Identifying the Roles of Insulin, Prolactin and Glucocorticoid in the Initiation of Murine Lactogenesis

Maurice H. T. Ling

A thesis submitted in partial fulfilment of the requirement for the degree of
Bachelor of Science (Degree with Honours)

Department of Zoology
The University of Melbourne
Parkville, Victoria 3010
Australia

April 2004 (Discovery)

Deca-annum, waxing noon, that day
Silent smile you lay
Unknowst your sight on me
Battling tears, I scribe this dedication for thee...

A decade ago, my Grandfather left abruptly.
Alone, he is, but with a smile.
I will never know what he dreamt of me.
Ten years, I had walked so far...

This day, twenty-five years ago, you first held me in your arms.
This day, twenty-five years later, I dedicate my thesis to you,

My Grandfather
who died but remains alive in my thoughts.

Abstract

Lactation is the only biological process evolved to meet all of the nutritional demands of the early stages of development in mammals. A recent microarray study, using whole mammary gland, showed changes in expression level in thousands of genes at lactogenesis. Previous studies using mammary explants demonstrated that insulin (I), prolactin (P) and glucocorticoid (F) are required for murine lactogenesis. However, no study had been done to demonstrate that IFP-induced lactogenesis in explants mimics what occurs *in vivo*. The first part of this project attempts to establish the validity of mammary explant culture in the study of lactogenesis. Microarray comparison was used to examine differential gene expression in explants cultured in IFP for 2 days and day 2 lactating mammary glands showed that there were no major differences in gene expression with the exception of inflammation response genes in cultured mammary explants. This indicated the need to culture explants for longer time to adapt to culture conditions. The second part of this project used mammary explants and microarray profiling to study the contributions of insulin, prolactin and glucocorticoid in murine lactogenesis. Mammary explants were allowed to adapt for 8 days in media before culturing in IF, IP, FP and IFP media for 2 days. Microarray comparisons were used to examine the levels of gene expression in the explants cultured in IFP and IF, IP and FP. Prolactin seemed to increase the potential for insulin signaling and induced gene transcription and translation factors. Furthermore, results suggested that prolactin may have a role to recruit other hormonal effects, such as, an enhanced response to angiotensin and 5-hydroxytryptamine-3. Insulin might have a role in promotion secretion, in addition to glucose metabolism. Glucocorticoid induced genes that may be implicated in protein folding and glycosylation. Taken together, this project has demonstrated the advantages of using the mammary explant model, together with microarray technology, to more clearly define the contributions of insulin, prolactin and glucocorticoid in the complex regulation of lactogenesis.

Acknowledgements

First and foremost, I will like to thank Dr. Kevin Nicholas and Dr. Matthew Digby for agreeing to take me on and be my supervisors. I am greatly indebted to both of you for your advice and guidance throughout this year, for teaching me how to see the big scheme of things while keeping sight of details. Your guidance had allowed me to grow in many ways, thank you.

I'll like to thank all the staff and students of the Nicholas group for all your assistance and teachings and accomodating with my weird and perhaps, cranky ways throughout the year. A person I must thank personally is Sonia Mailer, whom had helped me both in my project and at personal level. Thank you Sonia, at the very least, thank you for getting me back that night at Lorne and introducing me to Sparking Shiraz. Cate, although I don't know you very well but I'll remember you for that sausage-in-the-hallway analogy. Christophe, how can I forget you. Although I still cannot fully make sense of your accent, I truly appreciate your input into the project.

Thank you Dr. Mary Familari, for introducing me to this lab. Sorry, if I had gotten you running stressfully then. Thank you Mary and Kevin, for your attendance to my convocation and all your advices to me. To Prof. Derek Chan and Ms. Penelope Fairbank, your advices during those unsettled times in my final year had helped me pull through it. Thank you.

I wish to extend my appreciation to the CRC of Innovative Dairy Products for supporting my honours year. And the Dept. of Zoology for making my year a entertaining one. The tea sessions were simply great.

I must thank my dad and mum for putting me in Australia to further my studies. Even though I've never said it, I can deeply felt all 14 hours of cab-driving everyday to put me where I am. To my mum, despite not knowing what am I studying, you are always supporting me emotionally in pursuing my goals. Thank you for believing in me and telling me to believe in myself when everyone thinks otherwise. Mum, even though you don't know what am I doing here in Melbourne, but what I'm doing is integral to me since you had sustained me with your milk for the initial part of my life. Melvin, my brother, although I love to pull your face and calling you “edible cat”, I know it's tough to be forced to take my place in the family while I'm here. You are a good brother.

To all the friends who were with me from Singapore, especially Edwin and Joly, your faith in me exceeded what I had in myself. You are always so full of encouragements when I just needed it. I'd have probably wrecked myself without you guys. You guys are to me probably what heroin is to an addict. Mushy but I'm addicted on you people.....

Lastly, to all the honours student of 2003/2004, especially Bojun Bjorkman-Chriswell, April, Ange, Sarah, Paul, Mike and Malcolm. What can I say, you guys are great to have in honours room, never short of entertainment. With special thanks to Bojun, you are probably wondering what is that guy behind you doing half of his time, with his occasional spurt of abusive words. Thank you all for making the honours year an enjoyable one. Good luck for all your future work and hope we'll meet again.

Table of Contents

List of Figures

List of Tables

Abbreviations

aadNTPs	aminoallyl dNTPs
BASE	BioArray Software Environment
BSA	bovine serum albumin
cDNA	complementary DNA
CO_2	carbon dioxide
DEPC • H_2O	diethylpyrocarbonate treated water
DNA	deoxyribonucleic acid
dA	deoxyadenosine
dNTPs	deoxynucleic acid triphosphates
dT	deoxythymidine
EDTA	ethylenediaminetetraacetic acid
F	hydrocortisone
FCS	fetal calf serum
I	procine insulin
ipH_2O	water for injection
M199	Medium 199
M-MLV	Moloney Murine Leukemia Virus
milli-Q water	deionized water (>18.2MΩ per cm)
mRNA	messenger RNA
NIA	National Institute of Aging
P	ovine prolactin
PCR	polymerase chain reaction
RNA	ribonucleic acid
RT-PCR	reverse transcriptase PCR
SSC	standard sodium citrate
TAE	tris-acetate-EDTA
TBE	tris-borate-EDTA
TE	tris-EDTA
UV	ultraviolet

Units

MΩ	mega-Ohms
%	percent
°C	degrees Centigrade
cm	centimetres
g	gravities
k	thousand
M	molar
mg	milligrams
mm	millimetres
nm	namometres
pH	concentration of hydrogen ions in logarithm base 10 at 25°C
nmole	nanomoles
pmole	picomoles
w/v	weight per volume
U	units
µg	micrograms
µl	microlitres
µmole	micromoles

Chapter 1:
Introduction

The mammary gland is a complex tissue in terms of developmentally-regulated changes in cellular organization and function during the lactation cycle. Despite years of research, many questions pertaining to the endocrine regulation of the initiation of lactation (lactogenesis) remain unanswered. Before defining the questions relating to lactogenesis in this project, it is essential to understand the general development of the mammary gland. A brief overview of mammary gland development will be presented, followed by an in depth assessment of the molecular mechanisms underlying the roles of insulin, prolactin, and glucocorticoid in lactogenesis. Finally, the potential of functional genomics (microarray technology) will be considered as an approach to better define the role of the hormones regulating this process.

1.1 Mammary Gland Development

Mammary gland tissue develops dimorphically from the mammary crest from about Day 11 of mouse foetal development (Topper and Freeman, 1980). Further growth of the mammary crest is limited until late gestation when rapid epithelial growth occurs in females to form a mammary cord which opens to the eventual nipple. In contrast, in males, epithelial growth is not observed. Instead, the primitive mammary duct ruptures and is isolated in subepithelial mesenchyme (Topper and Freeman, 1980). In females, the mammary cord remains dormant (Topper and Freeman, 1980) until stimulated by a combination of hormones at sexual maturation (Ball, 1998; Brisken et al., 1999; Hennighausen and Robinson, 2001; Vonderhaar, 1988), at which time branching (from the nipple) is seen throughout the mammary fat-pad (see Figure 1) (Couldrey et al., 2002; Walden et al., 1998). The ducts remain a minimal distance of 0.25mm apart (Faulkin and Deome, 1960). During pregnancy, growth constraints between each duct are lifted and lobular-alveolar structures are observed throughout the fat pad, in preparation for lactation (Desjardins et al., 1968; Keough and Wood, 1979). Early clinical studies revealed elevated levels of progesterone during pregnancy has an inhibitory role in lactation, and the removal at parturition, forms part of the lactogenic trigger (Topper and Freeman, 1980). As well as the presence of a neuronal role in lactation (Jirikowski,

1992), prolactin and glucocorticoid were found to be essential for the lactogenic processes *in vivo* and *in vitro* (Chomczynski et al., 1986; Ganguly et al., 1979; Ganguly et al., 1980b; Nagaiah et al., 1981; Yoshimura and Oka, 1990). After weaning, the mammary lobuloalveolar duct system undergoes involution, characterized by apoptosis and tissue remodelling, which returns the glands to a near pre-pregnancy state (Accorsi et al., 2002; Baik et al., 1998; Li et al., 1996; Tatarczuch et al., 1997), as shown in Figure 1.

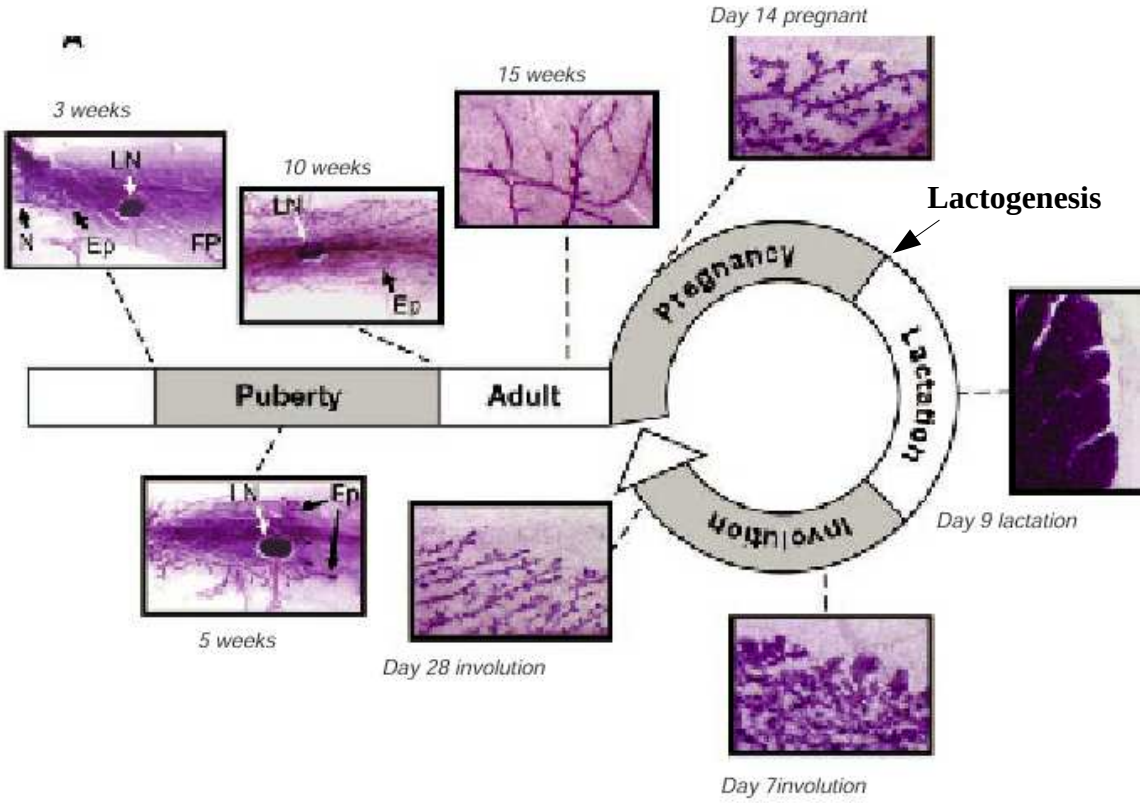

Fig. 1. Mammary Gland Development in Mouse. Whole mounts of murine mammary gland are shown to demonstrate the dramatic growth of the epithelial ductal tree during puberty and of the alveolar epithelial compartment during pregnancy. After lactation and postlactational involution, the gland returns to a state resembling the adult nulliparous gland. Glands shown were harvested at 3 weeks of age (onset of puberty), 5 weeks of age(mid-puberty), 10 weeks of age (completion of puberty), 15 weeks of age (adult nulliparous), day 14 pregnancy, day 9 lactation, day 7 involution, and day 28 involution. The nipple (N), epithelial tree (Ep), fat pad (FP), and lymph node (LN) are indicated. (Adapted from Master et al., 2002)

1.2 Endocrine Control of Lactogenesis

Research using the mammary explant model has confirmed the *in vivo* studies in mouse showing that insulin, prolactin and glucocorticoid are essential for the initiation of lactation (Topper and Freeman, 1980; Topper et al., 1984a). However, the contribution of each hormone, in the presence of the other two, is not well understood.

1.2.1 Prolactin

Prolactin, was one of the first hormones to be identified as a major regulator of lactation (Fulkerson et al., 1975; Sinha et al., 1974; Weinstein et al., 1976), and was shown to circulate at a high level in serum throughout lactation (Hart, 1972). Prolactin binds to a membrane-bound receptor, which is positively regulated by prolactin at physiological levels (Bohnet et al., 1977; Djiane et al., 1979), and activates two proteins; Signal Transducer and Activator of Transcription-5a (Stat5a) (Majumder and Turkington, 1971; Yamashita et al., 2001) and Janus Kinase 2 (JAK2) (Liu et al., 1997; Majumder and Turkington, 1971). Prolactin signalling results in a large number of downstream effects in mammary epithelial cells, for example, transcription of the β -casein gene, and stabilizing β -casein mRNA (Ganguly et al., 1980b; Kulski et al., 1983a). Stat5a is known to dimerize with Signal Tranducer and Activator of Transcription-5b (Stat5b) and the role of Stat5a/Stat5b heterodimer (Ahonen et al., 2002) is known to activate several transcription factors downstream required for expansion of mammary genes (Malewski et al., 2002; Zhao et al., 2002).

1.2.2 Glucocorticoid

Glucocorticoids are membrane permeable steroid hormones that bind to a cytoplasmic receptor, which is subsequently translocated into the nucleus. Prior to lactogenesis, glucocorticoid-bound receptor was found to be localized at endoplasmic reticulum and golgi apparatus (Mills and Topper, 1969; Mills and Topper, 1970). Glucocorticoids also had been suggested to have a role in determining the half-life of milk protein gene mRNA (Dahlquist et al., 2002).

Using the mammary explant model, glucocorticoid was found to play an essential role in the transcription and stabilization of β -casein mRNA (Chomczynski et al., 1986; Ganguly et al., 1979; Kulski et al., 1983b; Nagaiah et al., 1981) in the presence of prolactin and insulin (Ganguly et al., 1980b). However, early studies demonstrated that cortisol, the major glucocorticoid in mice, is retained in the mammary fat pad (Bolander et al., 1979), suggesting that this tissue may act as a cortisol reservoir. Therefore, particular care is required for the experimental design used to attribute a response to cortisol (Bolander et al., 1979) to prevent retention of cortisol in the fat pad.

1.2.3 Insulin

It was recognized very early that insulin played a role in lactation *in vivo* (Scmidt, 1966) but it was not until more recently, that insulin was shown to be essential for transcription of the β -casein gene (Bolander et al., 1981; Chomczynski et al., 1986; Kulski et al., 1983a; Topper et al., 1984a; Topper et al., 1984b) . These studies using the mammary explant model removed the "misconception" that insulin only plays a role in cell maintenance *in vitro* (Nicholas et al., 1983). Insulin-induced gene expression were activated by the insulin receptor on the cell surface (Nicholas and Topper, 1983). At the same time, insulin has an important role in uptake of nutrients by the mammary gland (Bequette et al., 2001; Griinari et al., 1997; Mackie et al., 2000) while causing other tissues to be insensitive to insulin by perpetual hyperinsulinemia (Bequette et al., 2002). In addition, insulin has a role in making mammary cells more responsive to other hormones, such as, growth hormone (Butler et al., 2003).

1.2.4 Complex endocrine regulation of lactogenesis

The regulation of milk protein gene expression is the result of a complex interplay of hormones and their cross-talk signal transduction pathways as shown in Figure 2. To accommodate this cross-talk, milk protein genes are regulated by composite response elements (CoRR), such as, Stat5/C/EBPβ , which regulates murine β -casein gene expression (Wyszomierski and Rosen, 2001). Converging signalling pathways potentially implicated in the regulation of lactation include: insulin-like growth factor-1 (IGF-1) and insulin by insulin receptor substrate-1 (IRS-1) (Giovannone et al., 2000); prolactin and insulin by Raf (MAP Kinase pathway) (Bole-Feysot et al., 1998); glucocorticoid and prolactin by Jak2 and Stat5 (Groner, 2002).

A volume of prior studies has collectively demonstrated that a combination of insulin, prolactin and glucocorticoid are essential for the transcription and translation of β -casein gene as a marker for lactogenesis (Baldwin and Louis, 1975; Ganguly et al., 1980a; Houdebine et al., 1985; Mills and Topper, 1970; Turkington et al., 1967). However, very little is known about the individual influence of each of the hormones within the endocrine complex required to initiate lactation.

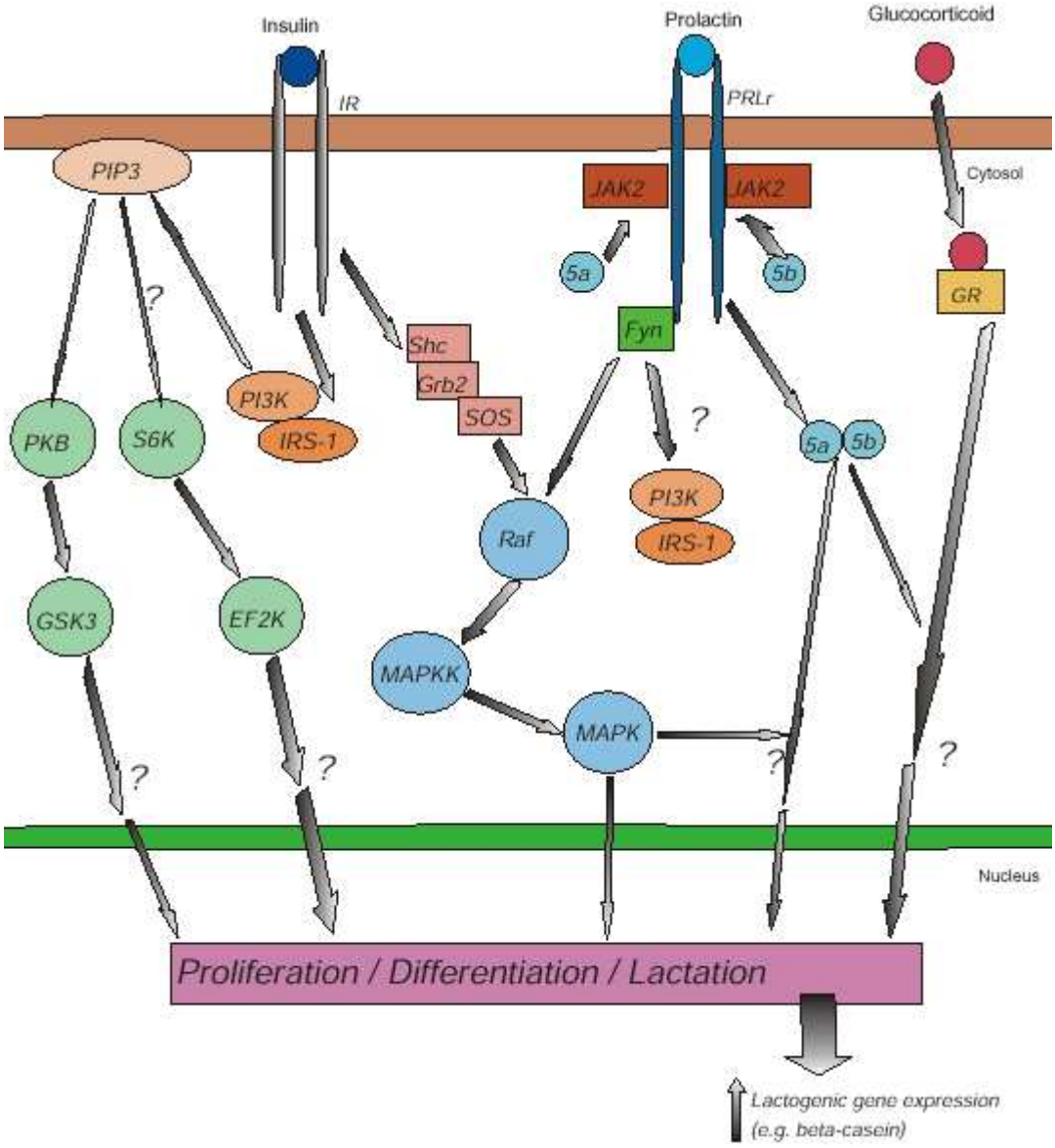

Fig. 2. Complex Endocrine Control of Mammary Gland. Insulin and prolactin bind transmembrane receptors, which will auto-transphosphorylate (activate). Activated insulin receptor (IR) activates MAP (mitogen activated protein) kinase pathway via Shc/Grb2/SOS complex. IR also activates PI3K (phosphatidylinositol-3 kinase) via IRS-1. PI3K cleaves PIP3, which activates PKB (protein kinase B) and maybe S6K. PKB activates GSK3 (glycogen synthase kinase 3). Activated prolactin receptor (PRLr) attracts JAK2, which dimerizes Stat5a (5a) and Stat5b (5b), and Fyn (a Src kinase), which activates MAP kinase pathway. Glucocorticoid binds to glucocorticoid receptor (GR) intracellularly and translocate into the nucleus. Stat5a/5b complex may interact with GR. Details of pathway interaction remains unclear, although it leads to the expression of lactogenic genes, such as, β -casein.

Microarray technology, which analyzes expression of thousands of genes in a single experiment, can enable us to study the resultant effects of this complex signal transduction mechanism. Therefore, this approach coupled with the mammary explant

culture model, will more clearly delineate the specific roles of insulin, prolactin, and glucocorticoid for lactogenesis.

1.3 Use of Microarray Technology to Study the Regulation of Lactogenesis

Microarray technology represents a high capacity technique that allows the analysis of the expression of thousands of genes per experiment. This contrasts with Northern hybridization, which is limited to the analysis of a few genes, per hybridization. As shown in Figure 3, a DNA sample representing a single gene is printed (blotted) in grid format on a microscope slide. Two samples of fluorescently-labeled mRNA (treatment versus control or treatment-x versus treatment-y), corresponding to "test" and "reference" in Figure 3, are hybridized on the slide, the slide is washed and the fluorescent signal measured using a scanner.

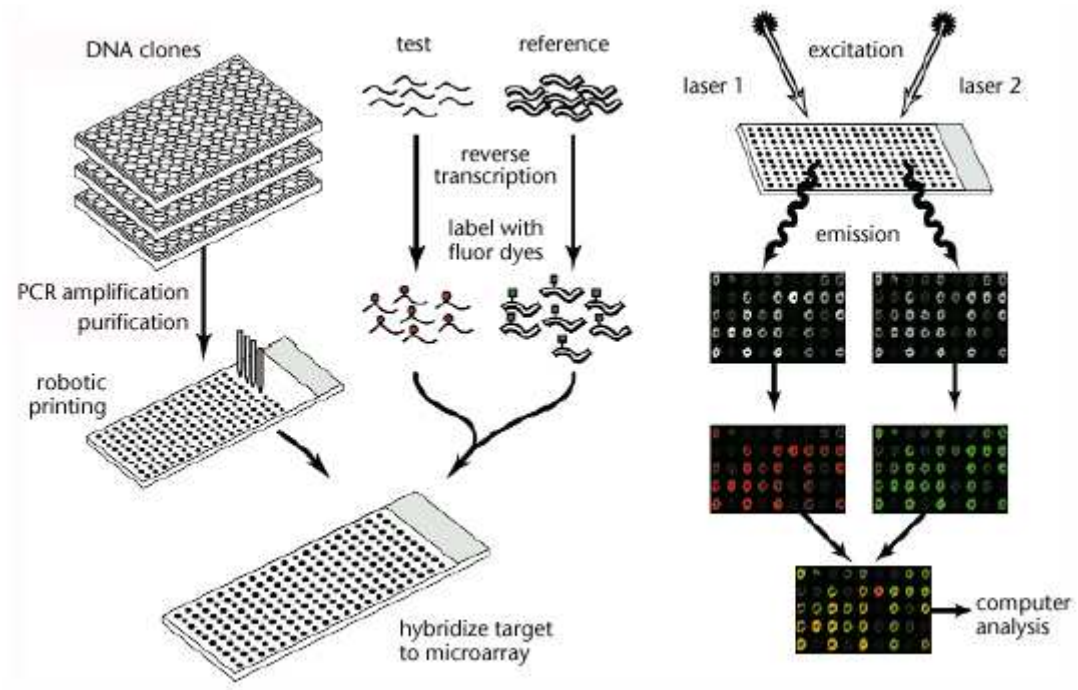

Fig. 3. cDNA Microarray Schema. Templates for genes of interest are obtained, amplified by PCR, and printed on coated microscopic slides. Total RNA from both the test and reference sample are fluorescently-labeled with either Cye3- or Cye5-dUTP using reverse transcription. Labeled samples are allowed to hybridize under stringent conditions to the clones on the array, after which the slide is scanned. Monochrome images from the scanner are processed. Data from a single hybridization experiment is viewed as a normalized ratio (that is, Cye3/Cye5) in which significant deviations from 1 (no change) are indicative of increased (>1) or decreased (<1) levels of gene expression relative to the reference sample. (Adapted from Duggan et al., 1999)

Two kinds of commonly used sources of DNA for microarray printing are either cDNA (complementary DNA)/EST (expressed sequence tags) or oligonucleotides. The

cDNA/EST microarrays are suited for expression studies and can be viewed as highly efficient counterparts of Northern hybridization. Both cDNA/EST and most oligonucleotide microarrays measure gene expression as a ratio between treatments. A more recent development has been oligonucleotide microarrays designed to measure absolute expression levels (Affymetrix).

1.3.1 Transcript Profiling of Mammary Tissue During the Lactation Cycle

A recent study has reported the use of Affimetrix microarrays to identify the murine mammary-specific transcriptome at each stage of pregnancy, lactation, and involution (Master et al., 2002). This study, using 5000-gene Affymetrix microarrays, showed that changes in the expression level of a large set of genes were observed during the transition from pregnancy, lactation and involution. Examining the data from late pregnancy and early lactation, suggested as much as 70% to 80% of all tested genes were differentially expressed across the phase. However, the study used an *in vivo* system, making it impossible, without the use of extensive drug treatments and invasive surgery, to identify the roles of insulin, prolactin and glucocorticoid, singly or in combination, at the initiation of lactation.

In contrast, to study lactogenesis *in vitro*, mammary explants from pregnant mice can be easily manipulated to study its responses to various hormone treatments. However, despite the assumption that results from studies using explants will resemble *in vivo* conditions, this assumption has never been formally tested. Therefore, even though the explant culture model is useful in studying hormonal regulation of specific mammary responses, for example, the expression of milk protein genes, the extend to which such results can be extrapolated to the animal model remains unclear.

1.4 High-Throughput Data Analysis

Prior to microarrays, no single biological experiment could yield such an enormous amount of data. Hence, a large area of mathematical statistics and informatics has been directed to support microarray experimental data analysis.

1.4.1 BASE as a Microarray Database and Development Platform

A number of databases had been specifically designed to store results from microarray experiments, such as, BioArray Software Environment (Saal et al., 2002). BioArray Software Environment (BASE) is an open-sourced software created by Lund University and is licenced under GNU General Public Licence. BASE is written as a microarray software platform, meaning that, BASE can be used as a developmental platform for software tools used in microarray analysis. Being a relatively new software, BASE is sparse in documentation to guide users in using all of the functions provided.

1.4.2 Microarray Data Analysis

Before analysis of gene expression (Figure 4), image scans from each microarray slide must be normalized. This is done within a slide to remove intra-array variations and across array to remove inter-array variations (Schadt et al., 2001; Tseng et al., 2001; Yang et al., 2002).

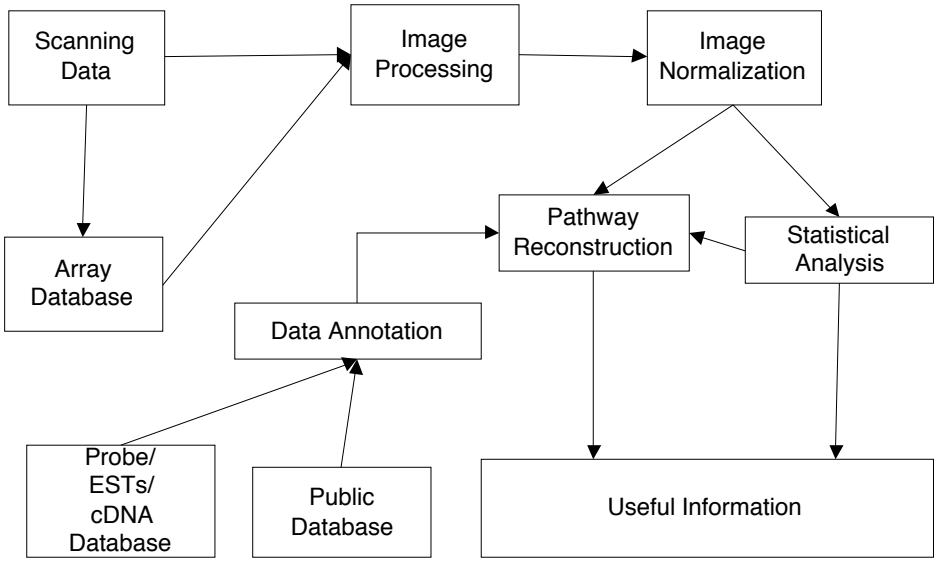

Fig. 4. General Scheme of Analyzing Microarray Data. Raw scanned images can be stored into a database or processed immediately. Image processing includes assembling repeated scans and format conversion whereas image normalizing adjusts image intensities, inter-chip and intra-chip, by using a series of standards built onto the microarray chip. Image processing and normalization forms the data preprocessing step. Information about the probe/ESTs/cDNAs embedded on the chip are annotated by referencing it to public databases. This knowledge will then be superimposed on the normalized image, giving each spot an identity. Only then, can it be further processed by either statistical analysis or pathway reconstruction approaches.

After data pre-processing, actual analysis can proceed using a combination of statistical and pathway reconstruction approaches. Statistical means includes classical statistics, such as, Student's t-test (Baldi and Long, 2001) and analysis of variance (Didier et al., 2002) and their derivatives (Broberg, 2003; Brown et al., 2001; Butte et al., 2001; Chapman et al., 2002; Chen et al., 2002; Culhane et al., 2002; Fuhrman et al., 2000; Hastie et al., 2000; Holter et al., 2001; Holter et al., 2000; Hoon et al., 2002; Kerr and Churchill, 2001; Kroll and Wolfl, 2002; Li and Hong, 2001; Manduchi et al., 2000; Sasik et al., 2001; Taib, 2003; Wall et al., 2001; Xing and Karp, 2001; Zhou and Abagyan, 2002) or learning methods, such as, neural network (Herrero et al., 2003), self-organising map (Kasturi et al., 2003; Tamayo et al., 1999) and support vector machines (Brown et al., 2000; Pavlidis et al., 2001). Using these approachs, patterns of gene expression which may be present across treatments can be easily identified.

Pathway reconstruction makes use of the readily available knowledge of metabolic or regulatory pathways and maps the genes expressed onto known pathways, which will provide information as to the nature of pathways being active in each treatment. This is in contrast with statistical analyzes which do not require prior knowledge of biochemical processes. Pathway reconstruction uses two different approaches, either regulatory pathways, such as, by promoter analysis, or metabolic pathways. Some of the techniques used in this area include, functional clustering (Hawse et al., 2003; Kalish et al., 2004; Kwon et al., 2004; Lee et al., 2003; Pusztai et al., 2003; Song et al., 2003; Tabibiazar et al., 2003), weight matrices (Weaver et al., 1999) and pathway scores (Zien et al., 2000), which provides increased biological relevance. Statistical analysis and pathway reconstruction are not mutually exclusive to each other but can be used together to yield important results (Herzel et al., 2001).

1.5 Conclusion and Significance of Research

A recent study has illustrated the mammary-specific transcriptional changes at the initiation of lactation (Master et al., 2002) and it is clear from early studies using *in vitro* models that insulin, prolactin and glucocorticoid are required for the transcription and accumulation of milk protein mRNA. However, the roles of insulin, prolactin and glucocorticoid for the initiation of lactation are unclear. Over the last 40 years, mammary explant culture has proven to be useful in studying hormonal regulation of mammary

cells, nevertheless, the relevance of results from explant studies to the response of the mammary gland *in vivo* remains unknown. Hence, this research described in this thesis uses microarray technology as a comprehensive approach to study the differences of the mammary explant model and *in vivo* system. Subsequently, the in vitro model has been used to better define the roles of the hormones (insulin, prolactin and glucocorticoid) during the initiation of lactation.

Chapter 2:

Materials and Methods

2.1 Mice

Pregnant (day 11 to 13) and lactating (day 2) Balb/C mice were used for experiments, under the permit AEESC 02077. The mice were housed and time-mated in the animal house of the Department of Zoology, The University of Melbourne, where feed and water were provided *ad libitum*. The mice were sacrificed by cervical dislocation.

2.2 Mammary Explant Cultures

All the mammary glands were excised under sterile conditions. The tissue was cut into 1-2mg pieces and cultured in 6-well (35mm) tissue culture plates (Nunc, 52795) in M199 under 5% CO_2 atmosphere at 37°C (Bolander et al., 1979). Explants were placed directly into the media. The concentrations of hormones used were: porcine insulin, abbreviated to "I", (Sigma, I-5500) at 0.1µg/ml; ovine prolactin, abbreviated to "P", (National Hormone and Pituitary Program, USA, NIDDK-oPRL-21) at 0.2µg/ml; and hydrocortisone, abbreviated to "F", (Sigma, H-4001) at 0.05µg/ml.

2.3 RNA Extraction and Quantification

The RNA was extracted from cultured tissues and whole mammary glands using a QIAgen RNeasy Lipid Tissue Mini Kit (74804), following the manufacturer's recommendations. The RNA was precipitated in 2 volumes of absolute ethanol and 0.1 volume of 3M sodium acetate (pH 5.5) and stored at -80°C until use.

When required, the precipitated suspension was briefly vortexed to resuspend the RNA before aliquoting. The aliquote was centrifuged at 16000g at 4°C, and the pellet was dried to remove traces of alcohol before re-suspending in a suitable volume of TE buffer or DEPC treated water (DEPC • H_2O).

The RNA was quantified using UV spectrophotometry at 230nm, 260nm and 280nm, and analyzed on 1% (w/v) agarose gel in TBE buffer.

2.4 Reverse Transcriptase PCR

First strand cDNA was synthesized using SuperScript II RNase H- Reverse Transcriptase (Invitrogen, 18064-014) following the manufacturer's recommendations. PCR was carried out with 1µl of cDNA using *Taq* DNA polymerase (Promega, M1661) in a volume of 50µl, containing 1X *Taq* buffer, 1.25U of *Taq* DNA polymerase, 12.5pmole of dNTPs, 45pmole of Mg^{++} , 0.025pmole of reverse primer (5'TCTAgA gTCCggTggTgTCACTACAgg3') and 0.25µg of forward primer (5'CCggAT CCATgATgCATTTCgTTCCTTTg3') (Coralie Reich, Personal Communication). The reaction was heated to 94ºC for 3 minutes, followed by 30 cycles of 94ºC, 59ºC and 72ºC for 30 seconds each, and finally heated at 72ºC for 7 minutes. The product was analyzed on 1.5% (w/v) agarose gel in TAE buffer.

2.5 Microarray Techniques

All solutions used in microarray hybridization are listed in Appendix F. Microarray slides used were the NIA 15000 murine cDNA microarray, printed by Peter MacCallum Cancer Research Institute, Victoria, Australia. The ESTs printed on this array are listed at http://lgsun.grc.nia.nih.gov/cDNA/15k.html

Reverse Transcription: 75µg of RNA, scorecard (Amersham) and 4µg of oligo dT primers in 20.9µl ipH_2O were incubated at 70ºC for 10 minutes before adding M-MLV buffer, 400U of M-MLV Reverse Transcriptase (Promega, M3682), 60nmoles of amino allyl dNTPs (Sigma), and incubating at 42ºC for 2.5 hours.

First Purification: The reaction was hydrolyzed with 2.5µmole of sodium hydroxide and 2.5µmole of ethylenediaminetetraacetate (EDTA), and incubated for 15 minutes at 65ºC, before neutralizing with 3µmole of acetic acid. The reaction products were purified using QIAgen PCR Purification Kit (28104), following manufacturer's recommendations but without elution.

Pre-Hybridization: Microarray slides were prepared by incubation in pre-hybridization solution for 40 minutes at 42ºC, followed by 3 washes with milli-Q water and dried.

Coupling: The Cy3 and Cy5 dye (Amersham) were each hydrated in 20µl of 0.1M fresh sodium bicarbonate and added to the labeled cDNA, followed by a 1 hour incubation in the dark at room temperature. After incubation, the probes were eluted with 80µl intra-peritonial water (ipH_2O).

Second Purification: Two samples were purified together using a QIAgen PCR Purification Kit, following manufacturer's recommendations. To the eluate, 12µg of tRNA, 30µg of Cot-1 DNA, 6µg of poly dA and 0.75µl of 50X Denhart's containing herring sperm DNA were added and the solution was concentrated to 11µl before adding 5µl of 20X SSC and 16µl of 100% de-ionized formamide, then incubating at 100ºC for 3 minutes.

Hybridization: A 0.4µl aliquot of 10% sodium dodecyl sulfate (SDS) was added to the probes before application onto a pre-hybridized NIA 15k mouse cDNA microarray slide, and incubated in a humidified chamber for 16 hours at 42ºC.

Washing: Slides were washed with 0.5X SSC with 0.1% SDS, 0.5X SSC, and 0.06X SSC for 3 minutes each before dying and scanning.

2.6 Data Preprocessing and Analysis

Microarray slides were scanned with a BioRad VersArray Scanner. The resultant images were analyzed by the accompanied software to identify individual spots. Data pertaining to each spot (all 15000 spots on a slide) were extracted and normalized using the local regression (LOESS) normalization algorithm, which is included in the software. Data analysis on the normalized data were mainly carried out using computer scripts, written as part of this project, which will be discussed in Chapter 3: Data Analysis.

2.7 Experimental Design

The primary approach in this project was using microarray for analyzing the mouse mammary gland transcriptome. There were 2 parts to this project.

Experiment 1: The project examinesthe differences between mammary explants from pregnant mice cultured with IFP for 2 days and mammary glands from mice at day 2 lactation (hereafter known as 'culture verification').

A direct microarray comparison was made between RNA extracted from lactating mammary glands (day 2 lactation) and explants, which had been cultured for 2 days in insulin, prolactin and cortisol (IFP). Previous studies had shown that milk proteins are inducible in mammary explants from day 12 pregnant mouse within 2 days of culture in IFP-supplemented media (Mills and Topper, 1970). Therefore, mammary glands from Day 2 lactating mice and Day 2 cultures were examined. A microarray comparison between RNA extracted from mammary glands at day 12 pregnant and that at day 2 lactating mammary glands was used to elucidate change in gene expression from mid-pregnancy to lactation, as shown in Figure 5.

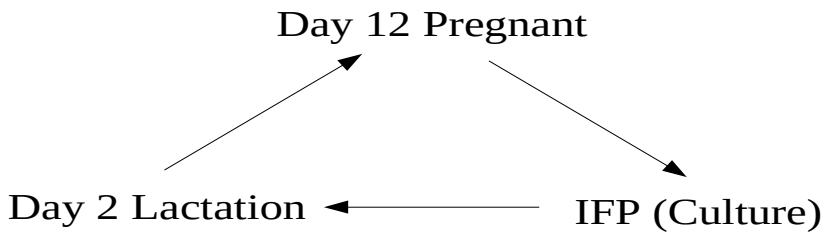

Fig. 5. Microarray Comparisons for Culture Verification Experiment. The arrows represent a microarray comparison, where the arrow tail represents Cy3-labeling of the sample while the arrowhead represents Cy5-labeling of the sample. For example, in the microarray comparison between Day 12 Pregnant and Day 2 Lactation, RNA from Day 12 Pregnant was labeled with Cy5 dye and that of Day 2 Lactation was labeled with Cy3 dye.

Experiment 2: Four combinations of insulin, prolactin and glucocorticoids, were used to elucidate the pattern of gene expression associated with each hormone in the presence of the other two hormones (hereafter known as 'hormone treatments'). Mammary explants from six day 12 pregnant mice were maintained in 10% fetal calf serum (FCS)-supplemented M199 before culture for 2 days in either IF, IP, FP or IFP. Explants cultured in IFP was used as the reference point for microarray comparisons as it accounted for the effects of all the 3 hormones examined. For example, a microarray comparison of RNA extracted from IFP-cultured explants and IF-cultured explants will elucidate the effects of P, in the presence of I and F. Hence, 3 microarray comparisons

were used, as shown in Figure 6. Mathematical reasoning for in this part of the project is discussed in Appendix A.

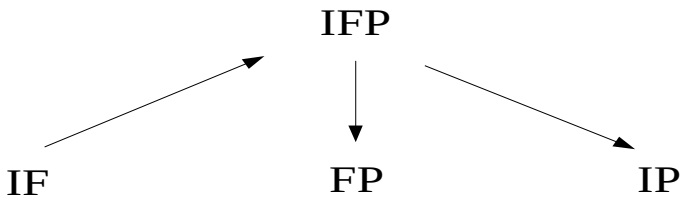

Fig. 6. Microarray Comparisons for Hormone Treatments Experiment. The arrows represents a microarray comparison, where the arrow tail represents Cy3-labeling of the sample while the arrowhead represents Cy5-labeling of the sample. For example, in the microarray comparison between IFP and IF, RNA from IFP-cultured explants was labeled with Cy5 dye and that of IF-cultured explants was labeled with Cy3 dye.

Chapter 3:
Analysis of Microarray Results

The results from microarray scans are normalized to remove intra-array variability which may affect the results. In this study, the results from microarray scans by BioRad VersArray software were normalized using LOESS normalization algorithm before exporting as a comma-separated value (CSV) file for further processing as described in this chapter. Six files were exported, with each file corresponding to one microarray slide, and transformed into BASEfile format. Subsequently, the files were further processed by a series of computer scripts (listed in Appendix D) written as part of this project. This chapter describes the logic behind these processes.

BASEfile format is the output file format of BioArray Software Environment, also known as, BASE (Saal et al., 2002). A detailed description of BASEfile format is given in Appendix B. For the purpose of these analyzes, only median net intensities (median total intensity subtract median local background) of both channels are used.

3.1 External Data Used

Two files of external data were used to facilitate analysis, namely, values of standard deviation of background for each channel, and a table of gene ontology from National Institute of Aging (http://lgsun.grc.nia.nih.gov/cDNA/NIA-CloneSet-GoTermFinder .html).

3.2 Analysis Steps

Normalized data exported from the BioRad VersArray Software provides extensive information describing a single spot on the microarray slide. The data used in this thesis included median raw spot intensity, which gives the median gross intensity of a spot, median background intensity, which gives the local median backgound intensity around the spot, and median net intensity, which is the difference of median raw intensity and mean background intensity. Using this information, exported data were analyzed (see Figure 7 for summary) as follows;

3.2.1 Determining Background Intensity

Within a microarray slide, the median raw intensities and median background intensities fall within a normal distribution when logarithmically transformed (Hoyle et al., 2002). Thus, the distribution of log median net intensities will be normally distributed. The distribution of median net intensities are assembled from individual median raw intensity subtracted by the calculated mean of median background intensity. Plotting the distributions of median background intensities and median net intensities on the same log-linear scale, the distribution of median net intensities will sit on the right of median background intensities. In classical statistical terms, background intensities are analogous to null hypothesis while net intensities are analogous to alternate hypothesis.

Given that there is substantial background on these microarray slides, two standard deviations of background intensity was chosen as the cut-off value (Costigan et al., 2002), instead of three standard deviations of background as it may be too stringent and increases the chances of false negatives (Listing 4 of Appendix D). This meant that only spots with intensity more than two standard deviations of the background intensity will be used for analysis.

3.2.3 Initial Filtering

The data were classified into single expression (expressed only in one of the two samples) and common expression (expressed in both samples), In the common expression, despite requiring the gene to be expressed in both samples, only one of the two intensities is required to be above background intensity. This was done to reduce false negatives, even though it will increase false positives (Listing 4 of Appendix D).

3.2.4 Examination for Patterns of Gene Expression

3.2.4.1 Genes Expressed in Two Experimental Groups

Using a arbitrary benchmark of two times difference in intensities, all of the genes expressed in both samples can be classed as: those that vary less than 2-fold in expression (known as “same level class”) and those that vary more than 2-fold in expression (known as “varied level class”). This criterion of differentially expressed genes had been used in a number of recent studies (Ahn et al., 2004; Stokes et al., 2002;

Xu et al., 2003). Two-fold difference is determined by a intensity ratio of more than 2 or less than 0.5 between each spot.

In the varied level class of genes, spots were grouped according to their difference in intensities, from more than two times, three times, four times, and five times (Listing 5 of Appendix D). This gives an indication of the trend in differential gene expression.

Subsequently, the entire varied level class of genes was divided into two groups. For example, if Gene 1 is expressed 3 times more in Group (Sample) A than B, then Gene 1 was allocated to Group A. These genes were sorted according the the difference in intensities (Listing 6 of Appendix D). The sorted list was then matched against gene ontology data and matching data were appended into the sorted list (Listing 7 of Appendix D), giving functional information about each gene.

Similarly, corresponding gene ontology was matched and appended to the spots in the same level class (Listing 8 of Appendix D).

The common genes expressed in the 3 microarray comparisons in the culture verification experiment will include genes needed for housekeeping and those genes that are mammary specific (Listing 10 of Appendix 10). The common genes expressed in the 3 microarray comparisons in hormone treatment experiment extracted using same means (Listing 10 of Appendix D) should share similarities to the common genes expressed in the culture verification experiment. However, the differences in genes expressed between the two may be illustrative of some of the fundamental effects of culturing for 10 to 12 days (Listing 11 of Appendix D).

3.2.4.2 Genes Expressed Exclusively in One Experimental Group

Similarly, each spot expressed only in one experimental group was matched against gene ontology and corresponding data were appended onto the spot (Listing 9 of Appendix D).

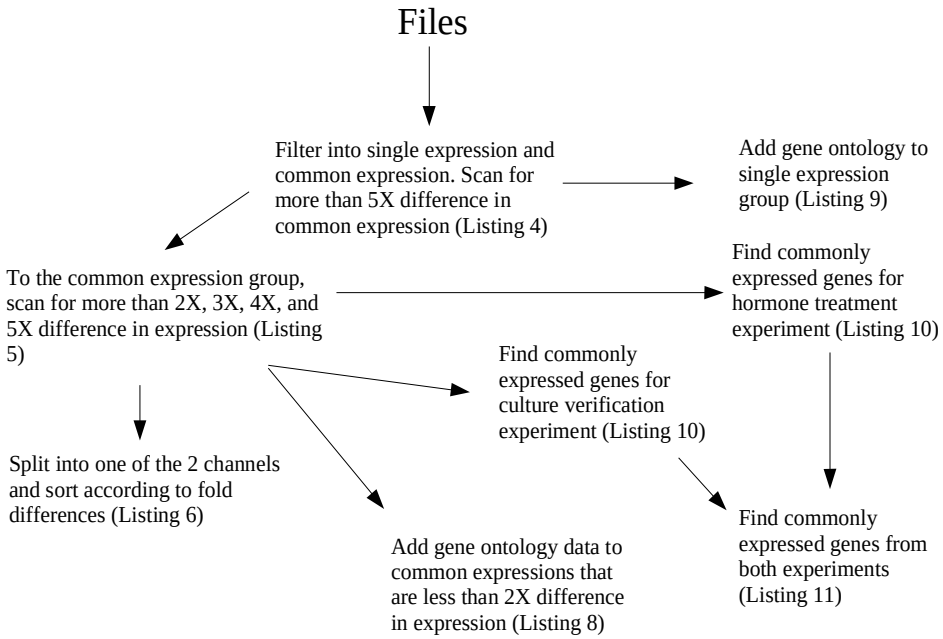

Fig. 7. Flow of Analysis.

Chapter 4:
Assessment of the Mammary Explant Model Using EST Microarray

4.1 Induction of Alpha-lactalbumin Gene Expression in Mammary Explants Cultured With Insulin, Prolactin and Glucocorticoid

Explants were cultured in 10% FCS-supplemented media for 8 days before changing to media that included either IP, IF, FP or IFP. Examination of alpha-lactalbumin gene expression using reverse transcriptase PCR of RNA isolated from the explants showed an amplicon (350bp) in tissue cultured with IFP and the control (RNA from lactating mammary tissue). No expression of the alpha-lactalbumin gene was seen in RNA from explants subsequently cultured for 2 days in either IP, IF or FP (Figure 8). This result indicated that all 3 hormones were required for milk protein gene expression and that 10% FCS did not contribute sufficient I, F or P to FP, IP or IF respectively to stimulate alpha-lactalbumin gene expression. This also showed that the explants were able to be induced to express milk protein genes after 8 days in culture, as alpha-lactalbumin mRNA is a marker for milk protein genes expression (Levieux and Ollier, 1999).

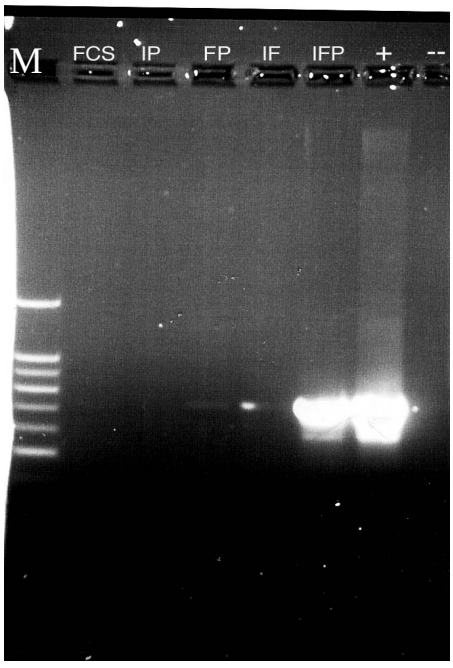

Fig. 8. RT-PCR for Alpha-Lactalbumin. Mammary explants were cultured for 8 days in 10% FCS and subsequently for 2 days in either IP, FP, IF or IFP together with 10% FCS. RNA was isolated from explants and analyzed for alpha-lactalbumin gene expression by RT-PCR. The amplicon was 350 bp. RNA from the mammary gland of a mouse at day 2 was the positive control (+) and water replaced RNA in the reaction mix of the negative control (-). Markers (M) are shown in bp.

4.2 Characterization of Extracted RNA for Microarray

RNA extracted from tissues was analyzed by agarose gel electrophoresis for integrity and spectrophotometry absorbance at 230, 260 and 280nm were used to examine purity. Figure 9 shows RNA isolated from mammary explants cultured in IFP, day 12 pregnant mammary gland and day 2 lactation mammary gland. Two intact bands of ribosomal RNA (18S, 28S) and no low molecular weight RNA smears, indicative of RNA degradation, were observed in all three samples indicating that there is minimal RNA degradation.

Table 1. Spectrophotometry of RNA for Culture Verification Experiment

Sample	***A260***	***A280***	***A260/A280***	***A230***
Day 12 pregnant	2.094	1.140	1.837	1.177
Day 2 Lactation	2.059	1.095	1.880	1.029
IFP	0.705	0.385	1.829	0.308

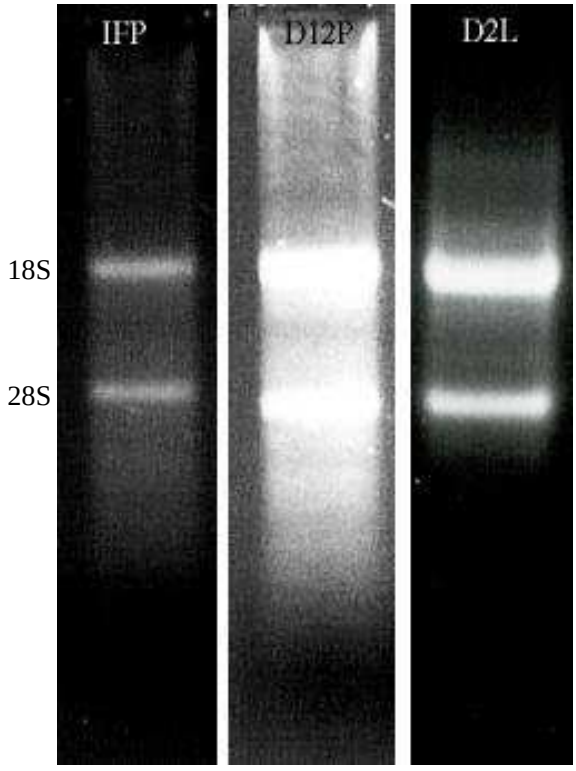

Fig. 9. Agarose Gel Analysis of Extracted RNA Samples for Culture Verification Experiment. Total RNA was electrophoresed on a 1% agarose gel and stained with ethidium bromide. IFP, D12P and D2L represents RNA isolated from mammary explants cultured in insulin, prolactin and glucocorticoid, day 12 pregnant mammary gland and day 2 lactation mammary gland respectively. The 18S and 28S ribosomal bands are shown.

The ratios of A260 to A280 of all 3 RNA samples are above 1.8 (Table 1), indicating that the RNA samples were free from protein contamination. In addition, the absorbance

readings at 230nm did not vary much from the corresponding reading at 280nm, indicating that the samples were free from phenol contamination.

4.3 Microarray Analysis to Examine the Differences Between Mammary Explant Culture and *In Vivo* Lactating Mammary Glands

The microarray experiment for this part of the project was directed toward the question: What are the differences in gene expression between mammary explants induced to lactate with IFP and lactating mammary gland?

4.3.1 General Overview of Microarray Results

Using the results of spots on the microarrays that were higher than background intensity, the microarray comparisons between day 12 pregnancy (D12P) and day 2 lactation (D2L), D12P and explants in IFP (IFP) and D2L and IFP showed the number of genes expressed in each microarray were as follows: in D12P with D2L, 1251 genes were expressed in D12P and 2998 genes were expressed in D2L. A comparison of D12P with IFP showed that 5098 genes were expressed in D12P and 11071 genes were expressed in IFP. A comparison of IFP with D2L showed that 7527 genes were expressed in IFP and 6578 genes were expressed in D2L.

Using the results of those spots with intensity higher than two standard deviations from background intensity (Ahn et al., 2004; Stokes et al., 2002; Xu et al., 2003), the microarray comparisons between day 12 pregnancy (D12P) and day 2 lactation (D2L), D12P and explants in IFP (IFP) and D2L and IFP showed the number of genes expressed in each microarray were as follows : in D12P with D2L (Figure 10), 1129 genes were expressed only in D12P, 1942 genes were expressed only in D2L, 1000 genes were commonly expressed, of which 91 genes were more than 2-fold differentially expressed. A comparison of D12P with IFP (Figure 11) showed 516 genes were expressed only in D12P, 67 genes were expressed only in IFP, 679 genes were commonly expressed, of which 214 genes were more than 2-fold differentially expressed. A comparison of D2L with IFP (Figure 12) showed 479 genes were expressed only in D2L, 74 genes were expressed only in IFP, 587 genes were commonly expressed, of which, 189 genes were

more than 2-fold differentially expressed.

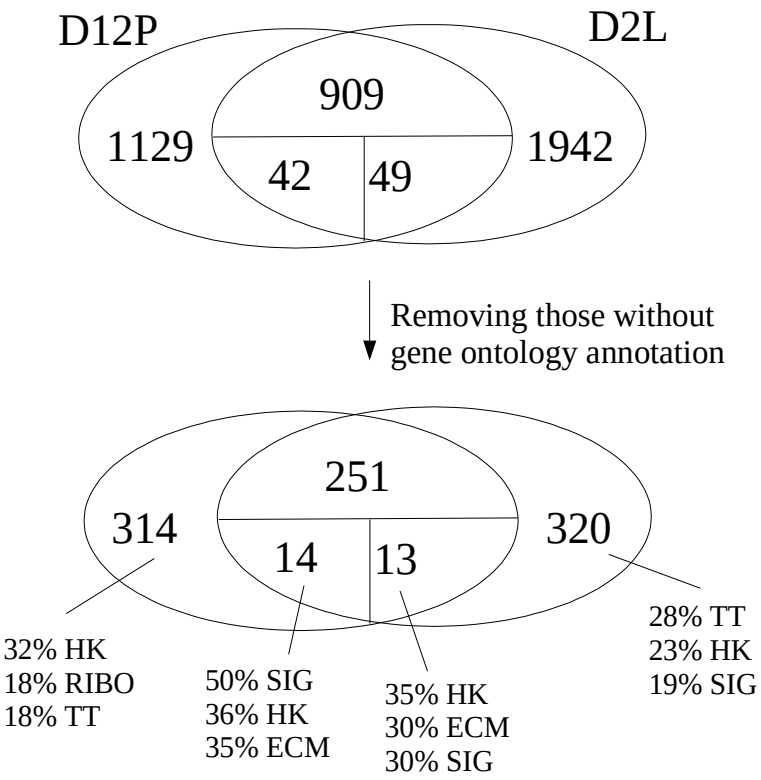

Fig. 10. Distribution of Gene Groups from Microarray Comparison of Day 12 Pregnant and Day 2 Lactation RNA. There were 1129 genes specific to Day 12 Pregnant RNA (D12P), 1942 genes specific to Day 2 Lactation RNA (D2L), of the 1000 commonly expressed genes, 909 showed no differential expression, 42 genes were expressed by 2-fold or more in D12P compared to D2L and 49 genes were expressed by 2-fold or more in D2L compared to D12P. Considering those genes with annotated gene ontology, 314 genes were specific to D12P, of which 32% were housekeeping genes (HK), 18% were ribosomal proteins (RIBO) 18% were associated with transcription and translation (TT); 320 genes were specific to D2L, of which, 28% were genes associated with transcription and translation (TT), 23% were housekeeping genes (HK), 15% were associated with extracellular matrix (ECM) and 19% were associated with signaling (SIG). 14 genes were expressed at least 2-fold higher in D12P, of which 50% were signaling molecules (SIG), 36% were housekeeping genes (HK) and 35% were genes associated with the ECM. 13 genes were expressed at least 2-fold higher in D2L, of which 35% were housekeeping genes (HK), 30% were associated with ECM and 30% were signaling molecules (SIG).

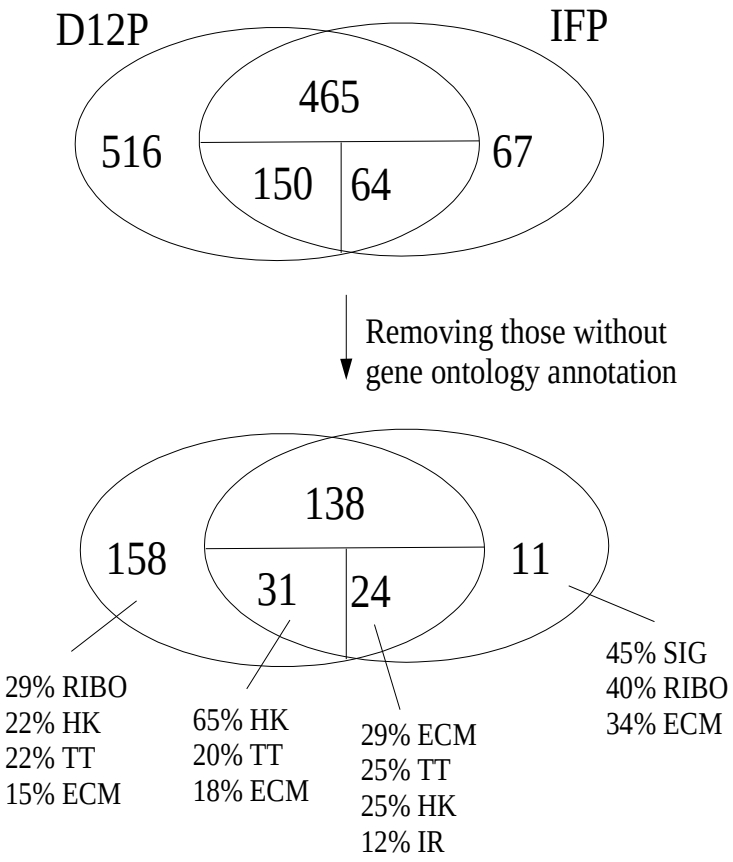

Fig. 11. Distribution of Gene Groups from Microarray Comparison of Day 12 Pregnant RNA and IFP. There were 516 genes specific to Day 12 Pregnant RNA (D12P), 67 genes specific to IFP, of the 679 commonly expressed genes, 465 showed no differential expression, 150 genes were expressed by 2-fold or more in D12P compared to IFP and 64 genes were expressed by 2-fold or more in IFP compared to D12P. Considering those genes with annotated gene ontology, 158 genes were specific to D12P, of which 29% were ribosomal proteins (RIBO), 22% were housekeeping genes (HK), 22% were associated with transcription and translation (TT), 15% were genes associated with the extracellular matrix (ECM); 11 genes were specific to IFP, of which, 45% were associated with signal transduction (SIG), 40% were ribosomal proteins (RIBO) , 34% were associated with extracellular matrix (ECM). 31 genes were expressed at least 2-fold higher in D12P, of which 65% were housekeeping genes (HK), 20% were associated with transcription and translation (TT) and 18% were genes associated with the ECM. 24 genes were expressed at least 2-fold higher in IFP, of which 29% were associated with ECM, 25% were associated with transcription and translation (TT), 25% were housekeeping genes (HK), and 12% were associated with inflammatory responses (IR).

Further examination of the comparison of D2L with IFP was warranted since this study represented the direct comparison between an *in vitro* and an *in vivo* system and was a part of the project.

Comparing IFP and D2L, of the 189 genes showing differential expression, 81 genes were up-regulated in IFP whereas 108 genes were up-regulated in D2L. Considering only genes annotated in gene ontologies (Figure 12), 17 genes were specific to IFP, 110 genes were specific to D2L, 123 genes were not differentially expressed and 50 genes were differentially expressed.

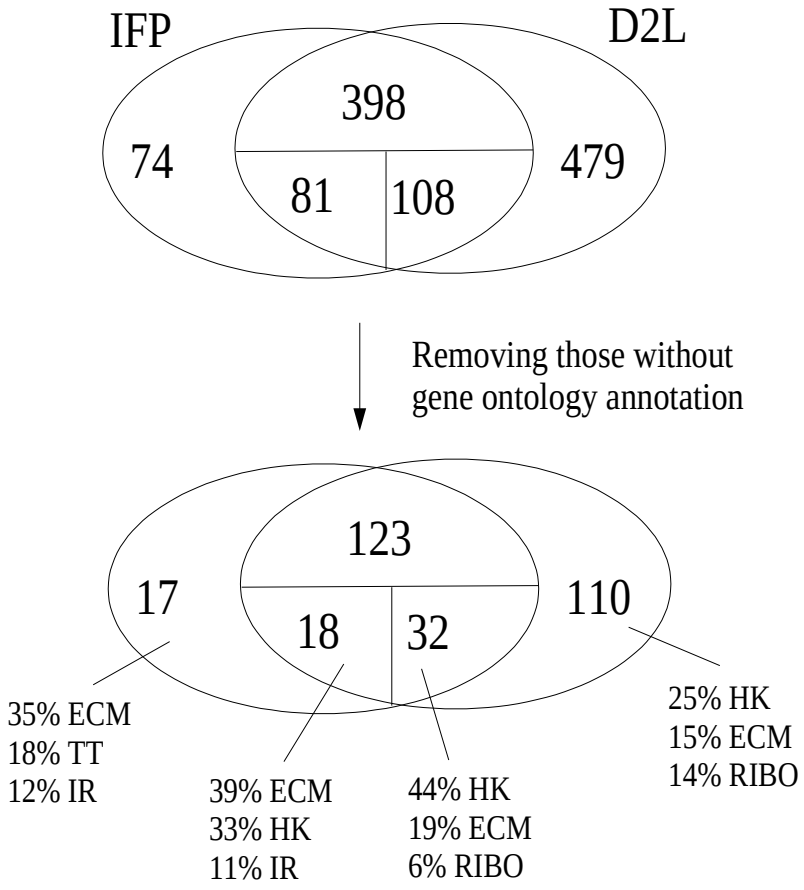

Fig. 12. Distribution of Gene Groups from Microarray Comparison of IFP and Day 2 Lactation RNA. There were 74 genes specific to IFP, 479 genes specific to Day 2 Lactation RNA (D2L), of the 587 commonly expressed genes, 398 showed no differential expression, 81 genes were expressed by 2-fold or more in IFP compared to D2L and 108 genes were expressed by 2-fold or more in D2L compared to IFP. Considering those genes with annotated gene ontology, 17 genes were specific to IFP, of which 35% were associated with extracellular matrix (ECM), 18% were associated with transcription and translation (TT), 12% were associated with inflammation responses (IR); 110 genes were specific to D2L, of which, 25% were housekeeping genes (HK), 15% were associated with ECM and 14% were ribosomal proteins (RIBO). 18 genes were expressed at least 2-fold higher in IFP, of which 39% were associated with ECM, 33% were housekeeping genes (HK) and 11% were associated with inflammatory responses. 32 genes were expressed at least 2-fold higher in D2L, of which 44% were housekeeping genes (HK), 19% were associated with ECM and 6% were ribosomal proteins (RIBO).

Based on the number of genes detected with intensity above background, it was clear that different number of expressed genes can be detected in replicates of the same treatment. For example, using D12P as example, 2129 genes (1000 common expression and 1129 specific to D12P) were detected on the D12P compared with D2L slide whereas 1195 genes were detected on the D12P compared with IFP slide. This was largely due to different levels of background on each slide, which had been shown to affect reproducibility (Kim et al., 2002; Nagarajan, 2003; Wildsmith et al., 2001), as as many as 93.4% of the genes were within two standard deviations from the background as in the case of IFP in the comparison of D12P with IFP.

4.3.2 Observed Trends from Microarray Results

There were six major observations which were identifiable with each of the three samples and was summarized in Table 2 below.

Firstly, there was a decline in the number of ribosomal proteins expressed in Day 2 Lactation (D2L), as compared to Day 12 Pregnant (D12P). The number of ribosomal proteins expressed in IFP was about 10% less than that of D2L. However, 15 genes for amino acid-tRNA synthetases, such as, serine-tRNA synthetase and tyrosine-tRNA synthetase, were present in D2L. Secondly, both D12P and D2L tissue expressed a mixture of growth promoters (12 genes), and inhibitors (10 genes), even though the general expression of growth promoters (5 genes) and inhibitors (6 genes) was lesser in D2L. This was in contrast to IFP, where there was more expression of oncogenic growth promotors, such as adenomatose polypopsis coli (APC). Thirdly, 4 inflammatory response genes, such as, complement activation factor and T-cell activation factor, were expressed only in IFP. Fourthly, D12P tissue demonstrated an emphasized on intracellular vesicle transport by expressing genes, such as, constitutive photomorphogenic (COP) and kinesin complexes, whereas D2L emphasized on extracellular export by expressing genes, such as, clathrin and topomyosins. However, adaptor protein-3 (AP3) was absent in IFP. Fifthly, 5 genes associated with the extracellular matrix (ECM) were expressed in D12P, as compared to 8 genes and 2 genes in IFP and D2L respectively. ECM proteinase inhibitors were more frequently found in IFP (5 genes) than in D2L (3 genes). Lastly, the profiles of cytoskeleton genes expressed were different for each sample. D2L expressed 6 genes encoding for myosin, whereas

D12P expressed both keratins (5 genes) and myosins (3 genes) . Explants cultured in IFP expressed mainly actin genes (7 genes encoding for actin, 1 gene encoding for myosin).

Table 2. Major Observed Differences from Microarray Results (Culture Verification)

Day 12 Pregnant	***Day 2 Lactation***	***IFP***
Large numbers of genes for ribosomal proteins (52 genes)	40 genes encoding for ribosomal proteins. More amino acid-tRNA synthetases	34 genes encoding for ribosomal proteins.
Controlled growth (mixture of growth promoters and inhibitors)	Dampening of controlled growth (mixture of growth promoters and inhibitors but less than pregnant) and preventing apoptosis	Massive growth promotion via adenomatose polypopsis coli (APC) (oncogenes), little inhibition of growth. Expresses acute phase response factor (APRF)
No inflammatory responses	No inflammation responses	Inflammation responses (complement activation, T-cell activation)
Expresses pre-Golgi trafficking components (clathrin, COP, Sec)	Expresses post-Golgi trafficking components (adaptor proteins, cavolin)	No expression of adaptor proteins (AP3)
Cytoskeletal buildup (keratin, myosin)	More types of myosins than in pregnancy	Mainly actin, with less types of myosin

4.3.3 Discussion

4.3.3.1 Comparing Current Results with Microarray Results of In Vivo Study of the Changes in Gene Expressions from Late Pregnancy to Early Lactation

These results shown for the total number of genes differentially expressed during lactation in the current study were significantly lower than those previously described. The latter study reported approximately 3500 of the 5000 genes tested to be differentially expressed between late pregnancy and early lactation mammary gland (Master et al., 2002). This difference may be partially be explained by the microarray used; Affymetrix microarray are able to measure absolute quantities of RNA whereas EST microarray measures relative quantities of RNA and single channel quantitation is difficult due to problems with background on the array.

4.3.3.2 Damage Responses

Mammary glands were excised from sacrified animals and processed by cutting into small pieces of tissue to culture as mammary explants. These challenges to the tissue

appear to induce an inflammatory response. Inflammatory responses *in vivo* act as a protective response in event of infection and cellular damage by increasing local blood circulation and have a role in mounting an immune response (Krishnadasan et al., 2003) in preparation of invasion of foreign bodies, such as, micro-organisms (Salvemini et al., 1999). After tissue processing, it is likely a substantial number of cells per explant were more damaged when compared to a lactating gland, which is consistent with an increased damage response in the cultured tissues.

4.3.3.3 Growth and Tissue Repair

Tissue damaged can be reversed by debris removal, growth and repair (Dembinski et al., 2000; Dubois-Dalcq and Armstrong, 1990; Michael et al., 2000; Miller and Debas, 1995; Riley et al., 1987; Shimizu et al., 1998). While debris removal is unlikely in cultured tissues, increased expression of genes for factors that promote growth and repair were elevated in cultured explants. The expression of oncogenes, such as, adenomatose polypopsis coli (APC) (Ahmed et al., 1998; Jeanteur, 1998; Venesio et al., 2003), in cultured explants might suggest cell growth and division to replace the damaged tissues. This is supported by the down-regulation of genes for growth inhibitors expressed in mid-pregnancy (programmed cell death 8, growth arrest-specific 7 and protein regulator of cytokinesis 1), indicating an increased growth promotion as compared to mid-pregnancy.

Mammary growth is observed during pregnancy, where the mammary glands grow from a ductal system to a secretory system by parturition (Couldrey et al., 2002). The microarray results showing significant expression of genes for growth promoters and inhibitors, suggesting that growth during pregnancy is well-controlled as observed in other tissue (Smith et al., 2000).

Cultured mammary explants showed an up-regulated expression of genes for actin and myosins, when comparing with the other two mammary samples, suggesting cytoskeletal repair was occurring (Cowin et al., 2003; Danjo and Gipson, 1998; Gordon and Buxar, 1997; Ichijima et al., 1993). Taken together, it may be concluded that during the initial two days in culture, replacement growth and repair is occurring to promote recovery from damage during tissue processing and the 2 days of culture. This is consistent with

prior studies indicating no net growth is observed in explants (Skarda et al., 1982).

4.3.3.4 Extracellular Matrix

The extracellular matrix is known to play an important role in tissue function in mammary glands (Lelievre et al., 1996; Lin and Bissell, 1993; Noel et al., 1994; Ricciardelli et al., 2002; Sudhakaran et al., 1999; Weaver et al., 1996). Results showed that there was increased expression of genes associated with extracellular matrix (ECM) in pregnancy and in mammary explants as compared to that in lactation. The former was expected as most mammary growth occurred during pregnancy (Banerjee and Walker, 1967). The latter may be explained as recovery and tissue repair from the damages sustained during the process of culturing as the synthesis of ECM components (Frisbie et al., 2003; Fu et al., 2001; Marchion et al., 2003; Qi and Scully, 2003) and ECM proteinase inhibitors (Holopainen et al., 2003; Tatekawa et al., 2003) was observed in tissue repair. The expression of genes for ECM proteinase inhibitors might be a means to prevent additional tissue damage in the mammary explant culture (Moorehead et al., 2001; Schwertfeger et al., 2001).

4.3.3.5 Protein Synthesis and Secretion

Even though there is a down-regulation of the genes for ribosomal proteins in lactation compared to pregnancy, there is a corresponding up-regulation of amino acid-tRNA synthetases. These is consistent with a change in cell function, from growth and maintenance towards protein production. This might suggest that the length of lactation is somewhat determined at its initiation, as a down-regulation of ribosomal protein synthesis may result in limiting the pool of ribosomes. Towards the end of lactation, the pool of ribosomes then, might not be sufficient to maintain lactation. However, this concept remains to be evaluated.

Although cultured explants express genes for ribosomal proteins at similar levels as in lactating tissues, cultured explants seemed to be deficient in export (Forsyth and Turvey, 1984) from the secretory cells as important components like adaptor proteins, such as, AP3 (Rodionov et al., 2002), seems to be lacking. However, the extend of secretory inhibition in the mammary explants resulting from the lack of AP3 remains unknown.

4.3.3.6 Main Differences Between Mammary Explant Culture and In Vivo Model

These studies have addressed the question of whether lactogenesis in the mammary explant culture model is representative of lactogenesis *in vivo* in the mammary glands. There is no significant differences in the genes expressed in either mammary explants or the mammary gland to suggest that the explant culture is not a viable model to examine to role of insulin, prolactin and glucocorticoid. However, it is important to realize that cellular export machinery may be significantly deficient in cultured explants compared to *in vivo* mammary glands. At the same time, from the microarray results obtained, it is clear that the cultured explants had not recovered from the physical damages sustained during tissue processing after two days in culture. This suggests that tissue recovery may take longer than two days, thus, a recovery and adaptation period of no less than two days should be planned for future experiments.

Chapter 5:
Hormone-induced Gene Expression in Mammary Explants

5.1 Characterization of RNA for Microarray Analysis

Total RNA was extracted from mammary explants and analyzed by agarose gel electrophoresis for integrity and spectrophotometric absorbance at 230, 260 and 280nm for purity. Two ribosomal bands of RNA were observed and no low molecular weight smears were evident in any of the samples (Figure 13), indicating an absence of RNA degradation.

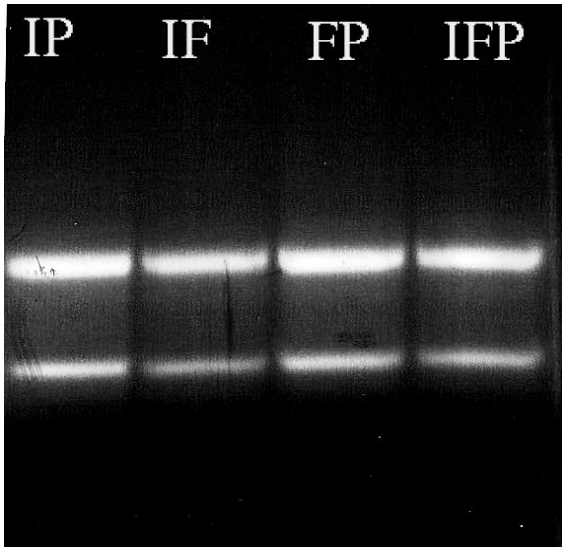

Fig. 13. Agarose Gel Analysis of Extracted RNA for Hormone Treatments Experiment. Total RNA was electrophoresed on a 1% agarose gel and stained with ethidium bromide. IP, IF, FP and IFP represent RNA extracted from explants cultured in IP-, IF-, FP- and IFP-supplemented media for 2 days respectively. The 18S and 28S ribosomal bands are shown.

Table 3. Spectrophotometry of RNA for Hormone Treatments Experiment

Sample	***A260***	***A280***	***A260/A280***	***A230***
IP	1.060	0.507	2.090	0.435
FP	1.055	0.483	2.183	0.478
IF	0.854	0.433	1.971	0.360
IFP	1.064	0.478	2.227	0.482

The A260/A280 of RNA from each treatment groups was above 1.8 (Table 3), indicating that the RNA samples were free from protein contamination. In addition, the absorbance at 230nm was similar to the corresponding reading at 280nm, indicating that the samples were free from phenol contamination and were suitable for microarray hybridization.

5.2 Microarray Analyzes of Mammary Explants Response to Insulin, Glucocorticoid and Prolactin

Three microarray comparisons were carried out in attempt to identify the mammary gland-specific genetic responses to;

[a] the effects of insulin in the presence of prolactin and glucocorticoid;
[b] the effects of glucocorticoid in the presence of insulin and prolactin; and
[c] the effects of prolactin in the presence of insulin and glucocorticoid.

5.2.1 Microarray Results

5.2.1.1 General Overview of Microarray Results

Using the results of those spots on the microarrays that were higher than background intensity, the microarray comparisons IFP and FP, IP and IF respectively showed the number of genes expressed in each microarray were as follows: in IFP with FP, 3126 genes were expressed in IFP and 5441 genes were expressed in FP. A comparison of IFP with IP showed that 3667 genes were expressed in IFP and 3378 genes were expressed in IP. A comparison of IFP and IF showed that 4734 genes were expressed in IFP and 4700 genes were expressed in IF.

Using the results of those spots with intensity higher than two standard deviations from background intensity, an analysis of microarray analysis of explants cultured in either FP or IFP revealed 32 genes that were specific to FP, 266 genes specific to IFP, 474 genes expressed in explants cultured in FP or IFP and 183 genes that were more than 2-fold differentially expressed. Considering only genes annotated in gene ontologies (Figure 14), 1 gene was specific to IFP, 82 genes were specific to FP, 85 genes were not differentially expressed and within this group, 56 genes were differentially expressed.

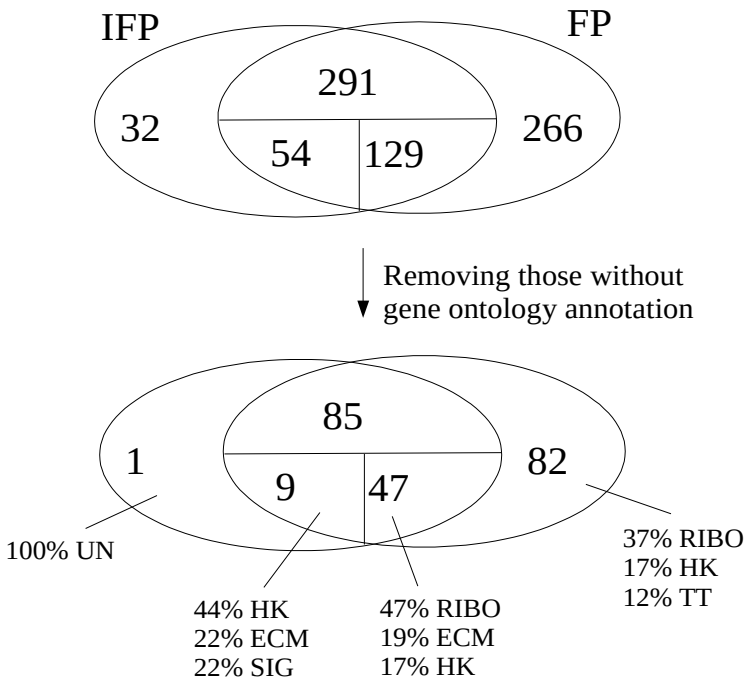

Fig. 14. Distribution of Gene Groups from Microarray Comparison of IFP and FP. There were 32 gene specific to IFP, 266 genes specific to FP, of the 475 commonly expressed genes, 291 showed no differential expression, 54 genes were expressed by 2-fold or more in IFP compared to FP and 129 genes were expressed by 2-fold or more in FP compared to IFP. Considering those genes with annotated gene ontology, 1 genes of unknown function (UN) was specific to IFP; 82 genes were specific to FP, of which, 37% were ribosomal proteins (RIBO), 17% were housekeeping genes (HK) and 12% were associated with transcription and translation (TT). 9 genes were expressed at least 2-fold higher in IFP, of which, 44% were housekeeping genes (HK), 22% were associated with the extracellular matrix (ECM) and 22% were associated with signaling (SIG). 47 genes were expressed at least 2-fold higher in FP, of which 47% were ribosomal proteins (RIBO), 19% were associated with ECM and 44% were housekeeping genes (HK).

A comparison of explants cultured with IF compared to IFP showed 1 gene was specific to IF, 550 genes were specific to IFP, 531 genes were commonly expressed and within the latter group, 60 genes were more than 2-fold differentially expressed. Considering only genes annotated in gene ontologies (Figure 15), 85 genes were specific to IFP, no genes were specific to IF, 79 genes were not differentially expressed and 12 genes were differentially expressed.

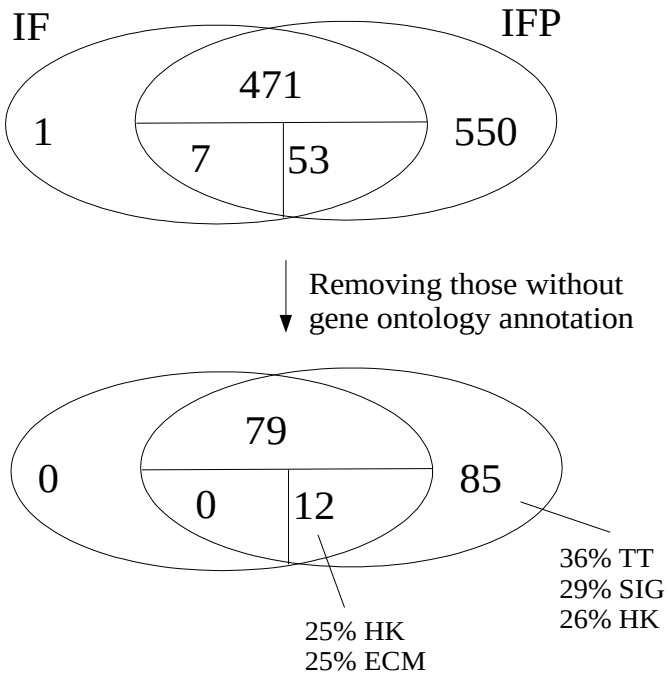

Fig. 15. Distribution of Gene Groups from Microarray Comparison of IFP and IF. There were 550 genes specific to IFP, 1 gene specific to IF, of the 531 commonly expressed genes, 471 showed no differential expression, 53 genes were expressed by 2-fold or more in IFP compared to IF and 7 genes were expressed by 2-fold or more in IF compared to IFP. Considering those genes with annotated gene ontology, 85 genes were specific to IFP, of which 36% were associated with transcription and translation (TT), 29% were involved in signaling (SIG) and 26% were housekeeping genes (HK); no gene was specific to IF. 12 genes were expressed at least 2-fold higher in IFP, of which 25% were associated with ECM and 25% were housekeeping genes (HK).

A comparison of explants cultured with IP compared to IFP showed 86 genes were specific to IP, 385 genes were specific to IFP, 358 genes were commonly expressed and within this group, 147 genes were more than 2-fold differentially expressed. Considering only genes annotated in gene ontologies (Figure 16), 63 genes were specific to IFP, 16 genes were specific to IP, 48 genes were not differentially expressed and 26 genes were differentially expressed.

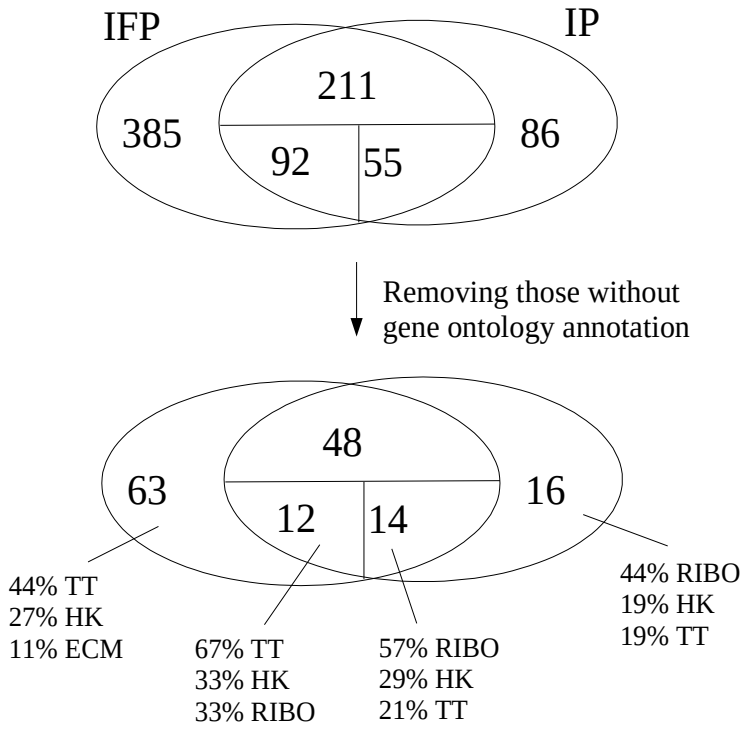

Fig. 16. Distribution of Gene Groups from Microarray Comparison of IFP and IP. There were 385 genes specific to IFP, 86 genes specific IP, of the 358 commonly expressed genes, 211 showed no differential expression, 92 genes were expressed by 2-fold or more in IFP compared to IP and 55 genes were expressed 2-fold or more in IP compared to IFP. Considering those genes with annotated gene ontology, 63 genes were specific to IFP, of which 44% were associated with transcription and translation (TT), 27% were housekeeping genes (HK) and 11% were associated with extracellular matrix (ECM); 16 genes were specific to IP, of which, 44% were ribosomal proteins (RIBO), 19% were housekeeping genes (HK) and 19% were associated transcription and translation (TT). 12 genes were expressed at least 2-fold higher in IFP, of which 67% were associated with transcription and translation (TT), 33% were housekeeping genes (HK) and 33% were ribosomal proteins (RIBO). 14 genes were expressed at least 2-fold higher in IP, of which 57% were ribosomal proteins (RIBO), 29% were housekeeping genes (HK) and 21% were associated with transcription and translation (TT) .

The number of genes found to be expressed in explants cultured in IFP were different in each of the 3 treatments. This was very likely due to varying levels of background between each slides (Kim et al., 2002; Nagarajan, 2003; Wildsmith et al., 2001), as as many as 86.9% of the genes were within two standard deviations from the background as in the case of IP in the comparison of IFP and IP.

5.2.1.2 Results Suggesting the Effects of Insulin in the Presence of Prolactin and Glucocorticoid

Comparing the transcriptome of mammary explants cultured in either FP or IFP for 2 days using microarray, 3 main trends could be observed. Firstly, there was a decrease in the expression 11 genes (see Appendix G) for ribosomal proteins (3.2 to 14.7-fold) and

secondly, an increase in 7 genes (see Appendix G) encoding for mitochondrial proteins (2.1 to 7.3-fold). Lastly, there was expression of a gene encoding for placental lactogen, adaptor protein-3 (AP3) and alpha-integrin in explants cultured only in the presence of insulin.

5.2.1.3 Results Suggesting the Effects of Glucocorticoid in the Presence of Insulin and Prolactin

An examination of gene expression in mammary explants cultured in either IP or IFP for 2 days showed 3 major trends. Firstly, 13 genes (see Appendix G) encoding for glycosylation-related proteins, such as, dolicol-phosphate-mannosyltransferase-2 and man-6-phosphate receptor, were elevated in the presence of cortisol. Secondly, there was an upregulation of chaperones (6 genes; see Appendix G), ubiquitin and translation repressor (NAT1). Lastly, the expressions of the genes for fibronectin, KDEL receptor (a receptor binding to a 4 amino acid sequence, Lys-Asp-Glu-Leu, at the carboxyl terminus) and erythroid differentiation regulator were specific to glucocorticoid.

5.2.1.4 Results Suggesting the Effects of Prolactin in the Presence of Insulin and Glucocorticoid

The gene expression profiles of mammary explants cultured in either IF or IFP for 2 days, showed that there was an increase in proteins in the MAP kinase pathway (7 genes see Appendix G), ATPases (5 genes; see Appendix G), transcription factors (10 genes; see Appendix G) and translation factors (7 genes; see Appendix G) in IFP when compared to IF. The expression of genes for cell membrane receptors, such as, angiotensin receptor, sterol receptors, LDL receptors and 5-hydroxytryptamine-3 receptor were specific to IFP cultures. At the same time, a number of growth promoters, and growth inhibitors (MA-3 and RBBP5) were expressed in IFP cultures.

5.2.2 Discussion of Microarray Results

5.2.2.1 Using Known Insulin- or Prolactin- or Glucocorticoid-Inducible Genes to Validate Microarray Results

Using analysis of genes that were both known to be inducible by specific hormones and present on the microarray would present a means of validating results from microarray

experiments. Insulin was known to induce myc (Jonas et al., 2001), phosphoinositol-3-kinase (Andreelli et al., 2000) and calmodulin (Solomon et al., 1995). These genes were found to be expressed in insulin-treated explants, namely, IP, IF and IFP. Glucocorticoids were known to regulate the ubiquitin-proteosome pathway (Chai et al., 2002a; Willoughby et al., 2003) and the microarray results demonstrated expression of components of the ubiquitin-proteosome pathway, such as, ubiquitin-conjugation enzyme, in all glucocorticoid-treated explants. Prolactin-treated explants expressed Janus kinase 2 (Jahn et al., 1997) and a range of transcription and translation factors, which is consistent with current literature attributing a role to this hormone in increasing protein synthesis in the mammary gland (Houdebine, 1983).

5.2.2.2 Possible Mammary-Specific Effects of Insulin in the Presence of Prolactin and Glucocorticoid

Insulin seemed to be down-regulating gene expression as there were 740 genes expressed in explants cultured in FP compared to 506 genes expressed in explants in IFP. At the same time, insulin appeared to down-regulate ribosomal protein genes and up-regulate mitochrondrial protein genes. This may suggest that a role for insulin is to promote a shift of emphasis from developing the cells for lactogenesis in pregnancy to the process of lactation post-partum. Taking into account the ubiquitous nature of insulin (O'Hare, 1988), it is likely that this is a prolactin-enabled mammary-specific effect of insulin as it was found that genes for ribosomal proteins were up-regulated in mid-pregnancy (Master et al., 2002).

The expression of placental lactogen or a placental-lactogen-like gene was surprising as it was generally known to be secreted from the placenta (Belanger et al., 1971; Friesen et al., 1969; Gusdon and Yen, 1967), and may have a role in mammary development as placental lactogen was found to exhibit autocrine effect in testicles (Untergasser et al., 1996), which might suggest the gene is more widely expressed, including in mammary glands and may have a similar autocrine function. It is interesting that the cow continues to lactate after the administration of bromocriptine, an ergot alkaloid used to inhibit the secretion of prolactin from the pituitary (Mills et al., 1989). It is conceivable that the secretion of placental lactogen in this species is capable of supporting lactation. However, this concept remains to be evaluated. Insulin is ubiquitous in circulation and it

is unlikely that the gene for placental lactogen is expressed in mammary tissues from non-pregnant animals but this requires further validation. Thus, further study is needed to ascertain the role and temporal expression of the gene for placental lactogen in mammary tissues, and to demonstrate the protein is synthesized and secreted.

The AP3 protein is an important component for secretory pathway from Golgi to cell membrane (Rodionov et al., 2002), whereas alpha-integrin is part of integrin complex for cell-ECM attachment (Hynes et al., 1989). From the results obtained in the current study, it seemed that these two proteins were specific to insulin-treated mammary glands. Insulin results in the expression of glucose receptors, such as, GLUT4 (Ezaki et al., 1993), on the cell membrane, by a "secretory" process (Hudson et al., 1992; Marette et al., 1992; Robinson et al., 1992; Yang et al., 1992), and glucose from the circulation is taken into the cell with the endocytosis of glucose-bounded GLUT4 receptor (Czech, 1995). The uptake of glucose is critical for the synthesis of lactose (Chaiyabutr et al., 1980), the major osmole in milk (Bleck and Bremel, 1993). Hence, insulin-specific AP3 gene expression may be explained as a trigger for expressing insulin-induced receptors, such as, GLUT4, on the cell surface as insulin-induced adaptor proteins had been previously suggested (Welsh et al., 1999). In the context, insulin might be an important trigger for milk secretion.

In the experiment to assess the mammary explant culture model, it was noted that AP3 was not expressed in the presence of IFP, which is in contradiction to the results presented here. This might be due to tissue damage which is still evident after two days in culture with IFP, as described in the last chapter. It was concluded that mammary explants are likely to require more than two days for the tissue to recover. The subsequent experiment had showed that eight days in culture is presumely sufficient for the explants to heal as no inflammatory response is observed and AP3 is induced.

5.2.2.3 Possible Mammary-Specific Effects of Glucocorticoid in the Presence of Insulin and Prolactin

It appears that the mammary-specific effects of glucocorticoid were primarily targetted towards protein translation and glycosylation of other proteins. Upregulation of chaperones (DeFranco et al., 1998; Gromov and Celis, 1991; Ryan et al., 1997) and the

glycosylation pathway (Parodi, 2000a; Parodi, 2000b; Wenzel-Seifert and Seifert, 2003; Wheatley and Hawtin, 1999; Wormald and Dwek, 1999) seemed to be acting in synchrony to ensure that proteins being produced are correctly folded into active forms. This is not surprising considering the amount of proteins being produced by the mammary gland during lactation as it had been demonstrated that the levels of casein mRNA is increased by as much as 12-fold in early lactation compared to pregnancy (Nakhasi and Quasba, 1979; Rosen and Barker, 1976). In the whole animal, glucocorticoid functions as an energy regulator (Parodi, 2000a; Parodi, 2000b; Wenzel-Seifert and Seifert, 2003; Wheatley and Hawtin, 1999; Wormald and Dwek, 1999), and act as an agonist to insulin (Steffensen and Gustafsson, 2004). Therefore, glucocorticoid acts as a "low-energy resource" signal in other tissues (non-mammary tissues), such as, liver, and activates gluconeogenesis (Holmes et al., 2001). However, this effect of glucocorticoid was not observed in mammary tissues. KDEL (Lys-Asp-Glu-Leu) receptor (Munro and Pelham, 1987) is found in the endoplasmic reticulum and is known to play a role in protein sorting (Ceriotti and Colman, 1988; Pelham, 1988; Pelham et al., 1988) and other post-translational processing (Majoul et al., 2001; Yamamoto et al., 2003).

Glucocorticoid appears to have a role expressing ubiquitin to increase the rate of protein breakdown by ubiquitin-proteosome pathway should the need arise due to any diminished capacity to secrete all the milk proteins produced (Chai et al., 2002b; Chai et al., 2003; Melman et al., 2002). In the event of incorrectly folded proteins, this pathway will be active (Luzikov, 2002). The expression of the gene for NAT1 (a translation repressor) may suggest that glucocorticoid has a role in regulating the rate of protein synthesis as NAT1 is known to limit translation (Yamanaka et al., 1997). Taken together, glucocorticoid seems to have a potential role in increasing the efficiency of post-translational processing in the mammary cell.

Glucocorticoid is known to induce the expression of the gene for erythroid differentiation regulator (Wessely et al., 1997) and is known to affect vascular formation (Dickson et al., 1995). This suggest that glucocorticoid might have a role in the increased rate of vascular formation within the mammary gland that occurs during lactation (Mepham, 1982; Prosser et al., 1996; Yasugi et al., 1989).

Fibronectin is an important component for cell to extracellular-matrix attachment (Carter, 1982; Carvalho et al., 1998; Stanislawski et al., 1985; Steele et al., 1991; Weinacker et al., 1994) and glucocorticoid-induced expression of the gene for fibronectin (Armelin and Armelin, 1983; Oliver et al., 1983; Zhou et al., 1998) seems to further support the idea of regulating complex processes by regulating key components as noted in the case of insulin-induced alpha-integrin.

5.2.2.4 Possible Mammary-Specific Effects of Prolactin in the Presence of Insulin and Glucocorticoid

As the name implies, prolactin is acknowledged as a crucial trigger of lactation, during parturition (Gow et al., 1983; Hart, 1974). Hence, it is likely that prolactin will elicit more cellular responses than either insulin or glucocorticoid. The essential role of prolactin in lactation is well-described in many mammals (Beaton et al., 2003). Therefore, it was not surprising that a large number of transcription and translation factors were highly expressed as a result of prolactin stimulation.

From the results obtained, it suggested that prolactin plays an important role in enhancing insulin signaling by increasing the amount of proteins in the mitogen-activated protein (MAP) kinase pathway, in addition to previously described roles in enhancing insulin signaling (Belkowski et al., 1999; Buckley et al., 1994; Das and Vonderhaar, 1996; Farrelly et al., 1999; Gao and Horseman, 1999; Piccoletti et al., 1994), such as, the MAP kinase pathway. Taken together, the mammary-specific actions of insulin may be potentially enabled by prolactin.

Prolactin potentially may affect the responses of mammary gland to other endocrine stimuli by up-regulating expression of genes for receptors on the epithelial cell surface. From the results, it was noted that the genes for angiotensin receptor and 5-hydroxytryptamine-3 receptor were expressed in explants following prolactin stimulation. Angiotensin has been shown to enhance blood supply to the lactating mammary gland (Gorewit et al., 1993; Prosser et al., 1996) which is consistent with an increase need for nutrient supply to the lactating mammary gland. Although the direct effects of angiotensin and 5-hydroxytryptamine-3 on mammary tissues have not been

examined, the current study suggested that there are probably a number of other hormones or hormone-like molecules which could affect lactation, which requires further studies.

Prolactin seemed to result in the expression of the gene for sterol receptor and LDL receptor in the mammary explants. This may augment nutrient uptake to meet energy demands of the lactating gland, as prolactin-enhanced calcium uptake had been previously demonstrated (Krishnamra and Taweerathitam, 1995). This was supported by prolactin-specific expression of ATPases, as seen in the results. ATPases are enzymes found in the inner mitochondria membrane (Bucheler et al., 1991; di Jeso, 1978; Ueno et al., 1984) and are critical in producing adenosine triphosphate (ATP) within the cell (Charnock et al., 1971; Kalasauskaite and Grinius, 1979; Tsuchiya and Rosen, 1975).

In summary, although prolactin directly acted on a number of cellular processes, it also seemed to act upstream to a number of mammary-specific processes, thus, potentially altering the responsiveness of the mammary gland to other hormones. However, further studies are necessary to examine the effects of prolactin on mammary glands, especially in the area of prolactin-enabled hormonal effects of hormones, such as, angiotensin, which do not have an identified role in the mammary gland.

5.3 Proposed Endocrine Model of Murine Lactogenesis

The current study suggested that prolactin action on mammary tissues enhances insulin signaling, which may result in the expression of placental lactogen and which may in turn exhibit autocrine effects on mammary tissues. Prolactin acts in combination with glucocorticoid to halt growth and differentiation of mammary glands and initiate lactation. Expression of translational machinery decreased at this point to divert resources into lactation, which might be a reason that the length of lactation cannot be extended indefinitely in most mammals. The turn over of this translational machinery, in the mammary epithelial cells, needs to be validated in future studies.

Milk secretion could be explained as the results of massive prolactin-induced protein expression by expression of an avalanche of transcription and translation factors. As large quantities of proteins were being produced, glucocorticoid enhances chaperones

and the glycosylation pathway to ensure correct folding of proteins (Majoul et al., 2001; Yamamoto et al., 2003), which is also supported by the results in this study. After glycosylation is completed, secretion is enhanced by the effects of insulin (Hudson et al., 1992; Marette et al., 1992; Robinson et al., 1992; Welsh et al., 1999; Yang et al., 1992).

As lactation is a highly energetically-demanding task for extended periods of time, LDL, sterols and glucose are needed in large quantities to meet energy demands, as prolactin had been shown to increase nutrient uptake (Krishnamra and Taweerathitam, 1995). At the mammary gland level, LDL and sterol uptake are enhanced by prolactin, whereas glucose uptake is enhance by insulin.

5.4 Conclusion and Future Directions

Lactation is a critical process directed to meet all of the nutritional requirements of an offspring from birth to weaning. Over the last 40 years, both the mammary explant culture system and *in vivo* system had been employed to study lactogenesis, and the knowledge gained formed the basis of this project. Recent advances in functional genomics has provided tools, such as, microarray technology, to study the expression of thousands of genes (even the entire transcriptome) in a single experiment. This has provided the opportunity to gain deeper insights into scope of gene expression at lactogenesis (Master et al., 2002) and to use *in vitro* models to more closely examine the endocrine contribution to this process. However, there are no reports examining the validity of culture systems, such as, mammary explants, to study the nature of hormonal stimuli which exist *in vivo*.

The current study demonstrates the first known attempt to use microarray technology to examine the validity of the mammary explant model and to use it to study complex hormonal stimuli of insulin, prolactin and glucocorticoid in lactogenesis. This study presents the preliminary results suggesting that mammary explants are valid and useful in the study of hormonal regulation at lactogenesis and provides both documented and new insights into the mammary-specific and general actions of insulin, prolactin and glucocorticoid. This investigation clearly demonstrates that the individual contribution of insulin, prolactin and glucocorticoid at lactogenesis can be delineated using mammary explant culture and microarray technology.

Using the preliminary information presented here, a more in depth study using a recently developed 34000 murine gene Affymetrix microarray, which is able to provide quantitative gene expression data (Suenaga et al., 2004), may be used to further elucidate the effects of insulin, prolactin and glucocorticoid at lactogenesis by examining the expressions of all the genes in the murine genome. These data may then be used to map the metabolic and synthetic cellular pathways controlled by these 3 hormones.

References

(In the format of Development Biology)

Accorsi, P. A., Pacioni, B., Pezzi, C., Forni, M., Flint, D. J., and Seren, E. (2002). Role of prolactin, growth hormone, and insulin-like growth factor 1 in mammary gland involution in the dairy cow. *J Dairy Sci* **85,** 507-513.

Ahmed, Y., Hayashi, S., Levine, A., and Wieschaus, E. (1998). Regulation of armadillo by a Drosophila APC inhibits neuronal apoptosis during retinal development. *Cell* **93,** 1171-82.

Ahn, W. S., Bae, S. M., Lee, J. M., Namkoong, S. E., Han, S. J., Cho, Y. L., Nam, G. H., Seo, J. S., Kim, C. K., and Kim, Y. W. (2004). Searching for pathogenic gene functions to cervical cancer. *Gynecol Oncol* **93,** 41-8.

Ahonen, T. J., Harkonen, P. L., Rui, H., and Nevaleinen, M. T. (2002). PRL Signal Transduction in the Epithelial Compartment of Rat Prostate Maintained as Long-Term Organ Cultures in Vitro. *Endocrinol* **143,** 228Ð238.

Andreelli, F., Laville, M., Vega, N., Riou, J. P., and Vidal, H. (2000). Regulation of gene expression during severe caloric restriction: lack of induction of p85 alpha phosphatidylinositol 3-kinase mRNA in skeletal muscle of patients with type II (non-insulin-dependent) diabetes mellitus. *Diabetologia* **43,** 356-63.

Armelin, M. C., and Armelin, H. A. (1983). Glucocorticoid hormone modulation of both cell surface and cytoskeleton related to growth control of rat glioma cells. *J Cell Biol* **97,** 459-65.

Baik, M. G., Lee, M. J., and Choi, Y. J. (1998). Gene expression during involution of mammary gland (review). *Int J Mol Med* **2,** 39-44.

Baldi, P., and Long, A. D. (2001). A Bayesian framework for the analysis of microarray expression data: regularized t-test and statistical inference of gene changes. *Bioinformatics* **17**.

Baldwin, R. L., and Louis, S. (1975). Hormonal actions on mammary metabolism. *J Dairy Sci* **58,** 1033-41.

Ball, S. M. (1998). The development of the terminal end bud in the prepubertal-pubertal mouse mammary gland. *Anat Rec* **250,** 459-64.

Banerjee, M. R., and Walker, R. J. (1967). Variable duration of DNA synthesis in mammary gland cells during pregnancy and lactation of C3H/He mouse. *J Cell Physiol* **69,** 133-42.

Beaton, A., Broadhurst, M. K., Wilkins, R. J., and Wheeler, T. T. (2003). Suppression of beta-casein gene expression by inhibition of protein synthesis in mouse mammary epithelial cells is associated with stimulation of NF-kappaB activity and blockage of prolactin-Stat5 signaling. *Cell Tissue Res* **311,** 207-15.

Belanger, C., Shome, B., Friesen, H., and Myers, R. E. (1971). Studies of the secretion of monkey placental lactogen. *J Clin Invest* **50,** 2660-7.

Belkowski, S. M., Levine, J. E., and Prystowsky, M. B. (1999). Requirement of PI3-kinase activity for the nuclear transport of prolactin in cloned murine T lymphocytes. *J Neuroimmunol* **94,** 40-7.

Bequette, B. J., Kyle, C. E., Crompton, L. A., Anderson, S. E., and Hanigan, M. D. (2002). Protein metabolism in lactating goats subjected to the insulin clamp. *J Dairy Sci* **85,** 1546-1555.

Bequette, B. J., Kyle, C. E., Crompton, L. A., Buchan, V., and Hanigan, M. D. (2001). Insulin regulates milk production and mammary gland and hind-leg amino acid fluxes and blood flow in lactating goats. *J Dairy Sci* **84,** 241-255.

Bleck, G. T., and Bremel, R. D. (1993). Sequence and single-base polymorphisms of the bovine alpha-lactalbumin 5'-flanking region. *Gene* **126,** 213-8.

Bohnet, H. G., Gomez, F., and Friesen, H. G. (1977). Prolactin and estrogen binding sites in the mammary gland of lactating and non-lactating rat. *Endocrinol* **101,** 1111-1121.

Bolander, F. F. J., Nicholas, K. R., and Topper, Y. J. (1979). Retention of glucocorticoid by isolated mammary tissue may complicate interpretation of results from *in vitro* experiments. *Biochem Biophys Res Commun* **91,** 247-252.

Bole-Feysot, C., Goffin, V., Edery, M., Binart, N., and Kelly, P. A. (1998). Prolactin (PRL) and its receptor: actions, signal transduction pathways and phenotypes observed in PRL receptor knockout mice. *Endocr Rev* **19,** 225-68.

Brisken, C., Kaur, S., Chavarria, T. E., Binart, N., Sutherland, R. L., Weinberg, R. A., Kelly, P. A., and Ormandy, C. J. (1999). Prolactin controls mammary gland development via direct and indirect mechanisms. *Dev Biol* **210,** 96-106.

Broberg, P. (2003). Statistical methods for ranking differentially expressed genes. *Genome Biol* **4,** R41.

Brown, C. S., GOodwin, P. C., and Sorger, P. K. (2001). Image metrics in the statistical analysis of DNA microarray data. *Proc Natl Acad Sci U S A* **98,** 8944-8949.

Brown, M. P. S., Grundy, W. N., Lin, D., Sugnet, C., M. Ares, J., and Haussler, D. (2000). Knowledge-based analysis of microarray gene expression data by using support vector machines. *Proc Natl Acad Sci U S A* **97,** 262-267.

Buckley, A. R., Rao, Y. P., Buckley, D. J., and Gout, P. W. (1994). Prolactin-induced phosphorylation and nuclear translocation of MAP kinase in Nb2 lymphoma cells. *Biochem Biophys Res Commun* **204,** 1158-64.

Butler, S. T., Marr, A. L., Pelton, S. H., Radcliff, R. P., Lucy, M. C., and Butler, W. R. (2003). Insulin restores GH responsiveness during lactation-induced negative energy balance in dairy cattles: effects on expression of IGF-1 and GH receptor 1A. *Journal of Endocrinol* **176,** 205-217.

Butte, A. J., Ye, J., Niederfellner, G., Rett, K., Haring, H. U., White, M. F., and Kohane, I. S. (2001). Determining significant fold differences in gene expression analysis. *Proceedings of the Pacific Symposium of Biocomputing* **6,** 6-17.

Carter, W. G. (1982). The cooperative role of the transformation-sensitive glycoproteins, GP140 and fibronectin, in cell attachment and spreading. *J Biol Chem* **257,** 3249-57.

Carvalho, R. S., Schaffer, J. L., and Gerstenfeld, L. C. (1998). Osteoblasts induce osteopontin expression in response to attachment on fibronectin: demonstration of a common role for integrin receptors in the signal transduction processes of cell attachment and mechanical stimulation. *J Cell Biochem* **70,** 376-90.

Ceriotti, A., and Colman, A. (1988). Binding to membrane proteins within the endoplasmic reticulum cannot explain the retention of the glucose-regulated protein GRP78 in Xenopus oocytes. *Embo J* **7,** 633-8.

Chai, J., Shen, C., and Sheng, Z. (2002a). Changes in skeletal muscle protein metabolism in burned rats with sepsis and the role of glucocorticoid in skeletal muscle proteolysis. *Zhonghua Wai Ke Za Zhi* **40,** 705-8.

Chai, J., Wu, Y., and Sheng, Z. (2002b). The relationship between skeletal muscle proteolysis and ubiquitin-proteasome proteolytic pathway in burned rats. *Burns* **28,** 527-33.

Chai, J., Wu, Y., and Sheng, Z. Z. (2003). Role of ubiquitin-proteasome pathway in skeletal muscle wasting in rats with endotoxemia. *Crit Care Med* **31,** 1802-7.

Chaiyabutr, N., Faulkner, A., and Peaker, M. (1980). The utilization of glucose for the synthesis of milk components in the fed and starved lactating goat in vivo. *Biochem J* **186,** 301-8.

Chapman, S., Schenk, P., Kazan, K., and Manners, J. (2002). Using biplots to interpret gene expression patterns in plants. *Bioinformatics* **18,** 202-204.

Charnock, J. S., Cook, D. A., and Casey, R. (1971). The role of cations and other factors on the apparent energy of activation of (Na + + K +)-ATPase. *Arch Biochem Biophys* **147**, 323-9.

Chen, Y., Kamat, V., Dougherty, E. R., Bittner, M. L., Meltzer, P. S., and Trent, J. M. (2002). Ratio statistics of gene expression level and applications to microarray data analysis. *Bioinformatics* **18,** 1207-1215.

Chomczynski, P., Qasba, P., and Topper, Y. J. (1986). Transcriptional and post-transcriptional roles of glucocorticoid in the expression of the rat 25,000 molecular weight casein gene. *Biochem Biophys Res Commun* **134,** 812-818.

Costigan, M., Befort, K., Karchewski, L., Griffin, R. S., D'Urso, D., Allchorne, A., Sitarski, J., Mannion, J. W., Pratt, R. E., and Woolf, C. J. (2002). Replicate high-density rat genome oligonucleotide microarrays reveal hundreds of regulated genes in the dorsal root ganglion after peripheral nerve injury. *BMC Neurosci* **3,** 16.

Couldrey, C., Moitra, J., Vinson, C., Anver, M., Nagashima, K., and Green, J. (2002). Adipose tissue: a vital in vivo role in mammary gland development but not differentiation. *Dev Dyn* **223,** 459-68.

Cowin, A. J., Hatzirodos, N., Teusner, J. T., and Belford, D. A. (2003). Differential

effect of wounding on actin and its associated proteins, paxillin and gelsolin, in fetal skin explants. *J Invest Dermatol* **120,** 1118-29.

Culhane, A. C., Perriere, G., Considine, E. C., Cotter, T. G., and Higgins, D. G. (2002). Between-group analysis of microarray data. *Bioinformatics* **18,** 1600-1608.

Czech, M. P. (1995). Molecular actions of insulin on glucose transport. *Annu Rev Nutr* **15,** 441-71.

Dahlquist, K. D., Salomonis, N., Vranizan, K., Lawlor, S. C., and Conklin, B. R. (2002). GenMAPP, a new tool for viewing and analyzing microarray data on biological pathways. *Nat Genet* **31,** 19-20.

Danjo, Y., and Gipson, I. K. (1998). Actin 'purse string' filaments are anchored by E-cadherin-mediated adherens junctions at the leading edge of the epithelial wound, providing coordinated cell movement. *J Cell Sci* **111 (Pt 22),** 3323-32.

Das, R., and Vonderhaar, B. K. (1996). Activation of raf-1, MEK, and MAP kinase in prolactin responsive mammary cells. *Breast Cancer Res Treat* **40,** 141-9.

DeFranco, D. B., Ramakrishnan, C., and Tang, Y. (1998). Molecular chaperones and subcellular trafficking of steroid receptors. *J Steroid Biochem Mol Biol* **65,** 51-8.

Dembinski, A., Warzecha, Z., Konturek, P. C., Ceranowicz, P., Stachura, J., Tomaszewska, R., and Konturek, S. J. (2000). Epidermal growth factor accelerates pancreatic recovery after caerulein-induced pancreatitis. *Eur J Pharmacol* **398,** 159-68.

Desjardins, C., Paape, M. J., and Tucker, H. A. (1968). Contribution of pregnancy, fetuses, fetal placentas and deciduomas to mammary gland and uterine development. *Endocrinol* **83,** 907-10.

di Jeso, F. (1978). Localization of inner mitochondrial membrane components by specific antibodies [proceedings]. *Arch Int Physiol Biochim* **86**, 856-7.

Dickson, M. C., Martin, J. S., Cousins, F. M., Kulkarni, A. B., Karlsson, S., and Akhurst, R. J. (1995). Defective haematopoiesis and vasculogenesis in transforming growth factor-beta 1 knock out mice. *Development* **121,** 1845-54.

Didier, G., Brezellec, P., Remy, E., and Henaut, A. (2002). Gene-ANOVA - gene expression analysis of variance. *Bioinformatics* **18,** 490-491.

Djiane, J., Clauser, H., and Kelly, P. A. (1979). Rapid down-regulation of prolactin receptors in mammary gland and liver. *Biochemi Biophys Acta* **90,** 1371-1378.

Dubois-Dalcq, M., and Armstrong, R. (1990). The cellular and molecular events of central nervous system remyelination. *Bioessays* **12,** 569-76.

Duggan, D., Bittner, M., Chen, Y., Meltzer, P., and Trent, J. (1999). Expression profiling using cDNA microarrays. *Nat Genet* **21,** 10-14.

Ezaki, O., Flores-Riveros, J. R., Kaestner, K. H., Gearhart, J., and Lane, M. D. (1993). Regulated expression of an insulin-responsive glucose transporter (GLUT4) minigene in 3T3-L1 adipocytes and transgenic mice. *Proc Natl Acad Sci U S A*

90, 3348-52.

Farrelly, N., Lee, Y. J., Oliver, J., Dive, C., and Streuli, C. H. (1999). Extracellular matrix regulates apoptosis in mammary epithelium through a control on insulin signaling. *J Cell Biol* **144,** 1337-48.

Forsyth, I. A., and Turvey, A. (1984) Fatty acid synthesis by explant culture from the mammary glands of goats on days 60 and 120 of pregnancy. *J Endocrinol* **100,** 87-92.

Franklyn F. Bolander, J., Nicholas, K. R., Wyk, J. J. V., and Topper, Y. J. (1981). Insulin is essential for accumulation of casein mRNA in mouse mammary epithelial cells. *Proc Natl Acad Sci U S A* **78,** 5682-5684.

Friesen, H. G., Suwa, S., and Pare, P. (1969). Synthesis and secretion of placental lactogen and other proteins by the placenta. *Recent Prog Horm Res* **25,** 161-205.

Frisbie, D. D., Oxford, J. T., Southwood, L., Trotter, G. W., Rodkey, W. G., Steadman, J. R., Goodnight, J. L., and McIlwraith, C. W. (2003). Early events in cartilage repair after subchondral bone microfracture. *Clin Orthop***,** 215-27.

Fu, J., Xu, Y., and Zhang, Z. (2001). mRNA expression of basic fibroblast growth factor from a single intratracheal instillation of papain-induced emphysema in rats. *J Tongji Med Univ* **21,** 297-300.

Fuhrman, S., Cunningham, M. J., Wen, X., Zweiger, G., Seihamer, J. J., and Somogyi, R. (2000). The application of Shannon entrophy in the identification of putative drug targets. *Biosystem* **55,** 5-15.

Fulkerson, W. J., McDowell, G. H., and Fell, L. R. (1975). Artificial induction of lactation in ewes: the role of prolactin. *Aust J Biol Sci* **28,** 525-30.

Ganguly, R., Ganguly, N., Mehta, N. M., and Banerjee, M. R. (1980a). Absolute requirement of glucocorticoid for expression of the casein gene in the presence of prolactin. *Proc Natl Acad Sci U S A* **77,** 6003-6.

Ganguly, R., Mehta, N. M., Ganguly, N., and Banerjee, M. R. (1979). Glococorticoid modulation of casein gene transcription in mouse mammary gland. *P Proc Natl Acad Sci U S A* **76,** 6466-6470.

Ganguly, R., Mehta, N. M., Ganguly, N., and Banerjee, M. R. (1980b). Absolute requirement of glucocorticoid for expression of the casein gene in the presence of prolactin. *Proc Natl Acad Sci U S A* **77,** 6003-6006.

Gao, J., and Horseman, N. D. (1999). Prolactin-independent modulation of the beta-casein response element by Erk2 MAP kinase. *Cell Signal* **11,** 205-10.

Giovannone, B., Scaldaferri, M. L., Federici, M., Porzio, O., Lauro, D., Fusco, A., Sbraccia, P., Borboni, P., Lauro, R., and Sesti, G. (2000). Insulin receptor substrate (IRS) transduction system: distinct and overlapping signaling potential. *Diabetes/Metabolism Res Rev* **16,** 434-441.

Gordon, S. R., and Buxar, R. M. (1997). Inhibition of cytoskeletal reorganization stimulates actin and tubulin syntheses during injury-induced cell migration in the

corneal endothelium. *J Cell Biochem* **67,** 409-21.

Gorewit, R. C., Jiang, J., and Aneshansley, D. J. (1993). Responses of the bovine mammary artery to angiotensins. *J Dairy Sci* **76,** 1278-84.

Gow, C. B., McDowell, G. H., and Jenkin, G. (1983). The importance of prolactin for initiation of lactation in the pregnant ewe. *Aust J Biol Sci* **36,** 357-67.

Griinari, J. M., McGuire, M. A., Dwyer, D. A., Bauman, D. E., Barbano, D. M., and House, W. A. (1997). The role of insulin in the regulation of milk protein synthesis in dairy cows. *J Dairy Sci* **80,** 2361-2371.

Gromov, P. S., and Celis, J. E. (1991). Identification of two molecular chaperons (HSX70, HSC70) in mature human erythrocytes. *Exp Cell Res* **195,** 556-9.

Groner, B. (2002). Transcription factor regulation in mammary epithelial cells. *Domest Anim Endocrinol* **23,** 25-32.

Gusdon, J. P., Jr., and Yen, S. S. (1967). In vitro biosynthesis of human placental lactogen (HPL) by placental tissue. *Obstet Gynecol* **30,** 635-8.

Hart, I. C. (1972). Level of prolactin in the blood of the goat during milking, throughout lactation and over a 24-h period. *J Endocrinol* **55,** 28.

Hart, I. C. (1974). The relationship between lactation and the release of prolactin and growth hormone in the goat. *J Reprod Fertil* **39,** 485-99.

Hastie, T., Tibshirani, R., Eisen, M. B., Alizadeh, A., Levy, R., Staudt, L., Chan, W. C., Botstein, D., and Brown, P. (2000). 'Gene shaving' as a method for identifying distinct sets of genes with similar expression patterns. *Genome Biol* **1,** research0003.1-0003.21.

Hawse, J. R., Hejtmancik, J. F., Huang, Q., Sheets, N. L., Hosack, D. A., Lempicki, R. A., Horwitz, J., and Kantorow, M. (2003). Identification and functional clustering of global gene expression differences between human age-related cataract and clear lenses. *Mol Vis* **9,** 515-37.

Hennighausen, L., and Robinson, G. W. (2001). Signaling pathways in mammary gland development. *Dev Cell* **1,** 467-75.

Herrero, J., Diaz-Uriarte, R., and Dopazo, J. (2003). Gene expression data preprocessing. *Bioinformatics* **19,** 655-6.

Herzel, H., Beule, D., Kielbasa, S., Korbel, J., Sers, C., Malik, A., Eickhoff, H., Lehrach, H., and Schuchhardt, J. (2001). Extracting information about cDNA arrays. *Chao* **11,** 98-107.

Holmes, M. C., Kotelevtsev, Y., Mullins, J. J., and Seckl, J. R. (2001). Phenotypic analysis of mice bearing targeted deletions of 11beta-hydroxysteroid dehydrogenases 1 and 2 genes. *Mol Cell Endocrinol* **171,** 15-20.

Holopainen, J. M., Moilanen, J. A., Sorsa, T., Kivela-Rajamaki, M., Tervahartiala, T., Vesaluoma, M. H., and Tervo, T. M. (2003). Activation of matrix metalloproteinase-8 by membrane type 1-MMP and their expression in human

tears after photorefractive keratectomy. *Invest Ophthalmol Vis Sci* **44,** 2550-6.

Holter, N. S., Maritan, A., Cieplak, M., Fedoroff, N. V., and Banavar, J. R. (2001). Dynamic modeling of gene expression data. *Proc Natl Acad Sci U S A* **98,** 1693-1698.

Holter, N. S., Mitra, M., Maritan, A., Cieplak, M., Banavar, J. R., and Fedoroff, N. V. (2000). Fundamental patterns underlying gene expression profiles: Simplicity from complexity. *Proc Natl Acad Sci U S A* **97,** 8409Ð8414.

Hoon, M. J. L. d., Imoto, S., and Miyano, S. (2002). Statistical analysis of a small set of time-ordered gene expression data using linear splines. *Bioinformatics* **18,** 1477-1485.

Houdebine, L. M. (1983). Recent data on the mechanism of action of prolactin. *Ann Endocrinol (Paris)* **44,** 85-100.

Houdebine, L. M., Djiane, J., Dusanter-Fourt, I., Martel, P., Kelly, P. A., Devinoy, E., and Servely, J. L. (1985). Hormonal action controlling mammary activity. *J Dairy Sci* **68,** 489-500.

Hoyle, D. C., Rattray, M., Jupp, R., and Brass, A. (2002). Making sense of microarray data distributions. *Bioinformatics* **18,** 576-84.

Hudson, A. W., Ruiz, M., and Birnbaum, M. J. (1992). Isoform-specific subcellular targeting of glucose transporters in mouse fibroblasts. *J Cell Biol* **116,** 785-97.

Hynes, R. O., Marcantonio, E. E., Stepp, M. A., Urry, L. A., and Yee, G. H. (1989). Integrin heterodimer and receptor complexity in avian and mammalian cells. *J Cell Biol* **109,** 409-20.

Ichijima, H., Petroll, W. M., Barry, P. A., Andrews, P. M., Dai, M., Cavanagh, H. D., and Jester, J. V. (1993). Actin filament organization during endothelial wound healing in the rabbit cornea: comparison between transcorneal freeze and mechanical scrape injuries. *Invest Ophthalmol Vis Sci* **34,** 2803-12.

Jahn, G. A., Daniel, N., Jolivet, G., Belair, L., Bole-Feysot, C., Kelly, P. A., and Djiane, J. (1997). In vivo study of prolactin (PRL) intracellular signalling during lactogenesis in the rat: JAK/STAT pathway is activated by PRL in the mammary gland but not in the liver. *Biol Reprod* **57,** 894-900.

Jeanteur, P. (1998). [The role of APC in colonic cancerogenesis: zeroing in on Myc]. *Bull Cancer* **85,** 925-8.

Jirikowski, G. F. (1992). Oxytocinergic neuronal systems during mating, pregnancy, parturition, and lactation. *Ann N Y Acad Sci* **652,** 253-70.

Jonas, J. C., Laybutt, D. R., Steil, G. M., Trivedi, N., Pertusa, J. G., Van de Casteele, M., Weir, G. C., and Henquin, J. C. (2001). High glucose stimulates early response gene c-Myc expression in rat pancreatic beta cells. *J Biol Chem* **276,** 35375-81.

Kalasauskaite, E., and Grinius, L. (1979). The role of energy-yielding ATPase and respiratory chain at early stages of bacteriophage T4 infection. *FEBS Lett* **99**, 287-91.

Kalish, J. A., Willis, D. J., Li, C., Link, J. J., Deutsch, E. R., Contreras, M. A., Quist, W. C., and Logerfo, F. W. (2004). Temporal genomics of vein bypass grafting through oligonucleotide microarray analysis. *J Vasc Surg* **39,** 645-54.

Kasturi, J., Acharya, R., and Ramanathan, M. (2003). An information theoretic approach for analyzing temporal patterns of gene expression. *Bioinformatics* **19,** 449-458.

Keough, E. M., and Wood, B. G. (1979). Mammary gland development during pregnancy in the dwarf mouse mutant, little. *Tissue Cell* **11,** 773-80.

Kerr, M. K., and Churchill, G. A. (2001). Bootstrapping cluster analysis: assessing the reliability of conclusions from microarray experiments. *Proc Natl Acad Sci U S A* **98,** 8961-8965.

Kim, J. H., Shin, D. M., and Lee, Y. S. (2002). Effect of local background intensities in the normalization of cDNA microarray data with a skewed expression profiles. *Exp Mol Med* **34,** 224-32.

Krishnadasan, B., Naidu, B. V., Byrne, K., Fraga, C., Verrier, E. D., and Mulligan, M. S. (2003). The role of proinflammatory cytokines in lung ischemia-reperfusion injury. *J Thorac Cardiovasc Surg* **125,** 261-72.

Krishnamra, N., and Taweerathitam, P. (1995). Acute effect of prolactin on active calcium absorption in rats. *Can J Physiol Pharmacol* **73,** 1185-9.

Kroll, T. C., and Wolfl, S. (2002). Ranking: a closer look on globalisation methods for normalization of gene expression arrays. *Nucleic Acid Research* **30,** e50.

Kulski, J. K., Nicholas, K. R., Topper, Y. J., and Qasba, P. (1983a). Essentiality of insulin and prolactin for accumulation of rat casein mRNAs. *Biochem Biophys Res Commun* **116,** 994-999.

Kulski, J. K., Topper, Y. J., Chomczynski, P., and Qasba, P. (1983b). An essential role for glucocorticoid in casein gene expression in rat mammary explants. *Biochem Biophys Res Commun* **114,** 380-387.

Kwon, H. C., Kim, S. H., Roh, M. S., Kim, J. S., Lee, H. S., Choi, H. J., Jeong, J. S., Kim, H. J., and Hwang, T. H. (2004). Gene expression profiling in lymph node-positive and lymph node-negative colorectal cancer. *Dis Colon Rectum* **47,** 141-52.

L. J. Faulkin, J., and Deome, K. B. (1960). Regulation of growth and spacing of gland elements in the mammary fat pad of C3H mouse. *J Natl Cancer Inst* **24,** 953-969.

Lee, D. K., Park, J. W., Kim, Y. J., Kim, J., Lee, Y., and Kim, J. S. (2003). Toward a functional annotation of the human genome using artificial transcription factors. *Genome Res* **13,** 2708-16.

Lelievre, S., Weaver, V. M., and Bissell, M. J. (1996). Extracellular matrix signaling from the cellular membrane skeleton to the nuclear skeleton: a model of gene regulation. *Recent Prog Horm Res* **51,** 417-32.

Levieux, D., and Ollier, A. (1999). Bovine immunoglobulin G, beta-lactoglobulin, alpha-lactalbumin and serum albumin in colostrum and milk during the early post

partum period. *J Dairy Res* **66,** 421-30.

Li, H., and Hong, F. (2001). Cluster-Rasch models for microarray gene expression data. *Genome Biol* **2,** research0031.1Ð0031.13.

Li, M., Hu, J., Heermeier, K., Hennighausen, L., and Furth, P. A. (1996). Apoptosis and remodeling of mammary gland tissue during involution proceeds through p53-independent pathways. *Cell Growth Differ* **7,** 13-20.

Lin, C. Q., and Bissell, M. J. (1993). Multi-faceted regulation of cell differentiation by extracellular matrix. *Faseb J* **7,** 737-43.

Liu, X., Robinson, G. W., Wagner, K. U., Garett, L., Wynshaw-Boris, A., and Hennighausen, L. (1997). Stat5a is mandatory for adult mammary gland development and lactogenesis. *Genes Dev* **11,** 179-186.

Luzikov, V. N. (2002). Quality control: proteins and organelles. *Biochemistry (Mosc)* **67,** 171-83.

Mackie, T. R., Dwyer, D. A., Ingvartsen, K. L., Chouinard, P. Y., Ross, D. A., and Bauman, D. E. (2000). Effects of insulin and postruminal supply of protein on use of amino acids by mammary gland for milk protein synthesis. *J Dairy Sci* **83,** 93-105.

Majoul, I., Straub, M., Hell, S. W., Duden, R., and Soling, H. D. (2001). KDEL-cargo regulates interactions between proteins involved in COPI vesicle traffic: measurements in living cells using FRET. *Dev Cell* **1,** 139-53.

Majumder, G. C., and Turkington, R. W. (1971). Hormonal regulation of protein kinases and adenosine 3', 5'-monophosphate-binding protein in developing mammary gland. *J Biol Chem* **246,** 5545-5554.

Malewski, T., Gajewska, M., Zebrowska, T., and Zwierzchowski, L. (2002). Differential induction of transcription factors and expression of milk protein genes by prolactin and growth hormone in the mammary gland of rabbits. *Growth Hormone & IGF Res* **12,** 41-53.

Manduchi, E., Grant, G. R., McKenzie, S. E., Overton, G. C., Surrey, S., and Stoeckert, C. J. (2000). Generation of patterns from gene expression data by assigning confidence to differentially expressed genes. *Bioinformatics* **16,** 685-198.

Marchion, D. C., Cheung, D. T., Grobe, A. C., Weber, P. A., and Duran, C. M. (2003). Cellular expression of PCNA and procollagen in vital and ethanol-treated autologous pericardial implants in sheep. *J Heart Valve Dis* **12,** 87-92.

Marette, A., Burdett, E., Douen, A., Vranic, M., and Klip, A. (1992). Insulin induces the translocation of GLUT4 from a unique intracellular organelle to transverse tubules in rat skeletal muscle. *Diabetes* **41,** 1562-9.

Master, S. R., Hartman, J. L., D'Cruz, C. M., Moody, S. E., Keiper, E. A., Ha, S. I., Cox, J. D., Belka, G. K., and Chodosh, L. A. (2002). Functional microarray analysis of mammary organogenesis reveals a developmental role in adaptive thermogenesis. *Mol Endocrinol* **16,** 1185-1203.

Melman, L., Geuze, H. J., Li, Y., McCormick, L. M., Van Kerkhof, P., Strous, G. J., Schwartz, A. L., and Bu, G. (2002). Proteasome regulates the delivery of LDL receptor-related protein into the degradation pathway. *Mol Biol Cell* **13,** 3325-35.

Mepham, T. B. (1982). Amino acid utilization by lactating mammary gland. *J Dairy Sci* **65**, 287-98.

Michael, R., Vrensen, G. F., van Marle, J., Lofgren, S., and Soderberg, P. G. (2000). Repair in the rat lens after threshold ultraviolet radiation injury. *Invest Ophthalmol Vis Sci* **41,** 204-12.

Miller, C. A., and Debas, H. T. (1995). Epidermal growth factor stimulates the restitution of rat gastric mucosa in vitro. *Exp Physiol* **80,** 1009-18.

Mills, E. S., and Topper, Y. J. (1969). Mammary alveolar epithelial cells: effect of hydrocortisone on ultrastructure. *Science* **165,** 1127-8.

Mills, E. S., and Topper, Y. J. (1970). Some ultrastructural effects of insulin, hydrocortisone, and prolactin on mammary gland explants. *J Cell Biol* **44,** 310-28.

Mills, S. E., Lemenager, R. P., and Horstman, L. A. (1989). Effect of suppression of prolactin secretion on adipose lipogenesis in the postpartum beef cow. *J Anim Sci* **67,** 2904-12.

Moorehead, R. A., Fata, J. E., Johnson, M. B., and Khokha, R. (2001). Inhibition of mammary epithelial apoptosis and sustained phosphorylation of Akt/PKB in MMTV-IGF-II transgenic mice. *Cell Death Differ* **8,** 16-29.

Munro, S., and Pelham, H. R. (1987). A C-terminal signal prevents secretion of luminal ER proteins. *Cell* **48,** 899-907.

Nagaiah, K., Franklyn F. Borlander, J., Nicholas, K. R., Takemoto, T., and Topper, Y. J. (1981). Prolactin-induced accumulation of casein mRNA in mouse mammary explants: a selective role of glucocorticoid. *Biochem Biophys Res Commun* **98,** 380-387.

Nagarajan, R. (2003). Intensity-based segmentation of microarray images. *IEEE Trans Med Imaging* **22,** 882-9.

Nakhasi, H. L., and Quasba, P. K. (1979). Quantitation of milk proteins and their mRNAs in rat mammary gland at various stages of gestation and lactation. *J Biol Chem* **254,** 6016-25.

Nicholas, K. R., Sankaran, L., and Topper, Y. J. (1983). A unique and essential role for insulin in the phenotypic expression of rat mammary epithelial cells unrelated to its function in cell maintenance. *Biochemi Biophys Acta* **763,** 309-314.

Nicholas, K. R., and Topper, Y. J. (1983). Anti-insulin receptor serum mimics the developmental role of insulin in mouse mammary explants. *Biochem Biophys Res Commun* **111,** 988-993.

Noel, A., Emonard, H., Polette, M., Birembaut, P., and Foidart, J. M. (1994). Role of matrix, fibroblasts and type IV collagenases in tumor progression and invasion.

Pathol Res Pract **190,** 934-41.

O'Hare, J. A. (1988). The enigma of insulin resistance and hypertension. Insulin resistance, blood pressure, and the circulation. *Am J Med* **84,** 505-10.

Oliver, N., Newby, R. F., Furcht, L. T., and Bourgeois, S. (1983). Regulation of fibronectin biosynthesis by glucocorticoids in human fibrosarcoma cells and normal fibroblasts. *Cell* **33,** 287-96.

Parodi, A. J. (2000a). Protein glucosylation and its role in protein folding. *Annu Rev Biochem* **69,** 69-93.

Parodi, A. J. (2000b). Role of N-oligosaccharide endoplasmic reticulum processing reactions in glycoprotein folding and degradation. *Biochem J* **348 Pt 1,** 1-13.

Pavlidis, P., Weston, J., Cai, J., and Grundy, W. N. (2001). Gene functional classification from heterogeneous data. *In* "Proceedings of the Fifth International Conference on Research in Computational Molecular Biology", pp. 249-255.

Pelham, H. R. (1988). Evidence that luminal ER proteins are sorted from secreted proteins in a post-ER compartment. *Embo J* **7,** 913-8.

Pelham, H. R., Hardwick, K. G., and Lewis, M. J. (1988). Sorting of soluble ER proteins in yeast. *Embo J* **7,** 1757-62.

Piccoletti, R., Maroni, P., Bendinelli, P., and Bernelli-Zazzera, A. (1994). Rapid stimulation of mitogen-activated protein kinase of rat liver by prolactin. *Biochem J* **303 (Pt 2),** 429-33.

Prosser, C. G., Davis, S. R., Farr, V. C., and Lacasse, P. (1996). Regulation of blood flow in the mammary microvasculature. *J Dairy Sci* **79,** 1184-97.

Pusztai, L., Ayers, M., Stec, J., Clark, E., Hess, K., Stivers, D., Damokosh, A., Sneige, N., Buchholz, T. A., Esteva, F. J., Arun, B., Cristofanilli, M., Booser, D., Rosales, M., Valero, V., Adams, C., Hortobagyi, G. N., and Symmans, W. F. (2003). Gene expression profiles obtained from fine-needle aspirations of breast cancer reliably identify routine prognostic markers and reveal large-scale molecular differences between estrogen-negative and estrogen-positive tumors. *Clin Cancer Res* **9,** 2406-15.

Qi, W. N., and Scully, S. P. (2003). Type II collagen modulates the composition of extracellular matrix synthesized by articular chondrocytes. *J Orthop Res* **21,** 282-9.

Ricciardelli, C., Brooks, J. H., Suwiwat, S., Sakko, A. J., Mayne, K., Raymond, W. A., Seshadri, R., LeBaron, R. G., and Horsfall, D. J. (2002). Regulation of stromal versican expression by breast cancer cells and importance to relapse-free survival in patients with node-negative primary breast cancer. *Clin Cancer Res* **8,** 1054-60.

Riley, D. J., Kramer, M. J., Kerr, J. S., Chae, C. U., Yu, S. Y., and Berg, R. A. (1987). Damage and repair of lung connective tissue in rats exposed to toxic levels of oxygen. *Am Rev Respir Dis* **135,** 441-7.

Robinson, L. J., Pang, S., Harris, D. S., Heuser, J., and James, D. E. (1992). Translocation of the glucose transporter (GLUT4) to the cell surface in permeabilized 3T3-L1 adipocytes: effects of ATP insulin, and GTP gamma S and localization of GLUT4 to clathrin lattices. *J Cell Biol* **117,** 1181-96.

Rodionov, D. G., Honing, S., Silye, A., Kongsvik, T. L., von Figura, K., and Bakke, O. (2002). Structural requirements for interactions between leucine-sorting signals and clathrin-associated adaptor protein complex AP3. *J Biol Chem* **277,** 47436-43.

Rosen, J. M., and Barker, S. W. (1976). Quantitation of casein messenger ribonucleic acid sequences using a specific complementary DNA hybridization probe. *Biochemistry* **15,** 5272-80.

Ryan, M. T., Naylor, D. J., Hoj, P. B., Clark, M. S., and Hoogenraad, N. J. (1997). The role of molecular chaperones in mitochondrial protein import and folding. *Int Rev Cytol* **174,** 127-93.

Saal, L. H., Troein, C., Vallon-Christersson, J., Gruvberger, S., Borg, A., and Peterson, C. (2002). BioArray Software Environment (BASE): a platform for comprehensive management and analysis of microarray data. *Genome Biol* **3,** SOFTWARE0003.

Salvemini, D., Riley, D. P., Lennon, P. J., Wang, Z. Q., Currie, M. G., Macarthur, H., and Misko, T. P. (1999). Protective effects of a superoxide dismutase mimetic and peroxynitrite decomposition catalysts in endotoxin-induced intestinal damage. *Br J Pharmacol* **127,** 685-92.

Sasik, R., Hwa, T., Iranfar, N., and Loomis, W. F. (2001). Percolation clustering: a novel algorithm applied to the clustering of gene expression patterns in dictyostelium development. *Proceedings of the Pacific Symposium of Biocomputing* **6,** 335-347.

Schadt, E. E., Li, C., Ellis, B., and Wong, W. H. (2001). Feature extraction and normalization algorithms for high-density oligonucleotide gene expression array data. *Journal of Cellular Biochemistry* **84,** 120-125.

Schwertfeger, K. L., Richert, M. M., and Anderson, S. M. (2001). Mammary gland involution is delayed by activated Akt in transgenic mice. *Mol Endocrinol* **15,** 867-81.

Scmidt, G. H. (1966). Effect of insulin on yield and composition of milk of dairy cows. *J Dairy Sci* **47,** 381-385.

Shimizu, A., Masuda, Y., Kitamura, H., Ishizaki, M., Sugisaki, Y., and Yamanaka, N. (1998). Recovery of damaged glomerular capillary network with endothelial cell apoptosis in experimental proliferative glomerulonephritis. *Nephron* **79,** 206-14.

Sinha, Y. N., Selby, F. W., and Vanderlaan, W. P. (1974). Relationship of prolactin and growth hormone to mammary function during pregnancy and lactation in the C3H-ST mouse. *J Endocrinol* **61,** 219-29.

Skarda, J., Urbanova, E., Becka, S., Houdebine, L. M., Delouis, C., Pichova, D., Picha, J., and Bilek, J. (1982). Effect of bovine growth hormone on development of goat

mammary tissue in organ culture. *Endocrinol Exp* **16,** 19-31.

Smith, S. K., He, Y., Clark, D. E., and Charnock-Jones, D. S. (2000). Angiogenic growth factor expression in placenta. *Semin Perinatol* **24,** 82-6.

Solomon, S. S., Palazzolo, M. R., Smoake, J. A., and Raghow, R. S. (1995). Insulin-stimulated calmodulin gene expression in rat H-411E cells can be selectively blocked by antisense oligonucleotides. *Biochem Biophys Res Comm* **210,** 921-30.

Song, J. H., Kim, J. M., Kim, S. H., Kim, H. J., Lee, J. J., Sung, M. H., Hwang, S. Y., and Kim, T. S. (2003). Comparison of the gene expression profiles of monocytic versus granulocytic lineages of HL-60 leukemia cell differentiation by DNA microarray analysis. *Life Sci* **73,** 1705-19.

Stanislawski, L., Simpson, W. A., Hasty, D., Sharon, N., Beachey, E. H., and Ofek, I. (1985). Role of fibronectin in attachment of Streptococcus pyogenes and Escherichia coli to human cell lines and isolated oral epithelial cells. *Infect Immun* **48,** 257-9.

Steele, J. G., Johnson, G., Norris, W. D., and Underwood, P. A. (1991). Adhesion and growth of cultured human endothelial cells on perfluorosulphonate: role of vitronectin and fibronectin in cell attachment. *Biomaterials* **12,** 531-9.

Steffensen, K. R., and Gustafsson, J. A. (2004). Putative metabolic effects of the liver X receptor (LXR). *Diabetes* **53 Suppl 1,** S36-42.

Stokes, D. G., Liu, G., Coimbra, I. B., Piera-Velazquez, S., Crowl, R. M., and Jimenez, S. A. (2002). Assessment of the gene expression profile of differentiated and dedifferentiated human fetal chondrocytes by microarray analysis. *Arthritis Rheum* **46,** 404-19.

Sudhakaran, P. R., Ambili, M., and Philip, S. (1999). Matrix metalloproteinase in mammary gland remodeling-modulation by glycosaminoglycans. *Biosci Rep* **19,** 485-90.

Suenaga, M., Kuwajima, M., Himeda, T., Morokami, K., Matsuura, T., Ozaki, K., Arakaki, N., Shibata, H., and Higuti, T. (2004). Identification of the up- and down-regulated genes in the heart of juvenile visceral steatosis mice. *Biol Pharm Bull* **27,** 496-503.

Tabibiazar, R., Wagner, R. A., Liao, A., and Quertermous, T. (2003). Transcriptional profiling of the heart reveals chamber-specific gene expression patterns. *Circ Res* **93,** 1193-201.

Taib, Z. (2003). New estimates of gene expression for oligonucleotide microchip arrays. *Comptes Rendus de I'Academie des Sciences B.*

Tatekawa, Y., Kanehiro, H., Hisanaga, M., and Nakajima, Y. (2003). Matrix metalloproteinase-9 and tissue inhibitor of metalloproteinase-1: expression in the lung of fetal rats with nitrofen-induced diaphragmatic hernia. *Pediatr Surg Int* **19,** 25-8.

Tamayo, P., Slonim, D., Mesirov, J., Zhu, Q., Kitareewan, S., Dmitrovsky, E., Lander, E. S., and Golub, T. R. (1999). Interpreting patterns of gene expression with self-

organizing maps: methods and applications to hematopoietic differentiation. *Proceedings of the Pacific Symposium of Biocomputing* **4,** 5-16.

Tatarczuch, L., Philip, C., and Lee, C. S. (1997). Involution of the sheep mammary gland. *J Anat* **190 (Pt 3),** 405-16.

Topper, Y. J., and Freeman, C. S. (1980). Multiple hormone interactions in the developmental biology of the mammary gland. *Physiol Rev* **60,** 1049-1106.

Topper, Y. J., Nicholas, K. R., Sankaran, L., and Kulski, J. (1984a). Insulin as a developmental hormone. *In* "Hormones and Cancer", pp. 63-77.

Topper, Y. J., Nicholas, K. R., Sankaran, L., and Kulski, J. K. (1984b). Insulin biology from the perspective of studies on mammary gland development. *In* "Biochemical actions of hormone", Vol. XI, pp. 163-186. Academic Press, Inc.

Tseng, G. C., Oh, M.-K., Rohlin, L., Liao, J. C., and Wong, W. H. (2001). Issues in cDNA microarray analysis: quality filtering, channel normalization, models of variation and assessment of gene effects. *Nucleic Acid Res* **29,** 2549-2557.

Tsuchiya, T., and Rosen, B. P. (1975). Energy transduction in Escherichia coli. The role of the Mg2+ATPase. *J Biol Chem* **250**, 8409-15.

Turkington, R. W., Juergens, W. G., and Topper, Y. J. (1967). Steroid structural requirements for mammary gland differentiation in vitro. *Endocrinol* **80,** 1139-42.

Ueno, S., Umar, H., Bambauer, H. J., and Ueck, M. (1984). Localization of ATPases in retinal receptor cells. *Ophthalmic Res* **16**, 15-20.

Untergasser, G., Kranewitter, W., Walser, F., Madersbacher, S., Dirnhofer, S., and Berger, P. (1996). The testis as eutopic production site of human growth hormone, placental lactogen and prolactin: possible autocrine/paracrine effects on testicular function. *Wien Klin Wochenschr* **108,** 541-6.

Venesio, T., Balsamo, A., Scordamaglia, A., Bertolaso, M., Arrigoni, A., Sprujevnik, T., Rossini, F. P., and Risio, M. (2003). Germline APC mutation on the beta-catenin binding site is associated with a decreased apoptotic level in colorectal adenomas. *Mod Pathol* **16,** 57-65.

Vonderhaar, B. K. (1988). Regulation of development of the normal mammary gland by hormones and growth factors. *Cancer Treat Res* **40,** 251-66.

Walden, P. D., Ruan, W., Feldman, M., and Kleinberg, D. L. (1998). Evidence that the mammary fat pad mediates the action of growth hormone in mammary gland development. *Endocrinol* **139,** 659-662.

Wall, M. E., Dyck, P. A., and Brettin, T. S. (2001). SVDMAN - singular value decomposition analysis of microarray data. *Bioinformatics* **17,** 566-568.

Weaver, D. C., Workman, C. T., and Stormo, G. D. (1999). Modeling regulatory networks with weight matrices. *Proceedings of the Pacific Symposium of Biocomputing* **4,** 112-123.

Weaver, V. M., Fischer, A. H., Peterson, O. W., and Bissell, M. J. (1996). The

importance of the microenvironment in breast cancer progression: recapitulation of mammary tumorigenesis using a unique human mammary epithelial cell model and a three-dimensional culture assay. *Biochem Cell Biol* **74,** 833-51.

Weinacker, A., Chen, A., Agrez, M., Cone, R. I., Nishimura, S., Wayner, E., Pytela, R., and Sheppard, D. (1994). Role of the integrin alpha v beta 6 in cell attachment to fibronectin. Heterologous expression of intact and secreted forms of the receptor. *J Biol Chem* **269,** 6940-8.

Weinstein, D., Ben-David, M., and Polishuk, W. Z. (1976). Serum prolactin and the suppression of lactation. *Br J Obstet Gynaecol* **83,** 679-82.

Welsh, M., Christmansson, L., Karlsson, T., Sandler, S., and Welsh, N. (1999). Transgenic mice expressing Shb adaptor protein under the control of rat insulin promoter exhibit altered viability of pancreatic islet cells. *Mol Med* **5,** 169-80.

Wenzel-Seifert, K., and Seifert, R. (2003). Critical role of N-terminal N-glycosylation for proper folding of the human formyl peptide receptor. *Biochem Biophys Res Commun* **301,** 693-8.

Wessely, O., Deiner, E. M., Beug, H., and von Lindern, M. (1997). The glucocorticoid receptor is a key regulator of the decision between self-renewal and differentiation in erythroid progenitors. *Embo J* **16,** 267-80.

Wheatley, M., and Hawtin, S. R. (1999). Glycosylation of G-protein-coupled receptors for hormones central to normal reproductive functioning: its occurrence and role. *Hum Reprod Update* **5,** 356-64.

Wildsmith, S. E., Archer, G. E., Winkley, A. J., Lane, P. W., and Bugelski, P. J. (2001). Maximization of signal derived from cDNA microarrays. *Biotechniques* **30,** 202-6, 208.

Willoughby, D. S., Taylor, M., and Taylor, L. (2003). Glucocorticoid receptor and ubiquitin expression after repeated eccentric exercise. *Med Sci Sports Exerc* **35,** 2023-31.

Wormald, M. R., and Dwek, R. A. (1999). Glycoproteins: glycan presentation and protein-fold stability. *Structure Fold Des* **7,** R155-60.

Wyszomierski, S. L., and Rosen, J. M. (2001). Cooperative effects of STAT5 (Signal Transducer and Activator of Transcription 5) and C/EBPβ (CCAAT/Enhancer-Binding Protein-β on β -casein gene transcription are mediated by the glucocorticoid receptor. *Molecular Endocrinol* **15,** 228-240.

Xing, E. P., and Karp, R. M. (2001). CLIFF: clustering of high-dimensional microarray data via iterative feature filtering using normalized cuts. *In* "ISMB".

Xu, S. H., Qian, L. J., Mou, H. Z., Zhu, C. H., Zhou, X. M., Liu, X. L., Chen, Y., and Bao, W. Y. (2003). Difference of gene expression profiles between esophageal carcinoma and its pericancerous epithelium by gene chip. *World J Gastroenterol* **9,** 417-22.

Yamamoto, K., Hamada, H., Shinkai, H., Kohno, Y., Koseki, H., and Aoe, T. (2003).

The KDEL receptor modulates the endoplasmic reticulum stress response through mitogen-activated protein kinase signaling cascades. *J Biol Chem* **278,** 34525-32.

Yamanaka, S., Poksay, K. S., Arnold, K. S., and Innerarity, T. L. (1997). A novel translational repressor mRNA is edited extensively in livers containing tumors caused by the transgene expression of the apoB mRNA-editing enzyme. *Genes Dev* **11,** 321-33.

Yamashita, H., Nevalainen, M. T., Xu, J., LeBaron, M. J., Wagner, K.-U., Erwin, R. A., Harmon, J. M., Hennighausen, L., Kirken, R. A., and Rui, H. (2001). Role of serine phosphorylation of Stat5a in prolactin-stimulated β-casein gene expression. *Mol Cell Endocrinol* **183,** 151Ð163.

Yang, J., Clark, A. E., Harrison, R., Kozka, I. J., and Holman, G. D. (1992). Trafficking of glucose transporters in 3T3-L1 cells. Inhibition of trafficking by phenylarsine oxide implicates a slow dissociation of transporters from trafficking proteins. *Biochem J* **281 (Pt 3),** 809-17.

Yang, Y. H., Dudoit, S., Luu, P., Lin, D. M., Peng, V., Ngai, J., and Speed, T. P. (2002). Normalization for cDNA microarray data: a robust composite method addressing single and multiple slide systematic variation. *Nucleic Acids Res* **30,** e15.

Yoshimura, M., and Oka, T. (1990). Hormonal induction of β-casein gene expression: requirement of ongoing protein synthesis for transcription. *Endocrinol* **126,** 427-433.

Yasugi, T., Kaido, T., and Uehara, Y. (1989). Changes in density and architecture of microvessels of the rat mammary gland during pregnancy and lactation. *Arch Histol Cytol* **52**, 115-22.

Zhao, F.-Q., Adachi, K., and Oka, T. (2002). Involvement of Oct-1 in transcriptional regulation of β-casein gene expression in mouse mammary gland. *Biochem Biophys Acta* **1577,** 27-37.

Zhou, L., Li, Y., and Yue, B. Y. (1998). Glucocorticoid effects on extracellular matrix proteins and integrins in bovine trabecular meshwork cells in relation to glaucoma. *Int J Mol Med* **1,** 339-46.

Zhou, Y., and Abagyan, R. (2002). Match-only intergral distribution (MOID) algorithm for high-density oligonucleotide array analysis. *BMC Bioinformatics* **3,** 3.

Zien, A., Kuffner, R., Zimmer, R., and Lengauer, T. (2000). Analysis of gene expression data with pathway scores. *In* "Proceedings of the Eighth International Conference on Intelligent Systems for Molecular Biology (AAAI Press)", pp. 407-417.

Appendix A:
Statistical Analysis and Analysis Modeling of Microarray Comparisons

Introduction

This appendix sets forth to analyze the statistical relevance of the microarray comparisons used to analyze the set of hormone treatment experiments. In order to do so, the research question is de-composed mathematically, followed by mathematical parameterization and statistical analysis of the microarray experimental design. After which, this appendix ends with a proposal of an analysis model based on mathematical grounds established here.

Decomposition of Research Question

The research question for hormone treatment experiments is stated as, “What are the contributions of each of the hormones (insulin, prolactin, and glucocorticoid) in lactation, using an explant model, in the presence of the other two hormones?”

For the purpose of mathematical analysis, the culture model and the types of hormones bear no significance. Hence, mathematically, the research question can be re-written to be, "What are the contributions of each of the three hormones in the presence of the other two?" Inherent in this question is our first experimental constraint, which is also an important criteria for design evaluation.

Constraint 1:

> Each of the hormones are equally important for analysis and their contributions must be addressed in equal weightage.

Material limitations can be stated as Constraint 2,

Constraint 2:

> There will be enough resources for three microarray comparisons.

Mathematically, let the contributions of insulin, prolactin and glucocorticoid (by itself, without interacting with other hormones) be denoted as I, P, and F respectively. (Definition 1)

Biologically, it is known that in a 2-stimuli environment, there will exist an interacting effect of the two stimuli that is not present when only one stimulus is used. This can be termed as synergistic effect. (Axiom 1)

At the same time, where there are no stimulus, there will be a baseline effect. (Axiom 2)

Let the term, U, denotes the baseline effects. (Defintion 2)

Extending Definition 1, let the terms, IP, IF, FP represent the interacting effect of two

hormones, and the term, IFP represents the interacting effect of the three hormones.
(Definition 3)

By the above, a three-stimuli environment will consist of eight effects, baseline, three singly stimuli, three 2-stimuli synergy, and one 3-stimuli synergy.
(Lemma 1)

Now, the research question can be decomposed into three questions and defined mathematically as,
1. What are the terms I, IP and IF in the presence of P, F and FP?;
2. What are the terms P, IP and FP in the presence of I, F and IF?; and
3. What are the terms F, IF and FP in the presence of I, P and IP?

At the same time, the term IFP is also inherent in these question.

Therefore, the research question can be mathematically viewed as an attempt to define the terms I, F, P, IF, IP, FP and IFP, with emphasis on the interacting terms, namely, IF, IP, FP and IFP. This can be formally stated as Constraint 3,

Constraint 3:
More emphasis should be placed on IF, IP, FP and IFP, over I, F and P.

Parameterization of Design

Given four cultured tissues as, FCS-IF-2, FCS-IP-2, FCS-FP-2 and FCS-IFP-2, we can parameterize the effects of each culture based on the expected hormonal effects as follows:

FCS-IF-2	U + I + F + IF	(Eq 1)
FCS-IP-2	U + I + P + IP	(Eq 2)
FCS-FP-2	U + F + P + FP	(Eq 3)
FCS-IFP-2	U + I + F + P + IF + IP + FP + IFP	(Eq 4)

Where U is the baseline of comparisons, corresponding to no hormone treatment. In terms of microarray experiments, the effects of the hormones, singly and in combination, can be viewed as a set of base-2 logarithmic ratio of intensities (including negative ratio).

For example, in Microarray 1, where FCS-IF-2 is hybridized with FCS-IFP-2 (FCS-IF-2 vs FCS-IFP-2), the set of non-zero ratio obtained is essentially the effect of P + IP + FP + IFP ,
(Eq 5)

as a slide of FCS-IF-2 vs FCS-IF-2 will theoretically give only zero ratio.
(Lemma 2)

Applying this reasoning, the parameters estimated by each of the microarray slides is,

Microarray 2: FCS-IP-2 vs FCS-IFP-2 will estimate F + IF + FP + IFP; and
(Eq 6)

Microarray 3: FCS-FP-2 vs FCS-IFP-2 will estimate I + IF + IP + IFP.
(Eq 7)

With these, we can now check if this design is optimal.

Statistical Analysis of Design

It is important to realize that given an unlimited space and resources, there can never be a best design but this is not the case, as cost and time are always limiting. At the same time, the number of possible comparisons escalates exponentially, making it impossible to disprove each design individually. Hence, we can only assess, based on the resources and constraints, is the above design "good enough". By that, meaning optimal.

It is assumed that each parameter is evaluated equally on a microarray slide. That is to say, if two parameters are to be estimated, there will not be parameter-bias. There is no technical nor biological reasons why this not the case. Therefore, we will be looking at the presence of parameter estimation.

(Axiom 3)

By Axiom 3, we can now calculate the weightage of each of the parameters in question from all of the seven microarray comparisons, which is given as

Eq	***I***	***F***	***P***	***IF***	***IP***	***FP***	***IFP***
5	0	0	1	0	1	1	1
6	0	1	0	1	0	1	1
7	1	0	0	1	1	0	1
Total	1	1	1	2	2	2	3

As observed, each of the seven parameters in the research question had been tested (fulfilling Constraint 1) using three microarray slides (fulfilling Constraint 2), and an emphasis on IF, IP, FP and IFP over I, F and P (fulfilling Constraint 3).Hence, the above design is optimal.

quod erat demonstrandum

Analysis Modeling

As stated in the research question, the purpose of this experiment is the define the terms, I, F, P, IF, IP, FP, and IFP.

Let S be the superset,

$S = (\text{Eq } 5) \cup (\text{Eq } 6) \cup (\text{Eq } 7)$ (Eq 8)

$I = S \cap ((\text{Eq } 5) \cup (\text{Eq } 6))'$ (Eq 9)
$F = S \cap ((\text{Eq } 5) \cup (\text{Eq } 7))'$ (Eq 10)
$P = S \cap ((\text{Eq } 6) \cup (\text{Eq } 7))'$ (Eq 11)

$IP = ((\text{Eq } 5) \cap (\text{Eq } 7)) \cup (\text{Eq } 6)'$ (Eq 12)

IF = ((Eq 6) ∩ (Eq 7)) ∪ (Eq 5)' (Eq 13)
FP = ((Eq 5) ∩ (Eq 6)) ∪ (Eq 7)' (Eq 14)

IFP = (Eq 5) ∩ (Eq 6) ∩ (Eq 7) (Eq 15)

Thus, all seven parameters are defined.

quod erat demonstrandum

Appendix B:
Specification of BASEfile File Handlers and Implementation of BASEFile File Reader

Introduction

BASEfile is the main file output format of BioArray Software Environment (BASE), to be used as input files to external plugins. These plugins will be required to present their output in the format of BASEfile, in order to allow the job controller script to read the output and feed it back into BASE. Hence, it is essential for plugin developers to understand the workings of BASEfile reader and writer (file handlers). However, BASE distribution includes very little documentation on the specification and operation of the file handlers.

In order to understand the operations of file handlers, an analysis of the current implementation is needed. As the source codes of BASE is released under GNU General Public License, it is possible to analyze the implementation from the source codes. The BASEfile file reader is coded by 4 files:

1. basefile.h (containing BaseFileReader and BaseFileSection class declarations),
2. basefile.cc (containing BaseFileReader and BaseFileSection class implementations),
3. basefile_plugin.h (containing PluginBaseFileReader class declaration), and
4. basefile_plugin.cc (containing PluginBaseFileReader class implementation).

For further insights into the workings of BASEfile file handlers, it is crucial to understand the layout of a BASEfile file, and how a plugin uses the file handlers. The latter is achieved by an analysis of the codes of a template plugin (file name: myplugin.cc) and its support file (file name: microarr_point.hh), as given by the developers of BASE.

Hence, the purpose of this appendix is to provide analysis of BASEfile file reader, from the source codes, in order to elucidate the internal workings. After which, a formal specification of the BASEfile file handlers will be proposed, which will serve as a documentation for future plugin developers and developers aiming to re-write BASEfile file handlers in any other programming languages.

BASEfile File Format

BASEfile files are essentially ASCII (American Standard Codes for Information Interchange) tab-delimited text files. Their characteristics are listed below.

1. Each BASEfile must begin with the string, "BASEfile".
2. Each BASEfile should have at least one section.
3. Each section consists of a number of <field> <attribute(s)> pairs, in that order.
4. Each section must have a "section" field and must have a value (the attribute).
5. With the exception of the "section" field, all other fields may have optional attribute(s).
6. There is no order of fields.

7. Each <field> <attribute(s)> pair will occupy a single line.
8. Each <field> <attribute(s)> pair may be preceded by the character "#" but it will not be taken into use by BASEfile file handlers.
9. If a section consists of data, then a "columns" field is mandatory.
10. Format of the "columns" field: columns <data_field>.....<data_field>
11. If a section consists of data, there will be a line with only "%", to seperate the section proper and the data lines.
12. If nested fields are desired, the string, "assayData" can be used as a data field as an attribute in the "columns" row, routinely as the last attribute.
13. If "assayData" is present in the "columns" row, there will exist an "assayFields" field.
14. Format of the "assayFields" field: assayFields <data_field> <data_field>.

To illustrate the above, a sniplet of a BASEfile file is given as, with field names in bold,

1. **BASEfile**
2. **section** assays
3. **count** 1
4. **columns** id name
5. **annotationColumns**
6. %
7. 3 D12Lact3/D8Preg5

8. **section** spots
9. **columns** reporterId geneName assayData
10. **channels** 2
11. **assayFields** intensity1 intensity2
12. **assays** 3
13. %
14. BG063496 Homo sapiens hypothetical protein FLJ10540 (FLJ10540), mRNA 319 341

Line 1 indicates that this file is a BASEfile file.

Lines 2 to 7 indicates that the current section is an "array" section, and consisting of data lines of 2 data fields, id, and name. Lines 6 and 13 indicates the subsequent lines, lines 7 and 14, respectively, as the data line.

Line 9 indicates that the current section consists of data lines, with the fields, reporterId and geneName. However, "assayData" is found on line 9, indicating the presence of "assayFields" field, as illustrated in line 11.

Line 11 defines 2 other fields, intensity1 and intensity2. Hence, the data line(s) in this section will be in the format of
<reporterId> <geneName> <intensity1> <intensity 2> as illustrated in line 14, where,

<reporterId> is BG063496,
<geneName> is Homo sapiens hypothetical protein FLJ10540 (FLJ10540), mRNA,
<intensity1> is 319, and
<intensity2> is 341.

It is essential that each section be seperated by a blank line.

Source Codes of basefile.h

```
1.#ifndef _BASEFILE_H #define _BASEFILE_H
2.#ifdef HAVE_CONFIG_H #include <config.h>
3.#endif
4.#include <fstream>
5.#include <map>
6.#include <string>
7.#include <vector>
8.#include <cmath>
9.#include <cstdlib>
10.namespace BASE{
11.typedef std::map<std::string,int> FieldList;
12.class BaseFileSection{
13.  public:
14.   typedef std::map<std::string,std::string> header_t;
15.   BaseFileSection();
16.   bool findIntOpt(const std::string &name, int &ret);
17.   bool findStringOpt(const std::string &name, std::string &ret);
18.   bool findFieldList(const std::string &name, FieldList &mapping);
19.   static bool findFieldPos(const std::string &name, int &ret,
20.        FieldList &mapping);
21.   std::string type;
22.   int startLine, dataStartLine;
23.   header_t headers;};
24.class BaseFileReader{
25.  public:
26.   BaseFileReader(std::istream &s, bool changebuf = true);
27.   ~BaseFileReader();
28.   enum{EOR = 0, FILEERR = 1, BADFORMAT, TOOMANY, TOOFEW, BADTYPE,
  BADSECTION, BADERROR};
29.   bool readSection(BaseFileSection &sect);
30.   template<class vectype_>
31.   bool readVec(std::vector<vectype_> &vec){
32.        if(!readLine()) return false;
33.        if(!buf[0]){_error = EOR; return false;}
34.        char *tok = &buf[0];
35.        char *otok;
36.        for(unsigned int i = 0; i < vec.size(); i++){
37.             if(!tok){_error = TOOFEW;return false;}
38.             otok = strchr(tok, '\t');
39.             if(otok) *(otok++) = 0;
40.             VecTranslate(tok, vec[i]);
41.             tok = otok;}
42.        if(tok){error = TOOMANY;return false;}
43.        return true;}
44.   bool readLines(std::vector<std::string> &vec);
45.   int getError() { return _error; }
46.   int getLine() { return line; }
47.   const char *errText(int e){return errtexts[std::min((int)BADERROR,
  std::max(0, e))];}
48.   const char *errText() { return errText(_error); }
49.  protected:
50.   int _error;
51.   bool isFirstSection() { return _firstSection; }
52.   static const float fnan;
53.   static const double dnan;
54.  private:
55.   bool readHeadersNew(BaseFileSection::header_t &nameValue);
56.   inline void VecTranslate(char *t, int &v){
57.        v = std::atoi(t);};
58.   inline void VecTranslate(char *t, float &v){
59.        if(!*t) v = fnan;
60.        else{char *p;
61.#ifdef HAVE_STRTOF
```

```
62.            v = std::strtof(t, &p);
63.#else
64.            v = std::strtod(t, &p);
65.#endif
66.            if(*p) v = fnan;}};
67.  inline void VecTranslate(char *t, double &v){
68.       if(!*t) v = dnan;
69.       else{char *p;
70.            v = std::strtod(t, &p);
71.            if(*p) v = dnan;}};
72.  template<class vectype_>
73.  inline void VecTranslate(char *t, vectype_ &v){v = t;};
74.  bool readLine();
75.  bool skipData();
76.  std::istream &str;
77.  int line;
78.  bool dataEnd;
79.  bool _firstSection;
80.  std::vector<char> buf, inbuf;
81.  static const int inbuflen;
82.  static const char *errtexts[];};
83.std::istream &getline(std::istream &i, std::vector<char> &v, char
   delim = '\n');
84.void split(const std::string &s, std::vector<std::string> &v,
       size_t maxp = 0, char c = '\t');
85.};#endif
```

Analysis of basefile.h

Standard Template Library (STL) is used in the implementation of BASEfile file handlers, as indicated from lines 4 to 9. Two classes are being defined, BaseFileSection and BaseFileReader. BaseFileSection encapsulates information of a section of a BASEfile file, whereas BaseFileReader defines the necessary functions needed to read a BASEfile file. The actual implementation of each function is defined in basefile.cc.

BaseFileSection defines two maps, FieldList and headers. FieldList is defined as a string-to-integer map while headers is defined as a string-to-string map.

Initialization of BaseFileReader class requires an input stream, as indicated by line 26. It defines five public functions,

1. readSection,
2. readVec,
3. readLines,
4. getError, and
5. getLine.

Of the five functions defined in BaseFileReader, only readVec is implemented. A vector is a sequence of elements and readVec is implemented as a verification routine, which takes a vector and scan it for presence of elements (not empty vector) and it is within bounds.

BaseFileReader defines three private functions,

1. readHeadersNew,
2. readLine, and
3. skipData.

Source Codes of basefile.cc

```
86.#ifdef HAVE_CONFIG_H
87.#include <config.h>
88.#endif
89.#include <cstring>
90.#include <iostream>
91.#include <basefile.h>
92.namespace BASE{
93.void split(const std::string &s, std::vector<std::string> &v, size_t
  maxp, char c){
94.       v.clear();
95.       size_t cp = 0, oldcp = 0;
96.       while((!maxp || v.size() < maxp) &&
              (cp = s.find(c, oldcp)) != std::string::npos){
97.             v.push_back(s.substr(oldcp, cp - oldcp));
98.             oldcp = cp + 1;}
99.       v.push_back(s.substr(oldcp));}
100.std::istream &getline(std::istream &i, std::vector<char> &v, char
  delim){
101.      if(v.size() < 32) v.resize(32);
102.      v[0] = 0;
103.      size_t vp = 0;
104.      if(i.peek() == delim){i.get();return i;}
105.      while(i.get(&v[vp], v.size() - vp, delim)){
106.                if(i.peek() == delim){i.get();break;}
107.                else{vp = v.size() - 1;
108.                     v.resize(v.size() * 2);}}
109.      return i;}
110.const double BaseFileReader::dnan = sqrt(-1.0);
111.const float BaseFileReader::fnan = float(sqrt(-1.0));
112.BaseFileSection::BaseFileSection(){startLine = dataStartLine = -1; }
113.bool BaseFileSection::findIntOpt(const std::string &name, int &ret){
114.      header_t::iterator i = headers.find(name);
115.      if(i == headers.end()) return false;
116.      ret = atoi((*i).second.c_str());
117.      return true;}
118.bool BaseFileSection::findStringOpt(const std::string &name,
  std::string &ret){
119.      header_t::iterator i = headers.find(name);
120.      if(i == headers.end()) return false;
121.      ret = (*i).second;
122.      return true;}
123.bool BaseFileSection::findFieldList(const std::string &name,
  FieldList &mapping){
124.      header_t::iterator i = headers.find(name);
125.      if(i == headers.end()) return false;
126.      mapping.clear();
127.      char buf[(*i).second.size()+1];
128.      strcpy(buf, (*i).second.c_str());
129.      char *tok = strtok(buf, "\t");
130.      while(tok){int s = mapping.size();
131.                mapping[std::string(tok)] = s;
132.                tok = strtok(NULL, "\t");}
133.      return true;}
134.bool BaseFileSection::findFieldPos(const std::string &name, int
  &ret,FieldList &mapping){
135.      FieldList::iterator i = mapping.find(name);
136.      if(i == mapping.end()) return false;
137.      ret = (*i).second;
138.      return true;}
139.#ifdef HAVE_PUBSETBUF
140.const int BaseFileReader::inbuflen = 4096;
141.#else
142.const int BaseFileReader::inbuflen = 0;
143.#endif
144.const char *BaseFileReader::errtexts[] = {"No error", "File I/O
  error", "Bad file format", "Too many fields", "Too few fields",
```

```
  "Wrong file type", "Bad section specification",
  "invalid error"};
145.BaseFileReader::BaseFileReader(std::istream &s, bool changebuf)
  :str(s){
146.       line = 0;
147.       _error = EOR;
148.       dataEnd = true;
149.       _firstSection = true;
150.#ifdef HAVE_PUBSETBUF
151.       if(changebuf){
152.                 inbuf.resize(inbuflen);
153.                 s.rdbuf()->pubsetbuf(&inbuf[0], inbuflen);}
154.#endif}
155.BaseFileReader::~BaseFileReader(){
156.#ifdef HAVE_PUBSETBUF
157.       if(inbuf.size() > 0) str.rdbuf()->pubsetbuf(NULL, 0);
158.#endif}
159.bool BaseFileReader::readLine(){
160.       line++;
161.       getline(str, buf);
162.       if(!str.good()){
163.                 _error = str.bad() ? FILEERR : EOR;
164.                 dataEnd = true;
165.                 return false;}
166.       if(!buf[0]) dataEnd = true;
167.       return true;}
168.bool BaseFileReader::readHeadersNew(BaseFileSection::header_t
  &nameValue){
169.       dataEnd = false;
170.       nameValue.clear();
171.       if(_firstSection){
172.                 if(!readLine()) return false;
173.                 char *bp = &buf[0];
174.                 if(*bp == '#') bp++;
175.                 if(strcmp(bp, "BASEfile")){
                       _error = BADFORMAT;return false;}
176.                 _firstSection = false;}
177.       if(!readLine()) return false;
178.       while(!buf[0]){
179.                 if(!readLine()) return false;}
180.       char *bp = &buf[0];
181.       if(*bp == '#') bp++;
182.       while(buf[0] && strcmp(bp, "%")){
183.                 char *tab = strpbrk(bp, "\t");
184.                 if(tab){
185.                      *tab = 0;
186.                      nameValue[std::string(bp)] =
  std::string(&tab[1]);}
187.                 else nameValue[std::string(bp)] = "";
188.                 if(!readLine()) return false;
189.                 bp = &buf[0];
190.                 if(*bp == '#') bp++;}
191.       return true;}
192.bool BaseFileReader::skipData(){
193.       if(!dataEnd){
194.                 if(!readLine()) return false;
195.                 while(!dataEnd){
196.                           if(!readLine()) return false;}}
197.       return true;}
198.bool BaseFileReader::readSection(BaseFileSection &sect){
199.       skipData();
200.       sect.startLine = line + 1;
201.       if(!readHeadersNew(sect.headers)) return false;
202.       sect.dataStartLine = line + 1;
203.       BaseFileSection::header_t::iterator i =
  sect.headers.find("section");
204.       if(i == sect.headers.end() || (*i).second.empty()){
```

```
205.                 _error = BADSECTION;
206.                 return false;}
207.       sect.type = (*i).second;
208.       return true;}
209.bool BaseFileReader::readLines(std::vector<std::string> &vec){
210.       dataEnd = false;
211.       if(!readLine()) return false;
212.       while(!dataEnd){
213.                 vec.push_back(&buf[0]);
214.                 if(!readLine()) return false;}
215.       return true;}};
```

Analysis of basefile.cc

FindIntOpt: It looks for the presence of a given field name in a section and if it is successful, it will give the attribute of the field as an integer in "ret".

FindStringOpt: It looks for the presence of a given field name in a section and if it is successful, it will give the attribute of the field as a string in "ret".

FindFieldList: It looks for the presence of a given field in a section and if it is successful, it will give the entire list of attributes in "mapping". It is used where there are more than one attributes.

FindFieldPos: It looks for the presence of a given attribute in a given map of fields or attributes, and if it is successful, it will give the position of the attribute in "ret". It is likely to be used after findFieldList function.

ReadSection: It calls readHeadersNew function and checks for the presence of a "section" field and the presence of its value. The value of "section" field is written into the "type" variable of BaseFileSection.

ReadHeadersNew: This is the main function that reads a section of the BASEfile file into a BaseFileSection class, given to it as "nameValue". If it is the first section, it checks for the presence of the string "BASEfile" as the first line, then, it will read every line until it encounters a blank line or "%". For every line that it reads, it will ignore "#" character, and insert the <field> <attribute> pair into "nameValue".

ReadLines: This function reads a multi-attribute line and puts the list of attribute into a vector, "vec".

Source Codes of basefile_plugin.h

```
216.#ifndef _BASEFILE_PLUGIN_H
217.#define _BASEFILE_PLUGIN_H
218.#include <basefile.h>
219.namespace BASE{
220.class PluginBaseFileReader: BaseFileReader{
221.  public:
222.       typedef std::map<std::string, std::string> ssmap;
223.       typedef std::map<std::string, int> simap;
224.       typedef std::vector<std::string> strvec;
225.       typedef std::vector<int> intvec;
226.       struct Assay         {
```

```
227.                int id;
228.                std::string name;
229.                strvec annots;};
230.      PluginBaseFileReader(std::istream &s);
231.      ~PluginBaseFileReader();
232.      bool registerSettings(const std::string &sectionname,
233.          const std::string &pluginversion, const strvec &s);
234.      const std::string &getSetting(const std::string &name);
235.      int getSettingInt(const std::string &name);
236.      double getSettingDouble(const std::string &name);
237.      enum { format_any, format_serial, format_normal };
238.      typedef int format_t;
239.      bool registerDataFormat(const strvec &columns, const strvec
  &fields,bool allIntensities = true, int minCh = 0, int maxCh =
  0,format_t format = format_any);
240.      bool readInfoSections();
241.      bool readSpotsSection();
242.      int getAssayCount() { return _dataassays.size(); }
243.      const Assay &getAssay(int assay);
244.      int getAnnotationCount() { return _annottypes.size(); }
245.      const std::string &getAnnotationName(int annot);
246.      int getChannelCount() { return _channels; }
247.      bool readDataLine();
248.      const char *getColumn(const std::string &name);
249.      std::string getColumnString(const std::string &name);
250.      int getColumnInt(const std::string &name);
251.      double getColumnDouble(const std::string &name);
252.      const char *getField(const std::string &name, int assay);
253.      std::string getFieldString(const std::string &name, int
  assay);
254.      int getFieldInt(const std::string &name, int assay);
255.      double getFieldDouble(const std::string &name, int assay);
256.      double getIntensity(int assay, int channel);
257.      enum{ MISSINGSETTINGS = 100, BROKENSETTINGS, MISSINGASSAYS,
  BROKENASSAYS, MISSINGSPOTS, BROKENSPOTS, BADSPOTS, BADVERSION,
  BADCALLSEQUENCE, BADINDEXARGUMENT, LASTERR};
258.      const char *errText(int e);
259.      const char *errText() { return errText(_error); }
260.      int getError() { return _error; }
261.      std::string errTextMore() { return _errdescr; };
262.  private:
263.      std::string toString(int i);
264.      ssmap _settings;
265.      std::string _settingstype, _pluginversion;
266.      strvec _annottypes;
267.      std::map<int, int> _assaynums;
268.      std::vector<Assay> _assays;
269.      strvec _reqcolumns, _reqfields;
270.      bool _reqallint, _reqserial, _reqnormal;
271.      int _channelsmin, _channelsmax;
272.      bool _datafmtregged;
273.      int _channels;
274.      int _spotssection;
275.      int _assayDatacolumn;
276.      intvec _intensityfields;
277.      simap _datacolumns, _datafields;
278.      intvec _dataassays;
279.      std::vector<char *> _databuf;
280.      static const char *_plerrtexts[];
281.      std::string _errdescr;
282.      static const std::string _undefstr;
283.      static const Assay _undefassay;};};
284.#endif
```

Analysis of basefile_plugin.h

PluginBaseFileReader is being defined as a desendent class from BaseFileReader class. It declares four data types, ssmap, simap, strvec and intvec, as string-to-string map, string-to-integer map, string vector, and integer vector respectively. In order to handle array data, an "Assay" data structure is created, consisting of "id" as integer, "name" as string and a vector of string, "annots".

PluginBaseFileReader class declares twenty-two public functions:
4. registerSettings,
5. getSetting,
6. getSettingInt,
7. getSettingDouble,
8. registerDataFormat,
9. readInfoSections,
10. readSpotsSection,
11. getAssayCount,
12. getAssay,
13. getAnnotationCount,
14. getAnnotationName,
15. getChannelCount,
16. readDataLine,
17. getColumn,
18. getColumnString,
19. getColumnInt,
20. getColumnDouble,
21. getField,
22. getFieldString,
23. getFieldInt,
24. getFieldDouble, and
25. getIntensity.

Source Codes of basefile_plugin.cc

```
285.#ifdef HAVE_CONFIG_H
286.#include <config.h>
287.#endif
288.#include <cstring>
289.#include <cstdlib>
290.#include <cstdio>
291.#include <iostream>
292.#include <basefile_plugin.h>
293.namespace BASE{
294.const std::string PluginBaseFileReader::_undefstr = "undefined";
295.const PluginBaseFileReader::Assay PluginBaseFileReader::_undefassay
  ={ 0, "undefined" };
296.const char *PluginBaseFileReader::_plerrtexts[] = {
297.      "Expected settings section", "Missing a setting in settings
  section","Expected 'assays' section", "Malformatted 'assays'
  section","No 'spots' section found in file", "Malformatted 'spots'
  section","Improper 'spots' section for this plugin","Version mismatch
  between plugin and data file","Bug in plugin: bad sequence of
  function calls","Bug in plugin: Bad argument in data retrival
  function"};
298.PluginBaseFileReader::PluginBaseFileReader(std::istream &s):
299.      BaseFileReader(s){}
300.PluginBaseFileReader::~PluginBaseFileReader(){}
```

```
301.const char *PluginBaseFileReader::errText(int e){
302.       if(e < MISSINGSETTINGS || e >= LASTERR)
              return BaseFileReader::errText(e);
303.       return _plerrtexts[e - MISSINGSETTINGS];}
304.bool PluginBaseFileReader::registerSettings(
305.       const std::string &sectionname,
306.       const std::string &pluginversion,
307.       const strvec &s){
308.       _settings.clear();
              for(strvec::const_iterator i = s.begin(); i != s.end();
              i++) {_settings[*i] = "";}
309.       _settingstype = sectionname;
310.       _pluginversion = pluginversion;
311.       return true;}
312.const std::string &PluginBaseFileReader::getSetting(const
  std::string &name){return _settings[name];}
313.int PluginBaseFileReader::getSettingInt(const std::string
  &name){return atoi(_settings[name].c_str());}
314.double PluginBaseFileReader::getSettingDouble(const std::string
  &name){return atof(_settings[name].c_str());}
315.bool PluginBaseFileReader::registerDataFormat(const strvec
  &columns,const strvec &fields, bool allIntensities, int minCh, int
  maxCh,format_t format){
316.       _reqcolumns = columns;
317.       _reqfields = fields;
318.       _reqallint = allIntensities;
319.       _reqserial = format == format_serial;
320.       _reqnormal = format == format_normal;
321.       _channelsmin = minCh;
322.       _channelsmax = maxCh;
323.       _datafmtregged = true;
324.       return true;}
325.bool PluginBaseFileReader::readInfoSections(){
326.       if(!isFirstSection()){
327.                  _error = BADCALLSEQUENCE;
328.                  _errdescr =
  "PluginBaseFileReader::readInfoSections() "+"after reading was
  started";return false;}
329.       BaseFileSection sect;
330.       if(_settingstype != ""){
331.            if(!readSection(sect)){
332.                 if(!_error) _error = MISSINGSETTINGS;
333.                 return false;}
334.                 if(sect.type != _settingstype){
335.                 _error = MISSINGSETTINGS;
336.                 _errdescr = std::string("Missing section '") +
  _settingstype + "'";return false;}
337.                 for(ssmap::iterator i = _settings.begin(); i !=
  _settings.end(); i++){
338.                 BaseFileSection::header_t::iterator j =
339.                           sect.headers.find((*i).first);
340.                 if(j == sect.headers.end()     {
341.                           _error = BROKENSETTINGS;
342.                           _errdescr = "Missing setting '" +
  (*i).first + "'";return false;}
343.                 (*i).second = (*j).second;}
344.                 _settings.swap(sect.headers);
345.                 if(_pluginversion != "" &&
346.                      getSetting("pluginVersion") !=
  _pluginversion){
347.                           _error = BADVERSION;
348.                           _errdescr = "Plugin executable is
  version " +_pluginversion + " but the input file is for " +"version "
  + getSetting("pluginVersion");
349.                           return false;}}
350.       if(!readSection(sect) || sect.type != "assays"){
351.                 if(!_error) _error = MISSINGASSAYS;
```

```
352.                 return false;}
353.       FieldList columns, annotationColumns;
354.       if(!sect.findFieldList("columns", columns) ||
355.            !sect.findFieldList("annotationColumns",
  annotationColumns)){
356.                 _error = BROKENASSAYS;
357.                 _errdescr = "Missing header 'columns' and/or
  'annotationColumns'";
358.                 return false;}
359.       int idCol, nameCol;
360.       if(!sect.findFieldPos("id", idCol, columns) ||
361.            !sect.findFieldPos("name", nameCol, columns)){
362.            _error = BROKENASSAYS;
363.            _errdescr = "Missing column 'id' and/or 'name'";
364.            return false;}
365.       for(FieldList::iterator i = annotationColumns.begin();
366.            i != annotationColumns.end(); i++){
367.            _annottypes.push_back((*i).first);
368.            int acol;
369.            if(!sect.findFieldPos((*i).first.c_str(), acol,
  columns)){
370.               _error = BROKENASSAYS;
371.               _errdescr = "Missing an annotation column ('" +
372.               (*i).first + "') in 'columns'";
373.               return false;}}
374.       strvec cols(columns.size());
375.       while(readVec(cols)){
376.                 Assay a;
377.                 a.id = atoi(cols[idCol].c_str());
378.                 a.name = cols[nameCol];
379.                 for(FieldList::iterator i =
  annotationColumns.begin(); i != annotationColumns.end(); i++){
380.                           a.annots.push_back(cols[columns[(*i).fir
  st]]);}
381.                 _assaynums[a.id] = _assays.size();
382.                 _assays.push_back(a);}
383.       if(getError()) return false;
384.       return true;}
385.bool PluginBaseFileReader::readSpotsSection(){
386.       if(_assays.empty()){
387.                 _error = BADCALLSEQUENCE;
388.                 _errdescr =
  "PluginBaseFileReader::readSpotsSection() "+"must be preceded by
  readInfoSections()";return false;}
389.       else if(!_datafmtregged){
390.                 _error = BADCALLSEQUENCE;
391.                 _errdescr =
  "PluginBaseFileReader::readSpotsSection() "="must be preceded by
  registerDataFormat()";return false;}
392.       BaseFileSection sect;
393.       while(1){
394.            if(!readSection(sect)){
395.            if(!_spotssection && _error == EOR) _error =
  MISSINGSPOTS;return false;}
396.       if(sect.type == "spots") break;
397.       std::cerr << "Warning: Unknown section of type '" <<
  sect.type <<"' skipped in
  PluginBaseFileReader::readSpotsSection()\n";}
398.       _spotssection++;
399.       FieldList assays;
400.       if(!sect.findFieldList("columns", _datacolumns) ||
401.            !sect.findFieldList("assayFields", _datafields) ||
402.            !sect.findFieldList("assays", assays)){
403.                 _error = BROKENSPOTS;
404.                 _errdescr = "Missing header 'columns' or
  'assayFields' or 'assays'";return false;}
405.       if(!BaseFileSection::findFieldPos("assayData",
```

```
   _assayDatacolumn,_datacolumns)){
406.                 _error = BROKENSPOTS;
407.                 _errdescr = "Missing column 'assayData'";
408.                 return false;}
409.       if(_reqserial && assays.size() != 1){
410.            _error = BADSPOTS;
                _errdescr = "Serial format (one assay per spots
          section) required";return false;}
411.       if(_reqnormal && assays.size() != _assays.size()){
412.           _error = BADSPOTS;
413.           _errdescr = "Normal format (all assays in one spots
   section) required";return false;}
414.       if(!sect.findIntOpt("channels", _channels)){
415.                 _error = BROKENSPOTS;
416.                 _errdescr = "Missing 'channels' header";
417.                 return false;}
418.       if(_channelsmin > 0 && _channels < _channelsmin ||
419.            _channelsmax > 0 && _channels > _channelsmax){
420.            _error = BADSPOTS;
421.            _errdescr = "The number of channels isn't in [" +
                              toString(_channelsmin) + ", " +
422.                 (_channelsmax > 0 ? toString(_channelsmax) :
                              std::string("inf")) +"]";
423.                 return false;}
424.       for(strvec::iterator i = _reqcolumns.begin(); i !=
   _reqcolumns.end(); i++) {
425.          int col;
426.          if(!BaseFileSection::findFieldPos((*i).c_str(), col,
                                        _datacolumns)){
427.             _error = BADSPOTS;
428.             _errdescr = "Missing required column '" + (*i) +
   "'";return false;}}
429.       for(strvec::iterator i = _reqfields.begin(); i !=
   _reqfields.end(); i++){
430.            int field;
431.            if(!BaseFileSection::findFieldPos((*i).c_str(), field,
   _datafields)){
432.                 _error = BADSPOTS;
433.                 _errdescr = "Missing required field '" + (*i) +
   "'";return false;}}
434.       if(_reqallint){
435.            for(int ch = 1; ch <= _channels; ch++){
436.                int field;
437.                std::string fname = std::string("intensity") +
   toString(ch);
438.              if(!BaseFileSection::findFieldPos(fname.c_str(),
   field, _datafields)){
439.                   _error = BADSPOTS;
440.                   _errdescr = "Missing required intensity field '"
   + fname + "'";return false;}}}
441.       _dataassays.clear();
442.       for(FieldList::iterator i = assays.begin(); i !=
   assays.end(); i++){
443.            int aid = atoi((*i).first.c_str());
444.            std::map<int, int>::iterator j = _assaynums.find(aid);
445.            if(j == _assaynums.end()){
446.                 _error = BADSPOTS;
447.                 _errdescr = "Reference to unknown assay '" +
   (*i).first + "'";return false;}
448.            _dataassays.push_back((*j).second);}
449.       _intensityfields.resize(_channels);
450.       for(int ch = 1; ch <= _channels; ch++){
451.            simap::iterator i = _datafields.find(
452.            std::string("intensity") + toString(ch));
453.            if(i == _datafields.end()) {_intensityfields[ch - 1] =
   -1;}
454.            else {_intensityfields[ch - 1] = (*i).second;}}
```

```
455.      _databuf.resize(_datacolumns.size() - 1 +
456.                  _dataassays.size() * _datafields.size());
457.      return true;}
458.const PluginBaseFileReader::Assay
  &PluginBaseFileReader::getAssay(int assay){
459.      if(assay < 0 || assay >= (int)_dataassays.size()){
460.                  _error = BADINDEXARGUMENT;
461.                  _errdescr = "Arg to
  PluginBaseFileReader::getAssay() outside range: " + toString(assay) +
  " not in [0, " + toString(_dataassays.size()+")";return _undefassay;}
462.      return _assays[_dataassays[assay]];}
463.const std::string &PluginBaseFileReader::getAnnotationName(int
  annot){
464.      if(annot < 0 || annot >= (int)_annottypes.size()){
465.                  _error = BADINDEXARGUMENT;
466.                  _errdescr = "Arg to
  PluginBaseFileReader::getAnnotationName() ""outside range: " +
  toString(annot) + " not in [0, " + toString(_annottypes.size())  +
  ")";return _undefstr;}
467.      return _annottypes[annot];}
468.bool PluginBaseFileReader::readDataLine(){
469.      return readVec(_databuf);}
470.const char *PluginBaseFileReader::getColumn(const std::string
  &name){
471.      simap::iterator i = _datacolumns.find(name);
472.      if(i == _datacolumns.end()){
473.                  _error = BADINDEXARGUMENT;
474.                  _errdescr = "Bad arg to
  PluginBaseFileReader::getColumn(): '" + name + "' is not a column in
  this file";return _undefstr.c_str();}
475.      return _databuf[(*i).second];}
476.std::string PluginBaseFileReader::getColumnString(const std::string
  &name){
477.      return std::string(getColumn(name));}
478.int PluginBaseFileReader::getColumnInt(const std::string &name){
479.      return atoi(getColumn(name));}
480.double PluginBaseFileReader::getColumnDouble(const std::string
  &name){
481.      const char *d = getColumn(name);
482.      if(!*d) return dnan;
483.      return atof(d);}
484.const char *PluginBaseFileReader::getField(const std::string &name,
  int assay){
485.      if(assay < 0 || assay >= (int)_dataassays.size()){
486.                  _error = BADINDEXARGUMENT;
487.                  _errdescr = "Bad assay number (" + toString(assay)
  +") in PluginBaseFileReader::getField()";
488.                  return _undefstr.c_str();}
489.      simap::iterator i = _datafields.find(name);
490.      if(i == _datafields.end()){
491.                  _error = BADINDEXARGUMENT;
492.                  _errdescr = "Undefined field '" + name +"' in
  PluginBaseFileReader::getField()";
493.                  return _undefstr.c_str();}
494.      return _databuf[(*i).second + _assayDatacolumn +
495.                  assay * _datafields.size()];}
496.std::string PluginBaseFileReader::getFieldString(const std::string
  &name, int assay){
497.      return std::string(getField(name, assay));}
498.int PluginBaseFileReader::getFieldInt(const std::string &name, int
  assay){
499.      return atoi(getField(name, assay));}
500.double PluginBaseFileReader::getFieldDouble(const std::string &name,
  int assay){
501.      const char *d = getField(name, assay);
502.      if(!*d) return dnan;
503.      return atof(d);}
```

```
504.double PluginBaseFileReader::getIntensity(int assay, int channel){
505.       if(assay < 0 || assay >= (int)_dataassays.size()){
506.                  _error = BADINDEXARGUMENT;
507.                  _errdescr = "Bad assay number (" + toString(assay)
  +") in PluginBaseFileReader::getIntensity()";
508.                  return dnan;}
509.       if(channel < 1 || channel > _channels){
510.                  _error = BADINDEXARGUMENT;
511.                  _errdescr = "Bad channel number (" +
  toString(channel) +") in PluginBaseFileReader::getIntensity()";
512.                  return dnan;}
513.       if(_intensityfields[channel - 1] < 0){
514.                  _error = BADINDEXARGUMENT;
515.                  _errdescr = "'intensity" + toString(channel)+ "'
  not in field list in PluginBaseFileReader::getIntensity()";
516.                  return dnan;}
517.       const char *d = _databuf[_intensityfields[channel - 1]+
518.            _assayDatacolumn + assay * _datafields.size()];
519.       if(!*d) return dnan;
520.       return atof(d);}
521.std::string PluginBaseFileReader::toString(int i){
522.       static char buf[32];
523.       std::sprintf(buf, "%d", i);
524.       return std::string(buf);}};
```

Analysis of basefile_plugin.cc

basefile_plugin.cc implements the functions declared by PluginBaseFileReader class.

RegisterSettings: This function does three things, initialize "_setting" (ssmap data type, see line 264) to the hold the number of fields as pre-emp by "s". Secondly, it initiates the type of settings into "_settingtype". Lastly, it initiates the version of plug in into "_pluginversion".

GetSetting, GetSettingInt, GetSettingDouble: Returns the setting of the given field in string, integer and double respectively.

RegisterDataFormat: It reads its parameters into a set of defined variables, in preparation for usage in the plugin(s).

ReadInfoSections: This is the function that reads the first section of a BASEfile file, after being verified. Hence, it takes over at typically the second line of the BASEfile file, after the "BASEfile" line. It contains six main checking routines to check for the presence of a section, the settings of a section, the compatibility of BASEfile file to plugin, the presence of data (as in assay), the presence of column(s), and the presence of either a name or id. It reads the section settings (the <field> <attribute> pairs) into "_settings", annotation column fields into "_annottypes", and merges the column names and annotated column names into "annots" in "array" data type.

ReadSpotsSection: This function reads the actual assay data into the data elements after performing five major checks, namely, the presence of a spots section, the number of field are correct, the data format is correct (serial or normal), the presence of channel(s), and the number fo channel(s) are correct. If there are more than a single channel, the

intensities of each channel are listed as intensity1, intensity2, and so on. Data of a particular spot are read into two variables, "_dataassay", which holds information about a spot, and "_intensityfields", which holds values for each channel.

GetAssay: This function retrieves data for a specific spot.

GetAnnotationName: This function retrieves name of an annotation column from the list of pre-defined annotated column.

ReadDataLine: This function checks for the presence of data is the "readVec".

GetColumn: This function searches for a column of the given name in the "_dataassay", and reads the immediate data of the column.

GetColumnString, GetColumnInt, GetColumnDouble: These functions call the "GetColumn" function and returns the data in appropriate data types.

GetField: This function searches for a field of a given name in the "_dataassay", and returns the attribute of the field, the length of the data column and the number of attributes.

GetFieldString, GetFieldInt, GetFieldDouble: These functions call the "GetField" function and returns the data in appropriate data types.

GetIntensity: This function retrieves the intensity value for a given channel and row of data.

Source Codes of microarr_point.hh

```
525.#ifndef MYPLUGIN_MICROARRAY_POINT_HH
526.#define MYPLUGIN_MICROARRAY_POINT_HH
527.#include <cstdlib>
528.struct microarr_point{
529.  int index;
530.  double R, G, ratio, intens;
531.  microarr_point(int i, double r, double g):
532.    index(i), R(r), G(g),
533.    ratio((R>0 && G>0) ? R/G : -1.),
534.    intens((R>0 && G>0) ? sqrt(R*G) : -1.){}
535.  double maxF(){return std::max(R, G);}
536.  double minF(){return std::min(R, G);}};
537.template <double microarr_point:: *member>
538.struct microarr_point_ptr_m_less:
539.  public std::binary_function<microarr_point*, microarr_point*,
  bool>{
540.  bool operator()(microarr_point *a, microarr_point *b)
541.  {return a->*member < b->*member;}};
542.template <double (microarr_point:: *member)()>
543.struct microarr_point_ptr_less:
544.  public std::binary_function<microarr_point*, microarr_point*,
  bool>{
545.  bool operator()(microarr_point *a, microarr_point *b)
546.  {return (a->*member)() < (b->*member)();}};
547.#endif
```

Analysis of microarr_point.hh

It declares a "microarr_point" structure which holds data for a spot on an array and two functions for binary comparison of two microarr_point structures.. Upon instantiation, the "ratio" and "intens" are calculated automatically from the input values for green (G) and red (R) channels.

The two functions are "microarr_point_ptr_m_less" and "microarr_point_ptr_less". Both functions takes in two points, a and b, and returns a boolean value for a < b. The difference between these two functions is completely implementationally specific.

Source Codes of myplugin.cc

```
548.#ifdef HAVE_CONFIG_H
549.#include <config.h>
550.#endif
551.#include <cstdlib>
552.#include <cstring>
553.#include <cstdarg>
554.#include <iostream>
555.#include <map>
556.#include <vector>
557.#include <algorithm>
558.#include <basefile_plugin.h>
559.#include "microarr_point.hh"
560.using namespace std;
561.using namespace BASE;
562.vector<string> makesvec(char *str, ...);
563.bool printErr(PluginBaseFileReader &pbfr);
564.bool normalize(PluginBaseFileReader& pbfr);
565.char *progname;
566.bool dothings(){
567.       PluginBaseFileReader pbfr(cin);
568.       vector<string> settings = makesvec("some_option",NULL);
569.       if(!pbfr.registerSettings("myplugin settings",
  PLUGIN_VERSION, settings))
570.       return printErr(pbfr);
571.       vector<string> columns = makesvec("position", "reporter",
  NULL);
572.       vector<string> fields = makesvec(NULL);
573.       if(!pbfr.registerDataFormat(columns, fields, true, 2,
  2,PluginBaseFileReader::format_serial)){return printErr(pbfr);}
574.       if(!pbfr.readInfoSections()) return printErr(pbfr);
            cout << "BASEfile\n";
575.       while(pbfr.readSpotsSection()){
576.                 cout <<"section\tspots\n" +"columns\tposition
  \treporter\tassayData\n"+"assayFields\tintensity1\tintensity2\n"+"ass
  ays\t"<< pbfr.getAssay(0).id <<"\n" +"%\n";
577.                 if(!normalize(pbfr)) return false;
578.                 cout << "\n";}
579.       return pbfr.getError() ? printErr(pbfr) : true;}
580.bool normalize(PluginBaseFileReader &pbfr){
581.       double something =pbfr.getSettingDouble("some_option");
582.       if(something < 0. || something > 1.){
583.            cerr << progname << ": Invalid something setting (" <<
  something << "). Please make sense." << endl;
584.         return false;}
585.       vector<microarr_point> data;
586.       typedef vector<microarr_point*>::iterator microarr_ptr_iter;
587.       map<int,int> posRep;
588.       while(pbfr.readDataLine()){
589.                 int pos = pbfr.getColumnInt("position");
```

```
590.                 data.push_back(microarr_point(pos,pbfr.
  getIntensity(0, 1), pbfr.getIntensity(0, 2)));
591.                 posRep[pos] = pbfr.getColumnInt("reporter");}
592.       for(vector<microarr_point>::iterator iter = data.begin();iter
  != data.end(); iter++){
593.                 cout << iter->index << "\t" << posRep[iter->index]
  << "\t" <<iter->R << "\t" <<iter->G << "\n";}
594.       return true;}
595.int main(int argc, char **argv){
596.#ifdef HAVE_PUBSETBUF
597.       char coutbuf[4096];
598.       cout.rdbuf()->pubsetbuf(coutbuf, sizeof(coutbuf));
599.#endif
600.       if(argc) {progname = argv[0];}
601.       else {progname = "unknown";}
602.       int ok = dothings();
603.#ifdef HAVE_PUBSETBUF
604.       cout.rdbuf()->pubsetbuf(NULL, 0);
605.#endif
606.       return !ok;}
607.vector<string> makesvec(char *str, ...){
608.       va_list va;
609.       va_start(va, str);
610.       vector<string> vec;
611.       if(str){
612.           vec.push_back(string(str));
613.           char *s;
614.           while((s = va_arg(va, char *)))
  vec.push_back(string(s));}
615.       va_end(va);
616.       return vec;}
617.bool printErr(PluginBaseFileReader &pbfr){
618.       if(pbfr.getError()){
619.           cerr << progname << ": " << pbfr.errText() << endl;
620.           string s = pbfr.errTextMore();
621.           if(!s.empty()) cerr << "  " << s << endl;}
622.       else {cerr << progname << ": " << "unknown error" << endl;?}
623.       return false;}
```

Analysis of myplugin.cc

This file is included in BASE 1.x distribution as a sample and prototype for plug-in developments, meaning that it is possible to gain deeper insights into BASEfile file handlers as it will demonstrate the file handlers in operation.

The execution entry point is at line 595, with the "main" statement, which calls the "dothings" function at line 566 and instantiate a BaseFileReader via PluginBaseFileReader. After which, the "registerDataFormat", "readInfoSection" and "readSpotsSection" will be called sequentially at lines 573, 574 and 575 respectively. Line 577 hands over the execution to "normalize" function which does the actual work of the plugin.

As a sample, "normalize" function does not normalize anything but it got some of hte data in the format that is easy to handle. Lines 581 to 584 reads the settings and checks for the presence of a correct setting. A vector of microarray points and a map of integer to integer are created at line 585 and 587 respectively. By calling "readDataLine" function, attribute of the "position" field is read into "pos", which is used as a reference to obtain values for the intensities (red and green) of the spot on the array. Every spot is read by repeatedly calling "readDataLine" function. Spot intensities are stored in "data"

whereas the corresponding reporter (as integer) is stored in "posRep". After this stage, actual manipulations and computations can happen. If everything goes well, "normalize" function will return a true value to "dothings" function, which will return a true value to the "main", signifying that the process is successful.

Along the way, all errors will be piped to the standard error output.

General Mode of Operation

An input file is created by BASE when data source(s) are selected and the plug in is initiated. This input file will take the name of "stdin.txt". The plugin controller script, "jobController.php", will feed "stdin.txt" into the plugin and execute the plugin.

Four sequential events must happen when the plugin executes, in order to get the data from "stdin.txt" into the appropriate data structures. Firstly, the BaseFileReader is instantiated via PluginBaseFileReader. Since "stdin.txt" represents the standard input (the keyboard), this file is fed to the BaseFileReader as the console input (another name for standard input), the "cin". Secondly, "registerDataFormat" routine kicks in to allocate the necessary data structures as the volume fo data for every execution of a plugin varies widely. Thirdly, "readInfoSection" routine is activated to read all sections before the spots section, which is essentially the section(s) holding specific information of the data. These data may be required by the plugin to carry out its tasks. Lastly, the spots' data are read via "readSpotsSection" routine. Although these four events happens within the plugin, it is a set of standard tasks required for nearly every plugin.

Within the plugin, it is free to create any files and sub-directories needed and to use any publically declared functions. All error codes that are reported via "stderr" (the standard error output), will be logged into "stderr.txt" file.

After the plugin terminates, the job controller will scan the directory recursively, logs all files produced into BASE as result files. In event where it recogizes a BASEfile file, it will write the data into BASE, after which, the job controller will clear the directory, in preparation for the next execution of plugin.

Proposed Specification of BASEfile File Handlers

A BASEfile file format can be defined as following form by extending Backus-Naur Form to cater for conditional terms,

```
basefile ::= <file_header> "\n" {<section>}
file_header ::= ["#"] "BASEfile"
section ::= ["#"] "section \t" <section_attribute> "\n"
                {<field_attribute_pair>} "\n"
                ["% \n" {<data_row>} "\n\n"]

section_attribute ::= <literals>
field_attribute_pair ::= ["#"] <fieldname>  "\t" {<field_attribute>}
fieldname ::= <literals>
field_attribute ::= {<literals> "\t"} <layered_attribute>
```

layered_attribute ::= "name" | "assayData"
$\exists_1$ section_attribute • section
$\exists_1$ fieldname • field_attribute_pair

If ($\exists$ name • layered_attribute) Then (<columns> $\in$ fieldname)
If ($\exists$ name • layered_attribute) Then (<annotate> $\in$ fieldname)
If ($\exists$ assayData • layered_attribute) Then (<assayF> $\in$ fieldname)
If ($\exists$ assayData • layered_attribute) Then (<channel> $\in$ fieldname)

columns ::= "columns \t" <column_attribute>
column_attribute ::= <literals>
If ($\exists$ name • layered_attribute) Then ("name" $\in$ column_attribute)
annotate ::= "annotationColumns \t" {<annotate_attribute>}
annotate_attribute ::= <literals>
assayF ::= "assayFields \t" {<assay_attribute>}
assay_attribute ::= <literals>
channel ::= "channel \t" <number_of_channels>
number_of_channels ::= integer
data_row ::= {<data_attribute> "\t"}
data_attribute ::= {<literals> | <integer>}
If ($\exists$ name • layered_attribute) Then (Ω (annotate_attribute • annotate +
column_attribute • column -1) =
# data_attribute • data_rows)
If ($\exists$ assayData • layered_attribute) Then (Ω (assay_attribute • assayF +
column_attribute • column -1) =
# data_attribute • data_rows)
If ($\exists$ assayF • field_attribute_pair) Then
(Ω assay_attribute • assayF = number_of_channels)

literals ::= {"A"..."Z" | "a"..."z" | "0"..."9" | "." | "," | "(" | ")" | "`" | "~" | "!" | "@" | "#" |
"$" | "%" | "^" | "&" | "*" | "_" | "-" | "+" | "=" | "{" | "}" | "|" | "[" | "]" | "\" | ":" |
"'" | "<" | ">" | "?" | ";" | "'" | "/" | " " }
integer ::= {"0"..."9"}

For the purpose of specification in Z, a BASEfile file can be specified as,

BASEfile = [<header,
{<field, {<attribute>}},
{{<data>}} >]

Subjected to the restrictions stated in Backus-Naur Form above.

At BASEfile >> BASEfileReader,

HeaderCheck = [$\exists!$ BASEfile |
$\exists$ (header = "BASEfile") = true]

$$\text{SectionHeadCheck} = [\ \Xi\ \text{BASEfile} \mid (\ \exists\ (\text{field} = \text{"section"}) = \text{true}) \wedge \exists\ \text{attribute}_{\text{section}}\ ]$$

$$\text{SectionRegister} = [\ \{\langle\text{field}\rangle\} \rightarrow \{\{\langle\text{attribute}\rangle\}\}\ ]$$

$$\text{ReadInfoSection} = [\ \Delta\ \text{SectionRegister} \mid \forall\ (\text{field(i)}_{\text{BASEfile}} \rightarrow \text{field(i)}_{\text{SectionRegister}}); \forall\ (\text{attribute(field(i))}_{\text{BASEfile}} \rightarrow \text{attribute(field(i))}_{\text{SectionRegister}})\ ]$$

$$\text{SpotRegister} = [\ \{\{\langle\text{spot\_info}\rangle\}\} \rightarrow \{\{\langle\text{spot\_intensity}\rangle\}\}\ ]$$

$$\text{ReadSpotsSection} = [\ \Delta\ \text{SpotRegister} \mid \forall\ (\text{column}_{\text{data}} \notin \text{assayFields}) \wedge (\ \text{"assayData"}) \Rightarrow (\text{data}_{\text{column(i), j}} \rightarrow \text{spot\_info}_{\text{column(i), j}}); \forall\ (\text{column}_{\text{data}} \in \text{assayFields}) \Rightarrow (\text{data}_{\text{column(i), j}} \rightarrow \text{spot\_intensity}_{\text{column(i), j}})\ ]$$

Therefore,

$$\text{BASEfileReader} = \text{SectionRegister} \cup \text{SpotRegister} \cup \langle \text{HeadCheck, SectionHeadCheck, ReadInfoSection}^{\text{bof()}}_{\text{eof()}}, \text{ReadSpotSection}^{\text{bof()}}_{\text{eof()}} \rangle$$

Consequently, from SectionRegister(i) and SpotRegister(i), where i = 1 to n, a BASEfileWriter can be defined as,

$$\text{HeadWriter} = [\ \text{"BASEfile \textbackslash n"} >> \text{BASEfile}\ ]$$

$$\text{WriteInfoSection} = [\ \Xi\ \text{SectionRegister} \mid \text{SectionRegister} >> \text{BASEfile}\ ]$$

$$\text{WriteSpotSection} = [\ \Xi\ \text{SpotRegister} \mid \text{SpotRegister} >> \text{BASEfile}\ ]$$

$$\text{BASEfileWriter} = [\ \Xi\ \text{SectionRegister} \wedge \text{SpotRegister} \mid \langle \text{HeadWriter, WriteInfoSection(i)}_{\text{i = 1 to n}}, \text{WriteSpotSection(i)}_{\text{i = 1 to n}} \rangle\ ]$$

Implementation of BASEFile Reader in Python

```
# basereader.py
# (c) 2004 Maurice Ling
# BSc(Hons), The University of Melbourne, Dept of Zoology
# created on 20th of January, 2004

import sys
import string

global SectionRegister
global FileHead
global SpotSectionInfo
```

```
global SpotRegisterInfo
global SpotRegisterIntensity
SectionRegister = []
FileHead = []
SpotSectionInfo = []
SpotRegister = []
SpotRegisterIntensity = []

def BaseReader(file):
          base = open(file)

          # reads 1st line of line (supposed to be "BASEfile") and
          # puts it in FileHead as a list
          str = base.readline()
          FileHead.append(str)
          str = base.readline()

          # reads 1st section (supposed to be information section),
          # until it hits a blank line
          while (str != '\n'):
                    #print str
                    str = string.split(str, '\t')
                    SectionRegister.append(str)
                    str = base.readline()

          # gets through the blank line
          str = base.readline()

          # reads information portion of 2nd section (supposed to be
spots section)
          # until it hit "%"
          str = base.readline()
          while (str != '%\n'):
                    str = string.split(str, '\t')
                    SpotSectionInfo.append(str)
                    str = base.readline()

          # reads spots data, puts everything into SpotRegister
          # extracts intensity values into SpotRegisterIntensity
          str = base.readline()
          while (str):
                    str = string.split(str, '\t')
                    SpotRegister.append(str)
                    SpotRegisterIntensity.append(str[-2:])
                    str = base.readline()

          #close the file
          base.close()

def SpotIntensityToFloat(list):
          t = []
          for s in list:
                    a = [float(s[0]), float(s[1])]
                    t.append(a)
          return t

def ClearRegister():
          del SectionRegister[0:]
          del FileHead [0:]
          del SpotSectionInfo[0:]
          del SpotRegister[0:]
          del SpotRegisterIntensity[0:]
```

Appendix B:
Implementation of pykegg

Introduction

This appendix supplements the chapter on data analysis by taking a technical view of the Kyoto Encyclopedia of Genes and Genome's Application Programming Interface (KEGG API) and its means of operation. In this project, connection and use of KEGG API is made easier by encapsulating it into a Python module, pykegg. A brief discussion into the advantages of Python as a developmental language will be discussed before listing the codes of pykegg. This appendix ends with a section on the use of pykegg. Although pykegg is Python-specific, this appendix can serve as supplementary documentation to that being provided by the maintainers of KEGG.

Kyoto Encyclopedia of Genes and Genome

Kyoto Encyclopedia of Genes and Genome (KEGG) is a set of publically-available, interconnected databases, maintained by Kyoto University and University of Tokyo, that is routinely updated from a number of other databases, including GeneBank. KEGG is available via http:// www.genome.ad.jp/KEGG.

An important feature of KEGG is that it provides a means to link pathways and the component, enzymes and substrates, that makes up the pathways. Another useful feature is that it allows for visualization of pathways, and making distinctions between reference pathways and organism-specific pathways. Both of these features makes KEGG a suitable reference database for this project.

Programmatic access to KEGG is made via the KEGG API. KEGG API is accessed on the client-end via Web Services Description Language (WSDL) component of the Simple Object Access Protocol (SOAP). WSDL will then be able to access the KEGG API. Implementationally, A SOAP client (the software on user's system) will need to call the KEGG WSDL file, at http://soap.genome.ad.jp/KEGG.wsdl, on KEGG server-end.

Why Python?

Python is chosen as the developmental language for a number of technical reasons. Firstly, it is usually faster to code in Python than in a number of other programming languages, such as, C++ and Java, making it a useful tool for prototyping, which serves the purposes of this project. Secondly, as with Java, Python is portable. Similar to Java, Python codes executes via a virtual machine. Thirdly, Python is extensible and can be embedded into Java codes, which means that in any event where there is a need to code in Java, there is a good chance that the codes developed in this project can be reused. Fourthly, Python has good text and numerical processing capabilities, which is crucial to this project. Lastly, and perhaps most importantly, Python is simpler to learn than many programming languages, such as, C. For the above reasons, one of the main users of Python programming language is in the National Aerospace Administration (NASA) mission planning control to track paths of heavenly bodies and man-made satellites.

Source Codes of pykegg

```
# KEGG.py
# (c) 2004 Maurice Ling
# BSc(Hons), The University of Melbourne, Dept of Zoology
# created on 9th of January, 2004

from SOAPpy import WSDL

wsdl = 'http://soap.genome.ad.jp/KEGG.wsdl'
serv = WSDL.Proxy(wsdl)

# KEGG Information Methods (4 methods)
def list_databases(): return serv.list_databases()
def list_organisms(): return serv.list_organisms()
def list_pathways(): return serv.list_pathways()
def dbinfo(dbname): return serv.dbinfo(dbname)

# DBGet/LinkDB Methods (2 methods)
def get_entries(entryidlist): return serv.get_entries(entryidlist)
def get_linkdb_by_entry(entryid, targetdb):
        return serv.get_linkdb_by_entry(entryid, targetdb)

# SSDB Methods (11 methods)
def get_all_neighbors_by_gene(keggid, threshold = 100, orglist = 'nil'):
        return serv.get_all_neighbors_by_gene(keggid, threshold = 100, orglist = 'nil')
def get_best_best_neighbors_by_gene(keggid, threshold, orglist = 'nil'):
        return serv.get_best_best_neighbors_by_gene(keggid, threshold, orglist = 'nil')
def get_best_neighbors_by_gene(keggid, threshold, orglist = 'nil'):
        return serv.get_best_neighbors_by_gene(keggid, threshold, orglist = 'nil')
def get_reverse_best_neighbors_by_gene(keggid, threshold, orglist = 'nil'):
        return serv.get_reverse_best_neighbors_by_gene(keggid, threshold, orglist = 'nil')
def get_paralogs_by_gene(keggid, threshold = 100):
        return serv.get_paralogs_by_gene(keggid, threshold = 100)
def get_best_homologs_by_genes(keggorg, keggidlist):
        return serv.get_best_homologs_by_genes(keggorg, keggidlist)
def get_best_best_homologs_by_genes(keggorg, keggidlist):
        return serv.get_best_best_homologs_by_genes(keggorg, keggidlist)
def get_score_between_genes(keggid1, keggid2):
        return serv.get_score_between_genes(keggid1, keggid2)
def get_definition_by_gene(keggid):
        return serv.get_definition_by_gene(keggid)
def get_common_motifs_by_genes(keggidlist):
        return serv.get_common_motifs_by_genes(keggidlist)
def get_genes_by_motifs(motiflist):
        return serv.get_genes_by_motifs(motiflist)

# PATHWAY Methods (20 methods)
def get_genes_by_pathway(pathwayid):
        return serv.get_genes_by_pathway(pathwayid)
def get_compounds_by_pathway(pathwayid):
        return serv.get_compounds_by_pathway(pathwayid)
def get_enzymes_by_pathway(pathwayid):
        return serv.get_enzymes_by_pathway(pathwayid)
def get_pathways_by_genes(keggidlist):
        return serv.get_pathways_by_genes(keggidlist)
def get_pathways_by_compounds(cpdlist):
        return serv.get_pathways_by_compounds(cpdlist)
def get_pathways_by_enzymes(enzymelist):
        return serv.get_pathways_by_enzymes(enzymelist)
def get_pathways_by_reactions(reactionid_list):
        return serv.get_pathways_by_reactions(reactionid_list)
```

```
def get_reactions_by_pathways(pathwayidlist):
        return serv.get_reactions_by_pathways(pathwayidlist)
def get_enzymes_by_reaction(reactionidlist):
        return serv.get_enzymes_by_reaction(reactionidlist)
def get_genes_by_enzymes(enzymeidlist, keggorg):
        return serv.get_genes_by_enzymes(enzymeidlist, keggorg)
def get_linked_pathways(pathwayid):
        return serv.get_linked_pathways(pathwayid)
def mark_all_pathways_by_genes(keggorg, genesidlist):
        return serv.mark_all_pathways_by_genes(keggorg, genesidlist)
def mark_all_pathways_by_enzymes(keggorg, enzymeidlist):
        return serv.mark_all_pathways_by_enzymes(keggorg, enzymeidlist)
def mark_all_pathways_by_compounds(keggorg, compoundidlist):
        return serv.mark_all_pathways_by_compounds(keggorg, compoundidlist)
def mark_pathways_by_genes(pathwayid, genesidlist):
        return serv.mark_pathways_by_genes(pathwayid, genesidlist)
def mark_pathway_by_enzymes(pathwayid, enzymeidlist):
        return serv.mark_pathway_by_enzymes(pathwayid, enzymeidlist)
def mark_pathway_by_compounds(pathwayid, compoundidlist):
        return serv.mark_pathway_by_compounds(pathwayid, compoundidlist)
def color_pathway_by_genes(pathwayid, genesidlist, colorlist):
        return serv.color_pathway_by_genes(pathwayid, genesidlist, colorlist)
def color_pathway_by_enzymes(pathwayid, enzymeidlist, colorlist):
        return serv.color_pathway_by_enzymes(pathwayid, enzymeidlist, colorlist)
def color_pathway_by_compounds(pathwayid, compoundidlist, colorlist):
        return serv.color_pathway_by_compounds(pathwayid, compoundidlist, colorlist)

# GENES Methods (4 methods)
def get_aaseqs(genesidlist): return serv.get_aaseqs(genesidlist)
def get_ntseqs(genesidlist): return serv.get_ntseqs(genesidlist)
def get_all_genes_by_organism(keggorg): return serv.get_all_genes_by_organism(keggorg)
def get_num_genes_by_organism(keggorg): return serv.get_num_genes_by_organism(keggorg)
```

How To Use KEGG API “Python-ically”?

This section is adapted from the original KEGG API reference manual, which can be found at http://www.genome.ad.jp/kegg/soap/doc/keggapi_manual.html, for use with the Python programming language.

The simplest way to use KEGG API on Python is to import pykegg into the codes. pykegg deals with all needed connections via SOAP to KEGG API.Once KEGGpy is imported, there is no other requirements needed to be fulfilled in order to use any KEGG API methods, except that the string “pykegg.” must prefix the methods.

Terminology

- 'keggorg' is a three letter organism code used in KEGG. The list can be found at: http://www.genome.ad.jp/kegg/kegg2.html#genes
- 'db_name' is a database name used in GenomeNet service. See the description of a list_databases method below.
- 'entry_id' is a unique identifier of which format is the combination of the database name and the identifier of an entry joined by a colon sign as 'database:entry' (e.g. 'embl:J00231' means an EMBL entry 'J00231'). 'entry_id' includes 'genes_id', 'enzyme_id', 'compound_id', 'reaction_id', 'pathway_id' and 'motif_id' described in below.
- 'genes_id' is a gene identifier used in KEGG/GENES which consists of 'keggorg' and a

gene name (e.g. 'eco:b0001' means an E. coli gene 'b0001').

- 'enzyme_id' is an enzyme identifier consisting of database name 'ec' and an enzyme code used in KEGG/LIGAND (e.g. 'ec:1.1.1.1' means an alcohol dehydrogenase enzyme)
- 'compound_id' is a compound identifier consisting of database name 'cpd' and a compound number used in KEGG/LIGAND (e.g. 'cpd:C00158' means a citric acid).
- 'reaction_id' is a reaction identifier consisting of database name 'rn' and a reaction number used in KEGG/REACTION (e.g. 'rn:R00959' is a reaction which metabolize cpd:C00103 into cpd:C00668)
- 'pathway_id' is a pathway identifier consisting of 'path' and a pathway number used in KEGG/PATHWAY. Pathway numbers prefixed by 'map' specify the reference pathway and pathways prefixed by the 'keggorg' specify pathways specific to the organism (e.g. 'path:map00020' means a reference pathway for the cytrate cycle and 'path:eco00020' means a same pathway of which E. coli genes are marked).
- 'motif_id' is a motif identifier consisting of motif database names ('ps' for prosite, 'bl' for blocks, 'pr' for prints, 'pd' for prodom, and 'pf' for pfam) and a motif entry name. (e.g. 'pf:DnaJ' means a Pfam database entry 'DnaJ').
- 'threshold' is a threshold value for the Smith-Waterman score (not fewer than 100).

Return values

Some of the methods described below will return a set of values in a complex data structure.

SSDBResult

SSDBResult data type contains the following fields:

kid1	genes_id of the query
kid2	genes_id of the target
sw_score	Smith-Waterman score between kid1 and kid2
ident	% identity between kid1 and kid2
overlap	overlap length between kid1 and kid2
s1_start	start position of the alignment in kid1
s1_end	end positoin of the alignment in kid1
s2_start	start position of the alignment in kid2
s2_end	end positoin of the alignment in kid2
b1	best-best flag from kid1 to kid2 (1 means best, otherwise 0)
b2	best-best flag from kid2 to kid1 (1 means best, otherwise 0)
definition1	definition string of the kid1
definition2	definition string of the kid2
length1	amino acid length of the kid1
length2	amino acid length of the kid2

SSDBResultArray

SSDBResultArray data type is a list of the SSDBResult data type.

MOTIFResult

MOTIFResult data type contains the following fields:

mid	motif_id of the motif definition definition of the motif
kid	genes_id of the gene containing the motif
seq_f	start position of the motif match

seq_t	end position of the motif match
score	score of the motif match (PROSITE Profile)
Evalue	E value of the motif match (Pfam, TIGRfam)

KEGG API server will return all of these fields but user and/or the client library can select a part of the fields as needed.

MOTIFResultArray
MOTIFResultArray data type is a list of the MOTIFResult data type.

GENESResult
GENESResult data type contains the following fields:

kid	genes_id of the gene
keggdef	definition of the gene

KEGG API server will return all of these fields but user and/or the client library can select a part of the fields as needed.

GENESResultArray
GENESResultArray data type is a list of the GENESResult data type.

LinkDBResult
LinkDBResult data type contains the following fields:

entry	entry_id of the entry
link_type	type of the link (direct, indirect)
links	link across the databases

LinkDBResultArray
LinkDBResultArray data type is a list of the LinkDBResult data type.

Python-Specific Information and Return Types
Python is a weakly typed language. By that, it means that Python does not enforce tight rules regarding data types. To Python, all data elements are treated as bytes. Hence, the data structures described above acts as interpretation rules, rather than programmatic enforcement rules.

All return data will exists as a comma-delimited list, in the order described. For example, LinkDBResult will be displayed as a list with 3 string elements. If a LinkDBResultArray has 2 records, it will be displayed as a list of 6 (repeating LinkDBResult “list”) string elements.

Methods
KEGG information
This section describes the APIs for retrieving the general information concerning latest version of the KEGG database.

list_databases()
List up the databases available on the GenomeNet.

Return value: string

list_organisms()
List up the organisms (in keggorg) in the KEGG/GENES database.
Return value: ArrayOfstring

list_pathways()
List up the pathway maps in the KEGG/PATHWAY database.
Return value: string

dbinfo(db_name)
Show the version information of the specified database.
Return value: string
Example: # Show the information of the latest GENES database.
print dbinfo('genes')

DBGET/LinkDB
get_entries(entry_id_list)
Retrieve all entries of the given entry_ids.
Return value: string
Example: get_entries(['eco:b0002','cpd:C00209'])

get_linkdb_by_entry(entry_id, target_db)
Retrieve the database entries linked from the user specified database entry. It can also be specified the targeted database.
Return value: LinkDBResultArray
Example: # Get the entries of KEGG/PATHWAY database linked from the entry 'eco:b0002'.
get_linkdb_by_entry('eco:b0002', 'pathway')

SSDB
This section describes the KEGG API for SSDB database. For more details on SSDB, see: http://www.genome.ad.jp/kegg/ssdb/

get_all_neighbors_by_gene(genes_id, threshold = 100, keggorg_list = nil)
Search homologous genes of the user specified 'genes_id' from all organisms having a 'sw_score' over the threshold. You can narrow the target organisms to search by passing a list of 'keggorg' as the third argument (how to pass the list will depend on the language specific implementations).
Return value: SSDBResultArray
Examples: # This will search all the homologous genes for E. coli gene 'b0002', # over the default threshold score 100.
get_all_neighbors_by_gene('eco:b0002')
# This will search homologous genes only in E. coli O157 strain and # H. influenzae with 'sw_score' over 500.
get_all_neighbors_by_gene('eco:b0002', 500, ['ece', 'hin'])

get_best_best_neighbors_by_gene(genes_id, threshold = 100, keggorg_list = nil)
Search best-best neighbor of the gene in each organism.

Return value: SSDBResultArray
Example: get_best_best_neighbors_by_gene('eco:b0002', 500)

get_best_neighbors_by_gene(genes_id, threshold = 100, keggorg_list = nil)
Search best neighbors in each organism.
Return value: SSDBResultArray
Example: # List up best neighbors of 'eco:b0002' having 'sw_score' over 500.
get_best_neighbors_by_gene('eco:b0002', 500)

get_reverse_best_neighbors_by_gene(genes_id, threshold = 100, keggorg_list = nil)
Search reverse best neighbors in each organism.
Return value: SSDBResultArray
Example:# List up reverse best neighbors of 'eco:b0002' having 'sw_score' over 500.
get_reverse_best_neighbors_by_gene('eco:b0002', 500)

get_paralogs_by_gene(genes_id, threshold = 100)
Search paralogous genes in the same organism.
Return value: SSDBResultArray
Example: # List up paralogous genes of 'eco:b0002' having 'sw_score' over 500.
get_paralogs_by_gene('eco:b0002', 500)

get_best_homologs_by_genes(keggorg, genes_id_list)
Search best neighbors in the target organism 'keggorg' from the list of genes user specified (how to pass the list of genes is language specific).
Return value: SSDBResultArray
Example: # Search the corresponding genes in H. influenzae list = ['eco:b0002', 'eco:b0003', 'eco:b0004', 'eco:b0005', 'eco:b0006']
get_best_homologs_by_genes('hin', list)

get_best_best_homologs_by_genes(keggorg, genes_id_list)
Similar to 'get_best_homologs_by_genes', but returns only genes having the best-best relationships.
Return value: SSDBResultArray
Example: list = ['eco:b0002', 'eco:b0003', 'eco:b0004', 'eco:b0005', 'eco:b0006']
get_best_best_homologs_by_genes('hin', list)

get_score_between_genes(genes_id1, genes_id2)
Returns a Smith-Waterman score between the two genes.
Return value: SSDBResult
Example: # Returns a 'sw_score' between two E. coli genes 'b0002' and 'b3940'
get_score_between_genes('eco:b0002', 'eco:b3940')

get_definition_by_gene(genes_id)
Returns a definition of the gene (annotated by KEGG) as a string.
Return value: string
Example: # Retrieve a definition of the E. coli gene 'b0002'
get_definition_by_gene('eco:b0002')

get_common_motifs_by_genes(target_db_list, genes_id_list, threshold)

Search common motifs among the specified gene list. As for 'target_db_list', It can be specified Pfam, Tigrfam, Prosite Pattern, Prosite Profile as 'pfam', 'tfam', 'prositepattern', 'prositeprofile', respectively. In the case of 'pfam' or 'tfam', it can be specified the threshold value as third argument.
Return value: MOTIFResultArray
Example: # Returns the common motifs among the two E. coli genes 'b0002' and 'b3940' target_db_list = ['pfam', 'tfam'] genes_id_list = ['eco:b0002', 'eco:b4024'] threshold = 1.0e-10
get_common_motifs_by_genes(target_db_list, genes_id_list, threshold)

get_genes_by_motifs(motif_id_list)

Search all the genes which contains all of the specified motifs.
Return value: GENESResultArray
Example: # Returns all the genes which have Pfam 'DnaJ' and Prosite 'DNAJ_2' motifs. list = ['pf:DnaJ', 'ps:DNAJ_2']
get_genes_by_motifs(list)

PATHWAY

This section describes the KEGG API for PATHWAY database. For more details on PATHWAY database, see: http://www.genome.ad.jp/kegg/kegg2.html#pathway

get_genes_by_pathway(pathway_id)

Search all genes on the specified pathway. Organism name is given by the name of a pathway map.
Return value: ArrayOfstring
Example: # Returns all the E. coli genes on the pathway map '00020'.
get_genes_by_pathway('path:eco00020')

get_enzymes_by_pathway(pathway_id)

Search all enzymes on the specified pathway.
Return value: ArrayOfstring
Example: # Returns all the enzymes on the pathway map '00020'.
get_enzymes_by_pathway('path:eco00020')

get_compounds_by_pathway(pathway_id)

Search all compounds on the specified pathway.
Return value: ArrayOfstring
Example: # Returns all the compounds on the pathway map '00020'.
get_compounds_by_pathway('path:eco00020')

get_pathways_by_genes(genes_id_list)

Search all pathways which include all the given genes. How to pass the list of genes_id will depend on the language specific implementations.
Return value: ArrayOfstring
Example: # Returns all pathways include E. coli genes 'b0077' and 'b0078'
get_pathways_by_genes(['eco:b0077' , 'eco:b0078'])

get_pathways_by_enzymes(enzyme_id_list)

Search all pathways which include all the given enzymes.
Return value: ArrayOfstring
Example: # Returns all pathways include enzyme '1.3.99.1'
get_pathways_by_enzymes(['ec:1.3.99.1'])

get_pathways_by_compounds(compound_id_list)
Search all pathways which include all the given compounds.
Return value: ArrayOfstring
Example: # Returns all pathways include compounds 'C00033' and 'C00158'
get_pathways_by_compounds(['cpd:C00033', 'cpd:C00158'])

get_pathways_by_reactions(reaction_id_list)
Retrieve all pathway_ids which include all the given reaction_ids.
Return value: ArrayOfstring
Example: # Returns ids of PATHWAY entries which include REACTION entries,
rn:R00959, # rn:R02740, rn:R00960 and rn:R01786
get_pathways_by_reactions(['rn:R00959', 'rn:R02740', 'rn:R00960',
'rn:R01786'])

get_reactions_by_pathways(pathway_id_list)
Retrieve all reaction_ids which include all the given pathway_ids.
Return value: ArrayOfstring
Example: # Returns ids of REACTION entries which include PATHWAY entries,
path:map00260.
get_reactions_by_pathways(['path:map00260'])

get_enzymes_by_reactions(reaction_id_list)
Retrieve all enzyme_ids which include all the given reaction_ids.
Return value: ArrayOfstring
Example: # Returns ids of ENZYME entries which include REACTION entries,
rn:R00100.
get_enzymes_by_reactions(['rn:R00100'])

get_genes_by_enzymes(enzyme_id_list, keggorg)
Retrieve all genes_ids of the keggorg which include all the given enzymen_ids.
Return value: ArrayOfstring
Example: # Returns ids of GENES entries of E.coli genes which are assigned EC
number, # ec:1.1.1.1 and ec:1.2.1.1
get_genes_by_enzymes(['ec:1.1.1.1','ec:1.2.1.1'],'eco')

get_linked_pathways(pathway_id)
Retrieve all pathway_ids which is linked from a given pathway_id.
Return value: ArrayOfstring
Example: # Returns IDs of PATHWAY entries linked from 'path:eco00620'.
get_linked_pathways('path:eco00620')

mark_all_pathways_by_genes(keggorg, genes_id_list)
Search the given organism's pathway maps including all the given genes, mark objects corresponding the genes on the map and return URLs of the maps.

Return value: ArrayOfstring
Example: # Returns URLs for pathway maps of E.coli which include eco:b0002 gene. The object corresponding eco:b0002 gene on the maps are colored by red.
mark_all_pathways_by_genes('eco',['eco:b0002'])

mark_all_pathways_by_enzymes(keggorg, enzyme_id_list)
Search the given organism's pathway maps including all the given EC number, mark objects having the EC number on the maps and return URLs of the maps.
Return value: ArrayOfstring
Example: # Returns URLs for pathway maps of H.influenzae includeing enzymes of which EC number are 1.2.3.1 and 4.3.1.5.
mark_all_pathways_by_enzymes('hin',['1.2.3.1','4.3.1.5'])

mark_all_pathways_by_compounds(keggorg, compound_id_list)
Search the given organism's pathway maps including all the given compounds, mark objects corresponding compounds on the maps and return URLs of the maps.
Return value: ArrayOfstring
Example: # Returns URLs for pathway maps of B.subtilis including compounds of which entry ids are C00011 and C00209
mark_all_pathways_by_compounds('bsu',['C00011','C00209'])

mark_pathway_by_genes(pathway_id, genes_id_list)
Mark objects corresponding the given genes on the given pathway map with red color, and return the URL of the map.
Return value: ArrayOfstring
Example: # Mark objects correspoding the eco:b0600 and eco:b1190 on path:eco00252 with red color, and return the URL of the map.
mark_pathway_by_genes('path:eco00252', ['eco:b0600','eco:b1190'])

mark_pathway_by_enzymes(pathway_id, enzyme_id_list)
Mark objects corresponding the given enzymes on the given pathway map with red color, and return the URL of the map.
Return value: ArrayOfstring
Example: #Mark objects corresponding EC4.1.1.5 on path:eco00660 with red color, and return the URL of the map.
mark_pathway_by_enzymes('path:eco00660', ['4.1.1.5'])

mark_pathway_by_compounds(pathway_id, compound_id_list)
Mark objects corresponding the given compounds on the given pathway map with red color, and return the URL of the map.
Return value: ArrayOfstring
Example: # Mark objects corresponding cpd:C000022, cpd:C05993, cpd:C00083 on path:ecp00620 with red color, and return the URL of the map.
mark_pathway_by_compounds('path:eco00620', ['C00022','C05993','C00083'])

color_pathway_by_genes(pathway_id, genes_id_list, color_list)
Mark objects corresponding the given genes on the given pathway map with the

given colors, and return the URL of the map.
Return value: ArrayOfstring
Example: # Mark objects corresponding the genes, eco:b0207 and eco:b1300 on the path:eco00053 with green and blue, respectively, and return the URL of the map.
color_pathway_by_genes('path:eco00053', ['eco:b0207','eco:b1300'], ['green','blue'])

color_pathway_by_enzymes(pathway_id, enzyme_id_list, color_list)
Mark objects corresponding the given EC numbers on the given pathway map with the given colors, and return the URL of the map.
Return value: ArrayOfstring
Example: # Mark objects corresponding EC number 3.1.2.1 and 6.4.1.1 on the path:eco00620 with yellow and green, respectively, and return the URL of the map.
color_pathway_by_enzymes('path:eco00620', ['3.1.2.1','6.4.1.1'], ['yellow','green'])

color_pathway_by_compounds(pathway_id, compound_id_list, color_list)
Mark objects corresponding the given compounds on the given pathway map with the given colors, and return the URL of the map.
Return value: ArrayOfstring
Example: # Mark objects corresponding cpd:C06010, cpd:C01011 and cpd:C00651 on the path:eco00660 with pink, green and blue, respectively, and return the URL of the map.
color_pathway_by_compounds('path:eco00660', ['C06010','C01011','C00651'], ['pink','green','blue'])

GENES
This section describes the KEGG APIs for GENES database. For more details on GENES database, see: http://www.genome.ad.jp/kegg/kegg2.html#genes

get_aaseqs(genes_id_list)
Retrieve the amino acid sequences of all the given GENES entries.
Return value: string
Example: get_aaseqs(['eco:b0002','bsu:BG10065'])

get_ntseqs(genes_id_list)
Retrieve the nucleotide sequences of all the given GENES entries.
Return value: string
Example: get_ntseqs(['eco:b0002','bsu:BG10065'])

get_all_genes_by_organism(keggorg)
Retrieve all entries on the specified organism's genome.
Return value: string
Example: # Retrive all GENES entries of the H.influenzae
get_all_genes_by_organism('hin')

get_num_genes_by_organism(keggorg)

Get the number of the genes on the specified organism's genome.
Return value: string (notice: will be changed to int shortly)
Example: # Get the number of the genes on the E.coli genome.
get_num_genes_by_organism('eco')

Appendix D:
Source Code Listings of Analysis Scripts

Introduction

Chapter 3: Data Analysis discusses the steps taken to analyze all the microarray data in this project and the corresponding logical explanation in these steps. This appendix appends to that by providing the source codes of the scripts used, which can serve as the ultimate form of documentation. All scripts are written in Python programming language, for reasons discussed in Appendix C, and interpreted by Python version 2.3.

A brief description of the purpose of a listing (a script) will be prefixed and the corresponding results in suffix. In most of the cases, the entire set of results from a script is impossible to be listed due to sheer volume. In these cases, the reader will be directed to a file. All listings shown here will be using data from day 12 pregnant and day 2 lactation as example.

Normalized data of each microarray scan can be found in “Microarray Data” folder in the accompanying compact disc, whereas result files are found in “Processed Results” folder. External data, such as, gene ontology, can be found in “External Data” folder.

Listing 1

To obtain the numerical value for cut-off for different intensities, where it is set at 5% of the maximum intensity for the specific channel.

```
# (c) 2004 Maurice Ling
# BSc(Hons), The University of Melbourne, Dept of Zoology
# created on 2nd of March, 2004

from basereader import *
ClearRegister()
BaseReader('/users/mauriceling/mauricehons/d12p-d2l.txt')
def max_intensities(Register, column):
          max = []
          for c in range(column):
                    max.append(0)
          for s in Register:
                    for col in range(column):
                              if (s[col] > max[col]):
                                        max[col] = s[col]
          return max
def part_max_intensities(Register, column, percent):
          for c in range(column):
                    Register[c] = Register[c] * percent;
          return Register
SpotRegisterIntensity = SpotIntensityToFloat(SpotRegisterIntensity)
print part_max_intensities(max_intensities(
          SpotRegisterIntensity, 2),  2, 0.05)
```

Results

D12P-D2L: 499, 3276
D12P-IFP: 2918, 986
IFP-D2L: 3123, 2733

IFP-FP: 1612, 2947
IF-IFP: 1945, 9806
IFP-IP: 2355, 13107

Listing 2

Using the cut-off values obtained from Listing 1, extract all data with intensities more than the cut-off values in any channel.

```
# (c) 2004 Maurice Ling
# BSc(Hons), The University of Melbourne, Dept of Zoology
# created on 2nd of March, 2004

from basereader import *
ClearRegister()
BaseReader('/users/mauriceling/mauricehons/d12p-d2l.txt')
SpotRegisterIntensity =
          SpotIntensityToFloat(SpotRegisterIntensity)
m = [499, 3276]
register = []
del register[0:]
c = 0
for s in range(len(SpotRegisterIntensity)):
          if (SpotRegisterIntensity[s][0] > m[0] or
                              SpotRegisterIntensity[s][1] > m[1]):
                    register.append(SpotRegister[s])
                    c = c + 1
print "LIST OF MORE THAN 5 PERCENTILE OF MAX INTENSITY (ANY CHANNEL) IN
DAY 12 PREGNANT VERSUS DAY 2 LACTATION"
print
for s in range(c):
          print register[s]
print
print
print "Total rows: " + str(c)
```

Results
(See List-2-result-d12p-d2l.rtf)
(See List-2-result-d12p-ifp.rtf)
(See List-2-result-ifp-d2l.rtf)
(See List-2-result-ifp-fp.rtf)
(See List-2-result-if-ifp.rtf)
(See List-2-result-ifp-ip.rtf)

Listing 3

Progressing from Listing 2, filters data into single expression (expressed only in one sample), and common expression requiring both channels above cut-off values, meaning, only common high expressing genes will be shown. For common expressions, those more than 5 times difference in intensities will be filtered for.

```
# (c) 2004 Maurice Ling
# BSc(Hons), The University of Melbourne, Dept of Zoology
# created on 3rd of March, 2004

from basereader import *
ClearRegister()
```

```
BaseReader('/users/mauriceling/mauricehons/d12p-d2l.txt')
SpotRegisterIntensity = SpotIntensityToFloat(SpotRegisterIntensity)
m = [499, 3276]
print "LIST OF MORE THAN 5 PERCENTILE OF MAX INTENSITY (ANY CHANNEL) IN
"
print "DAY 12 PREGNANT VERSUS DAY 2 LACTATION"
register = []
one = []
two = []
lap = []
fold = []
del register[0:]
del one[0:]
del two[0:]
del lap[0:]
del fold[0:]
one_count = 0
two_count = 0
lap_count = 0
fold_count = 0
for s in range(len(SpotRegisterIntensity)):
          if (SpotRegisterIntensity[s][0] > m[0] or
                              SpotRegisterIntensity[s][1] > m[1]):
                    register.append(SpotRegister[s])
for s in range(len(register)):
          #check for int1 < m[0], then int2 should be > m[1]
          if (float(register[s][2]) < m[0]):
                    two.append(register[s])
                    two_count = two_count + 1
          #if int2 <  m[1], the int1 should be > m[0]
          if (float(register[s][3]) < m[1]):
                    one.append(register[s])
                    one_count = one_count + 1
          #both more than cutoff
          if (float(register[s][2]) > m[0] and
                              float(register[s][3]) > m[1]):
                                        lap.append(register[s])
                                        lap_count = lap_count + 1
for s in range(len(lap)):
          if (float(lap[s][2])/float(lap[s][3]) > 5):
                    fold.append(lap[s])
                    fold_count = fold_count + 1
          if (float(lap[s][2])/float(lap[s][3]) < 0.2):
                    fold.append(lap[s])
                    fold_count = fold_count + 1
print
print "INTENSITY ONE ONLY"
for s in range(len(one)):
          print one[s]
print
print
print "INTENSITY TWO ONLY"
for s in range(len(two)):
          print two[s]
print
print
print "OVERLAPPING SPOTS"
for s in range(len(lap)):
          print lap[s]
print
print
print "OVERLAPPING SPOTS WITH >5X DIFFERENCE"
for s in range(len(fold)):
          print fold[s]
print
print
print "Intensity one only count: " + str(one_count)
```

```
print "Intensity two only count: " + str(two_count)
print "Overlap count: " + str(lap_count)
print ">5X difference count: " + str(fold_count)
```

Result
(See List-3-result-d12p-d2l-percent.rtf)
(See List-3-result-d12p-ifp-percent.rtf)
(See List-3-result-ifp-d2l-percent.rtf)
(See List-3-result-ifp-fp-percent.rtf)
(See List-3-result-if-ifp-percent.rtf)
(See List-3-result-ifp-ip-percent.rtf)
(See List-3-result-d12p-d2l-3sd.rtf)
(See List-3-result-d12p-ifp-3sd.rtf)
(See List-3-result-ifp-d2l-3sd.rtf)
(See List-3-result-ifp-fp-3sd.rtf)
(See List-3-result-ifp-ip-3sd.rtf)
(See List-3-result-if-ifp-3sd.rtf)

Listing 4

Using 3X and 2X standard deviations of background as cut-off values instead of 5% maximum intensities, data are filtered into single expression and common expression, which requires only one channel to be above cut-off and the other channel to be above zero, in contrast to Listing 3, which requires both channels to be above cut-off values. For common expressions, those more than 5 times difference in intensities will be filtered for.

```
# (c) 2004 Maurice Ling
# BSc(Hons), The University of Melbourne, Dept of Zoology
# created on 3rd of March, 2004

from basereader import *
ClearRegister()
BaseReader('/users/mauriceling/mauricehons/d12p-d2l.txt')
SpotRegisterIntensity = SpotIntensityToFloat(SpotRegisterIntensity)
m = [103*3, 328*3]
print "LIST OF MORE THAN 3 STANDARD DEVIATIONS OF BACKGROUND (ANY
CHANNEL) IN "
print "DAY 12 PREGNANT VERSUS DAY 2 LACTATION"
register = []
one = []
two = []
lap = []
fold = []
del register[0:]
del one[0:]
del two[0:]
del lap[0:]
del fold[0:]
one_count = 0
two_count = 0
lap_count = 0
fold_count = 0
for s in range(len(SpotRegisterIntensity)):
          if (SpotRegisterIntensity[s][0] > m[0] or
                              SpotRegisterIntensity[s][1] > m[1]):
                    register.append(SpotRegister[s])
for s in range(len(register)):
```

```
          #check for int1 < m[0], then int2 should be > m[1]
          if (float(register[s][2]) < m[0]):
                    two.append(register[s])
                    two_count = two_count + 1
          #if int2 <  m[1], the int1 should be > m[0]
          if (float(register[s][3]) < m[1]):
                    one.append(register[s])
                    one_count = one_count + 1
          #one channel more than cutoff, the other more than zero
          if ((float(register[s][2]) > m[0] and
                              float(register[s][3]) > 0) or
                              (float(register[s][3]) > m[1] and
                              float(register[s][2]) > 0)):
                                        lap.append(register[s])
                                        lap_count = lap_count + 1
for s in range(len(lap)):
          if (float(lap[s][2])/float(lap[s][3]) > 5):
                    fold.append(lap[s])
                    fold_count = fold_count + 1
          if (float(lap[s][2])/float(lap[s][3]) < 0.2):
                    fold.append(lap[s])
                    fold_count = fold_count + 1
print
print "INTENSITY ONE ONLY"
for s in range(len(one)):
          print one[s]
print
print
print "INTENSITY TWO ONLY"
for s in range(len(two)):
          print two[s]
print
print
print "OVERLAPPING SPOTS"
for s in range(len(lap)):
          print lap[s]
print
print
print "OVERLAPPING SPOTS WITH >5X DIFFERENCE"
for s in range(len(fold)):
          print fold[s]
print
print
print "Intensity one only count: " + str(one_count)
print "Intensity two only count: " + str(two_count)
print "Overlap count: " + str(lap_count)
print ">5X difference count: " + str(fold_count)
```

Result

(See List-4-result-d12p-d2l-3sd.rtf)
(See List-4-result-d12p-d2l-2sd.rtf)
(See List-4-result-d12p-ifp-3sd.rtf)
(See List-4-result-d12p-ifp-2sd.rtf)
(See List-4-result-ifp-d2l-3sd.rtf)
(See List-4-result-ifp-d2l-2sd.rtf)
(See List-4-result-ifp-fp-3sd.rtf)
(See List-4-result-ifp-fp-2sd.rtf)
(See List-4-result-if-ifp-3sd.rtf)
(See List-4-result-if-ifp-2sd.rtf)
(See List-4-result-ifp-ip-3sd.rtf)
(See List-4-result-ifp-ip-2sd.rtf)

Listing 5

Using 2X standard deviations of background as cut-off values, common expressions are filtered for, using identical criteria as of Listing 4. These are then further filtered for those more than 5X, 4X, 3X and 2X difference in gene expression.

```
# (c) 2004 Maurice Ling
# BSc(Hons), The University of Melbourne, Dept of Zoology
# created on 4th of March, 2004

from basereader import *
ClearRegister()
BaseReader('/users/mauriceling/mauricehons/d12p-d2l.txt')
SpotRegisterIntensity = SpotIntensityToFloat(SpotRegisterIntensity)
m = [103*2, 328*2]
print "LIST OF FOLD CHANGES OF MORE THAN 2 STANDARD DEVIATIONS OF
BACKGROUND (ANY CHANNEL) IN "
print "DAY 12 PREGNANT VERSUS DAY 2 LACTATION"
register = []
lap = []
fold5 = []
fold4 = []
fold3 = []
fold2 = []
del register[0:]
del lap[0:]
del fold5[0:]
del fold4[0:]
del fold3[0:]
del fold2[0:]
lap_count = 0
fold5_count = 0
fold4_count = 0
fold3_count = 0
fold2_count = 0
for s in range(len(SpotRegisterIntensity)):
          if (SpotRegisterIntensity[s][0] > m[0] or
                              SpotRegisterIntensity[s][1] > m[1]):
                    register.append(SpotRegister[s])
for s in range(len(register)):
          #one channel more than cutoff, the other more than zero
          if ((float(register[s][2]) > m[0] and
                              float(register[s][3]) > 0) or
                              (float(register[s][3]) > m[1] and
                              float(register[s][2]) > 0)):
                                        lap.append(register[s])
                                        lap_count = lap_count + 1
for s in range(len(lap)):
          if (float(lap[s][2])/float(lap[s][3]) > 5):
                    fold5.append(lap[s])
                    fold5_count = fold5_count + 1
          if (float(lap[s][2])/float(lap[s][3]) > 4):
                    fold4.append(lap[s])
                    fold4_count = fold4_count + 1
          if (float(lap[s][2])/float(lap[s][3]) > 3):
                    fold3.append(lap[s])
                    fold3_count = fold3_count + 1
          if (float(lap[s][2])/float(lap[s][3]) > 2):
                    fold2.append(lap[s])
                    fold2_count = fold2_count + 1

          if (float(lap[s][2])/float(lap[s][3]) < 0.2):
                    fold5.append(lap[s])
                    fold5_count = fold5_count + 1
          if (float(lap[s][2])/float(lap[s][3]) < 0.25):
```

```
                    fold4.append(lap[s])
                    fold4_count = fold4_count + 1
          if (float(lap[s][2])/float(lap[s][3]) < 0.33):
                    fold3.append(lap[s])
                    fold3_count = fold3_count + 1
          if (float(lap[s][2])/float(lap[s][3]) < 0.5):
                    fold2.append(lap[s])
                    fold2_count = fold2_count + 1
print
print "OVERLAPPING SPOTS WITH >5X DIFFERENCE"
for s in range(len(fold5)):
          print fold5[s]
print
print
print "OVERLAPPING SPOTS WITH >4X DIFFERENCE"
for s in range(len(fold4)):
          print fold4[s]
print
print
print "OVERLAPPING SPOTS WITH >3X DIFFERENCE"
for s in range(len(fold3)):
          print fold3[s]
print
print
print "OVERLAPPING SPOTS WITH >2X DIFFERENCE"
for s in range(len(fold2)):
          print fold2[s]
print
print
print "Overlap count: " + str(lap_count)
print ">5X difference count: " + str(fold5_count)
print ">4X difference count: " + str(fold4_count)
print ">3X difference count: " + str(fold3_count)
print ">2X difference count: " + str(fold2_count)
```

Result
(See List-5-result-d12p-d2l.rtf)
(See List-5-result-d12p-ifp.rtf)
(See List-5-result-ifp-d2l.rtf)
(See List-5-result-ifp-fp.rtf)
(See List-5-result-if-ifp.rtf)
(See List-5-result-ifp-ip.rtf)

Listing 6

Progressing from Listing 6 and using only those more than 2X difference in gene expression, the data are split into left and right (left versus right), then sort according to fold differences.

```
# (c) 2004 Maurice Ling
# BSc(Hons), The University of Melbourne, Dept of Zoology
# created on 8th of March, 2004

from basereader import *

ClearRegister()
BaseReader('/users/mauriceling/mauricehons/d12p-d2l.txt')
SpotRegisterIntensity = SpotIntensityToFloat(SpotRegisterIntensity)
m = [103*2, 328*2]
print "LIST OF MORE THAN 2 FOLD CHANGES OF MORE THAN 2 STANDARD
DEVIATIONS OF BACKGROUND (ANY CHANNEL) IN "
print "DAY 12 PREGNANT VERSUS DAY 2 LACTATION"
```

```
register = []
lap = []
fold2 = []
right = []
left = []
del register[0:]
del lap[0:]
del fold2[0:]
del right[0:]
del left[0:]
lap_count = 0
fold2_count = 0
right_count = 0
left_count = 0
for s in range(len(SpotRegisterIntensity)):
          if (SpotRegisterIntensity[s][0] > m[0] or
                              SpotRegisterIntensity[s][1] > m[1]):
                    register.append(SpotRegister[s])
for s in range(len(register)):
          #one channel more than cutoff, the other more than zero
          if ((float(register[s][2]) > m[0] and
                              float(register[s][3]) > 0) or
                              (float(register[s][3]) > m[1] and
                              float(register[s][2]) > 0)):
                                        lap.append(register[s])
                                        lap_count = lap_count + 1
for s in range(len(lap)):
          if (float(lap[s][2])/float(lap[s][3]) > 2):
                    fold2.append(lap[s])
                    fold2_count = fold2_count + 1
          if (float(lap[s][2])/float(lap[s][3]) < 0.5):
                    fold2.append(lap[s])
                    fold2_count = fold2_count + 1
for s in range(len(fold2)):
          if (float(fold2[s][2]) > float(fold2[s][3])):
                    left.append(fold2[s])
                    left_count = left_count + 1
          else:
                    right.append(fold2[s])
                    right_count = right_count + 1
for s in range(len(left)):
          left[s].insert(0, float(left[s][2])/float(left[s][3]))
for s in range(len(right)):
          right[s].insert(0, float(right[s][3])/float(right[s][2]))
right.sort()
left.sort()
right.reverse()
left.reverse()
print
print
print "OVERLAPPING SPOTS (LEFT > RIGHT)"
for s in range(len(left)):
          print left[s]
print
print
print "OVERLAPPING SPOTS (RIGHT > LEFT)"
for s in range(len(right)):
          print right[s]
print
print
print "Overlap count: " + str(lap_count)
print ">2X difference count: " + str(fold2_count)
print "Left > Right count: " + str(left_count)
print "Right > Left count " + str(right_count)
```

Result

(See List-6-result-d12p-d2l.rtf)
(See List-6-result-d12p-ifp.rtf)
(See List-6-result-ifp-d2l.rtf)
(See List-6-result-ifp-fp.rtf)
(See List-6-result-if-ifp.rtf)
(See List-6-result-ifp-ip.rtf)

Listing 7

Identical to Listing 6, but with gene ontology appended on.

```
# (c) 2004 Maurice Ling
# BSc(Hons), The University of Melbourne, Dept of Zoology
# created on 9th of March, 2004

from basereader import *
import string
ontology = []
del ontology[0:]
onto = open('/users/mauriceling/mauricehons/nia15k gene ontology.txt')
line = onto.readline()
while (line):
          line = string.split(line, '\t')
          ontology.append(line)
          line = onto.readline()
ClearRegister()
BaseReader('/users/mauriceling/mauricehons/d12p-d2l.txt')
SpotRegisterIntensity = SpotIntensityToFloat(SpotRegisterIntensity)
for s in range(len(SpotRegister)):
          for r in range(len(ontology)):
                    if (SpotRegister[s][0] == ontology[r][0]):
                              SpotRegister[s].append(ontology[r][1:])
m = [103*2, 328*2]
print "LIST OF MORE THAN 2 FOLD CHANGES OF MORE THAN 2 STANDARD
DEVIATIONS OF BACKGROUND (ANY CHANNEL) IN "
print "DAY 12 PREGNANT VERSUS DAY 2 LACTATION"
register = []
lap = []
fold2 = []
right = []
left = []
del register[0:]
del lap[0:]
del fold2[0:]
del right[0:]
del left[0:]
lap_count = 0
fold2_count = 0
right_count = 0
left_count = 0
for s in range(len(SpotRegisterIntensity)):
          if (SpotRegisterIntensity[s][0] > m[0] or
                              SpotRegisterIntensity[s][1] > m[1]):
                    register.append(SpotRegister[s])
for s in range(len(register)):
          #one channel more than cutoff, the other more than zero
          if ((float(register[s][2]) > m[0] and
                              float(register[s][3]) > 0) or
                              (float(register[s][3]) > m[1] and
                              float(register[s][2]) > 0)):
                                        lap.append(register[s])
                                        lap_count = lap_count + 1
for s in range(len(lap)):
```

```
          if (float(lap[s][2])/float(lap[s][3]) > 2):
                    fold2.append(lap[s])
                    fold2_count = fold2_count + 1
          if (float(lap[s][2])/float(lap[s][3]) < 0.5):
                    fold2.append(lap[s])
                    fold2_count = fold2_count + 1
for s in range(len(fold2)):
          if (float(fold2[s][2]) > float(fold2[s][3])):
                    left.append(fold2[s])
                    left_count = left_count + 1
          else:
                    right.append(fold2[s])
                    right_count = right_count + 1
for s in range(len(left)):
          left[s].insert(0, float(left[s][2])/float(left[s][3]))
for s in range(len(right)):
          right[s].insert(0, float(right[s][3])/float(right[s][2]))
right.sort()
left.sort()
right.reverse()
left.reverse()
print
print
print "OVERLAPPING SPOTS (LEFT > RIGHT)"
for s in range(len(left)):
          print left[s]
print
print
print "OVERLAPPING SPOTS (RIGHT > LEFT)"
for s in range(len(right)):
          print right[s]
print
print
print "Overlap count: " + str(lap_count)
print ">2X difference count: " + str(fold2_count)
print "Left > Right count: " + str(left_count)
print "Right > Left count " + str(right_count)
```

Result
(See List-7-result-d12p-d2l.rtf)
(See List-7-result-d12p-ifp.rtf)
(See List-7-result-ifp-d2l.rtf)
(See List-7-result-ifp-fp.rtf)
(See List-7-result-if-ifp.rtf)
(See List-7-result-ifp-ip.rtf)

Listing 8

Same as Listing 7 but with less than 2 fold changes in overlapping regions.

```
# (c) 2004 Maurice Ling
# BSc(Hons), The University of Melbourne, Dept of Zoology
# created on 10th of March, 2004

from basereader import *
import string

ontology = []
del ontology[0:]
onto = open('/users/mauriceling/mauricehons/nia15k gene ontology.txt')
line = onto.readline()
while (line):
```

```
          line = string.split(line, '\t')
          ontology.append(line)
          line = onto.readline()
ClearRegister()
BaseReader('/users/mauriceling/mauricehons/d12p-d2l.txt')
SpotRegisterIntensity = SpotIntensityToFloat(SpotRegisterIntensity)
for s in range(len(SpotRegister)):
          for r in range(len(ontology)):
                    if (SpotRegister[s][0] == ontology[r][0]):
                              SpotRegister[s].append(ontology[r][1:])
m = [103*2, 328*2]
print "LIST OF LESS THAN 2 FOLD CHANGES OF MORE THAN 2 STANDARD
DEVIATIONS OF BACKGROUND (ANY CHANNEL) IN "
print "DAY 12 PREGNANT VERSUS DAY 2 LACTATION"
register = []
lap = []
fold2 = []
del register[0:]
del lap[0:]
del fold2[0:]
lap_count = 0
fold2_count = 0
for s in range(len(SpotRegisterIntensity)):
          if (SpotRegisterIntensity[s][0] > m[0] or
                              SpotRegisterIntensity[s][1] > m[1]):
                    register.append(SpotRegister[s])
for s in range(len(register)):
          #one channel more than cutoff, the other more than zero
          if ((float(register[s][2]) > m[0] and
                              float(register[s][3]) > 0) or
                              (float(register[s][3]) > m[1] and
                              float(register[s][2]) > 0)):
                                        lap.append(register[s])
                                        lap_count = lap_count + 1
for s in range(len(lap)):
          if ((float(lap[s][2])/float(lap[s][3]) < 2) and
                    (float(lap[s][2])/float(lap[s][3]) > 0.5)):
                    fold2.append(lap[s])
                    fold2_count = fold2_count + 1
print
print
for s in range(len(fold2)):
          print fold2[s]
print
print
print "Overlap count: " + str(lap_count)
print "<2X difference count: " + str(fold2_count)
```

Result

(See List-8-result-d12p-d2l.rtf)
(See List-8-result-d12p-ifp.rtf)
(See List-8-result-ifp-d2l.rtf)
(See List-8-result-ifp-fp.rtf)
(See List-8-result-if-ifp.rtf)
(See List-8-result-ifp-ip.rtf)

Listing 9

Same as Listing 3 (listing single expression) with gene ontology appended on.

```
# (c) 2004 Maurice Ling
# BSc(Hons), The University of Melbourne, Dept of Zoology
```

```
# created on 11TH of March, 2004

from basereader import *
import string

ontology = []
del ontology[0:]
onto = open('/users/mauriceling/mauricehons/nia15k gene ontology.txt')
line = onto.readline()
while (line):
          line = string.split(line, '\t')
          ontology.append(line)
          line = onto.readline()
ClearRegister()
BaseReader('/users/mauriceling/mauricehons/d12p-d2l.txt')
SpotRegisterIntensity = SpotIntensityToFloat(SpotRegisterIntensity)
for s in range(len(SpotRegister)):
          for r in range(len(ontology)):
                    if (SpotRegister[s][0] == ontology[r][0]):
                              SpotRegister[s].append(ontology[r][1:])
m = [206, 656]
print "LIST OF NON-OVERLAPS FOR MORE THAN 2 STANDARD DEVIATIONS FROM
BACKGROUND"
print "DAY 12 PREGNANT VERSUS DAY 2 LACTATION"
register = []
one = []
two = []
del register[0:]
del one[0:]
del two[0:]
one_count = 0
two_count = 0
for s in range(len(SpotRegisterIntensity)):
          if (SpotRegisterIntensity[s][0] > m[0] or
                              SpotRegisterIntensity[s][1] > m[1]):
                    register.append(SpotRegister[s])
for s in range(len(register)):
          #check for int1 < m[0], then int2 should be > m[1]
          if (float(register[s][2]) < m[0]):
                    two.append(register[s])
                    two_count = two_count + 1
          #if int2 <  m[1], the int1 should be > m[0]
          if (float(register[s][3]) < m[1]):
                    one.append(register[s])
                    one_count = one_count + 1
print
print "INTENSITY ONE ONLY"
for s in range(len(one)):
          print one[s]
print
print
print "INTENSITY TWO ONLY"
for s in range(len(two)):
          print two[s]
print
print
print "Intensity one only count: " + str(one_count)
print "Intensity two only count: " + str(two_count)
```

Result

(See List-9-result-d12p-d2l.rtf)
(See List-9-result-d12p-ifp.rtf)
(See List-9-result-ifp-d2l.rtf)
(See List-9-result-ifp-fp.rtf)
(See List-9-result-if-ifp.rtf)

(See List-9-result-ifp-ip.rtf)

Listing 10

Using 2X of standard deviations of background as cut-off values, it first looks for common expressions in all microarray scan data (one of the two channels to be above cut-off value while the other to be above zero). Secondly, it looks for commonly lighted spots across each of the two experiments, also known as “common of common expressions”. Thirdly, it adds gene ontology information to the spots and subtracts the set of common of common expressions from the common expressions in each microarray. Finally, sorting according to difference in expressions (intensities) for all of the subtracted common expressions and prints out the data.

```
# (c) 2004 Maurice Ling
# BSc(Hons), The University of Melbourne, Dept of Zoology
# created on 16th of March, 2004

#processing files (culture verification)
from basereader import *
ClearRegister()
BaseReader('/users/mauriceling/mauricehons/d12p-d2l.txt')
SpotRegisterIntensity = SpotIntensityToFloat(SpotRegisterIntensity)
m = [103*2, 328*2]
register = []
lap = []
del register[0:]
del lap[0:]
for s in range(len(SpotRegisterIntensity)):
          if (SpotRegisterIntensity[s][0] > m[0] or
                              SpotRegisterIntensity[s][1] > m[1]):
                    register.append(SpotRegister[s])
for s in range(len(register)):
          if ((float(register[s][2]) > m[0] and
                              float(register[s][3]) > 0) or
                              (float(register[s][3]) > m[1] and
                              float(register[s][2]) > 0)):
                                        lap.append(register[s])
d12pd2l = []
del d12pd2l[0:]
d12pd2l = lap

from basereader import *
ClearRegister()
BaseReader('/users/mauriceling/mauricehons/d12p-ifp.txt')
SpotRegisterIntensity = SpotIntensityToFloat(SpotRegisterIntensity)
m = [1212, 4428]
register = []
lap = []
del register[0:]
del lap[0:]
lap_count = 0
for s in range(len(SpotRegisterIntensity)):
          if (SpotRegisterIntensity[s][0] > m[0] or
                              SpotRegisterIntensity[s][1] > m[1]):
                    register.append(SpotRegister[s])
for s in range(len(register)):
          #one channel more than cutoff, the other more than zero
          if ((float(register[s][2]) > m[0] and
                              float(register[s][3]) > 0) or
                              (float(register[s][3]) > m[1] and
                              float(register[s][2]) > 0)):
                                        lap.append(register[s])
```

```
d12pifp = []
del d12pifp[0:]
d12pifp = lap

from basereader import *
ClearRegister()
BaseReader('/users/mauriceling/mauricehons/ifp-d2l.txt')
SpotRegisterIntensity = SpotIntensityToFloat(SpotRegisterIntensity)
m = [1976, 1444]
register = []
lap = []
del register[0:]
del lap[0:]
for s in range(len(SpotRegisterIntensity)):
          if (SpotRegisterIntensity[s][0] > m[0] or
                              SpotRegisterIntensity[s][1] > m[1]):
                    register.append(SpotRegister[s])
for s in range(len(register)):
          #one channel more than cutoff, the other more than zero
          if ((float(register[s][2]) > m[0] and
                              float(register[s][3]) > 0) or
                              (float(register[s][3]) > m[1] and
                              float(register[s][2]) > 0)):
                                        lap.append(register[s])
ifpd2l = []
del ifpd2l[0:]
ifpd2l = lap

#processing files (hormone treatment)
from basereader import *
ClearRegister()
BaseReader('/users/mauriceling/mauricehons/ifp-fp.txt')
SpotRegisterIntensity = SpotIntensityToFloat(SpotRegisterIntensity)
m = [3924, 2122]
register = []
lap = []
del register[0:]
del lap[0:]
for s in range(len(SpotRegisterIntensity)):
          if (SpotRegisterIntensity[s][0] > m[0] or
                              SpotRegisterIntensity[s][1] > m[1]):
                    register.append(SpotRegister[s])
for s in range(len(register)):
          #one channel more than cutoff, the other more than zero
          if ((float(register[s][2]) > m[0] and
                              float(register[s][3]) > 0) or
                              (float(register[s][3]) > m[1] and
                              float(register[s][2]) > 0)):
                                        lap.append(register[s])
ifpfp = []
del ifpfp[0:]
ifpfp = lap

from basereader import *
ClearRegister()
BaseReader('/users/mauriceling/mauricehons/if-ifp.txt')
SpotRegisterIntensity = SpotIntensityToFloat(SpotRegisterIntensity)
m = [26080, 4188]
register = []
lap = []
del register[0:]
del lap[0:]
for s in range(len(SpotRegisterIntensity)):
          if (SpotRegisterIntensity[s][0] > m[0] or
                              SpotRegisterIntensity[s][1] > m[1]):
                    register.append(SpotRegister[s])
for s in range(len(register)):
```

```
          #one channel more than cutoff, the other more than zero
          if ((float(register[s][2]) > m[0] and
                              float(register[s][3]) > 0) or
                              (float(register[s][3]) > m[1] and
                              float(register[s][2]) > 0)):
                                        lap.append(register[s])
ififp = []
del ififp[0:]
ififp = lap

from basereader import *
ClearRegister()
BaseReader('/users/mauriceling/mauricehons/ifp-ip.txt')
SpotRegisterIntensity = SpotIntensityToFloat(SpotRegisterIntensity)
m = [1748, 3330]
register = []
lap = []
del register[0:]
del lap[0:]
for s in range(len(SpotRegisterIntensity)):
          if (SpotRegisterIntensity[s][0] > m[0] or
                              SpotRegisterIntensity[s][1] > m[1]):
                    register.append(SpotRegister[s])
for s in range(len(register)):
          #one channel more than cutoff, the other more than zero
          if ((float(register[s][2]) > m[0] and
                              float(register[s][3]) > 0) or
                              (float(register[s][3]) > m[1] and
                              float(register[s][2]) > 0)):
                                        lap.append(register[s])
ifpip = []
del ifpip[0:]
ifpip = lap

#common of common expressions (culture verification)
culture_common2 = []
del culture_common2[0:]
for s in range(len(d12pd2l)):
          for r in range(len(d12pifp)):
                    if (d12pd2l[s][0] == d12pifp[r][0]):
                              culture_common2.append(d12pd2l[s][0:2])
culture_common3 = []
del culture_common3[0:]
for s in range(len(culture_common2)):
          for r in range(len(ifpd2l)):
                    if (culture_common2[s][0] == ifpd2l[r][0]):
                              culture_common3.append(culture_common2[s
])

#common of common expressions (hormone treatment)
hormone_common2 = []
del hormone_common2[0:]
for s in range(len(ifpfp)):
          for r in range(len(ififp)):
                    if (ifpfp[s][0] == ififp[r][0]):
                              hormone_common2.append(ifpfp[s][0:2])
hormone_common3 = []
del hormone_common3[0:]
for s in range(len(hormone_common2)):
          for r in range(len(ifpip)):
                    if (hormone_common2[s][0] == ifpip[r][0]):
                              hormone_common3.append(hormone_common2[s
])

#common less common of commons (culture verification)
for s in range(len(ifpd2l)):
          for r in range(len(culture_common3)):
```

```
                    if (ifpd2l[s][0] == culture_common3[r][0]):
                              ifpd2l[s].insert(0, 'COMMON')
ifpd2lonly = []
del ifpd2lonly[0:]
for s in ifpd2l:
          if (s[0] != 'COMMON'):
                    ifpd2lonly.append(s)

for s in range(len(d12pd2l)):
          for r in range(len(culture_common3)):
                    if (d12pd2l[s][0] == culture_common3[r][0]):
                              d12pd2l[s].insert(0, 'COMMON')
d12pd2lonly = []
del d12pd2lonly[0:]
for s in d12pd2l:
          if (s[0] != 'COMMON'):
                    d12pd2lonly.append(s)

for s in range(len(d12pifp)):
          for r in range(len(culture_common3)):
                    if (d12pifp[s][0] == culture_common3[r][0]):
                              d12pifp[s].insert(0, 'COMMON')
d12pifponly = []
del d12pifponly[0:]
for s in d12pifp:
          if (s[0] != 'COMMON'):
                    d12pifponly.append(s)

#common less common of commons (hormone treatment)
for s in range(len(ifpfp)):
          for r in range(len(hormone_common3)):
                    if (ifpfp[s][0] == hormone_common3[r][0]):
                              ifpfp[s].insert(0, 'COMMON')
ifpfponly = []
del ifpfponly[0:]
for s in ifpfp:
          if (s[0] != 'COMMON'):
                    ifpfponly.append(s)

for s in range(len(ifpip)):
          for r in range(len(hormone_common3)):
                    if (ifpip[s][0] == hormone_common3[r][0]):
                              ifpip[s].insert(0, 'COMMON')
ifpiponly = []
del ifpiponly[0:]
for s in ifpip:
          if (s[0] != 'COMMON'):
                    ifpiponly.append(s)

for s in range(len(ififp)):
          for r in range(len(hormone_common3)):
                    if (ififp[s][0] == hormone_common3[r][0]):
                              ififp[s].insert(0, 'COMMON')
ififponly = []
del ififponly[0:]
for s in ififp:
          if (s[0] != 'COMMON'):
                    ififponly.append(s)

#prepare gene ontology
import string
ontology = []
del ontology[0:]
onto = open('/users/mauriceling/mauricehons/nia15k gene ontology.txt')
line = onto.readline()
while (line):
          line = string.split(line, '\t')
```

```
          ontology.append(line)
          line = onto.readline()

#add gene ontology to all
for s in range(len(culture_common3)):
          for r in range(len(ontology)):
                    if (culture_common3[s][0] == ontology[r][0]):
                              culture_common3[s].append(ontology[r][1:
])

for s in range(len(d12pd2lonly)):
          for r in range(len(ontology)):
                    if (d12pd2lonly[s][0] == ontology[r][0]):
                              d12pd2lonly[s].append(ontology[r][1:])

for s in range(len(d12pifponly)):
          for r in range(len(ontology)):
                    if (d12pifponly[s][0] == ontology[r][0]):
                              d12pifponly[s].append(ontology[r][1:])

for s in range(len(ifpd2lonly)):
          for r in range(len(ontology)):
                    if (ifpd2lonly[s][0] == ontology[r][0]):
                              ifpd2lonly[s].append(ontology[r][1:])

for s in range(len(hormone_common3)):
          for r in range(len(ontology)):
                    if (hormone_common3[s][0][1:-1] == ontology[r][0]):
                              hormone_common3[s].append(ontology[r][1:
])

for s in range(len(ifpfponly)):
          for r in range(len(ontology)):
                    if (ifpfponly[s][0][1:-1] == ontology[r][0]):
                              ifpfponly[s].append(ontology[r][1:])

for s in range(len(ififponly)):
          for r in range(len(ontology)):
                    if (ififponly[s][0][1:-1] == ontology[r][0]):
                              ififponly[s].append(ontology[r][1:])

for s in range(len(ifpiponly)):
          for r in range(len(ontology)):
                    if (ifpiponly[s][0][1:-1] == ontology[r][0]):
                              ifpiponly[s].append(ontology[r][1:])

#sort according to fold differences
for s in range(len(d12pd2lonly)):
          d12pd2lonly[s].insert(0,
float(d12pd2lonly[s][2])/float(d12pd2lonly[s][3]))
d12pd2lonly.sort()

for s in range(len(d12pifponly)):
          d12pifponly[s].insert(0,
float(d12pifponly[s][2])/float(d12pifponly[s][3]))
d12pd2lonly.sort()

for s in range(len(ifpd2lonly)):
          ifpd2lonly[s].insert(0,
float(ifpd2lonly[s][2])/float(ifpd2lonly[s][3]))
d12pd2lonly.sort()

for s in range(len(ifpfponly)):
          ifpfponly[s].insert(0,
float(ifpfponly[s][2])/float(ifpfponly[s][3]))
ifpfponly.sort()
```

```
for s in range(len(ififponly)):
          ififponly[s].insert(0,
float(ififponly[s][2])/float(ififponly[s][3]))
ififponly.sort()

for s in range(len(ifpiponly)):
          ifpiponly[s].insert(0,
float(ifpiponly[s][2])/float(ifpiponly[s][3]))
ifpiponly.sort()

#print results
print "LIST OF COMMON OF COMMON EXPRESSIONS FOR CULTURE VERIFICATION
EXPERIMENT"
for s in culture_common3:
          print s
          print
print
print "LIST OF COMMON EXPRESSIONS FOR D12P-D2L, LESS COMMON OF COMMON"
for s in d12pd2lonly:
          print s
          print
print
print "LIST OF COMMON EXPRESSIONS FOR D12P-IFP, LESS COMMON OF COMMON"
for s in d12pifponly:
          print s
          print
print
print "LIST OF COMMON EXPRESSIONS FOR IFP-D2L, LESS COMMON OF COMMON"
for s in ifpd2lonly:
          print s
          print
print
print "LIST OF COMMON OF COMMON EXPRESSIONS FOR HORMONE TREATMENT
EXPERIMENT"
for s in hormone_common3:
          print s
          print
print
print "LIST OF COMMON EXPRESSIONS FOR IFP-FP, LESS COMMON OF COMMON"
for s in ifpfponly:
          print s
          print
print
print "LIST OF COMMON EXPRESSIONS FOR IF-IFP, LESS COMMON OF COMMON"
for s in ififponly:
          print s
          print
print
print "LIST OF COMMON EXPRESSIONS FOR IFP-IP, LESS COMMON OF COMMON"
for s in ifpiponly:
          print s
          print
print
print
print "SUMMARY STATISTICS-------------------------------------------"
print "Common expressions for D12P-D2L: " + str(len(d12pd2l))
print "Common expressions for D12P-IFP: " + str(len(d12pifp))
print "Common expressions for IFP-D2L: " + str(len(ifpd2l))
print "Common expressions for IFP-FP: " + str(len(ifpfp))
print "Common expressions for IF-IFP: " + str(len(ififp))
print "Common expressions for IFP-IP: " + str(len(ifpip))
print "Common of common expressions for culture verification experiment:
" + str(len(culture_common3))
print "Common of common expressions for hormone treatment experiment: "
+ str(len(hormone_common3))
print "D12P-D2L specific: " + str(len(d12pd2lonly))
print "D12P-IFP specific: " + str(len(d12pifponly))
```

```
print "IFP-D2L specific: " + str(len(ifpd2lonly))
print "IFP-FP specific: " + str(len(ifpfponly))
print "IF-IFP specific: " + str(len(ififponly))
print "IFP-IP specific: " + str(len(ifpiponly))
```

Result
(See List-10-results.rtf)

Listing 11

Progressing from Listing 10, search for common of common of common expressions.

```
# (c) 2004 Maurice Ling
# BSc(Hons), The University of Melbourne, Dept of Zoology
# created on 16th of March, 2004

#processing files (culture verification)
from basereader import *
ClearRegister()
BaseReader('/users/mauriceling/mauricehons/d12p-d2l.txt')
SpotRegisterIntensity = SpotIntensityToFloat(SpotRegisterIntensity)
m = [103*2, 328*2]
register = []
lap = []
del register[0:]
del lap[0:]
for s in range(len(SpotRegisterIntensity)):
          if (SpotRegisterIntensity[s][0] > m[0] or
                              SpotRegisterIntensity[s][1] > m[1]):
                    register.append(SpotRegister[s])
for s in range(len(register)):
          if ((float(register[s][2]) > m[0] and
                              float(register[s][3]) > 0) or
                              (float(register[s][3]) > m[1] and
                              float(register[s][2]) > 0)):
                                        lap.append(register[s])
d12pd2l = []
del d12pd2l[0:]
d12pd2l = lap

from basereader import *
ClearRegister()
BaseReader('/users/mauriceling/mauricehons/d12p-ifp.txt')
SpotRegisterIntensity = SpotIntensityToFloat(SpotRegisterIntensity)
m = [1212, 4428]
register = []
lap = []
del register[0:]
del lap[0:]
lap_count = 0
for s in range(len(SpotRegisterIntensity)):
          if (SpotRegisterIntensity[s][0] > m[0] or
                              SpotRegisterIntensity[s][1] > m[1]):
                    register.append(SpotRegister[s])
for s in range(len(register)):
          #one channel more than cutoff, the other more than zero
          if ((float(register[s][2]) > m[0] and
                              float(register[s][3]) > 0) or
                              (float(register[s][3]) > m[1] and
                              float(register[s][2]) > 0)):
                                        lap.append(register[s])
d12pifp = []
del d12pifp[0:]
d12pifp = lap
```

```
from basereader import *
ClearRegister()
BaseReader('/users/mauriceling/mauricehons/ifp-d2l.txt')
SpotRegisterIntensity = SpotIntensityToFloat(SpotRegisterIntensity)
m = [1976, 1444]
register = []
lap = []
del register[0:]
del lap[0:]
for s in range(len(SpotRegisterIntensity)):
          if (SpotRegisterIntensity[s][0] > m[0] or
                              SpotRegisterIntensity[s][1] > m[1]):
                    register.append(SpotRegister[s])
for s in range(len(register)):
          #one channel more than cutoff, the other more than zero
          if ((float(register[s][2]) > m[0] and
                              float(register[s][3]) > 0) or
                              (float(register[s][3]) > m[1] and
                              float(register[s][2]) > 0)):
                                        lap.append(register[s])
ifpd2l = []
del ifpd2l[0:]
ifpd2l = lap

#processing files (hormone treatment)
from basereader import *
ClearRegister()
BaseReader('/users/mauriceling/mauricehons/ifp-fp.txt')
SpotRegisterIntensity = SpotIntensityToFloat(SpotRegisterIntensity)
m = [3924, 2122]
register = []
lap = []
del register[0:]
del lap[0:]
for s in range(len(SpotRegisterIntensity)):
          if (SpotRegisterIntensity[s][0] > m[0] or
                              SpotRegisterIntensity[s][1] > m[1]):
                    register.append(SpotRegister[s])
for s in range(len(register)):
          #one channel more than cutoff, the other more than zero
          if ((float(register[s][2]) > m[0] and
                              float(register[s][3]) > 0) or
                              (float(register[s][3]) > m[1] and
                              float(register[s][2]) > 0)):
                                        lap.append(register[s])
ifpfp = []
del ifpfp[0:]
ifpfp = lap

from basereader import *
ClearRegister()
BaseReader('/users/mauriceling/mauricehons/if-ifp.txt')
SpotRegisterIntensity = SpotIntensityToFloat(SpotRegisterIntensity)
m = [26080, 4188]
register = []
lap = []
del register[0:]
del lap[0:]
for s in range(len(SpotRegisterIntensity)):
          if (SpotRegisterIntensity[s][0] > m[0] or
                              SpotRegisterIntensity[s][1] > m[1]):
                    register.append(SpotRegister[s])
for s in range(len(register)):
          #one channel more than cutoff, the other more than zero
          if ((float(register[s][2]) > m[0] and
                              float(register[s][3]) > 0) or
```

```
                                (float(register[s][3]) > m[1] and
                                float(register[s][2]) > 0)):
                                        lap.append(register[s])
ififp = []
del ififp[0:]
ififp = lap

from basereader import *
ClearRegister()
BaseReader('/users/mauriceling/mauricehons/ifp-ip.txt')
SpotRegisterIntensity = SpotIntensityToFloat(SpotRegisterIntensity)
m = [1748, 3330]
register = []
lap = []
del register[0:]
del lap[0:]
for s in range(len(SpotRegisterIntensity)):
          if (SpotRegisterIntensity[s][0] > m[0] or
                                SpotRegisterIntensity[s][1] > m[1]):
                    register.append(SpotRegister[s])
for s in range(len(register)):
          #one channel more than cutoff, the other more than zero
          if ((float(register[s][2]) > m[0] and
                                float(register[s][3]) > 0) or
                                (float(register[s][3]) > m[1] and
                                float(register[s][2]) > 0)):
                                        lap.append(register[s])
ifpip = []
del ifpip[0:]
ifpip = lap

#common of common expressions (culture verification)
culture_common2 = []
del culture_common2[0:]
for s in range(len(d12pd2l)):
          for r in range(len(d12pifp)):
                    if (d12pd2l[s][0] == d12pifp[r][0]):
                                culture_common2.append(d12pd2l[s][0:2])
culture_common3 = []
del culture_common3[0:]
for s in range(len(culture_common2)):
          for r in range(len(ifpd2l)):
                    if (culture_common2[s][0] == ifpd2l[r][0]):
                                culture_common3.append(culture_common2[s
])

#common of common expressions (hormone treatment)
hormone_common2 = []
del hormone_common2[0:]
for s in range(len(ifpfp)):
          for r in range(len(ififp)):
                    if (ifpfp[s][0] == ififp[r][0]):
                                hormone_common2.append(ifpfp[s][0:2])
hormone_common3 = []
del hormone_common3[0:]
for s in range(len(hormone_common2)):
          for r in range(len(ifpip)):
                    if (hormone_common2[s][0] == ifpip[r][0]):
                                hormone_common3.append(hormone_common2[s
])

#common of common of common expressions
ccce = []
del ccce[0:]
for s in hormone_common3:
          for r in culture_common3:
                    if (s[0][1:-1] == r[0]):
```

```
                                ccce.append(r)

#print results
print "LIST OF COMMON OF COMMON OF COMMON EXPRESSIONS"
for s in ccce:
          print ccce
          print
print
print "Common of common expressions (culture verification): " +
str(len(culture_common3))
print "Common of common expressions (hormone treatments): " +
str(len(hormone_common3))
print "Common of common of common expressions: " + str(len(ccce))
```

Results
(See List-11-result.rtf)

Appendix E:
Culture Media, Buffers and Solutions

Water

DEPC • H_2O:	0.1% diethylpyrocarbonate
ipH_2O:	water for injection
milli-Q water:	deionized water (>18.2MΩ per cm)

Tissue Culture

M199 media:	4.75 g/L HEPES 2.2g/L sodium bicarbonate 3.4 ml/L L-glutamine 1.25µg/ml Fungizone 10% Bovine serum albumin 104 µg/ml Penicillin / Streptomycin pH 7.55 at room temperature and filter sterilized

DNA

1X TAE:	0.4 M Tris-acetate 1mM disodium EDTA (pH 8.0)
1.5% DNA agarose gel:	1.5% (w/v) agarose 1X TAE 0.5 µg/ml ethidium bromide

RNA

0.5X TBE:	5.4 g Tris 2.74 g Boric acid 2 ml of 0.5M disodium EDTA (pH 8.0) Make up to 1L with milli-Q water
1% RNA agarose gel:	1% (w/v) agarose 0.5X TBE 0.5µg/ml ethidium bromide

Appendix F:
Microarray Stock Solutions

Oligo dT Primer
Order as individual primers from Geneworks (1000nmoles, HPLC purity, 5'-3')

Oligo dT(A)	TTT TTT TTT TTT TTT TTA
Oligo dT(G)	TTT TTT TTT TTT TTT TTG
Oligo dT(CA)	TTT TTT TTT TTT TTT TCA
Oligo dT(CG)	TTT TTT TTT TTT TTT TCG
Oligo dT(CC)	TTT TTT TTT TTT TTT TCC

Resuspend each oligo at 10µg/µl
Vortex
Centrifuge for 1 minute
Combine 150µl of each
Make up volume to 3ml with IP • H_2O
Aliquot 40µl/tube
Store at -80°C

aadNTPs
(aminoallyl dNTPs)

Sigma #A0410 5-(3-aminoallyl)-2'-deoxyuridine 5'-triphosphate sodium salt
1mh vial

dATP (100mM)	47.8µl
dGTP (100mM)	47.8µl
dCTP (100mM)	47.8µl
aadUTP	Resuspend in 19.1µl
dTTP (100mM)	28.7µl

Aliquot 10µl volumes
Store at -80°C

Sodium Bicarbonate Buffer (pH 9.0)
Add 8.401g of sodium bicarbonate into 80ml milli-Q water
Adjust pH with 10M NaOH
Make up to 100ml with milli-Q water
Filter sterilize and store a 100µl aliquots of 1M solution at -20°C
Add 900µl of IP • H_2O and votex, prior to use in dye coupling step.

Blocking Solution
(for human and mouse microarray slides)

Human Cot-1 DNA (Invitrogen, 15279-011)
Mouse Cot-1 DNA (Invitrogen, 18440-016)

Yeast tRNA from *S. cerevisiae* (Sigma, R8759)

tRNA (4mg/ml)	3µl
Cot-1 DNA (10mg/ml) Human or Mouse	3µl
Poly dA (8mg/ml)	0.75µl
50X Denhart's Solution with herring sperm DNA	0.75µl

All components should be filter sterilized before use.
Mixture is aliquoted into 7.5µl volumes and stored at -20ºC.

Poly dA
Geneworks oligo (1000nmoles, sequencing purity, 50 bases)

50X Denhart's Solution: 0.5g Ficoll 400
0.5g Polyvinylpyrrolidone
0.5g Bovine Serum Albumin
Make up to 50ml with milli-Q water
Filter sterilize and store at -20ºC as 1ml aliquots.

20X SSC
175.3g of NaCl
88.2g of trisodium citrate
Dissolve in 900ml of milli-Q water and adjust pH to 7.0
Make up to 1000ml with milli-Q water.
Filter sterilize and store at room temperature.

10% SDS
Dissolve 100g of SDS in 800ml of milli-Q water
Make up to 1000ml with milli-Q water
Filter sterilize and store at room temperature.

Pre-hybridization Solution
1% BSA / 5X SSC / 0.1% SDS

Make 10% BSA solution and use laboratory stocks for SSC and SDS.

For 500ml: 50ml of 10% BSA

125ml 20X SSC
5ml 10% SDS
Make up to 500ml with milli-Q water
Filter sterilize and store at -20°C as 20ml aliquots.

Appendix G:
Partial List of Genes Differentially Expressed in Hormone-Treated Explants

This appendix shows the partial list of genes that were differentially expressed in hormone-treated mammary explant cultures as part of the results described in Chapter 5: Hormone-induced Gene Expression in Mammary Explants.

Effects of Insulin
(from Section 5.2.1.2)

Ribosomal Proteins (11 genes)
1. BG088791 Homo sapiens ribosomal protein L39 (RPL39), mRNA
2. BG074413 Mus musculus ribosomal protein L37a (Rpl37a), mRNA
3. BG085338 Mus musculus ribosomal protein L28 (Rpl28), mRNA
4. BG077162 Mus musculus ribosomal protein L3 (Rpl3), mRNA
5. BG086747 Human mRNA for ribosomal protein, complete cds
6. BG073412 Mus musculus ribosomal protein S7 (rpS7) gene, complete cds
7. BG086353 Mus musculus ribosomal protein S3 (Rps3), mRNA
8. BG072615 Mouse S16 ribosomal protein gene, complete cds
9. BG065195 M.musculus mRNA for ribosomal protein L5, 3\'end
10. BG088207 Mus musculus ribosomal protein L10A (Rpl10a), mRNA
11. BG087204 Mus musculus ribosomal protein L13a (Rpl13a), mRNA

Mitochondrial Proteins (7)
1. BG086028 Mus domesticus strain MilP mitocondrion genome, complete sequence
2. BG074900 Mus domesticus strain MilP mitocondrion genome, complete sequence
3. BG064785 Mus domesticus strain MilP mitocondrion genome, complete sequence
4. BG073099 Mus domesticus strain MilP mitocondrion genome, complete sequence
5. AU040550 Mus domesticus strain MilP mitocondrion genome, complete sequence
6. AW547111 Mus domesticus strain MilP mitocondrion genome, complete sequence
7. BI076451 Mus domesticus strain MilP mitocondrion genome, complete sequence

Effects of Glucocorticoid
(from Section 5.2.1.3)

Glycosylation-related Proteins (13 genes)
1. BG064540 Mouse man 6-P receptor (46MPR) mRNA, complete cds
2. BG078450 Mus musculus peptidylprolyl isomerase A (Ppia), mRNA
3. BG088585 Mus musculus dolichol-phosphate (beta-D) mannosyltransferase 2 (Dpm2), mRNA
4. BG078451 Mus musculus peptidylprolyl isomerase A (Ppia), mRNA
5. BG087495 Homo sapiens cDNA FLJ12564 fis, clone NT2RM4000833
6. BG065249 Mus musculus peptidylprolyl isomerase C (Ppic), mRNA
7. BG078453 Mus musculus peptidylprolyl isomerase A (Ppia), mRNA
8. BG087551 Homo sapiens UDP-glucose pyrophosphorylase 2 (UGP2), mRNA

9. BG088799 Mus musculus sialyltransferase 3 (Siat3), mRNA
10. BG078452 Mus musculus peptidylprolyl isomerase A (Ppia), mRNA
11. BG078454 Mus musculus peptidylprolyl isomerase A (Ppia), mRNA
12. BG064856 Mus musculus tripeptidyl peptidase II (Tpp2), mRNA
13. BG078452Mus musculus peptidylprolyl isomerase A (Ppia), mRNA

Chaperones (6 genes)
1. BG076350 Mus musculus Ccth gene for chaperonin containing TCP-1 eta subunit, complete cds
2. BG064814 Mus musculus chaperonin subunit 3 (gamma) (Cct3), mRNA
3. BG064811 Mus musculus chaperonin subunit 3 (gamma) (Cct3), mRNA
4. BG087043 Mus musculus heat shock 70 protein (Hsc70) gene, complete cds
5. BG079699 Mus musculus Cctd gene for chaperonin containing TCP-1 delta subunit, complete cds
6. BG064811 Mus musculus chaperonin subunit 3 (gamma) (Cct3), mRNA

Effects of Prolactin

(from Section 5.2.1.4)

MAP kinase Pathway Components (7 genes)
1. BG065128 Murine MAP kinase kinase 6c mRNA, complete cds
2. BG087465 Mus musculus mitogen activated protein kinase 14 (Mapk14), mRNA
3. BG083840 Mus musculus p38delta MAP kinase mRNA, complete cds
4. BG066357 Mus musculus mitogen activated protein kinase kinase kinase 12 (Map3k12), mRNA
5. BG084932 Mus musculus mitogen-activated protein kinase kinase kinase kinase 4 (Map4k4), mRNA
6. BG064189 Mus musculus MAP kinase-activated protein kinase 5 (Mapkapk5), mRNA
7. BG088472 M. musculus mRNA for MAP kinase-activated protein kinase 2

ATPases (5 genes)
1. BG083588 Mus musculus ATPase, H+ transporting, lysosomal (vacuolar proton pump), beta 56/58 kDa, isoform 2 (Atp6b2), mRNA
2. BG080527 Homo sapiens ATPase, H(+)-transporting, lysosomal, noncatalytic accessory protein 1B (ATP6N1B), mRNA
3. BG078023 Mus musculus 26S protease ATPase (mss1) mRNA, partial cds
4. BG071436 Homo sapiens proteasome (prosome, macropain) 26S subunit, ATPase, 6 (PSMC6), mRNA
5. BG065053 Mus musculus proteasome (prosome, macropain) 26S subunit, ATPase 3 (Psmc3), mRNA

Transcription Factors (10 genes)
1. BG088871 Mus musculus GATA-binding protein 4 (Gata4), mRNA
2. BG077360 Homo sapiens general transcription factor IIIA (GTF3A), mRNA
3. BG086978 Mus musculus transcription factor 20 (Tcf20), mRNA
4. BG077137 Mus musculus estrogen related receptor, alpha (Esrra), mRNA

5. BG086365 Mouse T-cell antigen receptor alpha-chain (TCR-ATF2) mRNA, partial cds
6. BG087490 Homo sapiens leucine-zipper-like transcriptional regulator, 1 (LZTR1), mRNA
7. BG086791 Homo sapiens CCR4-NOT transcription complex, subunit 8 (CNOT8), mRNA
8. BG078205 Mus musculus p53 binding protein 1 mRNA, partial cds
9. BG076421 Homo sapiens BPTF mRNA for bromodomain PHD finger transcription factor, complete cds
10. BG079912 Homo sapiens myeloid/lymphoid or mixed-lineage leukemia (trithorax (Drosophila) homolog); translocated to, 3 (MLLT3), mRNA

Translation Factors (7 genes)
1. BG085504 Homo sapiens eukaryotic translation initiation factor 2, subunit 1 (alpha, 35kD) (EIF2S1), mRNA
2. BG078205 Mus musculus p53 binding protein 1 mRNA, partial cds
3. BG073773 Mouse mRNA for elongation factor 1-alpha (EF 1-alpha)
4. BG063770 Mus musculus poly A binding protein, cytoplasmic 1 (Pabpc1), mRNA
5. BG087542 Mus musculus translational regulatory factor alpha NAC/1.9.2. protein mRNA, complete cds
6. BG085728 Mus musculus U5 small nuclear ribonucleoprotein 116 kDa (Snrp116-pending), mRNA
7. BG086427 Mus musculus protein synthesis elongation factor Tu (eEF-Tu, eEf-1-alpha) mRNA, complete cds