# WASHINGTON SEA GRANT PROGRAM

CIRCULAR NO. 71-11

A SYSTEMS ANALYSIS OF THE BRISTOL BAY SOCKEYE SALMON FISHERY

By Daniel H. McKenzie and Ole A. Mathisen

November 16, 1971 • University of Washington
College of Fisheries • Fisheries Research Institute

WSG 71-5

DIVISION OF MARINE RESOURCES
UNIVERSITY OF WASHINGTON 98105

Prepared under the
National Science Foundation
SEA GRANT PROGRAM

Fisheries Research Institute
College of Fisheries
University of Washington
Seattle, Washington 98195

Fisheries Research Institute Circular No. 71-11

# A SYSTEMS ANALYSIS OF THE BRISTOL BAY SOCKEYE SALMON FISHERY

by

Daniel H. McKenzie

and

Ole A. Mathisen

November 16, 1971

This work was supported by the
Washington Sea Grant Program
which is a part of the
National Sea Grant Program of the
National Oceanic and Atmospheric Administration,
U.S. Department of Commerce

Washington Sea Grant Publication No. 71-5

# A SYSTEMS ANALYSIS OF THE BRISTOL BAY SOCKEYE SALMON FISHERY

## INTRODUCTION

Commercial utilization of Bristol Bay sockeye salmon commenced in 1884 when a pack of salted salmon was prepared. During the subsequent years exploitation followed the same pattern as prevailed elsewhere along the Pacific Coast: At first production expanded with increasing fishing effort, but it soon leveled off, and then declined before stabilizing at a very much lower level.

Early in the history of the fishery a concern was voiced for the stocks and the need to provide an adequate escapement. This concern has been manifested since then in regulations aimed at limiting either the efficiency or the catchability coefficient of the gear. At first these regulations took the form of closures of sections adjacent to trunk stream estuaries to fishing, then soon thereafter by limitations on fishing time. These two types of restrictions have been the mainstay of fishery regulation in Bristol Bay, augmented by limitations on the length and mesh size of gill nets.

The development was a logical consequence of the absence of biological information vital to formulation of a scientific management scheme. Quantitative data were taken on catch each year and on escapement and age composition in some years, but a theoretical framework was lacking whereby this information could be utilized. Likewise all the data gathered by the industry itself, especially on effort, were not put to efficient use.

Since the end of the last world war, federal, state, and university agencies have been engaged in a large-scale, systematic study of the sockeye salmon stocks in Bristol Bay. As a result new data are generated in such volume that the old ways of preserving and disseminating them are quickly becoming obsolete.

Further, a rapid expansion in the field of population dynamics during the last two decades has made available techniques whereby the maximum sustainable yield can be estimated and thus rational management decisions can be made. Initially these were concerned largely with the biological aspects of optimum escapements. But since salmon fishing, like any other fishery, is an economic enterprise, the performance of any management scheme must be measured eventually in economic terms. Thus a number of alternative strategies present themselves, the choice of which depends on the interaction of biological and economic factors.

Concomitant with the development described above there has been a revolution in the development of high-speed computers and their application in almost any field of scientific and commercial enterprise. These tools are now being used to a very great extent by fishery biologists, and assuredly will be used to an even greater extent in the future. During the fishing season in Bristol Bay in 1971 a feasibility study was made with a field terminal connected to a remote processor in a cooperative venture among Alaska Department of Fish and Game, National Marine Fisheries Service, and the Fisheries Research Institute of the University of Washington.

The purpose of this paper is to outline a framework for future studies of management problems and strategies, to discuss briefly experiences gained in the use of computer access during the fishing season in Bristol Bay in the summer of 1971, and to describe in detail the construction of a data base for the Bristol Bay sockeye salmon fishery.

## MANAGEMENT OBJECTIVES IN RELATION TO THE LIFE HISTORY OF THE SOCKEYE SALMON

In salmon management, regulations are imposed on the fishery to provide the escapement that will produce the greatest return. To date this has been measured strictly in biological terms. The industrial operators need a forecast of the available catch, however, so that they can plan and equip commensurate with the available supply, which is the difference between the forecasted total return and the desired optimum escapement. The accuracy of the forecast is of utmost concern to the industry, but of no immediate concern to the biologist. A compromise is usually reached between the two.

Any forecasting technique is based on the life of the sockeye salmon in Bristol Bay, which ranges in length from 4 to 6 years. The salmon spends its life in a series of environments, linked together in time and space. It is spawned and resides during the next two or three years in freshwater. The freshwater phase culminates with migration through the estuary to the sea at smoltification. The estuarine phase is short but exceedingly critical. It spans the time needed by the smolts to move from the river mouth to commence feeding in the open sea. Ocean residence follows for a period of two or three years, and return migration begins with onset of maturity. Forecasts of total return have been made either on the basis of the size of the parent escapement, the size of the smolt migration, or the estimated oceanic abundance of immature or maturing salmon. A priori, one would expect a forecast to gain in accuracy as the time interval over which prediction is being made is reduced. Thus the most favorable forecast would be one based on estimated abundance in the sea. Unfortunately, the problems and cost relative to inshore sampling increase drastically with expansion of effort.

A penetrating analysis of the predictive value of the various forecasting methods and their associated costs, expressed in terms of economic risk to the industry and the fishermen, has not been made after the initial effort by Mathews (1967)[1]. Prior to such an analysis the available data must be assembled and made available to all biologists working in this field.

[1]Mathews, S. B. 1967. The economic consequences of forecasting sockeye salmon (Oncorhynchus nerka, Walbaum) runs to Bristol Bay, Alaska: A computer simulation study of the potential benefits to a salmon canning industry from accurate forecasts of the runs. Ph.D. Dissertation, University of Washington, Seattle, 238 p.

The same need exists with regard to optimum escapement size. Clearly, in the solution of this problem survival in all of the different life stages should be taken into consideration. For instance, for the period extending from spawning to smolt migration, the end products to be determined are the number of smolts produced and their average length and weight, which are the results of interaction of biotic and abiotic factors operating in the spawning grounds and in the freshwater nursery area. The resulting survival rate is an integrated expression of the action of these forces.

The simple concept that providing eggs is the sole function of the escapement is rapidly being replaced by an increasing understanding of the mechanical cleaning of the grounds by the redd-digging females and the biogenic enrichment of these grounds by the salmon carcasses. Feedback systems are being discovered between the various trophic levels and the production of juvenile sockeye salmon. Stock independence and genetic factors add a new dimension to the complexity of production in the nursery areas.

It is difficult to foresee that all relationships and interactions will be clarified in detail within the nearest future. Meanwhile, regulations must be formulated on the basis of existing information and decisions made, though they carry a certain element of risk. Computer simulation has proven to be a powerful tool in exploring alternative solutions.

Another difficult situation arises when the actual return falls short of the forecast. Adherence to the previously established escapement goal will usually mean great financial loss to industry and fishermen alike, and this must be weighed against the reproductive potential of various escapement levels before a rational management decision can be made. At the present time overfishing or overcautious management are the Scylla and Charybdis of economic success. One way to ameliorate the present situation is to explore simulation models of various alternative management schemes once a data base is available.

Essentially the same conclusion is reached where the third area of management decisions is concerned, the day-to-day regulation, where the catch is represented as the difference between estimated total run and optimum escapement size. The diagram in Fig. 1 outlines the various pieces of information available for management decisions today. One class deals with information gathered prior to the harvest, such as the test fishing data from Port Moller or immediately outside the fishing area. Another group includes information gathered during fishing operations, such as daily and cumulative total catch and expended effort, together with age, sex, and size composition of the catches. The third group concerns estimated escapement, derived from test fishing in the river estuaries, aerial surveys of the escapement in the trunk streams, and visual sample counts at selected locations. These indices vary in sampling variance and predictive value regarding daily and cumulative total escapement. But before one or more indices can be eliminated and the remaining ones improved, the past records must be examined for applicable relationships. Because of an incomplete or almost unavailable data base, analysis of these questions is highly impractical at the present time.

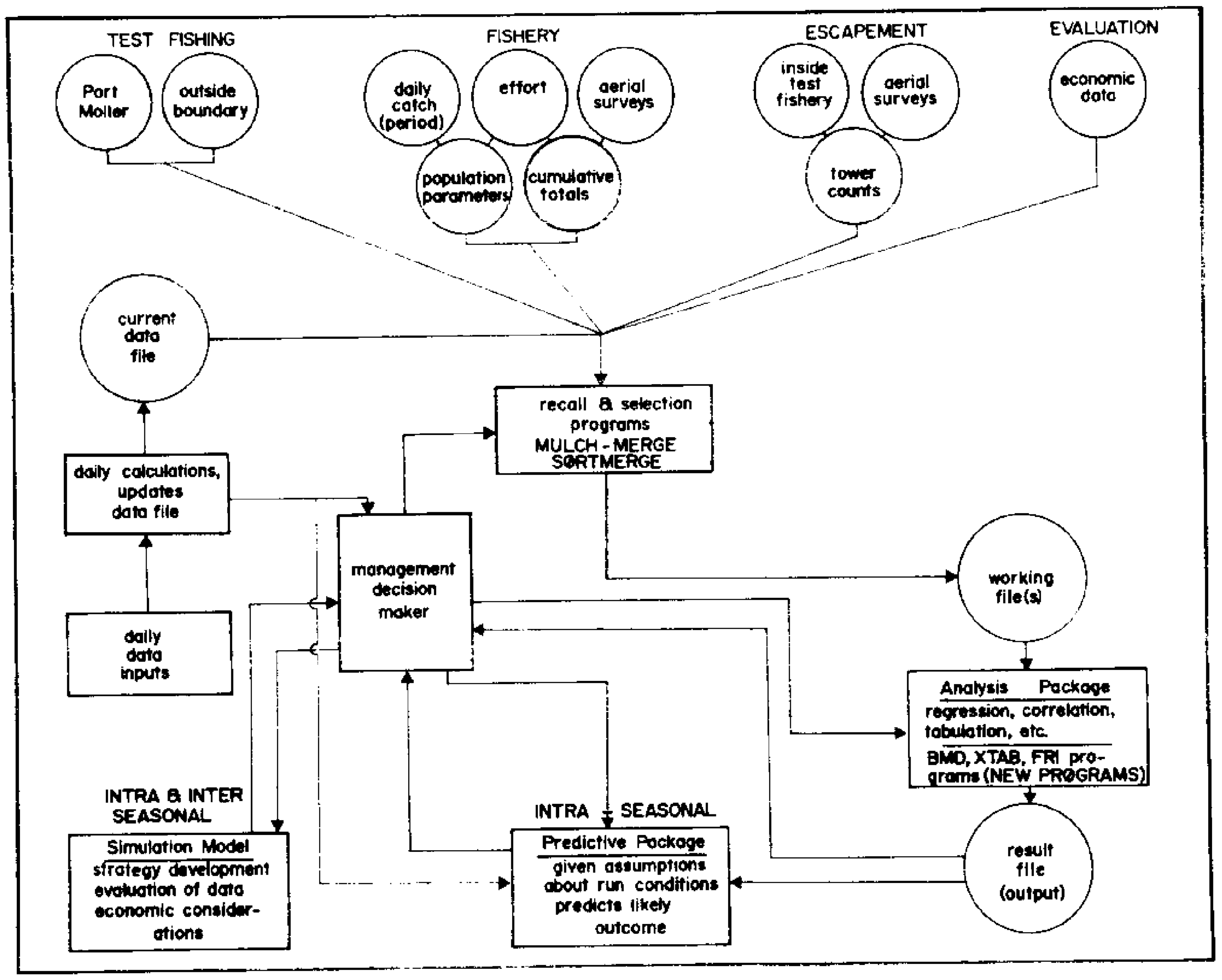


Fig. 1. Computer approach to management of the Bristol Bay sockeye salmon fishery.

At the present time there are no other ways in which regulations can affect production directly and decisively than in controlling the size of the total escapements and the composition of the various run segments. In the future, production may be enhanced by addition of fertilizers or by control of predators and food competitors in the nursery area, but control of the escapement remains the only feasible management tool. Today in Bristol Bay, efficient and accurate day-to-day regulation is complicated by the shortness of the fishing season and the restraints under which management must operate. By the provisions of the North Pacific treaty, harvest west of 175°E longitude cannot be controlled. Another restraint is the unrestricted entry into the fishery. Some constraints are economic in nature and related to the daily canning capacity, which in itself is an expression of the earning capacity of the fishery viewed over an extended time period.

## A DATA BASE FOR THE BRISTOL BAY FISHERY

In the three principal areas of salmon management - forecasting, assessment of the optimum escapement, and day-to-day decisions to achieve the established optimum escapement goals - success hinges on a complete and accessible data bank, or data base. A variety of remote terminals are now available that can be used to connect a field station with telephone facilities to any preselected computer center.

In the summer of 1971 a remote terminal with a high-speed printer was tried at the King Salmon base in Bristol Bay of the Alaska Department of Fish and Game. Through the existing telephone communication lines the terminal was connected with a Univac 1107 computer operated by Computer Science Corporation in Richland, Washington. No breakdown occurred in the lines of communications despite the use of a printer operating at a speed of 30 characters per second. A temporary data file was created and exploratory work started to estimate the cumulative daily escapement from data on catch per unit of effort, length of closed periods, and time of the season. These results will be discussed in a special report. It suffices here to state that this field trial demonstrated conclusively the feasibility of the system.

Since further progress in all three principal management areas - forecasting, estimation of optimum escapement, and intraseason regulations of the fishery - is contingent upon a data base, detailed instructions follow for its construction and maintenance. The three problem areas mentioned above differ from the biological point of view, and a separate base should be established for each. Of immediate concern is the intraseason regulation, and the proposed data base was planned with this problem in mind.

## Program USER GUIDE

### UPDATE system

UPDATE is a maintenance program that creates, corrects and manipulates the data files. It is maintained and documented by the Control Data Corporation. All corrections, additions, and expansion of the data bank are handled by this program. The data files are stored on a magnetic tape and UPDATE will be used to recall the specific type(s) of data to be analyzed. To use this program the user has to prepare two sets of cards, a set of control cards, and a set of text cards.

The first set of cards are used to control the operation of the UPDATE program and access the data files. The REQUEST card is used to make the data available to UPDATE and store the resulting corrected data on magnetic tape.

```
REQUEST,ØLDPL,VRN=mmm,HI,DI,IN.
```

This card creates a file on the computer named ØLDPL from the data file that was on magnetic tape number mmm.

```
REQUEST,NEWPL,VRN=xxx,HI,DI,ØUT.
```

This card saves the output file, NEWPL, from the UPDATE program on the tape number xxx. Note that there should be two tapes to provide a backup safety margin if the UPDATE program should fail.

The UPDATE program is controlled by the parameters and special options on the UPDATE card. Appendix I reproduced from the Scope Reference Manual[2] lists these parameters and their uses. However, the following examples should be sufficient explanation for most normal uses.

[2]Control Data 6000 Computer Systems, Scope Reference Manual, 6000 version 3.3, Publication No. 60305200, 1971.

The time and computer space requirements to run the UPDATE system are difficult to establish beyond the first approximation. In most cases a core storage request of 40000 octal is sufficient. Estimating the time requirement is much more difficult. It depends upon the number of cards handled, number of decks in the data library and the number of correction sets executed. However, the system is generally both fast and efficient and a 20 or 30 second time request will almost always complete the majority of jobs.

Example 1   Creating a Data Library

This example is to be utilized on the first run that creates the data file.

```
<job card>
REQUEST,NEWPL,VRN=xxx,HI,DI,ØUT.
UPDATE(N=NEWPL,F)
⚠RECØRD SEPARATØR CARD
*DECK dname
     DATA CARDS FOR DATA ØF TYPE 1
*DECK dname
     DATA CARDS FOR DATA ØF TYPE 2
.
.
.
*DECK dname
     DATA CARDS ØF LAST DATA TYPE
⚠END ØF JØB CARD
```

Each data type should have a unique deck name, dname, that is comprised of seven or fewer alphanumeric characters. The first character must be a letter. Some examples of valid names are: CATCH, KVIESC, C+E02, and TOWERS.

Example 2  Correcting and Updating the Data Library

Whenever it is necessary to correct an item in the data library or to insert additional information the following setup can be used.

```
<job card>
REQUEST,ØLDPL,VRN=xxx,HI,DI,IN.
REQUEST,NEWPL,VRN=mmm,HI,DI,ØUT.
UPDATE(P=ØLDPL,N=NEWPL,C=TAP,F)
△RECØRD SEPARATØR CARD
*IDENT idnam
*DELETE a.m
*INSERT c.m
     DATA CARD ØR CARDS TØ BE INSERTED INTØ THE LIBRARY
*INSERT d.l
     DATA TØ BE INSERTED
△END ØF JØB CARD
```

For this run the first request card recalls the previously created data library, and the second request card saves the corrected and updated data library. The update control card takes the old file, ØLDPL, creates the new file, NEWPL, using the input information from this run and writes the data on a file TAP. TAP is then available for listing or as input to a later program. The *IDENT introduces a correction set to the UPDATE system. All cards are identified by idnam, which is subject to the same rules as deck names. The *DELETE card or cards removes cards from the data library. In the example, the card identified by deck name a and sequence number m is deleted. The *INSERT card adds the data cards, that follow, to the data library immediately after card m in deck c. The second *INSERT card performs the same function in deck d after card 1. As many insertions and deletions as needed can be processed in any one run. The parameters a,c,d,n,m,l are obtained from the UPDATE listing of a previous run. To obtain a listing from the present program the file, TAP, can be copied to the ØUTPUT file, by inserting CØPYSBF(TAP,ØUTPUT), immediately after the UPDATE card.

The same general setup can also be used to add new data types to the program library. To do this use the *INSERT to insert the data after the last card of any of the previous data types. The first card to be inserted should be *DECK dname. This process allows the data library to be expanded.

Example 3   Selection of Data for Further Processing

The UPDATE system provides a very easy and efficient way of selecting certain sections of the data library.

```
<job card>
REQUEST,ØLDPL,VRN=mmm,HI,DI,IN.
UPDATE(P=ØLDPL,C=TAP)
(control cards for desired programs using TAP as input)
△7 RECØRD SEPARATØR CARD
*CØMPILE a,b,c
△7 RECØRD SEPARATØR CARD
   program deck
△7 RECØRD SEPARATØR CARD
   parameter cards for program
△6 END ØF JØB CARD
```

The *CØMPILE card directs UPDATE to write decks a, b, and c onto the file TAP. The user program can then utilize this file as input for subsequent analysis. To use this option the full assembly feature (F) must not be included on the UPDATE card.

Example 4 Housecleaning Process

The UPDATE system accumulates the corrections and additions to the data library. After a large number of such runs it becomes increasingly inefficient to maintain the record of changes. To correct this the following run should be made whenever the number of changes exceeds twenty.

```
<job card>
REQUEST,ØLDPL,VRN=mmm,HI,DI,IN.
REQUEST,UPDPL,VRN=xxx,HI,DI,ØUT.
UPDATE(P=ØLDPL,S=TAPL,F)
UPDATE(N=UPDPL,I=TAPL,F)
△ RECØRD SEPARATØR CARD
△ RECØRD SEPARATØR CARD
△ END OF JØB CARD
```

PRØGRAM SELECT

The program SELECT[3] allows the user to select specific sections of the data and to combine data from two or more data types into one format. When using the option that combines two data types the user can also specify a time lag factor or accumulate daily data and combine it with periodic data. This structure will allow the user great flexibility in recalling the data to be analyzed in subsequent programs.

[3]Daniel H. McKenzie, SELECT: A data retrieval program for the Bristol Bay Sockeye Salmon Bank. Fisheries Research Program No. FRM 328, Fisheries Research Institute, University of Washington, 1971.

Parameter cards

(integer values)

| | | | | |
|---|---|---|---|---|
| Card 1 | Col | 1-4 | KC1 | Data code of first data type |
| | | 5-8 | KC2 | Data code of second data type |
| | | 9-12 | KC3 | Data code for resulting data |
| | | 13-16 | KLAG | Time lag to be used in combining data |
| | | 17-20 | NV1 | Number of variables of first data type |
| | | 21-24 | NV2 | Number of variables of second data type |
| | | 25-28 | IYRB | First year to be selected |
| | | 29-32 | IYRE | Last year to be selected |
| | | 33-36 | KCOM | 1-one data code 2-combines two data codes |
| | | 37-40 | KCUM | 1-no accumulation 2-accumulates between dates |
| Card 2 | Col | 1-10 | KCD1 | System code of first data type |
| | | 11-20 | KCD2 | System code of second data type |
| | | 21-30 | KCD3 | System code for resulting data |
| Card 3 | Col | 1-80 | FMT1 | Format for reading first data type |
| Card 4 | Col | 1-80 | FMT2 | Format for reading second data type |
| Card 5 | Col | 1-80 | FMT3 | Format for outputing resultant data type |

Usage

The user will almost always utilize the UPDATE system to select the data from the data library. Data of code one, KC1, must be on TAPE1 and data of code two, KC2, must be on TAPE2. SELECT writes the output data onto TAPE3. If cards are used as input they must be copied onto the respective tape files before executing SELECT. The number of variables read, NV1 and NV2, must include as the first 4 variables, data code, system code, year, and date (month and day as one variable). This is also true for the formats, FMT1, FMT2, and FMT3. Note that FMT3 must contain fields for NV1+NV2-4 data fields, if two data codes are combined; if only one data code onlv NV1 fields are needed.

Restrictions:

The program will process a maximum of ten variables per data code as input, $NV1 \leq 10$, $NV2 \leq 10$. The number of data points selected must be less than 500 for normal usage. If greater capacity is required the dimension statements can be increased, however, 500 will probably be sufficient for most applications. Any number of parameter card sets may be submitted at one time, with the resulting data accumulating on TAPE3 as a single file.

Example 5   Selection of Catch and Effort Data for 1964 and Produce a Listing

```
<job card>
REQUEST,ØLDPL,VRN=xxx,HI,DI,IN.
NØREDUCE.
UPDATE(P=ØLDPL,C=TAPE1)
REDUCE.
FØRTRAN.
SETCØRE.
REWIND(TAPE1)
LGØ.
REWIND(TAPE3)
CØPYSBF(TAPE3,ØUTPUT)
 △RECØRD SEPARATØR CARD
*CØMPILE C&E02 (previously created UPDATE deck of catch and effort data)
```

```
△7 RECØRD SEPARATØR CARD

SELECT program deck

△7 RECØRD SEPARATØR CARD

  02       02   0   9      64  64   1   1

NAK-KVI                NAK-KVI

(I2,A10,I2,F4,F3,2F6,F10,F3)

Blank card

(I2,A10,I2,F4,F3,2F6,F10,F3)

△6 END ØF JØB CARD
```

Example 6   Selection of Catch and Effort Data and Combining with Escapement Data for 1964 Lagging Escapement by Seven Days and Punch Cards of the Resulting Data

```
<job card>

REQUEST,ØLDPL,VRN=mmm,HI,DI,IN.

NØREDUCE.

UPDATE(P=ØLDPL,C=TAPE1)

UPDATE(P=ØLDPL,C=TAPE2)

FØRTRAN.

SETCØRE.

REWIND(TAPE1,TAPE2)

LGØ.

REWIND(TAPE3)

CØPYBF(TAPE3,PUNCH)

△7 RECØRD SEPARATØR CARD
```

```
*CØMPILE C&E02 (UPDATE deck of catch and effort)

△7 RECØRD SEPARATØR CARD

*CØMPILE TC03 (UPDATE deck of tower counts)

△7 RECØRD SEPARATØR CARD

SELECT program deck

△7 RECØRD SEPARATØR CARD

  02  03  23   7   9   5  64 64   2   2

NAK-KVI   KVICHAK   NAK-KVI

(I2,A10,I2,F4,F3,2F6,F10,F3)

(I2,A10,I2,F4,F10)

(I2,A10,I2,F4,F3,2F6,F10,F3,F10)

△6 END ØF JØB CARD
```

## Example 7   Selection of Three Data Types

This example combines three data types and prints a listing of the results.

```
<job card>

REQUEST,ØLDPL,VRN=xxx,HI,DI,IN.

NØREDUCE.

UPDATE(P=ØLDPL,C=TAPE1)

UPDATE(P=ØLDPL,C=TAPE2)

FØRTRAN.

LGØ.

REWIND, TAPE3,TAPE1,LGØ.

CØPYBF,TAPE3,TAPE1.

REWIND,TAPE1,TAPE2.

UPDATE(P=ØLDPL,C=TAPE2)
```

```
REWIND,TAPE2.
REDUCE.
LGØ.
REWIND,TAPE3.
CØPYSBF,TAPE3,ØUTPUT.
△RECØRD SEPARATØR CARD
*CØMPILE dname (see example 6)
△RECØRD SEPARATØR CARD
*CØMPILE dname
△RECØRD SEPARATØR CARD
SELECT program deck
△RECØRD SEPARATØR CARD
    parameter cards for 1st select (see example 6)
△RECØRD SEPARATØR CARD
*CØMPILE dname
△RECØRD SEPARATØR CARD
    parameter cards for 2nd select.
▲END ØF JØB CARD.
```

Example 8   Select Two Data Types and Run the BMD Program BMD02R, Stepwise Regression (two separate jobs)

```
<job card>
REQUEST,ØLDPL,VRN=xxx,HI,DI,IN.
NØREDUCE.
UPDATE(P=ØLDPL,C=TAPE1)
UPDATE(P=ØLDPL,C=TAPE2)
REWIND,TAPE1,TAPE2.
```

```
FØRTRAN.
REDUCE.
SETCØRE.
LGØ.
REWIND,TAPE3,SAVEF.
CØPYBF,TAPE3,SAVEF.
REWIND,TAPE3.
CØPYSBF,TAPE3,ØUTPUT.
CØMMØN,SAVEF.
△7 RECØRD SEPARATØR CARD
*CØMPILE dname
△7 RECØRD SEPARATØR CARD
*CØMPILE dname
△7 RECØRD SEPARATØR CARD
SELECT program deck
△7 RECØRD SEPARATØR CARD
   parameter cards for selection
△6 END ØF JØB CARD
```

Second job (submitted after successful execution of first job)

```
<job card>
CØMMØN,SAVEF.
REWIND,SAVEF,ALTTAPE.
CØPYBF,SAVEF,ALTTAPE.
```

```
REWIND,ALTTAPE.

SETCØRE.

BMD02R.

△7 RECØRD SEPARATØR CARD

Control cards for BMD02R using tape 11 as input

△6 END OF JØB CARD
```

Program MULCH

The program MULCH[4] can be used to transform and screen the data prior to running a package program, for example the BMD series. This program has been written to give the user the power and generality of FØRTRAN when processing the data via the package programs. The following example demonstrates how one can use MULCH to calculate the CPUE from the common file, SAVEF, created on an earlier run by SELECT and then calculate the regression of CPUE and escapement.

Example 9 Calculation of CPUE from Catch, Units of Drift and Set Net Gear, and Hours of Fishing Time

```
<job card>

REQUEST,TAP,VRN=185,FILES=7-8,HI,DI,IN.

CØMMØN,SAVEF.
```

[4]Lawrence E. Gales, MULCH: A generalized data transformation program that prepares input data for other programs. Quantitative Science Paper No. 20, Center for Quantitative Science in Forestry, Fisheries and Wildlife, University of Washington, 1971.

```
REWIND,SAVEF.

NØREDUCE.

BEGIN(MULCH,TAP,TAP,INPUT,SAVEF,ALTTAPE,ØUTPUT)

BMD02R.

△7 RECØRD SEPARATØR CARD

NØ. ØF INPUT VARIABLES= 5

INPUT FØRMAT = (18XF3,2F6.1,2F10)

*      IF(EØFT) GØ TØ 201

       IF(X(1).GT.0.0.A.X(2).GT.0.0.A.X(3).GT.0.0.A.X(4).GT.0.0.A.

     +X(5).GT.0.0.DELETE=.F.

       IF(DELETE)GØ TØ 201

       X(6)=X(4)/(X(2)+X(3)*2.8)*X(1)/24.

       X(6)=ALØG10(X(6))

       X(5)=ALØG10( X(5)/10000.)

 201   CØNTINUE

ØUTPUT FØRMAT= (6F10.4)

THE ØUTPUT VARIABLES ARE X(1),X(2),X(3),X(4),X(5),X(6)

△7 RECØRD SEPARATØR CARD

     (parameter cards for BMD02R (see BMD Manual) using tape 11 as

     input on the PRØBLM card.

△6 END ØF JØB CARD
```

The two cards IF(EØFT) GØ TØ 201 and 201 CØNTINUE although not mentioned in the program description should always be included. When the program reads the end of the data tape it assigns zero to all the input variables and then proceeds to perform the transformations. If the transformations include division by one of the variables, or the logarithm function is used a mode error will occur causing the program to abort, unless these cards are included.

A simple application of this program would be to reformat the data. Several of the package programs, BMD especially, require that the X,Y variables be in a specific order. For this, MULCH requires only the four cards describing the input and output specifications.

The core requirements and time estimates for MULCH will vary with the amount of data processed and number of transformations performed. If MULCH is to be run alone, without other pre- or post-programs a CM of 50000 is sufficient. Unless the number of data transformations becomes excessive, more than 20 or 30, a time of 10 seconds per 1000 data cards processed will be adequate. If the MULCH program is expected to use the majority of the CM time and the subsequent programs require a large central memory it is much cheaper and more efficient to run two separate jobs. The reservation of large amounts of CM, which go unused during the MULCH program execution, for a BMD program can be avoided by using a COMMON file and running two separate jobs.

## Data Base Construction

The compilation of data for the data base is divisible into two separate tasks. The first of these is the job of obtaining the historical data and placing it onto the data library.

The second matter for consideration is the problem of entering current data into the data library. Here it is suggested that wherever feasible the data be taken directly from the observational forms upon which the data is originally recorded. This data can then be processed by the computer to generate both the summary reports for publication and the information for the data library. Such programs could be prepared or a variety of service bureau companies offer such systems in a generalized user package.

## Data Library Formats

In general each data source is catalogued by a data code, district, and date. This provides an efficient format for the retrieval of the data. The observational data is then fit into the appropriate format.

### I. Seasonal Performance Data, Totals

Description: Record of the total catch, escapement and run size for the entire bay, major districts, and river systems

FØRMAT:

| | | |
|---|---|---|
| Col. | 1-2 | DATA CODE 01 |
| | 3-12 | district identification |
| | | examples: BRISTØLBAY |
| | | NAK-KVI |
| | | KVICHAK |
| | 13-14 | year |
| | 15-24 | escapement |
| | 25-34 | catch |
| | 35-44 | total run size |

EXAMPLE:

```
01NAK-KVI   66   4965965   5397538  10363503
01NAKNEK    66   1016445   1092662   2109107
01EGIGIK    66    804246   2101174   2905420
```

## II. Period Catch and Effort

Description: Catch and effort data by district, period and gear type

FØRMAT:

|  |  |  |
|---|---|---|
| Col. | 1-2 | DATA CODE 02 |
|  | 3-12 | district identification<br>examples: NAK-KVI<br>EGEGIK<br>UGASHIK |
|  | 13-14 | year |
|  | 15-16 | month |
|  | 17-18 | day--date of opening of period |
|  | 19-21 | length of period (hrs) |
|  | 22-26 | number of units of drift gillnet effort |
|  | 27-32 | number of units of set gillnet effort |
|  | 33-42 | catch |
|  | 43-45 | area and gear code<br>example: |

Naknek-Kvichak district

| | |
|---|---|
| 1 | Entire district open |
| 2 | Kvichak district only |
| 3 | Naknek district only |
| 4 | Kvichak district + Naknek set nets |
| 5 | Naknek district + Kvichak set nets |

EXAMPLE:

```
02NAK-KVI   660620 48   480   142     36746  1
02NAK-KVI   660623 48   650   136     83714  1
```

Suggested additions:

It is suggested that a fishing conditions code be determined for the period that can be used to adjust the effort to estimate effective effort. Several factors operate to produce non-optimal or average fishing conditions. Inclement weather is reflected in the decreased number of deliveries, but a further reduction in effectiveness of the gear that does operate could be reflected in this parameter. On the other extreme, when the effort is on a daily limit imposed by the canneries, the number of deliveries does not adequately measure fishing effort. Thus a range of +5, -5 is suggested, with zero representing normal operating conditions. Intermediate points could then be estimated using such factors as, tidal change, day or night periods, weather, period length, etc.

III. Escapement

Description: Daily escapement estimates from tower counts

FØRMAT:

| | | |
|---|---|---|
| Col. | 1-2 | DATA CØDE 03 |
| | 3-12 | river or site identification |
| | 13-14 | year |
| | 15-16 | month |
| | 17-18 | day |
| | 19-28 | estimated daily escapement |

EXAMPLE:

```
03KVICHAK    660629       12
03KVICHAK    660630    12852
03NAKNEK     660620       24
```

Suggested additions:

An index of the conditions under which the tower counts were made should be included. The index would be a measure of the variance of the counts. A zero value would represent those conditions under which counting was unfeasible and indicate that the observation had been interpolated from other observations. A value of ten would be assigned to optimal conditions, with the intermediate points as a function of light conditions, weather conditions, and water conditions.

## IV. Aerial Escapement Survey

Description: Index counts of escapment from aerial surveys

FØRMAT: (following example for KVICHAK, other areas will differ slightly)

| | | |
|---|---|---|
| Col. | 1-2 | DATA CØDE 04 |
| | 3-12 | river identification |
| | 13-14 | year |
| | 15-16 | month |
| | 17-18 | day |
| | 19-22 | count, in 1000's, of $C_1 + C_2$ **(index areas)** |
| | 23-25 | $C_3+C_4$ |
| | 26-28 | $C_5+C_6$ |
| | 29-31 | $C_7+C_8$ |
| | 32-34 | $C_9+C_{10}$ |
| | 35-37 | $C_{11}+C_{12}$ |
| | 38-40 | $C_{13}+C_{14}$ |
| | 41-46 | estimated number index area |
| | 47-52 | estimated number NAKEEN to index area |

| | | |
|---|---|---|
| Col. | 53-58 | estimated number index area to tower |
| | 59-64 | total |
| | 65-66 | observer identification |

EXAMPLE:

```
04KVICHAK   700629  8 16 19  8 16 12  8   251   139    88   478SP
04KVICHAK   700705 89119173124161107184  2956  1602  2208  6766DB
```

Suggested addition:

An index representing observational conditions at the time of the survey should be included. The index would reflect the type of aircraft, water clarity, light conditions, and fish density. A value of 0-10 is suggested with 10 representing optimal conditions.

V. Inside Test Fishing

Description: Data from the inside test fishing program entered as catch per 1000 fathom minutes for each set

FØRMAT:

| | | |
|---|---|---|
| Col. | 1-2 | DATA CØDE 05 |
| | 3-12 | area identification |
| | 13-14 | year |
| | 15-16 | month |
| | 17-18 | day |
| | 19-21 | set number |
| | 22-24 | station identification |
| | 25-28 | gear length |
| | 29-31 | mean fishing time |
| | 32-35 | sockeye catch |
| | 36-42 | test fishing index |

EXAMPLE:

```
05NAKEEN    710701  8 50 30 100  66.67
```

Suggestion:

To maintain a high degree of efficiency and to minimize error it is suggested that this data be taken directly from the log sheets of the test boats. The data file should also include the relevant information that is associated with each set in the log book. Tide stage, water color, wind, etc. can be included. In following this suggestion the format should be altered to best fit the recorded information. For the cases where the data for a particular date or station is missing and the index is estimated the set number should reflect this fact.

## VI. Outside Test Fishing

Description: Data from the outside test fishing program entered as catch per 1000 fathom minutes for each set

FØRMAT and suggestion: See Section V - Inside Test Fishing

DATA CØDE 06

## VII. Offshore Test Fishing

Description: Data collected from the Port Moller based offshore test fishery recorded in catch per 100 fathom hours

FØRMAT and suggestions: See Section V - Inside Test Fishing

DATA CØDE 07

## VIII. Fishing Period Regulations

Description: The record of the opening and closing times for each fishing period by district

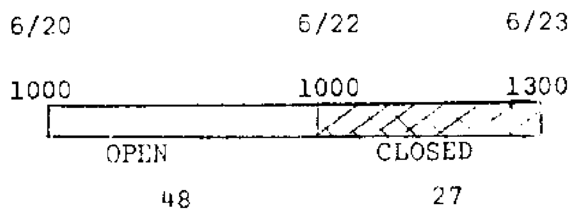
FØRMAT:

| | | |
|---|---|---|
| Col. | 1-2 | DATA CØDE 08 |
| | 3-12 | district identification |
| | 13-14 | year |
| | 15-16 | month of opening of period |
| | 17-18 | day of opening of period |
| | 19-22 | opening time |
| | 23-24 | month of closing of period |
| | 25-26 | day of closing of period |
| | 27-30 | closing time |
| | 31-33 | number of hours of fishing time |
| | 34-36 | number of hours in closed period |
| | 37-42 | fraction of elapsed time through season |
| | 43-44 | period code (ADF&G code) |

EXAMPLE:

```
08EGEGIK     660620100006221000 48 27  .037

08EGEGIK     660623130006251300 48 64  .130
```



$$\text{elapsed time} = \frac{(\text{hrs open} + \text{hrs closed})/2 + \Sigma(\text{open} + \text{closed periods})}{\text{Total season length}}$$

Total season length is to be defined as the length of time over which substantial catches are made in the fishery. For the Naknek-Kvichak District this has been approximately 40 days for the period 1960-1970. This index is an attempt to provide a time scale, independent of the calendar date, by which to measure the progress of the run.

## IX. Age and Sex Composition

Description: Age and sex composition of sockeye salmon from data from the escapement and catch by date

FØRMAT:

| | | | |
|---|---|---|---|
| Col. | 1-2 | DATA CØDE 09 | |
| | 3-12 | area identification | |
| | | example: NAK-KVI-C | |
| | | NAKNEK-E | |
| | | EGEGIK-C | |
| | 13-14 | year | |
| | 15-16 | month | |
| | 17-18 | day | |
| | 19 | 1 | sample from catch |
| | | 2 | sample from escapement |
| | 20 | 1 | male |
| | | 2 | female |
| | 21-25 | $3_2$ | |
| | 26-30 | $4_1$ | |
| | 31-35 | $4_2$ | |
| | 36-40 | $4_3$ | |
| | 41-45 | $5_2$ | |
| | 46-50 | $5_3$ | |
| | 51-55 | $6_3$ | |

| | |
|---|---|
| 56-60 | $6_4$ |
| 61-65 | $7_3$ |
| 66-70 | $7_4$ |

EXAMPLE:

```
09KVICHAK-E 66070821   10   20   17   44   29    1
09KVICHAK-E 66070822    7   19   27   38   41   10
```

X. Reference File

Description: This file should contain an explanation of all data codes and footnotes. It is essential that the interpretation of all coded material be included with the data. The inclusion of supplemental material can also be accommodated in this file.

FØRMAT:

| | | |
|---|---|---|
| Col. | 1-2 | DATA CØDE 00 |
| | 3-4 | DATA CØDE of referenced material |
| | 5-14 | identification of referenced material |
| | 15-16 | year |
| | 17-18 | month |
| | 19-20 | day |
| | 21-70 | text of reference material |

EXAMPLE:

```
0002NAK-KVI   660704 footnote on NAK-KVI catch of 7/4/66
```

## APPENDIX I

## PARAMETER VALUES

| Letter | Function | If not specified | Value to right of equals sign | Value if equals sign is omitted |
|---|---|---|---|---|
| P | Old program library file is processed by UPDATE. | File name OLDPL assumed | Old library file name | File name OLDPL assumed |
| N | New program library file is produced. | No new program library | New library file name | File name NEWPL assumed |
| I | Input stream contains directives and text processed by UPDATE. | File name INPUT assumed | Input file name | (Illegal) |
| O | Listable output file produced. | File name OUTPUT assumed | Output file name | (Illegal) |
| G | File containing PULLMOD output is produced. | PULLMOD output appended to source file | File to contain PULLMOD output | Output written to file name SOURCE |
| K | Same as C, except decks are written in the order specified on *COMPILE directives. | File name COMPILE assumed. Default is C | Compile file name | Output written to file name COMPILE |
| C | File produced contains source language decks in the order they appear on input and/or old program library, for assembler/compiler input. | File name COMPILE assumed | Compile file name; C=0 disables output to compile file | Output to file name COMPILE |
| T | Same as S, except common decks and *COMDECK directive card images are not written to the source file. | Source file not written | Source file name | File name SOURCE is assumed |
| S | Source file produced contains 80-column card images of all active cards, as well as their *DECK or *COMDECK directives. | Source file not written | Source file name | File name SOURCE is assumed |

PARAMETER VALUES (continued)

| Letter | Function | If not specified | Value to right of equals sign | Value if equals sign is omitted |
|---|---|---|---|---|
| M | Secondary library file is merged with an old program library file to form a new program library file. | No merging takes place | Name of secondary file | File name MERGE is assumed |

UPDATE ACTION PARAMETERS

| Letter | Specified Action | Default Action |
|---|---|---|
| F or Q | Selects Full or Quick mode for correction runs. | Normal (selective mode) is assumed (neither Q nor F). |
| A | Selects mode used to copy sequential OLDPL to random NEWPL. Suppresses all other action parameters except * and /. | None. |
| B | Selects mode used to copy random OLDPL to sequential NEWPL; suppresses all other action parameters except * and /. | None. |
| D | COMPILE file will contain 80 columns of data. | COMPILE file will contain 72 columns of data. |
| C=PUNCH | Selects both D and 8 options, as well as selecting PUNCH as compile file name. | None. |
| E | The OLDPL will be edited. | No editing will be done. |
| L=code | Selects type of listable output to be produced (see Listable Output). | L=A1234 on correction run; L-A12 on creation run. |
| R=list | Files specified in list will be rewound before and after UPDATE run. Only file parameter letters P, C, S, N may appear in list. | Files identified by P, C, S and N are rewound before and after UPDATE run. |
| R | No rewinds will be issued on files identified by P, C, S, and N. | Files identified by P, C, S and N are rewound before and after UPDATE run. |

UPDATE ACTION PARAMETERS (continued)

| Letter | Specified Action | Default Action |
|---|---|---|
| U | UPDATE continues regardless of normally fatal errors. | None. |
| W | New program library file will be in sequential format. | New library file will be in random format (if possible). |
| X | Compile file will contain 2-word header and compressed card image for each card written to the compile file. This parameter is for COMPASS 2.0 interface. | Compile file will contain uncompressed card images. |
| Z | UPDATE input file is assumed to be in compressed symbolic format. This option does not affect *READ files. | Input file is assumed to be in normal card image format. |
| 8 | Compile file output will contain 80-column card images. | Compile file output will contain 90-column images. |
| *=char | UPDATE master control character must precede a directive name. Any character having a display code value of 01 to 54 (octal) inclusive may be used, except for left and right parentheses (51 and 52 octal). | Master control character will be *. |
| /=char | Comment control character in column 2 listed with a correction set. Any character with display code value 01-54 may be used, except parentheses (51 and 52). | Comment control character will be /. |