
points.
  if($timerange ne '')
   {@{$flask.'timepoint'}=@timepointtemp;
   foreach $samplename (keys %{$flask.'sampletimeseries'})
    {${$flask.'sampletimeseries'}{$samplename}=" $timepoint[0]
$timepoint[1] ";
    }
   }
  }
 }
printoutput("Model (l for linear or any other key for exponential):");
$model=<>;
while($model=~/\!/)
 {print"Linear model assumes a constant predator ingestion speed while
exponential model proportionate it to the density of available prey.
The rate units are \"number of prey/(grazer*min)\" and
\"ml/(grazer*min)\", respectively. Exponential model is the
default.\n";
 print "\nModel (l for linear or any other key for exponential):";
 $model=<>;
```

```
 }
if($macro){print fdmacro $model;}
print fdlog $model;
$model=~tr/a-z/A-Z/;chomp $model;
printoutput("\nPredator Gate:");
$predatorgate=<>;
while($predatorgate=~/\!/)
 {print"Predator gate should be an exact full match. Press enter if you
have already aliased a \"predator\" gate. Perl RegExp control must not
be escaped.\n";
 print "\nPredator Gate:";
 $predatorgate=<>;
 }
if($macro){print fdmacro $predatorgate;}
print fdlog $predatorgate;
chomp $predatorgate;
$predatorgate=~tr/a-z/A-Z/;
if($predatorgate eq '') {$predatorgate='PREDATOR';}
printoutput("\nNumber of prey species:");
$preynumber=<>;
while($preynumber=~/\!/ || $preynumber ne "\n" && $preynumber!~/\d+/)
 {print"Please enter a integer for the number of grazing rate tables to
be generated. Any invalid input will default it to one:";
 $preynumber=<>;
 }
if($macro){print fdmacro $preynumber;}
print fdlog $preynumber;
if($preynumber<=0){$preynumber=1;}
for($preycount=1;$preycount<=$preynumber;$preycount++)
 {printoutput("\nPrey $preycount Gate:");
 $preygate=<>;
 while($preygate=~/\!/)
  {print"Prey gate should be an exact full match. Press enter if you
have already aliased a \"prey\"(\"prey2\"...). Perl RegExp control must
not be escaped.\n";
  print "\nPrey Gate:";
  $preygate=<>;
  }
 if($macro){print fdmacro $preygate;}
 print fdlog $preygate;
 chomp $preygate;
 $preygate=~tr/a-z/A-Z/;
 printoutput("\n");
 if($preygate eq '')
{$preygate=($preycount==1?'PREY':'PREY'.$preycount);}
 $preygate[$preycount]=$preygate;
 }
@allsample=sort keys %sampletimeseries;
undef %statdata;
for($preycount=1;$preycount<=$preynumber;$preycount++)
 {$preygate=$preygate[$preycount];
 printoutput("\nPreygate $preygate\n\n\t\t");
 foreach $samplename (@allsample)
  {$samplenametemp=$samplename;
  if($samplename=~/.(.{7})$/){$samplenametemp=$1;}
#Peculiar things will happen if use $samplename=$1 directly
  printoutput("%-7s\t",$samplenametemp);
```

```
   }
  printoutput("SAMPLE_AVE+/-S.D.\n\n");
 #Print output headline
 %sampletimeseriestemp=%sampletimeseries;
 #%sampletimeseriestemp will be altered during calculation,
 sampletimeseries is used to maintain the initial values.
 for($i=0;$i<$#timepoint;$i++)
   {if($PREY_CONTROLreplicate>0)

 {$controlgr=0;$controlnum=0;$controlpredator=0;$controlpredatornum=0;
    foreach $controlsample (keys %PREY_CONTROLsampletimeseries)
     {$controlnum++;
     $splittemp=$PREY_CONTROLsampletimeseries{$controlsample};
     $splittemp=~s/^ //;
     @controltp=sort{$a<=>$b}split(/\ /,$splittemp);
     for($j=0;$j<=$#controltp && $controltp[$j]<=$timepoint[$i];$j++){};
     if($j>0) {$j--;}

$controlstartdensity=$samples{$controlsample."T=".$controltp[$j]."MIN"}
{$preygate}{'FSC'}{'EVENTS'}/$samples{$controlsample."T=".$controltp[$j
]."MIN"}{'VOLUME'};
     for($j2=$#controltp; $j2>=0 &&
$controltp[$j2]>=$timepoint[$i+1];$j2--){};
     if($j2<$#controltp) {$j2++;}

$controlenddensity=$samples{$controlsample."T=".$controltp[$j2]."MIN"}{
$preygate}{'FSC'}{'EVENTS'}/$samples{$controlsample."T=".$controltp[$j2
]."MIN"}{'VOLUME'};
     if($j!=$j2)
      {$controlgr+=($model eq 'L')?($controlstartdensity-
$controlenddensity)/($controltp[$j2]-
$controltp[$j]):log($controlstartdensity/$controlenddensity)/($controlt
p[$j2]-$controltp[$j]);
      }
     else{$controlgr+=0;}
     if(defined
$samples{$controlsample."T=".$controltp[$j2]."MIN"}{$predatorgate}{'FSC
'}{'EVENTS'})

{$controlpredator+=$samples{$controlsample."T=".$controltp[$j2]."MIN"}{
$predatorgate}{'FSC'}{'EVENTS'}/$samples{$controlsample."T=".$controltp
[$j2]."MIN"}{'VOLUME'};
      $controlpredatornum++;
      }
     if(defined
$samples{$controlsample."T=".$controltp[$j]."MIN"}{$predatorgate}{'FSC'
}{'EVENTS'})

{$controlpredator+=$samples{$controlsample."T=".$controltp[$j]."MIN"}{$
predatorgate}{'FSC'}{'EVENTS'}/$samples{$controlsample."T=".$controltp[
$j]."MIN"}{'VOLUME'};
      $controlpredatornum++;
      }
    }
   }
  else{$controlgr=0;}
```

```
#Calculate control caliberation rates. If control samples do not exist
for the current experiment time points, the encompassing control time
points are used.
  if($PREY_CONTROLreplicate!=0) {$controlgr/=$controlnum;}
  if($controlpredatornum!=0) {$controlpredator/=$controlpredatornum;}
  printoutput($timepoint[$i]."--".$timepoint[$i+1]."Min\t");
  foreach $calcsample (@allsample)
   {do
    {$sampletimeseriestemp{$calcsample}=~/^\ (\d+)\ (\d+)/;
    $t1=$1;$t2=$2;
    if($t1<=$timepoint[$i] && $t2==$timepoint[$i+1])
     {$sampletimeseriestemp{$calcsample}=~s/\ $t1//;
     $startsample=$calcsample."T=".$t1."MIN";
     $endsample=$calcsample."T=".$t2."MIN";

$predatordensity=($samples{$startsample}{$predatorgate}{'FSC'}{'EVENTS'
}/$samples{$startsample}{'VOLUME'}+$samples{$endsample}{$predatorgate}{
'FSC'}{'EVENTS'}/$samples{$endsample}{'VOLUME'})/2;
     $predatordensity-=$controlpredator;
     $timelapse=$t2-$t1;

$startdensity=$samples{$startsample}{$preygate}{'FSC'}{'EVENTS'}/$sampl
es{$startsample}{'VOLUME'};

$enddensity=$samples{$endsample}{$preygate}{'FSC'}{'EVENTS'}/$samples{$
endsample}{'VOLUME'};
     $gr=($model eq 'L')?($startdensity-
$enddensity)/($timelapse*$predatordensity):log($startdensity/$enddensit
y)/($timelapse*$predatordensity);
     $gr-=$controlgr/$predatordensity;
     $gr*=1000000;
#Making grazing rates in the unit of microliter per grazer per minute.
     $samplerate{$t2}.=$gr." ";
     $timerate{$calcsample}.=$gr." ";
     $t2temp=$t2;

$ratestatdata{"SAMPLE_".$calcsample."_PREY_".$preygate."_PREDATOR_".$pr
edatorgate."_FROM_".$t1."_TO_".$t2."_MIN"}=$gr;
     if($preycount==2 && $preynumber==2)

{$preydiff{"SAMPLE_".$calcsample."_FROM_".$t1."_TO_".$t2."_MIN"}=$gr-
$ratestatdata{"SAMPLE_".$calcsample."_PREY_".$preygate[1]."_PREDATOR_".
$predatorgate."_FROM_".$t1."_TO_".$t2."_MIN"}
     }
     printoutput("%-6.3g\t",$gr);
     }
    else
     {if($t1>=$timepoint[$i+1]||$t1==0&&$t2==0)
{printoutput("xxxxxx\t");}
#Experiment time points unavailable.
     else {if($t2>$timepoint[$i+1]) {printoutput(">      \t");} }
#End time point extended to the next time range.
     }

}while($t2<$timepoint[$i+1]&&$sampletimeseriestemp{$calcsample}!~/\d+/)
   }
#Average and SD displayed on different lines for aesthetic purposes.
```

```
  printoutput("--->");
  undef %statdata;undef @statdata;
  @statdata=split(/\ /,$samplerate{$t2temp});
  foreach $statdata (@statdata)
   {$statdata{$statdata}=$statdata;}
  &ave();
  printoutput("%6.3g+/-%-6.3g\n",$ave,$sd);
  }
 printoutput("\nTIME AVERAGE\t");
 undef %statdata;undef @statdata;
 foreach $ratesample (@allsample)
  {@statdata=split(/\ /,$timerate{$ratesample});
  foreach $statdata (@statdata) {$statdata{$statdata}=$statdata;}
  &ave();
  $timesd{$ratesample}=$sd;
  printoutput("%-6.3g\t",$ave);
  undef @statdata;undef %statdata;
  }
 printoutput("--->");
 foreach $ratesample (keys %timerate)
  {push(@statdata,split(/\ /,$timerate{$ratesample}));
  }
 foreach $statdata (@statdata) {$statdata{$statdata}=$statdata;}
 &ave();
 printoutput("%6.3g+/-%-6.3g",$ave,$sd);
 printoutput("\n+/-S.D.\t\t");
 foreach $ratesample (@allsample){printoutput("%-
6.3g\t",$timesd{$ratesample});}
 printoutput("\n\n");
 undef %samplerate;undef %timerate;undef %statdata; undef @statdata;
 }
if($preynumber==2)
 {%statdata=%preydiff;
 &ave();
 $t=$ave/$sdm;
 $df=$count-1;
 &probt();
 printoutput("\n\nProbability of indiscrimination between $preygate[1]
and $preygate[2]: %-6.3g\n",$pt);
 }
do
 {printoutput("Delete rate data?(press enter to skip or specify a data
pattern):");
 $ratepattern=<>;
 while($ratepattern=~/\!/)
  {print("The following rate data exist:\n");
  foreach $ratesample (keys %ratestatdata)
   {printoutput("$ratesample: ");
   printoutput("%-6.3g\n",$ratestatdata{$ratesample});
   }
  print "You may substitute spaces for underscores, but the order of
elements should be kept and Perl RegExp escaped.\n";
  print "Rate data pattern :";
  $ratepattern=<>;
  }
 if($macro){print fdmacro $ratepattern;}
 print fdlog $ratepattern;
```

```
 chomp $ratepattern;
 printoutput("\n");
 $ratepattern=~tr/a-z/A-Z/;
 $ratepattern=~s/\s+/\_/g;
 if($ratepattern ne '')
  {foreach $ratename (keys %ratestatdata)
   {if($ratename=~/$ratepattern/i)
    {printoutput("Delete $ratename=");
    printoutput("%-6.3g",$ratestatdata{$ratename});
    printoutput("? (y for yes or any other key for no):");
    $rateundefselect=<>;
    if($macro){print fdmacro $rateundefselect;}
    print fdlog $rateundefselect;
    chomp $rateundefselect;$rateundefselect=~tr/a-z/A-Z/;
    printoutput("\n");
    if($rateundefselect eq 'Y') {delete $ratestatdata{$ratename};}
    }
   }
  }
 }while($ratepattern ne '');
#Rate data may be deleted to discard outlier.
printoutput("Save to rate file (.grr):");
$filename=<>;
while($filename=~/\!/)
  {print "You may omit the .grr extension. Skip by pressing enter.\n";
  print "Save to rate file (.grr):";
  $filename=<>;
  }
if($macro){print fdmacro $filename;}
print fdlog $filename;
chomp $filename;
printoutput("\n");
if($filename ne '')
 {if($filename!~/\.grr/i) {$filename.=".grr";}
 if(!open(fd1,">>$filename")) {printoutput("Unable to write to
$filename!\n");}
 else
  {foreach $rateentry (sort keys %ratestatdata)
   {print fd1 "$rateentry:";
   printf fd1 "%-6.3g",$ratestatdata{$rateentry};
   print fd1 "\n";
   }
  }
 }
close fd1;
&myundef();
}

sub myundef()
{undef %timesampleaggregate;
undef %PREY_CONTROLsampleaggregate;
undef %sampletimeseries;
undef %PREY_CONTROLsampletimeseries;
undef %samplerate;
undef %timerate;
undef @allsample;
undef %statdata;
```

```
undef @statdata;
undef %sampletimeseriestemp;
}
#Perl global variable lifespan has to be addressed.

sub ave()
{
$count=0;$sd=0;$ave=0;
for $statsample (keys %statdata)
 {$count++;
 $ave+=$statdata{$statsample};
 $sd+=$statdata{$statsample}*$statdata{$statsample};
 }
if($count==0){$ave=-1;$sd=-1;return;}
if($count==1){$sd=0;return;}
$ave/=$count;
$sd-=$count*$ave*$ave;
$sd/=$count-1;
$sd=sqrt($sd);
$sdm=$sd/sqrt($count);
}

sub ttest()
{undef %tested;undef %cv;undef %s;undef %n;undef %x;$sigdiff=0;
for $statsample (keys %statdata)
 {if($statsample=~/([^:]+):([^:]+):([^:]+):([^:]+)/)
  {$x{$statsample}=log10($statdata{$statsample});
  $n{$statsample}=$samples{$1}{$2}{$3}{'EVENTS'};
  $cv{$statsample}=$samples{$1}{$2}{$3}{'CV'};
  $s{$statsample}=$cv{$statsample}/100*$x{$statsample};
  }
 }
#Matching formats for individual parameter comparison
for $statsample (keys %statdata)
 {if($statdata{$statsample}=~/(\d+)\*([\d\.e\-]+)\+([\d\.e\-]+)/)
  {$n{$statsample}=$1;
  $x{$statsample}=$2;
  $s{$statsample}=$3;
  }
 }
#Matching formats for pattern-grouped comparison
foreach $testsample (sort keys %statdata)
 {foreach $testedsample (sort keys %statdata)
  {if($testsample ne $testedsample && ! defined $tested{"$testedsample-
$testsample"})
   {$tested{"$testsample-$testedsample"}=1;
   $df1=$n{$testsample}-1;$df2=$n{$testedsample}-1;
   if($s{$testsample}==0 ||$s{$testedsample}==0){next;}
   else{$f=$s{$testsample}**2/$s{$testedsample}**2;}
   &probf();
   $k=0;
   if($fp>0.025 && $fp<0.975)
    {$df=$n{$testsample}+$n{$testedsample}-2;}
   else

{$k=$s{$testsample}**2/$n{$testsample}/($s{$testsample}**2/$n{$testsamp
le}+$s{$testedsample}**2/$n{$testedsample});
```

```
    $df=floor(1/($k*$k/$n{$testsample}+(1-$k)**2/$n{$testedsample}));
    }
#Determination of equal expected standard deviation between two groups
for choosing degree of freedom.
   if($df1>50&&$df2>50||$k>0)
    {$t=($x{$testsample}-
$x{$testedsample})/sqrt($s{$testsample}**2/$n{$testsample}+$s{$testedsa
mple}**2/$n{$testedsample});
    }
   else
    {$t=($x{$testsample}-
$x{$testedsample})/sqrt(($s{$testsample}**2*$df1+$s{$testedsample}**2*$
df2)/($df1+$df2)*(1/$n{$testsample}+1/$n{$testedsample}));
    }
   if($t<0) {$t=-$t;}
   &probt();
   $siglevel=0.05;
   if($options[0]=~/s([\d\.]+)/) {$siglevel=$1;}
   if($pt<$siglevel)
    {$sigdiff++;
    printoutput("\n%-10s: ",$testsample);
    printoutput("%-6.3g * %-6.3g +/- %-
6.3g",$n{$testsample},$x{$testsample},$s{$testsample});
    printoutput("\tvs.\n%-10s: ",$testedsample);
    printoutput("%-6.3g * %-6.3g +/- %-
6.3g",$n{$testedsample},$x{$testedsample},$s{$testedsample});
    printoutput("\nTP=%-5.2g \n",$pt);
#Perl doesn't allow variables' appearance in format strings
    }
   }
  }
 }
printoutput("\nT-test found no significant difference at 0.05%
level!\n") if !$sigdiff;
}

sub vtest()
{for $statsample (keys %statdata)
 {if($statsample=~/([^:]+):([^:]+):([^:]+):([^:]+)/)
  {$x{$statsample}=log10($statdata{$statsample});
  $n{$statsample}=$samples{$1}{$2}{$3}{'EVENTS'};
  $cv{$statsample}=$samples{$1}{$2}{$3}{'CV'};
  $s{$statsample}=$cv{$statsample}/100*$x{$statsample};
  }
 }
for $statsample (keys %statdata)
 {if($statdata{$statsample}=~/(\d+)\*([\d\.e\-]+)\+([\d\.e\-]+)/)
  {$n{$statsample}=$1;
  $x{$statsample}=$2;
  $s{$statsample}=$3;
  }
 }
$ssa=0;$sse=0;$totalx=0;$totaln=0;$totals=0;$treatment=0;
for $statsample (sort keys %statdata)
 {$treatment++;
 $totalx+=$x{$statsample}*$n{$statsample};
 $totaln+=$n{$statsample};
```

```
 $sse+=($n{$statsample}-1)*$s{$statsample}**2;
 $ssa+=$n{$statsample}*$x{$statsample}**2;
 }
if($treatment==0||$treatment==1){return;}
$totalx/=$totaln;
$ssa-=$totaln*$totalx**2;
$df2=$totaln-$treatment;
$df1=$treatment-1;
$f2=$sse/$df2;
$f1=$ssa/$df1;
$f=$f1/$f2;
&probf();
printoutput("FP=%-10.4g\n",$fp);
}

sub probf()
{$switch=0;
if($df1%2==1 && $df2%2==1)
 {$tempq = $df1*$f/($df1*$f+$df2);
 $tempsa = sqrt($tempq);
 $tempsl = log($tempsa);
 $tempca = sqrt(1-$tempq);
 $tempcl = log(sqrt(1-$tempq));
 $tempal = atan2($tempsa,$tempca);
 $fp=1-2*$tempal/3.1416;
 $tempr=0;
 if ($df2!=1)
  {$tempc=log(2*$tempsa/3.1416);
  $fp-=exp($tempc+$tempcl);
  if($df2!=3)
   {$tempn=floor(($df2-3)/2);
   for($i=1;$i<=$tempn;$i++)
    {$tempx=2*$i+1;
    $tempr+=log(($tempx-1)/$tempx);
    $temprr=$tempr+$tempcl*$tempx+$tempc;
    if ($temprr>-78.4)    {$fp-=exp($temprr);}
    }
   }
  }
 if ($df1!=1)
  {$tempc=$tempr;
  if ($df2>1)  {$tempc+=log($df2-1)}
  $tempc+=log(2/3.1416) + $tempsl + $tempcl*$df2;
  if ($tempc>-78.4) {$fp+=exp($tempc);}
  if ($df1!=3)
   {$tempn =floor(($df1-3)/2);
   $tempr=0;
   for ($i=1;$i<=$tempn;$i++)
    {$tempx=$i*2+1;
    $tempr+=log(($df2+$tempx-2)/$tempx);
    $temprr=$tempr+$tempsl*($tempx-1)+$tempc;
    if ($temprr>-78.4) {$fp+=exp($temprr);}
    }
   }
  }
return;
}
```

```
if($df1%2==1 && $df2%2==0)
 {$f=1/$f;
 $temp=$df1;
 $df1=$df2;
 $df2=$temp;
 $switch=1;
 }
if($df1==0){$df1=1;}
if($df2==0){$df2=1;}
#to avoid log0 error
$tempq = $df1*$f/($df1*$f+$df2);
$tempql=log($tempq);
$fp=0;
$tempc =log(1-$tempq)*$df2/2;
if ($tempc>-78.4) {$fp =exp($tempc);}
if ($df1 != 2)
 {$tempn=floor($df1/2-1);
 $tempr=0;
 for ($i=1;$i<=$tempn;$i++)
  {$tempx=2*$i;
  $tempr+=log($df2+$tempx-2)-log($tempx) + $tempql;
  if ($tempr+$tempc> -78.4) {$fp+=exp($tempr+$tempc);}
  }
 }
if ($switch==1) {$fp = 1-$fp;$temp=$df1;$df1=$df2;$df2=$temp;}
}

sub probt()
{if($t<0){$t=-$t;}
$th=atan($t/sqrt($df));
if($df==1) {$pt=1-$th*2/3.1416;return;}
$sth=sin($th);
$cth=cos($th);
$zz=1;$statcom=$zz;
$statcomi=($df%2==1)?2:1;
while($statcomi<=($df-3))
{$zz*=$cth*$cth*$statcomi/($statcomi+1);$statcom+=$zz;$statcomi+=2;}
if(($df%2)==1) {$pt=1-($th+$sth*$cth*$statcom)*2/3.1416;}
else { $pt=1-$sth*$statcom;}
}
#Two-tail Student's t-test

sub comp()
{if(! defined %comparestatdata) {printoutput("\nNo data to
compare!\n");return;}
$comparepatternnum=0;
$comparepattern='NULL';
printoutput("\n");
where: for($i=1;$comparepattern ne '';$i++)
 {do
  {printoutput("Grouped comparison data pattern $i:");
  $comparepattern=<>;
  while($comparepattern=~/\!/)
   {print "Please refer to the format of previous output for proper
specification for a pattern. You may substitute spaces for underscores,
but the order of elements should be kept and Perl RegExp escaped.\n";
   print "Grouped comparison data pattern $i:";
```

```
    $comparepattern=<>;
    }
  if($macro){print fdmacro $comparepattern;}
  print fdlog $comparepattern;
  chomp $comparepattern;
  printoutput("\n");
  $comparepattern=~tr/a-z/A-Z/;
  $comparepattern=~s/\s+/\_/g;
  if($comparepattern eq '') {next where;}
  undef %statdata;
  $comparenum=0;
  foreach $comparesample (keys %comparestatdata)
   {if($comparesample=~/$comparepattern/i)
    {$comparenum++;
    $comparesampletemp=$comparesample;
    printoutput("$comparesampletemp: ");
    printoutput("%-6.3g\n",$comparestatdata{$comparesample});
    $statdata{$comparesample}=$comparestatdata{$comparesample};
    }
   }
  if($comparenum<2)
   {printoutput("Each group pattern must have at least two
matches.\n");
   $i++;next;
   }
  printoutput("Accept this grouping?(y for yes, any other key for
no):");
  $accept=<>;
  if($macro){print fdmacro $accept;}
  print fdlog $accept;
  chomp $accept;$accept=~tr/a-z/A-Z/;
  printoutput("\n");
  if($accept eq 'Y')
   {ave();
   $compare{$comparepattern}=$count."*".$ave."+".$sd;
#Transfer compare pattern statistics to be compared, see above.
   }
  }while($accept ne 'Y');
 $comparepatternnum++;
 }
if($comparepatternnum<2)
 {printoutput("Less than 2 groups. Grouped comparison aborted!\n");
 return;
 }
%statdata=%compare;
&ttest();
printoutput("\nANOVA: ");
&vtest();
undef %compare;
undef %statdata;
}

sub printoutput()
{if(! defined $_[1]) {print $_[0];print fdlog $_[0];}
else {printf @_;printf fdlog @_;}
}
```

```
sub help1()
{print "Please give me a partial or full sample name. Press '.' for all
samples. A Full sample name may help to locate all of its available
gates. Alternatively, partial sample pattern, or even perl regular
expression can be used to indicate a pool of samples.\n";
}

sub help2()
{print "Gate name patterns are handled similarly to sample names.
Remember to escape Perl regular expression control characters such as
'+' and '*'.\n"; }

sub help()
{print <<EOH;
#.....
#code printing Chapter IV of this thesis is omitted
#.....
EOH
exit;
 }
```

BIOGRAPHY OF THE AUTHOR

Yutao Fu was born in Beijing, P.R.China on October 28, 1975. He graduated from Nankai High School in 1994. He attended Nankai University and graduated in 1998 with a Bachelor's degree in Biochemistry and Molecular Biology. He entered the Biochemistry, Microbiology and Molecular Biology graduate program at the University of Maine in the spring of 1999.

Yutao Fu is a candidate for the Master of Science degree in Biochemistry from the University of Maine in December, 2002.